

PDE | PRONATEC

PROGRAMA NACIONAL DE ACESSO AO ENSINO TÉCNICO E EMPREGO

UTRAMIG

Fundação de Educação para o Trabalho de Minas Gerais

CURSO TÉCNICO

INFORMÁTICA

Etapa 3

Curso Técnico

INFORMÁTICA

Etapa 3

PROFESSORES ORGANIZADORES DE CONTEÚDO

Cássio A. Brito
Marcelo Ferreira do Nascimento
Rômulo Leonardo Vieira Silva
Tereza Sanguinete de Souza
Thales Vinícius Dias
Thiago Vinícius F. Souza

Curso Técnico

INFORMÁTICA

Etapa 3

CONTEÚDOS ESPECÍFICOS	PÁGINA
WEB 3	3
Banco de Dados III - Oracle	21
Ambiente operacional III - Linux	105
Projeto de qualidade de sistemas de informação	211
Linguagem Técnica de Programação II	239
Tecnologia em Sistemas de Informação	295
Marketing em Sistemas de Informação	371

CURSO TÉCNICO EM

INFORMÁTICA

ETAPA 3

WEB

Sumário

1. LINGUAGEM PHP 7

2. SINTAXE BÁSICA 7

3. COMENTÁRIOS 7

4. ECHO 8

5. OPERADORES MATEMÁTICOS 8

6. OPERADORES DE COMPARAÇÃO 8

7. OPERADORES LÓGICOS 9

8. VARIÁVEIS 9
    8.1. TIPOS DE VARIÁVEIS 9

9. ESTRUTURA DE CONTROLE 10

10. ESTRUTURA DE REPETIÇÃO 11
    10.1 WHILE 11
    10.2 DO-WHILE 12
    10.3 FOR 12
    10.4 SWITCH/CASE 13

11. MATRIZ OU ARRAY 14

12. FUNÇÕES 14

13. PHP E MYSQL 15
    13.1 CONNECT 15
    13.2 INSERT 15
    13.3 SELECT 16
    13.4 DELETE 17
    13.5 UPDATE 18

14. EXERCÍCIOS 18

15. BIBLIOGRAFIA 20

WEB 3
PHP

Objetivos:

- Apresentar a linguagem PHP e seus recursos para a construção de sites e aplicações web.
- Permitir que os alunos desenvolvam páginas geradas dinamicamente e de forma rápida.

CURSO TÉCNICO EM INFORMÁTICA
DISCIPLINA: WEB III
ETAPA: 3ª
CH: 60 h

16. Critérios de distribuição de pontos

	Atividade	Valor
1º Bimestre	Trabalho Escrito	10
	Avaliação Mensal	10
	Avaliação Bimestral	20
	Total	**40**

	Atividade	Valor
2º Bimestre	Trabalho Prático	15
	PHP/MYSQL	15
	Projeto Mostra	30
	Total	**60**

1. Linguagem PHP

PHP é uma sigla recursiva que significa **PHP HyperText Preprocessor**. O PHP é uma linguagem de código-fonte aberto, muito utilizada na Internet e especialmente criada para o desenvolvimento de aplicativos **Web.**

A melhor coisa em usar PHP está no fato de ele ser extremamente simples para um iniciante, mas oferece muitos recursos para o programador profissional.

Para testar scripts PHP é necessário um servidor com suporte a esta tecnologia. Normalmente, o mais utilizado é o Apache. O banco de dados mais utilizado com os scripts PHP é o MySQL.

2. Sintaxe Básica

Tags especiais indicam ao PHP onde estão os blocos de código. A tag de abertura é formada por um sinal de “menor que” (<), um sinal de interrogação (?) e a sigla php. A tag de fechamento é formada por um ponto interrogação (?) e sinal de “maior que” (>).

Ex:

```
<?php
...
?>
```

3. Comentários

Os comentários de mais de uma linha no PHP são obtidos através de /* e */.
Os comentários de apenas uma linha são obtidos através de //.
Os comentários não aparecem no **browser.**

```
<?php
/*
        O código abaixo soma duas variáveis
e exibe o valor encontrado.
*/
?>
```

4. Echo

A função echo é a instrução que envia para a **saída** qualquer informação, podendo conter texto, números ou **variáveis.**

```
<?php
echo "Olá, tenho ";
echo 22;
echo " anos.";
?>
```

5. Operadores Matemáticos.

São usados para efetuarem operações sobre as variáveis e constantes. Os operadores do PHP são:

+ soma
- subtração
* multiplicação
/ divisão
^ exponenciação
% módulo, resto da divisão
++ acrescenta um a uma variável
-- subtrai um de uma variável
+= soma um valor a uma variável e lhe atribui o resultado **Bibliografia**

6. Operadores de comparação.

Uma comparação sempre gera um dos dois valores possíveis: vazio, que corresponde a falso, e 1, que corresponde a verdadeiro.

= = é igual a
! = não é igual a

> é maior que
< é menor que
>= é maior ou igual a
<= é menor ou igual a

7. Operadores lógicos.

and ou && - operador lógico “e”, apenas retornando verdadeiro quando as duas condições envolvidas no teste forem verdadeiras
or ou | | operador lógico “ou”, retornando verdadeiro quando uma ou as duas condições envolvidas no teste forem verdadeiras
! operador lógico “não”, invertendo o resultado de um teste
xor – operador lógico “ou exclusivo” que determina se uma de duas condições é verdadeira mas não ambas. Se ambas forem verdadeiras, o teste final será falso

8. Variáveis

Variáveis armazenam valores. Pode-se referir a variáveis para obter seu valor ou para alterar seu conteúdo.

No PHP elas são representadas por um cifrão ($) mais o nome da variável. Os nomes de variáveis válidos são iniciados por letras ou por um subscrito (_). Existe diferenciação entre nomes de variáveis maiúsculas e minúsculas.
Ex: $a, $_A, $_a

8.1 Tipos de dados

O PHP suporta vários tipos de dados:

Inteiro – Números inteiros (isto é, números sem ponto decimal)

Números de dupla precisão – Números reais (isto é, números que contêm um ponto decimal)

String – Texto entre aspas simples (´ ´) ou duplas (“ “)

Booleanos – armazenam valores verdadeiros ou falsos, usados em testes de condições

Array – Grupo de elementos do mesmo tipo

Objeto – Grupo de atributos e métodos

Recurso – Uma origem de dados externa

Nulo – Nenhum valor

Exemplo:

```
<?php
/*
    O código abaixo soma duas variáveis
e exibe o valor encontrado
*/
$a = 10;
$b = 15;
$c = $a + $b;
echo "$a mais $b é igual a $c";
?>
```

9. Estruturas de Controle

No PHP, as estruturas de controle são formadas por declarações condicionais e de **looping:**

if – executa uma ação se uma condição for atendida. O bloco de comandos a ser executado deve ser escrito entre chaves;

else – pode-se colocar um conjunto de comandos alternativos caso o teste do if seja negativo. A declaração else deve vir logo após o bloco de código relacionado ao if . O comando if também pode ser usado após a declaração else .

Exemplo:

```
<?php
$cor = "branco";
if ($cor == "vermelho")
{
echo("A variável contém o valor 'vermelho'.");
}
else if ($cor == "azul")
{
echo("A variável contém o valor 'azul'.");
}
else if ($cor == "amarelo")
{
echo("A variável contém o valor 'amarelo'.");
}
else
{
echo("O valor da variável não foi identificado.");
}
?>
```

10. Estruturas de Repetição

10.1 While

Estrutura de looping que não necessita de um número determinado de iterações.

Ele é executado enquanto uma condição for verdadeira.

Exemplo:

```
<?php
```

```
$i = 1;
while ($i < 10000)
{
echo($i);
$i *= 2;
echo(" vezes 2 é igual a $i <br>");
}
?>
```

10.2 do-while

Outra forma de looping que executa um bloco de código, testa uma condição e repete novamente o bloco de código (ou não).

Exemplo:

```
<?php
$i = 1;
do
{
echo ("Linha $i <br>");
$i++;
}
while ($i < 10)
?>
```

10.3 For

Estrutura de looping que executa um bloco de código quantas vezes for indicado em uma variável. Deve-se definir a variável que será testada no looping, uma condição de teste e o incremento (ou decremento) da variável de controle.

Exemplo:

```
<?php
for ($i = 1; $i < 10; $i++)
{
echo("Linha $i <br>");
}
?>
```

10.4 Switch / case

Forma de testar uma dentre várias possibilidades. A declaração default executa caso nenhuma das opções for verdadeira. A declaração break faz com que o restante do código não seja executado caso o teste seja verdadeiro.

Exemplo:

```
<?php
$d = getdate();
switch ($d['wday'])
{
case 5:
echo("Finalmente Sexta");
break;
case 6:
echo("Super Sábado");
break;
case 0:
echo("Domingo Sonolento");
break;
default:
echo("Estou esperando pelo fim da semana");
}
?>
```

11. Matriz ou Array

As variáveis do tipo matriz ou array permitem o armazenamento de diversos elementos referenciados por uma mesma referência.

```
<?php
$frutas = array(
1 => "Laranja",
2 => "Maçã",
3 => "Uva");
echo "<li> $frutas[1]<br>";
echo "<li> $frutas[2]<br>";
echo "<li> $frutas[3]<br>";
?>
```

12. Funções

Uma função é um bloco de código reutilizável que é executado devido a um evento ou pela chamada de outra função. Deve-se usar a declaração function para criar uma função.

```
<?php
function escreveTexto()
{
echo("Já sei criar funções!");
}
escreve Texto();
?>
```

13. PHP E MYSQL

O MySQL é o gerenciador de banco de dados mais usado com o PHP. Existem muitas funções pré-definidas para manipulação de conexões com bancos de dados.

13.1 Connect

A função mysql_connect tenta uma conexão com um servidor MySQL. Deve-se passar como parâmetros: o nome do servidor (ou número IP) onde o MySQL está sendo executado, o nome de usuário e a senha deste usuário. O comando alternativo die trata um possível fracasso na conexão.

```
<?php
$host = "localhost";
$user = "root";
$senha = "";
$dbname = "bd";
//conecta ao banco de dados
mysql_connect($host, $user, $senha)
or die("Não foi possível conectar-se com o banco de dados");
//seleciona o banco de dados
mysql_select_db($dbname)or die("Não foi possível conectar-se com o banco de dados");
?>
```

13.2 Insert

Para se incluir dados em uma tabela MySQL, deve-se usar o comando INSERT.

```
<?php
//conexão com o db
include ("conectar.php");
//variáveis enviadas pelo método post
$nome = $_POST['nome'];
$senha = $_POST['senha'];
//insere os valores no db
$insere = mysql_query("INSERT INTO `login`( `nome`, `senha`) VALUES
('$nome','$senha')") or die(mysql_error());
if($insere)
echo "Cadastro realizado com sucesso!";
else
echo "Não foi possível fazer o seu cadastramento!";
?>
```

13.3 Select

A função mysql_query faz consultas à base previamente selecionada.

```
<?php
//conexão com o db
include ("conectar.php");
//consultar banco de dados
$consulta = mysql_query("select Nome, Senha from login") or die(mysql_error());
if($consulta)
echo "Consulta realizada com sucesso!";
else
echo "Não foi possível fazer a consulta!";
echo "<br />";
echo "<br />";
echo "<table border='1'>
```

```
<tr>
<td>Nome</td>
<td>Senha</td>
</tr>";
while($linha = mysql_fetch_array($consulta))
 {
 echo "<tr>";
 echo "<td>"  . $linha['Nome'] . "</td>";
 echo "<td>"  . $linha['Senha'] . "</td>";
   echo "</tr>";
 }
echo "</table>";
?>
```

13.4 Delete

O comando DELETE remove um registro de uma tabela.

```
<?php
//conexão com o db
include ("conectar.php");
//variáveis enviadas pelo método post
$nome = $_POST['nome'];
$senha = $_POST['senha'];
//deletar campos do banco de dados
$deletar = "DELETE FROM login WHERE Nome = '$nome' && Senha ='$senha'";
$resultado = mysql_query($deletar)
or die("Login ou senha não confere");
echo "Registro excluído com sucesso";
?>
```

13.5 Update

O comando UPDATE altera um registro de uma tabela.

```
<?php
//conexão com o db
include ("conectar.php");
//variáveis enviadas pelo método post
$nome = $_POST['nome'];
$senha = $_POST['senha'];
$nova senha=$_POST ['nova senha'];
//alterar campos do banco de dados
$consulta = "UPDATE login SET senha= '$nova senha'
WHERE nome = '$nome' && senha='$senha'";
$resultado = mysql_query($consulta)
or die("Falha na execução da consulta");
echo "Dados alterados com sucesso";
?>
```

14. Exercícios

1) Fazer os programas abaixo em php.

- Que imprima o seu nome no formulário.
- Calcule e imprima os 10 primeiros números pares.
- Receba um número e verifique se o mesmo é par ou impar.
- Que receba 3 números e mostre o maior e o menos deles.
- Que receba a nota final e verifique a situação do aluno. (Aprovado, Reprovado ou em Recuperação).

2) Crie e codifique as calculadoras abaixo:

Calculadora de Massa Corporal

O Índice de Massa Corporal (IMC) é uma medida do grau de obesidade de uma pessoa. Por meio do cálculo de IMC é possível saber se você está acima ou abaixo dos parâmetros ideais de peso para sua estatura. Coloque seus dados e compare o resultado com a tabela de referência.



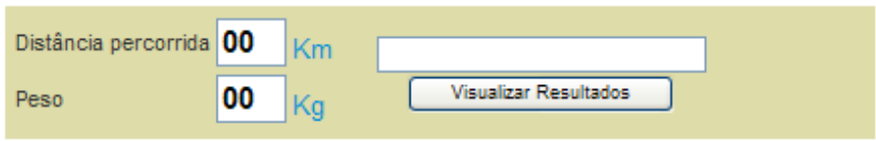
Compare seu resultado com os dados da tabela do IMC (Índice de Massa Corporal)

Tabela de referência (IMC)	
Abaixo de 18,5	Você está abaixo do peso ideal
Entre 18,5 e 24,9	Parabéns. Você está em seu peso normal
Entre 25,0 e 29,9	Você está acima de seu peso (sobrepeso)
Entre 30,0 e 34,9	Obesidade grau I
Entre 35,0 e 39,9	Obesidade grau II
Acima de 40,0	Obesidade grau III

IMC=massa/(altura^2)

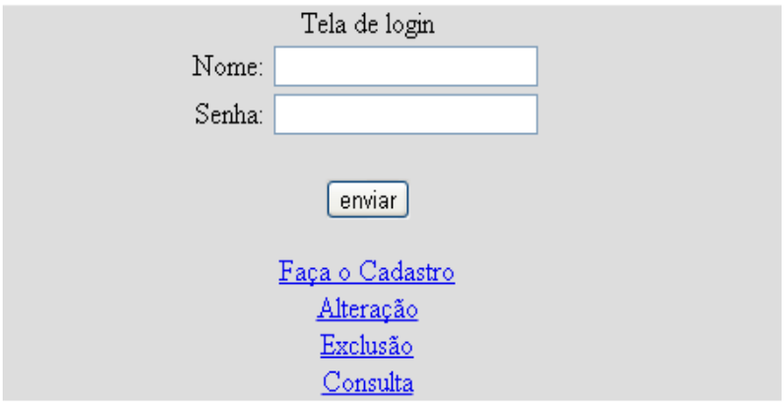
Calculadora de Calorias

A Calculadora de Calorias indica quantas calorias você perde quando corre uma determinada distância. Esse é um cálculo médio que pode variar em função da constituição física.



Cal=(distper*peso)*1.036

3) Criar o formulário abaixo e codificar as rotinas conexão, cadastro, alteração, exclusão e consulta em conexão com o banco de dados MYSQL.



15. Bibliografia

http://pt.wikibooks.org

Apostila de PHP - Prof. Cristiano Cachapuz e Lima

CURSO TÉCNICO EM

INFORMÁTICA

ETAPA 3

BANCO DE DADOS III
ORACLE

Sumário

OBJETIVOS GERAIS 23

AVALIAÇÕES 23

DISTRIBUIÇÃO DE PONTOS 24

1. INTRODUÇÃO 25

1.2.SOBRE O SERVIDOR ORACLE 25

1.3.ARMAZENANDO INFORMAÇÕES 26

1.4.CONCEITO DE BANCO DE DADOS RELACIONAL 26
- 1.4.1.COMPONENTES DO MODELO RELACIONAL 27
- 1.4.2.DEFINIÇÃO DE UM BANCO DE DADOS RELACIONAL 27
- 1.4.5.COMPONENTES DE MODELAGEM RELACIONAL 28
- 1.46.PROPRIEDADES DE UM BANCO DE DADOS RELACIONAL 28

2. SQL (STRUCTURED QUERY LANGUAGE) 29
- 2.1.INSTRUÇÕES SQL 29
- 2.2.TIPOS DE DADOS ORACLE 31

RECURSOS DE INSTRUÇÕES SQL SELECT 31

BIBLIOGRAFIA 103

Objetivos Gerais

Esse material tem por objetivo revisar os conhecimentos adquiridos nas etapas passadas e aprofundar a linguagem SQL, mas nesse modulo utilizando o SGBD Oracle.

Avaliações

- Apresentação da Disciplina e apresentação da turma
- Definição de dado, informação, banco de dados e modelagem de dados; o profissional de banco de dados; sistemas gerenciadores de banco de dados.
- Componentes do banco de dados – definição e exemplos: entidade, atributos, ocorrências, chave primária, chave secundária e chave candidata;
- Definição de relacionamento – tipos, definição, resolução e exemplos (1-1 1-N e M-N).
- Diagrama Entidade Relacionamento – DER: definição e modelos
- Exercício avaliativo – exercícios de fixação – correção de exercícios
- Exercícios diversos de fixação para elaboração de DER com situações do mundo real – correção destes exercícios em sala de aula
- Apresentação do MYSQL – interface
- Criação de tabelas – definição tipos de campo, chaves primárias, chave estrangeira;
- Relacionamentos.
- Apresentando a Linguagem MYSQL
- Commandos: SELECT, *, FROM, ORDER BY ASC E DESC
- MAX, MIN, AVG, COUNT E SUM.
- Apresentação de projetos
- Avaliação do momento

Distribuição de pontos

1º Bimestre

Critérios de distribuição de pontos		
Data	Atividades	**Valor**
Março	Lista de Exercícios	**20**
Março	Laboratório	**20**
Abril	Avaliação bimestral	**20**
Total de Pontos		**40**

Tabela 1 – Distribuição de pontos 1

2º Bimestre

Critérios de distribuição de pontos		
Data	Atividades	Valor
Junho	Lista de Exercícios	10
Junho	Laboratório	10
Julho	Trabalho Final	10
Julho	Avaliação bimestral	30
Total de Pontos		**60**

Tabela 1 – Distribuição de Pontos 2

1. Introdução

1.1. Oracle 11g

O Oracle 11g foi desenvolvido visando à computação em grid (**grid computing** - origem do "g" do nome Oracle 11g). A computação em grid prevê que o processamento e o armazenamento dos dados sejam alocados e realocados rapidamente, utilizando a rede corporativa.

A ideia da computação em grid é o compartilhamento de diversos recursos de vários usuários da rede. Corresponde ao cenário em que todos os sistemas operacionais sejam compartilhados, de modo a criar um grupo de recursos dinâmicos, em analogia à rede de energia elétrica ou à rede telefônica. Como cliente da grade, você não se importa com o local onde os dados são mantidos ou onde é realizada a computação. Você deseja que a computação seja concluída e que as informações sejam fornecidas quando especificado.

1.2. Sobre o Servidor Oracle

O servidor Oracle suporta modelos relacionais e modelos relacionais de objeto. O servidor estende os recursos de modelagem de dados para suportar um modelo de banco de dados relacional de objeto que inclui programação orientada a objeto, tipos de dados complexos, objetos de negócios complexos e compatibilidade integral com o mundo relacional.

Ele contém vários recursos para proporcionar melhor desempenho e funcionalidade de aplicações OLTP (**On-Line Transaction Processing**), como o compartilhamento mais eficiente de estruturas de dados durante o **runtime,** caches de buffer maiores e **constraints** adiáveis. As aplicações de **data warehouse** são beneficiadas por melhorias como a execução paralela de operações de inserção, atualização e deleção; o particionamento; e a otimização de consultas em paralelo. Operando na estrutura da NCA (**Network Computing Architecture**), o modelo Oracle suporta aplicações cliente/servidor e aplicações baseadas na Web distribuídas e com várias camadas.

1.3. Armazenando Informações

Toda organização tem necessidades de informações. Uma biblioteca mantém uma lista de membros, livros, datas de vencimento e multas. Uma empresa precisa guardar informações sobre funcionários, departamentos e salários. Essas informações são chamadas de dados.

As organizações podem armazenar dados em várias mídias e em formatos diferentes; por exemplo, um documento impresso em um armário de arquivo ou dados armazenados em planilhas eletrônicas ou em bancos de dados.

Um banco de dados é um conjunto organizado de informações. Para gerenciar bancos de dados, é necessário um DBMS (**Database Management System**).

Um DBMS é um programa que armazena, recupera e modifica dados de bancos de dados sob demanda. Existem quatro tipos principais de bancos de dados: hierárquico, de rede, relacional e (recentemente) relacional de objeto.

1.4. Conceito de banco de dados relacional

Os princípios do modelo relacional foram descritos primeiramente pelo Dr. E. F. Codd em um trabalho de junho de 1970 intitulado **"A Relational Model of Data for Large Shared Data Banks"**. Nesse trabalho, o Dr. Codd propunha o modelo relacional para sistemas de banco de dados.

Os modelos comuns usados na época eram o hierárquico e o de rede, ou até mesmo estruturas de dados simples de flat files. Logo depois, o RDBMS **(Relational Database Management System**) tornou-se muito popular, especialmente pela facilidade de uso e flexibilidade em termos de estrutura. Além disso, vários fornecedores inovadores, como a Oracle, complementaram o RDBMS com um conjunto de produtos eficientes para usuários e desenvolvedores de aplicações, que compunham uma solução integral.

1.4.1. Componentes do Modelo Relacional

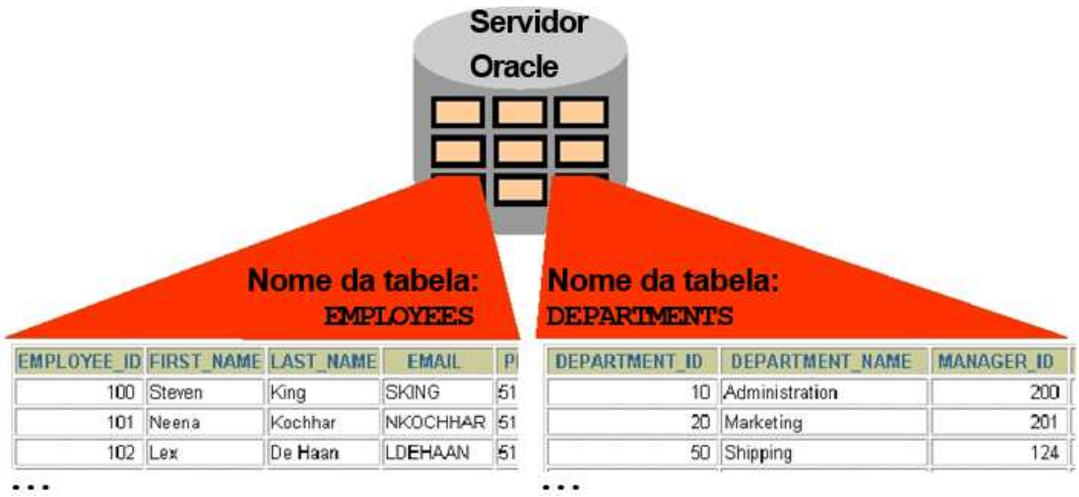
• Conjuntos de objetos ou relações que armazenam os dados

• Um conjunto de operadores que age sobre as relações para produzir outras relações

• Integridade de dados para precisão e consistência

1.4.2. Definição de um Banco de Dados Relacional

Um banco de dados relacional usa relações ou tabelas de duas dimensões para armazenar informações.



EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	P
100	Steven	King	SKING	51
101	Neena	Kochhar	NKOCHHAR	51
102	Lex	De Haan	LDEHAAN	51

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID
10	Administration	200
20	Marketing	201
50	Shipping	124

De forma simplificada, pode-se dizer que o banco de dados relacional tem por objetivo implementar os conceitos do modelo relacional com todas as suas características básicas, ou seja, como entidades, atributos e relacionamentos.

Por exemplo, é possível armazenar informações sobre todos os funcionários de uma empresa. Em um banco de dados relacional, você cria diversas tabelas para armazenar diferentes informações sobre os funcionários, como uma tabela de funcionários, uma tabela de departamentos e uma tabela de salários.

Para exemplificar, pode ser citada a tabela Cliente - também conhecida como entidade Cliente:

CPF	Nome	RG	Rua
372818	Carlos Silva	829092220-1	Rua Silva, 234

Onde, as colunas CPF, Nome, RG e Rua são os atributos. A única linha presenta na tabela, também é conhecida como tupla ou registro da tabela.

1.4.5. Componentes de modelagem relacional

- **Entidade:** Algo importante sobre o qual são necessárias informações. Como exemplos, podemos citar departamentos, funcionários e pedidos.
- **Atributo:** Algo que descreve ou qualifica uma entidade. Por exemplo, para a entidade de funcionário, os atributos serão o número, o nome, o cargo, a data de admissão, o número do departamento e outros dados sobre o funcionário. Cada um desses atributos é obrigatório ou opcional. Esse estado é denominado opcionalidade.
- **Relacionamento:** Uma associação nomeada entre entidades que exibe a opcionalidade e o grau. Como exemplos, podemos citar as associações entre funcionários e departamentos, e entre pedidos e itens.

1.46. Propriedades de um Banco de Dados Relacional

Em um banco de dados relacional, você não especifica a rota de acesso às tabelas e não precisa saber como os dados estão organizados fisicamente.

Para acessar o banco de dados, execute uma instrução SQL (**Structured Query Language),** que é a linguagem padrão ANSI (**American National Standards Institute**) de operação de bancos de dados relacionais. Essa linguagem contém um amplo conjunto de operadores para particionar e combinar relações. É possível usar instruções SQL para modificar o banco de dados.

2. SQL (Structured Query Language)

O uso de SQL permite a comunicação com o servidor Oracle. A linguagem SQL apresenta estas vantagens:

- Eficiente
- Fácil de aprender e usar
- Funcionalmente completa (permite definir, recuperar e manipular os dados das tabelas).



2.1. Instruções SQL

O Oracle SQL adota os padrões aceitos pelo setor. Para assegurar a compatibilidade futura com os padrões em desenvolvimento, a Oracle Corporation mantém ativamente uma equipe especializada nos comitês de padrões SQL. Os comitês aceitos pelo setor são o ANSI **(American National Standards Institute)** e o ISO (**International Standards Organization**). Os dois comitês aceitam SQL como a linguagem padrão para os bancos de dados relacionais.

Instrução	Descrição
SELECT INSERT UPDATE DELETE MERGE	Recupera dados do banco de dados, informa novas linhas, altera linhas existentes e remove linhas indesejadas das tabelas do banco de dados, respectivamente. Conhecida coletivamente como **DML (Data Manipulation Language)**.
CREATE ALTER DROP RENAME TRUNCATE COMMENT	Configura, altera e remove estruturas de dados das tabelas. Conhecida coletivamente como **DDL (Data Definition Language)**.
COMMIT ROLLBACK SAVEPOINT	Gerencia as alterações feitas por instruções **DML**. As alterações nos dados podem ser agrupadas em transações lógicas.
GRANT REVOKE	Concede ou revoga direitos de acesso ao banco de dados Oracle e às estruturas contidas nele. Conhecida coletivamente como **DCL (Data Control Language).**

2.2. Tipos de Dados Oracle

Tipos de Dados Caractere

- **CHARACTER(X) | CHAR(X):** Representa uma string de tamanho X. Se X for omitido, então é o equivalente a CHAR(1). Se a string armazenada for menor que X, o restante é preenchido com branco.
- **VARCHAR2(N):** Armazena uma string de tamanho variável. É possível armazenar uma string de até 4000 bytes. Subtipo: STRING v
- **CHARACTER VARYING(X) | VARCHAR(X):** É sinônimo de VARCHAR2 que representa uma string de tamanho X. Armazena exatamente o tamanho da string sem preencher o resto com branco. O X é obrigatório.
- **CHARACTER LARGE OBJECT | CLOB:** Armazena strings longas. Normalmente usado para armazenar documentos.

Tipo de Dados Bit String

- **BIT(X):** Permite armazenar uma quantidade X de bits.
- **BIT VARING(X) | VARBIT(X):** Armazena uma quantidade variável de bits até o tamanho X.

Tipo de Dados Binary Strings (BLOB)

Os Large Object (LOB) datatypes são usados para armazenar dados não estruturados como imagens, arquivos binários. Os tipos LOBs podem armazenar até 4GB de informação:

- **BINARY LARGE OBJECT | BLOB:** Para armazenar grande quantidade de bytes, como: arquivos de imagem, vídeo, áudio, etc.
- **CHARACTER LARGE OBJECT | CLOB:** Armazena até 4GB de dados de caractere.
- **BINARY FILE | BFILE:** Armazena até 4GB de dados em arquivos externos. Uma coluna BFILE armazena um ponteiro para um arquivo armazenado no SO.

Tipos de Dado Numérico

- **INTEGER | INT E SMALLINT:** Para representar inteiros.
- **NUMERIC (P, S):** Tem uma precisão (P) e a escala (S) – número de dígitos na parte fracionário. A escala não pode ser maior que a precisão. Muito utilizado para representar dinheiro.
- **DECIMAL (P, S):** Funciona igual ao NUMERIC.
- **REAL:** Ponto flutuante de precisão simples.
- **DOUBLE:** Ponte flutuante de precisão dupla.
- **FLOAT (P):** Permite especificar a precisão que se quer.

Recursos de Instruções SQL SELECT

Uma instrução SELECT recupera informações do banco de dados. Com essa instrução, é possível usar os seguintes recursos:

- **Projeção:** Escolha as colunas de uma tabela a serem retornadas por uma consulta.
- Escolha quantas colunas forem necessárias
- **Seleção**: Escolha as linhas de uma tabela a serem retornadas por uma consulta. É possível usar vários critérios para restringir as linhas recuperadas.
- **Join:** Una os dados armazenados em diferentes tabelas especificando o vínculo entre elas.

Instrução SELECT Básica

Em sua forma mais simples, uma instrução SELECT deve incluir:

- Uma cláusula SELECT, que especifica as colunas a serem exibidas.
- Uma cláusula FROM, que identifica a tabela com as colunas listadas na cláusula SELECT.

Sintaxe:

SELECT ***** **DISTINCT** **column|expression** **alias** **FROM table;**	é uma lista de uma ou mais colunas seleciona todas as colunas suprime as colunas duplicadas seleciona a coluna nomeada ou a expressão fornece cabeçalhos distintos às colunas selecionadas especifica a tabela que contém as colunas

Observação: Os termos palavra-chave, cláusula e instrução são usados da seguinte maneira:

- Uma palavra-chave é um elemento individual de SQL.

Por exemplo, SELECT e FROM são palavras-chave.

- Uma cláusula é uma parte de uma instrução SQL.

Por exemplo, SELECT employee_id,... É uma cláusula.

- Uma instrução é uma combinação de duas ou mais cláusulas.

Por exemplo, SELECT * FROM employees é uma instrução SQL.

Selecionar todas as colunas

SELECT * FROM departamento;

Selecionar colunas especificas

SELECT name, departamento_id FROM departamento;

DISTINCT

Para eliminar as linhas com registro repetido, utilize a clausula DISTINCT, conforme o seguinte código:

SELECT DISTINCT UF FROM CIDADE;

A clausula DISTINCT também pode ser utilizada em mais de uma coluna:

SELECT DISTINCT SALARIO, DEPARTAMENTO FROM FUNCIONARIO;

Operadores Aritméticos entre seleções

Durante a pesquisa os operadores aritméticos (++, -,*, /) podem ser utilizados, como demostra o código:

SELECT NOME, SALARIO, SALARIO * 12 FROM FUNCIONARIO;

Os operadores lógicos permitidos são os seguintes:

= igual

< menor que

<= menor ou igual

> maior

>= maior ou igual

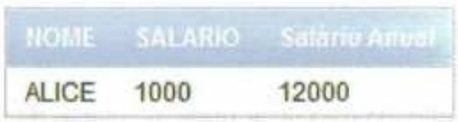
!= diferente

<> diferente

Clausula AS

A clausula AS nome, permite definir um apelido para qualquer coluna ou tabela durante a seleção:

SELECT NOME, SALARIO, SALARIO * 12 AS "Salário Anual" FROM FUNCIONARIO;



NOME	SALARIO	Salário Anual
ALICE	1000	12000

Note que a coluna aparece com um nome bem mais compreensível “Salário Anual”.

Clausula WHERE

Podem ser criados critérios para seleção das linhas, com a clausula WHERE:

SELECT CODIGO_CLIENTE, NOME, ENDERECO, TELEFONE, EMAIL, RENDA.

FROM CLIENTE

WHERE RENDA = 1000

ORDER BY NOME;

Condições Lógicas

Uma condição lógica combina o resultado de duas condições componentes para produzir um único resultado com base nessas condições, ou inverte o resultado de uma única condição. Só será retornada uma linha se o resultado geral da condição for verdadeiro.

Três operadores lógicos estão disponíveis em SQL:

- AND
- OR
- NOT

SELECT CODIGO_CLIENTE, NOME, ENDERECO, TELEFONE, EMAIL, RENDA.

FROM CLIENTE

```
WHERE (RENDA = 1000 AND CODIGO_CLIENTE > 1)
ORDER BY NOME;
```

Comparadores BETWEEN, AND, IN, LIKE e IS NULL

Também podem ser utilizados os comparadores **BETWEEN.. AND.., IN (lista), LIKE e IS NULL:**

```
SELECT CODIGO_CLIENTE, NOME, ENDERECO, TELEFONE, EMAIL, RENDA.
FROM CLIENTE
WHERE (RENDA BETWEEN 1000 AND 2000)
ORDER BY NOME;
```

A clausula **IN** compara o valor da coluna com o conjunto de valores informados:

```
SELECT CODIGO_CLIENTE, NOME, ENDERECO, TELEFONE, EMAIL, RENDA.
FROM CLIENTE IN (1, 2, 3, 4)
ORDER BY NOME;
```

A clausula **LIKE** é utilizada exclusivamente em colunas com sequência de caractere, como a coluna nome:

```
SELECT CODIGO_CLIENTE, NOME, ENDERECO, TELEFONE, EMAIL, RENDA.
FROM CLIENTE
WHERE NOME LIKE 'M%'
ORDER BY NOME;
```

O símbolo "%" representa nenhuma ou qualquer sequência de caracteres, já o símbolo "_" representa um único caractere.

Então, no SELECT acima, serão selecionados todos os clientes cujos nomes iniciam com a letra M.

Os símbolos podem ser utilizados em qualquer posição da string:

```
SELECT CODIGO_CLIENTE, NOME, ENDERECO, TELEFONE, EMAIL, RENDA.
FROM CLIENTE
WHERE NOME LIKE '%A%'
ORDER BY NOME;
```

No SELECT acima, será selecionado qualquer cliente que tenha a letra A no nome.

No exemplo a seguir, serão selecionados os nomes de todos os clientes que tenha a letra A, obrigatoriamente, na segunda posição do nome:

```
SELECT CODIGO_CLIENTE, NOME, ENDERECO, TELEFONE, EMAIL, RENDA.
FROM CLIENTE
WHERE NOME LIKE '_A%'
ORDER BY NOME;
```

Já a clausula **IS NULL** serve para selecionar as colunas nulas, pois o comando "=NULL" não funciona:

```
SELECT CODIGO_FUNCIONARIO, NOME, CPF.
FROM FUNCIONARIO
WHERE CPF IS NULL
ORDER BY NOME;
```

Clausula Rownum

Utilizada para limitar a quantidade de registros retornados.

Sintaxe:

```
SELECT * FROM table WHERE ROWNNUM;
```

Exemplo:

```
SELECT * FROM employees WHERE ROWNUM < 10;
```

Exercícios

1. Selecione toda a informação da tabela DEP. O resultado deve ser similar ao que se segue:

```
NDEP       NOME           LOCAL
---------- -------------- --------------
        10 Contabilidade  Condeixa
        20 Investigação   Mealhada
        30 Vendas         Coimbra
        40 Planeamento    Montemor

4 rows selected.
```

Resposta:

2. Mostre a lista de todos os empregados contendo o nome de cada empregado, a sua função, o salário e o número do departamento a que pertence.

```
NOME                 FUNCAO       SAL        NDEP
-------------------- ------------ ---------- ----------
Jorge Sampaio        Presidente       890000         10
Augusto Reis         Encarregado      450975         20
Duarte Guedes        Encarregado      380850         30
Silvia Teles         Encarregado      279450         10
Maria Dias           Analista         565000         20
Catarina Silva       Analista         435000         20
Joana Mendes         Vendedor         145600         30
Nelson Neves         Vendedor         212250         30
Ana Rodrigues        Vendedor         221250         30
Manuel Madeira       Vendedor         157800         30
Tome Ribeiro         Continuo          56950         30
Rita Pereira         Continuo          65100         20
Olga Costa           Continuo          68300         10
Antonio Silva        Continuo          70800         20

14 rows selected.
```

Resposta:

3. Apresente a lista de todos os empregados (nome, número de departamento e salário) cujo salário está entre 150000 e 300000.

```
NOME                 NDEP       SAL
-------------------- ---------- ----------
Silvia Teles                 10     279450
Nelson Neves                 30     212250
Ana Rodrigues                30     221250
Manuel Madeira               30     157800
4 rows selected.
```

Resposta:

4. Apresente a lista de todos os departamentos ordenados decrescentemente por número de departamento.

```
NDEP       NOME           LOCAL
---------- -------------- --------------
        40 Planeamento    Montemor
        30 Vendas         Coimbra
        20 Investigação   Mealhada
        10 Contabilidade  Condeixa

4 rows selected.
```

Resposta:

5. Exiba a lista de todas as funções existentes na empresa. Devem ser excluídas as repetições.

```
FUNCAO
------------
Analista
Continuo
Encarregado
Presidente
Vendedor

5 rows selected.
```

Resposta:

6. Mostre a informação detalhada (toda a informação disponível) dos empregados dos que pertencem ao departamento 10 ou 30.

```
NEMP    NOME            FUNCAO      ENCAR DATA_ENTR SAL     PREMIOS    NDEP
------- --------------- ----------- ----- --------- ------- ------- -------
   1839 Jorge Sampaio   Presidente        11-FEB-84  890000              10
   1698 Duarte Guedes   Encarregado  1839 25-NOV-91  380850              30
   1782 Silvia Teles    Encarregado  1839 03-NOV-86  279450              10
   1499 Joana Mendes    Vendedor     1698 04-OCT-84  145600   56300      30
   1521 Nelson Neves    Vendedor     1698 27-FEB-83  212250   98500      30
   1654 Ana Rodrigues   Vendedor     1698 17-DEC-90  221250   81400      30
   1844 Manuel Madeira  Vendedor     1698 21-APR-85  157800       0      30
   1900 Tome Ribeiro    Continuo     1698 05-MAR-94   56950              30
   1934 Olga Costa      Continuo     1782 22-JUN-86   68300              10

9 rows selected.
```

Resposta:

7. Apresente à lista de funcionários (nome e função) cujos nomes aparecem as letras ‘v’ ou ‘u’.

```
NOME                 FUNCAO
-------------------- ------------
Augusto Reis         Encarregado
Duarte Guedes        Encarregado
Silvia Teles         Encarregado
Catarina Silva       Analista
Nelson Neves         Vendedor
Ana Rodrigues        Vendedor
Manuel Madeira       Vendedor
Antonio Silva        Continuo

8 rows selected.
```

Resposta:

8. Mostre agora a lista de funcionários (nome e função) cujos nomes começam pela letra ‘A’ e contêm a sequência de letras ‘us’ e também os que começando por ‘R’ contêm a sequência de letras ‘ei’. Tenha em atenção às diferenças de maiúsculas e minúsculas.

```
NOME                 FUNCAO
-------------------- ------------
Augusto Reis         Encarregado
Rita Pereira         Continuo

2 rows selected.
```

Resposta:

9. Mostre a lista de vendedores cujos prémios foram menores do que 10% da remuneração anual (sal * 14 + prêmios). O resultado deve incluir o nome do vendedor, 10% da sua remuneração anual e ainda os prémios, e deve ser ordenado crescentemente pelos 10% de remuneração anual. No caso de haver vários vendedores com a mesma remuneração anual, estes devem surgir ordenados pelo nome do vendedor.

```
NOME                 10% Sal. Anual   PREMIOS
-------------------- -------------- ---------
Joana Mendes                209470     56300
Manuel Madeira              220920         0
Nelson Neves                307000     98500
Ana Rodrigues               317890     81400
4 linhas seleccionadas.
```

Resposta:

Junções

Para além das restrições (subconjunto de linhas) e projeções (subconjuntos de colunas), a operação mais comum é a junção. Uma junção corresponde a selecionar dados de mais de uma tabela ao mesmo tempo. Nas junções existe uma relação entre os dados de uma tabela e os dados da outra.

Se se fizer a seleção de dados de várias tabelas sem incluir na cláusula WHERE as condições de relação entre as várias tabelas obtém um produto de tabelas. De fato, este é um dos erros mais frequentes nos iniciados em junções. Note-se que ao obter-se o produto está a obterem-se todas as combinações possíveis de dados de uma tabela com os dados de outra(s) tabela(s). Uma técnica simples é a de garantir que numa junção existam sempre **n-1 restrições** com **n** a representar o número de tabelas incluídas na junção (Esta regra só é verdade para tabelas sem chaves concatenadas).

Existem dois tipos de junções: as equi-junções e as não equi-junções.

Equi-Junção

Existe uma equi-junção ou inner join (**equi join**) quando a relação entre a(s) coluna(s) das duas tabelas é a de igualdade. Exemplo:

```
SELECT nemp, sal, local
   FROM emp, dep
   WHERE emp.ndep = dep.ndep;
```

Neste caso estão a juntarem-se as linhas da tabela **emp** com todas as linhas da tabela **dep** que verifiquem a condição de o valor de **ndep** da tabela **emp** ser igual ao valor de **ndep** da tabela **dep**. O resultado seria:

```
NEMP       SAL        LOCAL
---------- ---------- --------------
      1839     890000 Condeixa
      1566     450975 Mealhada
      1698     380850 Coimbra
      1782     279450 Condeixa
      1788     565000 Mealhada
      1902     435000 Mealhada
      1499     145600 Coimbra
      1521     212250 Coimbra
      1654     221250 Coimbra
      1844     157800 Coimbra
      1900      56950 Coimbra
      1876      65100 Mealhada
      1934      68300 Condeixa
      1369      70800 Mealhada

14 rows selected.
```

NATURAL JOIN

O join é baseado em todas as colunas com os mesmos nomes em todas as tabelas, envolvidas no SELECT. Ex:

```
 -- as 2 tabelas abaixo possuem a coluna em comum ID_DEPARTAMENTO

SELECT ID_EMP,
 NOME_EMP,
 NOME_DEPARTAMENTO
 FROM CERTIFICACAO. TAB_EMPREGADOS NATURAL JOIN
 CERTIFICACAO. TAB_DEPARTAMENTOS

 -- colunas em comum não podem ter qualificadores.
 -- o exemplo abaixo gera o erro "ora-25155: column used in
 -- NATURAL join cannot have qualifier"

 SELECT E.ID_EMP,
 E.NOME_EMP,
 E. ID_DEPARTAMENTO,
 D. NOME_DEPARTAMENTO
 FROM CERTIFICACAO. TAB_EMPREGADOS E NATURAL JOIN
 CERTIFICACAO. TAB_DEPARTAMENTOS D

 -- para colunas comuns em todas as tabelas, basta omitir.
 -- o qualificador

 SELECT E. ID_EMP,
 E. NOME_EMP,
 ID_DEPARTAMENTO, --coluna comum.
 D. NOME_DEPARTAMENTO
 FROM CERTIFICACAO. TAB_EMPREGADOS E NATURAL JOIN
 CERTIFICACAO. TAB_DEPARTAMENTOS D
```

JOIN … USING

Especifica através da clausula USING, colunas com nomes comuns nas tabelas a

serem usadas nos JOINS. Ex:

```
-- na clausula using deve ser especificada a coluna comum
-- às tabelas

SELECT E.ID_EMP,
E.NOME_EMP,
D.NOME_DEPARTAMENTO
FROM CERTIFICACAO. TAB_EMPREGADOS E JOIN
CERTIFICACAO. TAB_DEPARTAMENTOS D
USING (ID_DEPARTAMENTO)

-- colunas em comum não podem ter qualificadores.
-- o exemplo abaixo gera o erro "ora-25154: column used in
-- USING join cannot have qualifier"

SELECT E.ID_EMP,
E.NOME_EMP,
E. ID_DEPARTAMENTO,
D. NOME_DEPARTAMENTO
FROM CERTIFICACAO. TAB_EMPREGADOS E JOIN
CERTIFICACAO. TAB_DEPARTAMENTOS D
USING (ID_DEPARTAMENTO)

-- para colunas comuns em todas as tabelas, basta omitir o.
-- qualificador

SELECT E. ID_EMP,
E. NOME_EMP,
ID_DEPARTAMENTO,
D. NOME_DEPARTAMENTO
FROM CERTIFICACAO. TAB_EMPREGADOS E JOIN
CERTIFICACAO. TAB_DEPARTAMENTOS D
USING (ID_DEPARTAMENTO)
```

JOIN… ON

Define especificamente as condições e colunas que farão o JOIN. Ex:

```
-- define que o join será feito pelas colunas id_departamento
-- de ambas as tabelas, ou seja, só retornara resultados caso as.
-- duas tabelas possuam departamentos iguais

SELECT E. ID_EMP,
E. NOME_EMP,
E. ID_DEPARTAMENTO,
D.NOME_DEPARTAMENTO
FROM CERTIFICACAO. TAB_EMPREGADOS E JOIN
CERTIFICACAO. TAB_DEPARTAMENTOS D
ON (E. ID_DEPARTAMENTO = D. ID_DEPARTAMENTO).

-- é possível especificar mais de uma coluna

SELECT E.ID_EMP,
E.NOME_EMP,
E. ID_DEPARTAMENTO,
D. NOME_DEPARTAMENTO
FROM CERTIFICACAO. TAB_EMPREGADOS E JOIN
CERTIFICACAO. TAB_DEPARTAMENTOS D
ON (E.). ID_DEPARTAMENTO = D. ID_DEPARTAMENTO
AND E.ID_EMP = D.ID_GERENTE)
```

CARTESIAN JOINS (CROSS JOIN)

Ocorrem quando os dados são selecionados de 2 ou mais tabelas, mas nenhuma condição é especificada. Cada linha da 1º tabela é unida com cada linha da 2º tabela. Se a tabela A possui 3 linhas e a tabela B possui 4 linhas, o select resultante retorna 12 (3 x 4) linhas. Ex:

```
-- une todas as linhas de TAB_EMPREGADOS com cada linha de
-- TAB_DEPARTAMENTOS
-- evite fazer isso. Além de não retornar informações úteis,
```

-- dependendo do tamanho das tabelas, isso pode ocupar toda a área -- de memória disponível para isso.

```
SELECT *
FROM CERTIFICACAO. TAB_EMPREGADOS E,
CERTIFICACAO. TAB_DEPARTAMENTOS D
```

OUTER JOINS

Retorna linhas que são encontradas em todas as tabelas, bem como linhas que não foram encontradas em alguma das tabelas envolvidas no SELECT.

LEFT OUTER JOIN

Citando o documento da própria Oracle para explicar esse conceito:

"A LEFT OUTER JOIN is one of the JOIN operations that allow you to specify a join clause. It preserves the unmatched rows from the first (left) table, joining them with a NULL row in the shape of the second (right) table."

A dica que eu passo a vocês, para facilitar é:
LEFT OUTER JOIN deve ser usado quando eu precisar de TUDO que estiver apenas na tabela a esquerda, não importando se existe registros equivalentes na tabela da direita.

```
-- retorna os registros que possuem o id_departamento comum
-- em ambas as tabelas,
-- bem como os departamentos que não possuem empregados associados
-- retorna dados da tabela à esquerda na clausula from, mesmo.
-- que esses dados não possuam correspondentes (NULL) na tabela
-- da direita

SELECT E. ID_EMP,
E. NOME_EMP,
E. ID_DEPARTAMENTO,
D. NOME_DEPARTAMENTO
FROM CERTIFICACAO. TAB_DEPARTAMENTOS D LEFT OUTER JOIN
CERTIFICACAO. TAB_EMPREGADOS E
ON (E. ID_DEPARTAMENTO = D. ID_DEPARTAMENTO).
```

Observação: A cláusula OUTER é opcional

RIGHT OUTER JOIN

Citando o documento da própria Oracle para explicar esse conceito:

"A RIGHT OUTER JOIN is one of the JOIN operations that allow you to specify a JOIN clause. It preserves the unmatched rows from the second (right) table, joining them with a NULL in the shape of the first (left) table. A LEFT OUTER JOIN B is equivalent to B RIGHT OUTER JOIN A, with the columns in a different order."

A dica que eu passo a vocês, para facilitar é: RIGHT OUTER JOIN deve ser usado quando eu precisar de TUDO que estiver apenas na tabela a direita, não importando se exista registros equivalentes na tabela da esquerda.

```
-- retorna os registros que possuem o id_departamento comum
-- em ambas as tabelas, bem como os empregados que não possuem.
-- departamentos associados.
-- retorna dados da tabela à direita na clausula from, mesmo.
-- que esses dados não possuam correspondentes na tabela esquerda

SELECT E.ID_EMP,
E.NOME_EMP,
E. ID_DEPARTAMENTO,
D. NOME_DEPARTAMENTO
FROM CERTIFICACAO. TAB_DEPARTAMENTOS D RIGHT OUTER JOIN
CERTIFICACAO.TAB_EMPREGADOS E
ON (E. ID_DEPARTAMENTO = D. ID_DEPARTAMENTO).
```

Observação: A cláusula OUTER é opcional

SELF-JOIN

Realiza um JOIN entre uma tabela e "ela mesma". O nome da tabela aparece na clausula FROM, duas vezes, com alias diferentes. Ex:

```
-- mostra os gerentes de cada empregado. Cada gerente também é
-- um empregado.
```

```
SELECT TA. NOME_EMP || ' trabalha para ' || TB.NOME_EMP
FROM CERTIFICACAO. TAB_EMPREGADOS TA JOIN
CERTIFICACAO. TAB_EMPREGADOS TB
ON (TA. ID_GERENTE = TB. ID_EMP).
```

Não Equi-Junção

Existe uma não equi-junção ou non join, quando é usado um comparador que não seja o de igualdade. **Exemplo:**

```
SELECT nome, sal, prêmios, escalão.
FROM emp, descontos.
WHERE sal BETWEEN salinf AND salsup;
```

Neste caso estão a juntarem-se as linhas da tabela empcom todas as linhas da tabela descontos que verifiquem a condição de o valor de salda tabela empestar contido no intervalo de salinf a salsup da tabela dep. O resultado seria:

```
NOME                 SAL        PREMIOS    ESCALAO
-------------------- ---------- ---------- ----------
Tome Ribeiro              56950                     1
Rita Pereira              65100                     1
Olga Costa                68300                     1
Antonio Silva             70800                     1
Joana Mendes             145600      56300          2
Manuel Madeira           157800          0          2
Silvia Teles             279450                     3
Nelson Neves             212250      98500          3
Ana Rodrigues            221250      81400          3
Augusto Reis             450975                     4
Duarte Guedes            380850                     4
Catarina Silva           435000                     4
Jorge Sampaio            890000                     5
Maria Dias               565000                     5

14 rows selected.
```

Funções

As **funtions** ou funções tornam o bloco básico da consulta mais eficiente e são usadas para manipular valores de dados. Esta é a primeira das duas lições que descrevem o uso de funções. Ela concentra-se em funções de caractere, número e data de uma única linha, bem como em funções que convertem os dados de um tipo em outro (por exemplo, dados de caractere em dados numéricos).

Funções SQL

As funções constituem um recurso eficiente de SQL. É possível usá-las com os seguintes objetivos:

- Executar cálculos em dados
- Modificar itens individuais de dados
- Manipular a saída de grupos de linhas
- Formatar datas e números para exibição
- Converter tipos de dados de colunas

As funções SQL às vezes aceitam argumentos e sempre retornam um valor.

Há dois tipos de funções:

- Funções de uma única linha
- Funções de várias linhas ou de grupo

Funções de linha ou de registro

- Atuam sobre cada registo
- Produzem apenas um valor por registo
- Podem receber um ou mais argumentos de entrada
- Podem ser encadeadas
- Podem ser utilizadas onde se utilizam colunas, expressões, cláusulas SELECT, WHERE e ORDER BY.

Funções de manipulação de caracteres

- ASCII(char)

  Devolve o número que corresponde a esse caráter no código ASCII. Ver CHR.

Exemplo:

```
SQL>SELECT ASCII('S')
  2    FROM dual;

ASCII('S')
----------
        83
```

> **NOTA:** A tabela **DUAL** é uma tabela especial do sistema que existe apenas para se poderem fazer chamadas de funções. Em SQL não existe a possibilidade de fazer chamadas de funções directamente na linha de comando. A única alternativa é incluir as chamadas de funções num comando SQL. Assim, existe uma tabela especial do sistema, com uma coluna e uma linha, onde ninguém pode inserir mas todos podem seleccionar. Essa tabela, é a **DUAL**.

- CHR(n)

  Devolve o caracter com esse número. Ver ASCII. Exemplo:

```
SQL>SELECT RPAD(CHR(65), 2) "65",
  2        RPAD(CHR(67), 2) "67",
  3        RPAD(CHR(69), 2) "69",
  4        RPAD(CHR(71), 2) "71",
  5        RPAD(CHR(73), 2) "73",
  6        RPAD(CHR(75), 2) "75",
  7        RPAD(CHR(77), 2) "77",
  8        RPAD(CHR(79), 2) "79",
  9        RPAD(CHR(81), 2) "81",
 10        RPAD(CHR(83), 2) "83",
 11        RPAD(CHR(85), 2) "85",
 12        RPAD(CHR(87), 2) "87",
 13        RPAD(CHR(89), 2) "89",
 14        CHR(83) || CHR(73) || CHR(70) || CHR(73) ||
 15        CHR(67) || CHR(65) || CHR(80) "Juntar"
 16    FROM dual;

65 67 69 71 73 75 77 79 81 83 85 87 89 Juntar
-- -- -- -- -- -- -- -- -- -- -- -- -- -------
A  C  E  G  I  K  M  O  Q  S  U  W  Y  SIFICAP
```

- CONCAT (frase1, frase2)

  Junta duas strings ou frases. Exemplo:

```
SQL>SELECT CONCAT('Sifi', 'cap') "Juntar"
  2     FROM dual;

Juntar
-------
Sificap
```

  Também se podem concatenar strings com o operador ||. Exemplo:

```
SQL>SELECT 'Sifi' || 'cap' "Juntar"
  2     FROM dual;

Juntar
-------
Sificap
```

- **INITCAP:** Converte a primeira letra para maiúscula e as restantes para minúsculas. Ver
- **LOWER:** Converte strings de caracteres em maiúsculas ou em maiúsculas e minúsculas em strings de caracteres em minúsculas.
- **UPPER**: Converte strings de caracteres em minúsculas ou em maiúsculas e minúsculas em strings de caracteres em maiúsculas.

Function	Resultado
LOWER('SQL Course')	sql course
UPPER('SQL Course')	SQL COURSE
INITCAP('SQL Course')	Sql Course

- **CONCAT**: Une valores (Você está limitado a usar dois parâmetros com CONCAT.).
- **SUBSTR**: Extrai uma string de tamanho determinado
- **LENGTH**: Mostra o tamanho de uma string como um valor numérico

- **INSTR**: Localiza a posição numérica de um caractere nomeado
- **LPAD**: Preenche o valor do caractere à direita
- **RPAD**: Preenche o valor do caractere à esquerda
- **TRIM**: Reduz os caracteres à direita ou à esquerda (ou nas duas direções) de uma string de caracteres (**Se trim_character ou trim_source** for um literal de caractere, delimite-o por aspas simples.).

Observação: É possível usar functions como UPPER e LOWER com a substituição de E comercial (&). Por exemplo, use UPPER ('&job_title') para que o usuário não precise especificar o cargo em letras maiúsculas ou minúsculas.

- **LPAD:** Coloca espaços ou repetições de 'cadeia'à esquerda de stringaté o comprimento total da concatenação atingir um comprimento de n. Se nfor menor que o tamanho da string inicial, esta é cortada.
- **LTRIM:** Remove todos os espaços (ou ocorrências do caracter 'car', à esquerda da string de entrada).
- **REPLACE:** Procura em string sub-strings iguais a cadeia_inicial e substitui-as por cadeia_final. Se cadeia_final não for especificada, as sub-strings são apenas retiradas.
- **RPAD:** Devolve uma nova string que representa a string de entrada mais uma string constituída por n espaços se 'cadeia' for omitida ou então com repetições da string 'cadeia'.
- **RTRIM:** Retira todas as ocorrências do caracter 'car' que estejam à direita da string, ou caso 'car' não seja especificado, retirada todos os espaços em branco.
- **SUBSTR** (col|valor, pos, [n]): Este função devolve sub-strings da função de entrada a começar no caracter pos e de comprimento (facultativo) n. Se não for especificado, a sub-string começa no caracter pose vai até ao fim.

Exemplo:

```
SQL> SELECT nome,
  2         SUBSTR('ORACLE', 2,4) COL1,
  3         SUBSTR(nome, 2)       COL2,
  4         SUBSTR(nome,3,5)      COL3
  5     FROM DEP;

NOME            COL1 COL2            COL3
--------------- ---- --------------- -----
Contabilidade   RACL ontabilidade    ntabi
Investigação    RACL nvestigação     vesti
Vendas          RACL endas           ndas
Planeamento     RACL laneamento      aneam
```

Funções numéricas

- **ABS** (col|num): Devolve o valor absoluto da coluna ou expressão.
- **CEIL** (col|num): Devolve o menor inteiro que seja maior ou igual ao parâmetro de entrada.
- **FLOOR** (col|num): Devolve o maior inteiro que seja menor ou igual ao parâmetro de entrada.
- **MOD** (col1|num1, col2|num2): A função MOD localiza o resto do primeiro argumento dividido pelo segundo argumento.

Observação: A função MOD é usada com frequência para determinar se o valor é par ou ímpar.

- **ROUND:** Arredonda a coluna, a expressão ou o valor para n casas decimais; se n for omitido, não haverá casas decimais (Se n for negativo, os números à esquerda da vírgula decimal serão arredondados.).
- **TRUNC:** Trunca a coluna, a expressão ou o valor para n casas decimais; se n for omitido, n assumirá o default zero.

Funções de grupo

Atuam sobre conjuntos de registos.

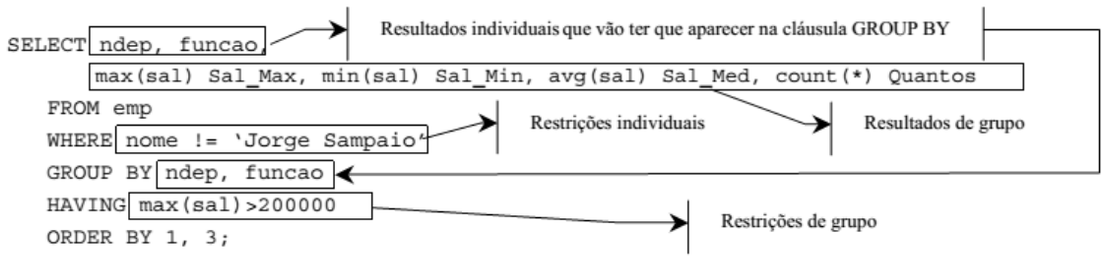
Produzem um valor por cada conjunto de registos. Por omissão, todos os registos de uma tabela são considerados um único grupo.

A cláusula GROUP BY da instrução SELECT permite subdividir uma tabela em grupos mais pequenos.

A cláusula HAVING permite efetuar condições de restrição sobre os resultados de conjuntos de registos da mesma maneira que a cláusula WHERE efetua condições de restrição sobre registos individuais.

NOTA: Se se incluir funções de grupo num comando SELECT não é possível incluir também resultados individuais a não ser que os resultados individuais apareçam na cláusula
GROUP BY.

Exemplo:



Cláusula GROUP BY

A clausula GROUP BY seguida pelo nome da coluna não precisa estar na lista SELECT.

```
SELECT      AVG (salary)
FROM        employees
GROUP BY department_id ;
```

Todas as funções de grupo, com a exceção da COUNT (*), ignoram os valores nulos.

Para cada uma das funções de grupo existe a possibilidade de se contarem apenas os valores distintos usando a palavra-chave DISTINCT ou contarem todos usando a palavra-chave ALL. A opção ALLé a usada por omissão.

1. AVG([DISTINCT|ALL] col|num)

Devolve a média de valores que essa expressão numérica representa. Note que apesar de os valores da coluna serem apenas inteiros, o valor da média pode ser um número fracionário.

2. COUNT ([DISTINCT|ALL] valor | *)

Devolve o número de registos na tabela que correspondem a uma expressão não nula. O caso especial COUNT (*) devolve o número de registos desse grupo mesmo que existam valores nulos.

Exemplo:

```
SQL> SELECT COUNT(*)              "Todos",
  2         COUNT(sal)            "#Sal não null",
  3         COUNT(premios)        "#Premios não null",
  4         COUNT(sal*14+premios) "#Expressão não null"
  5     FROM emp;

    Todos #Sal não null #Premios não null #Expressão não null
--------- ------------- ----------------- -------------------
       14            14                 4                   4
```

Exemplo :

```
SQL> SELECT count(ALL funcao)      "Todos",
  2>        count(DISTINCT funcao) "Profissões",
  3>        count(funcao)          "Igual a ALL"
  4>   FROM emp;

Todos       Profissões  Igual a ALL
----------- ----------- -----------
         14           5          14
1 row selected.
```

3. **MAX** ([DISTINCT|ALL] valor)

Devolve o máximo dos valores que essa expressão representa.

4. **MIN** ([DISTINCT|ALL] valor)

Devolve o mínimo dos valores que essa expressão representa.

5. **SUM** ([DISTINCT|ALL] valor)

Devolve a soma dos valores que essa expressão representa.

```
SELECT AVG(salary), MAX(salary),
       MIN(salary), SUM(salary)
FROM   employees
WHERE job_id LIKE '%REP%';
```

AVG(SALARY)	MAX(SALARY)	MIN(SALARY)	SUM(SALARY)
8150	11000	6000	32600

Cláusula HAVING

```
SELECT      column, group_function
FROM        table
[WHERE      condition]
[GROUP BY   group_by_expression]
[HAVING     group_condition]
[ORDER BY   column];
```

Não é possível usar a cláusula WHERE para restringir grupos.

Use a cláusula HAVING para restringir grupos.

Não é possível usar funções de grupo na cláusula WHERE.

Use a cláusula HAVING para especificar quais grupos devem ser exibidos e, dessa forma, restringir ainda mais os grupos com base nas informações agregadas.

Na sintaxe, group_condition restringe os grupos de linhas retornados aos grupos cuja condição especificada é verdadeira.

Quando você usa a cláusula HAVING, o servidor Oracle executa as seguintes etapas:

1. As linhas são agrupadas.
2. As funções de grupo são aplicadas ao grupo.
3. Os grupos que correspondem aos critérios na cláusula HAVING são exibidos.

A cláusula HAVING pode anteceder a cláusula GROUP BY, mas é recomendável usar a cláusula GROUP BY primeiro por razões lógicas. Os grupos são formados e as funções de grupo são calculadas antes de a cláusula HAVING ser aplicada aos grupos na lista SELECT.

Exercícios

1. Mostre o nome, função e departamento dos vendedores e apenas deles. O nome deverá aparecer todo em maiúsculas, a função em minúsculas e o departamento com a primeira letra em maiúscula e o resto em minúsculas. Ordene o resultado por nome do funcionário.

```
NOME                 FUNCAO       DEPARTAMENTO
-------------------- ------------ ---------------
ANA RODRIGUES        vendedor     Vendas
JOANA MENDES         vendedor     Vendas
MANUEL MADEIRA       vendedor     Vendas
NELSON NEVES         vendedor     Vendas
```

Resposta:

2. Escreva um comando que devolva o número de empregados. O resultado deve ser semelhante ao que se segue.

```
Total de empregados
-------------------
                 14
```

Resposta:

3. Escreva um comando que determine quantos empregados não ganham prémios. O resultado deve ser semelhante ao que se segue.

```
Empregados sem premio
---------------------
                   10
```

Resposta:

4. Escreva um comando que conte o número de empregados existentes, calcule o salário médio mensal e o total de remuneração anual auferido pelo conjunto de todos os empregados.

```
Total de empregados Salário médio mensal Remuneração total anual
------------------- -------------------- ------------------------
                 14            285666.07                 56226750
```

Resposta:

5. Mostre a lista dos Contínuos e o respectivo salário com um aumento de 13,55% para estes empregados. O salário depois do aumento deverá ser arredondado na primeira casa decimal.

```
NOME                 FUNCAO              SAL SALARIO_COM_AUMENTO
-------------------- ------------ --------- --------------------
Tome Ribeiro         Continuo          56950                64666.7
Rita Pereira         Continuo          65100                73921.1
Olga Costa           Continuo          68300                77554.7
Antonio Silva        Continuo          70800                80393.4
```

6. Escreva um comando que produza o seguinte resultado. Note que antes de cada nome de empregado existem 4 sinais de '>' (maior que) e um espaço, e que depois de cada nome existe um espaço e caracteres '<' (menor que) suficientes até se atingir um comprimento total de 25 caracteres.

```
Mariquices com strings
-------------------------
>>>> Ana Rodrigues <<<<<<
>>>> Antonio Silva <<<<<<
>>>> Augusto Reis <<<<<<<
>>>> Catarina Silva <<<<<
>>>> Duarte Guedes <<<<<<
>>>> Joana Mendes <<<<<<<
>>>> Jorge Sampaio <<<<<<
>>>> Manuel Madeira <<<<<
>>>> Maria Dias <<<<<<<<<
>>>> Nelson Neves <<<<<<<
>>>> Olga Costa <<<<<<<<<
>>>> Rita Pereira <<<<<<<
>>>> Silvia Teles <<<<<<<
>>>> Tome Ribeiro <<<<<<<
```

7. Indique o salário máximo, mínimo e médio e a quantidade de empregados para cada função e de cada departamento (grupos e subgrupos). Exclua registos individuais de nome 'Jorge Sampaio' e resultados coletivos que apresentem um salário máximo inferior ou igual a 200 contos. Ordene por departamento e depois por salário máximo.

NDEP	FUNCAO	SAL_MAX	SAL_MIN	SAL_MED	QUANTOS
10	Encarregado	279450	279450	279450	1
20	Encarregado	450975	450975	450975	1
20	Analista	565000	435000	500000	2
30	Vendedor	221250	145600	184225	4
30	Encarregado	380850	380850	380850	1

Funções de datas

Em Oracle as datas guardam valores sobre o século, ano, mês, dia, hora, minuto e segundo. Podem representar valores entre 1 de Janeiro de 4712 AC e 31 de Dezembro de 4712 DC. A data é guardada internamente através de um formato de dados desconhecido para o utilizador/programador. No entanto, é possível alterar o formato usado pelo Oracle para mostrar datas. Pode, por exemplo, ver-se datas com 2 dígitos no ano ou com 4 dígitos. Mas seja qual for a escolha, internamente é guardada toda a informação do ano.

- Operações aritméticas

As seguintes operações aritméticas são válidas para datas:

```
data  + dias      = nova_data
data  - dias      = nova_data
data1 - data2     = dias_diferenca (numero real positivo)
data  + horas/24 = data_mais_horas
data  + dias + horas/24 + minutos/1440 + segundos/86400 = nova_data;
```

Exemplo a somar dias, horas, minutos e segundos a uma data:

```
SQL>SELECT TO_CHAR(TO_DATE('98-07-01 00:00:00', 'YY-MM-DD hh24:mi:ss')
  2                          + 8          -- mais  8 dias
  3                          + 12/24      -- mais 12 horas
  4                          + 34/1440    -- mais 34 minutos
  5                          + 56/86400, -- mais 56 segundos
  6                 'yyyy-mm-dd hh24:mi:ss') "Somar"
  7    FROM DUAL;

Somar
-------------------------------------------------------------------
1998-07-09 12:34:56
```

Exemplo para subtrair datas:

```
SQL>SELECT TO_DATE('98-07-01 00:00:00', 'YY-MM-DD hh24:mi:ss')
  2      - TO_DATE('98-07-01 00:00:00', 'YY-MM-DD hh24:mi:ss')
  3                          + 8          -- soma 8 dias
  4                          + 12/24      -- mais 12 horas
  5                          + 34/1440    -- mais 34 minutos
  6                          + 56/86400   -- mais 56 segundos
  7           "Subtrai datas"
  8    FROM DUAL;

Subtrai datas
-------------
    8.5242593
```

- NLS_DATE_FORMAT

  O Oracle representa as datas segundo o formato definido em NLS_DATE_FORMAT. Pode alterar-se o seu valor através do comando ALTER SESSION.

  Alguns dos campos válidos do NLS_DATE_FORMAT são:

DDD	Dia do ano (número)
DD	Dia do mês (número)
DAY	Dia da semana (extenso)
MM	Número do mês
MON	Nome abreviado do mês
MONTH	Nome por extenso do mês
YYYY YYY YY Y	Representar o ano com 4, 3, 2 ou 1 dígitos respectivamente
HH HH12	Hora do dia (0-12)
HH24	Hora do dia (0-24)
MI	Minutos
SS	Segundos
SSSSS	Segundos depois da meia-noite
- / , . ; : "texto"	pontuação ou texto entre aspas

  Exemplo para alterar o formato da data:

```
SQL>ALTER SESSION
  2    SET NLS_DATE_FORMAT = '"Data:" YYYY-MON-DD HH24:MI:SS';

Session altered.

SQL>SELECT SYSDATE FROM dual;

SYSDATE
-------------------------
Data: 1998-OCT-27 20:23:35
```

NOTA: O comando ALTER SESSION permite modificar muitas outras opções NLS (*National Language Support* - ou numa tradução livre *Suporte à Língua Nacional* - tais como a língua, o território, a moeda, etc.) e ainda outro tipo de opções relacionadas com a ligação à base de dados. Consulte os manuais do Oracle para saber mais sobre o comando ALTER SESSION. Para poder ser usado, o utilizador precisa ter atribuído o privilégio de sistema com o mesmo nome.

Subconsultas

Uma das pesquisas mais comuns é encontrar, por exemplo, os dados do empregado que possuí o maior salário. Para obter essa informação é preciso fazer duas coisas: primeiro, determinar qual é o maior salário da empresa e depois selecionar o empregado com o salário igual a esse valor. Este problema e toda a sua classe de problemas (encontrar a informação em dois ou mais passos) resolve-se com uma técnica chamada subconsulta.

Uma subconsulta simples é uma consulta dentro de outra consulta. E os resultados

da consulta interna vão influenciar os da externa. Mais à frente, vamos ver subconsultas correlacionadas, onde é a consulta externa que influencia o consulta interna.
Exemplo:

```
SELECT  col1, col2, ..
      FROM lista_tabelas
      WHERE  colx = (SELECT  col_a1
                     FROM lista_tabelas2
                     WHERE ...);
```

Até agora foram vistos SELECTs internos que apenas devolvem uma coluna de dados (embora possam devolver mais de uma linha). No entanto pode desejar-se que o SELECT interno devolva várias linhas e várias colunas. Para tal devem colocar-se os nomes das colunas do WHERE externo entre parêntesis.
Exemplo:

```
SQL>SELECT  nome, sal, ndep
  2    FROM emp
  3    WHERE  (ndep, sal) IN (SELECT ndep, max(sal)
  4                           FROM emp
  5                           GROUP BY ndep)
  6    ORDER BY ndep, sal;

NOME                      SAL      NDEP
-------------------- --------- ---------
Jorge Sampaio           890000        10
Maria Dias              565000        20
Duarte Guedes           380850        30
```

Note que o SELECT interno pode representar por si só uma consulta bastante complicada, fazendo uso das cláusulas WHERE, GROUP BY e HAVING.

Regras para subconsultas simples

Pode usar-se subconsultas nas cláusulas WHEREe HAVING.

As subconsultas devem aparecer sempre entre parêntesis.

As subconsultas são executadas primeiro e os seus valores passados para a consulta imediatamente a exterior (da mais interior para a mais exterior).

Não existem limites para o número de SELECTs encadeados numa instrução.

Os SELECTs encadeados podem conter todas as cláusulas à exceção da ORDER BY.

A aparecer um ORDER BY este terá que ser referido apenas como cláusula do

SELECT externo.

Se uma subconsulta devolver apenas uma linha e uma coluna podem ser usados todos os operadores lógicos (=, >, <, >=, <= e!=) e operadores SQL (BETWEEN… AND…, NOT, etc.).

Se uma subconsulta devolver mais do que uma linha ou coluna ter-se-á que usar o comparador IN.

Se uma subconsulta devolver mais do que uma coluna (n colunas) então a comparação terá que ser feita entre n expressões e a subconsulta. As *n* expressões devem aparecer entre parêntesis, separadas por vírgulas e devem corresponder em tipo de dados às colunas devolvidas pela subconsulta.

Operador ANY

O operador ANY compara um valor com todos os valores de uma lista. Devolve verdade se a comparação resultar em **verdade para pelo menos um desses valores da lista**. Usa-se antes de uma lista e depois de um termo de comparação.
Exemplo:

```
SQL>SELECT nome, sal, ndep
  2    FROM emp
  3    WHERE sal > ANY (SELECT sal
  4                        FROM emp
  5                        WHERE ndep = 10
  6                           AND funcao != 'Presidente')
  7       AND funcao != 'Presidente'
  8    ORDER BY sal;

NOME                        SAL      NDEP
-------------------- --------- ---------
Antonio Silva            70800        20
Joana Mendes            145600        30
Manuel Madeira          157800        30
Nelson Neves            212250        30
Ana Rodrigues           221250        30
Silvia Teles            279450        10
Duarte Guedes           380850        30
Catarina Silva          435000        20
Augusto Reis            450975        20
Maria Dias              565000        20
```

O comando anterior devolve a lista de empregados (sem contar com o 'Presidente') com salário superior ao **menor** dos salários dos empregados do departamento 10 (sem contar com o 'Presidente').

Operador ALL

O operador ALL compara um valor com todos os valores de uma lista. Devolve verdade se a comparação resultar em **verdade para todos os valores da lista**. Usa-se antes de uma lista e depois de um termo de comparação.
Exemplo:

```
SQL>SELECT nome, sal, ndep
  2    FROM emp
  3    WHERE sal > ALL (SELECT sal
  4                     FROM emp
  5                     WHERE ndep = 10
  6                        AND funcao != 'Presidente')
  7       AND funcao != 'Presidente'
  8    ORDER BY sal;

NOME                      SAL      NDEP
-------------------- --------- ---------
Duarte Guedes           380850        30
Catarina Silva          435000        20
Augusto Reis            450975        20
Maria Dias              565000        20
```

O comando anterior devolve a lista de empregados (sem contar com o 'Presidente') com salário superior ao **maior** dos salários dos empregados do departamento 10 (sem contar com o 'Presidente').

Exercícios

1. Sem executar o comando seguinte determine e escreva em Português, o que o comando faz e mencione qual o seu resultado. Teste a sua resposta.

```
SQL>SELECT  avg(sal) "Media", ndep "NDep"
  2     FROM emp
  3     WHERE  (ndep, sal) IN (SELECT ndep, max(sal)
  4                                 FROM emp
  5                                 GROUP BY ndep)
  6         OR (ndep, sal) IN (SELECT ndep, min(sal)
  7                                 FROM emp
  8                                 GROUP BY ndep)
  9     GROUP BY ndep
 10     HAVING ndep = (SELECT ndep
 11                       FROM emp
 12                       GROUP BY ndep
 13                       HAVING count(*) = (SELECT max(count(*))
 14                                          FROM emp
 15                                          GROUP BY ndep))
 16     ORDER BY ndep, avg(sal);
```

Operadores de conjuntos

- Exemplo com UNION

```
SELECT funcao
   FROM emp
   WHERE ndep = 10
UNION
SELECT funcao
   FROM emp
   WHERE ndep = 30
ORDER BY funcao;
```

Devolve:

```
FUNCAO
------------
Continuo
Encarregado
Presidente
Vendedor
4 rows selected.
```

Os operadores de conjuntos UNION, UNION ALL, INTERSECT e MINUS permitem que se construam comandos com resultados de diferentes SELECTs combinados. Os comandos de SELECT a combinar podem mesmo referir-se a tabelas diferentes.

UNION - Combina linhas de duas queries eliminando duplicidades.

Note que independentemente de quantos SELECTs se tiver só existe uma única cláusula de ORDER BY que terá que aparecer no fim. Note também que existe apenas um ponto e vírgula no fim do comando todo.

Note que o primeiro SELECT, se executado sozinho, devolveria:

```
FUNCAO
------------
Presidente
Encarregado
Continuo
3 rows selected.
```

Enquanto que o segundo devolveria:

```
FUNCAO
------------
Encarregado
Vendedor
Vendedor
Vendedor
Vendedor
Continuo
6 rows selected.
```

Ou seja, o UNION elimina as linhas repetidas.

- Outro exemplo com `UNION`

```
SELECT funcao "Misturada"
   FROM emp
   WHERE funcao LIKE 'C%'
UNION
SELECT nome "Nome Emp"
   FROM emp
   WHERE nome LIKE 'C%'
UNION
SELECT nome "Nome Dep"
   FROM dep
   WHERE nome LIKE 'C%'
ORDER BY 1;
```

Devolve:

```
Misturada
--------------------
Catarina Silva
Contabilidade
Continuo
3 rows selected.
```

UNION ALL - Combina linhas de duas queries e não elimina as duplicidades.

- Exemplo com `UNION ALL`

```
FUNCAO
------------
Continuo
Continuo
Encarregado
Encarregado
Presidente
Vendedor
Vendedor
Vendedor
Vendedor
9 rows selected.
```

Repare como se podem misturar colunas diferentes de tabelas diferentes num único resultado. Note ainda que a cláusula ORDER BY terá que ter uma referência numérica. Repare também que no caso de existirem pseudónimos aparecem os usados

no primeiro SELECT mesmo que este não apresente as suas linhas em primeiro no resultado.

INTERSECT - Combina linhas e só retorna as que estão presentes nas duas queries.

Exemplo com INTERSECT:

```
SELECT funcao
   FROM emp
   WHERE ndep = 10
INTERSECT
SELECT funcao
   FROM emp
   WHERE ndep = 30
ORDER BY funcao;
```

Devolve

```
FUNCAO
------------
Continuo
Encarregado
2 rows selected.
```

MINUS - Combina linhas e retorna as linhas da primeira query que não estão presentes na segunda query.

Instruções DML

Entende-se por DML (**Data Manipulation Language)** ou Linguagem de Manipulação de Dados a recuperação de dados do banco de dados, informação novas linhas, alteração linhas existentes e remoção linhas indesejadas das tabelas do banco de dados, respectivamente.

Inserção de dados (INSERT)

O comando INSERT tem as seguintes opções:

INSERT INTO (tabela|vista) [(nome_col1, nome_col2,…)].

VALUES (valor1, valor2,…)

Ou

INSERT INTO (tabela|vista) [(nome_col1, nome_col2,…)] subconsulta.

A primeira opção insere apenas um registo de cada vez enquanto que a segunda insere um ou mais registos. Na segunda opção, os valores a inserir são retirados de uma subconsulta.

Exemplo 1:

Inserção de um registo e uso dos nomes das colunas da tabela destino:

```
INSERT INTO emp (nemp, nome, função, encar,
data_entrada, sal, prêmios, ndep).
VALUES (2222, 'Lobo Mau', 'Comilao', null,
sysdate, 555, null, null);
1 row processed.
```

O primeiro conjunto de parêntesis serve apenas para indicar a que correspondem os valores que estão a ser inseridos. É facultativo desde que os valores mencionados a seguir à cláusula VALUE estejam na mesma ordem das colunas da tabela e desde que sejam todos mencionados. Assim, o comando do exemplo 1 produz o mesmo resultado que o comando do exemplo 2.

Exemplo 2:

Inserção de um registo sem especificar os nomes das colunas (produz o mesmo resultado que o exemplo 1):

```
INSERT INTO emp
VALUES (2222, 'Lobo Mau', 'Comilao', null,
sysdate, 555, null, null);
1 row processed.
```

Exemplo 3:

Inserção de um registo sem incluir todos os dados e alterando a ordem de algumas colunas (produzem o mesmo resultado que os dois exemplos anteriores):

```
INSERT INTO emp (nome, função, nemp, data_entrada, sal).
VALUES ('Lobo Mau', 'Comilao', 2222, sysdate, 555)
1 row processed.
```

Exemplo 4:

Inserção de registos através de uma subconsulta. Comando que insere todos os empregados da tabela **emp** outra vez. Note que se somou 1000 ao número do empregado para garantir que após a inserção cada empregado tivesse um valor de **nemp** diferente de todos os outros. Os restantes valores de cada empregado foram mantidos com a exceção do **nome** a que foi acrescentado o prefixo de 'CLONE'e a data_entrada que ficou com o valor da data atual.

```
INSERT INTO emp
SELECT nemp+1000, 'CLONE ' || nome, funcao, encar,
sysdate, sal, premios, ndep
FROM emp;
14 rows processed.
```

Depois de executado, se selecionar alguma informação da tabela obtém-se:

```
SELECT nemp, nome, funcao, encar, data_entrada
   FROM emp
   ORDER BY nemp;

NEMP       NOME                 FUNCAO       ENCAR      DATA_ENTR
---------- -------------------- ------------ ---------- ---------
      1369 Antonio Silva        Continuo           1902 22-DEC-96
      1499 Joana Mendes         Vendedor           1698 04-OCT-84
      1521 Nelson Neves         Vendedor           1698 27-FEB-83
      1566 Augusto Reis         Encarregado        1839 13-FEB-85
      1654 Ana Rodrigues        Vendedor           1698 17-DEC-90
      1698 Duarte Guedes        Encarregado        1839 25-NOV-91
      1782 Silvia Teles         Encarregado        1839 03-NOV-86
      1788 Maria Dias           Analista           1566 07-NOV-82
      1839 Jorge Sampaio        Presidente                11-FEB-84
      1844 Manuel Madeira       Vendedor           1698 21-APR-85
      1876 Rita Pereira         Continuo           1788 07-FEB-96
      1900 Tome Ribeiro         Continuo           1698 05-MAR-94
      1902 Catarina Silva       Analista           1566 13-APR-93
      1934 Olga Costa           Continuo           1782 22-JUN-86
      2369 CLONE Antonio Silva  Continuo           1902 04-NOV-98
      2499 CLONE Joana Mendes   Vendedor           1698 04-NOV-98
      2521 CLONE Nelson Neves   Vendedor           1698 04-NOV-98
      2566 CLONE Augusto Reis   Encarregado        1839 04-NOV-98
      2654 CLONE Ana Rodrigues  Vendedor           1698 04-NOV-98
      2698 CLONE Duarte Guedes  Encarregado        1839 04-NOV-98
      2782 CLONE Silvia Teles   Encarregado        1839 04-NOV-98
      2788 CLONE Maria Dias     Analista           1566 04-NOV-98
```

NOTA: Apenas no fim da execução do comando `INSERT` é que o SGBD da ORACLE verifica se existiu ou não violação de integridade. No exemplo anterior, se o primeiro elemento a ser inserido tivesse um valor de encar de por exemplo 5555 (que é um número de empregado que não existe) a inserção continuava até ao último dos elementos. Só então, no fim do comando, seria detectada a violação e desfeito o comando. Mas se em vez de 5555, o primeiro elemento (`'CLONE Antonio Silva'`) tivesse o encarregado número 2934 (o número de empregado da última inserção) o resultado já seria diferente. Durante a inserção do `'CLONE Antonio Silva'` existe realmente violação de integridade uma vez que o 2934 ainda não foi inserido. A violação não é detectada porque o comando `INSERT` ainda não terminou. Por outro lado, quando o comando `INSERT` termina, a referência a 2934 já não representa uma violação de integridade porque esse elemento foi entretanto inserido. Ou seja, o `INSERT` permite violar as restrições de dados temporariamente.

Modificação de Dados (UPDATE)

A instrução que permite modificar dados é a UPDATE. As suas opções são as seguintes:

UPDATE (tabela|vista)
SET col1 = val1, col2 = val2, …,
(col101, col102, …) = (subconsulta)
[WHERE (condição)];

A cláusula UPDATE determina sobre que tabela ou vista se vão executar as alterações.

A cláusula SET indica pares de colunas e valores a atribuir a colunas. Os pares estão separados por vírgulas. Além de ou invés dos pares de colunas pode ainda incluir-se atribuições através de subconsultas. Nesse caso, as colunas a terem valores atribuídos deverão aparecer entre parêntesis e separadas por vírgulas.

A cláusula WHERE indica sobre que registos da tabela ou vista serão executadas as alterações. A condição pode ser também uma subconsulta.

As subconsultas podem ser correlacionadas.

Exemplo 1:

Aumenta o salário de todos os empregados em 10%:

```
UPDATE emp
SET sal = sal * 1.1;
14 rows processed.
```

Exemplo 2:

Aumenta o salário de todos os empregados do departamento 20 em 10%:

```
UPDATE emp
SET sal = sal * 1.1
WHERE ndep = 20;
5 rows processed.
```

Exemplo 3:

Muda o salário e a função do 'Jorge Sampaio':

```
UPDATE emp
SET sal = sal * 2, função = 'CHEFAO'.
WHERE nome = 'Jorge Sampaio';
1 rows processed.
```

Exemplo 4:

Mistura atribuições normais e subconsultas na cláusula SET. Usa subconsulta correlacionada na cláusula SET e subconsulta na cláusula WHERE. O exemplo altera o número do departamento dos registos afetados para o número do departamento de nome 'Coimbra'(1ª subconsulta do SET). Altera os pares salário, prémios para 110% da média dos salários e 150% da média dos prémios dos empregados do seu departamento (2ª subconsulta do SET– subconsulta correlacionada). Efetua as alterações apenas para os registos que correspondam a empregados dos departamentos de 'Coimbra' e da 'Mealhada' (subconsulta do WHERE).

```
UPDATE emp a
SET ndep = (SELECT ndep
FROM dep
WHERE local = 'Coimbra'),
(sal, premios) = (SELECT 1.1*AVG(sal), 1.5*AVG(premios)
FROM emp b
WHERE a.ndep = b.ndep)
WHERE ndep IN (SELECT ndep
FROM dep
WHERE local = 'Mealhada' mik,,,
```

('Mealhada') e passaram para o 30 ficaram com um salário de 349113.

Remoção de Dados (DELETE)

Usa-se o comando DELETE para remover registos das tabelas ou vistas. O comando permite apagar apenas registos inteiros. Não é possível apagar um campo com o DELETE. No máximo poder-se-ia colocar o valor NULLnesse campo através de um comando UPDATE.

As opções do comando DELETE são as seguintes:

DELETE FROM tabela|vista
WHERE condições

Como de costume, as condições na cláusula WHERE podem incluir subconsultas e as vistas na cláusula FROM podem ser definidas através de subconsultas.

Exemplo 1:

Apagamento de todos os registos da tabela emp.

```
DELETE FROM emp;
14 rows processed.
```

NOTA: Um erro comum é escrever 'DELETE * FROM emp'. O asterisco usa-se no SELECT para indicar as colunas todas. Mas no comando DELETE não faz sentido indicar a lista de colunas a apagar porque ou se apaga o registo todo ou não se apaga o registo. Não é possível apagar apenas algumas colunas.

Exemplo 2:

Apagamento dos registos da tabela emp que correspondem aos empregados com salários menores que 100000.

```
DELETE
FROM emp
```

WHERE sal < 100000;
4 rows processed.

Exemplo 3:

Apagamento dos registos da tabela emp que correspondem aos empregados com salários menores que 100000 e que pertençam ao departamento 20.

DELETE
FROM emp
WHERE sal < 100000
AND ndep = 20;
2 rows processed.

Exemplo 4:

Apagamento dos registos da tabela empem que a seleção é feita através de subconsultas. O comando apaga todos os empregados do departamento com mais pessoas.

```
DELETE
   FROM emp
   WHERE ndep = (SELECT ndep
                 FROM emp
                 GROUP BY ndep
                 HAVING COUNT(*) = (SELECT MAX(COUNT(*))
                                    FROM emp
                                    GROUP BY ndep));
6 rows processed.
```

Seleção de Dados (SELECT)

O comando SELECT é utilizado para recuperar registros de uma ou mais tabelas, sintaxe:

SELECT [DINSTINCT] {*| (nome_coluna1|expr1) alias,
(nome_coluna1|expr1) alias,

…}
FROM nome_da_tabela
[WHERE condição].

Exemplo 1:

```
SELECT * FROM PESSOAS;
```

Nesse exemplo foram selecionadas todas as colunas da tabela pessoas (o asterisco indica que todas as colunas devem ser mostradas).

Exemplo 2:

```
SELECT fun. cod_funcionario, fun.val_sal FROM funcionários fun ;
```

Aqui foram recuperados o código e o salário de todos os funcionários (perceba o ALIAS utilizado para a tabela de funcionários “fun”)

Exemplo 3:

```
SELECT *
FROM departments
WHERE flg_ativo = ‘S’
AND sal_max > 2000
AND sal_min between 1000 and 2000;
```

Instrução DDL

Configurar, alterar e remover estruturas de dados das tabelas. Conhecida coletivamente como DDL (Data Definition Language) ou Liguagem de Definição de Dados.

Criar Tabelas (CREATE TABLE)

Uma tabela define-se pelo seu nome, pelo nome das suas colunas, pelo tipo de dados que comporta cada coluna, por restrições de colunas e por restrições de tabela. Na realidade existem ainda outras opções avançadas que se podem definir nas tabelas.

Essas opções avançadas têm haver com a forma como o servidor guarda a tabela em disco, com parâmetros de optimização de acesso aos dados e outros. Por agora vamos ficar apenas pelas características básicas.

O comando SQL para criar uma tabela é o seguinte:

```
CREATE TABLE nome_tabela
( col1  tipo_col1  [DEFAULT expre] [restrição1 [restrição2 […]]],
  col2  tipo_col2  [DEFAULT expre] [restrição1 [restrição2 […]]],
  col3  tipo_col3  [DEFAULT expre] [restrição1 [restrição2 […]]],
  …,
  coln  tipo_coln  [DEFAULT expre] [restrição1 [restrição2 […]]],
  [restrição1_tabela,]
  [restrição2_tabela,]
  [restrição3_tabela,]
  …,
  [restriçãon_tabela]
);
```

Restrições de coluna

Restrições de tabela

A seguir às palavras CREATE TABLE declara-se o nome da tabela a criar. Entre parênteses define-se uma lista de colunas ou restrições à tabela separadas por vírgulas.

Ao definirem-se as colunas determina-se qual o nome da coluna, que tipo de dados comporta, qual o valor por omissão dos dados nessa coluna e que restrições se devem verificar sobre os dados dessa coluna. As restrições da tabela indicam normalmente restrições sobre dados de mais do que uma coluna e usam uma sintaxe ligeiramente diferente porque indicam explicitamente a que coluna(s) se refere. As restrições de coluna, por outro lado, indicam implicitamente a que coluna se refere através da linha em que se localizam.

Exemplo 1:

Criar uma tabela, alunos, com duas colunas, numero e nome. A primeira coluna é do tipo numérico com até 10 algarismos e a segunda é do tipo varchar2 (cadeia de caracteres) com espaço até 40 caracteres por nome. Não é indicada nenhuma restrição sobre nenhuma coluna da tabela nem é definida nenhuma chave primária.

CREATE TABLE **alunos**
(**numero** NUMBER (10),
nome VARCHAR2(40)

Restrição de Chave Primária (PRIMARY KEY)

A chave primária de uma tabela é a coluna ou conjunto de colunas que servem para definir univocamente um registo na tabela. Uma coluna de chave primária não pode conter valores nulos nem pode conter valores repetidos.

Apesar de o Oracle permitir que se crie uma tabela sem definir uma chave primária normalmente nenhuma tabela é criada sem uma.

Pode declarar-se a chave primária na zona das restrições de coluna da chave primária ou no conjunto de restrições da tabela. No caso de chaves primárias concatenadas (quando a chave é definida pelos valores de mais de uma coluna) a única maneira de definir a chave é através das restrições da tabela.

Colunas com dados do tipo LONG e LONG RAW não podem ser chaves primárias de uma tabela.

O prefixo normalmente usado para os nomes das restrições de chave primária é pk_ (as iniciais de primary key, “chave primária” em inglês).

Para as chaves primárias, o caracter usado internamente pelo Oracle para distinguir restrições é o P(de primary). Investigue as vistas sobre restrições do dicionário de dados.

A sintaxe para se definir uma coluna como chave primária é uma das seguintes:
[CONSTRAINT nome_restrição] PRIMARY KEY

Exemplo 2:

Definir uma chave primária nas restrições de coluna.

CREATE TABLE **alunos**
(**numero** NUMBER(10) PRIMARY KEY,
nome VARCHAR2(40)

Exemplo 3:

Definir uma chave primária nas restrições de coluna e dar um nome à restrição.

CREATE TABLE **alunos**
(**numero** NUMBER (10) CONSTRAINT **pk_numero_alunos** PRIMARY KEY,
nome VARCHAR2(40)

Note que o nome da restrição é opcional. Caso seja omitido, o Oracle define automaticamente um nome para a restrição. Neste exemplo chama-se pk_numero_alunos.

Exemplo 4:

Definir uma chave primária nas restrições de tabela e dar um nome à restrição.

CREATE TABLE **alunos**
(numero NUMBER (10),
nome VARCHAR2(40),
CONSTRAINT pk_numero_alunos PRIMARY KEY (numero)

Note que quando se define a restrição de chave primária nas restrições da tabela é necessário indicar qual das colunas é a chave primária e indicar o seu nome entre parêntesis a seguir às palavras PRIMARY KEY.

Novamente note que o nome da restrição é opcional. Caso seja omitido, o Oracle define automaticamente um nome para a restrição.

Exemplo 5:

Definir uma chave primária concatenada nas restrições de tabela e dar um nome à restrição. No exemplo assume-se que o aluno é definido univocamente pelo seu ano de entrada e pelo seu número de entrada nesse ano.

CREATE TABLE **alunos**
(**ano_entr** NUMBER (6),
num_entr NUMBER(4),
nome VARCHAR2(40),
CONSTRAINT pk**_ano_num_alunos** PRIMARY KEY (**ano_entr, num_entr**));

Para chaves primárias concatenadas, o único local possível para a definição da restrição é na secção de restrições da tabela. A chave primária da tabela pode ser definida pela concatenação de todas as colunas da tabela. O nome da restrição é opcional.

Restrição de Unicidade (UNIQUE)

Uma coluna que obedeça à restrição de Unicidade não pode ter valores repetidos. Note no entanto que uma coluna nestas condições PODE conter valores nulos. Assim, uma coluna deste género não é nunca uma chave candidata por causa da possibilidade de existirem valores nulos.

Pode declarar-se a coluna com restrições de Unicidade nas restrições dessa coluna ou no conjunto de restrições da tabela.

O prefixo normalmente usado para os nomes das restrições de unicidade é uni_ou unq_ (as iniciais de unique, “único” em inglês).

Para as colunas do tipo UNIQUE, o caracter usado internamente pelo Oracle para distinguir restrições é o U (de unique).

A sintaxe para se definir uma coluna com restrição de Unicidade é uma das seguintes:

[CONSTRAINT nome_restrição] UNIQUE

Exemplo 1:

Definir uma coluna, bi, com restrição de Unicidade e dar um nome à restrição.

CREATE TABLE **alunos**
(**numero** NUMBER(10) PRIMARY KEY,
nome VARCHAR2(40),
bi NUMBER (10) CONSTRAINT **uni_bi_alunos** UNIQUE

Exemplo 2:

Definir uma coluna, bi, com restrição de Unicidade e dar um nome à restrição. A diferença em relação ao Exemplo 1 é que a restrição é declarada na zona de restrições da tabela em vez de estar na zona de restrições da coluna.

CREATE TABLE **alunos**
(**Numero** NUMBER(10) PRIMARY KEY,

nome VARCHAR2(40) ,
bi NUMBER(10) ,
CONSTRAINT **uni_bi_alunos** UNIQUE (bi));

Restrição de Integridade Geral (CHECK)

É possível obrigar a que na inserção de dados sejam verificadas condições. As condições a verificar apenas permitem a inserção de dados se o valor devolvido for Verdade ou Desconhecido no caso de existir comparações com valores nulos.

As condições a verificar podem incluir expressões aritméticas e chamadas de funções SQL.

As condições a verificar não podem incluir:

- Chamadas às funções SYSDATE, UID, USERe USERENV
- Referências às pseudo-colunas CURRVAL, NEXTVAL, LEVELe ROWNUM.
- Subconsultas
- Referências a colunas de outras tabelas ou a outras linhas da mesma tabela .

Numa restrição de integridade se aparecer referências a colunas da mesma tabela assume-se que indicam campos desse mesmo registo. Este tipo de restrições onde se refere mais do que uma coluna na mesma restrição terá que ser definida na zona de restrições da tabela.

Pode declarar-se a restrição de integridade na zona de restrições da coluna ou na zona de restrições da tabela.

O prefixo normalmente usado para os nomes das restrições de integridade gerais é ck_ (de check, “verifica que” em inglês).

Para as colunas deste tipo, o caracter usado internamente pelo Oracle para distinguir restrições é o C (de check).

A sintaxe para se definir uma coluna com restrição de integridade geral é:
[CONSTRAINT **nome_restrição**] CHECK (**condição**)

Exemplo 1:

Definir uma coluna, ano, que se contiver um valor terá que ser obrigatoriamente acima de 1980 e dar um nome à restrição.

CREATE TABLE **alunos**

(**numero** NUMBER (10) PRIMARY KEY,
nome VARCHAR2(40),
ano NUMBER (4) CONSTRAINT **ck_ano_alunos** CHECK (ano>1980)

Note que pode inserir-se um registo com o valor do campo ano a NULL. Nesse caso a condição devolve o valor de Desconhecido e permite a inserção.

SQLWKS> INSERT INTO **alunos** VALUES (1111, 'Ana', 1970);
ORA-02290: **check constraint** (UTILIZADOR1. CK_ANO_ALUNOS**) violated.**
SQLWKS> INSERT INTO **alunos** VALUES (2222, 'Julia', NULL);
1 row processed.
SQLWKS> INSERT INTO alunos VALUES (3333, 'Romeu', 1981);
1 row processed.

Exemplo 2:

Faz o mesmo que o Exemplo 8.8, mas define a restrição na zona de restrições da tabela.

CREATE TABLE **alunos**
(**numero** NUMBER (10) PRIMARY KEY,
nome VARCHAR2(40),
ano NUMBER (4),
CONSTRAINT **ck_ano_alunos** CHECK (ano>1980)

Restrição de Proibição de Valores Nulos (NOT NULL)

Uma coluna sem valores nulos é uma em que é sempre obrigatória a entrada de um valor. Os valores podem ser repetidos.

Pode declarar-se a coluna sem valores nulos nas restrições dessa coluna ou no conjunto de restrições da tabela.

O prefixo normalmente usado para os nomes das restrições de colunas sem valores nulos é nn_ (as iniciais de not null, “não nulos” em inglês).

Para as colunas do tipo NOT NULL, o caracter usado internamente pelo Oracle para distinguir restrições é o C (de check).

Na realidade, a restrição NOT NULL é um caso particular da restrição geral CHECK,

e daí o uso da letra C.

A sintaxe para se definir uma coluna com restrição de valores não nulos é uma das seguintes:
[CONSTRAINT **nome_restrição**] NOT NULL no caso de se definir a restrição nas restrições de coluna, ou [CONSTRAINT **nome_restrição**] CHECK (**coluna** IS NOT NULL) no caso de se definir a restrição nas restrições de tabela.

Exemplo 1:

Definir uma coluna, nome, sem valores nulos e dar um nome à restrição.

CREATE TABLE **alunos**
(**numero** NUMBER (10) PRIMARY KEY,
nome VARCHAR2(40) CONSTRAINT **nn_nome_alunos NOT NULL**

Exemplo 2:

Definir uma coluna, nome, sem valores nulos e dar um nome à restrição. A diferença em relação ao Exemplo 1 é que a restrição é declarada na zona de restrições da tabela em vez de estar na zona de restrições da coluna.

CREATE TABLE **alunos**
(**numero** NUMBER (10) PRIMARY KEY,
nome VARCHAR2(40),
CONSTRAINT **nn_nome_alunos** CHECK (**nome** IS NOT NULL)

Restrições de Chave Forasteira (FOREIGN KEY… REFERENCES).

Uma chave forasteira ou estrangeira (FOREIGN KEY) é uma referência a um valor que existe noutra coluna da mesma tabela ou de outra tabela. A coluna referenciada tem que ser uma chave primária ou uma coluna UNIQUE. A chave forasteira (simples ou composta) e a(s) coluna(s) referenciada(s) precisam concordar em tipo de dados e em número de colunas. Uma chave forasteira não pode ser do tipo LONG nem do tipo LONG RAW. Numa coluna de chave forasteira podem existir valores repetidos e valores nulos.

O prefixo normalmente usado para os nomes das restrições de colunas de chave forasteira é fk_ (as iniciais de foreign key, “chave forasteira” em inglês).
Para as colunas de chave forasteira, o caracter usado internamente pelo Oracle para distinguir restrições é o R (de references).

A sintaxe para se definir uma coluna com restrição de chave forasteira é uma das seguintes:
[CONSTRAINT **nome_restrição**] REFERENCES **tabela** [(coluna)]
[ON DELETE CASCADE] no caso de se definir a restrição nas restrições de coluna. A seguir à palavra REFERENCES indica-se o nome da tabela e coluna referenciadas. O nome da coluna é opcional e se não for mencionado é usada à coluna de chave primária da tabela referida.

A opção ON DELETE CASCADE indica que se o registo da chave referenciada for apagado então geram-se apagamentos em cascata para todos os registos dependentes desse.

Alternativamente pode definir-se uma chave forasteira da seguinte forma:

[CONSTRAINT **nome_restrição**] FOREIGN KEY(col1, col2, ...)
REFERENCES **tabela** (colA colB,...).
[ON DELETE CASCADE] no caso de se definir a restrição nas restrições de tabela. Note que a
única maneira de definir chaves forasteiras compostas é usando a sintaxe anterior.

Exemplo 1:

Uso de chave forasteira simples. Neste exemplo considera-se que cada aluno teve um padrinho de praxe académica. O padrinho não é mais que uma referência a outro aluno. Cada aluno pertence também a um departamento cuja descrição está guardada numa tabela de departamentos. Para simplificar, as restantes restrições são declaradas sem nome.

CREATE TABLE **alunos**
(numero NUMBER(10) PRIMARY KEY,
nome VARCHAR2(40) NOT NULL,
bi NUMBER (10) UNIQUE,

morada VARCHAR2(80),
padrinho NUMBER (10) CONSTRAINT **fk_padrinho_alunos**
REFERENCES **alunos (numero),**
Ndep NUMBER (3) CONSTRAINT **fk_ndep_alunos**
REFERENCES **departamentos (ndep)**
ON DELETE CASCADE

A restrição fk_padrinho_alunos indica apenas que a coluna padrinho guarda um número que terá que aparecer na coluna numero da tabela alunos. Da mesma forma, a restrição fk_ndep_alunosindica que o número de departamento do aluno, ndep, terá que indicar um valor que já exista na coluna com o mesmo nome na tabela departamentos.

A opção ON DELETE CASCADE nesta chave forasteira indica que quando se apaga o registo pai, todos os dependentes são apagados também. Ou seja, se se apagar um departamento da tabela de departamentos todos os seus alunos são também apagados.

No caso da chave forasteira padrinho, isso não acontece. Assim, só é possível apagar um aluno se ele não for o padrinho de ninguém.

Exemplo 2:

Uso de chave forasteira compostas ou concatenadas. Neste exemplo considera-se que a BD tem 4 tabelas: alunos, disciplinas, exame_freq e notas. A tabela exame_freq tem como chave forasteira o código da disciplina. A tabela notas tem várias chaves forasteiras. Uma é o número de aluno que se refere à tabela de alunos, outra é o código da disciplina que se refere à tabela das disciplinas (sem referir à coluna) e outra que indica a época em que saiu a nota que é uma chave forasteira composta que se refere à chave primária (composta também) da tabela exame_freq. Como na tabela de notasse indica ON DELETE CASCADE para a chave forasteira do número de aluno isso significa que se o aluno for apagado todas as suas notas são apagadas também. Pelo contrário, como não se indica ON DELETE CASCADE para a chave forasteira disc em notas, isso significa que uma disciplina só pode ser apagada se não existirem notas atribuídas a essa disciplina.

```
CREATE TABLE alunos
(num          NUMBER(10)    PRIMARY KEY,
 nome         VARCHAR2(30) NOT NULL
);

CREATE TABLE disciplinas
(disc         NUMBER(4)     PRIMARY KEY,
 nome         VARCHAR2(30) NOT NULL);

CREATE TABLE exame_freq
(ano_lec      NUMBER(4)     ,
 epoca        VARCHAR2(20) ,
 disc         NUMBER(4)     REFERENCES disciplinas,
 data         DATE          ,
 PRIMARY KEY (ano_lec, epoca, disc)
);

CREATE TABLE notas
(ano_lec      NUMBER(4)     ,
 epoca        VARCHAR2(20) ,
 disc         NUMBER(4)     REFERENCES disciplinas,
 num_alu      NUMBER(10)    REFERENCES alunos(num)
                               ON DELETE CASCADE,
 CONSTRAINT   fk_notas      FOREIGN KEY(ano_lec, epoca, disc)
                            REFERENCES exame_freq(ano_lec, epoca, disc)
);
```

Acrescentar Colunas

Para acrescentar uma coluna a uma tabela já existente usa-se a seguinte sintaxe:

ALTER TABLE **tabela**
ADD (**coluna tipo_coluna [restrições] [valor omissão**]);

Exemplo 1:

Acrescentar uma coluna a empregados. Definir restrições sobre a coluna bem como valores por omissão.

ALTER TABLE **empregados**
ADD (**sexo** CHAR (1)
DEFAULT 'F'

CONSTRAINT **ck_sexo_empregados** CHECK (sexo IN ('M', 'F')).

Alterar Colunas

Para alterar colunas (tipo, comprimento dos dados, valores por omissão) usa-se o comando:

ALTER TABLE **tabela**
MODIFY (**coluna [tipo_coluna]** [**valor omissão**]);

Note que não é possível com a cláusula MODIFY coluna alterar ou remover restrições sobre essas colunas. Para isso é preciso à cláusula DROP restrição. Os parêntesis são opcionais porque a cláusula MODIFY refere-se sempre a uma e uma só coluna.

Exemplo 1:

Alterar o tipo de dados

ALTER TABLE **empregados**
MODIFY (**sexo** CHAR (3));

Exemplo 2:

Alterar o valor por omissão

ALTER TABLE **empregados**
MODIFY (**sexo** DEFAULT '**Fem'**);

Note que este valor por omissão entra em conflito com a restrição de integridade, ck_sexo_empregados.

Remover Restrições

Para remover uma restrição usa-se:

ALTER TABLE **tabela**
DROP CONSTRAINT **restrição;**

Exemplo 1:

Remover a restrição da coluna sexo
ALTER TABLE **empregados**
DROP CONSTRAINT **ck_sexo_empregados**;

Acrescentar Restrições

Para acrescentar uma restrição usa-se:

ALTER TABLE **tabela**
ADD CONSTRAINT **nome restrição condição** [DISABLE];

Se se usar a palavra DISABLE, a restrição é acrescentada, mas não é ativada. Ou seja, existe, mas ainda não está a funcionar.

Exemplo 1:

Acrescentar uma restrição à coluna sexo.

ALTER TABLE **empregados.**
ADD CONSTRAINT **ck_sexo_empregados** CHECK (sexo IN ('Fem', 'Mas'));

Repare que a menos que a tabela esteja vazia este comando resulta no seguinte código de erro:

ORA-02293: **cannot enable** (UTILIZADOR1. CK_SEXO_EMPREGADOS) - **check. constraint violated**

Isso acontece porque os registos já inseridos na tabela têm o campo sexo com o valor 'F'ou 'M'.

Para poder acrescentar a restrição pretendida ou ter-se-ia que apagar todos os registos da tabela ou ter-se-ia que alterar os 'F's para 'Fem's e os 'M's para 'Mas's. Um comando possível seria:

UPDATE empregados
SET sexo = DECODE (sexo, ‘F’, ‘Fem’, ‘M’, ‘Mas’, null);

Agora, garantidamente, já seria possível acrescentar a restrição de acordo com o. Exemplo 1.
Notar que a sintaxe a usar para acrescentar restrições segue a sintaxe

Desativar Restrições (DISABLE CONSTRAINT)

Podem desativar-se restrições. Apesar de existir, temporariamente a restrição não será verificada. Todos os empregados têm um encarregado. Isso obriga a que só depois de se inserir os encarregados é possível inserir os subordinados. Para evitar que se tenha que inserir os dados de exemplo da tabela numa ordem específica pode temporariamente desativar a restrição. No fim da inserção basta ativar de novo a restrição
A sintaxe para desativar restrições é:

ALTER TABLE **tabela**
DISABLE CONSTRAINT **nome_condição;**

Exemplo 1:

Desativa a restrição ck_sexo_empregados

ALTER TABLE **empregados**
DISABLE CONSTRAINT **ck_sexo**_empregados;

Ativar Restrições

Da mesma forma que é possível desativar restrições também é possível ativá-las.
A sintaxe para ativar restrições é:

ALTER TABLE **tabela**
ENABLE CONSTRAINT **nome_condição;**

Exemplo 1:

Ativa a restrição ck_sexo_empregados

ALTER TABLE **empregados**
ENABLE CONSTRAINT **ck_sexo_empregados**;

Apagar Tabelas (DROP TABLE)

Para apagar tabelas usa-se o comando DROP TABLE seguido do nome da tabela.
Exemplo para apagar a tabela emp:

DROP TABLE **emp**;

Se existirem chave forasteiras, noutras tabelas, a apontar para algum dos registos da tabela a apagar, o comando devolve um erro:

SQLWKS> DROP TABLE **dep**
2>
DROP TABLE **dep**
*
ORA-02449: **unique/primary keys in table referenced by foreign keys**

Para forçar o apagamento da tabela, anulando as restrições de integridade necessárias, o comando a executar é:

DROP TABLE **dep** CASCADE CONSTRAINTS;

Objetos de Banco de Dados

Objeto	Descrição
Tabela	Unidade básica de armazenamento; composta de linhas
View	Representa logicamente subconjuntos de dados de uma ou mais tabelas
Seqüência	Gera valores numéricos
Índice	Melhora o desempenho de algumas consultas
Sinônimo	Fornece nomes alternativos a objetos

Além das tabelas, existem vários outros objetos em um banco de dados. Nesta lição, você conhecerá objetos como views, sequências, índices e sinônimos.

As views permitem exibir e ocultar dados de tabelas.

Muitas aplicações exigem o uso de números exclusivos como valores de chave primária.

Você pode criar um código na aplicação para atender a esse requisito ou usar uma sequência para gerar números exclusivos.

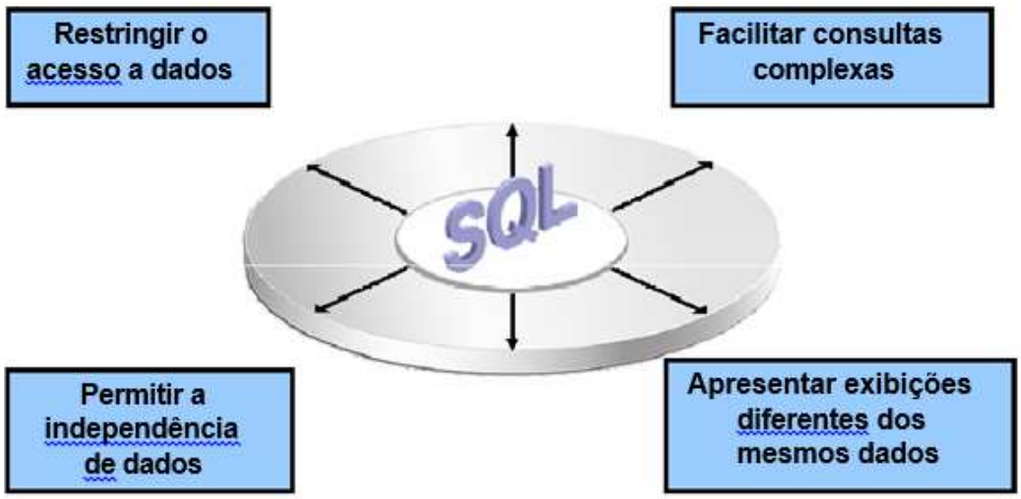
Para melhorar o desempenho de algumas consultas, considere a criação de um índice. Use índices também para impor exclusividade em uma coluna ou conjunto de colunas.

Você pode fornecer nomes alternativos a objetos usando sinônimos.

Criar Views (CREATE VIEW)

O Que É uma View?

Você pode apresentar combinações ou subconjuntos lógicos de dados criando views de tabelas. Uma view é uma tabela lógica baseada em uma tabela ou em outra view. Uma view em si não contém dados, mas é semelhante a uma janela por meio da qual é possível exibir ou alterar dados de tabelas. As tabelas nas quais uma view é baseada são denominadas tabelas-base. A view é armazenada como uma instrução SELECT no dicionário de dados.



Vantagens

- As views restringem o acesso aos dados, pois podem exibir colunas selecionadas da tabela.
- As views permitem fazer consultas simples para recuperar os resultados de consultas complicadas. Por exemplo, as views permitem aos usuários consultar informações de várias tabelas mesmo sem saber criar uma instrução de join.
- As views permitem a independência de dados a usuários ad hoc e programas aplicativos. É possível usar uma view para recuperar dados de várias tabelas.
- As views permitem o acesso de grupos de usuários a dados de acordo com critérios específicos.

Views Simples e Complexas

Existem duas classificações para views: simples e complexas. A diferença básica está relacionada às operações DML (INSERT, UPDATE e DELETE).

- Uma view simples é aquela que:
  - É derivada de dados de uma única tabela - Não contém funções ou grupos de dados
  - Permite a execução de operações DML
- Uma view complexa é aquela que:
  - É derivada de dados de várias tabelas - Contém funções ou grupos de dados
  - Nem sempre permite a execução de operações DML

Criando

```
CREATE [OR REPLACE] [FORCE|NOFORCE] VIEW view
  [(alias[, alias]...)]
 AS subquery
[WITH CHECK OPTION [CONSTRAINT constraint]]
[WITH READ ONLY [CONSTRAINT constraint]];
```

OR REPLACE	recria a view quando ela já existe
FORCE	cria a view independentemente da existência das tabelas-base
NOFORCE	só cria a view quando as tabelas-base existem (Este é o default.)
view	é o nome da view
alias	especifica nomes para as expressões selecionadas pela consulta da view (O número de apelidos deve corresponder ao número de expressões selecionadas pela view.)
subquery	é uma instrução SELECT completa (Você pode usar apelidos para as colunas na lista SELECT.)
WITH CHECK OPTION	especifica que apenas as linhas acessíveis à view podem ser inseridas ou atualizadas
Constraint	é o nome designado à constraint CHECK OPTION
WITH READ ONLY	garante que não seja possível executar operações DML na view

Exemplo 1:

```
CREATE OR REPLACE VIEW       salvu50
AS SELECT employee_id ID_NUMBER, last_name NAME,
salary*12 ANN_SALARY
FROM          employees
WHERE         department_id = 50;
View created.
```

Recuperando Dados de uma View

Você pode recuperar dados de uma view como faria com qualquer tabela. É possível exibir todo o conteúdo da view ou apenas linhas e colunas específicas.

```
SELECT * FROM salvu50;
```

Criando uma View Complexa

Crie uma view complexa que contenha funções de grupo para exibir valores de duas tabelas:

```
CREATE VIEW  dept_sum_vu
(name, minsal, maxsal, avgsal)
AS SELECT  d.department_name, MIN(e.salary),
MAX (e.salary),AVG(e.salary)
FROM  employees e, departments d
WHERE  e.department_id = d.department_id
GROUP BY  d.department_name;
View created.
```

Removendo uma View

Você pode remover uma view sem perder dados, pois uma view é baseada em tabelas subjacentes do banco de dados.

DROP VIEW salvu50;

Criar Sequências (CREATE SEQUENCE)

Uma sequência é um objeto que permite gerar números inteiros únicos. Mesmo que vários utilizadores (ou o mesmo utilizador através de várias sessões) acedam à sequência, é garantido que nenhum obtém um número que outro já tenha obtido.

O principal uso de sequências é para a geração automática de chaves primárias.

O comando para criar sequências é:

CREATE SEQUENCE **nome_sequência**
START WITH **num_principio**
INCREMENT BY **num_intervalo**
[MAXVALUE **num_max** | NOMAXVALUE]
[MINVALUE **num_min** | NOMINVALUE]
[CYCLE | NOCYCLE]
[ORDER | NOORDER]
[CACHE **num_cache** | NOCACHE]

Onde:

sequence	é o nome do gerador de sequência
INCREMENT BY n	especifica o intervalo entre os números da sequência, em que n é um inteiro (Se esta cláusula for omitida, a sequência será incrementada em 1)
START WITH n	especifica o primeiro número da sequência a ser gerado (Se esta cláusula for omitida, a sequência começará com 1.)
MAXVALUE n	especifica o valor máximo que a sequência pode gerar especifica o valor máximo 10^27 para uma sequência em ordem crescente e -1 para uma sequência em ordem decrescente (Esta é a opção default.)
NOMAXVALUE	
MINVALUE n	especifica o valor mínimo da sequência
	especifica o valor mínimo 1 para uma

NOMINVALUE	sequencia em ordem crescente e -(10^26) para uma sequência em ordem decrescente (Esta é a opção default.)

Criar uma sequência denominada DEPT_DEPTID_SEQ a ser usada para a chave primária da tabela DEPARTMENTS.

```
CREATE SEQUENCE dept_deptid_seq
INCREMENT BY 10
START WITH 120
MAXVALUE 9999
NOCACHE
NOCYCLE;
Sequence created.
```

- Não use a opção CYCLE.

CYCLE | NOCYCLE	especifica se a sequência continuará a gerar valores depois de atingir seu valor máximo ou mínimo (NOCYCLE é a opção default.)
CACHE n | NOCACHE	especificam quantos valores o servidor Oracle pré-aloca e mantém na memória (Por default, servidor Oracle armazena 20 valores no cache.)

Pseudocolunas NEXTVAL E CURRVAL

- NEXTVAL retorna o próximo valor disponível da sequência. Quando referenciada, mesmo que por usuários distintos, ela retorna um valor exclusivo.
- CURRVAL obtém o valor atual da sequência.
- É necessário executar NEXTVAL para a sequência antes que CURRVAL contenha um valor.

Regras de Uso de NEXTVAL e CURRVAL

Você pode usar NEXTVAL e CURRVAL nos seguintes contextos:

- Lista SELECT de uma instrução SELECT que não faz parte de uma subconsulta
- Lista SELECT de uma subconsulta em uma instrução INSERT
- Cláusula VALUES de uma instrução INSERT
- Cláusula SET de uma instrução UPDATE
-

Você não pode usar NEXTVAL e CURRVAL nos seguintes contextos:

- Lista SELECT de uma view
- Instrução SELECT com a palavra-chave DISTINCT
- Instrução SELECT com as cláusulas GROUP BY, HAVING ou ORDER BY.
- Subconsulta em uma instrução SELECT, DELETE ou UPDATE.
- Expressão DEFAULT em uma instrução CREATE TABLE ou ALTER TABLE

Utilizando uma Sequencia

Para usar as sequências basta fazer:

Nome_seq. nextval para obter o próximo valor (também funciona para o primeiro).
Nome_seq. currval para devolver o valor atual (não funciona a primeira vez).

O uso destas pseudo-colunas funciona como se tratasse de funções. Ou seja, pode-se incluir qualquer destas formas num comando SELECT, INSERT, UPDATE, etc. mas não se pode executar diretamente na linha de comando. Pode-se no entanto fazer uso da tabela especial DUAL.

Exemplo 1:

Usar uma sequência várias vezes consecutivas. Note que a primeira vez que se usa a sequência não se pode usar a forma currval (porque ainda não existe um valor corrente).

SQLWKS> SELECT **minha_sequencia.currval** FROM **dual;**
ORA-08002: **sequence** MINHA_SEQUENCIA.CURRVAL **is not yet defined in this session**
SQLWKS> SELECT **minha_sequencia.nextval** FROM **dual;**
NEXTVAL

----------1

1 row selected.

SQLWKS> SELECT **minha_sequencia.nextval** FROM **dual;**

NEXTVAL

----------2

1 row selected.

SQLWKS> SELECT **minha_sequencia.nextval** FROM **dual**;

NEXTVAL

----------3

1 row selected.

SQLWKS> SELECT **minha_sequencia.currval** FROM **dual**;

CURRVAL

----------3

1 row selected.

Apagar Sequências (DROP SEQUENCE)

Para apagar sequências usa-se o comando DROP SEQUENCE seguido do nome da sequência.

Exemplo:

DROP SEQUENCE **minha_sequencia***;*

Índices (INDEX)

Os índices são objetos de banco de dados que você pode criar para melhorar o desempenho de algumas consultas. Eles também podem ser criados automaticamente pelo servidor quando você cria uma constraint exclusiva ou de chave primária.

Um índice do servidor Oracle é um objeto de esquema que pode acelerar a recuperação de linhas usando um ponteiro. É possível criar índices de forma explícita ou automática. Se você não tiver um índice na coluna, ocorrerá uma varredura integral da tabela.

Um índice permite acesso direto e rápido a linhas de uma tabela. Seu objetivo é

reduzir a necessidade de entrada/saída de disco usando um caminho indexado para localizar os dados com rapidez. O servidor Oracle usa e mantém automaticamente o índice. Depois que um índice é criado, não é exigida qualquer atividade direta do usuário.

Os índices são independentes do ponto de vista lógico e físico da tabela que indexam. Isso significa que é possível criá-los ou eliminá-los a qualquer momento sem causar efeitos nas tabelas-base ou em outros índices.

Observação: Quando você elimina uma tabela, os índices correspondentes também são eliminados.

Tipos de Índice

É possível criar dois tipos de índice.

Índice exclusivo: O servidor Oracle cria automaticamente este índice quando você define uma constraint de chave PRIMARY KEY ou UNIQUE para a coluna de uma tabela. O nome do índice é aquele fornecido à constraint.

Índice não exclusivo: É um índice que um usuário pode criar. Por exemplo, você pode criar um índice de coluna FOREIGN KEY para uma join em uma consulta a fim de aumentar a velocidade de recuperação.

Observação: Você pode criar um índice exclusivo de forma manual, mas é recomendável criar uma constraint exclusiva, que gera implicitamente um índice exclusivo.

Criando um Índice

Crie um índice em uma ou mais colunas executando a instrução CREATE INDEX.

Na sintaxe:

Index é o nome do índice

Table é o nome da tabela

Column é o nome da coluna da tabela a ser indexada

CREATE INDEX **emp_last_name_idx**
ON **employees(last_name);**
Index created.

Diretrizes para criação de Índices

Crie um índice quando:	
	Uma coluna contiver uma grande faixa de valores
	Uma coluna contiver um grande número de valores nulos
	Uma ou mais colunas forem usadas em conjunto com freqüência em uma cláusula WHERE ou em uma condição de join
	A tabela for grande e for esperado que a maioria das consultas recupere menos de 2% a 4% das linhas da tabela
Não crie um índice quando:	
	As colunas não forem usadas com freqüência como uma condição na consulta
	A tabela for pequena ou for esperado que a maioria das consultas recupere mais de 2% a 4% das linhas da tabela
	A tabela for atualizada com freqüência
	As colunas indexadas forem referenciadas como parte de uma expressão

Removendo um Índice

Não é possível modificar índices. Para alterar um índice, elimine-o e, depois, recrie-o.

Remova uma definição de índice do dicionário de dados executando a instrução DROP INDEX. Para eliminar um índice, você precisa ser o proprietário dele ou ter o privilégio DROP ANY INDEX.

Na sintaxe, index é o nome do índice.

DROP INDEX emp_last_name_idx;
Index dropped.

Observação: Se você eliminar uma tabela, os índices e as constraints serão eliminados automaticamente, mas as views e as sequências permanecerão.

Sinônimos (SINONYM)

Os sinônimos são objetos de banco de dados que permitem chamar uma tabela

usando outro nome. É possível criar sinônimos para atribuir um nome alternativo a uma tabela.

Criando um Sinônimo para um Objeto

Para fazer referência a uma tabela pertencente a outro usuário, é preciso inserir o nome do autor dessa tabela seguido de um ponto como prefixo do nome da tabela. A criação de um sinônimo elimina a necessidade de qualificar o nome do objeto com o esquema e fornece um nome alternativo para uma tabela, uma view, uma sequência, um procedure ou outros objetos. Esse método pode ser útil especialmente com nomes longos de objetos, como views.

Na sintaxe:

PUBLIC cria um sinônimo acessível a todos os usuários
Synonym é o nome do sinônimo a ser criado
Object identifica o objeto para o qual o sinônimo é criado

Diretrizes

- O objeto não pode estar contido em um package.
- O nome de um sinônimo privado deve ser diferente do de outros objetos pertencentes ao mesmo usuário.

Crie um nome abreviado para a view DEPT_SUM_VU:

CREATE SYNONYM d_sum
FOR dept_sum_vu;
Synonym Created.

Elimine um sinônimo:

DROP SYNONYM d_sum;
Synonym dropped.

Instrução DCL

Concede ou revoga direitos de acesso ao banco de dados Oracle e às estruturas contidas nele. Conhecida coletivamente como **DCL (Data Control Language)** ou Linguagem de Controle de Dados**.**

Comando GRANT

Permite conceder privilégios aos objetos do banco de dados.
A sintaxe do para conceder privilégios é:
GRANT [PRIVILEGES] ON [object] TO [USER];

Exemplo:

```
GRANT SELECT ON cliente TO usuario_teste;
```

A instrução indica que estou dando privilegio de SELECT na tabela cliente para (TO) o usuario_teste;

Comando REVOKE

Utilizado para revogar privilégios concedidos anteriormente.

No caso, vamos revogar os privilégios do usuário "usuario_teste", então, utilizaremos o comando abaixo:

```
REVOKE SELECT ON cliente FROM usuario_teste;
```

A instrução indica que retiramos privilegio de SELECT na tabela cliente do usuario_teste;

Bibliografia

- Oracle University – Banco de Dados Oracle 10 g – Fundamentos de Banco de Dados I e II
- Curso de Oracle – Treina Web
- Pesquisas elaboradas na Internet.

CURSO TÉCNICO EM

INFORMÁTICA

ETAPA 3

AMBIENTE OPERACIONAL III - LINUX

Sumário

1- INTRODUÇÃO 109

2 - UMA BREVE HISTÓRIA DO LINUX 109
2.1 A EVOLUÇÃO DO LINUX 111
2.2 DETALHAMENTO DO LINUX 113

3 - SOFTWARE LIVRE 114
3.1 – LINUX – DISTRIBUIÇÃO 115
3.2 - O UBUNTU 116

4 - NA DUVIDA? – LOCAIS DE BUSCA, COMUNIDADES E SITES 117

5 – INTERFACES 119

6 - TESTANDO E CONHECENDO O UBUNTU 122
6.1 MENU DE CONTEXTO 133
6.2 O MOUSE 135

7 - INSTALAÇÃO, BACKUP E RESTAURAÇÃO 139

8 - CONTROLE DO SISTEMA 149
8.1 – CENTRAL DE PROGRAMAS DO UBUNTU 149

9 – TERMINAL 160

10 - ESCRITÓRIO E PRODUTIVIDADE 164

11 - SISTEMA DE ARQUIVOS E PERMISSÕES 166

12 - PRINCIPAIS COMANDOS DO LINUX 172
12.1 – USUÁRIOS 174
12.2 – GRUPOS 175

13 - GERENCIAMENTO DE PONTOS DE MONTAGEM 176
13.1. FHS – O padrão hierárquico do sistema de arquivos do GNU/Linux 176
13.2 –Pontos de montagem: diretórios "recipientes" 177
13.3 – Identificando discos e partições para montagem: o utilitário blkid 178
13.4 –Montagens manuais de discos e partições: o comando mount 180
13.5 – Desmontagem de discos e partições: o comando umount 183

14 - SMB - Server Message Block 184

15 - INTERPRETADOR DE COMANDOS 191

16 - ENTENDENDO E CONFIGURANDO O FIREWALL (SHOREWALL) 200
16.1 Configurando as regras do firewall 202
16.2 - Organização do arquivo de regras 202
16.3 - Regas do firewall 203
16.4 - Redirecionamento e Proxy Transparente 205
16.5 - Acesso às estações da rede local 206
16.6 - Ativando as alterações 207

BIBLIOGRAFIA SUGERIDA 209

Disciplina: Ambiente Operacional III

Carga horária semanal: 3

Carga horária total: 60

Objetivos Gerais:

- Apropriar dos conhecimentos acerca do Sistema Operacional – Linux – e aplicá-los no desenvolvimento de suas funções.

Avaliação:

1º Bimestre

Critérios de distribuição de pontos

Atividades	Valor
Exercicios	5
Pratica	5
prova	10
Avaliação bimestral	20

2º Bimestre

Critérios de distribuição de pontos

Atividades	Valor
Exercicio	5
Prática	5
Seminário Tecnologia	20
Avaliação bimestral	30

Total de pontos: 100

1. INTRODUÇÃO

O Linux é possivelmente um dos mais importantes software livre já produzidos. Ele evoluiu para um sistema operacional completo para negócios, educação e uso pessoal. O Linux não esta mais restrito ao universo de programadores (embora muitos de seus usuários se enquadrem neste grupo).

O Linux é um sistema operacional UNIX que executa em uma variedade de plataformas, principalmente computadores pessoais. Ele suporta uma vasta gama de softwares e compilador GNU, e o TCP/IP. É uma implementação versátil e confiável do UNIX, distribuída livremente nos termos da Licença Geral de Uso GNU (GNU General Public. License).

O Linux pode transformar qualquer computador 386 ou superior em uma estação de trabalho que coloca em suas mãos todo o poder e flexibilidade do UNIX. Empresas instalam Linux em suas redes de computadores, e usam o sistema operacional para gerenciar sistemas financeiros e hospitalares, ambientes de computação distribuída e telecomunicações. Universidades do mundo inteiro utilizam o Linux para ministrar cursos de definição e implementação de sistemas operacionais. Entusiastas por computadores usam o Linux em casa para programação, como ferramenta de produtividade, e uma infinidade de outros usos.

O que torna o Linux tão diferente de outras opções de sistemas operacionais é que ele é uma implementação gratuita e de código aberto. Ele foi e ainda está sendo desenvolvido cooperativamente por um grupo de voluntários, principalmente através da Internet, que trocam códigos, relatam “bugs” e resolvem problemas em um ambiente aberto. Todos são bem vindos ao esforço de desenvolvimento do Linux. Tudo que é necessário é o interesse em mexer em uma implementação do UNIX e algum conhecimento de programação.

2 – UMA BREVE HISTÓRIA DO LINUX

O Linux evoluiu a partir do UNIX, cuja história começou em 1965, quando se formou um grupo de programadores, que incluía Ken Thompson, Dennis Ritchie, Douglas McIlroy e Peter Weiner, em um esforço conjunto da AT&T (Laboratórios Bell), General Electric (GE) e MIT (Massachussets Institute of Technology) para o

desenvolvimento de um sistema operacional chamado Multics O Multics seria um sistema de compartilhamento de tempo para uma comunidade de usuários. Porém, o computador GE 465, que era o recurso computacional disponível na época, foi insuficiente para os objetivos do projeto. Em 1969, a Bell retirou-se do projeto, alegando que três instituições com objetivos distintos dificilmente alcançariam uma solução satisfatória para cada uma delas, e os participantes sofriam da síndrome do segundo projeto e, por isso, queriam incluir no Multics tudo que tinha sido excluído dos sistemas experimentais até então desenvolvidos.

Ken Thompson, no mesmo ano, usou um computador DEC PDP-7 para reescrever o Multics emum contexto menos ambicioso, com ideias e apoio de Rudd Canaday, Doug McIlroy, Joe Ossanna e Dennis Ritchie. Usou linguagem de máquina e chamou o sistema de Unics. O sistema de propósito geral e compartilhamento de tempo eram confortáveis o suficiente para atrair o interesse dos usuários e credibilidade para a aquisição de uma máquina maior, um PDP-11/20. Ritchie também escreveu um compilador para a linguagem de programação C. Brian Kernighan batizou o sistema de UNIX.

Em 1973, Dennis Ritchie e Ken Thompson reescreveram o UNIX na linguagem C, para um computador PDP-11, quebrando a tradição de que o software do sistema deveria ser escrito em linguagem assembly.

A linguagem C havia sido desenvolvida por Ritchie para substituir a linguagem B, desenvolvida por Thompson. O seu uso tornou o sistema portável e é uma das razões para a rápida difusão do UNIX.

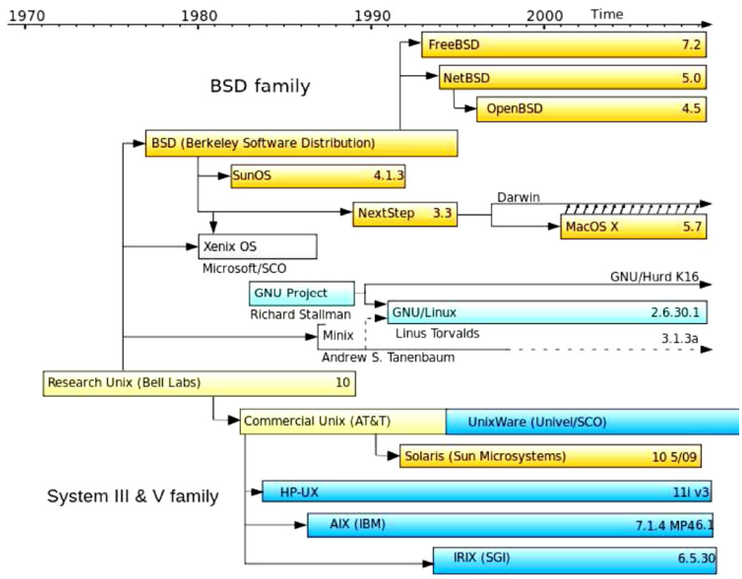
Ao longo dos anos 1970 e 1980 foram desenvolvidas as primeiras distribuições profissionais, como os sistemas BSD, na Universidade de Berkeley na Califórnia, e o System III e V, no Laboratório Bell.

Em 1977, a AT&T começou a fornecer o UNIX para instituições comerciais, graças ao esforço de Peter Weiner, cientista da Universidade de Yale e fundador da Interactive System Corporation. Weiner conseguiu da AT&T uma licença para portar e comercializar o UNIX para o computador Interdata 8/32, usado em automação de escritório. Com a crescente oferta de microcomputadores, outras empresas portaram o UNIX para outras máquinas.

De 1977 a 1981, a AT&T integrou variantes no primeiro sistema UNIX comercial, chamado de System III. A primeira instalação do UNIX no Brasil foi feita na UFPB, em dezembro de 1980, em um PDP11/34, gerenciado pelo Prof. Jacques

Sauvé, do Departamento de Sistemas de Computação. O autor foi um dos primeiros usuários do sistema. A versão instalada na ocasião foi a Research Unix, Version 6. Embora a versão 7 existisse, não foi instalada, porque tinha sido desenvolvida para o PDP11/45, que tinha um espaço de endereçamento maior. O PDP 11/34 só podia rodar a V6 porque a V7 não cabia.

Em 1983, após melhorar o System III, a AT&T apresentou o UNIX comercial, chamado de System V. Atualmente, o UNIX System V é o padrão internacional no mercado UNIX. Há muitos sistemas desenvolvidos a partir de padrões como o Portable Operating System Interface (POSIX). Alguns dos sistemas operacionais derivados do UNIX são: BSD (FreeBSD, OpenBSD e NetBSD), Solaris (antes conhecido como SunOS), IRIXG, AIX, HP-UX, Tru64, SCO, Linux e o Mac OS X (baseado no núcleo Mach BSD chamado Darwin). Existem mais de quarenta sistemas operacionais derivados, rodando em celulares ou supercomputadores, relógios de pulso e sistemas de grande porte.



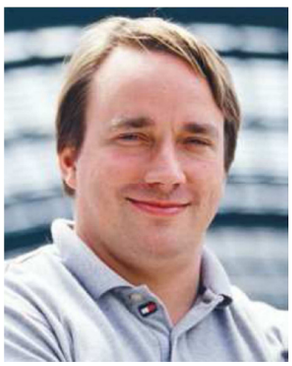
A evolução histórica do UNIX até o Linux está mostrada na Figura acima.

2.1 A Evolução do Linux

O núcleo Linux foi escrito por Linus Torvalds, do Departamento de Ciência da Computação da Universidade de Helsinki, na Finlândia, com a ajuda de muitos

programadores voluntários por meio da Usenet, um sistema de listas de discussão existente desde os primórdios da Internet.

Linus Torvalds começou o desenvolvimento do núcleo inspirado por seu interesse no Minix, um pequeno sistema UNIX desenvolvido por Andrew S. Tanenbaum (5). Ele criou, como disse “a better Minix than Minix”. Após trabalhar algum tempo no projeto, enviou a seguinte mensagem para comp.os. minix: “Você suspira pelos bons tempos do Minix-1.1, quando os homens eram homens e escreviam seus próprios device drivers? Você está sem um bom projeto em mãos e deseja trabalhar em um sistema operacional que possa modificar de acordo com as suas necessidades? Acha frustrante quando tudo funciona no Minix? Chega de noite ao computador para conseguir que os programas funcionem? Então esta mensagem pode ser exatamente para você. Como eu mencionei, há um mês, estou trabalhando em uma versão independente deum sistema operacional similar ao Minix para computadores AT-386. Ele está, finalmente, próximo do estado em que poderá ser utilizado (embora possa não ser o que você espera),e eu estou disposto a disponibilizar o código-fonte para ampla distribuição. Ele está na versão 0.02... contudo eu tive sucesso ao executar bash, gcc, gnu-make, gnu-sed,compress etc. nele.”

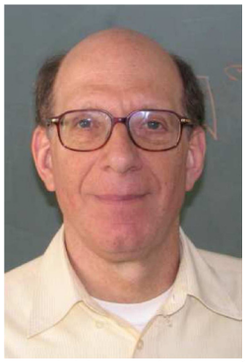


Linus Torvalds desenvolveu o núcleo Linux inspirado por seu interesse no Minix

O nome Linux foi criado por Ari Lemmke, administrador do site ftp.funet.fi que deu esse nome ao diretório FTP no qual o núcleo Linux estava disponível. Antes disso, Linus havia nomeado núcleo “Freax”.

Em 5 de outubro de 1991, Linus Torvalds anunciou a primeira versão “oficial” do núcleo Linux, versão 0.02. Muitos programadores atenderam ao seu apelo, e têm ajudado a fazer do Linux um dos melhores sistemas operacionais existentes. No

início era utilizado por programadores ou por quem tinha conhecimento de programação e usava linhas de comando. Atualmente existem muitas empresas que produzem os ambientes gráficos, as distribuições mais amigáveis, de forma que todos conseguem usar o Linux.



O Minix foi desenvolvido por Andrew S. Tanenbaum

2.2 Detalhamento do Linux

O termo Linux refere-se ao núcleo (kernel) do sistema operacional, mas também é usado pelos meios de comunicação e usuários para indicar os sistemas operacionais baseados no núcleo Linux agregado a outros programas. Segundo Tanenbaum e Silberschatz, um núcleo pode ser considerado o próprio sistema operacional, quando ele é definido como um gerenciador de recursos de hardware.

O Linux é um núcleo monolítico, no qual as funções do núcleo (escalonamento de processos, gerenciamento de memória, operações de entrada e saída, acesso ao sistema de arquivos) são executadas no espaço de núcleo. Uma característica do núcleo Linux é que algumas das funções (drivers de dispositivos, suporte à rede, sistema de arquivos, por exemplo) podem ser compiladas e executadas como módulos (loadable kernel modules), que são bibliotecas compiladas separadamente da parte principal do núcleo e podem ser carregadas e descarregadas após o núcleo estar em execução.

Linus Torvalds não pretendia fazer do Linux um sistema portável, mas ele evoluiu assim. Torvalds imaginava um tipo diferente de portabilidade, relacionada com a facilidade de compilar aplicações de uma variedade de códigos-fonte no sistema. Assim, o Linux tornou-se popular em parte devido ao esforço para que os

códigos-fonte, do tipo General Public License (GPL), rodassem nele. Hoje o Linux funciona em dezenas de plataformas.

3 – SOFTWARE LIVRE

O que define o software livre e o software pago é a Licença de Uso. Os programas pagos se baseiam em contratos de direito de cópia (Copy Rights, tendo um símbolo próprio para representá-lo), normalmente limitam o número de máquinas em que o software pode ser instalado, as obrigações do proprietário e até período de validade do suporte. Os programas livres normalmente trabalhão sobre a licença GPL (General Public Licence), que tem como principal produto o GNU/Linux.

Freeware não é um software livre, é vasta a lista de programas disponibilizados gratuitamente na Internet, mas existem detalhes sobre a licença de utilização destes softwares que precisam ser levadas em conta.

Um grande equivoco é achar que software freeware é o mesmo que livre. Na verdade, estes programas realmente disponibilizam a utilização sem custos, mas as suas limitações na utilização se assemelham mais as licenças CopyRight que as Livres.

Alguns exemplos de limite são:

- Não é permitido fazer alterações no código;
- A finalidade de utilização pode ser restringida, por exemplo, não profissional;
- O número de licenças é restringido.

Diferentemente do Livre que normalmente utiliza alguns tipos de licenças registrados e bem definidos, pode possuir licença única. Registrada exclusivamente pelo desenvolvedor.

A Free Software Foundation definiu quatro "fundamentos" como base para softwares livres:

- Liberdade de executar o programa, para qualquer propósito (liberdade 0);
- Liberdade de estudar como o programa funciona e adaptá-lo às suas necessidades (liberdade 1), sendo o acesso ao código-fonte* um pré-requisito para este aspecto;

- Liberdade de distribuir cópias de forma que você possa ajudar ao seu próximo (liberdade 2);
- Liberdade de melhorar o programa e liberar os seus aperfeiçoamentos, de modo que toda a comunidade se beneficie (liberdade 3). Novamente, aqui o acesso ao código-fonte é um pré-requisito.

3.1 – LINUX – Distribuição

Ao contrário dos sistemas operacionais comerciais, o Linux é completamente gratuito. Porém, o fato de ser gratuito, não significa que seja de domínio público. Tanto o Kernel, quanto a maioria dos aplicativos para Linux são distribuídos sob uma licença especial, chamada de GPL. A GNU especifica que o programa pode ser distribuído livremente, e até mesmo alterado, porém, devem ser mantidos os créditos ao autor original, assim como podem ser incluídos créditos para quem vier a contribuir no desenvolvimento do programa.

A existência da GNU protege quem trabalha no desenvolvimento dos programas, ao mesmo tempo em que protege os próprios usuários. Não existe um "dono" do Linux, existem sim "autores do Linux", isto significa que jamais este sistema poderá vir a se tornar um sistema comercial, como o Windows por exemplo.

Você pode perguntar então: "-Ok, mas eu já vi várias caixas com CDs do Linux, sendo vendidas em lojas, como por exemplo, o Linux da Conectiva, que é vendido, como fica a GNU nessa história?".

Como disse, o Linux pode ser distribuído livremente. Você pode inclusive gravar um CD com o Linux e mais alguns programas e vendê-lo para quem se interessar. Isto é o que chamamos de "distribuição". Significa que você pode até mesmo ganhar dinheiro vendendo CDs do Linux, mas não pode estabelecer nenhum tipo de restrição de uso, inclusive contra cópias. Ao comprar apenas um CD, você pode instala-lo em quantas máquinas quiser, e até mesmo copia-lo e distribuir para amigos, sem incorrer em pirataria. Este é o significado de "software livre". Você pode cobrar pela gravação do CD, pelos manuais, pelo suporte técnico, mas o software em sí continua sendo de livre distribuição.

A diferença entre comprar uma distribuição do Linux, e simplesmente baixar o sistema pela Internet, é que você tem a comodidade de receber CDs com vários

aplicativos, manuais, suporte técnico etc. coisas que você não teria caso fosse instalar na unha.

Existem varias distribuições de Linux. As principais diferenças entre as distribuições podem ser encontradas nos sistemas de pacotes, nas estruturas dos diretórios e na biblioteca básica.

3.2 O UBUNTU

O Ubuntu é um sistema operacional colaborativo (uma das distribuições), mas o comando por trás do sucesso deste sistema está nas mãos de uma empresa: a Canonical, que tem sua sede na Ilha de Man, na África do Sul.

Você pode adquiri-lo de várias formas, são elas:

- Fazendo o download da última versão pela web.
- Solicitando o seu CD via ShipIt no site do Ubuntu
- Em revistas especializadas em Linux que trazem junto CDs.
- Com algum usuário de Ubuntu.
- Em eventos da comunidade ou eventos de software livre em geral, que sempre acontecem nos mais diferentes lugares do Brasil e no mundo.

Por ser livre e não existir pirataria, é perfeitamente aceitável você conseguir com um amigo uma cópia do sistema ou cedê-lo uma cópia do Ubuntu, disseminando assim a cultura Open Source.

Se você possui conhecimento em programação, tem opções fantásticas à sua frente, como por exemplo, a opção de personalização do Ubuntu, modelando-o ao seu gosto.

Versões LTS do Ubuntu são lançadas a cada dois anos e normalmente seu suporte é de 36 meses, contudo, o Ubuntu 12.04 será um LTS diferente: 5 anos! Outro detalhe importante na hora de migrar para uma nova versão é lembrar que além do suporteextendido, LTS's são mais estáveis do que as versões não-LTS's.

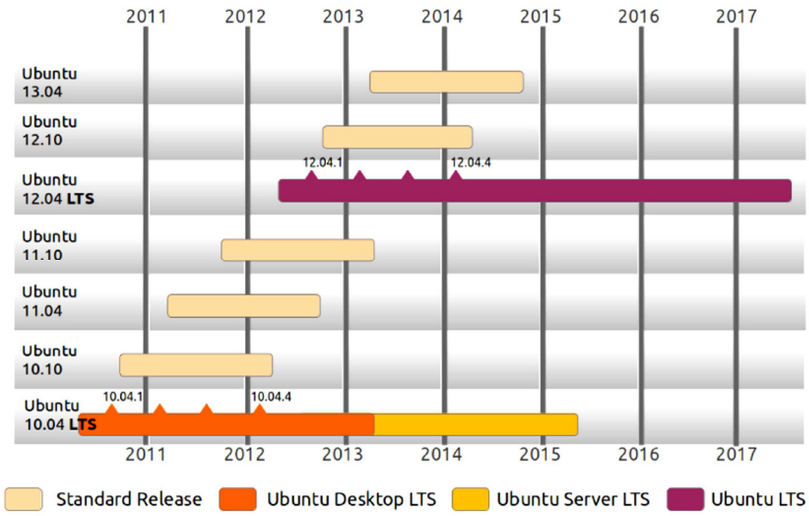


Gráfico mostrando o tempo de suporte das versões do Ubuntu

4 – NA DUVIDA? – LOCAIS DE BUSCA, COMUNIDADES E SITES

Comando com: man comando

Consulta os manuais dos comandos do sistema.

whatis.

Consulta manuais do sistema a partir de um banco de dados. Esse banco de dados deve ser atualizado periodicamente com o comando makewhatis (somente com o usuário root). O whatis não tem página do manual. Para obter ajuda, digite: makewhatis -?

Sintaxe: whatis palavra-chave

Exemplo:

$ whatis find

Apropos

Exibe informações sobre um assunto a partir de um banco de dados, é o mesmo do comando whatis.

Sintaxe: apropos palavra-chave

Exemplo:

$ apropos directory

Obs.: a diferença entre esses dois comandos, é que o whatis precisa do nome exato do comando e o apropos pode colocar o nome do comando ou o que ele faz.

Exemplo: ele vai puxar todas as informações sobre comandos que utilizam mails:

$ apropos mail

Sintaxe: **man [opções] [seção] comando**

Opções:

-a: exibe todas as páginas do manual.

-h: mensagem de ajuda.

-w: exibe a localização de páginas do manual a serem exibidas.

Seção:

bin: binários essenciais para o funcionamento do sistema.

sys :: chamadas do sistema.

lib :: funções das bibliotecas.

dev :: arquivos de dispositivos (ex.: HD, pendrive, placa de vídeo, impressora, etc).

etc :: arquivos de configuração.

games

misc :: miscelânea.

sbin :: binários essenciais à administração e manutenção do sistema (root).

boot :: kernel.

Exemplos:

Informações sobre o comando ls

$ man 5 ls

Exibe informações sobre o arquivo de configuração do "rpc-portmap":

$ man 5 rpc

Endereços:

Comunidade Ubuntu Brasil

http://ubuntu-br.org/comunidade

Lista de discussão do Ubuntu Brasil

- **ubuntu-br@lists.ubuntu.com**

Fórum para países de Língua Portuguesa

- **http://forum.ubuntubrasil.org**

Planeta Ubuntu Brasil

- **http://planeta.ubuntubrasil.org**

Kubuntu Brasil

- **http://www.kubuntu-br.org**

Ubuntu Games

- **http://www.ubuntugames.org**

5– INTERFACES

O sistema operacional Linux possui dois tipos de interfaces para a interação com o usuário:

• CLI: interface em modo texto, por linha de comando.

• GUI: interface gráfica, em que se usa o mouse para clicar em ícones e atalhos.

É importante notar que em alguns programas CLI e GUI ambas estão integradas. Exemplo:Matlab.

CLI - o modo texto

A interface de linha de comando ou CLI ("Command Line Interface") permite o acesso no modo texto, ou seja, possibilita ao usuário interagir com o sistema digitando comandos para realizar tarefas específicas em uma linha de comando para comandar as mais variadas tarefas no sistema.

Originou-se na época em que máquinas teletipo estavam ligadas a computadores. É a principal forma de controle remoto e gerenciamento do sistema. Nessa interface se digita o comando, que é imediatamente executado ao pressionarmos [Enter] no teclado. Entretanto, os comandos devem ser digitados perfeitamente, pois a sintaxe é rígida: o comando só funciona se estiver escrito perfeitamente, sem nenhum erro. Como há muitos comandos possíveis é um modo mais difícil para os que têm pouco conhecimento de Informática. Mas, certas tarefas só podem ser executadas com o uso de comandos digitados. E, também, em determinado momento, o modo gráfico pode não estar disponível para a inserção de comandos. Nessas situações será necessário conhecer alguns comandos do Linux.

É importante ressaltar que, dependendo da distribuição utilizada, um ou outro comando pode não estar disponível. E, também que alguns comandos só podem ser

executados por usuários com privilégios de administrador. Mas, um bom conhecimento dos comandos dá poder e controle total sobre o sistema operacional.

Note-se que é possível usar uma linha de comando mesmo estando em um ambiente gráfico. A tela onde se escreve é chamada de Terminal (ou Consola)

1. Na interface GNOME, clicar em Aplicativos - Acessórios - Terminal.
2. Se a interface KDE estiver sendo utilizada, pode-se usar o aplicativo "KDE Terminal".

Em algumas distribuições existe o modo texto. Entretanto em outras, como o Ubuntu, essa interface é pouco necessária, mas, se for preciso, o modo texto está disponível.

GUI - Interfaces gráficas

O linux é capaz de oferecer interação com o usuário por meio de atalhos e ícones, permitindo utilizar a maioria de seus recursos por meio de cliques do mouse.

A interface gráfica, também chamada de ambiente gráfico, desktop ou GUI ("Graphical UserInterface") começou a ser desenvolvida por Douglas C. Engelbart, o inventor do mouse, ainda na década de 60.

O modo gráfico possibilita a utilização do mouse, para clicar em botões, opções, menus ou figuras, para dar comandos ao sistema.

Como a variedade é característica do ambiente Linux, muitas interfaces gráficas foram desenvolvidas:

GNOME, KDE, XFCE, Windowmaker, Fluxbox, Blanes, Blackbox, Afterstep, Enlightenment, IceWM, FVWM, Kahakai, ZappWM, dwm, SithWM, Whim, Karmen, Sawfish, XIGE, Framer, Mavosxwm, WindowLab, OpenBox...

Entretanto, KDE e GNOME são os dois ambientes gráficos mais comumente usados e dividem a preferência dos usuários e das distribuições. Ambos rodam sobre o X, que cuida do acesso à placa de vídeo, ao teclado, ao mouse e a outras funções "base", enquanto a interface gráfica comanda o que se vê na tela.

GNOME

A interface GNOME ("GNU Network Object Model Environment"), proposta por Miguel de Icaza e Federico Mena em agosto de 1997, é leve e prática, sendo a preferida por muitos, devido à facilidade em usar.

O GNOME é baseado na biblioteca GTK, do Projeto GNU e sua licença é a GNU General Public License (GPL).

É importante lembrar que foi o primeiro projeto a oferecer um desktop completo, gratuito e inteiramente livre para sistemas baseados em Unix.

Conta com uma coleção rica de ferramentas, bibliotecas e de componentes para desenvolvimento de aplicações. E inclui uma suíte para escritório, por meio de projetos independentes: processador de texto (AbiWord), folha de cálculo (Gnumeric), gestão de projetos (Planner), editor de diagramas (Dia), programa para desenhos vetoriais (Inkscape) e de imagem (GIMP).

KDE

A interface KDE ("K Desktop Environment") apresenta uma infinidade de opções de configuração possíveis.

É um programa de origem alemã, sendo um ambiente gráfico desenvolvido com base na biblioteca Qt, que inclui um gerenciador de janelas e uma plataforma de desenvolvimento livre e de código aberto.

O gerenciador de janelas é o KWin e fornece uma interface gráfica organizada e consistente tanto para usar aplicativos específicos quanto funções básicas como manipulação de arquivos e dispositivos. Tem uma barra de tarefas intuitiva semelhante à do Windows, minimizando o contraste da interface ao conhecido em outros sistemas operacionais.

Um possível problema é que o KDE é desenvolvido com base na biblioteca gratuita para aplicações Open Source, Qt, que é de propriedade da empresa Trolltech. As bibliotecas do KDE são licenciadas como LGPL ou BSD-compatível. O licenciamento prevê que quem desenvolver aplicações de código fechado com Qt seja obrigado a comprar uma licença.

Entretanto, o KDE tem um framework com uma IDE multi-linguagem, a KDevelop, e várias tecnologias como KParts, KIO, DCOP, KHTML, e um potente conjunto de bibliotecas.

O KDE inclui uma suíte de escritório simples, mas com muitos componentes: processador de texto (KWord), folha de cálculo (KSpread), programa de apresentações (KPresenter), editor de diagramas gráficos (KChart), de equações (KFormula), gerador de relatórios (Kugar), editor de diagramas e fluxogramas (Kivio), base de dados (Kexi), editor de gráficos bitmap (Krita) e vetoriais (Karbon14).

6 – TESTANDO E CONHECENDO O UBUNTU

É possível testar o Ubuntu antes de instalá-lo em seu computador. O Ubuntu utiliza uma tecnologia chamada live-cd, que permite que você rode o sistema diretamente do CD ou do pen drive antes de instalá-lo definitivamente em seu computador ou notebook.

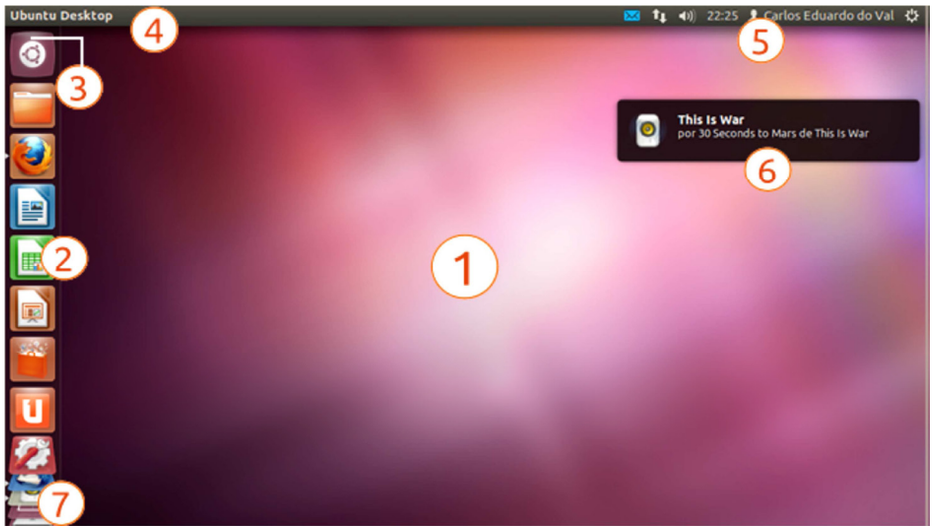
Ao inicializar o Ubuntu diretamente do CD ou do pen drive você terá algumas funcionalidades limitadas e a velocidade de execução do sistema e dos aplicativos não pode ser comparada com a velocidade do sistema instalado diretamente no computador

O ambiente de trabalho

O ambiente de Desktop do Ubuntu traz uma interface leve, atraente e voltada para a produtividade. Criado pela Canonical, o tema Unity foi uma inovadora proposta na ocasião do lançamento da versão 10.04 do Ubuntu Netbook Edition, uma versão do Ubuntu desenhada para aproveitar melhor as pequenas telas dos netbooks, contudo, a simplicidade, leveza e beleza do Unity foi trazida à versão Desktop do Ubuntu, e já na versão 11.04, fazia parte do sistema.

O Ubuntu 12.04, além do Unity melhorado, traz o GNOME 3 como gerenciador de janelas.

O Unity é uma interface dinâmica e atraente, que traz rapidez na execução de tarefas diárias e integra-se aos aplicativos de forma harmônica.



A área de trabalho, também chamada de “Desktop”.

1) **Ubuntu Desktop**: A tela inicial do sistema.

2) **Lançador**: Agrega os principais programas aplicativos do sistema ou de escolha do usuário, além de funcionar como barra de tarefas.

3) **Painel Inicial:** Exibe um menu contendo os programas, pastas e arquivos pessoais do usuário, além de possuir um campo de busca.

4) **Barra superior:** Mostra o nome e os menus dos aplicativos em primeiro plano, também chamados de Menu Global.

5) **Indicadores:** Exibe alguns indicadores de aplicativos e de status, como indicador de nova mensagem, bluetooth, wireless, menu de áudio, hora e menu de configurações do sistema.

6) **Notificações do sistema:** Informa algum evento, como nova mensagem de e-mail, mensageiro instantâneo, troca de música, atualizações de sistema, entre outras.

7) **Lixeira:** Pasta onde os arquivos excluídos ficam armazenados até que sejam definitivamente excluídos do sistema, ou até que o sistema necessite do espaço ocupado pelos arquivos dentro dela.

Tecla Super

Antes de começar a falar das funcionalidades do Ubuntu, vamos abrir um parêntese para explicar uma confusão muito comum: a tecla **Winkey**.

Aquela tecla, que fica normalmente entre as teclas **CTRL** e **Alt** na maioria dos teclados, e em alguns, entre as teclas **Fn** e **Alt**, é muito conhecida por **Winkey.** Isso porque elas, normalmente, vêm com o logo do Windows desenhado, e por isso, ganhou o nome de "Tecla Windows", contudo, como o mundo da informática não se resume à Microsoft e ao Windows, no Ubuntu e nos outros sistemas Linux, ela ganhou o nome de **"Super".**

Painel Inicial

O Painel Inicial, também conhecido por "Dashboard", ou "Dash", é um substituto para os menus presente nos demais sistemas. O Painel é integrado ao painel superior (barra do superior) do Ubuntu e ao Lançador.

Para ter acesso ao Painel, você pode clicar no primeiro ícone do Lançador, identificado com o logo do Ubuntu, ou pressionar a tecla Super (tecla com o logo do Windows) em seu teclado.

Através do Painel você pode encontrar documentos, fotos, músicas, vídeos, aplicativos e acessórios clicando na categoria relacionada, ou apenas digitando o nome do item desejado no campo de busca.



O Painel Inicial do Ubuntu

1) **Ícone de acesso**: Abre o Painel Inicial.
2) **Campo de busca**: Efetua busca de aplicativos e arquivos do computador relacionados ao conteúdo digitado em tempo real.
3) **Filtrar resultados:** Permite a filtragem de acordo com o tipo de arquivo ou programa pesquisado, e muda seu contexto conforme o tipo de busca, assim, ao procurar por arquivos, será possível definir filtros de tamanho, data de modificação ou tipo.
4) **Aplicativos usados com maior frequência**: Exibe os programas mais usados. Para ver os outros resultados, clique na opção “Ver mais x resultados”.
5) **Aplicativos instalados:** Mostra todos os programas instalados em seu computador. Para vê-los, clique em “ver mais x resultados”.
6) **Disponíveis para download**: Mostra os aplicativos disponíveis para instalação. É sensível à busca no campo “Pesquisar aplicativos”, assim, ao digitar o nome de um programa, será exibido sugestões de programas similares.
7) **Lentes**: Mini aplicativos de busca, exibição de conteúdo e outros, que podem ser adicionados. O sistema possui quatro Lentes padrão: Início, Aplicativos, Arquivos e Músicas. A Lente de músicas, por exemplo, integra-se ao aplicativo padrão de áudio e escaneia toda sua biblioteca musical.

Lançador

O Lançador, também conhecido por “Launcher”, juntamente com o Painel é uma melhoria muito bem-vinda no Ubuntu, pois une a facilidade de manter os ícones de seus programas mais usados por perto, e funciona como uma barra de tarefas, mantendo minimizados os aplicativos abertos.

O Lançador identifica os aplicativos abertos com uma pequena seta do lado direito do ícone (ícone do navegador de internet Mozilla Firefox). Seta do lado esquerdo indica que o programa, além de estar em execução, está em primeiro plano (ícone da Pasta pessoal). A quantidade de setas à esquerda do ícone indica a quantidade de instâncias do mesmo aplicativo. Caso mais de uma instância esteja aberta, clique no ícone pausadamente duas vezes para ver a exposição de suas janelas.

O comportamento padrão do Lançador é fixo à esquerda da tela, contudo, conforme será visto adiante, é possível alterar este estado “fixo” para “escondido até que o mouse encoste-se à lateral esquerda” através das configurações de sistema.

Além dos ícones de aplicativos inclusos no Ubuntu, o Lançador traz o ícone de acesso à Lixeira, à pasta pessoal (home, que será abordada mais adiante), ao Alternador de espaços de trabalho (Ubuntu workspaces) e aos dispositivos de armazenamento (HDs e CD/DVDs) montados.

Os ícones presentes no Lançador podem ser reorganizados de acordo com o gosto pessoal de cada usuário, arrastando-o até a posição desejada. Também podem ser removidos através do clique com botão direito do mouse e desmarque a opção “Manter no lançador”.

Incluir um aplicativo no Lançador pode ser feito arrastando seu ícone de dentro do Painel até o Lançador, ou clicando no ícone de um aplicativo aberto com o botão direito do mouse e selecionando a opção “Manter no lançador”.

No Lançador, a posição dos dez primeiros ícones é associada a números em ordem de 1a 9 e 0 e você pode abri-los ao manter pressionada a tecla Super e pressionar o número correspondente ao item numerado no Lançador.

O clique direito do mouse permite a você interagir com alguns programas diretamente pelo Lançador através de “quicklists”. Ao abrir um aplicativo que não está no Lançador, você pode travá-lo ao selecionar a opção correspondente, assim, você pode manter sempre à mão os aplicativos que usa com maior frequência.

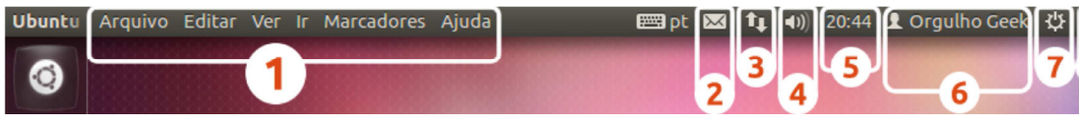


Quicklists

Painel Superior

O painel superior do Ubuntu é um agregador de indicadores e mini aplicativos do Ubuntu. Aplicativos que fazem uso da do painel superior, quando executados, exibem um pequeno ícone à direita do painel.

Além de exibição de mini aplicativos, o painel superior traz o menu de aplicativo, atalhos para os aplicativos padrão do Ubuntu, como mensagens e e-mail, gerenciamento de bluetooth e interfaces de rede, menu de som, relógio, menu de contas de usuário (My Menu) e menu de sistema.



Painel Superior

1) Menu de aplicativo

O Menu de aplicativo (também conhecido por Menu Global) é uma implementação de menus dinâmicos no painel superior do Ubuntu. Sua função é integrar os menus padrão dos aplicativos (Arquivo, Editar, Exibir, Inserir...) ao painel superior do sistema, economizando espaço em tela, dando uma aparência mais

limpa aos programas e deixando o sistema com uma maior integração com seus aplicativos. Sua adoção visa à economia de espaço em telas pequenas, como notebooks. O Menu de aplicativo fica oculto até que se passe o mouse sobre o painel superior.

2) Menu de mensagem

O menu de mensagem provê acesso rápido aos programas padrão de e-mail, ao mensageiro instantâneo.

3) Menu de conexões

Troca rápida de interfaces de rede pode ser feita através do menu de conexões. Através dele, você tem acesso rápido à mudança de rede cabeada para redessem fio (wireless), 3G, VPNs, ou às configurações de conexões de rede.

Ao ligar seu computador ou notebook equipado com placa de rede sem fio, o Ubuntu irá automaticamente detectar as redes sem fio disponíveis.

A opção “Informações da conexão” exibe o tipo de interface utilizada, o endereço MAC, driver utilizado, velocidade de conexão, endereços IP e DNS. Editar Conexões permite criar novas conexões de qualquer uma das interfaces de rede suportadas e será explicado mais adiante.

4) Menu de som

O menu de som integra-se ao player de música Rhythmbox, fornecendo uma experiência musical envolvente e dinâmica, além de controlar o volume de som do computador e ter acesso ao painel de controle de som. Através deste menu, você tem acesso rápido às funções de iniciar música, pausar, avançar para a próxima música, retroceder à música anterior, fila de reprodução e painel de configuração de som, que é parte do painel de configurações do sistema.

Outros tocadores, como o Banshee (opcional, instalável a partir da Central de Programas do Ubuntu) também se integra ao menu de som. O painel de configurações de som permite ativar ou desativar entradas e saídas de som e controle de som por aplicativos.

5) Tempo

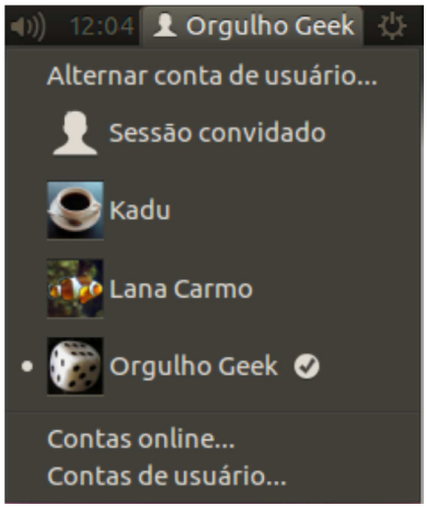
Além de exibir as horas, o relógio do sistema exibe também o calendário e acesso às configurações de zona (GMT) e modos de exibição, como opções de ativação de exibição de segundos, dia da semana, mês, formato de hora e horas de outras cidades.

6) Menu de usuário

O menu de usuário exibe o nome do usuário autenticado (estado também conhecido com “logado”) e contém atalhos de troca rápida entre contas de usuários cadastrados no sistema.

Durante o uso de uma sessão, é possível voltar à tela de escolha de usuário sem a necessidade de encerrar a sessão ativa através da opção “Alternar conta de usuário”. Também é possível selecionar diretamente o usuário para o qual se deseja alternar.

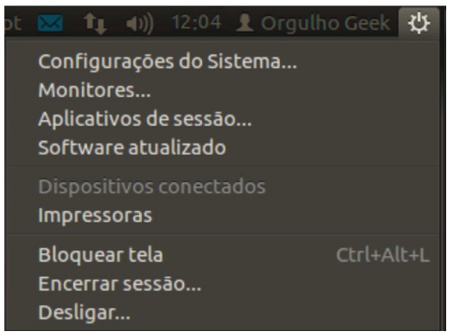
Usuários autenticados são identificados com um ícone marcador ao lado do nome. Através do Me Menu, é possível ter acesso ao painel de configurações de contas de usuários e contas online.



Me Menu

7) Menu de sistema

Acesso rápido ao Painel de Configurações do sistema, seleção de monitores, aplicativos de início de sessão (que se abrem ao iniciar o sistema), atualizações de sistema, controles de dispositivos conectados, impressoras, webcam, bloqueio de tela, encerramento de sessão e desligamento de sistema.



Menu Sistema

Espaços de Trabalho

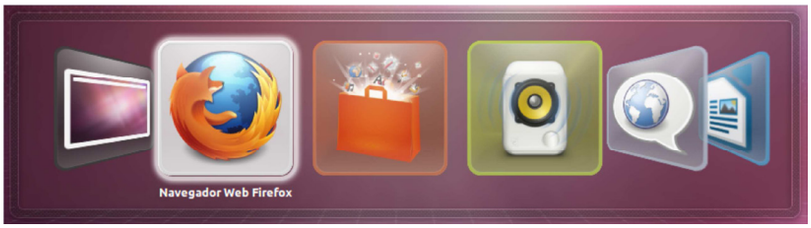
O Ubuntu possui quatro áreas de trabalho interligadas. É como se fosse uma grande área de trabalho dividida em quatro.

Isto pode ajudar em questões como: evitar a perda de foco em tarefas pelo fato de haver dezenas de janelas de aplicativos abertas em uma mesma área de trabalho.

Você pode enviar uma janela para uma área de trabalho ao clicar com o botão direito do mouse sobre a barra de títulos e selecionar a opção “Mover para outro espaço de trabalho”, ou clicando no ícone do Alternador de espaços de trabalho e arrastando a janela para a área desejada. Ao clicar no ícone do aplicativo em outro espaço de trabalho, a tela irá trocar dinamicamente para a área correspondente.

Alternativamente, você pode usar o atalho Ctrl + Alt + seta (na direção desejada) para alternar as telas até o espaço de trabalho desejado, ou a tecla Super + S para visualizar as quatro áreas de trabalho.

Alternador e exposição de janelas



Alternando janelas através do atalho Alt+ Tab

Alternar entre janelas no Ubuntu funciona da mesma forma que nos outros sistemas de computador: através da combinação de teclas Alt + Tab.

O alternador permite a troca rápida entre aplicativos e janelas, sem a necessidade de abrir um por um. Uma pré-visualização do conteúdo da janela pode ser exibido através da tecla apóstrofe ou seta para baixo enquanto segura alt+tab.

Meu computador?

É uma das primeiras perguntas que um veterano usuário de Windows faz ao ver um computador com Linux. Mais especificamente, com Ubuntu:

“Onde encontro o meu computador?”

A resposta é simples: está bem na sua frente, com tudo o que você tem direito!

Enquanto o Windows possui um navegador de arquivos chamado “Explorer” e um local chamado “Meu Computador”, que reúne os discos locais, discos de rede, dispositivos de mídia removível (CD, DVD, etc) e dispositivos removíveis (flash drives, etc), o Ubuntu possui o Nautilus, um navegador de arquivos que traz as mesmas informações, mas de um modo dinâmico.

Através da pasta “Meu computador” do Windows, o usuário pode entrar nas pastas de sistema, de programas e se não tiver experiência ou não souber o que está fazendo, pode causar um belo estrago no sistema, enquanto no Ubuntu, é irrelevante que um usuário que não possui conhecimentos avançados em configuração de sistema tenha acesso aos arquivos do sistema. Eles estão lá, de forma discreta, mas estão. Mais especificamente na raiz do disco “Sistema de arquivos”.

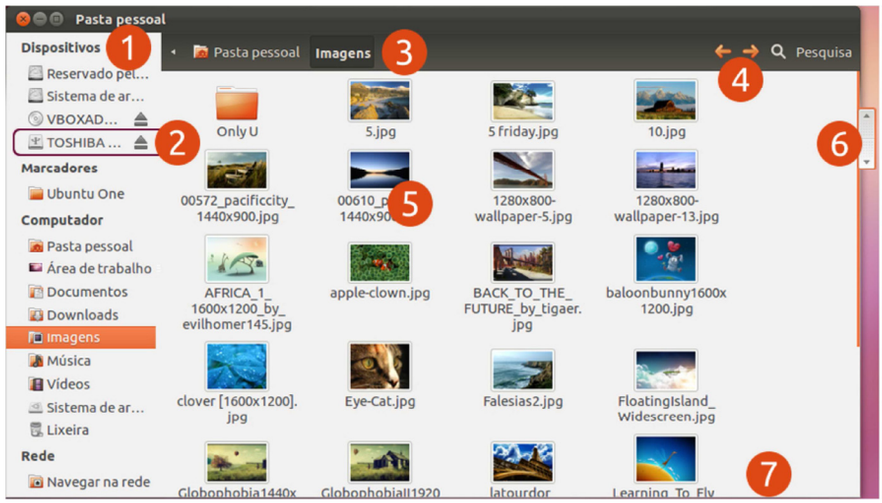
Enquanto no Windows, o acesso aos drives está em Iniciar > Computador > Unidade de DVD (D:), por exemplo, no Ubuntu, você pode ter acesso a estes mesmos itens através do ícone Pasta pessoal > Computador > [HDs locais] (na barra lateral esquerda).

A partir daí, torna-se desnecessário ter uma pasta “Meu computador”, já que, para acessar os outros dispositivos, basta clicar no ícone correspondente. Para abrir o dispositivo em outra janela, basta segurar a tecla CTRL e clicar no ícone desejado com o botão “scroll” (o botão central) do mouse.

A redundância aí é um ponto que foi eliminado do Ubuntu, sendo, portanto, desnecessário manter no sistema um link para um dispositivo de armazenamento que pode ser acessado com apenas dois cliques de distância.

O Nautilus permite que você gerencie graficamente seus diretórios e arquivos, criando, excluindo, copiando, colando, compartilhando ou alterando.

O gerenciador de arquivos, como o próprio nome diz, permite que “naveguemos” por pastas dentro do sistema. Também por esse motivo pode ser chamado de navegador de arquivos.



1) **Barra lateral**: Exibe dispositivos de armazenamento, como pen drives, CDs, DVD e HDs, pastas do usuário e locais de rede. Pode ser ocultada ou exibida pressionando-se a tecla F9.

2) **Dispositivo montado**: O ícone de ejetar ao lado do nome do dispositivo indica que o mesmo está montado e pronto para acesso.

3) **Caminho**: Exibe os diretórios acessados até a pasta atual.

4) **Avançar, retroceder e pesquisa**: Permite voltar em pastas, retroceder e efetuar busca por arquivos em diretórios.

5) **Arquivo**: Arquivos contidos dentro da pasta atual.

6) **Barra de rolagem**: Este ícone é sensível à proximidade do ponteiro (do mouse ou touchpad) e só aparece quando o ponteiro se aproxima da borda direita da janela. Permite rolar a página para cima ou para baixo.

7) **Barra de status**: Exibe informações sobre a quantidade de espaço em disco livre e a quantidade de arquivos no diretório atual.

Modos de visualização do Navegador de arquivos

A forma como procuramos arquivos no computador varia de cada pessoa. Algumas preferem visualizar ícones grandes, para identificar seus arquivos pelos ícones. Outras preferem ver os detalhes de seus arquivos e navegar com poucos cliques entre eles. O Nautilus oferece três modos de visualização de arquivos: **Ícones, Lista e Compacta.**

Você pode mudar a forma de visualização de arquivos do Nautilus acessando o menu de aplicativo Ver e escolher uma das três opções que mais lhe convier.

6.1 MENU DE CONTEXTO

Menus de contexto são aqueles que mudam conforme o contexto, como o nome diz e são acessados clicando-se em determinadas áreas ou arquivos com o botão direito do mouse.

1) **Desktop**

No Desktop, ou área de trabalho, ele nos dá algumas possibilidades, como por exemplo, criar pastas, criar lançadores (que são atalhos para aplicativos), criar documentos (com o Open Office ou algum editor de textos), alinhar os ícones no Desktop ou ter acesso às configurações de tema em “Alterar plano de fundo”, que serão abordados no decorrer do livro.

2) **Diretórios**

Em diretórios, ou pastas do sistema, o menu de contexto de comporta forma um pouco diferente do que no Desktop. Sai à função “Alterar plano de fundo” e “Manter alinhado” e entra “Sincronizar com o Ubuntu One” (se a sua versão do Ubuntu for superior a 10.04), “Ampliar”, para aumentar o tamanho de visualização das pastas, “Reduzir” para diminuir o tamanho de visualização, “Tamanho normal” para voltar ao padrão e “Propriedades,” que nos mostra quantos arquivos têm naquela pasta, quantas pastas têm e qual o tamanho total da mesma.

3) **Arquivos**

Menu de contexto em arquivos é um pouco maior que em outras áreas do sistema. Isto porque é possível realizar várias tarefas com apenas dois cliques do mouse. Sobre arquivos ou pastas, temos as seguintes opções: “Abrir”, para executar o arquivo ou abrir a pasta, “Abrir em nova aba” no gerenciador de janelas, “Abrir em nova janela”, “Abrir com outro aplicativo”, para abrir com um programa que não é o padrão, os tradicionais “Recortar”, “Copiar” e “Colar”, “Criar link”, “Renomear” o arquivo ou pasta, “Copiar para” outro local, “Mover para” outro local, “Mover para a lixeira” para excluir o arquivo “Enviar para” para enviar o arquivo por e-mail, “Comprimir”, para compactar o arquivo e transformá-lo em .Rar, .Zip ou .Tar, configurar as “Opções de Compartilhamento” para redes, “Sincronizar no Ubuntu One” para enviar seu arquivo para a rede de cópia de segurança on-line do Ubuntu (gratuito até 2GiB) e acesso as “Propriedades”, que lhe mostrará o tamanho do arquivo ou pasta, definir qual o programa padrão para abrir o arquivo, mudar o ícone ou outras configurações avançadas.

Login - Autenticação de usuário

Quando instalado em um computador, o para entrar no espaço de usuário o Ubuntu, será necessário possuir uma conta de usuário, que pode ser criada na hora da instalação para o usuário administrador, ou através do Painel de Configurações do Ubuntu.

Detalhes de instalação são mostrados mais a frente.

Através de um menu de seleção de usuários, é possível selecionar sua conta e no campo “Senha”, deve ser digitada a senha de usuário.

6.2 O MOUSE

O Unity incorporou ao Ubuntu alguns gestos de mouse, a fim de tornar o uso da área de trabalho tão dinâmico quanto às teclas de atalho. Confira a seguir os gestos configurados por padrão no Ubuntu:

Botão esquerdo

Segurar a janela pela barra de título e arrastá-la, encostando-a no topo da tela	Maximiza a janela
Segurar a janela pela barra de títulos e arrastá-la, encostando-a nas laterais da tela.	Posiciona a janela na metade da tela
Segurar a janela maximizada pelo painel superior e arrastá-la para baixo	Restaura a janela

Botão central

Clique em um ícone no Lançador	Inicia uma nova instância do item
Clique na barra de títulos da janela ativa	Envia a janela trás nas outras janelas abertas
Clique no botão “Maximizar” da janela ativa	Expande a janela verticalmente
Rolar a roda (scroll) na barra de título da janela ativa (função depende da direção)	Encolhe ou expande a janela

Botão direito

Clique em um ícone no Lançador	Mostra opções do item
Clique na barra de títulos da janela ativa	Mostra opções para a janela
Clique no botão "Maximizar" da janela ativa	Expande a janela horizontalmente

Teclas de atalho

Teclas de atalho são excelentes companheiras da produtividade. Através de combinação de teclas, o usuário realiza a maioria das tarefas que o faria mover uma das mãos da tecla para o mouse (por exemplo, ao editar de um documento de texto), afim de selecionar ou ver um detalhe em outro lugar do computador.



Lista de atalhos

Ao segurar a tecla Super por alguns segundos, você poderá ver na própria tela os atalhos do Unity, mas juntamos aqui as combinações de teclas para aumento de produtividade no Ubuntu.

Painel

Super	Abre o Painel. Você pode digitar a palavra desejada, navegar pelos resultados através das teclas setas do teclado e selecionar o item desejado pressionando Enter. Também é Exibida a lista de atalhos do Unity.
Alt + F2	Abre o painel de execução de comandos (gksudo) e aplicativos
F10	Abre o primeiro item da área de notificações do Ubuntu. Você pode deslocar o foco entre as opções dos menus comas setas do teclado.
Esc	Fecha os menus sem efetuar qualquer seleção

Lançador e gerenciamento de aplicativos

Super + 1 (ou 2, 3... 0)	Abre ou põe em foco um aplicativo na posição correspondente no Lançador
Super + Shift + 1 (ou 2, 3...0)	Abre uma nova instância do aplicativo se ele já estiver em execução.
Super + D	Expõe os aplicativos em execução
Super + Q	Fecha a janela em foco
Super + S	Expõe os quatro espaços de trabalho para visualização ampla. Equivalente ao clique no ícone do Alternador
Super + T	Abre a Lixeira
Super + W	Expõe as janelas dos aplicativos abertos dispostos lado alado.

Gerenciamento de janelas

Ctrl + Alt + Seta (←↑↓→)	Alterna para o espaço de trabalho de acordo com a direção indicada pela seta pressionada.
Ctrl + Alt + Shift + Seta (←↑↓→)	Desloca a janela em foco para a área de trabalho na direção indicada pela seta pressionada.
Ctrl + Alt + Num.0	Maximiza a janela ativa
Ctrl + Alt + Num.1	Posiciona a janela ativa na lateral inferior esquerda da tela
Ctrl + Alt + Num. 2	Posiciona a janela ativa na metade inferior da tela
Ctrl + Alt + Num. 3	Posiciona a janela ativa na lateral inferior direita da tela
Ctrl + Alt + Num. 4	Posiciona a janela ativa na metade esquerda da tela
Ctrl + Alt + Num. 5	Posiciona a janela ativa completamente aberta no centro da tela sem maximizar
Ctrl + Alt + Num. 6	Posiciona a janela ativa na metade direita da tela
Ctrl + Alt + Num. 7	Posiciona a janela ativa no canto superior esquerdo da tela
Ctrl + Alt + Num. 8	Posiciona a janela ativa na metade superior da tela
Ctrl + Alt + Num. 9	Posiciona a janela ativa no canto superior direito da tela
Ctrl + Alt + L	Bloqueia a tela

7 - INSTALAÇÃO, BACKUP E RESTAURAÇÃO.

O Ubuntu é um sistema tão flexível, que, diferentemente do Windows, ele permite que você personalize sua instalação. É possível, por exemplo, instalar o Ubuntu lada-a-lado com outro sistema, colocar o a inicialização do sistema em outro HD, ou ainda, guardar seus documentos em uma partição ou HD separados.

Como visto no capítulo anterior, é possível executar o Ubuntu direto do DVD ou do pen drive, contudo, seu melhor desempenho acontece mesmo quando sua execução é a partir da instalação no HD.

Embora a instalação tenha sido projetada para ser rápida, durante este processo, o Ubuntu baixa da internet algumas atualizações do sistema, pacotes de linguagem para o seu idioma, o que, dependendo da velocidade de sua conexão com a internet, pode ser um pouco acima do tempo previsto na instalação sem conexão com a internet.

Este capítulo apresentará a forma padrão de instalação do Ubuntu, a instalação lado alado e a instalação avançada. Você aprenderá como liberar espaço em seu HD com Windows e utilizar este espaço com o Ubuntu.

Configurações recomendadas:

Processador: 1,6Ghz ou superior;

Memória RAM: 1GB ou mais (2GB para 64Bits)

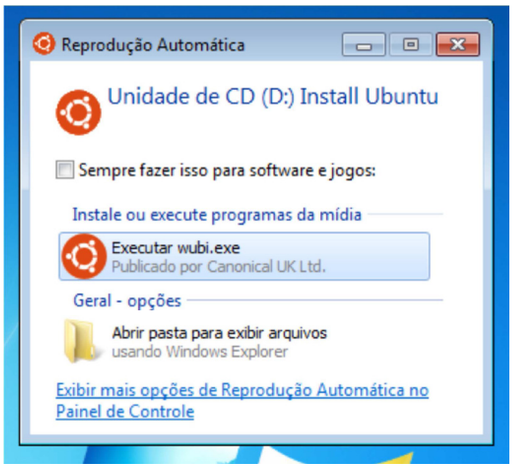
Espaço em disco rígido (HD): 15GB

Chipset de vídeo: Nvidia GeForce 6200 ou superior / AMD Radeon HD 3450 ou superior

/ Intel GMA 3000+ ou superior

Partindo de dentro do Windows

A instalação padrão do Ubuntu é feita de forma simples e rápida e permite que você instale apenas o Ubuntu em seu computador, ou instale-o lado a lado com o Windows.



Você pode, opcionalmente, iniciar a o processo de dentro do próprio Windows. Para isso, insira o disco de instalação do Ubuntu em sua unidade de DVD. Caso o Windows esteja configurado para executar CDs e DVDs automaticamente, uma janela irá solicitar sua atenção. Escolha a opção “Executar Wubi.exe”.



Tela do instalador

No Menu do Ubuntu selecione a opção “Demonstração e instalação completa”. Sistema irá reiniciar.

Instalação lado a lado

A instalação lado a lado lhe permitirá manter os dois sistemas em seu computador, sem que um interfira no funcionamento do outro.

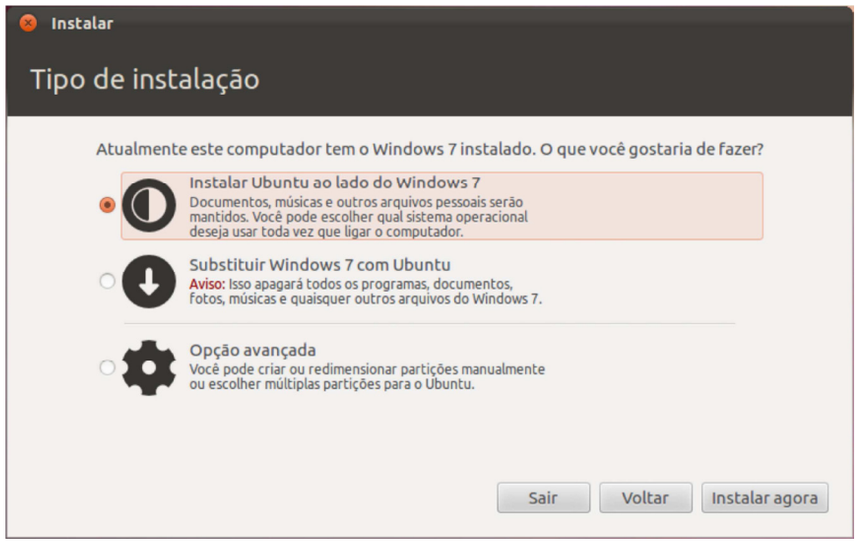


Tela de Boas Vindas do instalador e de seleção de idioma

Após seguir reiniciar seu computador, O Ubuntu irá exibir a tela de carregamento do Ubuntu. Aguarde alguns segundos até que a o sistema seja carregado do DVD para a memória do computador.

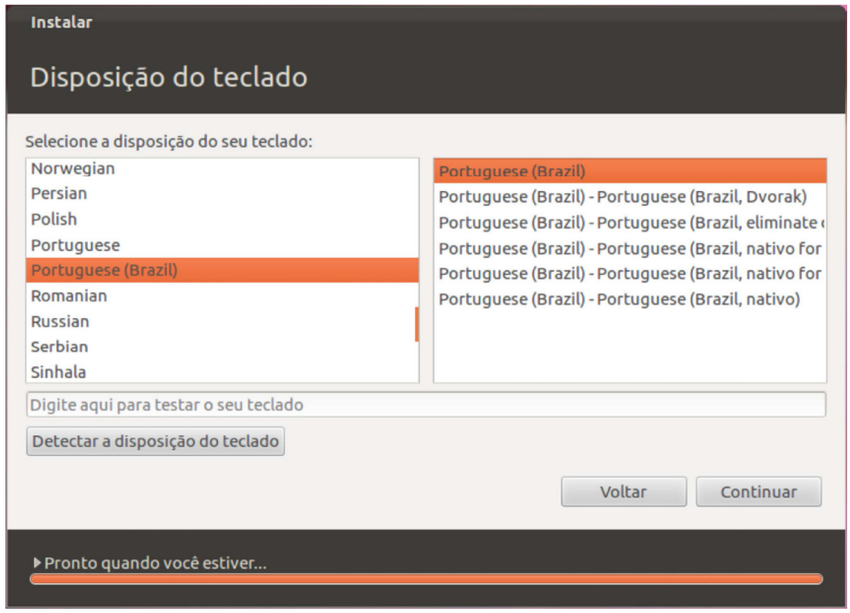
Ao ser carregado, uma janela de boas vindas irá permitir a seleção de linguagem e execução. Selecione na lateral esquerda da tela a linguagem de sua preferência. Se sua linguagem for “Português” ou “Português do Brasil”, clique no botão “Instalar o Ubuntu” para iniciar o processo de configuração de instalação.

A tela seguinte “Preparando para instalar o Ubuntu” irá checar a quantidade de espaço mínimo disponível em disco necessária para a instalação e se o computador está conectado à internet. Certifique-se de marcar as opções “Baixar atualizações enquanto instala” e “Instalar esse programa de terceiro”. Clique no botão “Continuar”.



Na tela “Tipo de instalação”, selecione a opção “Instalar Ubuntu ao lado do Windows ”e clique no botão “Instalar agora”.

O instalador do Ubuntu irá solicitar sua localização. Selecione no mapa e clique em “Continuar”



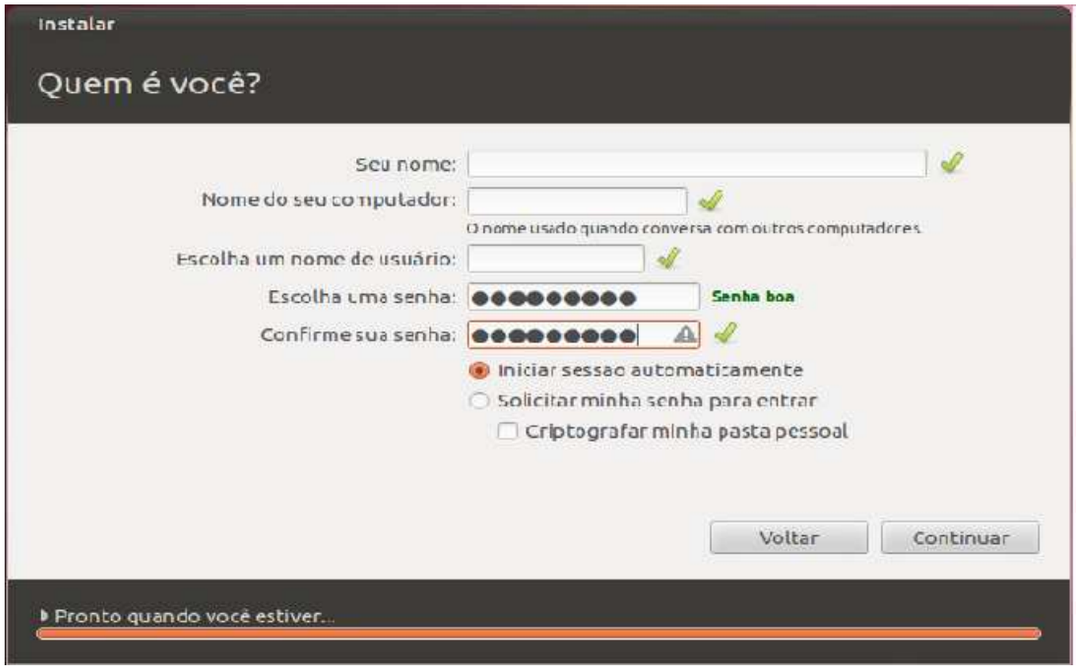
Seleção de tipo de teclado

Em seguida, o Ubuntu solicitará que você escolha o padrão de seu teclado. O instalador normalmente reconhece o tipo de teclado automaticamente, porém, é

recomendado que teste-se a disposição do teclado digitando as teclas especiais e letras acentuadas, como: à,à, ê, ã, ç, etc.

A tela de informações do instalador lhe solicitará algumas informações, como seu nome, nome do seu computador (o sistema sugere automaticamente) e nome de usuário. Por padrão, o Ubuntu sugere o primeiro nome como nome do usuário, mas se desejar é possível alterar, desde que não haja espaços ou caracteres especiais. Digite a senha e repita-a no campo “Confirme sua senha”.

O Ubuntu possui um esquema de proteção de contas de usuários avançado, a fim de proteger os dados contra invasores e malwares, por isso, é muito importante que esta senha não seja fácil de descobrir e não seja compartilhada. Recomenda-se também uma senha que não possa ser perdida ou esquecida, pois em conjunto com a opção de criptografia, a senha fornece máxima proteção de dados, sendo impossível recuperá-los no caso de perda da senha.



Tela de configuração dos dados do usuário

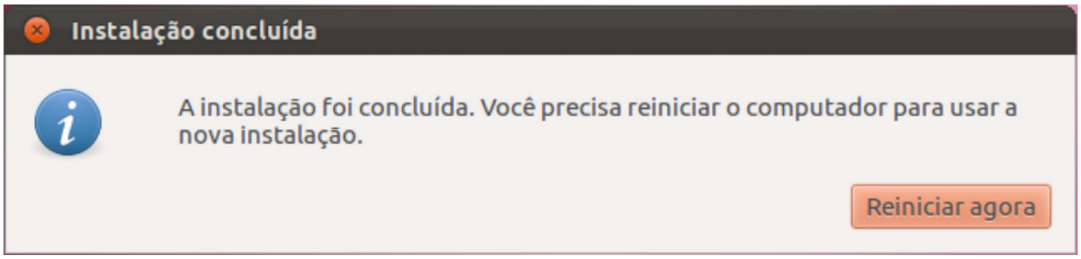
Caso você seja a única pessoa a usar o computador e quiser você o sistema faça o login automático, pode marcar a opção “Iniciar sessão automaticamente”, caso contrário, deixe selecionada a opção “Solicitar minha senha para entrar”.

A opção “Criptografar minha pasta pessoal” irá inserir uma proteção adicional aos seus documentos armazenados no computador. Indica-se esta opção quando o sistema estiver instalado em um notebook. Clique em Avançar.

Se seu computador for dotado de um webcam, será possível tirar uma foto para ser usada como imagem de usuário. Caso não queira, escolha uma das opções à direita da imagem da webcam, clique no botão “Próximo”.



Tire uma foto ou escolha umas das imagens oferecidas como imagem de usuário



Diálogo de finalização de instalação

Instalação Avançada

O Objetivo da instalação avançada é permitir que, caso seja necessária uma reinstalação do sistema, os arquivos e dados do usuário continuem intactos em sua partição. Também é uma opção para quem deseja instalar uma nova versão do Ubuntu e utilizar o mesmo diretório /home, ou ainda, realizar um upgrade (atualizar para uma nova versão) com uma instalação limpa.

Este tipo de instalação permite a você dividir o HD de seu computador em partes lógicas, mantendo o sistema já instalado em um espaço, a instalação do Ubuntu em outro e uma parte será separada especialmente para seus arquivos,

assim, caso seja necessário reinstalar ou formatar o Ubuntu futuramente, seus arquivos e dados pessoais continuarão no computador.

Instalando o Ubuntu por um pen drive

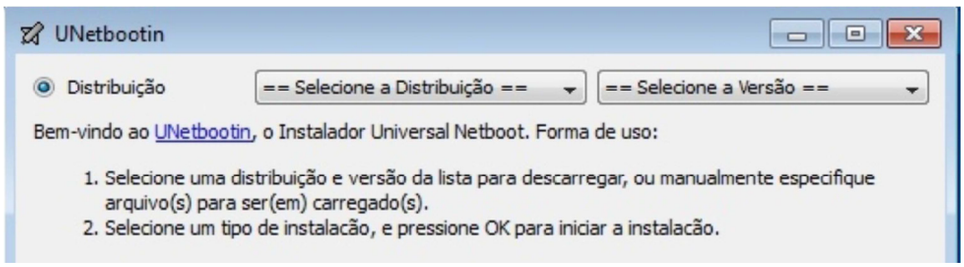
Alguns modelos de computadores, como notebooks, ultra books e alguns notebooks, já não vêm equipados com unidade de CD/DVD, porém, o que numa hora dessas poderia ser uma dor de cabeça, torna-se irrelevante quando o usuário possui um pen drive com pelo menos 2GB de espaço.

O site oficial do Ubuntu sugere uma solução, contudo, apresentaremos uma solução diferente. Conheça a seguir o **UNetbootin.**

Antes de iniciar, será necessário efetuar o download da imagem ISO do Ubuntu no site oficial. Lembre-se também de efetuar um backup dos dados armazenados em seu pendrive. O UNetbootin copia o conteúdo de instalação do sistema para o pen drive e torna-o inicializável. Não importa se a fonte original é um CD ou uma imagem de disco para gravação que você tenha baixado da internet. Ele copiará o conteúdo e tornará seu pen-drive um dispositivo de inicialização.

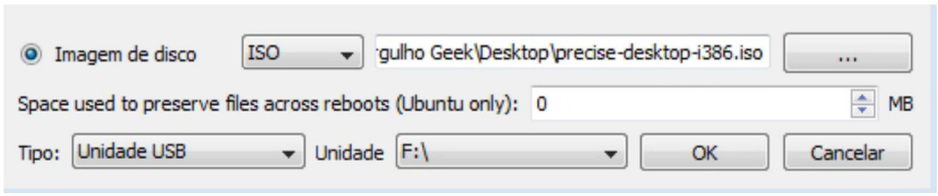
O download do UNetbootin pode ser feito gratuitamente e está disponível para Windows, Linux e Mac OS. Acesse o site do UNetbootin através do endereço: **http://unetbootin.sourceforge.net** e escolha o sistema que você possui. Por exemplo, se você tiver o Windows em seu computador, baixe o UNetbootin para Windows.

Após baixar e o UNetbootin, execute-o. No Windows, não há necessidade de instalação. Apenas dê dois cliques no arquivo unetbootin-Windows-xxx.exe.



Unetbootin, disponível para Windows, Mac e Linux.

No menu Drop-down “Selecione a distribuição”, selecione o Ubuntu. No menu “Selecione a Versão”, escolha a opção 12.04_Live. Se a versão que você baixou for a 64Bits,escolha 12.04_Live_x64.



Antes de formatar, verifique se o tipo escolhido é realmente o seu pen-drive.

Na parte inferior da janela do programa, ao lado de “Imagem de disco”, selecione a opção ISO, clique no botão “…” e navegue até a imagem ISO baixada no site do Ubuntue selecione-a. Em “Tipo”, selecione “Unidade USB e em “Unidade”, escolha a unidade de disco com a letra do seu pen drive exibida no Explorer do Windows. Finalmente, clique em OK para iniciar o processo de cópia, que pode demorar até 30 minutos.

Ao final do processo, reinicie seu computador, localize a opção de inicialização por portas USB e instale o Ubuntu como se fosse pelo CD.

Backup e restauração de dados

O Ubuntu traz uma ferramenta eficiente de cópia de segurança (backup) e restauração de dados, capaz de salvar suas configurações pessoais, arquivos como músicas, documentos, imagens e outros em arquivos que podem ser criptografados. O backup é uma forma eficiente de manter seus dados protegidos contra uma quebra no sistema, defeitos no computador ou outras eventualidades e pode ocorrer de forma incremental.

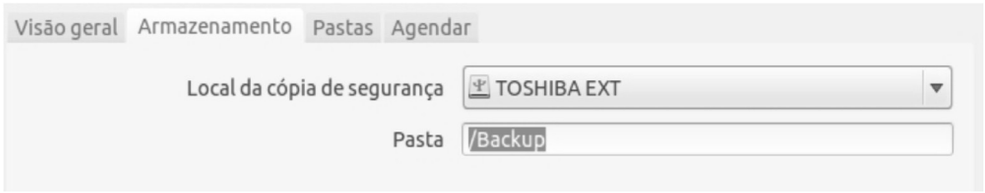
Devido à velocidade de evolução do software aberto, o backup não oferece por padrão a possibilidade de efetuar a cópia de segurança dos programas instalados, pois em uma restauração futura, os programas estariam obsoletos, ou antigos demais e essa função, como poderá ser visto no próximo capítulo, pode ser substituída com a sincronização de informações de programas instalados pela Central de Programas do Ubuntu.



Ferramenta de cópia de segurança

Para iniciar a cópia de segurança, acesse o Painel de Configurações do Sistema e selecione a opção “Cópia de segurança” ou acesse o Painel principal do Ubuntu e no campo de busca de aplicativos, digite “Cópia” ou “Segura” e clique no ícone do aplicativo.

Ao Escolher a opção “Eu quero restaurar arquivos de uma cópia de segurança anterior”, você deverá ter os arquivos de cópia de segurança. Enquanto não for feita uma cópia de segurança, clique na opção “Apenas mostrar minhas configurações de cópia de segurança”.

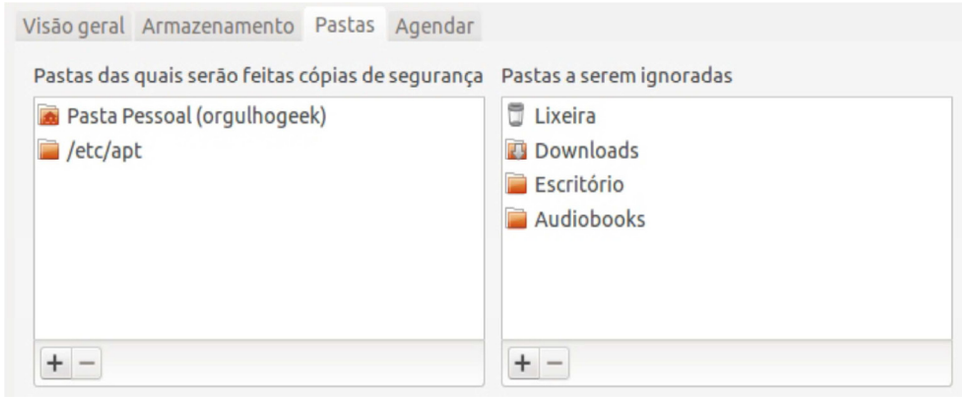


Escolha do local de armazenamento da cópia de segurança

A Ferramenta de cópia possui algumas abas, que dá acesso a algumas configurações de como, quando e onde a cópia de segurança deverá ser feita. É possível, por exemplo, salvar sua cópia de segurança em um HD externo, um pen drive ou até mesmo em servidores da internet, como o Ubuntu One, ou através de FTP. Para configurar o local onde seus dados serão salvos, clique na aba “Armazenamento” e selecione o local desejado. No campo “Pasta”, especifique um diretório onde sua cópia de segurança será arquivada.

A aba “Pastas” traz a opção de seleção de pastas que deverão ser inseridas na cópia de segurança e as pastas que deverão estar fora da cópia. Por padrão,

toda a pasta de usuário (o diretório do usuário no momento da cópia, localizado na pasta /home) é inserida na cópia de segurança, exceto as pastas Lixeira e Downloads.



Escolha das pastas a serem salvas na cópia de segurança

Você pode incluir novos diretórios a serem copiados, e pode também ignorar diretórios na cópia de segurança através do botão de adição (+), na parte inferior dos referidos campos. Caso tenha inserido um diretório e queira removê-lo, ou, por exemplo, que ira realizar a cópia de segurança do diretório Downloads, selecione-o e clique no botão de subtração (-) localizado na parte inferior da janela do referido campo.

Em “Agendar”, é possível configurar o intervalo entre cópias de segurança e o tempo que estas cópias deverão ficar armazenadas. Esta opção evita, por exemplo, a manutenção de cópias de segurança muito obsoletas. Também permite automatizar o processo de backup, proporcionando mais comodidade ao usuário.

Tenha em mente que cópias de segurança ocupam espaço em disco, por isso, recomenda-se gerenciar estas cópias com atenção especial a fim de evitar perda desnecessária de dados.

Restauração de cópia de segurança



Para retornar as configurações pessoais e arquivos após uma formatação, ou, para retornar alguns arquivos ao estado anterior, é necessário recorrer às cópias de segurança previamente feitas. Para iniciar a restauração de cópia de segurança, execute a ferramenta de cópia de segurança e selecione a opção “Eu quero restaurar arquivos de uma cópia de segurança anterior”, ou abra a aba “Visão geral” e clique no botão “Restaurar”.

Ao selecionar a opção de restauração, será necessário informar a localização da cópia de segurança. Em “Local da cópia de segurança”, selecione o dispositivo, local da rede ou outro local da cópia. Em “Pasta”, digite o nome da pasta contendo os arquivos de restauração. Clique em “Avançar”.

Em caso de múltiplas cópias de segurança, será necessário escolher a data que você deseja restaurar. Também será necessário escolher a pasta de restauração da cópia de segurança. Você pode restaurar os arquivos para os locais originais, ou restaurar em um local específico, como uma pasta ou outro dispositivo.

8 – CONTROLE DO SISTEMA

8.1 – Central de Programas do Ubuntu

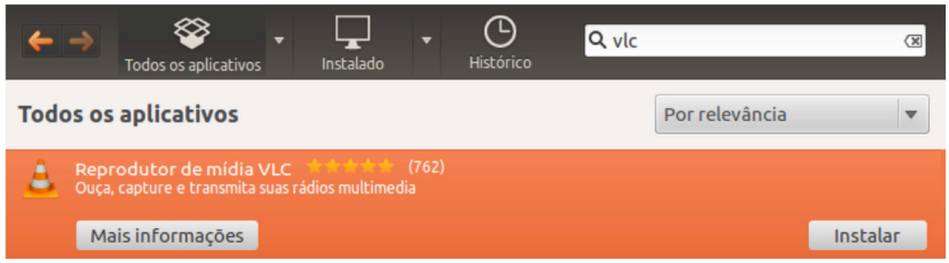
A Central de Programas do Ubuntu é uma loja de aplicativos integrada ao sistema que e permite instalar e desinstalar programas com poucos cliques, de forma simples, com informações úteis, imagens e avaliações de usuários.

Através da Central de Programas do Ubuntu também é possível adicionar fornecedores de softwares que não são inclusos por padrão no Ubuntu. Programas

podem ser facilmente encontrados através de categorias, distribuidor ou simplesmente digitando-se o nome (e às vezes, basta digitar a função do programa) para que ele seja encontrado.

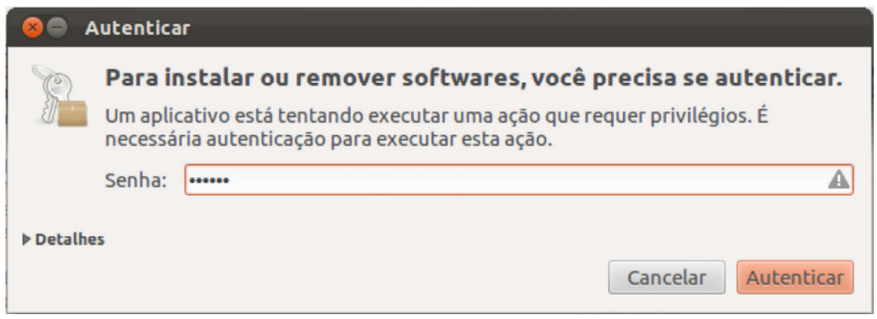
Instalação de aplicativos pela Central de Programas

A instalação de aplicativos pela Central de Programas do Ubuntu. Tanto pela busca através do campo de pesquisa, quanto pelos programas de fornecedores, pelos destaques, ou por pacotes DEB baixados independentemente do sistema. Um detalhe a ser observado, é que o Ubuntu se conecta a internet para buscar e baixar o programa para seu computador, portanto, ao procurar um software para instalação via Central de Programas, não se esqueça de verificar sua conexão com a internet.



Se você já conhece o software que irá instalar, poderá clicar diretamente no botão “Instalar”, ou então, se desejar saber mais informações sobre o programa, visualizar as telas de execução do programa ou conferir as avaliações de usuários, clique em “Mais informações”.

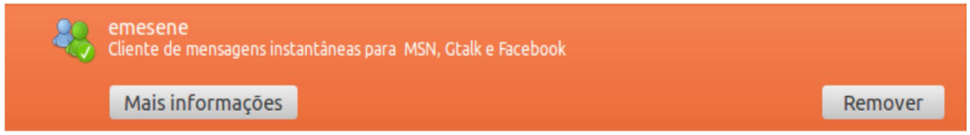
A fim de proteger o computador contra mudanças não autorizadas pelo proprietário, o Ubuntu solicita a senha de administrador, que é definida no momento da instalação ou na criação de conta de usuário (explicado mais adiante neste capítulo), ao clicar no botão “Instalar”.



Para instalar um programa, será necessário informar a senha de administrador.

Desinstalação de aplicativos pela Central de Programas

Para remover um programa, você poderá também localizá-lo através do campo de busca ou do ícone “Instalado”. Dentro desta opção, você poderá acessar os programas instalados através de categorias, como Internet, escritório, jogos, etc.



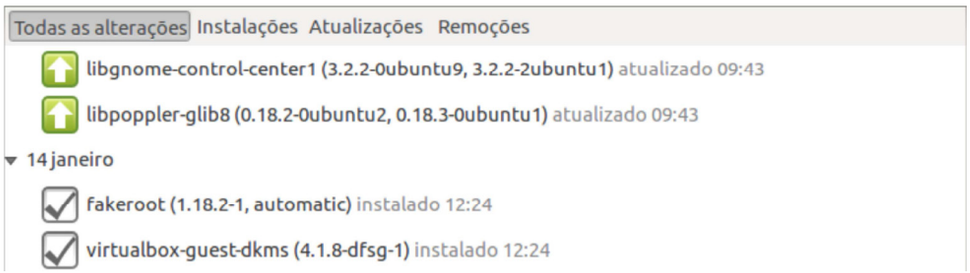
Aplicativos instalados, além do indicador verde no ícone, exibem o botão "Remover".

Quando instalado, a Central de Programas exibirá o botão “Remover” no lugar do botão instalar. Caso clique na opção de remoção, o Ubuntu lhe solicitará a senha de administrador.

Histórico

O botão “Histórico” da Central de Programas do Ubuntu armazena a data, hora, pacotes e bibliotecas de aplicativos instalados.

Através do histórico, é possível saber se um determinado programa sofreu atualização, quais os componentes que foram instalados em uma data e quais foram removidos pordes instalação ou remoção do usuário.



Histórico

Atualizações automáticas

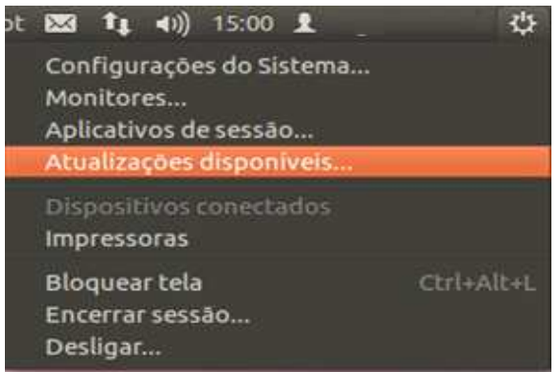
Atualizar um sistema é uma das formas que os fabricantes de sistemas possuem para manter seu produto em dia com as novas tecnologias – no caso do Ubuntu, atualizações de Kernel -, pacotes de segurança e melhorias em drivers.

Durante a construção de qualquer sistema, podem ocorrer pedaços de código que permitam a utilização do computador de forma indevida, funcionamento indesejado de alguns itens, brechas de segurança e coisas do gênero. Contudo, mesmos aos olhos mais atentos, existem falhas que, se não existiam, passam a existir devido à integração entre programas e sistemas.

Hackers mal-intencionados aproveitam-se de brechas em sistemas, para infectar os computadores dos usuários com códigos maliciosos, que se auto executam, copiam e enviam para outros usuários, criando assim, uma rede de computadores infectados, também chamada de “Rede zumbi”.

Como nenhum sistema está livre de bugs, o Ubuntu recebe regularmente pacotes de atualizações que colocam o seu computador em dia com a segurança e a correção de erros de programação, mesmo sendo um sistema livre e gratuito. No caso do Ubuntu 12.04, este suporte será estendido até 2015 para computadores domésticos e 2017 para computadores corporativos.

Para verificar as atualizações de sistema, acesse o menu de sistema (ao lado do nome de usuário, no painel de indicadores) e selecione a opção “Atualizações disponíveis”. Casos eu sistema já esteja atualizado, você verá “Software atualizado” ao invés de “Atualizações disponíveis”.

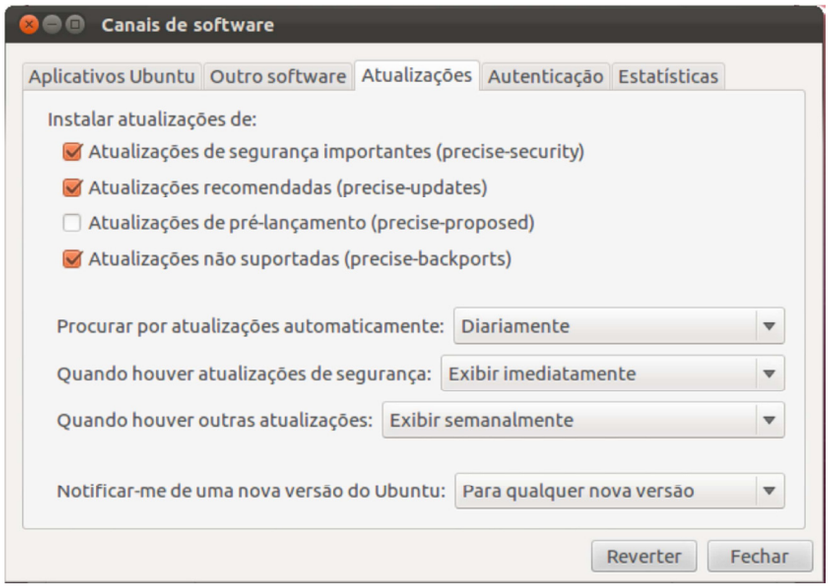


O menu de sistema do Ubuntu informa quando há atualizações disponíveis

A janela do gerenciador de atualizações será aberta. Você poderá receber todas as atualizações de uma só vez, ou então marcar apenas as atualizações para os softwares que lhe interessar, marcando o item desejado no quadrado à direita.

Após marcar os itens desejados, clique em “Instalar atualizações” e, se lhe for solicita da à senha, digite a senha de administrador.

É possível alterar a frequência de busca por atualizações e seus tipos. Para alterar as configurações de atualizações automáticas, clique no botão “Configurações”, dentro da janela de atualizações. Será aberta então a janela “Canais de software” na aba “Atualizações”.



Configurações de atualizações

Configurações do sistema

As configurações do sistema no Ubuntu 12.04 foram profundamente melhoradas, reunindo em apenas uma janela, todas as condições necessárias para que você possa administrar seu sistema com muita facilidade, sem a necessidade de recorrer às linhas de comando.



O painel de configurações do Ubuntu

O painel de configurações do sistema funciona de modo dinâmico, onde quase todas as ações ocorrem dentro de uma só janela, que mudará seu contexto conforme a opção escolhida. Para retornar ao painel de configurações do sistema, basta clicar no botão “Todas as configurações” no alto da janela.

Pessoal

Configurações pessoais afetarão apenas o comportamento do sistema em sua conta de usuário. Nesta categoria são alterados apenas itens superficiais.

Aparência

Planos de fundo (papéis de parede) ou cores podem ser selecionados neste item, bem como as opções de tema padrão disponíveis para o Ubuntu, que são o Ambiance, um tema com barras escuras e janelas claras, ou Radiance, que ilumina seu Desktop com um tema decores claras.



Temas padrão do Ubuntu

Brilho e bloqueio

Através da opção Brilho e bloqueio é possível configurar seu computador para desligara tela quando estiver inativo por um determinado período de tempo, que pode ser escolhido nas opções pré-definidas, assim como ajustar o nível de brilho da tela. Também é possível bloquear a tela, sendo necessária a senha do usuário autenticado no sistema para que a tela seja liberada novamente.

Teclado

É possível alterar o layout do teclado mesmo depois de ter o sistema instalado deforma simples e rápida.

Caso você tenha selecionado um layout diferente de teclado durante a instalação, ou trocou seu teclado para outro padrão, é possível escolher entre diversas opções de disposição de teclas através dessa opção.

Privacidade

O Ubuntu possui um sistema de gravação de registros de atividades, a fim de facilitar o acesso aos itens mais usados através do Painel inicial.



Opções de privacidade, onde é possível apagar registros de atividades recentes.

Sites visitados, músicas tocadas, vídeos assistidos, mensagens de internet enviadas, documentos criados e itens salvos em pastas ficam gravados em históricos do sistema. Você pode configurar algumas opções de privacidade de acordo com alguns programas.

É possível, por exemplo, configurar a gravação de logs – registros de atividades, que incluem aplicativos de mensagens instantâneas, áudio, apresentações, sites visitados, vídeos assistidos, e-mails recebidos e enviados, textos, imagens ou tabelas e configurar anão-gravação de atividades em pastas especificas.

Ubuntu One

O Ubuntu One é uma rede de serviços online da Canonical, voltados para usuários de diversos sistemas. Estes serviços podem ser gratuitos ou pagos, dependendo do tipo de serviço e modalidade.

Os serviços Ubuntu One incluem armazenamento de arquivos online, vendas de músicas através da loja Ubuntu One Music Store e outros.

HARDWARE

As configurações relativas aos dispositivos anexos ao seu computador poderão ser realizadas nesta categoria.

Bluetooth

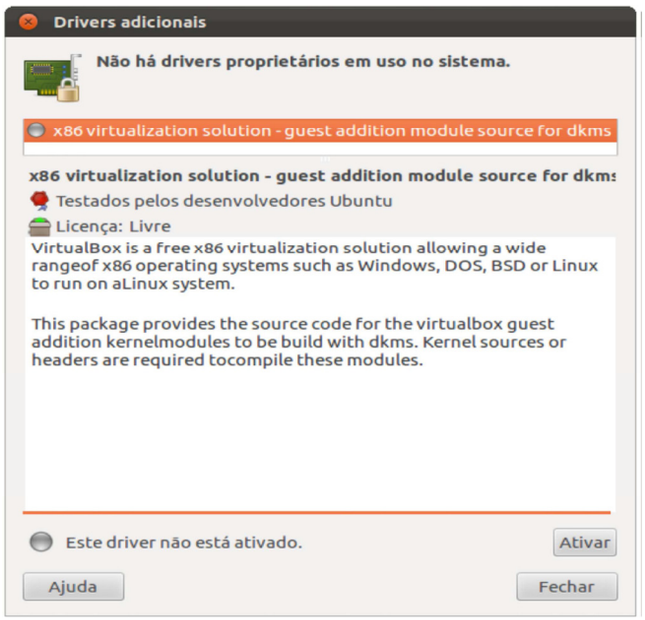
Permite conectar fones de ouvido, celulares, modems, câmeras, teclados, mouses e outros periféricos ao computador sem a utilização de fios através do "pareamento" entre o computador e o dispositivo.

Cor

Através do item "Cor", é possível incluir uma configuração de cores ideal para seu monitor através de arquivos de configuração.icc. Normalmente, os drivers das placas de vídeo e monitores mais recentes trazem a melhor configuração de cores, entretanto, caso você queira uma configuração mais específica às suas necessidades e possua o arquivo adequado, basta utilizar a opção "Adicionar perfil > outro perfil".

Drivers adicionais

Normalmente o Ubuntu instala drivers de dispositivos automaticamente durante a instalação do sistema, contudo, o Ubuntu sempre instala os drivers de código-fonte aberto, que, vez ou outra, não é a melhor opção para o dispositivo. Nesses casos, um ícone é apresentado ao lado dos ícones do menu de mensagens, indicando que um driver não livre está disponível para ser instalado.



Tela de ativação de driver proprietário

Drivers proprietários, em alguns casos, possuem melhor desempenho do que os drivers livres, entretanto, sua atualização não depende do Ubuntu, mas sim do fabricante.

Caso queira instalar um driver proprietário e este esteja disponível para algum item de hardware, você poderá ativá-lo através do item “Drivers adicionais” em “Configurações do sistema”.

Energia

As configurações de economia de energia são feitas neste item. Nele, é possível definir o tempo para suspensão do sistema quando em estado ocioso, o comportamento que o sistema deve tomar ao ser fechada a tampa do notebook e o comportamento de energia extremamente baixa.

Impressão

A maioria das impressoras USB são normalmente instaladas automaticamente quando conectadas ao computador com Ubuntu instalado. O driver é instalado pelo próprio sistema e através do painel de configurações é possível controlar a fila de impressão.

O clique direito do mouse sobre o ícone da impressora, permite uma série de imprimir páginas de teste, definir políticas de uso e compartilhamento e outras ações.

Monitores

O Ubuntu 12,04 trouxe diversas melhorias em relação à ligação de múltiplos monitores. Novos monitores adicionados são automaticamente reconhecidos. É possível utilizar diversos monitores ligados ao mesmo tempo no Ubuntu, incluindo opções de clonagem ou extensão de área de trabalho.

O controle de monitores permite que você configure resolução, rotação, desligue e ligue monitores separadamente ou espelhe sua área de trabalho. Também é possível definir a posição do lançador, se em apenas um monitor, no monitor principal, no secundário ou em todos.

Mouse e touchpad



Controle do mouse e touchpad

Alterações nos dispositivos de interação com o sistema, como mouse ou touchpad podem ser feitas neste item, como a velocidade do ponteiro do mouse ou a sensibilidade do touchpad no notebook.

Configurações interessantes para notebooks podem ser feitas para o touchpad, como a possibilidade de rolar páginas com dois dedos ao invés de usar a zona de rolagem, desabilitar o teclado enquanto usa o touchpad, cliques com o touchpad, velocidade, etc.

Rede

Configurações de rede cabeada ou sem fio, Proxy, VPN e outras opções são encontradas nesse item.

O modo avião permite desligar todos os sinais de rede enviados por seu computador, seja wireless ou 3G.

As opções de alteração ou inclusão de endereço IPv4, IPv6, DHCP, rotas, servidores de DNS, domínios, ou segurança sem fio podem ser feitas ao clicar no botão “Opções”.

SISTEMA

A administração do sistema em níveis mais elevados são feitas nessa categoria, como configuração de acessibilidade, usuários, backup (cópia de segurança) e restauração dedados e seleção de programas padrão. Confira a seguir.

Acessibilidade

Pessoas com necessidades especiais possuem um tratamento especial no Ubuntu, assim como no Linux em geral, que é um sistema pensado para atingir todos os tipos de pessoas.

Contas de usuários

Durante a instalação do Ubuntu, é necessário definir apenas um usuário, que será também o administrador do sistema. Após a instalação é possível definir as contas para outros usuários, caso seja necessário.

Para criar um novo usuário, será necessário “desbloquear” a opção, pois ela requer a senha de administração do sistema, que é definida no momento da instalação. Clique no botão de adição (““ +”) e em “Tipo de conta” poderá ser definido os privilégios do novo usuário.

A remoção de usuários é feita clicando-se em seu perfil no campo “Outras contas” e no botão de subtração (“ - “).

Contas padrão não possuem autonomia para instalação, desinstalação ou alterações no perfil dos outros usuários.

9 – TERMINAL

Como mostrado anteriormente o Linux possui 2 tipos de interfaces (Gráfica e por linha de comando), que podem ser acessíveis com a combinação das teclas CTRL + Alt + F [1 a 6, sendo que 7 é o ambiente gráfico], para a utilização por linha de comando utiliza-se o terminal, o usuário possui uma pasta própria no sistema, que fica dentro de outra pasta chamada “home”. Logo, o Terminal opera por padrão, dentro da pasta home/seu_nome, mas você pode mudar isso a qualquer momento com o comando direcionado à navegação de pastas. Além do seu nome de usuário seguido de “@”, é mostrado também o nome dado à instalação do sistema.

O Ubuntu, assim como em qualquer Linux, que é um sistema similar ao Unix, à entrada de caracteres é sensível à caixa (conforme citado anteriormente, “Case-sensitive”). Então “A” e “a” são diferentes. Isso vale tanto para o Terminal, quanto para a criação de pastas, renomear arquivos e outras funções via interface gráfica.

Os comandos básicos do terminal

Existem inúmeros comandos básicos que você precisa aprender se deseja operar o Terminal com eficiência e iniciar o seu domínio no Linux, mas sem os comandos mostrados abaixo, torna-se irrelevante seu uso.

Administrador <sudo> / <root>

Sudo é uma abreviação para “SuperuserDo”, ou “Superusuário Faça”. O uso deste comando habilita o usuário a executar comandos com um nível elevado de permissões. Também é chamado de “root”.

Como usuário comum, você pode navegar em sites da internet, ler e-mails, digitar textos, ouvir músicas, ver filmes, editar fotos, jogar... enfim, uma infinidade de tarefas que o Ubuntu lhe proporciona. No entanto, para efetuar qualquer alteração no sistema, você precisará provar ao sistema que você é administrador do sistema através do comando sudo<o que você deseja>.

A sintaxe de uso do comando sudo é simples: digite sudo + comando.

Exemplo:

Sudo./install. run

Sudo deve ser digitado antes do início de um comando. Por exemplo, ao usar o comando de atualização de listas de pacotes do Ubuntu:

sudo apt-get update

Você saberá que precisará usar o comando sudo para executar determinada tarefa quando, ao entrar com o comando, receber a seguinte mensagem:

bash: comando: Permissão negada

Da mesma forma que você pode instalar aplicativos no Ubuntu como Superusuário, você pode iniciar programas como sudo. Por exemplo, se você deseja abrir o navegador de arquivos Nautilus como administrador e alterar alguma pasta ou arquivo na raiz da instalação do Ubuntu, basta digitar o seguinte comando:

sudo nautilus

Entretanto, ao iniciar um aplicativo pelo terminal traz a desvantagem de que se o Terminal for fechado, o aplicativo será fechado junto com ele.

A fim de prevenir alterações indesejadas no sistema e proteger o computador contra aplicações potencialmente perigosas, toda vez que o comando sudo for usado, será solicitada a senha de administrador.

Veja o exemplo abaixo:

adm@notebook:~$ sudo ./install.run

[sudo] password for adm:

Durante a digitação da senha (password) do administrador, não são mostrados quaisquer caracteres na tela. Ao final da introdução da senha, tecle Enter para que a execução da ação desejada prossiga.

Você pode, também, assumir o controle de administrador antes de entrar com qualquer outro comando. Para isso, entre com o comando sudo su.

Veja o exemplo:

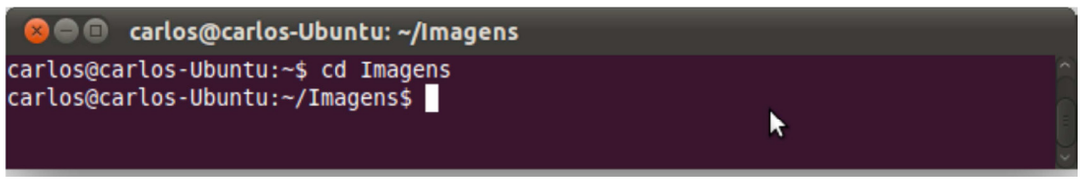
adm@notebook:~$ sudo su

[sudo] password for adm:

Após digitar o comando sudo su, seu usuário será trocado para "root", que é modo de usuário administrador do sistema.

Navegar por pastas <cd>

Desde os remotos tempos do UNIX, o comando para navegar em pastas tem sido o "cd". O comando cd é simples e não possui segredos. Para entrar em uma pasta, basta inserir o comando "cd <nome_da_pasta>". Veja abaixo:



Caso a pasta tenha nome composto, como "Trabalhos da Faculdade", por exemplo, você deverá inserir uma barra invertida (\) ao final de cada nome, exceto o último nome.

adm@notebook:~$ cd Trabalhos\ da\ Faculdade

Para voltar um diretório (pasta), basta digitar "**cd -".**

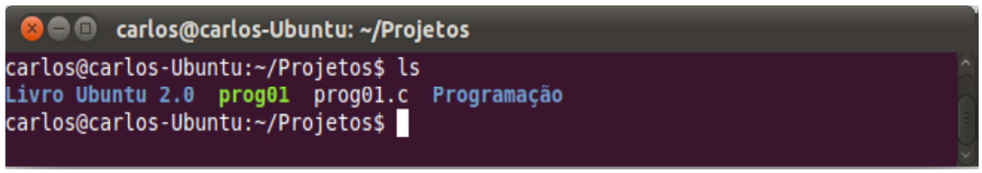
Para voltar direto para o diretório Pasta de pessoal (home), basta digitar "cd".

Uma função muito útil associada ao comando “cd” é a de auto completar palavras. Por exemplo, uma pasta com um nome muito longo ou um pacote de instalação com nome extenso, pode ter sua digitação auto completada ao pressionar a tecla “Tab”. É muito eficaz quando temos de digitar nomes compostos. Por exemplo, “Área de trabalho”:

Listar <ls>

Você já aprendeu a navegar por diretórios via Terminal, mas ainda não sabe como listar os arquivos dentro dele? Basta entrar com o comando ls (de list - listar) e conferir os arquivos na pasta.

Ao listar os arquivos dentro de um diretório, o terminal irá diferenciá-los de acordo com seu tipo. Confira um exemplo abaixo:



Limpar tela <clear>

Clear é um comando de limpeza de tela.

Algumas vezes, após digitar inúmeros comandos ou centenas de palavras tomarem conta de seu terminal (ao finalizar a instalação de um pacote via apt-get, por exemplo), você pode limpar sua tela com um simples comando clear.

Criar e apagar arquivos e diretórios:

<mkdir> / <rmdir> / <rm> / <touch>

“mkdir” é um comando rápido para criação de pastas via Terminal no Ubuntu (e em qualquer Linux). Para criar um diretório, digite o seguinte comando:

mkdir <nome>.

É possível criar diretórios com subdiretórios, usando o argumento -p antes dos diretórios.

Assim como você pode criar um diretório via linha de comando, você também pode remover um diretório vazio. O comando de remoção de diretório é **rmdir <nome>.**

Caso o diretório a ser excluído contenha arquivos e subdiretórios, será necessário usar o argumento -rf antes do nome do diretório, como mostrado a seguir:

rm -rf <nome_do_diretório>:

Você também pode deletar qualquer arquivo, exceto diretórios, usando o comando **rm**.

Iniciar um programa <programa>

Iniciar um programa no terminal requer que você apenas digite o nome do aplicativo desejado. Por exemplo, para iniciar o navegador de internet Mozilla Firefox, simplesmente digite **firefox** no terminal.

Forçar encerramento de processos <kill>

Algumas vezes, independentemente do sistema, um determinado aplicativo pode se comportar de maneira inesperada, como por exemplo, parar de responder, ou fechar a interface de usuário e continuar executando em segundo plano, tornando-se então um comportamento indesejado para um software. O comando kill força àquele aplicativo a fechar-se. À ação do kill damos o nome “Matar tal aplicativo”.

A sintaxe do comando é: **kill <nome ou id do aplicativo>.**

Opcionalmente, você pode “matar” todos os processos ativos ligados ao aplicativo que você deseja forçar a finalização com o comando **killall <aplicativo>.**

10 –ESCRITÓRIO E PRODUTIVIDADE

Gravação de discos com o Brasero

O Brasero é a ferramenta de gravação de discos do Ubuntu.Com uma interface simples e direcionada à tarefa de gravação de mídias, o Brasero é intuitivo e fácil de utilizar e através dele é possível criar discos de áudio, vídeo, dados, DVDs, criar e gravar imagens ISO e afins.

Diferentemente de algumas ferramentas pagas, que oferecem dezenas de funções que fogem ao principal foco do assunto “Gravação de discos”, o Brasero é funcional e excelente em efetuar as tarefas que é destinado, sem pesar o

computador, sem interferir em seu sistema de arquivos, sem oferecer “upgrades” para versões “mais completas”.

A simplicidade do Brasero é tamanha, que para começar a criar um projeto de gravação, basta clicar no botão relacionado ao projeto que você deseja iniciar, arrastar os arquivos que irão ser gravados no disco, inserir o disco a ser gravado na unidade de CD/DVD e clicar no botão “Gravar”.



O gravador de discos Brasero

Writer

Um processador de textos completo, com suporte a múltiplos formatos de arquivos e uma interface clássica. O Writer é capaz de salvar em diversos formatos de arquivos, incluindo os formatos da Suíte Microsoft Office, pdf, html e em outros formatos abertos.

Planilhas eletrônicas no Calc

O Calc é o editor de planilhas eletrônicas da suíte LibreOffice e pode substituir o Microsoft Excel para tarefas de criação de planilhas, cálculos matemáticos, gráficos e outros. Além destas tarefas, o Calc pode ser usado, por exemplo, para administrar materiais, folhas de pagamentos, listas de presença e notas de colégio, contas a pagar, despesas do mês, entre inúmeras outras funções.

Em resumo, o Calc é uma grande folha composta de colunas, linhas. Ao cruzar duas linhas verticais com duas linhas horizontais, temos uma célula.

O Calc presente na suíte LibreOffice 3.4 contém 1024 colunas e 1.048.576 células. Cada célula possui um endereço e seu endereço é relativo à sua localização

entre as linhas e as colunas. Observe que ao clicar em uma célula, ela se torna uma célula ativa.

Apresentações com o Impress

O Impress é a ferramenta de criação de slides e apresentações multimídia doLibreOffice e através de sua interface, é possível criar as mais diversas apresentações compatíveis com Microsoft Office PowerPoint.

Apresentações do Impress podem ser dinâmicas e ricas com adições de imagens 2D e3D, efeitos de transição, efeitos especiais e ferramentas de desenho incorporadas à interface do programa.

11 – SISTEMA DE ARQUIVOS E PERMISSÕES

Quem vem do Windows, inicialmente pode sentir-se desconfortável por não ver um ícone “Meu Computador”, tampouco pela ausência da pasta “Arquivos de Programas”. A seguir, você poderá conhecer a estrutura de diretório Linux e entender como funciona o sistema.

Também chamada às vezes de Sistema de Arquivos, uma estrutura de diretórios é uma forma de armazenamento de arquivos de maneira hierárquica. Assim, o usuário não precisa conhecer os detalhes técnicos dos meios de armazenamento. Ele precisa apenas conhecer a estrutura de diretórios (árvore) para poder navegar dentro dela e ter acesso às suas informações.

A estrutura de diretórios do Ubuntu segue o padrão LSB – Linux Standard Base (Base Padrão de Linux). Tal padrão é comum às principais distribuições Linux do mercado e é definido pelo The Free Standards Group, a fim de padronizar a estrutura interna dos sistemas operacionais baseados em Linux e é baseado na especificação POSIX4 e UNIX, FHS, entre outros.

Algumas distribuições, como o Ubuntu, embora sigam os padrões, incluem alguns diretórios extras na raiz do disco local, entretanto, os diretórios definidos na POSIX continuam intactos.

Segue abaixo alguns dos diretórios do Linux

/srv **Serviço de dados**	Contém dados de serviços fornecidos pelo sistema e cuja aplicação possui um alcance geral (e não apenas dados específicos de um usuário), como servidores web, servidores ftp e outros.
/sys **Suporte aos dispositivos de hardware**	Assim como o /proc, o /sys armazena informações temporárias relacionadas aos dispositivos de hardware, assim como módulos para equipamentos USB.
/tmp **Armazenamentotemporáriocomum**	Armazena arquivos temporariamente, como, por exemplo, downloads em andamento, cache de vídeos da internet (Youtube, Vimeo e outros); • Não é um diretório de armazenamento fixo. Em algumas distribuições Linux, ele é limpo a cada reinicialização do sistema.
/usr **Programas de usuário**	Diretório de armazenamento de programas instalados pelo usuário do computador. É um dos mais importantes diretórios do sistema; • Armazena além de programas, bibliotecas (libs), códigos-fonte, documentação e binários e dados compartilhados em geral. • Também armazena aplicações exclusivas de superusuário.
/bin **Executáveis binários**	•Contém executáveis binários • Comandos padrão no Linux essenciais para a execução do sistema. • Contém comandos usados por todos os usuários • Exemplos: ls, cp, rm, ping, grep.
/dev **Arquivos de dispositivos**	• Armazena o caminho dos dispositivos instalados no sistema; • Cada peça de hardware reconhecido pelo sistema é representada por um arquivo, que fica contida dentro da pasta. /dev (device – dispositivo), como câmeras, modems, placas de Som, vídeo, mouse, teclado, flash drives, HDs externos, etc.
/mnt **Montagem temporária**	• Ponto de montagem para sistemas e arquivos temporários; • Pode receber a montagem de Cds, DVDs, HDs e outros dispositivos.
/opt **Aplicações opcionais**	• O /opt é um diretório que foge à regra FHS e tem como objetivo manter compatibilidade com programas de terceiros (não suportados pelo fabricante do sistema); • LibreOffice, Google Chrome e World of Goo são exemplos de programas que são instalados no diretório /opt.
/root **Administração do sistema**	• Diretório do administrador do sistema. É similar ao /home, porém, o /root é de uso exclusivo do superusuário.
/run	• Diretório recentemente adicionado às principais distribuições

Armazenamento temporário	Linux, incluindo o Ubuntu, seu uso pelo sistema é similar ao uso do diretório /tmp e /var/tmp; • Permite que aplicações armazenem dados necessários para seu funcionamento, mas que não pode ser armazenados em /tmp por motivos de proteção aos arquivos relacionados aos processos em execução que não podem ser apagados por outros processos.

PERMISSÕES

A partir do momento em que um computador é utilizado por mais de um usuário passa a ser importante definir quem pode fazer o que com os arquivos. E mesmo o que o dono do arquivo pode fazer com ele.

Permissões são definições de permitem limitar o acesso, leitura, gravação ou execução de arquivos. O Linux trata todas as partes do sistema, assim como dispositivos de hardware, que são acessados por meio de arquivos5. Também representa a diferença entre um malware ser executado ou não no sistema, logo, você não verá sistemas Linux e UNIX sendo infectados por “vírus de pen drives”, pois, por padrão, executáveis não são executados automaticamente em mídias removíveis.

Permissão pode ser definida como “o ato de dar liberdade, poder, licença ou consentimento”. São usadas para vários fins, mas sua principal finalidade é proteger o sistema e os arquivos dos usuários. A instalação de alguns programas de terceiros (como programas que não são distribuídos pelos canais oficiais do Ubuntu – a Central de Programas, por exemplo), requer a compilação do mesmo e, para compilar programas, torna-se necessário o conhecimento sobre permissões de arquivos.

Ao modificar as permissões de arquivos, é possível permitir que eles alterem e sofram alterações de acordo seu objetivo. Existem três tipos de permissões: Leitura (R), Escrita(W) e Execução (X – não disponível para pastas e diretórios).

Um arquivo com permissão de leitura pode ser aberto, assim como é possível visualizar os arquivos de uma pasta com esta permissão, arquivos com permissão de escrita permitem que sejam feitas alterações nele, já arquivos com permissão de execução, como o nome da permissão sugere, podem ser executados como programas.

No Linux-Ubuntu as permissões de acesso aos arquivos (e diretórios) são divididas em três conjuntos:

• o que o dono do arquivo pode fazer

• o que os usuários do mesmo grupo do dono pode fazer

• outros usuários que não façam parte do grupo

• todos os outros usuários

São quatro as operações sobre os arquivos ou diretórios:

• **r** (Read — leitura)

• **w** (Write — escrita)

• **x** (eXecute — execução)

• **-** (Desabilitada — permissão negada)

O proprietário de um arquivo pode definir de diversas maneiras possíveis, as combinações de permissões para cada entidade. Enquanto o Proprietário pode ler, escrever e executar, o Grupo pode ter permissões diferentes, assim como Outros.

Os bits de atributo

Bits de atributo definem um controle adicional às permissões do arquivo [Ferreira (2003)]. Cada entidade deve ter suas permissões configuradas pelo dono do arquivo, e, tais configurações são também representadas pelos bits de atributo. Sendo assim, já que existem três entidades, existem também três bits de atributos:

• **Bit SetUID (Set User ID upon execution) –** Quando ativado, o arquivo pode ser executado da mesma forma que é executado pelo proprietário. Esta configuração não é válida para diretórios.

• **Bit SetGID (Set User ID upon execution) –** Se ativado, seleciona a "identidade do grupo". Caso este bit esteja ativado, o arquivo será executado como se o fosse por um membro do grupo de seu proprietário. Arquivos criados em diretórios com este bit ativo, é criado com o mesmo grupo do diretório.

• **Bit Sticky –** Caso este bit esteja ativo, o arquivo com esta permissão não poderá ser apagado, a não ser pelo próprio proprietário.

S S T

A representação simbólica dos bits de atributo deve ser lida da esquerda para a direita através das siglas SST. O primeiro bit, (SetUID) é representado pelo

primeiro S, assim como o segundo bit (SetGID) é representado pelo segundo S, e, por fim, o T representa o bit terceiro bit (Sticky).

Bits ativos podem ser representados por “1” ou “+” (o numeral 1 ou o sinal de gráfico de adição) e bits inativos podem ser representados por “0” ou “-” (o numeral 0 ou o sinal gráfico de subtração).

Os bits de proteção

Bits de proteção, são, a grosso modo, as definições de proteções básicas de cada arquivo, que, como mostrado anteriormente, são leitura, escrita e execução. Ativar ou desativar estes bits irá definir as permissões de quem poderá manusear o arquivo (por manusear, entende-se visualizar, alterar/excluir ou/e executá-lo).

Os bits de proteção são divididos em três grupos de três bits. Cada bit grupo possui três bits e cada bit representa uma permissão. A representação gráfica dos bits de proteção desativados pode ser uma sequência de nove zeros, como mostrado a seguir:

000 000 000 ou 000000000

Acima, as permissões de cada entidade são representadas por um grupo de três zeros e devem ser lidas na sequência correta – esquerda para direita, respectivamente permissões de Proprietário, de Grupo e de Outros.

Ao listar os arquivos de um diretório com o comando “ls -l” (sem aspas) é possível visualizaras permissões dos arquivos da seguinte maneira:

rwx rwx rwx

Veja no exemplo, a exibição das permissões na primeira coluna (o primeiro caractere, hora “d”, hora “-”, representa diretório e arquivo, respectivamente e será explicado adiante):



Legenda Informação

1) Permissões
2) Quantidade de subdiretórios
3) Proprietário
4) Grupo
5) Tamanho (em bytes)
6) Data de criação ou última modificação
7) Hora da criação ou última modificação
8) Nome do arquivo ou diretório

Os tipos de arquivos

No Linux, tudo, inclusive dispositivos, é representado em formato de arquivo. Com isso, surgiu a necessidade de se dividir os arquivos em tipos distintos[2], sendo atualmente sete tipos. Confira a tabela:

Tipo	ID	Exemplo de aplicação
Arquivos Comuns	-	Arquivos binário, de texto e de dados
Diretórios	d	Arquivos que contém nomes de outros arquivos armazenados, que estão armazenados em grupos[2]
Links	l	Arquivos que referenciam outros arquivos dentro do sistema de arquivos. Também conhecidos como "atalhos" e podem ser softlinks (links simbólicos) ou hardlinks (links diretos)
Caracteres	c	São arquivos que representam dispositivos físicos (dispositivos de hardware) que podem ser acessados sequencialmente, como dispositivos conectados por portas seriais ou paralelas
Blocos	b	Arquivos que representam dispositivos físicos e que podem ser acessados em blocos de bytes[2], como HDs, pen drives, CDs e outros
Sockets	s	Arquivos utilizados na comunicação entre processos do sistema
Pipes	p	Assim como os sockets, também são utilizados na comunicação entre processos

Tipos de arquivos

Assim, ao listar as permissões de arquivos em um diretório com o comando ls -l é possível identificar seu tipo através da primeira letra, o caractere que antecede as permissões.

12-PRINCIPAIS COMANDOS DO LINUX

Existem vários sites na internet que apresentam listas de comandos em Linux. Alguns sites interessantes estão listados:

http://www.comandoslinux.com/

http://wiki.ubuntuUbr.org/ComandosBasicos

http://www.tecmundo.com.br/atalho/1982UcomandosUbasicosUdoUlinux.htm

Seguem abaixo alguns comandos comumente usados em Linux. Cada um dos mesmos pode ter vários parâmetros de uso. Consulte o manual online (man nome -do -comando).

Comandos de manipulação de permissões de arquivos

• **chmod -**O comando chmod (abreviação de change mode – alterar modo em português) altera as permissões de acesso a arquivos7. O chmod atua basicamente sobre os níveis proprietário, grupo e outros8. Sua alteração pode ser feita utilizando a notação simbólica ou octal. Tudo o que puder ser inserido entre colchetes é opcional.

• **chown -** Assim como o comando chmod altera as permissões do arquivo, o comando “chown” altera o proprietário do arquivo. Também vale a informação sobre os itens opcionais (os itens entre colchetes).

A sintaxe de uso do comando chown para troca de proprietário sem alterar o grupo do

arquivo:

chown [opção] <novo_proprietário><arquivo>

• **chgrp -** O comando chgrp altera o grupo de um arquivo, conforme mostra a sintaxe de utilização abaixo:

A sintaxe de uso do comando chown para troca de proprietário sem alterar o grupo do

arquivo:

chgrp [opção] <novo_grupo><arquivo>

Outros Comandos básicos

apropos – mostra todos os comandos com a tecla e sua descrição.

cat - exibe o conteúdo do arquivo na tela. Também permite criar arquivos

cal – exibe um calendário do mês corrente concatenar arquivos.

cd- muda o diretório corrente

clear – limpa a tela

cp – copia arquivos

date – mostra na tela a hora e data atuais

exit - termina o shell.

grep – exibe na tela as linhas de um arquivo que contém um padrão (expressão

regular)

head / tail - exibe linhas no inicio / fim do arquivo.

kill – mata um processo

less / more - lista o conteúdo do arquivo.

ln -s - cria links simbólicos (symlinks) para arquivos ou diretórios.

ls – lista o conteúdo de arquivos do diretório

man – exibe o manual online do comando.

mkdir - cria diretórios.

mv- mover ou renomear arquivos

ps – lista processos ativos

pwd – mostra o caminho do diretório corrente

rm - deletar arquivos ou diretórios

rmdir - deleta diretórios (desde que estejam vazios)

who – exibe os usuários logados

wc – exibe a contagem de linhas, palavras e caracteres de um arquivo

whoami – mostra o nome do usuário logado no sistema.

Comandos de Linux podem ser agregados por meio de pipes (|). Por exemplo, se quisermos contar quantas sequências estão presentes num arquivo em formato.

FASTA:

grep “>” vector.seq |wc –l

12.1 – Usuários

Quando falamos em Gerenciamento de usuários no Linux, os dois comandos mais básicos são **adduser** e **passwd,** que permitem, respectivamente, adicionar novos usuários e alterar as senhas de acesso posteriormente.

 # adduser ítalo

Cria o usuário italo, onde será pedido algumas informações do mesmo, dentre elas a senha.

 # passwd italo

Altera a senha do usuário ítalo

Para remover um usuário anteriormente criado utilize o comando **deluser**

 # deluser ítalo

Este comando remove apenas a conta do usuário, sem apagar seus arquivos no diretório **/home**, para apagar esses arquivos, utilize o comando com o parâmetro **–remove-home.**

 # deluser ítalo –remove-home

Se você quiser bloquear a conta de um usuário, utilize o comando:

 # passwd –l ítalo

Para desbloquear a conta use:

 # passwd –u ítalo

Os usuários são cadastrados no sistema através do arquivo **/etc/passwd**. Acessando este arquivo você verá informações referentes aos usuários, tais como:

italo:x:1001:1001:,,,:/home/ítalo:/bin/bash

Neste arquivo, cada campo é separado por :

- **ítalo –** Nome de Login do usuário
- **x –** Indica que a senha do usuário está localizada no arquivo /etc./shadow
- **1001 –** UID do usuário
- **1001 –** GID do usuário
- **,,, –** Informações do usuário, tais como Nome, Telefone. Observe que neste exemplo essas informações não foram preenchidas.
- **/home/ítalo –** Diretório HOME do usuário.
- **/bin/bash –** Shell do usuário, ou seja, o programa que irá interpretar os comandos que o usuário executar.

É possível editar as configurações do usuário manualmente através deste arquivo.

No arquivo **/etc./shadow** são salvas as senhas dos usuários.

12.2 – Grupos

Além das configurações de usuários, temos as configurações de grupos, que permitem organizar os usuários e definir as permissões de acesso de forma mais genérica.

Para adicionar um grupo, utilizamos o comando **groupadd**, como em:

 # groupadd professores

Para adicionar os usuários desejados ao grupo, usamos o próprio comando **adduser**, seguido pelo **login** e o **grupo** ao qual ele deve ser adicionado (uma conta de cada vez).

 # adduser ítalo professores

Para mostrar os grupos a qual o usuário ítalo pertence, use o comando:

 # groups ítalo

As informações sobre os grupos e seus usuários ficam salvas em **/etc/group**.

Neste arquivo é possível adicionar um usuário a um grupo manualmente, basta colocar uma vírgula seguida do nome do usuário que deseja acrescentar:

professores:x:1002:ítalo, gabriella

Assim, o usuário **gabriella** agora fará parte do grupo **professores**.

13 -GERENCIAMENTO DE PONTOS DE MONTAGEM EM SISTEMA GNU/LINUX

Neste capítulo abordaremos um tema fundamental para administradores de sistemas GNU/Linux e também para usuários avançados deste sistema: o gerenciamento de pontos de montagem em modo Shell. Todas as técnicas explanadas são aplicáveis a qualquer distro GNU/Linux.

13.1. FHS – O padrão hierárquico do sistema de arquivos do GNU/Linux

FHS (Filesystem Hierarchy Standard) é o padrão estabelecido pela LSB (Linux Standard Base) o qual define a estrutura (hierarquia) de diretórios para sistemas GNU/Linux certificados por este instituto, bem como a localização de cada tipo de arquivo dentro desta estrutura. O padrão FHS é originário do sistema Unix.

Dentre as vantagens proporcionadas pelo FHS para sistemas GNU/Linux, destaco as seguintes:

- Facilitar a administração de sistemas baseados em GNU/Linux, independente da distro adotada, pois proporciona uma estrutura de diretórios padronizada.

- Proporcionar segurança ao sistema, pois arquivos binários específicos e fundamentais à administração do sistema são localizados em diretórios específicos do FHS cujo acesso e execução é restritos ao usuário root, impedindo sua utilização ou remoção indevidas.
- Facilitar o desenvolvimento de softwares para a plataforma GNU/Linux, pois define a localização de arquivos específicos como as bibliotecas compartilhadas do sistema.

Destas vantagens, a que mais nos interessa para o âmbito deste artigo é a primeira, referente à administração de sistemas GNU/Linux.

Veja na figura a seguir os diretórios do FHS abaixo do diretório raiz (/) do sistema:



13.2 –Pontos de montagem: diretórios "recipientes"

Para podermos ter acesso a dados em discos e partições (leitura/gravação) precisamos antes montar tal disco ou partição em um determinado ponto de montagem previamente criado dentro da estrutura de diretórios do FHS em um sistema GNU/Linux. As montagens poderão ser feitas manual ou automaticamente, conforme explanarei mais adiante.

Ponto de montagem é um diretório, preferentemente vazio, que será o "recipiente" do sistema de arquivos do disco ou partição ao qual queremos ter acesso.

No FHS há dois diretórios indicados para a criação de pontos de montagem:

/media/

/mnt/

A maioria dos administradores de sistemas GNU/Linux prefere utilizar o diretório /media/ para criar pontos de montagem. E eu também. Portanto, utilizaremos este diretório para a criação de pontos de montagem.

13.3 – Identificando discos e partições para montagem: o utilitário blkid

Agora que você já sabe o que são o FHS e pontos de montagem, é necessário saber identificar os dispositivos (discos e partições) detectados pelo GNU/Linux para poder montá-los. Basicamente, o GNU/Linux detecta estes dispositivos automaticamente nos seguintes momentos:

Na da inicialização do sistema são detectados os dispositivos já conectados ao sistema (coldplug: conectados a frio) ou já disponíveis no próprio hardware do computador, como é o caso de partições existentes no HD.

Após o sistema já estar inicializado, serão detectados todos os dispositivos USB, como pen drives, que forem conectados ao sistema (hotplug: conectados a quente) ou novas partições que forem criadas no HD, bem como CD/DVD's que forem inseridos.

Obs.: Note que quando você estiver utilizando um desktop (ambiente gráfico, como GNOME, KDE, XFCE, dentre outros) no GNU/Linux, a montagem de dispositivos USB (como pen drives) e CD/DVD's é, geralmente, feita automaticamente, assim que estes forem conectados ou inseridos no sistema. Para o âmbito deste artigo, considerarei que esta característica de montagem automática esteja desabilitada no sistema.

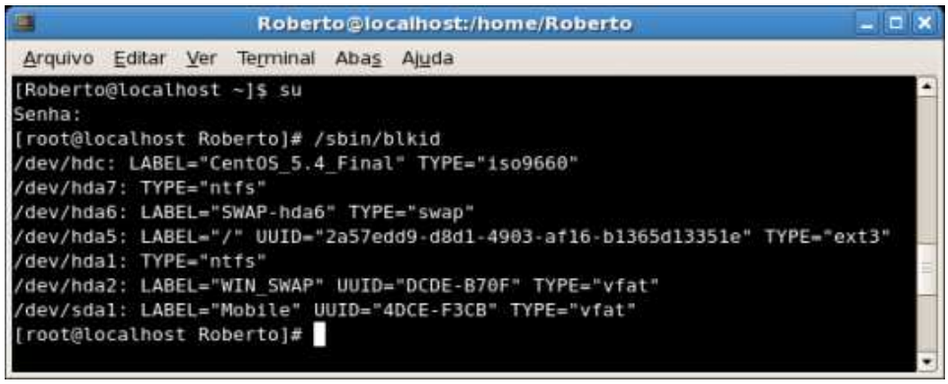
Os dispositivos detectados pelo GNU/Linux são identificados como arquivos especiais dentro do diretório /dev/ do FHS. Para montagem de um dispositivo (disco ou partição), precisaremos obter as seguintes informações sobre o mesmo:

A identificação atribuída ao dispositivo pelo GNU/Linux dentro do diretório /dev/ do FHS.

O tipo do sistema de arquivos utilizado pelo dispositivo.

Para obter estas informações, abra um Shell (Terminal) do GNU/Linux, logue-se como superusuário (su) e digite o seguinte comando:

 # /sbin/blkid



No exemplo acima, o utilitário **blkid** identificou os seguintes dispositivos montáveis em meu sistema

01 - **hda:** um HD master com cinco partições:

hda1: primeira partição com o tipo de sistema de arquivos NTFS na qual tenho o MS-Windows instalado.

hda2: segunda partição com o tipo de sistema de arquivos VFAT (FAT32) a qual é utilizada pelo MS-Windows como memória virtual.

hda5: quinta partição com o tipo de sistema de arquivos Ext3 na qual tenho o CentOS instalado.

hda6: sexta partição com o tipo de sistema de arquivos SWAP a qual é utilizada pelo CentOS como memória virtual.

hda7: sétima partição com o tipo de sistema de arquivos NTFS na qual mantenho documentos e arquivos compartilhados entre os sistemas CentOS e MS-Windows

02 – **sda:** um Pendrive (dispositivo USB) com uma única partição:

sda1: única partição deste dispositivo com o tipo de sistema de arquivos VFAT (FAT32).

03 – **hdc:** uma unidade de CD/DVD (tipo de sistema de arquivos iso9660).

13.4 –Montagens manuais de discos e partições: o comando mount

Para a montagem de discos e partições os passos a serem seguidos são:

a) Abrir o Shell (Terminal) do GNU/Linux e logar-se como superusuário (su);

b) Identificar o dispositivo que será montado e o tipo de sistema de arquivos do mesmo utilizando o utilitário **blkid**:

 # /sbin/blkid

c) Criar, caso ainda não exista, o ponto de montagem correspondente dentro do diretório /media/ do FHS com o comando **mkdir**:

 # mkdir /media/<ponto_montagem>, onde:

- <ponto_montagem>: é o ponto de montagem, isto é, o diretório "recipiente" para a montagem do dispositivo.

  Ex.: **# mkdir /media/Docs**

d) Montar o dispositivo com o comando mount cuja sintaxe geral é a seguinte:

 # mount -t <tipo_sistema_arquivos><dispositivo><ponto_montagem>, onde:

- <tipo_sistema_arquivos>: é o tipo de sistema de arquivos utilizado pelo dispositivo (disco ou partição) a ser montado; esta informação é obtida com o comando blkid.
- <dispositivo>: é identificação do dispositivo pelo GNU/Linux, por exemplo, /dev/hda1, /dev/sda1, /dev/hdc.
- <ponto_montagem>: é o ponto de montagem (diretório "recipiente") no qual o dispositivo será montado.

  Ex.: **# mount -t ntfs-3g /dev/hda7 /media/Docs**

Para que fique bem claro a montagem manual de discos e partições apresentarei os três exemplos a seguir:

Exemplo 1 – Montagem de uma partição NTFS local:

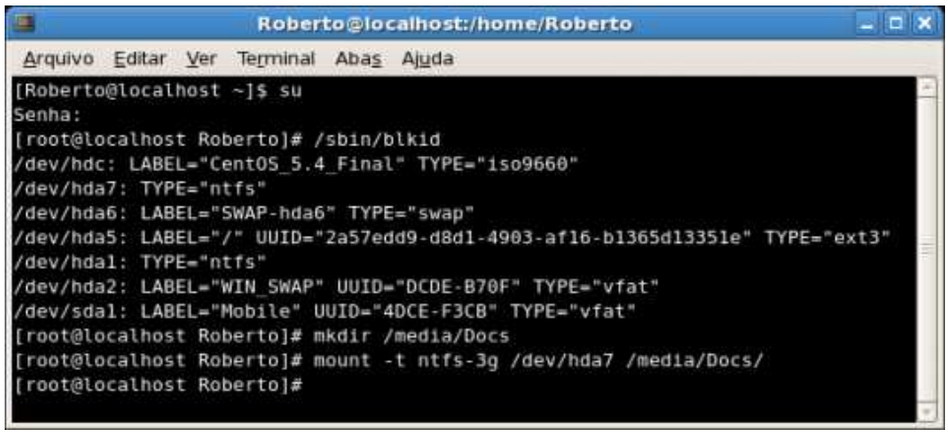
Estando logado como superusuário (su) no Shell para montar a partição /dev/hda7 identificada pelo GNU/Linux como /dev/hda7, a qual utiliza o tipo de

sistema de arquivos NTFS, em um ponto de montagem chamado Docs (/media/Docs), a sequência de comandos é a seguinte:

 # /sbin/blkid

#mkdir /media/Docs

 # mount -t ntfs-3g /dev/hda7 /media/Docs



Obs.: Note que o tipo de sistema de arquivos utilizado pelo dispositivo /dev/hda7 informado pelo utilitário blkid é NTFS. Porém, montei informando o tipo ntfs-3g. A explicação para isso é que o tipo ntfs padrão do comando mount é correspondente ao tipo NTFS utilizado pelo MS-Windows 2000. A partição em questão foi formatada utilizando o MS-Windows 7 o qual utiliza um formato NTFS mais recente.

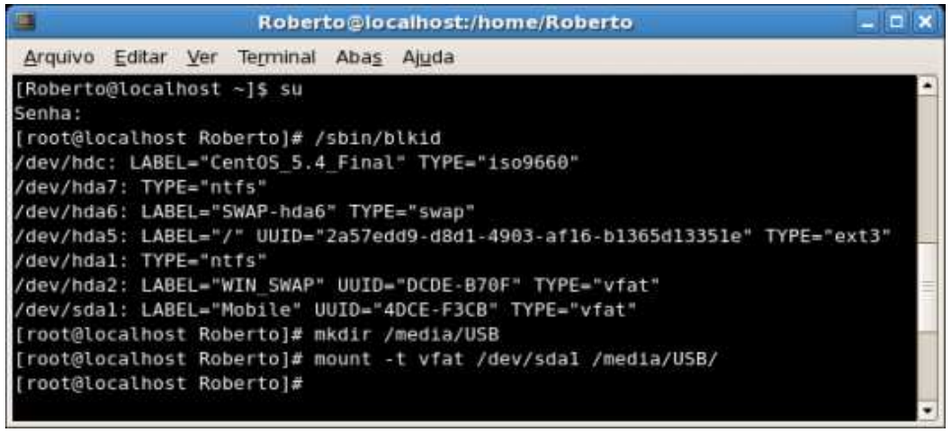
Exemplo 2 – Montagem de um pendrive (dispositivo USB):

Estando logado como superusuário (su) no Shell, para montar um pendrive identificado pelo GNU/Linux como /dev/sda1 o qual utiliza o tipo de sistema de arquivos VFAT (FAT32) em um ponto de montagem chamado USB (/media/USB), a sequência de comandos é a seguinte:

 # /sbin/blkid

 # mkdir /media/USB

 # mount -t vfat /dev/sda1 /media/USB



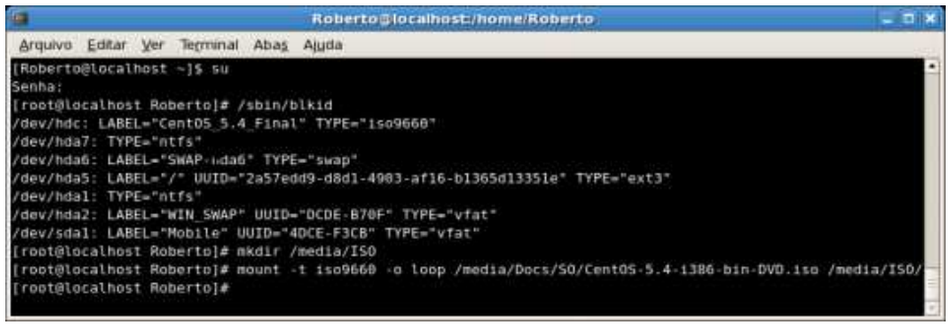
Exemplo 3 – Montagem de uma imagem ISO de DVD armazenada no HD local:

Estando logado como superusuário (su) no Shell, para montar um arquivo de imagem ISO do DVD do CentOS localizada na partição NTFS (/media/Docs/SO/CentOS-5.4-i386-bin-DVD.iso) de meu HD em um ponto de montagem chamado ISO (/media/ISO), a qual já está montada, a sequência de comandos é a seguinte:

 # /sbin/blkid

 # mkdir /media/ISO

 # mount -t iso9660 -o loop /media/Docs/SO/CentOS-5.4-i386-bin-DVD.iso /media/ISSO



Obs.: Note que o tipo de sistema de arquivos utilizado por CD's e DVD's é iso9660. Para que seja possível navegar pelos diretórios da imagem ISO como em um diretório normal no sistema é necessário utilizar o parâmetro -o loop.

13.5 – Desmontagem de discos e partições: o comando umount

Para desmontar um disco ou partição montado no sistema o comando a ser utilizado é o umount. Para executá-lo, você deverá estar logado em um Shell (Terminal) do GNU/Linux como usuário superusuário (su) e o dispositivo (disco ou partição) a ser desmontado não poderá estar em uso.

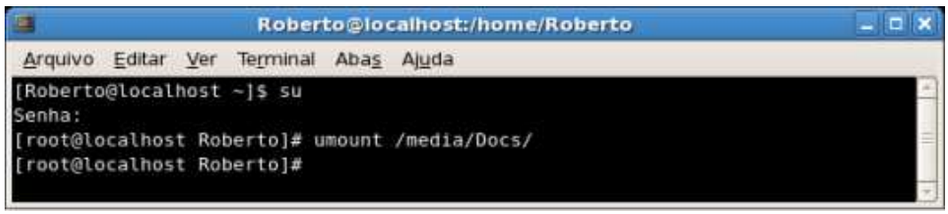
Há duas formas de desmontar um dispositivo com o comando umount:

01 – Informando o ponto de montagem a ser desmontado:

umount <ponto_montagem>, onde:

<ponto_montagem>: é o ponto de montagem (diretório “recipiente”) no qual o dispositivo (disco ou partição) foi montado.

Ex.: # umount /media/Docs

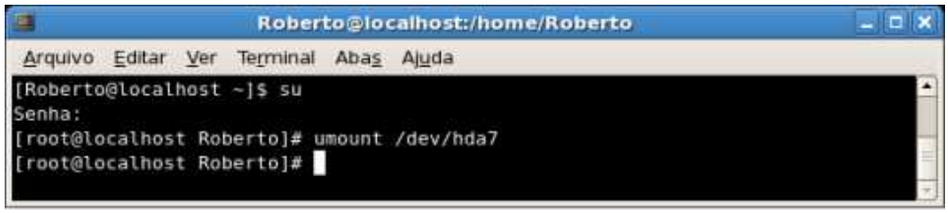


02 – Informando o dispositivo a ser desmontado:

umount <dispositivo>, onde:

<dispositivo>: é a identificação do dispositivo (disco ou partição) atribuída ao mesmo pelo GNU/Linux, a qual pode ser obtida com o utilitário blkid já explanado neste artigo.

Ex.: # /dev/hda7



14 – SMB - Server Message Block

Este é um sistema de troca de mensagens desenvolvido pela IBM para uso no seu protocolo NetBIOS e depois aperfeiçoado pela Microsoft para uso no sistema de compartilhamento de arquivos do Windows. O SMB permite que os clientes entrarem em contato com o servidor, ver quais compartilhamentos estão disponíveis, solicitar listas de diretório ou arquivos e assim por diante.

Apesar disso, a sigla é mais comumente usada é **Samba**, uma vez que por uma implementação deste protocolo (SMB), permite que máquinas Linux compartilharem e acessarem arquivos em redes Microsoft, como se fossem máquinas Windows.

Quem descobriu como usar o SMB pra comunicação entre Linux e Windows foi o Australiano Andrew Tridgell. Dizem que para ele dar nome a esta descoberta, procurou no dicionário por palavras com as letras SMB, ele encontrou e adotou a palavra SaMBa sem saber o significado dela no Brasil.

O SAMBA é um pacote Linux e portanto roda em maquinas que possuem este sistema operacional, permitindo o compartilhamento de arquivos e impressora, em redes Linux x Windows (não importa a direção), podendo configurar o Linux para utilizar uma impressora Windows ou compartilhar uma impressora Linux para estações Windows (servidor de impressão). O mesmo vale para arquivos (servidor de arquivos), compartilhando um diretório com clientes Windows.

A configuração é fácil e simples de entender, trata-se do arquivo **smb.conf** .

ALGUNS CONCEITOS IMPORTANTES

NetBios – Nome da maquina

Toda a máquina é identificada por um nome, este **nome deve ser único na rede** e permite que outras máquinas acessem os recursos disponibilizados ou que sejam enviadas mensagens para a máquina. Este nome é composto de 16 caracteres, sendo 15 que identificam a máquina e o último o tipo de serviço que ela disponibiliza

Workgroup -Grupo de trabalho

O grupo de trabalho organiza a estrutura de máquinas da rede, facilitando sua pesquisa e localização e compartilhamentos. Tomemos como exemplo uma escola

como a **utramig** com os departamentos professores, alunos, web, rh, as máquinas que pertencem ao grupo de redes serão agrupadas no programa de navegação:

alunos
 aluno1
 aluno2

rh
 mrpaoduro

web
 web1
 web2
 web3

professores
 prof1
 prof2

A segurança no acesso a arquivos e diretórios na configuração de grupo de trabalho é controlada pela própria máquina, normalmente usando segurança em nível de compartilhamento.

Domínio

O funcionamento é semelhante ao grupo de trabalho, com a diferença que a segurança é controlada pela máquina central (PDC) usando diretivas de acesso e grupos. O PDC (Primary Domain Controller) deverá ter todas as contas de acesso que serão utilizadas pelo usuário para acessar os recursos existentes em outras máquinas, script de logon que poderá ser executado em cada máquina para fazer ajustes, sincronismo, manutenção ou qualquer outra tarefa programada pelo administrador do sistema.

Compartilhamento

Um compartilhamento é um recurso da máquina local que é disponibilizado para acesso via rede, que poderá ser mapeada por outra máquina. O compartilhamento pode ser um diretório, arquivo, impressora. Além de permitir o

acesso do usuário, o compartilhamento pode ser protegido por senha, e ter controle de acesso de leitura/gravação, monitoração de acessos, diretórios ocultos, autenticação via PDC (domínio) e outras formas para restringir e garantir segurança na disponibilização dos dados. Um compartilhamento no SAMBA pode ser acessível publicamente (sem senha) ou estar rigidamente protegido tendo que passar por diversas barreiras para chegar ao seu conteúdo, como senhas, endereço de origem, interfaces, usuário autorizado, permissões de visualização, etc.

Mapeamento

Mapear significa pegar um diretório/arquivo/impressora compartilhado por alguma máquina da rede para ser acessada pela máquina local. Para mapear algum recurso de rede, é necessário que ele seja compartilhado na outra. Por exemplo, o diretório /usr compartilhado com o nome usr, pode ser mapeado por uma máquina Windows como a unidade F:, ou mapeado por uma máquina Linux no diretório /mnt/samba.

Navegação na Rede e controle de domínio

Esta função é controlada pelo **nmbd** que fica ativo o tempo todo disponibilizando os recursos da máquina na rede. A função de navegação na rede é feita utilizando o compartilhamento IPC$. Este compartilhamento possui acesso público somente leitura e utiliza o usuário **"guest"** para disponibilização de seus.

Arquivo de configuração do samba

Toda a configuração relacionada com nomes, grupo de trabalho, tipo de servidor, log, compartilhamento de arquivos e impressão do samba é colocada no arquivo de configuração **/etc./samba/smb.conf**. Este arquivo é dividido em **seções e parâmetros**. Uma seção no arquivo de configuração do samba (smb.conf) é definido por um nome entre "[]". As seções tem o objetivo de organizar os parâmetros pra que tenham efeito somente em algumas configurações de compartilhamento do servidor (exceto os da seção [global] que não especificam compartilhamentos, mas suas diretivas podem ser válidas para todos os compartilhamentos do arquivo de configuração). Alguns nomes de seções foram reservados para configurações específicas do samba:

[global]

Define configurações que afetam o servidor samba como um todo, fazendo efeito em todos os compartilhamentos existentes na máquina. Por exemplo, o grupo de trabalho, nome do servidor, página de código, restrições de acesso por nome, etc.

[homes]

Especifica opções de acesso a diretório homes de usuários. O diretório home é disponibilizado somente para seu dono, após se autenticar no sistema.

[printers]

Define opções gerais para controle das impressoras do sistema. Este compartilhamento mapeia os nomes de todas as impressoras encontradas no **/etc./printcap.** Configurações especiais podem ser feitas separadamente.

[profile]

Define um perfil quando o servidor samba é usado como PDC de domínio.

Qualquer outro nome de [seção] no arquivo smb.conf que não sejam as acima, são tratadas como um compartilhamento ou impressora.

Parâmetro

Um parâmetro é definido no formato nome = valor.

0 ou 1

yes ou no

true ou false

Assim, as seguintes configurações são equivalentes

master browse = 0

master browse = no

master browse = false

Todos significam "NÃO ser o navegador principal de domínio".

Obs: Linhas iniciadas por # ou ; são tratadas como comentário. Quebras de linha podem ser especificadas com uma \ no final da linha.

workgroup = [grupo de trabalho/domínio]

Diz qual o nome do grupo de trabalho/domínio que a máquina samba pertencerá.

netbios aliases = [nomes alternativos ao sistema]

Permite o uso de nomes alternativos ao servidor, separados por espaços.

server string = [identificação]

Identificação enviada do servidor samba para o ambiente de rede.

Restrições de acesso/mapeamento de usuários

guest account = [conta]

Define a conta local de usuário que será mapeada quando um usuário se conectar sem senha (usuário guest)

invalid users

Define uma lista de usuários que não terão acesso aos recursos do servidor ou compartilhamento.

valid users

Semelhante à opção invalid users mas permite que somente os usuários especificados tenham acesso ao sistema.

default service = nome

Caso o serviço que o usuário deseja se conectar não for encontrado no servidor, o SAMBA mapeará o serviço especificado nesta diretiva como alternativa.

Para ficar mais claro vamos criar fazer um exemplo

File Server

Primeiramente vamos instalar o pacote Samba no Ubunto Server, via terminal.

```
sudo apt-get install samba
```

Depois de terminada a instalação é hora de configurar o Samba, o principal arquivo de configuração esta localizado no diretório **/etc./samba/smb.conf .**

```
#Arquivo de configuração do samba no Ubuntu server
[global]
#nome do grupo de trabalho
workgroup = Exemplo
#como a maquina irá aparecer na rede Windows
```

```
netbios name = servidor de arquivos
```

#modo de acesso ao servidor

```
security = share
```

#compartilhamentos

```
[share]
```

```
1. comment = Ubuntu File Server Share
2. path = /srv/samba/share
3. browsable = yes
4. guest ok = yes
5. read only = no
6. create mask = 0755
```

1 - descrição do compartilhamento

2 – caminho do diretório a compartilhar.

3 – permite que o usuário procure o diretório, por exemplo, pelo Windows Explorer.

4 – permite aos clientes se conectarem ao diretório se autenticação (senha).

5 – determina se o cliente possui aceso somente para leitura.

6 – Determina a permissão relacionada a novos arquivos.

Agora que o Samba foi configurado é necessário criar o diretório e alterar as permissões.

```
7.sudo mkdir -p /srv/samba/share
```

```
sudo chown nobody.nogroup /srv/samba/share/
```

7 - Cria o diretório caso ele não exista.

```
sudo restart smbd
```

sudo restart nmbd

Após isso verifique se as configurações de rede do Windows esta correta (em caso de duvida consulte apostila de Ambiente Operacional 2) e se o Windows consegue enxergar o diretório compartilhado.

Caso queira que a pasta possua restrições de acesso, pode-se cadastrar um usuário do no samba como no exemplo abaixo:

sudo smbpasswd -a nomedousuario

Este comando vai pedir uma senha e a confirmação desta senha.

Agora temos que escolher uma pasta dentro do seu /home para o compartilhamento. Se preferir criar a pasta em questão, faça no terminal:

mkdir /home/nome_do_usuario/nome_da_pasta

Agora vamos dar permissão de leitura e escrita.

sudo chmod -R 777 /home/nome_do_usuario/nome_da_pasta

Agora vamos deixar claro que esta pasta pertence ao seu usuário e ao seu grupo de usuário:

sudo chown -R nome_do_usuario.nome_do_usuario /home/nome_do_usuario/nome_da_pasta

Agora é hora da sua configuração

sudo nano /etc./samba/smb.conf

```
[global]
        netbios aliases = nome_que_você_quer_que_apareça_na_rede
        server string = nome_que_você_quer_que_apareça_na_rede
        interfaces = eth0
        bind interfaces only = Yes
        log file = /var/log/samba/log.%m
        max log size = 50
        socket options = TCP_NODELAY SO_SNDBUF=8192 SO_RCVBUF=8192
        cups options = raw
[Nome_da_Pasta]
        path = /home/nome_do_usuario/nome_da_pasta/
        valid users = nome_do_usuario
        admin users = nome_do_usuario
        read list = nome_do_usuario
        write list =  nome_do_usuario
```

15 - INTERPRETADOR DE COMANDOS

Também conhecido como “shell”. É o programa responsável em interpretar as instruções enviadas pelo usuário e seus programas ao sistema operacional (o kernel). Ele que executa comandos lidos do dispositivo de entrada padrão (teclado) ou de um arquivo executável. É a principal ligação entre o usuário, os programas e o kernel. O GNU/Linux possui diversos tipos de interpretadores de comandos, entre eles podemos destacar o bash, ash, csh, tcsh, sh,etc. Entre eles o mais usado é o bash. O interpretador de comandos do DOS, por exemplo, é ocommand.com.

Os comandos podem ser enviados de duas maneiras para o interpretador: interativa e não interativa:

- **Interativa** Os comandos são digitados no aviso de comando e passados ao interpretador de comandos um a um. Neste modo, o computador depende do usuário para executar uma tarefa, ou próximo comando.

- **Não interativa** São usados arquivos de comandos criados pelo usuário (scripts) para o computador executar os comandos na ordem encontrada no arquivo. Neste modo, o computador executa os comandos do arquivo um por um e dependendo do término do comando, o script pode checar qual será o próximo comando que será executado e dar continuidade ao processamento.

Este sistema é útil quando temos que digitar por várias vezes seguidas um mesmo comando ou para compilar algum programa complexo.

15.1 – Arquivo Padrão

Quando um programa é executado ele "herda" três arquivos abertos:

• stdin – entrada padrão (descritor 0)
• stdout – saída padrão (descritor 1)
• stderr – saída de erros (descritor 2)

No modo interativo, o shell associa todos estes arquivos ao terminal do usuário. Mais adiante veremos como enviar o resultado do programa para um arquivo ou impressora e como ler os dados de um arquivo.

O shell é um programa como qualquer outro, lê a entrada padrão e escreve na saída padrão. Quando você digita o nome de um programa e o shell o executa, o shell para de ler e escrever e passa os arquivos padrão (entrada, saída e erros) para o programa.

Tudo o que você digitar será lido pelo programa. E o que o programa escrever aparecerá na tela. Quando o programa terminar, o controle do computador retornará ao shell que voltará a ler e escrever no terminal.

Vamos a um exemplo:

```
mimbar:~:2> wc
Procurando bem
Todo mumto tem pereba
Marca de bexiga ou vacina
E tem piriri, tem lombriga, tem ameba
^D
      2      18     101
mimbar:~:3>
```

O shell estava esperando um comando (indicado pelo prompt “mimbar:~:2>“), você digitou wc. O programa wc foi executado, leu às primeiras quatro linhas, o Ctrl-D (^D) que indica fim de texto, e imprimiu os números 4 (número de linhas no texto), 18 (número de palavras) e 101 (número de caracteres), e terminou. O shell voltou a esperar um comando (indicado pelo prompt).

Uma característica de muitos dos utilitários do Unix é que eles são escritos como filtros.

Os Filtros leem a entrada padrão, processam cada linha do arquivo e escrevem os resultados na saída padrão. Eventuais erros são enviados para a saída de erros.

15.2 - Histórico de Comandos

O bash armazena os últimos comandos que você digitou. Através do recurso de histórico de comandos você pode reexecuta-los, com ou sem alteração. Isto é muito útil quando os comandos são mais longos.

Dependendo da configuração do seu terminal, as setinhas para cima e para baixo “andam” no histórico de comandos. As setinhas para a esquerda e para a direita “andam” dentro da linha. Para inserir um texto, basta digitá-lo. Para apagar um texto use a tecla DEL ou BACKSPACK ou ^U (Ctrl-U).

O comando history lista a relação de últimos comandos. Você pode repetir o último deles teclando **!.** Pode repetir um comando arbitrário digitando !nnn onde nnn é o número do comando na lista. Ou pode utilizar !?palavra para repetir o último comando que contém a palavra.

Veja a seguir um exemplo:

```
mimbar:~:12> cd /usr/share/linux.see
mimbar:/usr/share/linux.see:13> echo $PS1
\h:\w:\#>
mimbar:/usr/share/linux.see:14> date
Wed Dec 14 22:52:53 BRST 2005
mimbar:/usr/share/linux.see:15> ls
Bilhetes            assinatura.tcl  dominios.txt      rascunho.txt
Lista-de-Palavras.txt  campeoes.txt     doms         revision-1.txt
MaryPoppins-1.txt   capitais  frutas         revision-2.txt
MaryPoppins-2.txt   cidades   mlang10.zip    sigla-capital
Receitas            citacao   morse.txt      sigla-estado
assinatura.pl          dict-port.txt    mword10.zip
assinatura.sh          discurso.txt     pequeno.dicionario
mimbar:/usr/share/linux.see:16> history 5
 512   cd /usr/share/linux.see
 513   echo $PS1
 514   date
 515   ls
 516   history 5
mimbar:/usr/share/linux.see:17> !date
date
Wed Dec 14 22:53:14 BRST 2005
mimbar:/usr/share/linux.see:18> !?see
cd /usr/share/linux.see
mimbar:/usr/share/linux.see:19> !513
echo $PS1
\h:\w:\#>
```

history 5 lista os últimos 5 comandos.

!date reexecuta o ultimo comando date

!?see reexecuta a ultima linha que contem o texto **see**

Cuidado com a repetição de comandos, principalmente o !? porque você pode estar no diretório errado para o comando!

15.3 - Completar Nomes

Você pode digitar apenas o início do nome do programa ou arquivo e o shell o completa para você. Para isto você digita o início do nome e pressiona TAB. Se só existir uma alternativa o shell completa a palavra. Se existir mais de uma soará um beep. Neste caso, tecle TAB duas vezes para ver a lista de alternativas, digite mais alguns caracteres e pressione TAB novamente.

Quando se pressiona TAB na primeira palavra da linha de comando (o nome do programa a executar) o shell tenta fazer a expansão examinando cada alternativa na variável **$PATH** (esta e outras variáveis são descritas logo abaixo).

Após a primeira palavra o shell tenta expandir nomes de arquivos no diretório corrente.

Por exemplo, digite “cd /etc.” (sem as aspas). Depois digite “echo pasTAB”. O shell completará o comando como “echo passwd”, porque o único arquivo que existe no diretório /etc. que começa com “pas” é “passwd”. Experimente.

Observação: quando o shell expande um nome de diretório, com o recurso de completar nomes, ele coloca uma barra (/) para facilitar a digitação de um nome de arquivo.

15.4 – Variáveis

Pode-se imaginar uma variável como uma caixa onde se pode guardar um texto. Como pode existir mais de uma caixa, cada caixa tem um nome único. Por exemplo, você pode ter uma caixa azul onde guarda a conta de telefone e uma caixa vermelha onde guarda o telefone da padaria.

Seguindo a metáfora da caixa, você pode pedir a alguém: “me passe a caixa vermelha” e “me passe o conteúdo da caixa vermelha”. São duas coisas bem diferentes.

No shell, para você colocar algo em uma variável você utiliza um comando da forma:

```
nome=value
```

Exemplo:

```
mimbar:~:117>  PADARIA=”3261-3460"
```

Para obter o valor de uma variável você coloca um caractere dollar/cifrão ($) antes do nome da variável. Veja:

```
mimbar:~:118>  echo  PADARIA
PADARIA
mimbar:~:119>  echo  $PADARIA
3261-3460
```

E quando usamos variáveis? Quando queremos utilizar um texto mais de uma vez. Se vamos utilizar o mesmo texto diversas vezes, podemos coloca-lo em uma variável e referenciar a variável diversas vezes.

Por fim, duas observações sobre variáveis:

• Elas são efêmeras, ou seja, quando o shell termina (ou sua sessão) as variáveis são perdidas .

• A qualquer momento você pode alterar o valor de uma variável, afinal ela não é uma

constante :-)

15.5 - Variáveis do Shell

O shell suporta e utiliza diversas variáveis. Algumas variáveis importantes são:

PS1 indicador que o shell esta pronto para receber um comando (prompt)

HOME diretório principal do usuário

PATH relação de diretórios onde estão os programas

TERM definição do terminal

Além destas, existem outras variáveis definidas no ambiente que são utilizados por outros programas. Por exemplo,

EDITOR editor preferido pelo usuário (ver emacs e vi)

PAGER programa de paginação (ver more & less)

O usuário pode definir suas próprias variáveis, desde que não conflitem com as utilizadas pelo sistema. Por convenção, as variáveis do sistema são escritas sempre em maiúsculas.

As variáveis podem ser utilizadas para passar informações entre programas ou para armazenar valores temporários.

Você pode listar todas as variáveis definidas no ambiente através do programa 'env'. Ou listar individualmente com o comando 'echo $variável'.

Exemplo:

```
mimbar:~:1#  env
HZ=100
SHELL=/bin/bash
TERM=xterm
USER=root
MAIL=/var/mail/root
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/
bin:/usr/bin/X11
PWD=/root
PS1=\h:\w:\##
SHLVL=1
HOME=/root
LOGNAME=root
_=/usr/bin/env
mimbar:~:2#
```

15.6 - Entrada e Saída

Muitos programas recebem seus dados via entrada padrão e escrevem os resultados na saída padrão. Porém nem sempre isto é desejado, por exemplo, seus dados podem estar gravados em um arquivo. Para estas situações o shell implementa o redirecionamento. Através do redirecionamento o programa lê ou escreve em arquivos definidos pelo usuário. E isto sem que o programa perceba.

As formas mais comuns de redirecionamento são:

< da entrada padrão

> da saída padrão, sobrescreve

>> da saída padrão, adiciona

2> da saída de erros

| da saída padrão para a entrada padrão (pipe)

Vamos dar alguns exemplos utilizando os comandos **echo, cat, wc e date**.

echo imprime os argumentos.

cat lista o conteúdo de um arquivo.

wc conta as linhas, palavras e caracteres de um arquivo.

date imprime a data corrente.

```
mimbar:~/local/wikit/sh-exemplos:18> ls -l
total 0
mimbar:~/local/wikit/sh-exemplos:19> echo era uma vez...
era uma vez...
mimbar:~/local/wikit/sh-exemplos:20> echo era uma vez...
>historia
mimbar:~/local/wikit/sh-exemplos:21> ls -l
total 4
-rw-r--r--  1 edesio edesio 15 Feb 19 21:36 historia
mimbar:~/local/wikit/sh-exemplos:22> cat historia
era uma vez...
mimbar:~/local/wikit/sh-exemplos:23> wc historia
 1  3 15 historia
mimbar:~/local/wikit/sh-exemplos:24> echo era uma vez... | wc
      1       3      15
mimbar:~/local/wikit/sh-exemplos:25> cat <historia
era uma vez...
mimbar:~/local/wikit/sh-exemplos:26> date >>historia
mimbar:~/local/wikit/sh-exemplos:27> cat historia
```

```
era uma vez...
Sun Feb 19 21:38:06 BRT 2006
mimbar:~/local/wikit/sh-exemplos:28> date >historia
mimbar:~/local/wikit/sh-exemplos:29> cat historia
Sun Feb 19 21:38:16 BRT 2006
mimbar:~/local/wikit/sh-exemplos:30> rm historia
mimbar:~/local/wikit/sh-exemplos:31> cat historia
cat: historia: No such file or directory
mimbar:~/local/wikit/sh-exemplos:32> cat historia 2>erros
mimbar:~/local/wikit/sh-exemplos:33> ls -l
total 4
-rw-r--r--  1 edesio edesio 41 Feb 19 21:44 erros
mimbar:~/local/wikit/sh-exemplos:34> cat erros
cat: historia: No such file or directory
mimbar:~/local/wikit/sh-exemplos:35> cd ..
mimbar:~/local/wikit:36> ls -1
CVS
Index.txt
Pages
dump.log
error.txt
history
linux.see
notes.tkd
rascunho.txt
savelog.txt
sh-exemplos
should-not-be-here-linux.see
tmp
wikit.copia
wikit.tkd
mimbar:~/local/wikit:37> ls -1 | wc -l
15
mimbar:~/local/wikit:38>
```

O que aconteceu acima? Vejamos usando o número do comando como referência:

#18 o diretório está vazio.

#19 usamos o echo que escreve na saída padrão.

#20 redirecionamos a saída o echo para o arquivo historia.

#21 dados do arquivo criado.

#22 usando o cat para exibir o conteúdo do arquivo criado.

#23 contando as linhas, palavras e caracteres do arquivo.

#24 usando o pipe ao invés de um arquivo intermediário.

#25 o cat listando a entrada padrão (porque foi executado sem argumentos).

#26 colocando a data atual no final do arquivo de historia (adicionado).

#27 novo conteúdo do arquivo historia.

#28 sobrescrevendo o arquivo historia, o conteúdo anterior é perdido.

#29 novo conteúdo do arquivo historia.

#30 removendo o arquivo criado.
#31 tentando exibir o conteúdo de um arquivo inexistente (erro)
#32 a mensagem de erro é escrita no arquivo erro
#33 conteúdo do diretório
#34 conteúdo do arquivo erro
#35 movendo para o diretório acima
#36 listando os arquivos do diretório
#37 contando os arquivos do diretório

15.7 - Utilitários de manipulação de arquivos e diretórios

1. **ls** listar conteúdo de diretório
2. **find** procurar arquivos
3. **ln** criar um "sinônimo" para um arquivo
4. **cp & mv & rm** copiar, renomear e remover arquivos
5. **chown & chmod** alterar dono e permissões dos arquivos
6. **file** informar o tipo de um arquivo examinando seu conteúdo
7. **du & df** utilização de espaço em disco
8. **mkdir & rmdir** criar e remover diretórios
9. **cmp** comparar dois arquivos

15.8 Find

Find quer dizer ache/procure. Este utilitário procura um ou mais arquivos ou diretórios dentro do sistema de arquivos. Ele é flexível para pesquisar por diversos atributos.

As principais opções são:

name padrão – procura por padrão

size tamanho – procura por arquivos de um determinado tamanho

xdev – não muda de sistema de arquivos

follow – segue os links simbólicos (symlinks)

type – tipo de arquivo (arquivo, diretório, symlinks, dispositivos)

maxdepth n – permite limitar a profundidade da pesquisa

16 – ENTENDENDO E CONFIGURANDO O FIREWALL (SHOREWALL)

Montar um firewall para a proteção de uma única máquina não é tarefa das mais simples, imagine montar um firewall para uma rede inteira, que requer muito mais atenção e conhecimento, tanto dos serviços quanto das ferramentas utilizadas. Qualquer descuido pode comprometer toda a segurança da rede.

No caso específico do Linux, trabalhar com o Iptables na linha de comando torna as coisas um pouco mais complicadas, principalmente para usuários novatos ou acostumados apenas com cliques na interface gráfica. O domínio de seus comandos para uma utilização eficiente e segura em situações mais avançadas exige bastante experiência por parte do usuário. Pensando nisso, **Thomas M. Eastep** teve a brilhante ideia de desenvolver uma ferramenta que funcionasse como uma camada entre o administrador de rede e o prolixo mas poderoso Iptables: o **Shorewall**, é a ferramenta utilizada como solução para firewall. Na inicialização do serviço, essas regras serão compiladas e convertidas em regras do Iptables, que, no final das contas, fará a função de firewall. Ou seja, o Shorewall foi feito com o objetivo de facilitar a criação e a manutenção de regras para o Iptables e a vida do administrador de rede.

Antes de começarmos vamos falar um pouco de alguns conceitos importantes.

Zonas

A zonas representam as redes que estarão conectadas ao servidor e o próprio servidor (firewall), por exemplo, a rede local será representada pela zona loc, a Internet será representada pela zona net, o servidor será representado pela zona fw. Caso pretenda adicionar uma DMZ ao servidor, devemos criar a zona dmz, o importante é representar cada rede com uma zona, mais adiante isso ficará mais claro pra você.

Para verificar quais as zonas foram definidas no servidor após a instalação da solução, acesse **opl > Firewall > Configurações Avançadas > Zonas**. Caso queira saber mais sobre o arquivo de configuração das zonas digite **man shorewall-zones** no terminal.

Interfaces

Cada zona criada anteriormente deverá ser associada a uma interface (placa de rede) do servidor (aquela em que a rede da zona estará conectada). Para nós a zona net estará associada à interface **eht0,** ou seja, o roteador ou modem ADSL deverá ser conectado à placa de rede **eth0** do servidor, e a zona **loc** estará associada à interface **eth1**. A zona fw não precisa ser associada a nenhuma interface, pois será o próprio servidor.

Para verificar como as interfaces foram definidas no servidor após a instalação da solução, acesse **opl > Firewall > Configurações Avançadas > Interfaces.** Caso queira saber mais sobre o arquivo de configuração das interfaces digite **man shorewall-interfaces** no terminal.

Políticas

Politica padrão de acesso entre as zonas, onde ACCEPT significa que o acesso entre as zonas será livre e REJECT e DROP o acesso será bloqueado, sendo que no primeiro haverá respostas ao solicitante e no segundo nada será informado.

Para verificar como as políticas foram definidas no servidor após a instalação da solução, acesse **opl > Firewall > Configurações Avançadas > Políticas (policy).** Caso queira saber mais sobre o arquivo de configuração das políticas digite **man shorewall-policy** no terminal.

Mascaramento

Agora iremos definir para quais interfaces será permitido o mascaramento ou encaminhamento de pacotes, graças a esta configuração será possível realizar **NAT/DNAT** através do servidor.

Para verificar o mascaramento (masq) definido no servidor após a instalação da solução, acesse **opl > Firewall > Configurações Avançadas > Mascaramento (Masq).** Repare que a interface da zona net, no nosso caso a eth0, deverá sempre ser a primeira, seguida pela interface da zona que se quer ativar o mascaramento.

Caso queira saber mais sobre o arquivo de configuração do mascaramento, digite **man shorewall-masq** no terminal.

16.1 Configurando as regras do firewall

Todas as configurações realizadas até aqui somente deverão ser feitas uma única vez, a não ser que haja alguma alteração na infraestrutura da rede, como a adição de uma DMZ. Mas as alterações realizadas no arquivo de configuração das regras (opl > Firewall > Editar Regras do Firewall) serão frequentes e necessárias sempre que se desejar a liberar algum acesso que não foi previsto pela solução, por isso daremos maior atenção a essa configuração.

Após a instalação das soluções **OpçãoLinux PDC** e **E-jovem Proxy,** o firewall (Shorewall) estará pré-configurado para permitir o acesso a todos os serviços em execução no servidor, no entanto, devido à forma como foram configuradas as políticas de acesso no arquivo policy (DROPs e REJECTs), o tráfego entre as zonas estará bloqueado para os demais protocolos e portas, logo, qualquer liberação de acesso não prevista na configuração inicial deverá ser configurada nesse arquivo, por isso o perfeito entendimento da sua sintaxe é essencial para a configuração do firewall.

16.2 - Organização do arquivo de regras

Abra o arquivo, observe as regras pré-configuradas e procure entender como ele está organizado, repare que as regras foram dividias pelo tipo de acesso: loc para net, loc para fw etc. É muito importante procurar manter essa organização, embora não seja obrigatório, pois assim será mais fácil ter o controle sobre a configuração do firewall. Repare também que não será necessário criar regras do firewall para as demais zonas, uma vez que a zona fw teve sua política definida como ACCEPT para as demais zonas. O arquivo de regra será lido de forma sequencial, assim as regras que estiverem mais acima terão precedência em relação às que estiverem mais embaixo.

O Shorewall vem com várias regras (macros) predefinidas e isso facilita bastante à configuração, pois não será necessário saber quais são os protocolos e portas de determinados serviços na definição das regras, basta usar a macro

correspondente ao serviço, caso exista uma. Para saber quais são as macros disponibilizadas pelo Shorewall, basta acessar o diretório **"/usr/share/shorewall"** no servidor. Cada macro corresponde a um arquivo com o nome **macro.SERVIÇO**. Utilizamos muitas delas na configuração do firewall.

16.3 - Regas do firewall

A melhor maneira de ensinar e aprender como configurar o firewall é com exemplos práticos, vamos mostrar isso agora, mas lembre-se que, para o perfeito entendimento dos exemplos que serão mostrados aqui, é necessário saber que protocolos e portas são utilizados pelos serviços que se quer configurar, e ensinar isso foge do objetivo deste tutorial.

Vamos analisar uma das regras configuradas da rede local para o firewall (loc para fw). Observe que a primeira regra definida no arquivo de configuração é esta listada abaixo:

```
DNS/ACCEPTloc     fw
```

Esta regra está permitindo o acesso ao serviço DNS que está em execução no firewall a todas as estações da rede local. Repare que não definimos o protocolo (tcp ou udp) nem a porta (53) usados pelo serviço, isso porque utilizamos uma macro fornecida pelo Shorewall, caso contrário a regra seria assim:

```
ACCEPT     loc    fw    tcp    53
ACCEPT     loc    fw    udp    53
```

Repare que agora foi necessário definir duas regras, uma para o protocolo **tcp** e outra para o protocolo **udp**, mas com o uso da macro DNS bastou criar uma única regra. Macros são bastante úteis quando temos serviços que utilizam várias portas e protocolos como o Samba.

Não será necessário criar regras para o tráfego de retorno (sentido inverso), como ocorreria se estivéssemos configurando o Iptables na mão, isso será feito automaticamente pelo Shorewall.

Agora você já tem como entender a maioria das regras pré-configuradas na solução que instalou no seu servidor, aproveite e dê uma nova vistoriada no arquivo de configuração. Mas vamos estender um pouco mais o nosso exemplo.

Suponhamos agora que você precise liberar acesso ao serviço SMTP na Internet a todas as máquinas da sua rede local, para isso devemos adicionar a regra abaixo:

```
SMTP/ACCEPTloc     net
```

Sem o uso de macros teríamos:

```
ACCEPT       loc     nettcp 25
```

E se quiséssemos limitar essa regra apenas para a estação da rede local com o endereço 192.168.0.100? Basta adicionar dois pontos e o endereço da estação após o nome da zona *loc*.

```
SMTP/ACCEPT   loc:192.168.0.100     net
ou
ACCEPT       loc:192.168.0.100     net    tcp    25
```

Para adicionar mais estações à regra, pode-se criar uma nova regra para cada estação ou inserir o endereço precedido de vírgula após o endereço existente (sem espaços):

```
SMTP/ACCEPT   loc:192.168.0.100,192.168.0.101     net
```

ou

```
ACCEPT       loc:192.168.0.100,192.168.0.101     net    tcp    25
```

Caso quiséssemos restringir o acesso a apenas um host específico na Internet? pois da maneira como a regra foi definida acima, as estações locais terão acesso ao serviço SMTP de todas as máquinas da Internet. Nesse caso, adicione o endereço do host após no nome da zona net:

```
SMTP/ACCEPTloc:192.168.0.100,192.168.0.101     net:200.200.200.200
```

E para dar acesso ao serviço SMTP do host 200.200.200.200 a todas as estações rede local, devemos definir a regra abaixo:

```
SMTP/ACCEPTloc     net:200.200.200.200
```

Podemos também definir uma faixa de portas que serão abertas.

Por exemplo: (1024 a 65535) do protocolo tcp para estação 192.168.0.100 na Internet:

```
ACCEPT       loc:192.168.0.100     net    tcp    1024:65535
```

Até aqui definimos apenas regras da zona *loc* para a zona *net*, mas devem ser definidas também regras para as demais zonas se necessário, por exemplo, suponhamos que você tenha instalado o serviço Web no servidor e queira dar acesso tanto à rede local quanto à Internet, será necessário então criar duas regras no firewall:

```
Web/ACCEPT   loc     fw
```

e

```
Web/ACCEPT   net     fw
```

Crie regras apenas para os serviços que deseje permitir, se no exemplo acima o serviço Web fosse apenas para a rede local, não seria necessário, nem prudente, criar a segunda regra.

16.4 - Redirecionamento e Proxy Transparente

Vamos falar agora sobre o redirecionamento que é utilizado pelo Proxy Transparente, isso significa que os pacotes com destino à porta 80 (http) serão desviados automaticamente pelo firewall para o serviço Proxy na porta 3128 (Squid), isso é feito através do comando mostrado abaixo:

```
REDIRECTloc   3128   tcp   http   -   -
```

Dessa maneira, nenhum acesso à porta 80 (http) será feito diretamente pelas estações, tudo passará pelo proxy e será controlado pelas suas regras de acesso. No entanto, se precisarmos dar acesso direto à porta 80 a uma determinada estação? Neste caso temos que fazer duas coisas, remover a estação do redirecionamento para o proxy (o ponto de exclamação que funciona como exclusão):

```
REDIRECTloc:!192.168.0.1003128   tcp   http   -   -
```

Após isso, liberar o tráfego da estação para a porta 80:

```
HTTP/ACCEPTloc:192.168.0.100     net
```

Agora a estação 192.168.0.100 acessará diretamente a Internet através de NAT, sem ser desviada para o proxy, e não será mais controlada pelas regras de controle acesso do Squid, nem fará uso do cachê, o que deixará o acesso a sites na Internet mais lento.

Caso queira excluir alguns hosts da Internet do redirecionamento, fazendo com que o acesso a eles seja feito diretamente sem o uso do proxy, exclua seus endereços da regra REDIRECT:

```
REDIRECTloc 3128   tcp   http   -   !200.200.200.200,200.200.200.201
```

depois crie uma regra permitindo o acesso direto a eles:

```
Web/ACCEPTloc     net:200.200.200.200,200.200.200.201
```

Repare que macros Web e HTTP nos exemplos mostrados. No Shorewall a macro Web se refere as porta 80 e 443, já a macro HTTP se refere apenas à porta 80. Temos ainda a macro HTTPS que se refere à porta 443.

16.5 - Acesso às estações da rede local

Agora falaremos sobre o trafego da zona *net* para a zona *loc*, que é um pouco diferente dos demais. Como não temos IPs válidos nas estações da rede local, será

necessário fazer um redirecionamento das requisições ao IP válido do servidor para o endereço não válido da estação que se deseja acessar remotamente. A este tipo e conexão dar-se o nome de **DNAT**. Vejamos então outro exemplo prático, vamos supor que o endereço do servidor seja 100.100.100.100 (um endereço válido) e queiramos que o host 200.200.200.200 da Internet acesse o serviço VNC (porta 5900) na estação 192.168.0.100 da rede local. Basta definir a regra abaixo no arquivo de configuração do firewall:

```
DNAT:info  net:200.200.200.200  loc:192.168.0.100:5900 tcp 5900 - 00.100.100.100-
```

Agora, basta acessar o servidor (através do seu endereço IP) na porta 5900 que o tráfego será redirecionado para o serviço VNC, porta 5900, na estação. No entanto é necessário que o servidor tenha um endereço IP válido. Se não tiver, caso esteja utilizando um modem ADSL roteado (Velox), será necessário fazer uma configuração semelhante no modem. Assim, o tráfego será redirecionado do modem para o servidor que o redirecionará para a estação.

No entanto deve-se ter atenção, você está abrindo uma vulnerabilidade de segurança na sua rede, pois está permitindo que estações na Internet acessem a sua rede local. Caso realmente necessite dar acesso às estações da sua rede local através da Internet há outras soluções mais seguras.

16.6 - Ativando as alterações

Sempre que for feita alguma alteração nas regras do firewall, será necessário ativar as alterações: **opl > Firewall > Ativar Alterações.**

Ao fazer isso, surgirá na tela uma listagem semelhante à mostrada abaixo:

```
Compiling...
Compiling /etc./shorewall/zones...
Compiling /etc./shorewall/interfaces...
Determining Hosts in Zones...
Preprocessing Action Files...
  Pre-processing /usr/share/shorewall/action.Drop...
  Pre-processing /usr/share/shorewall/action.Reject...
Compiling /etc./shorewall/policy...
```

```
Compiling Kernel Route Filtering...
Compiling Martian Logging...
Compiling /etc./shorewall/masq...
Compiling MAC Filtration -- Phase 1...
Compiling /etc./shorewall/rules...
Generating Transitive Closure of Used-action List...
Processing /usr/share/shorewall/action.Reject for chain Reject...
Processing /usr/share/shorewall/action.Drop for chain Drop...
Compiling MAC Filtration -- Phase 2...
Applying Policies...
Generating Rule Matrix...
Creating iptables-restore input...
Shorewall configuration compiled to /var/lib/shorewall/.start
Processing /etc/shore wall/params ...
Starting Shorewall....
Initializing...
Processing /etc./shorewall/init ...
Setting up ARP filtering...
Setting up Route Filtering...
Setting up Martian Logging...
Setting up Accept Source Routing...
IP Forwarding Enabled
Setting up Proxy ARP...
Setting up Traffic Control...
Preparing iptables-restore input...
Running /sbin/iptables-restore...
Processing /etc./shorewall/start ...
Processing /etc./shorewall/started ...
done.
```

Caso alguma coisa dê errado, devido a um erro de configuração por exemplo, não será mostrada a palavra "**done**" (feito em inglês) no final da listagem. Reparem que podemos observar as duas etapas de inicialização do Shorewall: a compilação das configurações e a inicialização do firewall com a ativação das regras do Iptables.

BIBLIOGRAFIA SUGERIDA

Mota, João Eriberto Filho. **Descobrindo o Linux Entenda o sistema operacional GNU/Linux**. Novatec, 3ª Edição,2012

JUNIOR Antonio Gomes. **A COMPUTAÇÃO PESSOAL E O SISTEMA OPERACIONAL LINUX - Utilizando Debian e Gnu/Linux 4.0, Guariba** SP, julho 2007.

NEGUS, Christopher. **Linux: Edição especial - A Biblia.**

RUBEM E. Ferreira. **Linux Guia do administrador do sistema**. Novatec, 3ª Edição.

CURSO TÉCNICO EM

INFORMÁTICA

ETAPA 3

PROJETO DE QUALIDADE DE SISTEMAS DE INFORMAÇÃO

Sumário

1 SISTEMA DE CONTROLE DE VERSÃO 213
1.1 SISTEMAS DE CONTROLE DE VERSÃO LOCAIS 214
1.2 SISTEMAS DE CONTROLE DE VERSÃO CENTRALIZADOS 215
1.3 SISTEMAS DE CONTROLE DE VERSÃO DISTRIBUÍDOS 216
1.4 FUNCIONAMENTO BÁSICO 217
1.5 ENVIO E RESGATE DE VERSÕES 218

2 TESTE DE SOFTWARE 221
2.1 CONCEITOS BÁSICOS ASSOCIADOS A TESTE DE SOFTWARE 222
2.2 O QUE É TESTE DE SOFTWARE 223
2.3 O QUE NÃO É TESTE DE SOFTWARE 225
2.4 DEFEITOS NO DESENVOLVIMENTO DE SOFTWARE 226
2.5 NÍVEIS DE TESTE DE SOFTWARE 228
2.6 A IMPORTÂNCIA DE TESTES DE SOFTWARE 229

3 ANÁLISE DE PONTOS DE FUNÇÃO 231
3.1 ANÁLISE DE PONTOS DE FUNÇÃO (APF) 231
3.2 CONTAGEM DAS FUNÇÕES DE DADOS 236
3.3 CONTAGEM DAS FUNÇÕES TRANSACIONAIS 236

1 SISTEMA DE CONTROLE DE VERSÃO

Um sistema de controle de versão (ou versionamento), **VCS** (do inglês **version control system**) ou ainda **SCM** (do inglês **source code management**) na função prática da Ciência da Computação e da Engenharia de Software, é um software com a finalidade de gerenciar diferentes versões no desenvolvimento de um documento qualquer. Esses sistemas são comumente utilizados no desenvolvimento de software para controlar as diferentes versões – histórico e desenvolvimento – dos códigos-fontes e também da documentação.

Esse tipo de sistema é muito presente em empresas e instituições de tecnologia e desenvolvimento de software. É também muito comum no desenvolvimento de software livre. É útil, em diversos aspectos, tanto para projetos pessoais pequenos e simples como também para grandes projetos comerciais.

Entre os mais comuns encontram-se as soluções livres: CVS, Mercurial, Git e SVN; e as comerciais: SourceSafe, PVCS (Serena) e ClearCase. O desenvolvimento de software livre prefere o GIT ou SVN que vem substituindo o clássico CVS. Muitas empresas também adotam o SVN, embora algumas empresas prefiram uma solução comercial, optando pelo ClearCase (da IBM) ou SourceSafe (da Microsoft). Optar por uma solução comercial geralmente está relacionado à garantia, pois as soluções livres não se responsabilizam por erros no software e perdas de informações, apesar das soluções livres poderem ter melhor desempenho e segurança que as comerciais. As soluções comerciais apesar de supostas garantias adicionais não garantem o sucesso da implementação nem indenizam por qualquer tipo de erro mesmo que comprovadamente advindo do software.

As principais vantagens de se utilizar um sistema de controle de versão para rastrear as alterações feitas durante o desenvolvimento de software ou o desenvolvimento de um documento de texto qualquer são:

- **Controle do histórico:** facilidade em desfazer e possibilidade de analisar o histórico do desenvolvimento, como também facilidade no resgate de versões mais antigas e estáveis. A maioria das implementações permitem analisar as alterações com detalhes, desde a primeira versão até a última.

- **Trabalho em equipe:** um sistema de controle de versão permite que diversas pessoas trabalhem sobre o mesmo conjunto de documentos ao mesmo tempo e minimiza o desgaste provocado por problemas com conflitos de edições. É possível que a implementação também tenha um controle sofisticado de acesso para cada usuário ou grupo de usuários.

- **Marcação e resgate de versões estáveis:** a maioria dos sistemas permite marcar onde é que o documento estava com uma versão estável, podendo ser facilmente resgatado no futuro.

- **Ramificação de projeto:** a maioria das implementações possibilita a divisão do projeto em várias linhas de desenvolvimento, que podem ser trabalhadas paralelamente, sem que uma interfira na outra.

1.1 Sistemas de Controle de Versão Locais

O método preferido de controle de versão por muitas pessoas é copiar arquivos em outro diretório (talvez uns diretórios com data e hora se forem espertos). Esta abordagem é muito comum por ser tão simples, mas é também muito suscetível a erros. É fácil esquecer em qual diretório você está e gravar acidentalmente no arquivo errado ou sobrescrever arquivos sem querer.

Para lidar com esse problema, alguns programadores desenvolveram há muito tempo VCSs locais que armazenavam todas as alterações dos arquivos sob controle de revisão, como demonstra a figura 3.1.

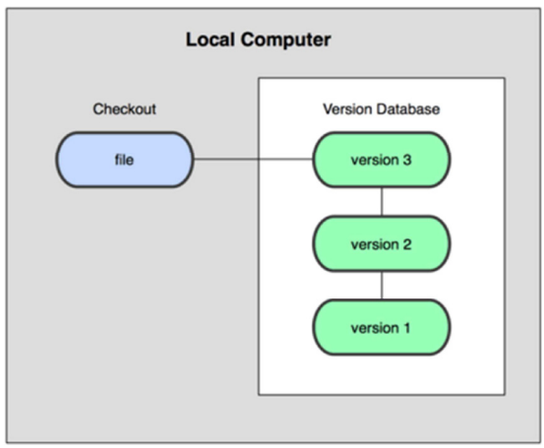


Figura 1-1. Diagrama de controle de versão local.

Uma das ferramentas de VCS mais populares foi um sistema chamado rcs, que ainda é distribuído em muitos computadores até hoje. Até o popular Mac OS X inclui o comando *rcs* quando se instala o kit de ferramentas para desenvolvedores. Basicamente, essa ferramenta mantém conjuntos de patches (ou seja, as diferenças entre os arquivos) entre cada mudança em um formato especial; a partir daí qualquer arquivo em qualquer ponto na linha do tempo pode ser recriado ao juntarem-se todos os patches.

1.2 Sistemas de Controle de Versão Centralizados

Outro grande problema que as pessoas encontram estava na necessidade de trabalhar em conjunto com outros desenvolvedores, que usam outros sistemas. Para lidar com isso, foram desenvolvidos Sistemas de Controle de Versão Centralizados (Centralized Version Control System ou CVCS). Esses sistemas, como por exemplo, o CVS, Subversion e Perforce, possuem um único servidor central que contém todos os arquivos versionados e vários clientes que podem resgatar (check out) os arquivos do servidor. Por muitos anos, esse foi o modelo padrão para controle de versão.

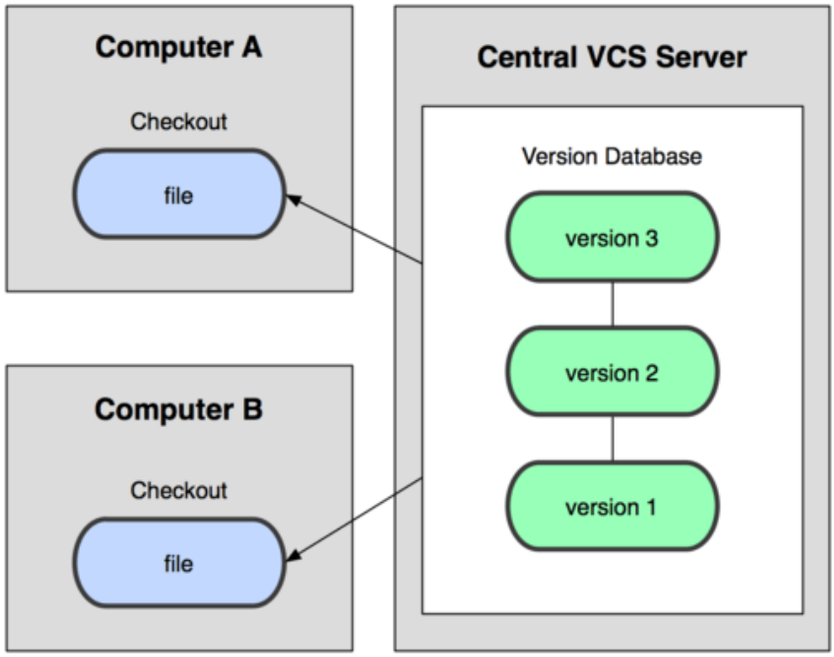


Figura 3.2. Diagrama de Controle de Versão Centralizado.

Tal arranjo oferece muitas vantagens, especialmente sobre VCSs locais. Por exemplo, todo mundo pode ter conhecimento razoável sobre o que os outros

desenvolvedores estão fazendo no projeto. Administradores têm controle específico sobre quem faz o quê; sem falar que é bem mais fácil administrar um CVCS do que lidar com bancos de dados locais em cada cliente.

Entretanto, esse arranjo também possui grandes desvantagens. O mais óbvio é que o servidor central é um ponto único de falha. Se o servidor ficar fora do ar por uma hora, ninguém pode trabalhar em conjunto ou salvar novas versões dos arquivos durante esse período. Se o disco do servidor do banco de dados for corrompido e não existir um backup adequado, perde-se tudo – todo o histórico de mudanças no projeto, exceto pelas únicas cópias que os desenvolvedores possuem em suas máquinas locais. VCSs locais também sofrem desse problema – sempre que se tem o histórico em um único local, corre-se o risco de perder tudo.

1.3 Sistemas de Controle de Versão Distribuídos

É aí que surgem os Sistemas de Controle de Versão Distribuídos (Distributed Version Control System ou DVCS). Em um DVCS (tais como Git, Mercurial, Bazaar or Darcs), os clientes não apenas fazem cópias das últimas versões dos arquivos: eles são cópias completas do repositório. Assim, se um servidor falha, qualquer um dos repositórios dos clientes pode ser copiado de volta para o servidor para restaurá-lo. Cada checkout (resgate) é na prática um backup completo de todos os dados (veja Figura 3.3).

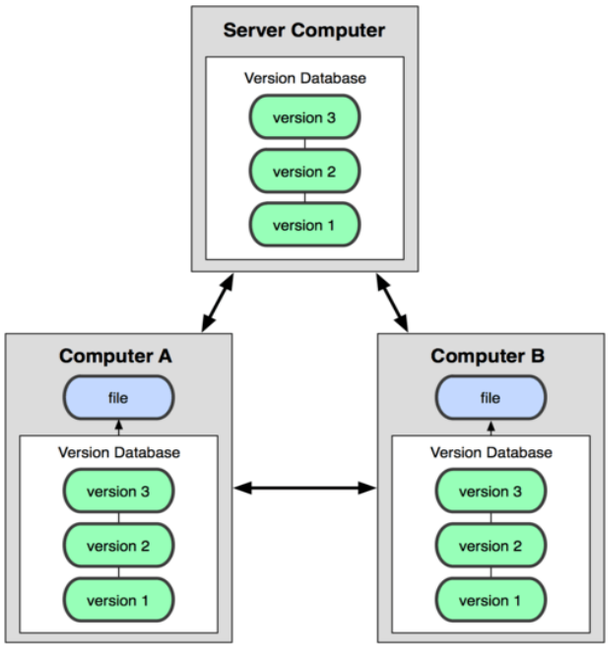


Figura 1-3. Diagrama de Controle de Versão Distribuído.

Além disso, muitos desses sistemas lidam muito bem com o aspecto de ter vários repositórios remotos com os quais eles podem colaborar, permitindo que você trabalhe em conjunto com diferentes grupos de pessoas, de diversas maneiras, simultaneamente no mesmo projeto. Isso permite que você estabeleça diferentes tipos de workflow (fluxo de trabalho) que não são possíveis em sistemas centralizados, como por exemplo o uso de modelos hierárquicos.

1.4 FUNCIONAMENTO BÁSICO

Cada implementação possui sua particularidade, mas a maioria deles compartilham alguns conceitos básicos.

A maior parte das informações - com todo o histórico - ficam guardadas num repositório, num servidor qualquer. Geralmente o acesso é feito por um cliente pela rede e pode ser feito localmente quando o cliente está na mesma máquina do servidor.

O repositório armazena a informação - um conjunto de documentos - de modo persistente num sistema de arquivos ou banco de dados qualquer. É possível que o armazenamento seja feito em outros dispositivos capazes de "eternizar" e resgatar facilmente a informação.

Cada servidor pode ter vários sistemas de controle de versão e cada sistema pode ter diversos repositórios, limitando-se na capacidade de gerenciamento do software e também no limite físico do hardware. Geralmente um repositório possui um endereço lógico que permite a conexão do cliente. Esse endereço pode ser um conjunto IP/porta, uma URL, um caminho do sistema de arquivos etc.

Cada desenvolvedor possui em sua máquina uma cópia local (também chamada de **working copy** em inglês) somente da última versão de cada documento. Essa cópia local geralmente é feita num sistema de arquivos simples (FAT, NTFS, ext3 etc.). A cada alteração relevante do desenvolvedor é necessário "atualizar" as informações do servidor submetendo (**commit** em inglês) as alterações. O servidor então guarda a nova alteração junto de todo o histórico mais antigo. Se o desenvolvedor quer atualizar sua cópia local é necessário atualizar as informações locais, e para isso é necessário baixar novidades do servidor (ou fazer **update** em inglês).

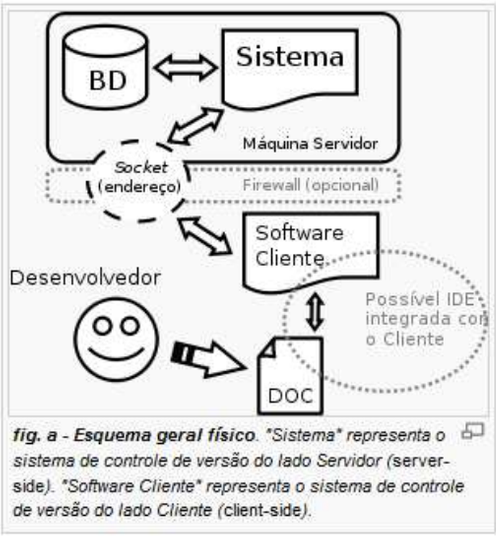


***fig. a - Esquema geral físico**. "Sistema" representa o sistema de controle de versão do lado Servidor (server-side). "Software Cliente" representa o sistema de controle de versão do lado Cliente (client-side).*

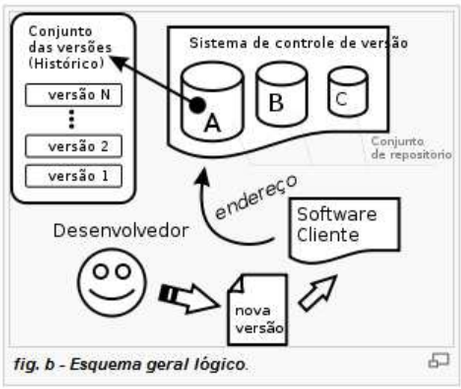


***fig. b - Esquema geral lógico**.*

1.5 ENVIO E RESGATE DE VERSÕES

A principal função do sistema de controle de versão é armazenar todo o histórico de desenvolvimento do documento, desde o primeiro envio até sua última versão. Isso permite que seja possível resgatar uma determinada versão de qualquer data mais antiga, evitando desperdício de tempo no desenvolvimento para desfazer alterações quando se toma algum rumo equivocado.

O envio das alterações é feito a gosto do desenvolvedor (do lado do cliente), quando ele desejar; mas, para minimizar conflitos de versões, facilitar no desfazer de alterações e também no controle do histórico, recomenda-se que uma alteração seja enviada cada vez que o software estiver minimamente estável, a cada nova parte ou a cada alteração relevante que esteja funcionando corretamente. Não é recomendável o envio quando o documento como um todo possa causar alguma dificuldade no desenvolvimento de outro colaborador, como por exemplo, um código não compilado ou com algum defeito que comprometa a execução geral.

Cada "envio" é na maioria dos sistemas chamado de "**commit**" (às vezes "**submit**"), ou seja, efetivar as alterações no (ou "submeter" ao) repositório. Cada envio produz uma nova versão no repositório e é armazenado como "uma fotografia" do momento.

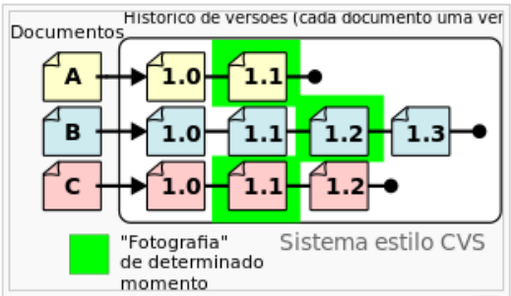


fig. c - Em sistemas no estilo do CVS, cada documento tem sua versão controlada individualmente. Assim, é necessário adicionar marcas ('tags' - em inglês) para que se tenha uma "fotografia" de determinado momento. Ou seja, o versionamento do conjunto fica a cargo do usuário pois a numeração gerada pelo sistema de controle de versão não fornece "alinhamento" aos documentos.

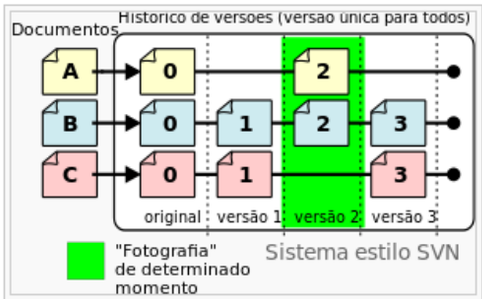


fig. d - Já em sistemas no estilo do SVN, o controle de versão é feito por cada envio ao servidor. Ou seja, há uma versão global para todos os documentos. Se você enviar 4 versões do arquivo B enquanto você enviou apenas 2 versões do A, ambos estão na versão final que é a 4. Perceba que toda "fotografia" de qualquer momento será sempre uma coluna alinhada como mostra a figura.

Se o arquivo fosse armazenado diretamente num sistema de arquivos simples, não haveria histórico, a menos que isso fosse feito manualmente através de cópias completas dos documentos numa organização (que pode ou não envolver pastas) que dependerá do desenvolvedor.

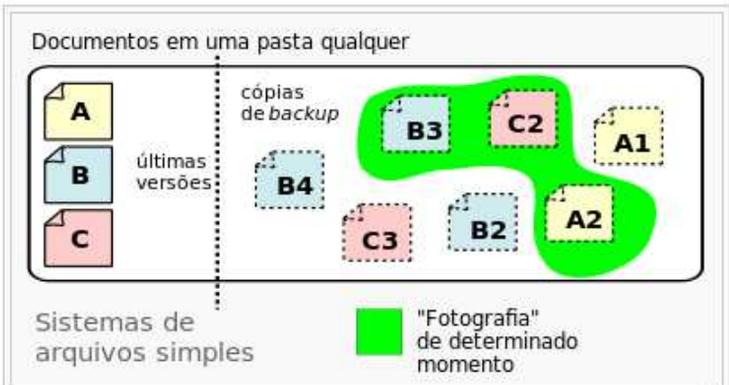


fig e - Num sistema de arquivos simples, sem controle de versões, seria necessário fazer cópias manualmente para manter as versões anteriores, o que poderia se tornar muito confuso com o tempo conforme o número de versões aumentam.

Muitas vezes, é possível acrescentar comentários no envio das alterações, o que facilita também uma possível análise do histórico. Geralmente o relatório com as versões e os comentários de cada envio é chamado de "**histórico**" ou "**log**", e uma análise deste relatório pode facilitar em muitos aspectos no desenvolvimento do produto.

A documentação do CVS, SVN e outros, recomendam que o comentário do histórico seja amplo, geral e abstrato, ou seja, que não se limite à explicação da mudança do código em si, mas sim sobre o que foi mudado ou acrescentado no conceito ou no funcionamento como um todo. A mudança do código pode ser analisada através de uma

diferença entre duas versões, portanto o comentário seria útil apenas para explicar a mudança de forma lógica.

Sistemas de controle de versão também são convenientes quando diversos desenvolvedores trabalham sobre o mesmo projeto simultaneamente, resolvendo eventuais conflitos entre as alterações. A maioria dos sistemas possui diversos recursos como ramificação e mesclagem de histórico para auxiliar nessas tarefas.

Para que seja possível o trabalho em equipe, o sistema de controle de versão pode possuir um mini sistema de controle de usuários embutido ou pode utilizar algum outro sistema de autenticação separado. Assim, é possível identificar cada usuário, que geralmente fica protegido por uma senha pessoal, ou alguma senha criada pelo administrador de sistemas.

No CVS, por exemplo, é possível escolher o método de autenticação a ser usado, dentre várias opções. No caso do SVN, por exemplo, se ele estiver rodando via Apache, o controle de usuários poderá ser feito pela autenticação padrão do Apache. Embora menos comum, é possível também configurar o SVN para utilizar o usuário do sistema, como os usuários do Linux ou do Windows.

Referências do Capítulo:

[Internet] http://pt.wikipedia.org/wiki/Sistema_de_controle_de_vers%C3%A3o

[Internet] http://git-scm.com/book/pt-br/Primeiros-passos-Sobre-Controle-de-Vers%C3%A3o

2 TESTE DE SOFTWARE

Não se pode garantir que todo software funcione corretamente, sem a presença de erros, visto que os mesmos muitas vezes possuem um grande número de estados com fórmulas, atividades e algoritmos complexos. O tamanho do projeto a ser desenvolvido e a quantidade de pessoas envolvidas no processo aumentam ainda mais a complexidade. Idealmente, toda permutação possível do software deveria ser testada. Entretanto, isso se torna impossível para a ampla maioria dos casos devido à quantidade impraticável de possibilidades. A qualidade do teste acaba se relacionando à qualidade dos profissionais envolvidos em filtrar as permutações relevantes.

Falhas podem ser originadas por diversos motivos. Por exemplo, a especificação pode estar errada ou incompleta, ou pode conter requisitos impossíveis de serem implementados, devido a limitações de hardware ou software. A implementação também pode estar errada ou incompleta, como um erro de um algoritmo. Portanto, uma falha é o resultado de um ou mais defeitos em algum aspecto do sistema.

O teste de software pode ser visto como uma parcela do processo de qualidade de software. A qualidade da aplicação pode e, normalmente, varia significativamente de sistema para sistema.

A qualidade de um software pode estar fortemente relacionada à existência de defeitos inseridos durante o desenvolvimento ou manutenção de um produto. Uma das maneiras de identificar os defeitos de uma aplicação de forma que eles possam ser corrigidos é através das atividades de teste de software. Devido à sua importância e complexidade, os testes de software pode ser responsável por uma parcela considerável dos custos de um projeto, por isso, esse assunto merece atenção por parte das organizações que desejam sucesso com a condução das atividades do processo de testes.

Teste de software é o processo de execução de um produto para determinar se ele atingiu suas especificações e funcionou corretamente no ambiente para o qual foi projetado. O seu objetivo é revelar falhas em um produto, para que as causas dessas falhas sejam identificadas e possam ser corrigidas pela equipe de desenvolvimento antes da entrega final. Por conta dessa característica das atividades de teste, dizemos que sua natureza é “destrutiva”, e não “construtiva”, pois visa ao aumento da confiança de um produto através da exposição de seus problemas, porém antes de sua entrega ao usuário final. O teste é um processo realizado pelo testador de software, que permeia outros

processos da engenharia de software, e que envolve ações que vão do levantamento de requisitos até a execução do teste propriamente dito.

O conceito de teste de software pode ser compreendido através de uma visão intuitiva ou mesmo de uma maneira formal. Existem atualmente várias definições para esse conceito. De uma forma simples, testar um software significa verificar através de uma execução controlada se o seu comportamento corre de acordo com o especificado. O objetivo principal desta tarefa é revelar o número máximo de falhas dispondo do mínimo de esforço, ou seja, mostrar aos que desenvolvem se os resultados estão ou não de acordo com os padrões estabelecidos.

2.1 Conceitos básicos associados a Teste de Software

Antes de iniciarmos uma discussão sobre teste de software precisamos esclarecer alguns conceitos relacionados a essa atividade. Inicialmente, precisamos conhecer a diferença entre **Defeitos**, **Erros** e **Falhas**. As definições que iremos usar aqui seguem a terminologia padrão para Engenharia de Software do IEEE – Institute of Electrical and Electronics Engineers – (IEEE 610, 1990).

- **Defeito** é um ato inconsistente cometido por um indivíduo ao tentar entender uma determinada informação, resolver um problema ou utilizar um método ou uma ferramenta. Por exemplo, uma instrução ou comando incorreto.

- **Erro** é uma manifestação concreta de um defeito num artefato de software. Diferença entre o valor obtido e o valor esperado, ou seja, qualquer estado intermediário incorreto ou resultado inesperado na execução de um programa constitui um erro.

- **Falha** é o comportamento operacional do software diferente do esperado pelo usuário. Uma falha pode ter sido causada por diversos erros e alguns erros podem nunca causar uma falha.

A Figura 4.1 expressa a diferença entre esses conceitos. Defeitos fazem parte do universo físico (a aplicação propriamente dita) e são causados por pessoas, por exemplo, através do mal uso de uma tecnologia. Defeitos podem ocasionar a manifestação de erros em um produto, ou seja, a construção de um software de forma diferente ao que foi

especificado (universo de informação). Por fim, os erros geram falhas, que são comportamentos inesperados em um software que afetam diretamente o usuário final da aplicação (universo do usuário) e pode inviabilizar a utilização de um software.



Figura 4.1 – Defeito x erro x falha

Dessa forma, ressaltamos que teste de software revela simplesmente falhas em um produto. Após a execução dos testes é necessária a execução de um processo de **depuração** para a identificação e correção dos defeitos que originaram essa falha, ou seja, **Depurar** não é **Testar**!

2.2 O QUE É TESTE DE SOFTWARE

Falar de testes de software é muito mais do que falar de execução de testes. Testar um software e relatar impressões e não conformidades é fornecer um diagnóstico do estado da aplicação, e é muito importante que esse diagnóstico seja o mais completo e preciso possível, porque ele provavelmente vai servir de base para tomadas de decisões em relação ao projeto que está sendo analisado. Testar uma aplicação é questioná-la, através de casos de teste e principalmente de observações, para analisar as respostas obtidas, pois estas podem revelar defeitos.

Podemos entender por defeito tudo o que ameaça a qualidade do produto, levando em consideração que qualidade é o que o cliente quer. E o que o cliente quer é relativo, porque vai depender da finalidade da aplicação a ser desenvolvida. Por exemplo, uma aplicação pode estar em conformidade com a documentação de requisitos e ainda assim o cliente achar que ela não tem qualidade, pois apresenta demora em realizar determinada operação. Dependendo de onde a aplicação será utilizada, restrições de tempo podem ser cruciais.

A atividade de teste é composta por alguns elementos essenciais que auxiliam na formalização desta atividade. Esses elementos são os seguintes:

- **Caso de Teste:** descreve uma condição particular a ser testada e é composto por valores de entrada, restrições para a sua execução e um resultado ou comportamento esperado (CRAIG e JASKIEL, 2002).

- **Procedimento de Teste:** é uma descrição dos passos necessários para executar um caso (ou um grupo de casos) de teste (CRAIG e JASKIEL, 2002).

- **Critério de Teste:** serve para selecionar e avaliar casos de teste de forma a aumentar as possibilidades de provocar falhas ou, quando isso não ocorre, estabelecer um nível elevado de confiança na correção do produto (ROCHA et al., 2001). Os critérios de teste podem ser utilizados como:
  - **Critério de Cobertura dos Testes:** permitem a identificação de partes do programa que devem ser executadas para garantir a qualidade do software e indicar quando o mesmo foi suficientemente testado (RAPPS e WEYUKER, 1982). Ou seja, determinar o percentual de elementos necessários por um critério de teste que foram executados pelo conjunto de casos de teste.
  - **Critério de Adequação de Casos de Teste:** Quando, a partir de um conjunto de casos de teste T qualquer, ele é utilizado para verificar se T satisfaz os requisitos de teste estabelecidos pelo critério. Ou seja, este critério avalia se os casos de teste definidos são suficientes ou não para avaliação de um produto ou uma função (ROCHA et al., 2001).
  - **Critério de Geração de Casos de Teste:** quando o critério é utilizado para gerar um conjunto de casos de teste T adequado para um produto ou função, ou seja, este critério define as regras e diretrizes para geração dos casos de teste de um produto que esteja de acordo com o critério de adequação definido anteriormente (ROCHA et al., 2001).

2.3 O QUE NÃO É TESTE DE SOFTWARE

Para um melhor entendimento do que é teste, podemos analisar o que não é, ou pelo menos não deveria ser, teste de software. Teste de software **não é**:

- **Um processo burocrático:** Apesar de existir um processo bem definido para a realização das atividades de testes de software, com atividades e papéis bem explicados, esse processo não deve ser burocrático, nem inibir a criatividade do testador, que deve se sentir a vontade para explorar a aplicação em busca de defeitos, mesmo que exista um roteiro de testes a seguir. Isso quer dizer que o testador não precisa deter-se a documentações, pois nem sempre estas estão atualizadas e livres de erros. O testador pode seguir sua intuição, desde que utilizando experiência e bom senso. Nesse caso, a documentação do projeto pode servir como guia, mas não deve ser considerada uma verdade absoluta e inquestionável.

- **Atividade limitada a seguir um roteiro:** Se os testadores se limitam a fazer somente o que pede o roteiro com os casos de teste, provavelmente deixarão de perceber situações estranhas merecedoras de investigação na aplicação. A preocupação em ler o passo a passo do roteiro que é designado ao testador pode tirar seu poder de pensamento crítico em relação ao que está sendo avaliado. O testador, no exercício da execução de testes, deve ter plena liberdade de experimentar situações novas, mesmo que estas lhe venham à cabeça naquele momento de sua atividade. Ele deve também ter liberdade de discutir situações duvidosas que possa encontrar, mesmo que estas venham de uma observação feita por ele, que não estava previamente planejada pela pessoa que escreveu os casos de teste.

- **Uma atividade que pode ser bem desempenhada por qualquer pessoa:** Atividades de testes não serão bem desempenhadas se quem as tiver realizando não for um profissional preparado para tal. Para se testar eficientemente uma aplicação, é preciso possuir algumas habilidades, um conjunto de conhecimentos específicos de técnicas e alguma experiência no assunto.

- **Uma atividade de garantia de qualidade:** Não devemos confundir informar sobre a qualidade do produto com garantir a qualidade do produto gerado. As atividades de teste podem contribuir para a qualidade do produto final, mas a garantia dessa qualidade é de responsabilidade de toda a equipe do projeto. Dizer que testes são atividades de garantia de qualidade pode dar a impressão de que um software que não tem qualidade é um produto mal testado.

- **Uma atividade que pode ser deixada para o final do projeto:** Quando se trata testes de software como apenas uma fase do ciclo de desenvolvimento de um projeto, corre-se o risco de ter o tempo dedicado a essas atividades reduzido, seja por atrasos no cronograma de desenvolvimento ou alocação tardia de testadores no projeto.

- **Uma atividade destrutiva:** O livro Lessons Learned in Software Testing traz uma visão interessante sobre testes como sendo uma atividade tão construtiva quanto o desenvolvimento do software que será testado. No livro, os autores afirmam que o teste não quebra um software, mas nos tira a ilusão de que ele funciona corretamente.

2.4 DEFEITOS NO DESENVOLVIMENTO DE SOFTWARE

No processo de desenvolvimento de software, todos os defeitos são humanos e, apesar do uso dos melhores métodos de desenvolvimento, ferramentas ou profissionais, permanece presentes nos produtos, o que torna a atividade de teste fundamental durante o desenvolvimento de um software. Já vimos que esta atividade corresponde ao último recurso para avaliação do produto antes da sua entrega ao usuário final.

O tamanho do projeto a ser desenvolvido e a quantidade de pessoas envolvidas no processo são dois possíveis fatores que aumentam a complexidade dessa tarefa, e consequentemente aumentam a probabilidade de defeitos. Assim, a ocorrência de falhas é inevitável. Mas o que significa dizer que um programa falhou? Basicamente significa que o funcionamento do programa não está de acordo com o esperado pelo usuário. Por exemplo, quando um usuário da linha de produção efetua consultas no sistema das quais só a gerência deveria ter acesso. Esse tipo de falha pode ser originado por diversos motivos:

- A especificação pode estar errada ou incompleta;
- A especificação pode conter requisitos impossíveis de serem implementados devido a limitações de hardware ou software;
- A base de dados pode estar organizada de forma que não seja permitido distinguir os tipos de usuário;
- Pode ser que haja um defeito no algoritmo de controle dos usuários.

Os defeitos normalmente são introduzidos na transformação de informações entre as diferentes fases do ciclo de desenvolvimento de um software. Vamos seguir um exemplo simples de ciclo de vida de desenvolvimento de software: os requisitos expressos pelo cliente são relatados textualmente em um documento de especificação de requisitos. Esse documento é então transformado em casos de uso, que por sua vez foi o artefato de entrada para o projeto do software e definição de sua arquitetura utilizando diagramas de classes da UML. Em seguida, esses modelos de projetos foram usados para a construção do software em uma linguagem que não segue o paradigma orientado a objetos. Observe que durante esse período uma série de transformações foi realizada até chegarmos ao produto final. Nesse meio tempo, defeitos podem ter sido inseridos. A Figura 4.2 expressa exatamente à metáfora discutida nesse parágrafo.



Figura 4.2 – Diferentes Interpretações ao longo do ciclo de desenvolvimento de um software

Essa série de transformações resultou na necessidade de realizar testes em diferentes níveis, visando avaliar o software em diferentes perspectivas de acordo com o produto gerado em cada fase do ciclo de vida de desenvolvimento de um software.

2.5 NÍVEIS DE TESTE DE SOFTWARE

O planejamento dos testes deve ocorrer em diferentes níveis e em paralelo ao desenvolvimento do software (Figura 4.3). Buscando informação no Livro “Qualidade de Software – Teoria e Prática” (ROCHA et al., 2001), definimos que os principais níveis de teste de software são:

- **Teste de Unidade:** também conhecido como testes unitários. Tem por objetivo explorar a menor unidade do projeto, procurando provocar falhas ocasionadas por defeitos de lógica e de implementação em cada módulo, separadamente. O universo alvo desse tipo de teste são os métodos dos objetos ou mesmo pequenos trechos de código.

- **Teste de Integração:** visa provocar falhas associadas às interfaces entre os módulos quando esses são integrados para construir a estrutura do software que foi estabelecida na fase de projeto.

- **Teste de Sistema:** avalia o software em busca de falhas por meio da utilização do mesmo, como se fosse um usuário final. Dessa maneira, os testes são executados nos mesmos ambientes, com as mesmas condições e com os mesmos dados de entrada que um usuário utilizaria no seu dia-a-dia de manipulação do software. Verifica se o produto satisfaz seus requisitos.

- **Teste de Aceitação:** são realizados geralmente por um restrito grupo de usuários finais do sistema. Esses simulam operações de rotina do sistema de modo a verificar se seu comportamento está de acordo com o solicitado.

- **Teste de Regressão:** Teste de regressão não corresponde a um nível de teste, mas é uma estratégia importante para redução de “efeitos colaterais”. Consiste em se aplicar, a cada nova versão do software ou a cada ciclo, todos os testes que já foram aplicados nas versões ou ciclos de teste anteriores do sistema. Pode ser aplicado em qualquer nível de teste.

Dessa forma, seguindo a Figura 4.3, o planejamento e projeto dos testes devem ocorrer de cima para baixo, ou seja:

1. Inicialmente é planejado o teste de aceitação a partir do documento de requisitos;
2. Após isso é planejado o teste de sistema a partir do projeto de alto nível do software;
3. Em seguida ocorre o planejamento dos testes de integração a partir o projeto detalhado;
4. E por fim, o planejamento dos testes a partir da codificação.

Já a execução ocorre no sentido inverso.

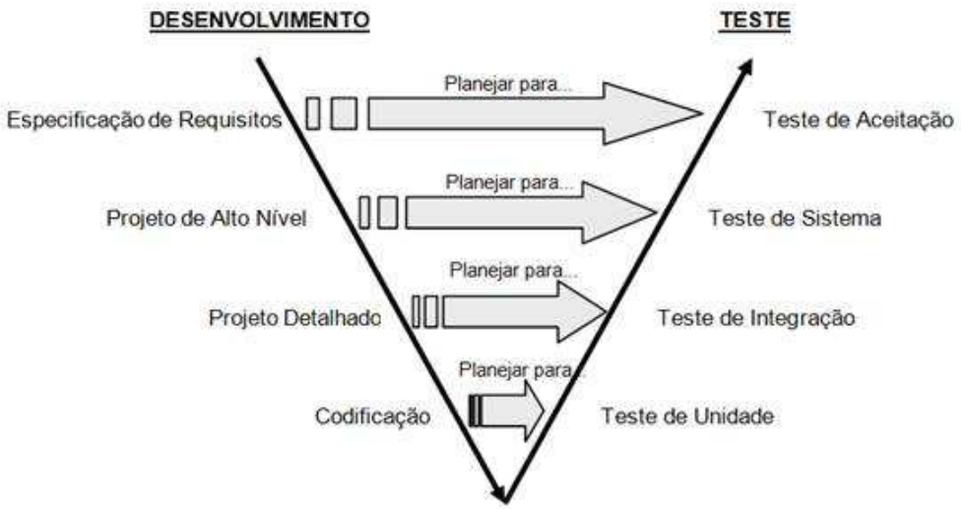


Figura 4.3. Modelo V descrevendo o paralelismo entre as atividades de desenvolvimento e teste de software

2.6 A IMPORTÂNCIA DE TESTES DE SOFTWARE

Quanto mais cedo um defeito for encontrado no ciclo de vida de um software, mais barato será o custo da sua correção. De acordo com Myers, autor do livro The Art of Software Testing, corrigir um defeito que se propagou até o ambiente de produção pode chegar a ser 100 vezes mais caro do que corrigir esse mesmo defeito em fases iniciais do projeto. Muitos dos defeitos que são encontrados no código da aplicação são originados na fase de levantamento de requisitos.

Alguns estudos publicados no Chaos Report, um relatório elaborado pelo grupo de pesquisa Standish Group em 2004, revelam que aproximadamente 45% dos recursos de um projeto de software nos Estados Unidos e Europa chegam a ser dedicados a

atividades de testes e simulações. Assim, testes de software devem ser encarados como investimento em qualidade, e até mesmo economia, já que é uma atividade que identifica defeitos o quanto antes em um projeto.

O teste de software é uma das atividades mais custosas do processo de desenvolvimento de software, pois pode envolver uma quantidade significativa dos recursos de um projeto. O rigor e o custo associado a esta atividade dependem principalmente da criticidade da aplicação a ser desenvolvida. Diferentes categorias de aplicações requerem uma preocupação diferenciada com as atividades de teste.

Um ponto bastante importante para a viabilização da aplicação de teste de software é a utilização de uma infraestrutura adequada. Realizar testes não consiste simplesmente na geração e execução de casos de teste, mas envolvem também questões de planejamento, gerenciamento e análise de resultados. Apoio ferramental para qualquer atividade do processo de teste é importante como mecanismo para redução de esforço associado à tarefa em questão, seja ela planejamento, projeto ou execução dos testes. Após ter sua estratégia de teste definida, tente buscar por ferramentas que se encaixem na sua estratégia. Isso pode reduzir significantemente o esforço de tal tarefa.

Além disso, é importante ressaltar que diferentes tipos de aplicações possuem diferentes técnicas de teste a serem aplicadas, ou seja, testar uma aplicação web envolve passos diferenciados em comparação aos testes de um sistema embarcado. Cada tipo de aplicação possui características especificas que devem ser consideradas no momento da realização dos testes. O conjunto de técnicas apresentado neste artigo cobre diversas características comuns a muitas categorias de software, mas não é completo.

Referências do Capítulo:

[Barbosa00] BARBOSA, E.; MALDONADO, J.C.; VINCENZI, A.M.R.; DELAMARO, M.E; SOUZA, S.R.S. e JINO, M.. “Introdução ao Teste de Software. XIV Simpósio Brasileiro de Engenharia de Software”, 2000.

[Craig02] CRAIG, R.D., JASKIEL, S. P., “Systematic Software Testing”, Artech House Publishers, Boston, 2002.

[IEEE] IEEE Standard 610-1990: IEEE Standard Glossary of Software Engineering Terminology, IEEE Press.

[Pressman05] PRESSMAN, R. S., “Software Engineering: A Practitioner’s Approach”, McGraw-Hill, 6th ed, Nova York, NY, 2005.

[Rocha01] ROCHA, A. R. C., MALDONADO, J. C., WEBER, K. C. et al., “Qualidade de software – Teoria e prática”, Prentice Hall, São Paulo, 2001.

3 HISTÓRICO

No final dos anos 70 a IBM sentiu a necessidade de desenvolver uma abordagem independente da linguagem para estimar o esforço de desenvolvimento de software. Um empregado chamado Allan Albrecht foi designado a desenvolver esta abordagem, o que resultou na técnica de pontos de função.

No início dos anos 80, a técnica de pontos de função e um manual de contagem foram produzidos pela organização IBM's Guide. O Grupo Internacional de Usuários de Pontos de Função (IFPUG) foi fundado no fim dos anos 80. Essa organização produziu seu próprio Manual de Práticas de Contagem. Enquanto a publicação da GUIDE e cada versão nova das publicações da IFPUG continham refinamentos para a técnica originalmente apresentada por Albrecht, elas sempre pretendiam ser consistentes com seu pensamento original.

Nos anos 80 e 90, muitas pessoas que sugeriram técnicas de contagem de pontos de função pretendiam substancialmente estender ou completamente repor o trabalho feito por Albrecht.

3.1 ANÁLISE DE PONTOS DE FUNÇÃO (APF)

É uma técnica para a medição de projetos de desenvolvimento de software, visando estabelecer uma medida de tamanho, em Pontos de Função (PF), considerando a funcionalidade implementada, sob o ponto de vista do usuário. A medida é independente da linguagem de programação ou da tecnologia que será usada para implementação.

O Ponto de Função é uma medida de tamanho de claro significado ao nível do negócio. A APF foi elaborada por Allan Albrecht da IBM e trazida a público em 1979 através da publicação do trabalho **Measuring Application Development Productivity** (Medindo a Produtividade no Desenvolvimento de Aplicativos). A APF quantifica as funções contidas no software em termos significativos para os respectivos usuários. A medida relaciona-se diretamente aos requisitos do negócio que o software pretende tratar. Dessa forma, a APF é imediatamente aplicável a um amplo espectro de ambientes e ao longo da vida de um projeto de desenvolvimento, desde a definição inicial dos requisitos até a fase de plena utilização operacional. Também podem ser derivadas outras medidas úteis ao negócio, tais como a produtividade do processo de desenvolvimento e o custo unitário de suporte ao software.

Sob esse contexto, os objetivos da APF são:

- Medir a funcionalidade solicitada pelo usuário, antes do projeto de software, de forma a estimar seu tamanho e seu custo;
- Medir projetos de desenvolvimento e manutenção de software, independentemente da tecnologia utilizada na implementação, de forma a acompanhar sua evolução;
- Medir a funcionalidade recebida pelo usuário, após o projeto de software, de forma a verificar seu tamanho e custo, comparando-os com o que foi originalmente estimado;

As organizações podem aplicar a Análise de Pontos por Função como:

- Uma ferramenta para determinar o tamanho de pacotes de software adquiridos, através da contagem de todos os PF incluídos no pacote;
- Uma ferramenta para apoiar a análise da qualidade e da produtividade;
- Um mecanismo para estimar custos e recursos envolvidos em projetos de desenvolvimento e manutenção de software;
- Um fator de normalização para comparação de software.

O procedimento para contagem de PF compreende sete passos, mostrados na figura 2.1.

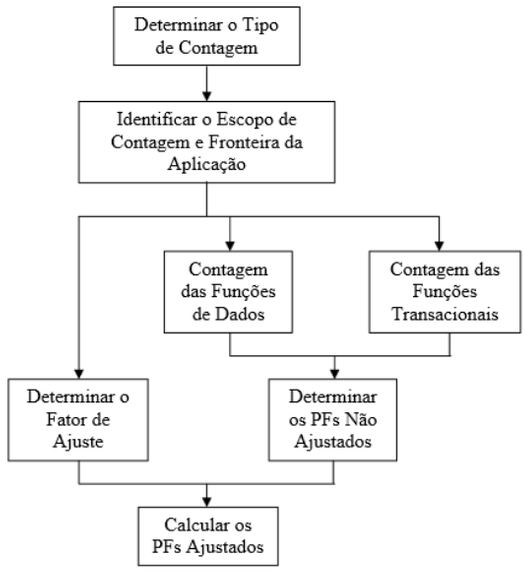


Figura 1 – O Procedimento de Contagem de Pontos de Função.

Determinar o tipo de contagem de pontos de função: este é o primeiro passo no processo de contagem, sendo que existem três tipos de contagem: contagem de PF de projeto de desenvolvimento, de aplicações instaladas e de projetos de manutenção.

- **Identificar o escopo de contagem e a fronteira da aplicação:** neste passo, definem-se as funcionalidades que serão incluídas em uma contagem de PFs específica. A fronteira da aplicação é definida estabelecendo um limite lógico entre a aplicação que está sendo medida, o usuário e outras aplicações. O escopo de contagem define a parte do sistema (funcionalidades) a ser contada.

- **Contagem das funções de dados:** as funções de dados representam as funcionalidades relativas aos requisitos de dados internos e externos à aplicação. São elas os arquivos lógicos internos e os arquivos de interface externa. Ambos são grupos de dados logicamente relacionados ou informações de controle que foram identificados pelo usuário. A diferença está no fato de um **Arquivo Lógico Interno (ALI)** ser mantido dentro da fronteira da aplicação, isto é, armazenar os dados mantidos através de um ou mais processos elementares da aplicação, enquanto que um **Arquivo de Interface Externa (AIE)** é apenas referenciado pela aplicação, ou seja, ele é mantido dentro da fronteira de outra aplicação. Assim, o objetivo de um AIE é armazenar os dados referenciados por um ou mais processos elementares da aplicação sendo contada, mas que são mantidos por outras aplicações.

- **Contagem das funções transacionais:** as funções transacionais representam as funcionalidades de processamento de dados do sistema fornecidas para o usuário. São elas: as entradas externas, as saídas externas e as consultas externas. As **Entradas Externas (EEs)** são processos elementares que processam dados (ou informações de controle) que entram pela fronteira da aplicação. O objetivo principal de uma EE é manter um ou mais ALIs ou alterar o comportamento do sistema. As **Saídas Externas (SEs)** são processos elementares que enviam dados (ou informações de controle) para fora da fronteira da aplicação. Seu objetivo é mostrar informações recuperadas através de um processamento lógico (isto é, que envolva cálculos ou criação de dados derivados) e não apenas uma

simples recuperação de dados. Uma SE pode, também, manter um ALI ou alterar o comportamento do sistema. Por fim, uma **Consulta Externa (CE)** , assim como uma SE, é um processo elementar que envia dados (ou informações de controle) para fora da fronteira da aplicação, mas sem realização de nenhum cálculo nem a criação de dados derivados. Seu objetivo é apresentar informação para o usuário, por meio apenas de uma recuperação das informações. Nenhum ALI é mantido durante sua realização, nem o comportamento do sistema é alterado.

- **Determinar o valor do fator de ajuste:** o fator de ajuste é baseado em 14 características gerais de sistemas, que avaliam a funcionalidade geral da aplicação que está sendo contada, e seus níveis de influência. O nível de influência de uma característica é determinado com base em uma escala de 0 (nenhuma influência) a 5 (forte influência).

- **Determinar a contagem de pontos de função não ajustados:** os pontos de função não ajustados (PFNA) refletem as funcionalidades fornecidas pelo sistema para o usuário. Essa contagem leva em conta dois tipos de função: de dados e transacionais, bem como sua complexidade (simples, média ou complexa).

- **Calcular os pontos de função ajustados:** finalmente, os PFs ajustados são calculados, considerando-se o tipo de contagem definido no primeiro passo.

A figura 2.2 apresenta uma visão geral dos tipos de função que são considerados na contagem da APF.

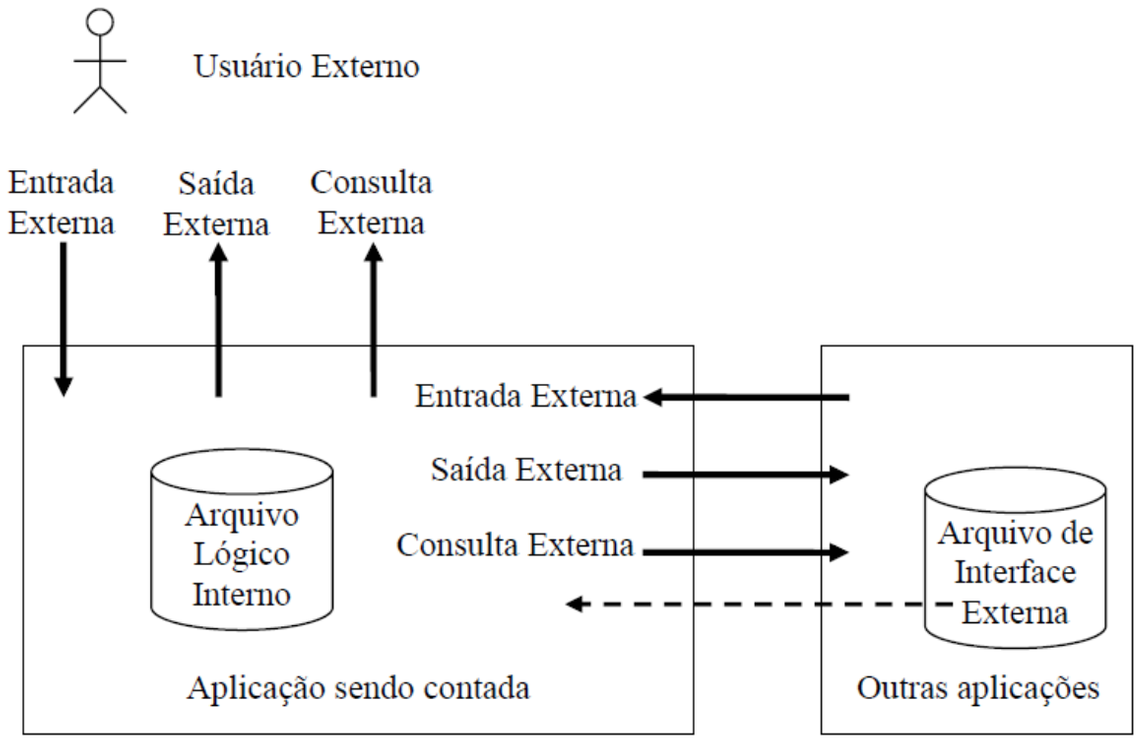


Figura 2.2 – Visão Geral das Funções de uma Aplicação segundo a APF.

Uma aplicação, vista sob a ótica do usuário, é um conjunto de funções ou atividades do negócio que o beneficiam na realização de suas tarefas. Estas funções são divididas nos seguintes grupos ou tipos:

- **Arquivo lógico interno:** Representam os requerimentos de armazenamento de grupos de dados logicamente relacionados, cuja manutenção é efetuada pela própria aplicação;

- **Arquivo de interface externa:** Representam as necessidades de grupos de dados logicamente relacionadas, utilizados pela aplicação, mas que sofrem manutenção a partir de outra aplicação;

- **Entradas externas:** Representam as atividades vindas diretamente do usuário, através de um processo lógico único, com o objetivo de inserir, modificar ou remover dados dos arquivos lógicos internos;

- **Saídas externas:** Representam as atividades da aplicação (processos) que têm como resultado a extração de dados da aplicação;

- **Consultas externas:** Representam as atividades que, através de uma requisição de dados (entrada), gera uma aquisição e exibição imediata da dados (saída).

As funções contribuem para o cálculo de Pontos de Função com base na quantidade (número de funções) e na complexidade funcional relativa atribuída a cada uma delas.

3.2 Contagem das Funções de Dados

O primeiro passo para a contagem das funções de dados consiste em identificar arquivos lógicos internos (ALIs) e arquivos de interface externa (AIEs). Cada uma dessas funções de dados deve ser classificada segundo sua complexidade funcional. Essa complexidade é definida com base em dois conceitos: registros lógicos e itens de dados.

Registros Lógicos são subconjuntos de dados dentro de um ALI/AIE, que foram reconhecidos pelo usuário. Se o usuário não reconhecer subconjuntos de dados em um ALI/AIE, então se deve contar o ALI/AIE como um registro lógico.

Um Item de Dados, por sua vez, é um campo reconhecido pelo usuário como único e não repetido. Vale destacar que só devem ser contados os itens de dados utilizados pela aplicação em contagem.

Contando-se os registros lógicos e os itens de dados de um ALI/AIE, pode-se chegar à sua complexidade, utilizando a tabela 1.

Tabela 1 – Tabela de Identificação da Complexidade das Funções de Dados .

Número de Registros Lógicos	Número de Itens de Dados Referenciados		
	De 1 a 19	De 20 a 50	51 ou mais
Apenas 1	Simples	Simples	Média
De 2 a 5	Simples	Média	Complexa
6 ou mais	Média	Complexa	Complexa

3.3 Contagem das Funções Transacionais

De maneira análoga à contagem das funções de dados, a contagem das funções transacionais envolve a identificação de funções transacionais (entradas externas, saídas externas e consultas externas) e sua classificação de acordo com a complexidade funcional envolvida (simples, média ou complexa). A definição da complexidade funcional é feita com base no número de arquivos referenciados e dos itens de dados manipulados

pela função, utilizando as tabelas 2 para entradas externas e 3 para saídas e consultas externas. Nessas tabelas, um arquivo referenciado pode ser um ALI lido ou mantido pela função transacional, ou um AIE lido pela função transacional. Já o número de itens de dados referenciados é calculado considerando apenas os itens de dados efetivamente referenciados pela função transacional em questão.

Tabela 2 – Tabela de Identificação da Complexidade de Entradas Externas.

Número de Arquivos Referenciados	Número de Itens de Dados Referenciados		
	De 1 a 4	De 5 a 15	16 ou mais
0 ou 1	Simples	Simples	Média
2	Simples	Média	Complexa
3 ou mais	Média	Complexa	Complexa

Tabela 3 – Tabela de Identificação da Complexidade de Saídas e Consultas Externas.

Número de Arquivos Referenciados	Número de Itens de Dados Referenciados		
	De 1 a 5	De 6 a 19	20 ou mais
0 ou 1	Simples	Simples	Média
2 ou 3	Simples	Média	Complexa
4 ou mais	Média	Complexa	Complexa

Referências do Capítulo:

[R. Dias] "Análise por Pontos de Função: Uma Técnica para Dimensionamento de Sistemas de Informação", on-line.

[Internet] http://pt.wikipedia.org/wiki/An%C3%A1lise_de_pontos_de_fun%C3%A7%C3%A3o

[FPUG, 1] IFPUG Manual de Práticas de Contagem de Pontos de Função Versão 4.2.1
Este manual descreve Metodologia Padrão da APF.

CURSO TÉCNICO EM

INFORMÁTICA

ETAPA 3

LINGUAGEM TÉCNICA DE PROGRAMAÇÃO II

Sumário

INTRODUÇÃO 244

INTERFACE GRÁFICA COM SWING 246

COMPONENTES BÁSICOS 247

LOOK AND FEEL 247

COMPONENTES DO SWING 249

COMPONENTES: JFRAME, JPANEL E JBUTTON 251

TRATAMENTO DE EVENTOS COM SWING 252

TIPOS DE EVENTOS DOS COMPONENTES SWING 257

EVENTOS DE MOUSE 257

EVENTOS DO TECLADO 259

EVENTOS DE JABELAS 259

TRABALHANDO COM JTABLE 260

CRIANDO MENU UTILIZANDO JAVA SWING 261

ESTUDANDO UM POUCO MAIS SOBRE INTERFACES GRÁFICAS NO JAVA. 264

JAVA API COLLECTION 265

GENERICS 267

FOR-EACH 268

ITERATOR 268

IMPLEMENTANDO COLLECTIONS BÁSICAS 271

DATA E HORA EM JAVA 272

DATAS E CALENDÁRIOS 273

ENTRADA E SAÍDA DE DATAS 276

JPA COM HIBERNATE 277

UTILIZANDO O HIBERNATE 279

EXEMPLO DE APLICAÇÃO UTILIZANDO HIBERNATE 279

DESENVOLVIMENTO DA INTERFACE GRÁFICA DA NOSSA APLICAÇÃO 280

BANCO DE DADOS 281

MAPEANDO O BANCO DE DADOS 281

CONFIGURANDO O HIBERNATE 282

IMPLEMENTANDO A CLASSE DA SESSÃO 283

IMPLEMENTANDO A CLASSE DE PERSISTÊNCIA 284

IMPLEMENTANDO AS CLASSES 286

API JAVA - STRING TOKENIZER 291

EXERCITANDO COM O STRINGTOKENIZER 292

API JAVA – MATH 292

REFERENCIAS 294

Curso: Informática

Disciplina: Linguagem Técnica de Programação II

Carga horária da etapa: 80 horas

Objetivos gerais: Apresentar os conceitos da linguagem de programação utilizando Java, tornando-os capazes de: Desenvolver sistemas com interfaces gráficas com SWING, com Criação de painéis, janelas e menus utilizando Java.

Avaliações:

	Atividade	**Pontos**
1º Bimestre	Exercícios	10
	Prova Intermediária	10
	Avaliação por área	20
	Total	**40**
	Atividade	**Pontos**
2º Bimestre	Exercícios	10
	Prova Intermediária	10
	Trabalho prático	10
	Avaliação por área	30
	Total	**60**

Bibliografia

- Swing, Second Edition Paperback – Fevereiro 1, 2003
- Java Swing, O'Reilly Media – Setembro 1998.
- http://www.argonavis.com.br/cursos/java/j100/

Objetivo

Esta apostila tem por objetivo, ajudar ao aluno na compreensão:

- Conceitos linguagem de programação java;
- Desenvolvimento de sistemas com interfaces gráficas com SWING;
- Criação de painéis, janelas e menus utilizando java;
- Entender como trabalhar com Datas em java;
- Criar códigos de exemplo com conexão a banco de dados;
- Configurar e trabalhar com Hibernate.

Introdução

Neste curso vamos aperfeiçoar nosso conhecimento em Java e entender melhor esta linguagem de programação orientada a objeto, que foi desenvolvida na década de 90, por uma equipe de programadores chefiada por James Gosling, na empresa Sun Microsystems.

Diferentemente das linguagens convencionais, que são compiladas para código nativo, a linguagem Java é compilada para um bytecode que é executado por uma máquina virtual. A linguagem de programação Java é a linguagem convencional da Plataforma Java, mas não sua única linguagem.

Vamos trabalhar também com o Hibernate, que é um framework para o mapeamento objeto-relacional escrito na linguagem Java, mas também é disponível em Net como o nome NHibernate. Este framework facilita o mapeamento dos atributos entre uma base tradicional de dados relacionais e o modelo objeto de uma aplicação, mediante o uso de arquivos (XML) ou anotações Java (veja Annotation (java)).

Hibernate é um software livre de código aberto distribuído com a licença LGPL.

O objetivo do Hibernate é diminuir a complexidade entre os programas Java, baseado no modelo orientado a objeto, que precisam trabalhar com um banco de dados do modelo relacional (presente na maioria dos SGBDs). Em especial, no desenvolvimento de consultas e atualizações dos dados.

Sua principal característica é a transformação das classes em Java para tabelas de dados (e dos tipos de dados Java para os da SQL). O Hibernate gera as chamadas SQL e libera o desenvolvedor do trabalho manual da conversão dos dados resultante, mantendo o programa portável para quaisquer bancos de dados SQL, porém causando um pequeno aumento no tempo de execução.

Nas questões relacionadas para o gerenciamento de transações e na tecnologia de acesso à base de dados são de responsabilidade de outros elementos na infraestrutura do programa. Apesar de existirem API no Hibernate

para possuir operações de controle transacional, ele simplesmente delegará estas funções para a infraestrutura na qual foi instalada.

No caso de aplicações construídas para serem executadas em servidores de aplicação, o gerenciamento das transações é realizado segundo o padrão JTA. Já nas aplicações standalone, o programa delega o tratamento transacional ao driver JDBC.

Hibernate pode ser utilizado em aplicações Java standalone ou em aplicações Java EE, utilizando servlet ou sessões Enterprise Java Beans.

Esta apostila também detalha a API SWING. Swing é uma API Java para interfaces gráficas. Ela é compatível com a API AWT, mas trabalha de uma maneira totalmente diferente. A API Swing procura renderizar\desenhar por contra própria todos os componentes, ao invés de delegar essa tarefa ao sistema operacional, como a maioria das outras APIs de interface gráfica trabalham.

Por ser uma API de mais alto nível, ou seja, mais abstração, menor aproximação das APIs do sistema operacional, ela tem bem menos performance que outras APIs gráficas e consome mais memória RAM em geral. Porém, ela é bem mais completa, e os programas que usam Swing têm uma aparência muito parecida, independente do Sistema Operacional utilizado.

Interface Gráfica com SWING

Este material apresenta os conceitos iniciais da programação utilizando a API do Swing, hoje uma das mais utilizadas para o desenvolvimento de interfaces visuais para aplicações desktop em Java. Atualmente, o Java suporta, oficialmente, dois tipos de bibliotecas gráficas: AWT e Swing. A AWT foi a primeira API para interfaces gráficas a surgir no Java e foi, mais tarde, superada pelo Swing (a partir do Java 1.2), que possui diversos benefícios em relação a seu antecessor.

As bibliotecas gráficas são bastante simples no que diz respeito a conceitos necessários para usá-las. A complexidade no aprendizado de interfaces gráficas em Java reside no tamanho das bibliotecas e no enorme mundo de possibilidades; isso pode assustar, em um primeiro momento.

AWT e Swing são bibliotecas gráficas oficiais inclusas em qualquer JRE ou JDK. Além destas, existem algumas outras bibliotecas de terceiros, sendo a mais famosa, o SWT - desenvolvida pela IBM e utilizada no Eclipse e em vários outros produtos.

As APIs de interface gráfica do Java favorecem, ao máximo, o lema de portabilidade da plataforma Java. O look-and.-feel do Swing é único em todas as plataformas onde roda, seja ela Windows, Linux, ou qualquer outra. Isso implica que a aplicação terá exatamente a mesma interface (cores, tamanhos etc.) em qualquer sistema operacional.

Grande parte da complexidade das classes e métodos do Swing está no fato da API ter sido desenvolvida tendo em mente o máximo de portabilidade possível. Favorece-se, por exemplo, o posicionamento relativo de componentes, em detrimento do uso de posicionamento fixo, que poderia prejudicar usuários com resoluções de tela diferentes da prevista.

Com Swing, não importa qual sistema operacional, qual resolução de tela, ou qual profundidade de cores: sua aplicação se comportará da mesma forma em todos os ambientes.

Componentes Básicos

O seguinte esquema mostra a maioria das classes que compõem o Java Swing e mostra também a relação entre as classes AWT (a amarelo) e as classes Swing (a azul):

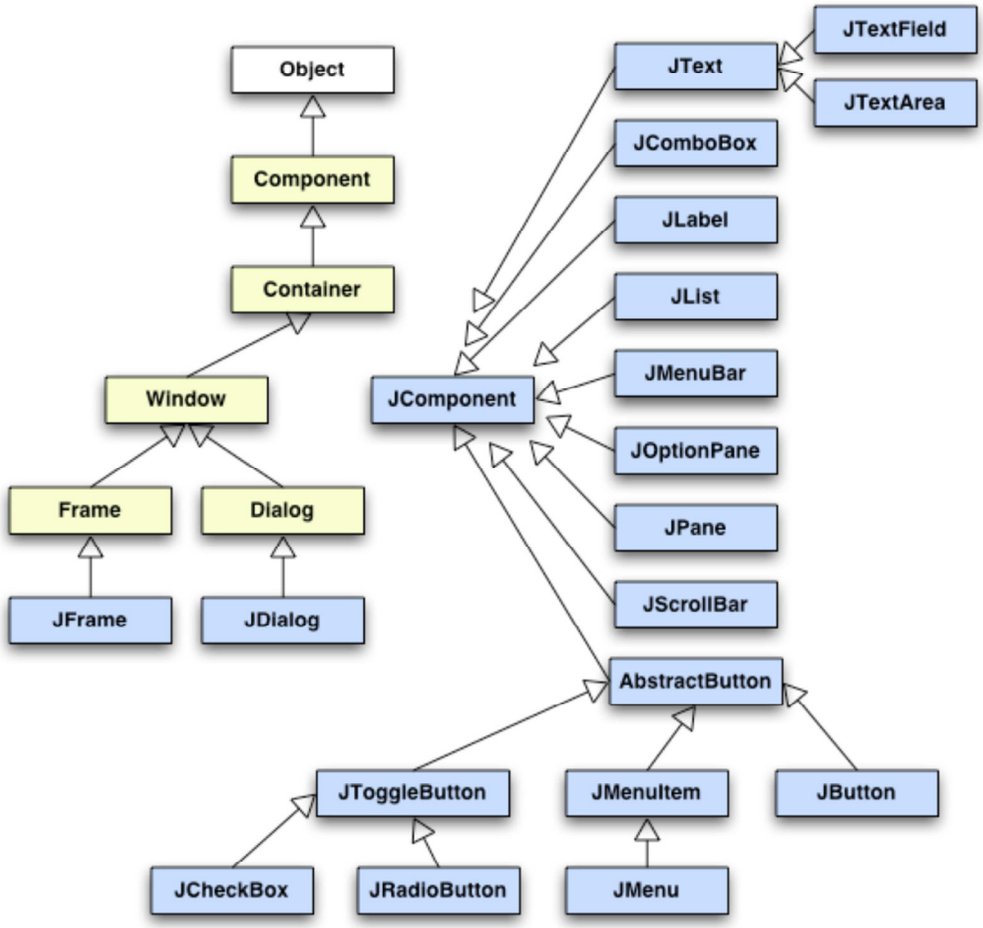


Figura 1: Diagrama resumido das classes AWT (amarelo) e Swing (azul).

Look And Feel

Look-and.-Feel (ou LaF) é o nome que se dá à "cara" da aplicação (suas cores, formatos e etc.). Por padrão, o Java vem com um look-and.-feel próprio, que se comporta exatamente da mesma forma em todas as plataformas suportadas.

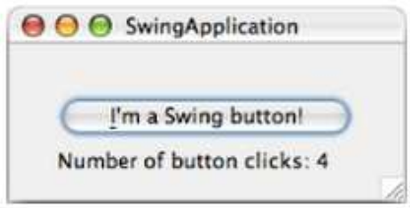
Mas, às vezes, esse não é o resultado desejado. Quando rodamos nossa aplicação no Windows, por exemplo, é bastante gritante a diferença em relação ao visual das aplicações nativas. Por isso é possível alterar qual o look-and-feel a ser usado em nossa aplicação.

Além do padrão do Java, o JRE 5 da Sun ainda traz LaF nativos para Windows e Mac OS, além do Motif e GTK. E, fora esses, você ainda pode baixar diversos LaF na Internet ou até desenvolver o seu próprio.

Veja esses screenshots da documentação do Swing mostrando a mesma aplicação rodando com 4 LaF diferentes:









Componentes do Swing

O Swing traz muitos componentes para usarmos: botões, entradas de texto, tabelas, janelas, abas, scroll, árvores de arquivos e muitos outros. Durante o treinamento, veremos alguns componentes que nos ajudarão a fazer as telas do Argentum. A biblioteca do Swing está no pacote javax.swing (inteira, exceto a parte de acessibilidade, que está em javax.accessibility).

Vamos iniciar apresentando uma mensagem na tela.

A classe mais simples do Swing é a JOptionPane que mostra janelinhas de mensagens, confirmação e erros, entre outras.

Podemos mostrar uma mensagem para o usuário com a seguinte linha:

```
JOptionPane.showMessageDialog(null, "Minha mensagem!");
```

Escolhendo arquivos no sistema operacional.

A classe JFileChooser é a responsável por mostrar uma janela de escolha de arquivos. É possível indicar o diretório inicial, os tipos de arquivos a serem mostrados, selecionar um ou vários e muitas outras opções.

```
JFileChooser fileChooser = new JFileChooser();
fileChooser.showOpenDialog(null);
```

O argumento do showOpenDialog indica qual o componente pai da janela de mensagem (pensando em algum frame aberto, por exemplo, que não é nosso caso). Esse método retorna um int indicando se o usuário escolheu um arquivo ou cancelou. Se ele tiver escolhido um, podemos obter o File com getSelectedFile.

Veja exemplo do código abaixo:

```
JFileChooser fileChooser = new JFileChooser();
int retorno = fileChooser.showOpenDialog(null);

if (retorno == JFileChooser.APPROVE_OPTION) {
   File file = fileChooser.getSelectedFile();
   // faz alguma coisa com arquivo
} else {
   // dialogo cancelado
}
```

Exercitando...

Veja abaixo o programa que calcule a média de salários de uma empresa, pedindo ao usuário a grade de funcionários e os salários em JOptionPane e devolvendo a média.

```
import javax.swing.JOptionPane;
public class Media {
    public static void main(String[] args) {

        int quantidadeFuncionarios = 0;
        float salarioFuncionario = 0;
        float totalSalarios = 0;
        float mediaSalarial = 0;
        //variável contadora para laço
        int contadora = 0;

        quantidadeFuncionarios = Integer.parseInt(JOptionPane.showInputDialog(null,"Digite a quantidade de funcionários","Número de funcionários",JO
ptionPane.QUESTION_MESSAGE));

        while (contadora < quantidadeFuncionarios){
            contadora++;
            salarioFuncionario = Float.parseFloat(JOptionPane.showInputDialog(null,"Digite o salário do funcionário","Salário",JOptionPane.QUESTION_
MESSAGE));
            totalSalarios = totalSalarios + salarioFuncionario;
        }

        mediaSalarial = totalSalarios / quantidadeFuncionarios;
        System.out.println("Média Salarial = " + mediaSalarial);

        System.exit(0);
    }

}
```

Componentes: JFrame, JPanel e JButton

Os componentes mais comuns já estão frequentemente prontos e presentes na API do Swing. Contudo, para montar as telas que são específicas do seu projeto, será necessário compor alguns componentes mais básicos, como JFrames, JPanels, JButtons, etc.

A tela que vamos para o Argentum terá vários componentes. Para começarmos, faremos dois botões importantes: um dispara o carregamento do XML e outro permite ao usuário sair da aplicação. Criar um componente do Swing é bastante simples!

Veja o exemplo abaixo, onde criamos um botão:

```
JButton botao = new JButton("Carregar XML");
```

Contudo, esse botão apenas não faz nossa tela ainda. É preciso adicionar o botão para sair da aplicação. O problema é que, para exibirmos vários componentes organizadamente, é preciso usar um painel para agrupar esses componentes: o JPanel:

```
JButton botaoCarregar = new JButton("Carregar XML");
JButton botaoSair = new JButton("Sair");

JPanel painel = new JPanel();
painel.add(botaoCarregar);
painel.add(botaoSair);
```

Importante: Ordem dos componentes

Repare que a ordem em que os componentes são adicionados importa! Repare também que não definimos posicionamento algum para os nossos componentes: estamos usando o comportamento padrão dos painéis - falaremos mais sobre os layouts managers logo mais.

Por fim, temos apenas objetos na memória. Ainda é preciso mostrar esses objetos na tela do usuário. O componente responsável por isso é o JFrame, a moldura da janela aberta no sistema operacional.

```
JFrame janela = new JFrame("Argentum");
janela.add(panel);
janela.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
janela.pack();
janela.setVisible(true);
```

O método pack() de JFrame, chamado acima, serve para organizar os componentes do frame para que eles ocupem o menor espaço possível. E o setVisible recebe um boolean indicando se queremos que a janela esteja visível ou não.

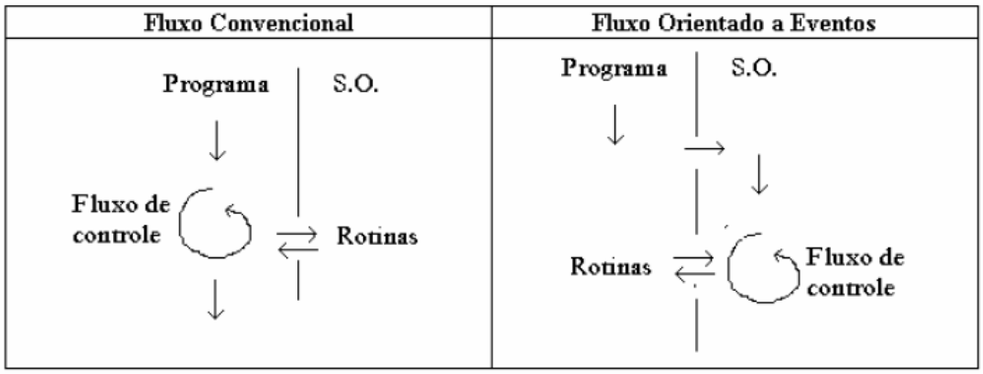
Adicionamos também um comando que indica ao nosso frame que a aplicação deve ser terminada quando o usuário fechar a janela (caso contrário à aplicação continuaria rodando).

Tratamento de eventos com SWING

Eventos é a forma utilizada por Java para adicionar interatividade em interfaces gráficas. Cada vez que o usuário pressionar um botão, digitar um caractere ou movimentar o mouse, por exemplo, ocorre um evento.

Neste modelo de programação, uma vez criados os componentes da interface (ex: janela), o sistema operacional se encarrega de monitorar as ações do usuário. Não existe a necessidade de prever, na aplicação, um laço aguardando uma entrada via mouse ou teclado. Essa responsabilidade passa a ser do sistema operacional.

Veja a representação deste modelo de programação na figura abaixo.



Até agora os botões que fizemos não têm efeito algum, já que não estamos tratando os eventos que são disparados no componente. E quando falamos de botões, em geral, estamos interessados em saber quando ele foi disparado (clicado). No linguajar do Swing, queremos saber quando uma ação (action) aconteceu com o botão.

Mas como chamar um método quando o botão for clicado? Como saber esse momento?

O Swing, como a maioria dos toolkits gráficos, nos traz o conceito de Listeners (ouvintes), que são interfaces que implementamos para sobrescrever métodos que serão disparados pelas ações sobre um botão, por exemplo. Os eventos do usuário são capturados pelo Swing que chama o método que implementamos.

No nosso caso, para fazer um método disparar ao clique do botão, usamos a interface ActionListener. Essa interface nos dá um método actionPerformed, apresentada na figura abaixo.

```
public void actionPerformed(ActionEvent e) {
  // ... tratamento do evento aqui...
}
```

Agora, precisamos indicar que essa ação (esse ActionListener) deve ser disparada quando o botão for clicado. Fazemos isso através do método addActionListener, chamado no botão. Ele recebe como argumento um objeto que implementa ActionListener, conforme no exemplo abaixo.

```
ActionListener nossoListener = ????;
botao.addActionListener(nossoListener);
```

A dúvida que fica é: onde implemento meu ActionListener? A forma mais simples e direta seria criar uma nova classe normal que implemente a interface ActionListener e tenha o método actionPerformed. Depois basta dar new nessa classe e passar para o botão.

Essa forma funciona consideravelmente bem para botões que têm o mesmo comportamento em diversas telas do sistema, mas torna-se um problema para aqueles botões que devem chamar métodos diferentes de acordo com o contexto da página onde estão inseridos.

Por exemplo, em um botão de Ok: os métodos que precisam ser chamados quando um botão com esse texto é clicado podem ser os mais variados. Nesses casos, que são a maioria, o mais comum é encontrar a implementação dos ActionListeners através de classes internas e anônimas.

Utilização de classes anônimas e internas.

Esta forma de desenvolvimento é muito utilizada em java e ajuda na organização do código.

Vamos entender melhor como ela pode nos ajudar no desenvolvimento de aplicações Java Swing. Um bom exemplo é um grande sistema, com várias funcionalidades e em uma tela grande frequentemente temos muitos componentes que disparam eventos e muitos eventos diferentes. Temos que ter um objeto de listener para cada evento e temos que ter classes diferentes para cada tipo de evento disparado.

Só que é muito comum que nossos listeners sejam bem curtos, em geral chamando algum método da lógica de negócios ou atualizando algum componente. E, nesses casos, seria um grande problema de manutenção criar uma classe top level para cada evento.

Voltando ao exemplo do botão Ok, seria necessário criar uma classe para cada um deles e, aí, como diferenciar os nomes desses botões: nomes de

classes diferentes? Pacotes diferentes? Ambas essas soluções causam problemas de manutenibilidade.

O mais comum para tais casos é criarmos classes internas que manipula os componentes que desejamos tratar junto à classe principal. Veja que as classes internas são classes declaradas dentro de outras classes.

```
public class Externa {
  public class Interna {

  }
}
```

Uma classe interna tem nome Externa. Interna pois faz parte do objeto da classe externa. A vantagem é não precisar de um arquivo (*.java) separado e que classes internas podem acessar tudo que a externa possui (métodos, atributos etc.). É possível até encapsular essa classe interna marcando-a como private. Dessa forma, apenas a externa pode enxergar.

Fora isso, são classes normais, que podem implementar interfaces, ter métodos, ser instanciadas etc.

Uma forma mais específica de classe interna é a chamada classe anônima que é muito vista em códigos com Swing. De forma simples, cria-se uma classe sem mesmo declarar seu nome em momento algum, isto é, o código **public class SairListener implements ActionListener** {não existirá em lugar algum}.

Vamos entender melhor como isso funciona na prática, pois em um primeiro momento, a sintaxe das classes anônimas pode assustar - tente não criar preconceitos.

Vamos usar uma classe anônima para tratar o evento de clique no botão que desliga a aplicação.

```
JButton botaoSair = new JButton("Sair");

ActionListener sairListener = new ActionListener() {
  public void actionPerformed(ActionEvent e) {
    System.exit(0);
  }
};

botaoSair.addActionListener(sairListener);
```

Repare que precisamos de um objeto do tipo ActionListener para passar para nosso botão. Normalmente criaríamos uma nova classe para isso e daríamos new nela. Usando classes anônimas damos new e implementamos a classe ao mesmo tempo, usando a sintaxe que vimos acima.

E podemos simplificar mais ainda sem a variável local sairListener.

```
JButton botaoSair = new JButton("Sair");
botaoSair.addActionListener(new ActionListener() {
  public void actionPerformed(ActionEvent e) {
    System.exit(0);
  }
});
```

Cuidado para não entender errado. Embora a sintaxe diga new ActionListener() sabemos que é impossível criar um objeto a partir de uma interface Java. O que essa sintaxe indica (e as chaves logo após o parêntesis indicam isso) é que estamos dando new em uma nova classe que implementa a interface ActionListener e possui o método actionPerformed cuja implementação chama o System.exit(0).

Como criamos uma classe no momento que precisamos, é comum nos perguntarmos qual é o nome dessa classe criada. A resposta para essa pergunta é exatamente a esperada: não sabemos! Ninguém definiu o nome

para essa implementação particular de um ActionListener, então, por não ter nome definido, classes como essas são chamadas de classes anônimas.

Tipos de eventos dos componentes Swing

Os componentes Swing podem gerar vários tipos de eventos diferentes.

Alguns exemplos:

- Usuário clica em um botão - ActionListener;
- Usuário fecha um frame - WindowListener;
- Usuário pressiona um botão do mouse - MouseListener;
- Usuário move o mouse - MouseMotionListener;
- Componentes se tornam visíveis - ComponentListener;

Eventos de Mouse

Capturar os clicks e os movimentos do mouse dentro de um componente de uma GUI. Responder a atividade de mouse com ações apropriadas. Criar programas interativos que são dirigidos pela atividade do mouse.

Vamos trabalhar com um novo exemplo.

```
public class GUIDemo extends JFrame implements
  ActionListener{
  protected int numCl=0;
  protected JButton meuJButton = null;
  protected JLabel meuJLabel = null;

  public GUIDemo(String title) {
         super(title);
         initialize();
  }

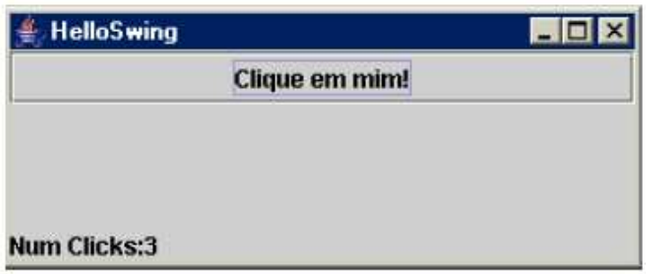
  public GUIDemo() {this("Demonstrando eventos");}
```

```
protected void initialize() {
     this.setSize( new Dimension(200,75) );
     meuJButton = new JButton("Clique em mim!");
     meuJButton.addActionListener(this);
     meuJLabel = new JLabel("Num. clicks="+numCl);
     this.getContentPane().add(meuJButton,
                                BorderLayout.NORTH);
     this.getContentPane().add(meuJLabel,
                                BorderLayout.SOUTH);

     this.setDefaultCloseOperation(JFrame.EXIT_ON_CLO
SE);
     this.pack();
     this.setVisible(true);
}

  public void actionPerformed(ActionEvent e) {
       numCl++;
       meuJLabel.setText("Num Clicks:"+numCl);
  }
}
```

O resultado do código acima é:



Eventos do teclado

São usados para ouvir atividade de teclado dentro de um componente UI (geralmente um panel). Com eles respondemos a atividade de teclado com as ações apropriadas.

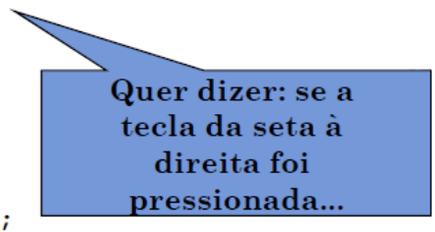
A interface KeyListener deve ser implementada para ouvirmos entradas do teclado. Assim como nos casos anteriores, existe uma classe Adapter que implementa versões vazias de todos os métodos desta interface. Neste caso a classe a ser estendida é KeyAdapter.

Vamos trabalhar com o exemplo abaixo:

```
class PacManKeyListener extends KeyAdapter {
  public void keyPressed(KeyEvent event) {
    char keyChar = event.getKeyChar();
    int keyCode  = event.getKeyCode();

    if (keyCode == KeyEvent.VK_RIGHT) {
      pacman.setX(pacman.getX() + 1);
      pacpanel.repaint();
    } else if (keyChar == 'Q')
      System.exit(0);
  }
}

PacPanel panel = new PacPanel();
panel.addKeyListener(new PacKeyListener());
```



Eventos de jabelas

Os eventos de janela são tratados por classes que implementem a interface WindowListener. Como seria de se esperar, existe uma classe chamada WindowAdapter que tem uma implementação vazia de cada um destes métodos.

Definição da classe WindowListener

```
public interface WindowListener {
   public void windowClosing(WindowEvent e)
   public void windowClosed(WindowEvent e)
   public void windowOpened(WindowEvent e)
   public void windowIconified(Window Event e)
   public void windowDeIconified(Window Event e)
   public void windowActivated(Window Event e)
   public void windowDeactivated(Window Event e)
}
```

Trabalhando com JTable

Para mostrarmos tabelas em Swing, usamos o JTable que é um dos componentes mais complexos de todo o Java. Tanto que tem um pacote javax.swing.table só para ele.

Um JTable é extremamente flexível. Ele serve para exibição e edição de dados, pode ter outros componentes dentro, reorganizar as colunas, fazer drag and drop, ordenação, entre outros.

Basicamente, o que precisamos é o nosso JTable, que representa a tabela, e um TableModel, que representa os dados que queremos exibir na tabela. Podemos ainda definir muitas outras configurações, como o comportamento das colunas com TableColumn.

Começamos criando o JTable. Veja o código abaixo:

```
JTable table = new JTable();

// por padrão, vem sem bordas, então colocamos:
table.setBorder(new LineBorder(Color.black));
table.setGridColor(Color.black);
table.setShowGrid(true);
```

Um JTable não tem comportamento de rolagem para tabelas muito grandes. Mas podemos adicionar esse comportamento compondo com um JScrollPane.

```
JScrollPane scroll = new JScrollPane();
scroll.getViewport().setBorder(null);
scroll.getViewport().add(table);
scroll.setSize(450, 450);
```

Ou seja, adicionamos a tabela ao scrollpane. Depois esse painel com barra de rolagem será adicionado ao painel Principal.

Criando Menu utilizando Java Swing

Para a criação de menus iremos usar objetos das classes JMenuBar, JMenu e JMenuItem.

Os menus podem ser adicionados aos objetos das classes que disponibilizam o método setMenuBar, como é o caso de JFrame.

A classe JMenuBar permite criar uma barra de menus, herda de JComponent e é um contentor para objetos de JMenu. JMenu é um contentor para itens de menu, objetos da classe JMenuItem.

Também podem ser usados como itens de menu botões de opção (JRadioButtonMenuItem) e caixas de seleção (JCheckBoxMenuItem) mas não serão vistos aqui.

Construir uma janela com uma barra de menus, Menu1 e Menu2, definir para o Menu1 itens de menu com um texto, com texto e imagem, só com imagem e ainda um item com submenus, de modo a obter-se o seguinte aspecto, carregando em Verde ou Amarelo assim muda a cor da janela.

Nossa classe vai precisar utilizar os seguintes **imports.**

import java.awt.*;

import java.awt.event.*;

import javax.swing.*;

A classe deve ser chamar **MinhaJanela** e deverá ser implementada conforme abaixo:

```
package br.aula.swing;

import java.awt.Color;
import java.awt.Container;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyEvent;

import javax.swing.ImageIcon;
import javax.swing.JFrame;
import javax.swing.JMenu;
import javax.swing.JMenuBar;
import javax.swing.JMenuItem;
import javax.swing.JOptionPane;
import javax.swing.KeyStroke;

public class MinhaJanela extends JFrame {
        JMenuBar menuBar;
        JMenu menu, submenu;
        JMenuItem menuItem;
        Container c;

        // Construtor da janela que vai conter o menu
        public MinhaJanela(String s) {
                super(s);

                // Criaçao da barra de menu
                menuBar = new JMenuBar();

                // O primeiro menu
                menu = new JMenu("Menu1 ");
                menu.setMnemonic(KeyEvent.VK_1);
                menuBar.add(menu); // junta à barra de menus o menu

                // Um grupo de JMenuItems
                menuItem = new JMenuItem ("Texto", KeyEvent.VK_T);
                menuItem.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_X,
                                ActionEvent.ALT_MASK));
                menu.add (menuItem); // junta o menu item ao menu

                // Criar objecto de escuta
                MenuItemEscuta mesc = new MenuItemEscuta ();
                menuItem.addActionListener (mesc); // junta objecto de escuta ao menu
                                                                                // item
                menuItem = new JMenuItem ("Texto e Imagem",
                                new ImageIcon("SBWeb_04.gif"));
                menuItem.setMnemonic(KeyEvent.VK_I);
                menu.add(menuItem);
                menuItem = new JMenuItem(new ImageIcon("SBWeb_04.gif"));
                menu.add (menuItem);
                menu.addSeparator (); // junta separador

                // Um submenu é do tipo JMenu
                submenu = new JMenu("Cores");
                submenu.setMnemonic (KeyEvent.VK_C);
                menuItem = new JMenuItem ("Verde");
```

```
                menuItem.setAccelerator (KeyStroke).getKeyStroke (KeyEvent).VK_V,
                                ActionEvent.ALT_MASK));
                submenu.add(menuItem);
                menuItem.addActionListener (mesc); // junta objecto de escuta ao menu
                                                                                        // item
                menuItem = new JMenuItem ("Amarelo");
                submenu.add(menuItem);
                menuItem.addActionListener (mesc); // junta objecto de escuta ao menu
                                                                                        // item
                menu.add (submenu); // junta o submenu ao menu

                // Construção do segundo menu na barra de menu.
                menu = new JMenu("Menu_2");
                menu.setMnemonic(KeyEvent.VK_2);
                menuBar.add (menu); // junta o menu à barra de menu

                // Junta à barra de menu à janela
                this.setJMenuBar(menuBar);

                // Obtem o content pane do frame e pinta-o de azul
                c = getContentPane();
                c.setBackground(Color.BLUE);
        } // termina o construtor

        // Classe Interna - MenuItemEscuta
        class MenuItemEscuta implements ActionListener {
                public void actionPerformed(ActionEvent e){
                        if(e.getActionCommand().equals("Texto")) {
                                JOptionPane.showMessageDialog (MinhaJanela.this,"Executou o
item TEXTO");
                        } else {
                                if((e.getActionCommand()).equals("Verde")) {
                                        c.setBackground(Color.GREEN);
                                } else if((e.getActionCommand()).equals("Amarelo"))
                                        c.setBackground(Color.YELLOW);
                        }
                }
        }// fecha a classe interna
} // fecha a classe Minha Janela
```

Agora precisamos implementar a classe principal que deverá utilizar nossa classe com o menu.

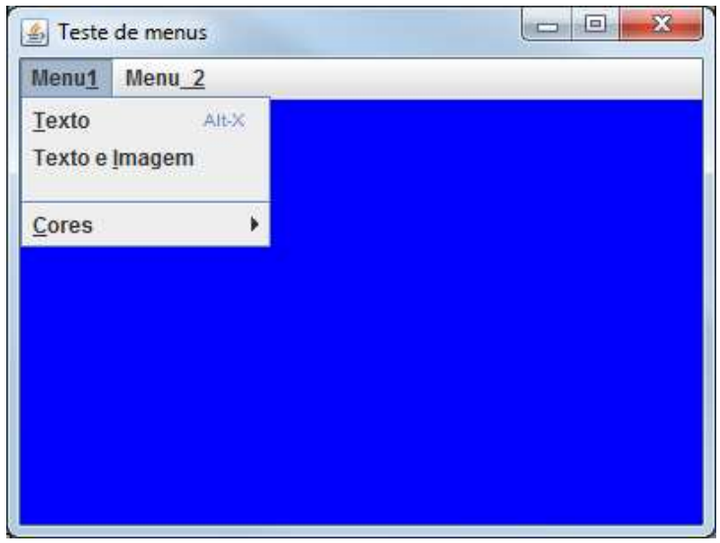
Segue abaixo:

```
package br.aula.swing;

import javax.swing.JFrame;

public class TesteMenu {
      public static void main(String[] args) {
            JFrame janela = new Minha Janela("Teste de menus");
            janela.setSize(400, 300);
            janela.setVisible(true);
            janela.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

O resultado final do nosso programa será o seguinte :



Estudando um pouco mais sobre interfaces gráficas no Java.

Consultar o javadoc do Swing pode não ser muito simples. Por isso, a Sun disponibiliza um ótimo tutorial online sobre Swing, que hoje está hospedado no site da Oracle: **http://docs.oracle.com/javase/tutorial/uiswing/**

Aplicações grandes com Swing podem ganhar uma complexidade enorme e ficar difíceis de manter. Alguns projetos tentam minimizar esses problemas; há, por exemplo, o famoso projeto Thinlet, onde um XML substitui classes como a nossa ArgentumUI.

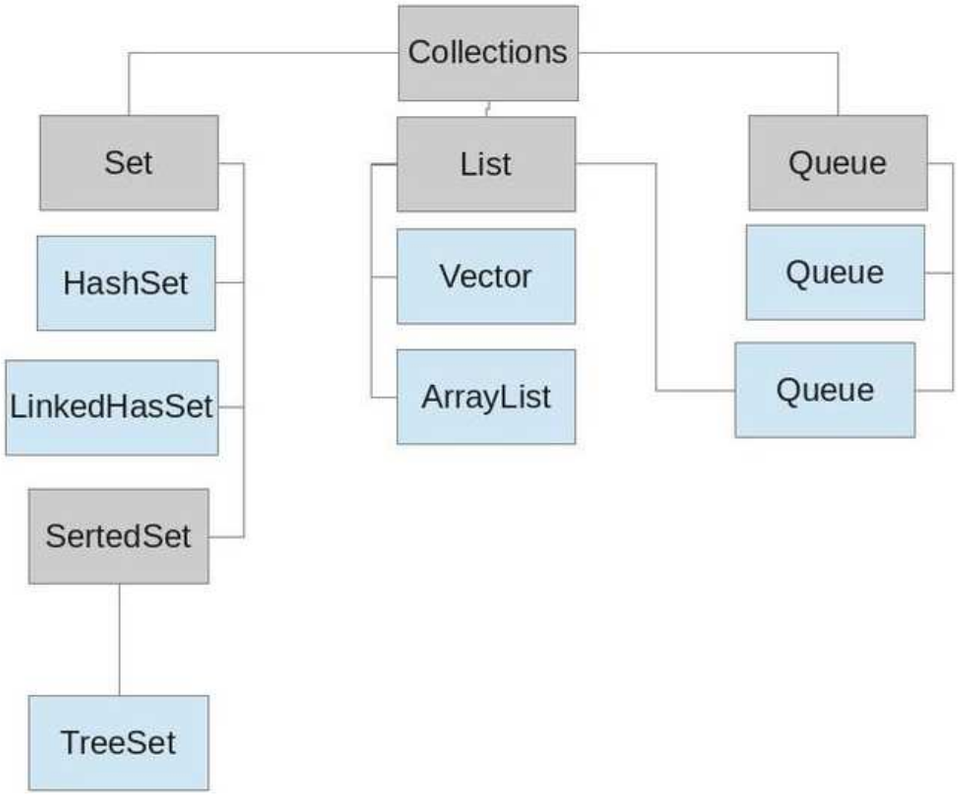
Existem diversos frameworks, como o JGoodies, que auxiliam na criação de telas através de APIs teoricamente mais simples que o Swing. Não são editores, mas sim bibliotecas mais amigáveis. Apesar da existência de tais bibliotecas, é importante conhecer as principais classes e interfaces e seu funcionamento no Swing.

Java API Collection

A infraestrutura Collections ou coleção do Java disponibiliza os recursos de estrutura de dados, ou seja, contem as principais funcionalidades para se trabalhar com conjunto de elementos (coleções).

Segundo Deitel com as Collections "você utiliza estruturas de dados existentes, sem se preocupar com a maneira como são implementadas.

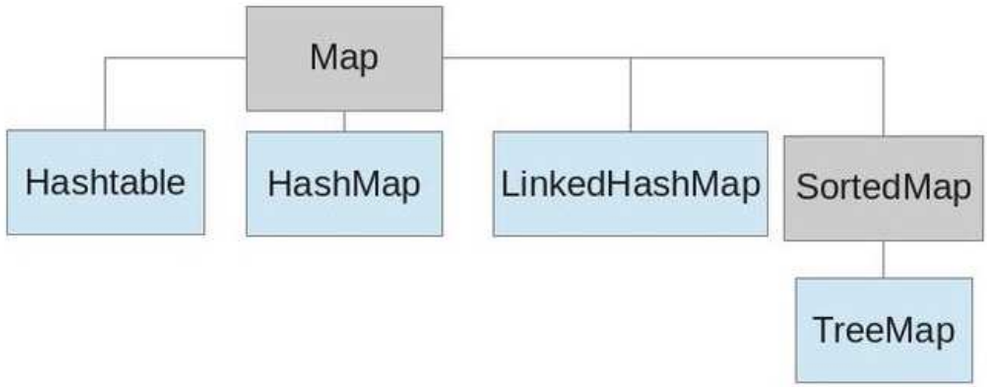
O Java possui uma interface raiz denominada Collections, com disponibilidade de outras estruturas para ser utilizada, em geral as Collections são estruturas dinâmicas, ou seja, podem crescer conforme a necessidade de expansão, diferente do array por exemplo. O conjunto de interfaces e classes concretas que fazem parte da API Collection pode ser Figura 1, os quadros em cinza representam as Interfaces de coleções e os quadros em azul representam classes concretas.



Quanto mais genérica, por exemplo, a Interface Collection, possui menos acoplamento, quanto mais especifica, mais acoplamento. Segundo Silveira et. al. "Trabalhar com coleções, escolher a implementação certa para cada caso é uma tarefa difícil. Cada uma delas, como ArrayList, LinkedList ou HashSet, é melhor para resolver determinadas categorias de problemas"

Outro ponto, é que quanto mais genérica, mais estável, pois possui menos chances de modificação, pois dela, outras classes dependem, quanto mais especifica, menos estável e mais modificações pode existir.

Há ainda classes que implementam a interface Map ao invés da Collection, mesmo assim, faz parte da API Collection, veja a figura abaixo:



Vamos entender que Collection é o topo da API Collections, disponível no pacote java.util. Collections é a API que oferece diversas coleções (interfaces e classes concretas).

As Interfaces e classes concretas que implementam a interface Collection, terão obrigatoriamente que fazer a implementação de diversos métodos que manipulam coleções de dados, tais como adicionar e remover elementos.

As Collections podem ser organizadas e ou ordenadas, mas qual a diferença?

Algumas Collections são organizadas, ou seja, garante que as coleções serão percorridas na mesma ordem em que os elementos foram inseridos, já

quando a Collection é ordenada, esta possui métodos, regras para ordenação dos elementos.

Organizada: LinkedHashSet, ArrayList, Vector, LinkedList, LinkedHashMap

Não organizada: HashSet, TreeSet, PriorityQueue, HashMap, HashTable, TreeMap

Ordenada: TreeSet, PriorityQueue, TreeMap

Não ordenada: HashSet, LinkedHashSet, ArrayList, Vector, LinkedList, HashMap, HashTable, LinkedHashMap.

Alguns conceitos para utilizar as Collections que pode ser realmente de grande ajuda.

Generics

O padrão das Collections é aceitar Generics, ou seja, aceitar qualquer tipo de elemento, inclusive elementos diferentes: Double, String, Integer, ou outro objeto que desejar.

Mas é possível especificar o tipo a ser aceito, da seguinte forma:

Exemplo de Generics

```
//Sintax:
Tipo<tipo> nomeObjeto = new Tipo<tipo>();
///Ex.:
List<String> minhaLista = new ArrayList<String>();
```

For-each

O For-each é um ciclo for, mas que é adaptado para utilização em Collections e outras listas. Ele serve para percorrer todos os elementos de qualquer Collection contida na API Collections.

```
//Sintaxe:

for(tipo elemento:tipo)

//Ex.:

List<Integer> minhaLista = new ArrayList<Integer>();
                minhaLista.add(1);
                minhaLista.add(2);
                for (Integer listaElementos : minhaLista) {
                        System.out.println(listaElementos);
                }
```

Iterator

O iterator é uma interface disponível no pacote java.util que permite percorrer coleções da API Collection, desde que tenham implementado a Collection, fornecendo métodos como o next (), hasnext() e remove().

```
List<Integer> minhaLista = new ArrayList<Integer>();
                minhaLista.add(1);
                minhaLista.add(2);
                Iterator iMinhaLista = minhaLista.iterator();

                for(Integer listaElementos: minhaLista){
                        System.out.println(iMinhaLista.next());
                }
```

As Collections possuem padrões para uso, vamos ver um exemplo no código a seguir:

```
Set<Integer> meuSet = new HashSet<Integer>();
                meuSet.add(1);
                meuSet.add(2);
                meuSet.add(3);
                meuSet.add(1);
                Iterator<Integer> iMeuSet = meuSet.iterator();
                while(iMeuSet.hasNext()){
                        System.out.println(iMeuSet.next());
                }
```

Este código por utilizar o Set, ao exibir os dados, vai ter como resultado 1, 2 e 3, se utilizarmos o List, teríamos como resultado 1,2,3,1 por quê?

Bem cada tipo apresentado, tem uma forma de trabalhar, vamos entender cada uma delas:

- Set: Não aceita itens duplicados, aceita nulos, não possui índice, rápido para inserir e pesquisar elementos;

- HashSet: esta é uma implementação concreta de Set não organizada, ou seja, os elementos são percorridos aleatoriamente, e também não é ordenada, não há regras de ordenação. Além disso, assim como Set, não aceita itens duplicados.

- TreeSet: implementação concreta de Set organizada, mas também não aceita itens duplicados.

- TreeSet: Também é uma implementação concreta, não aceitando itens duplicados, mas é ordenada.

- List: Aceita itens duplicados, organizada e todas as implementações seguem este padrão, elementos percorridos por ordem ser inserção, não ordenada, aceita nulo, trabalha com índices da mesma forma que os

arrays, é rápido para inserir elementos, mas um pouco mais lento que o Set, já as pesquisas são mais lentas que o Set.

- ArrayList: implementação concreta de List, aceita itens duplicados e seu funcionamento é semelhante a um array convencional, com a principal diferença de ser dinâmica, ou seja, pode crescer conforme a necessidade.

- Vector: Outra implementação de List, pode ser visto como uma ArrayList, mas os métodos são sincronizados, ou seja, o acesso simultâneo por diversos processos será coordenado, é também mais lento que o ArrayList quando não há acesso simultâneo.

- LinkedList: outra implementação de List, mas também implementa Queue, aceitando itens duplicado e sendo organizada. É similar ao ArrayList e ao Vector, mas fornece alguns métodos adicionais para a inserção, remoção e acesso aos elementos no inicio e no final da lista. Possui melhor performance do que o ArrayList e o Vector quando se trata de inserir, remover e acessar elementos no inicio ou no final da lista, mas se for precisar acessar algum elemento pelo índice, a performance é muito inferior, sendo lento para pesquisas.

- Queue: é a fila, semelhante à lista, tendo como padrão o aceite duplicado de elementos, normalmente organizado, normalmente utilizado em itens que a ordem é importante, como uma fila de banco.

- PriorityQueue: implementação concreta de Queue e não aceita nulos

- Map: Um Map é um tipo de coleção que identifica os elementos por chaves, desta forma aceita itens duplicados com chaves diferentes.

- HashMap: implementação concreta de Map, aceita itens duplicados com chaves diferentes, não organizado, ou seja, os elementos são percorridos aleatoriamente, não ordenada e aceita nulos.

- HashTables: outra implementação de Map, aceita itens duplicados com chaves diferentes, semelhante ao HashMap mas os métodos são sincronizados.

- TreeMap é uma outra implementação concreta de Map, também suporta itens duplicados com índices diferentes, é ordenado.

Implementando Collections básicas

Segue uma implementação básica de Collections:

```
import java.util.ArrayList;
import java.util.HashSet;
import java.util.Iterator;
import java.util.List;
import java.util.Set;

public class MinhasCollections {
        public static void main(String[] args) {
                List<Integer> minhaLista = new ArrayList<Integer>();
                minhaLista.add(1);
                minhaLista.add(2);
                minhaLista.add(2);

                for (Integer listaElementos : minhaLista) {
                        System.out.println(listaElementos);
                }
                Set<Integer> meuSet = new HashSet<Integer>();
                meuSet.add(1);
                meuSet.add(2);
                meuSet.add(3);
                meuSet.add(1);
                Iterator<Integer> iMeuSet = meuSet.iterator();
                while(iMeuSet.hasNext()){
                        System.out.println(iMeuSet.next());
                }



        }

}
```

As Collections possuem ampla utilização em Java, muito utilizadas para os mais diversos fins. Por serem genéricas por padrão, permite a utilização em diversas soluções. Collection é comumente utilizada como um tipo de parâmetro nos métodos para permitir processamento polimórfico de todos os objetos que implementam a interface Collection.

Data e Hora em Java

O tratamento de data e hora em uma aplicação é geralmente um pesadelo para os programadores de qualquer linguagem. O problema é ainda mais grave se for considerado que praticamente qualquer sistema de informação precisa em algum momento desse tipo de dado. Ao contrário de algumas estruturas de dados mais simples, como inteiros ou strings, datas são tipos bastante irregulares. Algumas das dificuldades encontradas nesse tipo de controle são as seguintes:

1. Meses com dias irregulares;
2. Anos bissextos;
3. Ausência de um ano 0 (vai de -1 a.C para 1 a.C);
4. Dias da semana úteis e não úteis;
5. Feriados nacionais, estaduais e municipais;
6. Minutos e segundos na base 60 (embora medidas mais precisas como centésimos e milésimos de segundo estejam em outra base);
7. Hora AM e PM;
8. Fuso horário;
9. Horário de verão;

O problema com datas e horas é tão comum, que até o famoso bug do ano 2000 foi acarretado pelo armazenamento de anos com dois dígitos (01/07/48) ao invés dos quatro dígitos atuais (01/07/1948). Esperamos que daqui a 7.990 anos os programadores não fiquem muito chateados de não termos considerado a virada para o ano 10.000.

Essa extensa lista de obstáculos no tratamento de datas faz com que operações simples, como adicionar ou subtrair uma hora de uma data, se torne algo complexo. Em Java não poderia ser diferente: a especificação inicial da

Sun conta com vários métodos depreciados (deprecated), representação ambígua e incapacidade em lidar com intervalos e períodos de tempo de forma simples.

Datas e Calendários

De qualquer forma, vale a pena estudar as principais maneiras de manipular objetos do tipo data na linguagem. Java possui a classe java.util.Date para representação de uma data simples. O comando a seguir cria uma instância de um objeto com a data corrente do sistema.

```
Date dt1 = new Date();
```

O tipo Date armazena internamente um long que representa o número de millisegundos que se passaram desde meia-noite do dia 01/01/1970. Ou seja, é possível intercambiar os dois tipos de dados da seguinte forma:

```
long agora = System.currentTimeMillis();
data dtAgora = new Date (agora);
```

E o método getTime () da classe Date retorna também esse long. Então ao imprimir o long agora e a data dtAgora obteremos algo similar aos seguintes resultados:

```
Fri Jan 15 03:35:06 BRST 2010
1263533706375
```

Apenas por curiosidade, o número é um pouco maior que 1.261.440.000.000, que é 40 * 365 * 24 * 60 * 60 * 1000 (40 anos * 365 dias *

24 horas * 60 minutos * 60 segundos * 1000 millisegundos), pois não foram considerados anos bissextos e já se encontrava no dia 15 de janeiro de 2010, algumas horas após a meia-noite. Isso significa que o tipo Date NÃO é para ser operado diretamente, como no exemplo abaixo:

```
Date amanha = new Date(agora.getTime() + Timer.ONE_DAY);
```

Pois não estão sendo considerados fatores como horário de verão (um dia do ano possui 23 horas, e um outro dia possui 25 horas), além de ser muito trabalhoso programar rotinas desse tipo. Felizmente, existe a classe java.util.Calendar que se encarrega de fornecer métodos para manipular as datas de forma eficaz.

```
GregorianCalendar cal = new GregorianCalendar();
cal.setTime(dt1);
```

Por que foi utilizada a classe GregorianCalendar?

GregorianCalendar é uma subclasse da Calendar, e considera o nosso calendário gregoriano, de 365 dias, anos bissextos, etc. A partir de uma instância dessa classe, é possível obter cada informação da data:

```
System.out.print("Ano: " + cal.get(Calendar.YEAR));
System.out.print("Mês: " + cal.get(Calendar.MONTH));
System.out.print("Dia: " +  cal.get(Calendar.DAY_OF_MONTH));
System.out.print("Semana: " + cal.get(Calendar.DAY_OF_WEEK));
```

A saída será:

Ano: 2010, Mês: 0, Dia: 15 e Semana: 6. Janeiro é o mês zero, e o dia da semana 6 é a quinta-feira, pois começa a contagem do domingo.

YEAR, MONTH, DAY_OF_MONTH e DAY_OF_WEEK são constantes definidas na classe Calendar e que referenciam os diferentes campos da data.

É recomendável verificar essas constantes na API da classe (ver referências) para entender o seu funcionamento.

Também é possível usar essas constantes para operações de calendário:

```
cal.add(gc.MONTH, 1) // soma um mês à data
cal.add(gc.HOUR, -3) // substrai 3 horas à data
```

O método cal.isLeapYear () retorna true caso o ano da instância do calendário seja um ano bissexto, e false caso contrário. Os métodos after() e before() podem ser usados para comparar duas datas:

```
if(cal1.after(cal2)) // cal1 é depois de cal2
if(cal1.before(cal2)) // cal1 é antes de cal2
```

Finalmente, a data pode ser retornada ao tipo java.util.Date por meio do método getTime () da classe GregorianCalendar:

```
Date dt = cal.getTime();
```

Entrada e Saída de Datas

Uma operação comum com datas é a leitura a partir da entrada do usuário (por ex: data de nascimento em um formulário web). Outra operação frequente é a impressão de uma data em um formato padronizado e legível para o usuário.

Para essas duas operações são recomendadas o uso da classe SimpleDateFormat (do pacote java.text).

```
Date dts;
String s="20/02/2010";

SimpleDateFormat sdf= new SimpleDateFormat("dd/MM/yyyy");
try {
       dts= sdf.parse(s);
}
catch(ParseException pe) {
       System.out.println(pe.getMessage());
       dts = null;
}

System.out.println(dts);
```

O funcionamento do código acima é o seguinte:

A partir de uma String s (que pode ser obtida por entrada em um formulário), é criado um objeto Date, usando a máscara “dd/MM/yyyy” especificada no objeto do tipo SimpleDateFormat.

Note que o MM em maiúsculas corresponde ao MONTH da classe Calendar. A saída do código é algo como “Sat Feb 20 00:00:00 BRST 2010”, ou seja, 20 de fevereiro de 2010, e a hora por padrão é meia-noite. Outras máscaras podem ser usadas, como a seguir:

```
// 15/01/2010 03:35:16
DateFormat df1 = new SimpleDateFormat("dd/MM/yyyy HH:mm:ss");

// Formato Americano: 01/15/2010
DateFormat df1 = new SimpleDateFormat("MM/dd/yyyy");

// 15/01/2010 03h35
DateFormat df1 = new SimpleDateFormat("dd/MM/yyyy HH'h'mm");
```

A operação inversa, ou seja, imprimir formatado uma data, é bem simples:

```
SimpleDateFormat sdf= new SimpleDateFormat("dd/MM/yyyy");
Date dt1 = new Date();
System.out.print(sdf.format(dt1));
```

A saída será algo do tipo “15/01/2010”.

JPA com Hibernate

Com a popularização do Java em ambientes corporativos, logo se percebeu que grande parte do tempo do desenvolvedor era gasto na codificação de queries SQL e no respectivo código JDBC responsável por trabalhar com elas.

Além de um problema de produtividade, algumas outras preocupações aparecem: SQL que, apesar de ter um padrão ANSI, apresenta diferenças significativas dependendo do fabricante. Não é simples trocar um banco de dados pelo outro.

Há ainda a mudança do paradigma. A programação orientada a objetos difere muito do esquema entidade relacional e precisamos pensar das duas

maneiras para fazer um único sistema. Para representarmos as informações no banco, utilizamos tabelas e colunas. As tabelas geralmente possuem chave primária (PK) e podem ser relacionadas por meio da criação de chaves estrangeiras (FK) em outras tabelas.

Quando trabalhamos com uma aplicação Java, seguimos o paradigma orientado a objetos, onde representamos nossas informações por meio de classes e atributos. Além disso, podemos utilizar também herança, composição para relacionar atributos, polimorfismo, enumerações, entre outros. Esse buraco entre esses dois paradigmas gera bastante trabalho: a todo momento devemos "transformar" objetos em registros e registros em objetos.

Como são chamadas as ferramentas que auxiliam os desenvolvedores nesta “transformação”? Elas tornaram-se popular entre os desenvolvedores Java e são conhecidas como ferramentas de mapeamento objeto-relacional (ORM).

O Hibernate é uma ferramenta ORM open source e é a líder de mercado, sendo a inspiração para a especificação Java Persistence API (JPA). O Hibernate nasceu sem JPA mas hoje em dia é comum acessar o Hibernate pela especificação JPA. Como toda especificação, ela deve possuir implementações. Entre as implementações mais comuns, podemos citar: Hibernate da JBoss, EclipseLink da Eclipse Foundation e o OpenJPA da Apache. Apesar o Hibernate originou o JPA, o EclipseLink é implementação referencial.

O Hibernate abstrai o seu código SQL, toda a camada JDBC e o SQL será gerado em tempo de execução. Mais que isso, ele vai gerar o SQL que serve para um determinado banco de dados, já que cada banco fala um "dialeto" diferente dessa linguagem. Assim há também a possibilidade de trocar de banco de dados sem ter de alterar código Java, já que isso fica de responsabilidade da ferramenta.

Como usaremos JPA abstraímos mais ainda, podemos desenvolver sem conhecer detalhes sobre o Hibernate e até trocar o Hibernate com uma outra implementação como OpenJPA.

Utilizando o Hibernate

Vamos usar o JPA com Hibernate, ou seja, precisamos baixar os JARs no site do Hibernate. O site oficial do Hibernate é o www.hibernate.org, onde você baixa a última versão na seção ORM e Download.

Com o ZIP baixado em mãos, vamos descompactar o arquivo. Dessa pasta vamos usar todos os JARs obrigatórios (required). Não podemos esquecer o JAR da especificação JPA que se encontra na pasta jpa.

Para usar o Hibernate e JPA no seu projeto é necessário colocar todos esses JARs no classpath.

O Hibernate vai gerar o código SQL para qualquer banco de dados. Continuaremos utilizando o banco MySQL, portanto também precisamos o arquivo .jar correspondente ao driver JDBC.

Exemplo de aplicação utilizando Hibernate

Iremos desenvolver uma simples aplicação que realizará as funções básicas do SQL: SELECT, INSERT, UPDATE e DELETE, o famoso CRUD. Vamos utilizar o Java 6, Hibernate 3.6.6, MySQL5 e o NetBeans 7 neste projeto.

Para baixar o Hibernate acesse: http://hibernate.org/downloads.

Primeiramente crie um novo projeto no NetBeans com o nome de sua preferência, aqui daremos o nome de HelloHibernate. Feito isso iremos importar as APIs necessárias para o funcionamento do Hibernate, são elas:

- hibernate3.jar;
- hibernate-jpa-2.0-api-1.0.1.Final.jar;
- antlr-2.7.6.jar;
- commons-collections-3.1.jar;
- dom4j-1.6.1.jar;

- javassist-3.12.0.GA.jar;
- jta-1.1.jar;
- slf4j-api-1.6.1.jar;
- slf4j-simple-1.6.1.jar.

A API slf4j-simple-1.6.1.jar não vem no pacote de instalação do Hibernate, mas ainda assim é necessário para seu funcionamento. Para baixá-la acesse: http://slf4j.org/download.html.

Agora iremos criar as seguintes classes:

- Main (Formulário JFrame);
- HibernateUtil (Classe que realizará a persistência dos dados);
- Sessão (Responsável pela conexão com o Banco de Dados);
- Pessoas (POJO da tabela pessoas que será criada mais a frente).

Nestes campos: nome, idade e sexo que são do tipo, respectivamente, string, integer e character.

Desenvolvimento da interface gráfica da nossa aplicação

A interface gráfica deve ser feita parecida com a figura abaixo:



Lembrando um pouco da API Swing...

Nesta interface encontramos os seguintes componentes Swing:

- JTextField: txtNome;
- JSpinner: spnIdade;
- JRadioButton: rdoM e rdoF (Com o ButtonGroupgrpSexo);
- JButton: btnInserir, btnAtualizar, btnDeletar;
- JTable: tblRegistros.

Banco de Dados

O nome do Banco de Dados será hello_hibernate e o nome da tabela pessoas. A estrutura da tabela ficará da seguinte forma:

Mapeando o banco de dados

Chegou a hora de criarmos o POJO, a classe responsável por mapear a tabela pessoas. Ela deve ficar da seguinte forma:

```
public class Pessoas {
    // ATRIBUTOS
    private int id;
    private String nome;
    private int idade;
    private char sexo;

    // CONSTRUTOR
        public Pessoas() {
    }


    // GETTERS
    public int getId() {
        return id;
    }
    public String getNome() {
        return nome;
    }
    public int getIdade() {
        return idade;
    }
    public char getSexo() {
        return sexo;
    }


    // SETTERS
    public void setId(int id) {
        this.id = id;
    }
    public void setNome(String nome) {
        this.nome = nome;
    }
    public void setIdade(int idade) {
        this.idade = idade;
    }
    public void setSexo(char sexo) {
        this.sexo = sexo;
    }
}
```

Até o momento apenas declaramos os quatro atributos com seus respectivos tipos e com o modificador de acesso private. Em seguida criamos os métodos getters e setters para cada atributo.

Configurando o Hibernate

Agora iremos criar as pessoas.hbm.xml, o arquivo contendo o mapeamento da tabela pessoas com os tipos de dados do Hibernate e que também informa o POJO responsável pelo mapeamento, neste caso a classe Pessoas. O arquivo deve ficar desta forma:

```
<?xml version="1.0" encoding="UTF-8"?>
http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<hibernate-mapping>
    <class name="Pessoas" table="pessoas">
        <id name="id">
            <generator class="native"/>
        </id>
        <property name="nome" type="string" length="50"/>
        <property name="idade" type="integer" length ="3"/>
        <property name="sexo" type="character" length ="1"/>
    </class>
</hibernate-mapping>
```

Feito isso, iremos criar o hibernate.cfg.xml, arquivo responsável pela conexão com o Banco de Dados e pelo comportamento geral do Hibernate. Iremos informar o Dialeto, Driver e URL do Banco de Dados, usuário, senha e habilitaremos as opções de mostrar o SQL no console da IDE com formatação (Apenas para facilitar o entendimento do código) e a geração de estatísticas.

Repare que no final deste arquivo indicamos o local das pessoas.hbm.xml na propriedade resource da Tag mapping.

O arquivo deve ficar da seguinte forma:

```
<?xml version="1.0" encoding="UTF-8"?>
http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
    <session-factory>
        <property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>
        <property name="hibernate.connection.driver_class">com.mysql.jdbc.Driver</property>
        <property
name="hibernate.connection.url">jdbc:mysql://localhost:3306/hello_hibernate</property>
        <property name="hibernate.connection.username">root</property>
        <property name="hibernate.connection.password"/>
        <property name="hibernate.show_sql">true</property>
        <property name="hibernate.format_sql">true</property>
        <property name="hibernate.generate_statistics">true</property>

        <!-- ARQUIVO DE MAPEAMENTO -->
        <mapping resource="pessoas.hbm.xml"/>
    </session-factory>
</hibernate-configuration>
```

Implementando a classe da sessão

A classe Sessão será responsável por ler o arquivo hibernate.cfg.xml e estabelecer a conexão com o Banco de Dados. Nela usaremos apenas três classes: Configuration, SessionFactory e Session.

A classe Configuration é responsável por ler o arquivo hibernate.cfg.xml e iniciar a sessão propriamente dita.

A classe SessionFactory possui um alto custo de criação, é deve ser criado uma única vez, no início da execução da aplicação, a partir da instância de uma Configuration.

Uma Session é um objeto de baixo custo de criação e deve ser usado uma vez, para uma única requisição e então deve ser descartada (sessao.close()).

Uma transação precisa ser o mais curta possível, para reduzir a disputa pelo bloqueio na Base de Dados. Transações longas impedirão que sua aplicação seja altamente concorrente.

A classe Sessão deve ficar da seguinte maneira:

```
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.cfg.Configuration;
public class Sessao {
    private static SessionFactory fabricaSessao;
    private static Configuration hibernateConfig;
    // Estrutura static para garantir que a SessionFactory seja iniciada apenas uma vez
    static {
        try {
            hibernateConfig = new Configuration().configure("hibernate.cfg.xml");
            fabricaSessao = hibernateConfig.buildSessionFactory();
            atualizarBD();
        } catch (Exception e){
            e.printStackTrace();
        }
    }
    // Retorna a sessão
    public static Session getSessao(){
        return fabricaSessao.openSession();
    }

    // Atualiza o Schema do Banco de Dados
    private static void atualizarBD(){
        SchemaUpdate se = new SchemaUpdate(hibernateConfig);
        se.execute(true, true);
    }
}
```

O método atualizarBD irá ser chamado assim que a conexão com o Banco de Dados for efetuada. Ele irá garantir que o nosso mapeamento seja "refletido" no banco de dados.

Implementando a classe de persistência

Feita a classe Sessão iremos implementá-la na classe HibernateUtil junto com os métodos select, insertUpdate e delete.

Nesta classe precisaremos das classes Session, Transaction, Query e List.

A Session nos permite acessar a SessionFactory. A Transaction é quem iniciará a Session e executará a Query. Ao realizar o SELECT será retornada uma List.

Veja como deve ficar a classe:

```
import java.util.ArrayList;
import java.util.List;
import javax.swing.JOptionPane;
import org.hibernate.HibernateException;
import org.hibernate.Query;
import org.hibernate.Session;
import org.hibernate.Transaction;
public class HibernateUtil {
    private Session sessao;
    private Transaction transacao;
    private Query query;

    public List select(){
        List<Pessoas> lista = new ArrayList();
        try {
            this.sessao = Sessao.getSessao();
            transacao = sessao.beginTransaction();
            query = sessao.createQuery("from Pessoas");
            lista = query.list();
            sessao.close();
        } catch (HibernateException e) {
            JOptionPane.showMessageDialog(null, "Erro ao realizar Select.\n"+e.getMessage(), null, JOptionPane.ERROR_MESSAGE);
           e.printStackTrace();
       }
       return lista;
   }

    public void insertUpdate(Pessoas pessoa){
        try{
            this.sessao = Sessao.getSessao();
            transacao = sessao.beginTransaction();
            sessao.saveOrUpdate(pessoa);
            transacao.commit();
            sessao.close();
            JOptionPane.showMessageDialog(null, "Registro salvo com sucesso!", null, JOptionPane.INFORMATION_MESSAGE);
        } catch (HibernateException e) {
            JOptionPane.showMessageDialog(null, "Erro ao persistir os dados.\n"+e.getMessage(), null, JOptionPane.ERROR_MESSAGE);
            e.printStackTrace();
        }
    }
    public void delete(Pessoas pessoa){
        try{
            this.sessao = Sessao.getSessao();
            transacao = sessao.beginTransaction();
            sessao.delete(pessoa);
            transacao.commit();
            sessao.close();
            JOptionPane.showMessageDialog(null, "Registro deletado com sucesso!", null, JOptionPane.INFORMATION_MESSAGE);
        } catch (HibernateException e) {
            JOptionPane.showMessageDialog(null, "Erro ao deletar registro.\n"+e.getMessage(), null, JOptionPane.ERROR_MESSAGE);
            e.printStackTrace();
        }
    }
}
```

Entendendo o código acima.

Criamos três métodos: select, insertUpdate e delete.

1. Em select utilizamos o HQL para realizar uma consulta que retornará uma List.
2. Em insertUpdate utilizamos o método saveOrUpdate() que realiza tanto o INSERT quanto o UPDATE (apenas se o atributo id já existir).
3. Em delete utilizamos o método delete() para apagar determinado registro.

Note que tanto em insertUpdate e delete passamos por parâmetro o POJO Pessoas e no método select retornamos uma List<Pessoas>. Observe também que utilizamos o sessao.beginTransaction() para iniciar a sessão, em seguida executamos a query com o transacao.commit() e fechamos a sessão, sessao.close(). Isso é muito importante pois, como vimos anteriormente, uma transação tem que ser o mais curta possível.

Implementando as classes

Agora vamos implementar o HibernateUtil e outros métodos na classe principal Main, tornando a aplicação funcional. Vamos começar com a declaração dos atributos e do método construtor Main.

```
private HibernateUtil hibernate;
private Pessoas pessoas;
private DefaultTableModel tblModel;
private int id;

public Main() {
    hibernate = new HibernateUtil();
    pessoas = new Pessoas();
    initComponents();
    setLocationRelativeTo(null);
    setListeners();
    atualizarTabela();
}
```

Entendendo o código acima.

Instanciamos o HibernateUtil e o POJO Pessoas;

Iniciamos o componente Swing;

Centralizamos a tela;

Adicionamos os listeners necessários (A seguir);

Atualizamos a tabela (A seguir).

Agora iremos criar os métodos limparCampos e validarCampos:

```
private void limparCampos(){
     id = 0;
     txtNome.setText("");
     spnIdade.setValue(0);
     grpSexo.clearSelection();
}

private boolean validarCampos(){
     if(txtNome.getText().isEmpty()){
          JOptionPane.showMessageDialog(null, "Informe o nome", null,    JOptionPane.WARNING_MESSAGE);
          return false;
     } else if((Integer)spnIdade.getValue() < 1){
          JOptionPane.showMessageDialog(null, "Informe uma idade válida", null,  JOptionPane.WARNING_MESSAGE);
          return false;
     } else if(!rdoF.isSelected() && !rdoM.isSelected()){
          JOptionPane.showMessageDialog(null, "Informe o sexo", null,    JOptionPane.WARNING_MESSAGE);
          return false;
     }
     return true;
}
```

Feito isso, o próximo passo é criar os métodos setPessoas (Que irá pegar os valores do formulário e setar no POJO) e atualizarTabela (Que chamará o método select do HibernateUtil e preencherá a tabela sempre que um evento for disparado).

```
private void setPessoas() {
    pessoas.setId(id);
    pessoas.setNome(txtNome.getText());
    pessoas.setIdade((Integer)spnIdade.getValue());
    if(rdoM.isSelected())
    pessoas.setSexo('M');
    else if(rdoF.isSelected())
   pessoas.setSexo('F');
}

private void atualizarTabela(){
    try {
        limparCampos();
        tblModel.getDataVector().clear();
        List<Pessoas> select = hibernate.select();
        if(!select.isEmpty()){
       for(Pessoas p : select)
                tblModel.addRow(new Object[]{p.getId(), p.getNome(),  p.getIdade(), p.getSexo()});
        }
    } catch (Exception e) {
        JOptionPane.showMessageDialog(null, "Erro ao atualizar     tabela\n"+e.getMessage(), null, JOptionPane.ERROR_MESSAGE);
    }
}
```

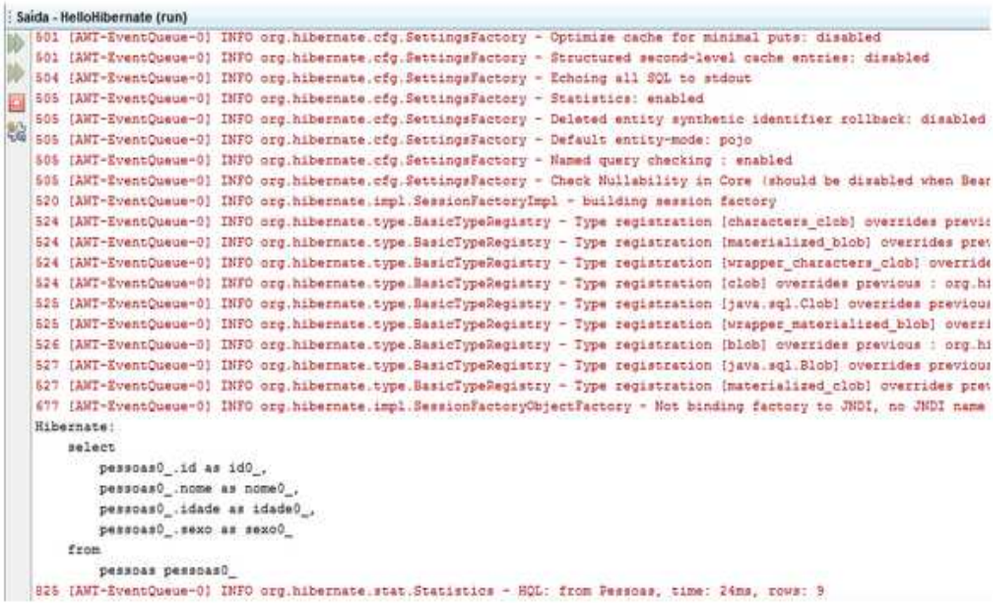
Por último iremos adicionar os listeners nos botões e na tabela. Nos botões btnInserir, bntAtualizar e btnDeletar iremos adicionar um ActionListener que, primeiramente, realizará uma validação e caso a mesma seja positiva executará o setPessoas, depois o método do HibernateUtil (insertUpdade ou delete) e por último atualizará a tabela.

Na tabela adicionaremos um MouseListener e sobrescreveremos o método mouseClicked. Ao clicarmos em uma linha, os campos nome, idade e sexo serão preenchidos e o atributo id terá seu valor atualizado.

```
private void setListeners() {
btnInserir.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            if(validarCampos()){
                id = 0;
                setPessoas();
                hibernate.insertUpdate(pessoas);
                atualizarTabela();
            }
        }
    });
    btnAtualizar.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            if(validarCampos()){
                setPessoas();
                hibernate.insertUpdate(pessoas);
                atualizarTabela();
            }
        }
    });
    btnDeletar.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            if(id > 0){
                setPessoas();
                hibernate.delete(pessoas);
                atualizarTabela();
            }
        }
    });
    tblRegistros.addMouseListener(new MouseAdapter() {
        @Override
        public void mouseClicked(MouseEvent e) {
            int linha = tblRegistros.getSelectedRow();
            id = (Integer)tblRegistros.getModel().getValueAt(linha, 0);
            txtNome.setText(tblRegistros.getModel().getValueAt(linha,1)+"");
            spnIdade.setValue(tblRegistros.getModel().getValueAt(linha, 2));
            if((Character)tblRegistros.getModel().getValueAt(linha, 3)=='M')
                rdoM.setSelected(true);
            else if((Character)tblRegistros.getModel().getValueAt(linha, 3)=='F')
                rdoF.setSelected(true);
        }
    });
}
```

Com isso feito nossa aplicação já está pronta para ser testada.

Agora vamos dar um Build (F11) no projeto e observar no console do NetBeans se houve alguma exceção. Vamos rodar a aplicação (F6) e novamente observar o console. Agora você está diante do Log de inicialização do Hibernate que deve ser parecido com a figura abaixo.



É muito importante prestarmos atenção neste Log, pois nele encontramos informações úteis a respeito de sua configuração e a possíveis HibernateException que podem vir a acontecer.

Agora observe o final do Log, mais precisamente na consulta SQL que foi feita: um SELECT gerado automaticamente. Lembram-se que no método construtor da classe principal Main chamamos o método atualizarTabela? Se você estiver vendo esta consulta SQL em seu console e não houve nenhum HibernateException significa que o Hibernate está configurado perfeitamente e que o SELECT foi executado sem erros.

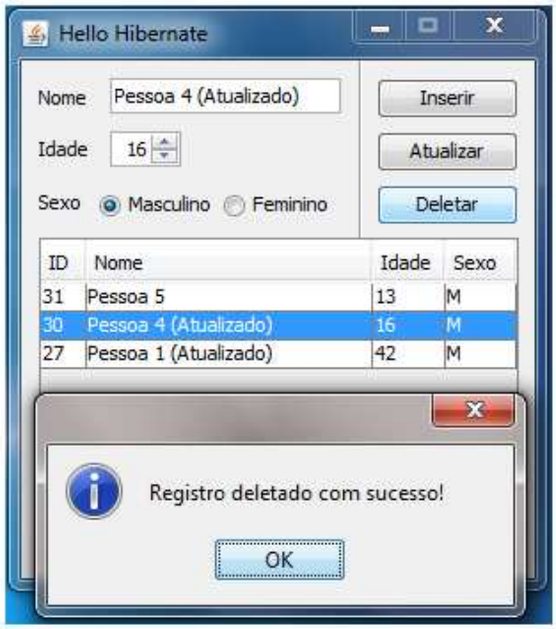
Vamos voltar para a aplicação. Como ainda não temos nenhum registro no Banco de Dados, mesmo com o SELECT sendo feito, temos uma tabela vazia na aplicação. Para resolver isso vamos inserir alguns registros.



Enquanto você inseria os registros, você observou o console do NetBeans com as consultas SQL (INSERT e SELECT) sendo geradas automaticamente? Até agora não precisamos nos preocupar com nenhum código SQL, não é mesmo? E vamos continuar assim graças ao Hibernate.

Agora vamos testar o UPDATE e DELETE:





Acabamos de persistir os dados com o Hibernate provando a facilidade de uso e eficiência deste incrível FrameWork. Se compararmos com o modo tradicional (JDBC, ResultSet, Next), o Hibernate nos livra da mão de obra e stress de construir diversas consultas SQL que muitas vezes precisam ser refeitas e podem acabar complicando o entendimento do código e comprometendo sua integridade.

Com pouco tempo de estudo e mínimos conhecimentos em Banco de Dados, podemos desenvolver aplicações magníficas que se beneficiam do Paradigma da Orientação a Objetos para persistir os dados em praticamente qualquer Banco de Dados relacional utilizado atualmente. Isso mesmo, aprendendo Hibernate aprenderemos a trabalhar também com o Oracle, MySQL, PosgreSQL, SQL Server, SQLite, Firebird e muitos outros SGBDs famosos pela sua segurança e robustez.

API Java - String Tokenizer

Para quebrar uma string em tokens (conjunto de caracteres separado por algum caractere delimitador pré-definido) devemos utilizar a classe StringTokenizer. Essa classe pode ser empregada, por exemplo, na migração de dados em um arquivo texto para banco de dados.

O exemplo abaixo ilustra o uso da classe StringTokenizer para separar tokens de uma string. Este exemplo usa o método construtor que separa os tokens com delimitadores default que são: tab, nova linha, carriage-return, form-feed.

```
import java.util.*;
class BuscaPalavrasEmUmaString {
       public static void main (String args[ ]) {
              String texto = "Isto é um texto de teste";
              // usando delimitadores default
              StringTokenizer st = new StringTokenizer (texto);
              // usando # como delimitador
          // StringTokenizer st = new StringTokenizer (texto, "#");
              while (st.hasMoreTokens( )) {
                     String palavra = st.nextToken();
                     System.out.println (palavra);
              }
       }
}
```

Exercitando com o StringTokenizer

Faça um programa que busca palavras por palavra dentro de um texto e as imprima em maiúscula. Considere um texto uma string inicial.

API Java – Math

Já usamos, por diversas vezes, as classes String e System. Vimos o sistema de pacotes do Java e nunca precisamos dar um import nessas classes.

Isso ocorre porque elas estão dentro do pacote java.lang, que é automaticamente importado para você. É o único pacote com esta característica.

Na classe Math, existe uma série de métodos estáticos que fazem operações com números como, por exemplo, arredondar (round), tirar o valor absoluto (abs), tirar a raiz (sqrt), calcular o seno (sin) e outros.

```
double d = 4.6;
long i = Math.round(d);

int x = -4;
int y = Math.abs(x);
```

Consulte a documentação para ver a grande quantidade de métodos diferentes.

No Java 5.0, podemos tirar proveito do import static aqui:

```
import static java.lang.Math.*;
```

Isso elimina a necessidade de usar o nome da classe, sob o custo de legibilidade:

```
double d = 4.6;
long i = round(d);
int x = -4;
int y = abs(x);
```

Referencias

http://pt.wikipedia.org/wiki/Swing_(Java)

http://www.caelum.com.br

http://www.devmedia.com.br

http://www.rbper.com/

http://adrielcafe.com

CURSO TÉCNICO EM

INFORMÁTICA

ETAPA 3

TECNOLOGIA EM SISTEMAS DE INFORMAÇÃO

Sumário

SISTEMAS CORPORATIVOS 300

DEFINIÇÃO DE CORPORAÇÃO 300

EXEMPLOS DE CORPORAÇÕES 300

DEFINIÇÃO DE SISTEMAS DE INFORMAÇÃO 300

O VALOR DA INFORMAÇÃO 302

CARACTERÍSTICAS DE UM SISTEMA DE INFORMAÇÃO 303
- DADOS PERSISTENTES 303
- ACESSO CONCORRENTE 304
- INTEGRAÇÃO COM OUTRAS APLICAÇÕES 305
- REGRAS E LÓGICAS DE NEGÓCIOS 305
- UMA FIGURA CLÁSSICA 305

ARQUITETURA DE SOFTWARE 306

ARQUITETURA CLIENT SERVER 307
- VANTAGENS 308
- PROBLEMAS 308

ARQUITETURA COM 3/4* CAMADAS WEB - BASED 309
- VANTAGENS 309

SERVIDOR DE APP X SERVIDOR WEB 309

ARQUITETURA WEB 311
- INFRAESTRUTURA: HARDWARE, SERVIÇOS E PROTOCOLOS 311
- SOFTWARE: NAVEGADOR, SERVIDORES E LINGUAGENS 311
- VANTAGENS 312

TECNOLOGIA JEE 312

COMPONENTES JEE (ABORDADOS NO CURSO) 313
- VANTAGENS 313
- MULTICAMADAS 314
- CAMADA CLIENTE 314
- CAMADA WEB 315
- CAMADA DE NEGÓCIOS 315
- CAMADA EIS 315

CONTAINERS 316
WEB TIER – A CAMADA WEB 317
CICLO DE VIDA WEB APP 320
WEB MODULE – ESTRUTURA 321
WAR 321

AMBIENTE DE DESENVOLVIMENTO WEB 322
COMPONENTES 322
CONFIGURAÇÃO 322

PROTOCOLO HTTP 323
O QUE É URL? 325
PROTOCOLO: 325
MÁQUINA OU SERVIDOR: 325
CAMINHO: 325

RECURSO: 325
MÉTODOS 326
GET 326
POST 327

GET X POST 328

HTML E SERVLET 331

SERVLET 332

ESTRUTURA DE CLASSES 334

CICLO DE VIDA DE UM SERVLET 337

WEB.XML 338

SERVLETCONFIG 339

SERVLETCONTEXT 339

ALGUNS MÉTODOS DA INTERFACE HTTPSERVLETREQUEST 340

PARÂMETROS DA REQUISIÇÃO 340

COMO LER OS PARÂMETROS DA REQUISIÇÃO 341

ALGUNS MÉTODOS DA INTERFACE HTTPSERVLETRESPONSE 341

COMO GERAR UMA RESPOSTA 342

MAIS SOBRE OS PARÂMETROS DA REQUISIÇÃO 342

JSP – JAVA SERVER PAGES 344
- O JSP É BASEADO EM SERVLET 345
- ELEMENTOS JSP 346
- A DIRETIVA PAGE 351
- A DIRETIVA TAGLIB 351
- A DIRETIVA INCLUDE 351

SESSÕES E COOKIES 352
- TÉCNICAS PARA CONTROLE DE SESSÃO 352
- COOKIES 353
- TIPOS DE COOKIES 354
- SESSION 354

O PADRÃO ARQUITETURAL MVC – MODEL VIEW CONTROLLER 357
- SERVLET OU JSP? 357
- RETIRANDO O CÓDIGO JAVA DOS JSP'S: 358
- EL – EXPRESSION LANGUAGE 358
- REQUEST DISPATCHER 359
- JSTL JAVASERVER PAGES STANDARD TAG LIBRARY 361
- CAMADA DE MODELO (MODEL) 365
- CAMADA DE CONTROLE (CONTROLLER) 365
- CAMADA DE VISÃO (VIEW) 365

PADRÃO DAO – DATA ACCESS OBJECT 366

PADRÃO BO – BUSINESS OBJECT 367

Objetivo

Esta apostila tem por objetivo, ajudar ao aluno na compreensão:

- Conceitos de Sistemas de Informação;
- Da evolução arquitetural Cliente Servidor;
- Arquitetura Web e JEE;
- Camada Web – Servlet + JSP;
- Apresentação do modelo arquitetural MVC.

Sistemas Corporativos

Definição de corporação

Organismo social que congrega todos os membros de uma mesma profissão. Associação de pessoas sujeitas às mesmas regras, obrigações, direitos, deveres, privilégios etc.f. Conjunto de pessoas, sujeitas à mesma regra ou estatuto. Indivíduos que, coletivamente, administram ou dirigem certos negócios de interesse público. Associação. (Dicionário Web)

Exemplos de Corporações



Definição de Sistemas de Informação

Segundo Buckingham et al. 1987, um Sistema de Informação é uma entidade sociotécnica que junta, armazena, processa e disponibiliza informação relevante para uma organização de modo a torná-la acessível e útil para quem a deseje e possa utilizar.

Em resumo:

Foco na Informação

+

Pessoas, processos, informações e documentos.

+

Máquinas, computadores e redes de comunicação.

Carvalho (1996) define a entidade sociotécnica:

A componente técnica corresponde aos aspectos mais ligados às tecnologias da informação e aos métodos, técnicas e ferramentas apropriadas à definição e construção de sistemas de informação. Sob um ponto de vista estritamente técnico, o objetivo do desenvolvimento de sistemas de informações é a procura da eficiência da organização, procurando tirar partido das potencialidades das tecnologias.

A componente social do desenvolvimento dos SI´s corresponde principalmente aos aspectos relacionados com o impacto que este processo tem nas pessoas e grupos que irão ser afetados pela intervenção. Refiram-se em particular as seguintes situações onde a componente social deverá ser considerada:

- Sendo um processo de intervenção, o DSI irá introduzir mudança num determinado contexto organizacional; diversas pessoas irão ser afetadas por essa mudança;
- A “interface” dos sistemas de informação que eventualmente serão construídos deverá ser adequada às tarefas no contexto das quais serão utilizados e às capacidades dos seus utilizadores;
- A introdução de mudança, em particular através da adoção de sistemas de informação obriga a que as pessoas (os futuros utilizadores dos sistemas), sejam devidamente treinadas na utilização dos mesmos.

Um sistema de informação vai muito além do software, segundo Bezerra (2006):

"O objetivo principal e final da construção de um sistema de informações é adição de valor à empresa ou organização na qual esse sistema será utilizado." Deixar de lado o fator social, pode significar o fracasso de um SI.

O VALOR DA INFORMAÇÃO

O valor da informação pode ser classificado nos seguintes tipos (Cronin, 1990):

- Valor de uso: baseia-se na utilização final que se fará com a informação;
- Valor de troca: é aquele que o usuário está preparado para pagar e variará de acordo com as leis de oferta e demanda, podendo também ser denominado de valor de mercado;
- Valor de propriedade, que reflete o custo substitutivo de um bem;
- Valor de restrição, que surge no caso de informação secreta ou de interesse comercial, quando o uso fica restrito apenas a algumas pessoas.

Assim, a informação terá valor econômico para uma organização, se ela gerar lucros ou for alavancadora de vantagem competitiva. Cronin (1990) afirma que, de modo geral, a percepção de valor pode ser influenciada pelos seguintes fatores:

- Identificação de custos;
- Entendimento da cadeia de uso;
- Incerteza associada ao retorno dos investimentos em informação;
- Dificuldade de se estabelecerem relações causais entre os insumos de informação e produtos específicos;
- Tradição de se tratar a informação como uma despesa geral;
- Diferentes expectativas e percepções dos usuários;
- Fracasso em reconhecer o potencial comercial e o significado da informação

O valor da informação é uma função do efeito que ela tem sobre o processo decisório. Se a informação adicional resultar em uma decisão melhor, então ela terá valor. Caso contrário, ela terá pouco ou nenhum valor (Wetherbe, 1987).

Para sobreviver no próximo século e não ser expulsa do mercado, uma empresa precisa fornecer valor superior aos seus clientes. Fornecer valor é dar aos clientes tudo o que eles querem e nada do que não querem; a melhor qualidade e os melhores preços; seja rápido e sirva o cliente com um sorriso! Os clientes atuais, sofisticados e atentos, fazem quatro julgamentos quando decidem fazer negócio com uma organização: (1) qualidade, (2) serviço, (3) custos e (4) tempo (Band, 1997).

“É preciso qualificar estes quatro fatores dando enfoque particular à essência desse sistema, que é a informação. Estes fatores transformam-se em qualidade da informação, portfólio de produtos e serviços, custo e tempo de resposta. Um sistema de informação para fornecer valor superior deverá disponibilizar a seus clientes todos estes fatores simultaneamente” (Moresi, 2000).

CARACTERÍSTICAS DE UM SISTEMA DE INFORMAÇÃO

Dados persistentes

- **Dados:** são factos que descrevem os objetos de informação (eventos e entidades). Os dados referem-se a mais de um facto. Um determinado facto é referido como item.
- **Informação:** constituída por um conjunto de dados com características específicas, isto é, trata-se de um conjunto de dados significativos e relevantes para a componente ou sistema a quem se destinam.
- **Dados significativos:** para serem significativos, os dados devem ser representados por símbolos compreensíveis, devem ser completos e devem expressar ideias não ambíguas.

- **Dados relevantes:** são os dados que podem ser utilizados na resolução dos problemas propostos.

Através de um bom sistema de banco de dados, é possível transformar uma grande base de dados em vantagem competitiva elaborando um sistema que atue no sentido de agrupar informações que demonstrem alterações de padrões. Tomemos como exemplo uma cadeia de supermercado, que analisando as informações referentes às saídas de mercadorias, identifica um crescimento na venda de carne nos finais de semana. Através de um relacionamento dos dados, a rede descobre que grande parte das pessoas que compram carne, também levam carvão e bebidas. Com base nestas informações, a rede pode traçar estratégias de vendas mais elaboradas. Pode inclusive, decidir por evitar colocar o carvão em oferta nos finais de semana.

Acesso concorrente

Uma das qualidades dos sistemas desenvolvidos é a multiprogramação, que permite a execução de diversas transações visando o compartilhamento do processador. Nesse ambiente multiprogramado, diversas transações podem executar concorrentemente. Por isso os sistemas precisam controlar a interação entre transações concorrentes com o objetivo de prevenir a violação da consistência do banco de dados. Este controle é feito por um conjunto de mecanismos definidos como esquemas de controle de concorrência.

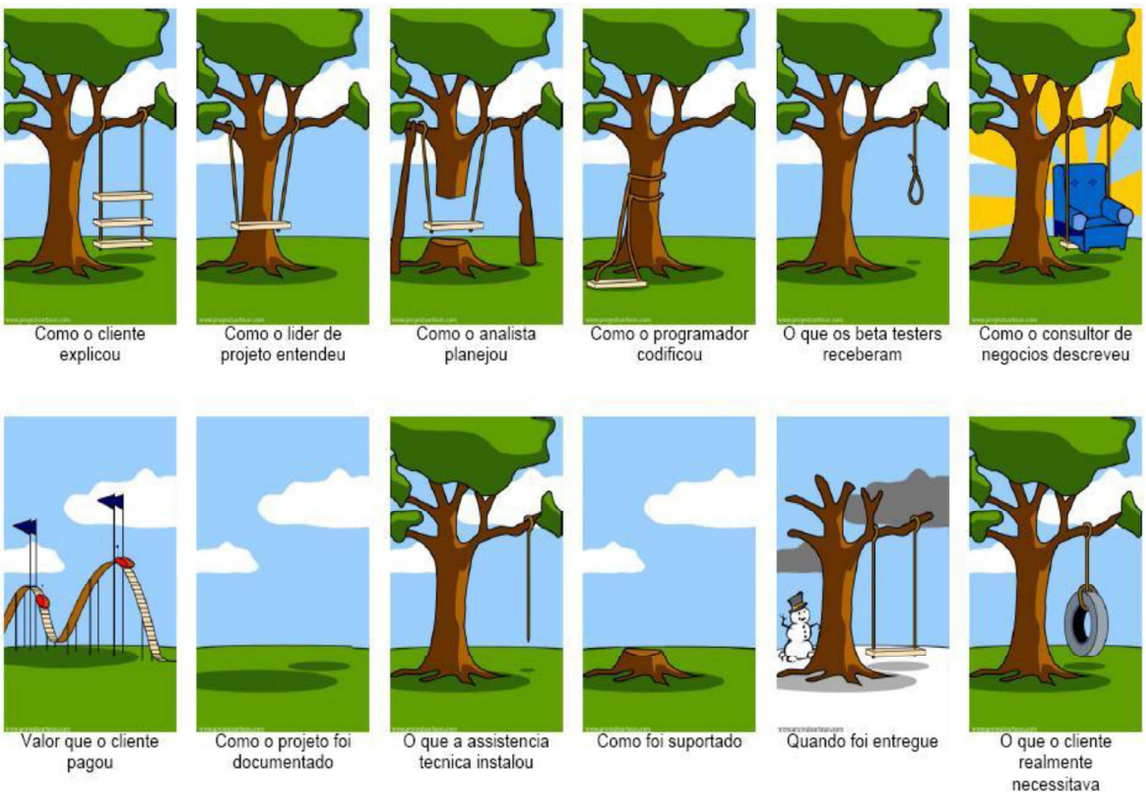
Integração com outras aplicações

O desafio atual é desenvolver um ambiente de comunicação, que permita aos diversos Sistemas de Informação da organização trocar dados de forma eficaz, atendendo à crescente demanda dos processos de negócio por comunicação instantânea, ou seja, em tempo real. (Murphy, 2003)

Regras e lógicas de negócios

“Por princípio, regras de negócio definem as diretrizes do negócio de uma organização, com base em condições específicas. Elas são utilizadas para orientar em qual direção seguir, face a uma situação determinada. Para entendermos melhor, vamos imaginar o cenário de um banco, por exemplo, que tem regras específicas para seus processos internos. No protocolo bancário de abertura de contas, por exemplo, pode haver uma regra de negócio (nesse caso restritiva) na qual clientes menores de dezoito anos não podem abrir contas. Esse é um exemplo clássico do que é uma regra de negócio e sua aplicação.” (Neto, 2011) ·.

Uma figura clássica



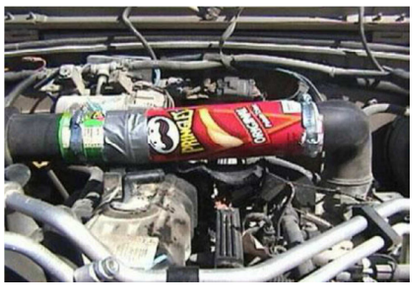
Arquitetura de software

Arquitetura de um software ou sistema computacional é a estrutura dos componentes, seus inter-relacionamentos, princípios e diretrizes que guiam o projeto e a evolução do produto ao longo do tempo.



"As empresas de desenvolvimento de sistemas de grande porte vêm apresentando grandes problemas e desejam produzir software de qualidade. Na visão do usuário, software de qualidade é aquele que além de satisfazer as suas necessidades, é feito no custo e prazo combinados. Já na visão do projetista de software, um bom software é aquele que apresenta as seguintes características:" (SILVA, 2003).

- Maior flexibilidade - Possibilita satisfazer novos requisitos de negócio (funcionalidade) de forma fácil e rápida.
- Melhor adaptabilidade - Possibilita personalizar uma aplicação para vários usuários, usando várias alternativas para oferecer os serviços da aplicação com o mínimo de impacto no seu núcleo.
- Melhor manutenibilidade - Possibilita alterar partes de uma aplicação, de modo que as outras partes sofram um impacto mínimo.
- Melhor reusabilidade - Possibilita montar aplicações únicas e dinâmicas rapidamente.
- Melhor aproveitamento do legado - Possibilita reusar a funcionalidade de sistemas legados em novas aplicações.
- Melhor interoperabilidade - Possibilita que duas aplicações que executam em plataformas diferentes troquem informações.
- Melhor escalabilidade - Possibilita distribuir e configurar a execução da aplicação de modo a satisfazer a vários volumes de transação.
- Melhor robustez - Possibilita implementar soluções de software com menos defeitos.





Arquitetura Client Server

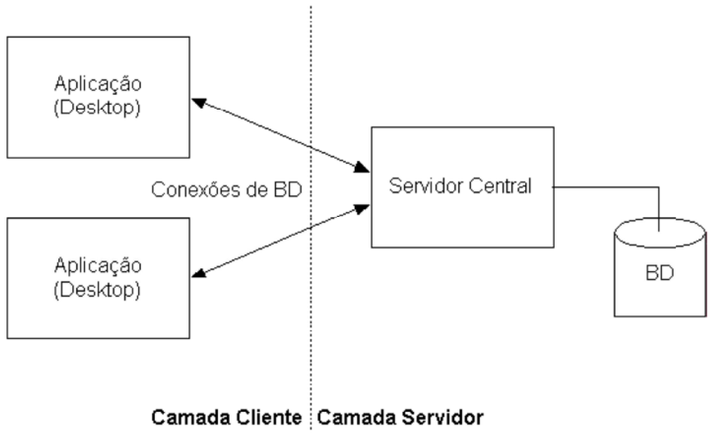


O termo Cliente/Servidor refere-se ao método de distribuição de aplicações computacionais através de muitas plataformas. Tipicamente essas aplicações estão divididas entre um provedor de acesso e uma central de dados e numerosos clientes contendo uma interface gráfica para usuários para acessar e manipular dados. Geralmente refere-se a um modelo onde dois ou mais computadores interagem de modo que um oferece os serviços aos outros.

Este modelo permite aos usuários acessarem informações e serviços de qualquer lugar.

Vantagens

- Melhor aproveitar os PCs da empresa
- Oferecer sistemas com interfaces gráficas amigáveis
- Integrar o desktop e os dados corporativos
- Os primeiros sistemas cliente-servidor eram de duas camadas

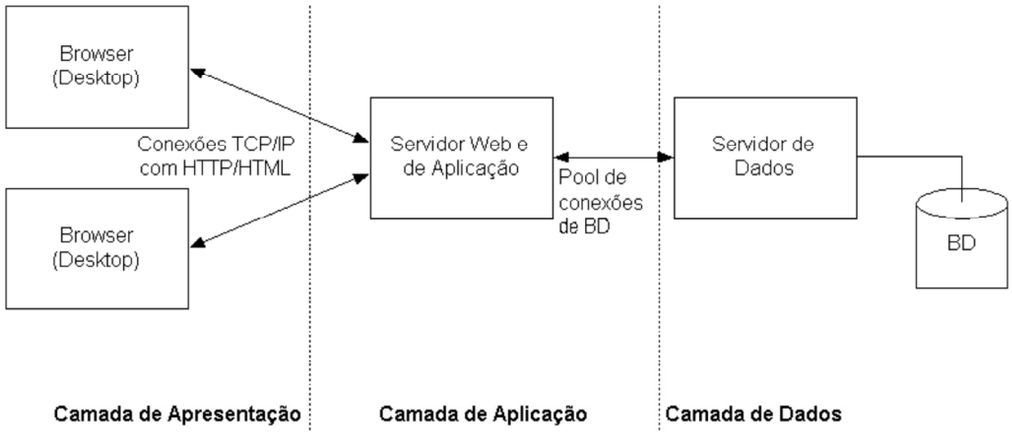


Problemas

- Falta de escalabilidade (conexões a bancos de dados)
- Enormes problemas de manutenção (mudanças na lógica de aplicação forçava instalações)
- Dificuldade de acessar fontes distintas (Outros sistemas)
- A instalação inicial dos programas no desktop é cara

Na tentativa de suprir os problemas o modelo sofre alterações e evolui:

Arquitetura com 3/4* camadas Web - Based



* Às vezes, continua-se a chamar isso de 3 camadas porque as camadas Web e Aplicação frequentemente rodam na mesma máquina.

Vantagens

- Browser como Cliente Universal
- Conceito de Intranet
- A camada de aplicação se quebra em duas: Web e Aplicação
- Evitamos instalar qualquer software no desktop e, portanto, problemas de manutenção.
- Evitar instalação em computadores de clientes, parceiros, fornecedores, etc.

http://www.dsc.ufcg.edu.br/~jacques/cursos/j2ee/html/intro/intro.htm

Servidor de App x Servidor Web

- Um servidor Web lida com o protocolo HTTP. Quando o servidor Web

recebe uma solicitação HTTP, ele responde com uma resposta HTTP, por exemplo, com o envio de volta uma página HTML.

- O servidor Web não fornece nenhuma funcionalidade além de simplesmente oferecer um ambiente em que o programa do lado do servidor pode executar e passar de volta as respostas geradas.
- Servidor de App é um servidor que disponibiliza um ambiente para a instalação e execução de certas aplicações. Em java, implementam a especificação JEE.
- Exemplos:
    - Comerciais: WebSphere Application Server – IBM, WebLogic – BEA Systems.
    - Livres: JBoss, Apache Tomcat.

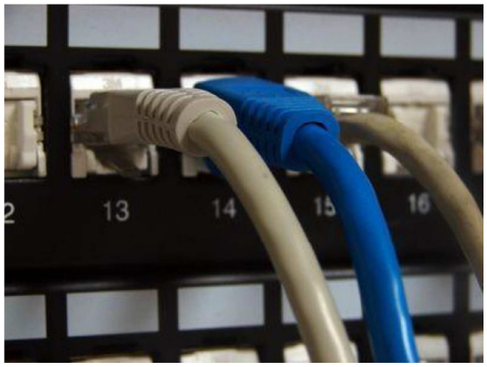
Concluímos que enquanto um servidor Web lida principalmente com o envio de HTML para exibição em um navegador da Web, um servidor de aplicativo fornece acesso à lógica de negócios para uso em aplicativos do cliente.

Existem diversas dessas implementações. Já que esse software tem papel de servir sua aplicação para auxilia-la com serviços de infraestrutura, esse software ganha o nome de servidor de aplicação. A própria Sun/Oracle desenvolve uma dessas implementações, o Glassfish que é open source e gratuito, porém não é o líder de mercado apesar de ganhar força nos últimos tempos.

Existem diversos servidores de aplicação famosos compatíveis com a especificação do J2EE 1.4, Java EE 5 e alguns já do Java EE 6. O JBoss é um dos líderes do mercado e tem a vantagem de ser gratuito e open source. Alguns softwares implementam apenas uma parte dessas especificações do Java EE, como o Apache Tomcat, que só implementa JSP e Servlets (como dissemos, duas das principais especificações), portanto não é totalmente correto chamá-lo de servidor de aplicação. A partir do Java EE 6, existe o termo “application server web profile”, para poder se referencia a servidores que não oferecem tudo, mas um grupo menor de especificações, consideradas essenciais para o desenvolvimento web.

Arquitetura Web

Infraestrutura: Hardware, Serviços e Protocolos.



Protocolo IP: é um protocolo de comunicação usado entre duas ou mais máquinas em rede para encaminhamento dos dados. Ex: 74.125.234.104

Protocolo HTTP: Hipertext Transfer Protocol – Recupera recursos referenciados por uma URL

URL: Uniform Resource Locator – É um endereço de um recurso disponível na rede.

Serviço DNS: Domain Name Service – Traduz nomes para números IP – Ex: 74.125.234.104 / www.google.com

Software: Navegador, Servidores e Linguagens.



Vantagens

- Acesso universal navegador web
- Acesso de qualquer local
- Não possui instalação de aplicativos cliente

Tecnologia JEE

Para suportar a complexidade das regras de negócio, conhecidas como requisitos funcionais e não funcionais que estão diretamente ligados à infraestrutura, como: persistência em banco de dados, transação, acesso remoto, web services, gerenciamento de threads, gerenciamento de conexões HTTP, cache de objetos, gerenciamento da sessão web, balanceamento de carga, entre outros a Sun desenvolveu uma série de especificações que quando implementadas, serão utilizadas como ferramentas para os desenvolvedores.

Esta infraestrutura poderia ser implementada por qualquer pessoa, trazendo uma gama de implementações open source ou pagas, de boa qualidade.

JEE não é um software em sua essência, e sim um conjunto de especificações de como este software deve se comportar permitindo esta distribuição por parte de distribuidores distintos, graças a uma padronização e o detalhamento desta "receita". Caso seja necessária a troca desta implementação, o sistema que a utiliza, não será alterado. Mudando apenas a parte da implementação desta especificação o sistema não ficará preso a um código de um fornecedor específico, evitando o que é chamado vendor lock-in.

O nome J2EE era usado nas versões mais antigas, até a 1.4. Atualmente, o nome correto é Java EE, por uma questão de marketing, podendo ser encontradas muitas referências ao antigo termo J2EE.

Componentes JEE (Abordados no curso)

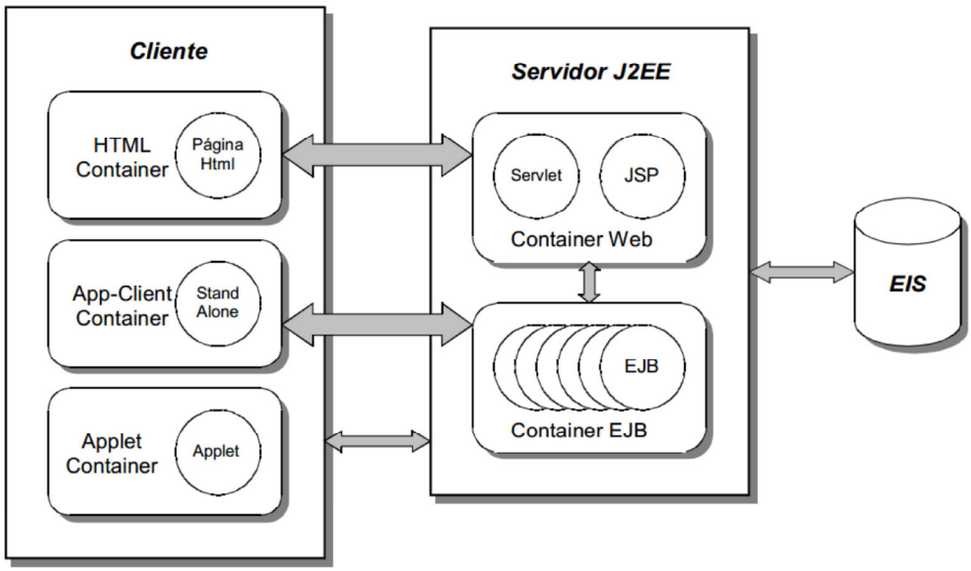
- Páginas JSP
- Servlets
- EJB´s

JSP e Servlets são sem dúvida as especificações essenciais que todo desenvolvedor Java vai precisar para desenvolver com a Web. Mesmo usando frameworks e bibliotecas que facilitam o trabalho para a Web, conhecer bem essas especificações é certamente um diferencial, e fará com que você entenda motivações e dificuldades, auxiliando na tomada de decisões arquiteturais e de design.

Vantagens

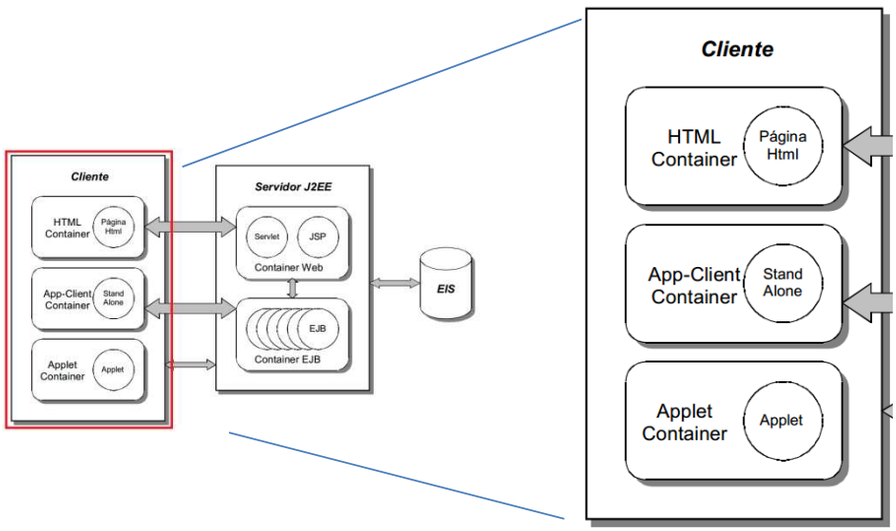
- O modelo de aplicações JEE define uma arquitetura para implementação de serviços como aplicações multicamadas provendo escalabilidade, acessibilidade e manutenibilidade.
- A infraestrutura oferecida pela JEE possibilita que estas camadas, possivelmente localizadas em máquinas diferentes, possam se comunicar remotamente e juntas comporem uma aplicação.

Multicamadas



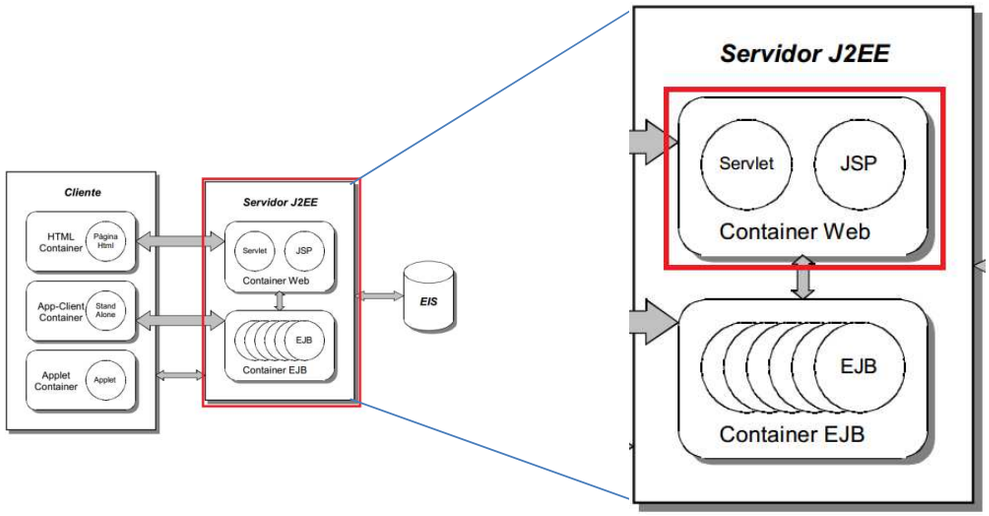
Camada Cliente

Acesso por interfaces stand-alone (aplicações java), páginas HTML ou Applets



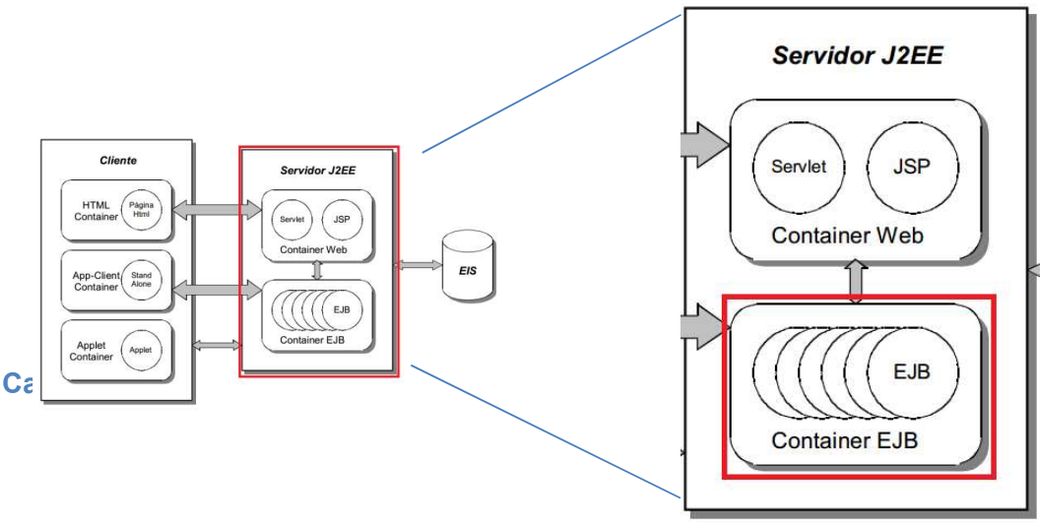
Camada Web

Esta camada é implementada por JSPs e Servlets, que fornecem a lógica para a camada cliente. Em destaque pelo foco dado no decorrer do curso.

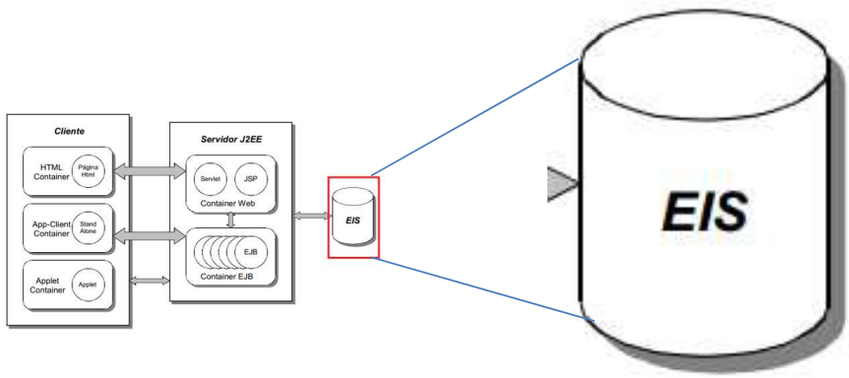


Camada de Negócios

Esta camada trata da lógica de negócio da aplicação providos por componentes conhecidos por EJB´s.



Enterprise Information System, ou Sistema de informações empresariais. Nesta camada é que se encontram os sistemas de banco de dados, sistemas legados, integração com outros sistemas não JEE.



Containers

Um container é um ambiente de execução padronizado que provê serviços específicos a um componente. Os componentes deverão estar contidos em um container apropriado. Com a padronização os componentes JEE podem esperar que, independente de fornecedor, os serviços estarão disponíveis.

O desenvolvedor não precisa se preocupar, por exemplo, com o ciclo de vida, o controle de threads, a persistência, o controle de sessões etc. Todos estes serviços são gerenciados pelo container.

Basicamente existem os containers web, que gerenciam servlets e JSPs, e os containers EJB, que gerenciam os EJBs.





O container utilizado durante o curso será um Servlet Container. Este servidor suporta as funcionalidades abordadas, mas não necessariamente o Java EE completo. Recomendado para quem não precisa de tudo do Java EE como, por exemplo, os EJB's. A maioria das aplicações de pequeno e médio porte irá requerer apenas a parte web. Utilizaremos o mais famoso dos Servlet Container, o Apache Tomcat.

O container permite que o desenvolvedor foque na lógica de negócios e não se preocupe com implementações de segurança, threads e rede. Com uma maneira mais simples do que a construção de um ServerSocket para comunicação entre servidor e aplicação, o Servlet Container possui o conhecimento do protocolo para trafegar as informações.

Entre os serviços oferecidos por ele, está o gerenciamento do ciclo de vida dos Servlets. O container instancia, inicializa e destrói os Servlets de acordo com as necessidades de sua aplicação. E ainda a tradução das paginas JSP, que serão detalhadas posteriormente.

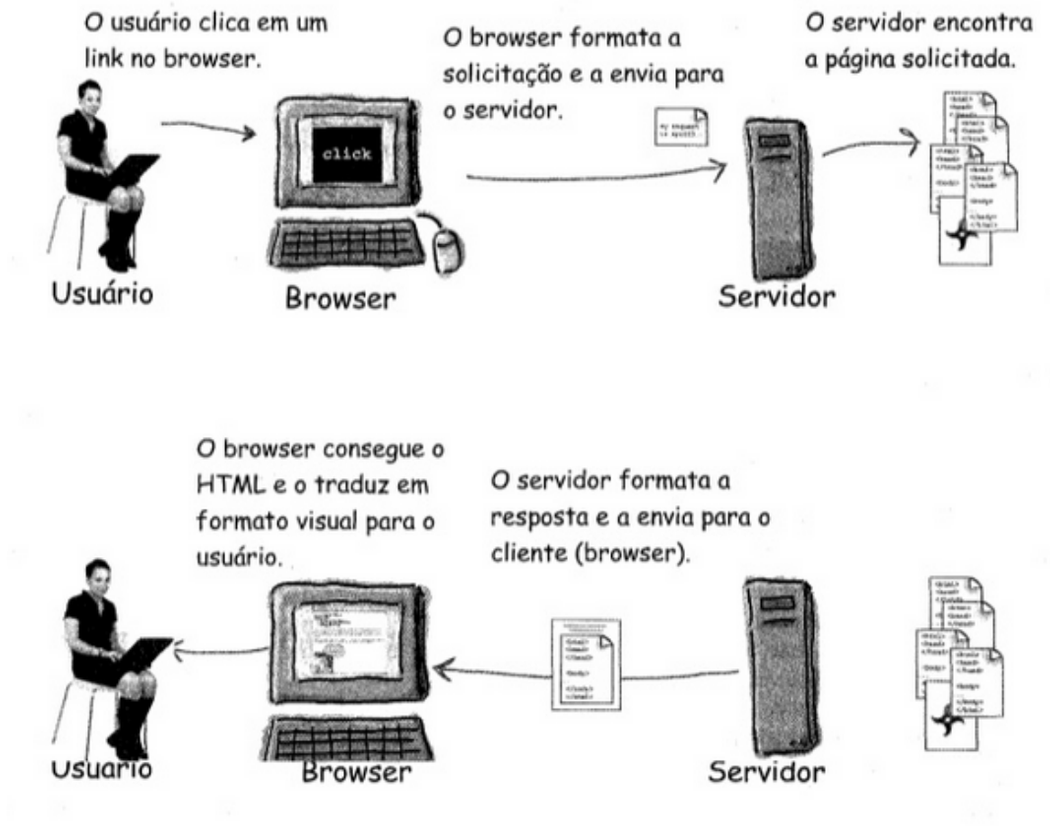
Web Tier – A camada web

Camada de apresentação responsável por entregar os recursos demandados pelo browser.

Quais recursos?

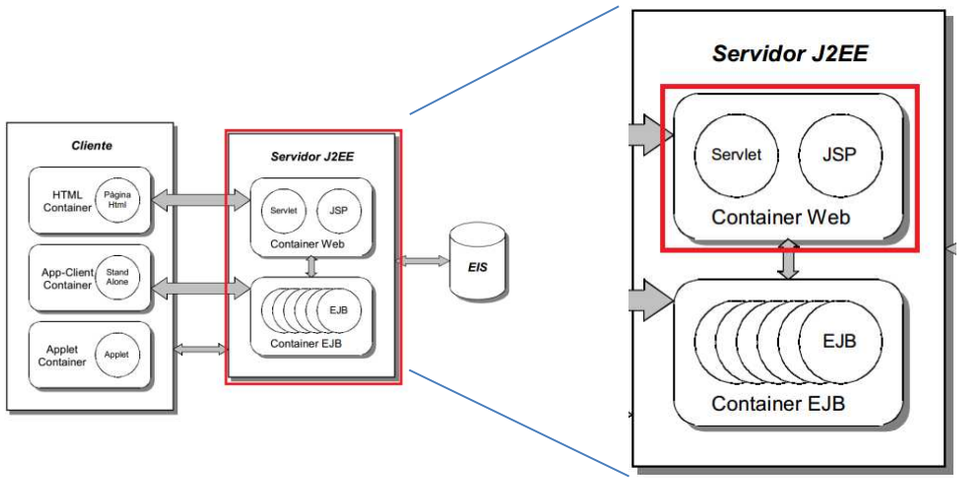
- Paginas HTML´s
- Imagens (JPEG, PNG, etc.).

- PDF´s
- Áudios
- Vídeos
- Etc...

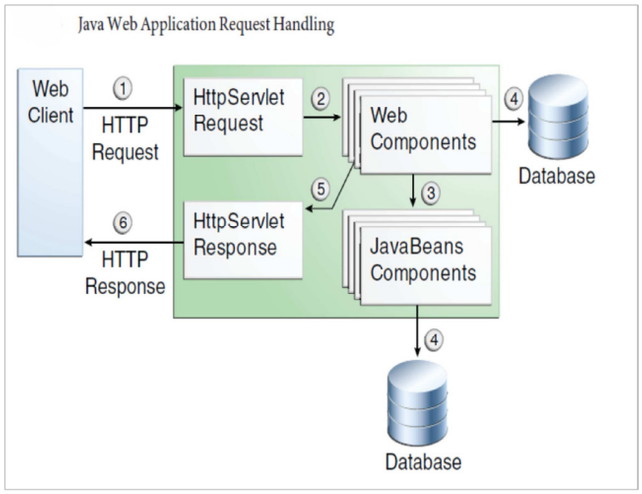


Este recurso é recuperado pelo Servlet Container disposto em um servidor web. O Servlet Container executa o Servlet correspondente, que responde os HTTP requests com um HTTP response apropriado. Cenas para os próximos capítulos...

Uma aplicação Web é uma extensão dinâmica de um servidor web/aplicação. Na plataforma Java EE, os componentes da Web fornecem as capacidades de extensão dinâmica de um servidor web.



O curso aborda a camada web e os componentes JEE: Servlet e JSP que serão detalhados no decorrer deste material. Os recursos dinâmicos solicitados à camada web recebem outro tipo de tratamento pelo Servlet Container. Os servlets não possuem um método main(), Eles estão sob o controle de outra aplicação Java chamada container, quando a aplicação web recebe uma requisição para um servlet, o servidor entrega a requisição para o Container no qual o Servlet é distribuído. É ele que entrega ao Servlet também, o request e o response HTTP.



A figura representa a interação entre cliente web e uma aplicação web

1 - O Cliente web envia um HTTP request para servidor web;

2 - O servidor web converte a HTTP request para um objeto HttpServletRequest;

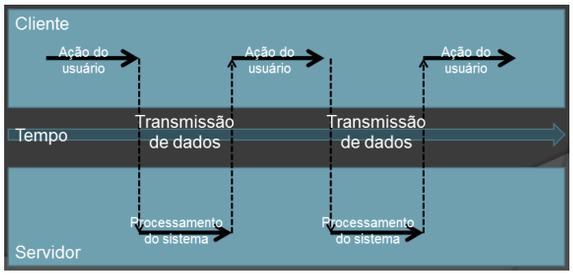
3 - O objeto é enviado para web componente que interage JavaBeans Componete ou Database;

5 - Web Componete gera um objeto HttpServletResponse;

6 - O servidor web converte HttpServletResponse para um objeto HTTP Response e envia para o cliente web.

Ciclo de vida Web App

No ciclo clássico a iteração é síncrona, caracterizada por sequências uniformes de transações entre cliente e servidor.



Existe o modelo assíncrono, onde a interface com o usuário não age diretamente com o lado servidor, permitindo a independência desta camada.

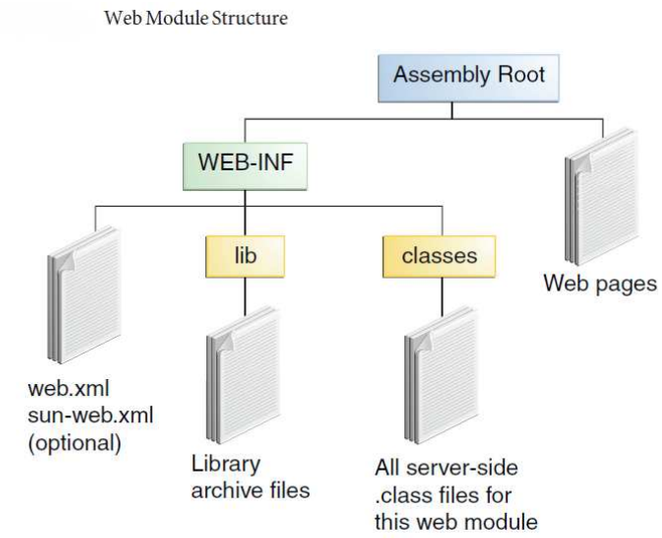
Não faz parte do cronograma este modelo, portanto, para maiores informações vide AJAX.

Web Module – Estrutura

Todos os componentes de uma aplicação são conhecidos como Web Resources.

A menor unidade utilizável de Web Resources é um Web Module, também chamada de aplicação Web.

Se um Web Module possuí apenas JSP´s e conteúdos estáticos, não é necessário a inclusão do arquivo web. xml



WAR

Um web module pode ser implantado como uma estrutura de um arquivo descompactado, ou como um arquivo semelhante a um arquivo JAR, conhecido como WAR (Web Archive).

Isto torna a aplicação portátil, permitindo a sua instalação em qualquer web container padronizado pela especificação JEE.

Ambiente de desenvolvimento web

Componentes

- Linguagem de programação
    - Java JDK 6.0
    - Surgiu aproximadamente em 1995;
    - Sun Microsystem, que foi adquirida pela oracle;
    - Possibilitava a interação com sites da Web, utilizando applets;
- Servidor de aplicações / Web Container
    - Tomcat 6.0– Catalina
    - Início por volta de 1999 na fusão de dois projetos
    - Java Web Server (Sunmicrosystem);
    - Jserv (ASF - Apache Software Foundation);
    - Implementa os serviços de Web/HTTP e Servlet;
    - Servidor Web;
    - Servidor Servlet.
- IDE de desenvolvimento
    - Eclipse
    - Desenvolvida inicialmente pela IBM;
    - Contém uma variedade de plugins específicos;
    - É considerado o principal IDE para desenvolvimento JAVA.

Configuração

- Java JDK 6.0
    - Variáveis de ambiente
        - JAVA_HOME;
        - PATH;
- Tomcat
    - Variáveis de ambiente
        - CATALINA_HOME;
        - PATH;

- IDE Eclipse
    - Servidor Servlet
        - Container Apache Tomcat

Exercício – Revisão HTML – Valor (3 Pontos)

1. Qual a sua definição?
2. Existe algum padrão de construção? Quem define estes padrões?
3. Cite 5 tags HTML e suas definições
4. Por que é importante a utilização da tag “div”?
5. O que é CSS? Cite uma vantagem na sua utilização.
6. Criar uma página HTML contendo os campos Login e Senha e um botão Logar.
7. Criar uma pagina HTML contendo uma tabela com os dados a seguir:

Cod	Descrição	Categoria	Preço	QTD Estoque
1	Teclado	HW	R$ 19,90	10
2	Mouse	HW	R$ 9,90	15
3	Anti Virus X	SW	R$ 199,90	-
4	Placa Mãe	HW	R$ 500,00	5

Protocolo HTTP

É um protocolo de comunicação, e significa em português Protocolo de Transferência de Hipertexto, presente no topo do TCP/IP (Transporte). O TCP é o responsável por garantir que um arquivo enviado de um nó da rede para o outro chegue íntegro ao seu destino, ainda que o arquivo esteja dividido em blocos no momento do envio.

A maioria das conversas que ocorre na web entre clientes e servidores é mantida através do protocolo HTTP, que possibilita conversas de request e resposta simples. A estrutura do protocolo é simples, baseada em

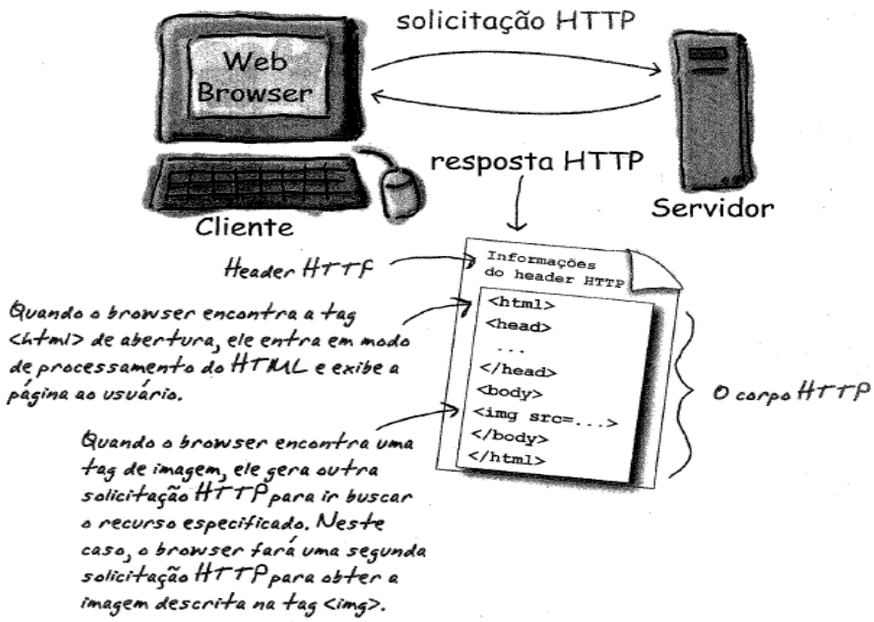
pedidos/respostas (Request/Response). O cliente envia uma solicitação HTTP e o servidor retorna uma resposta HTTP.

O que compõe uma requisição (Request)?

- Um método HTTP (responsável pela ação)
- O endereço do recurso (URL)
- Parâmetros de um formulário

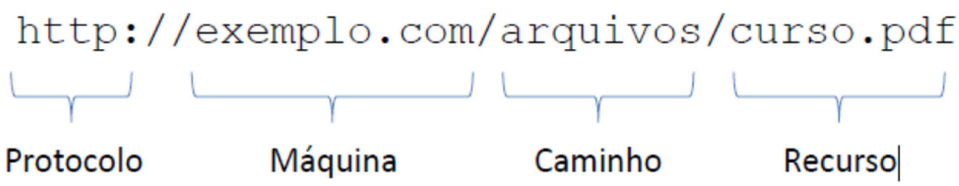
O que compõe uma resposta (Response)?

- Um código de retorno:
  - status code, sucesso, erro, etc.
- Tipo do conteúdo:
  - content-type,
  - text/plain, text/html,
  - Image/jpeg, application/zip, etc.
- Um conteúdo:
  - HTML, imagem, áudio, vídeo, arquivos, etc.



O que é URL?

URL é abreviação de Uniform Resource Locator (Localizador Padrão de Recursos, em português), e é o caminho de um recurso em uma rede. Todo recurso na web tem seu endereço único próprio, no padrão URL.



Protocolo:

- Informa ao servidor qual protocolo de comunicação que será utilizado.

Máquina ou Servidor:

- O nome único do servidor físico pelo qual você está procurando. Este nome aponta para um único endereço IP. Os endereços IP são numéricos e assumem a forma “xxx.yyy.zzz.aaa”. Neste exemplo utilizamos um nome (DNS) por ser mais fácil de memorizar.

Caminho:

- O caminho para a localização no servidor, do recurso que esta sendo solicitado.

Recurso:

- O nome do conteúdo que está sendo solicitado. Pode ser uma página HTML, um Servlet ou outro tipo de arquivo, conforme vimos anteriormente. Se esta parte for omitida na URL, o servidor irá buscar a página padrão definida na aplicação ou web site.

Métodos

O protocolo HTTP possui 8 métodos. Um método determina o que o servidor deve fazer com o URL informado na requisição. GET e POST são os métodos mais conhecidos, e estes serão os abordados no curso.

Além do GET e do POST, o protocolo HTTP possui ainda mais 6 métodos: PUT, DELETE, HEAD, TRACE, CONNECT e OPTIONS. Muitas pessoas conhecem apenas o GET e POST, pois, são os únicos que HTML 4 suporta.

Em uma requisição HTTP, a primeira coisa encontrada será o nome do método. Estes não são métodos Java, mas a ideia é semelhante. O nome do método informa ao servidor o tipo de solicitação que está sendo feita e como o resto da mensagem será formatada.

GET

É o método HTTP mais simples e utilizado para obtermos os recursos de um servidor. Requisições diretas são realizadas por meio deste método.

Exemplo, ao digitar o endereço:

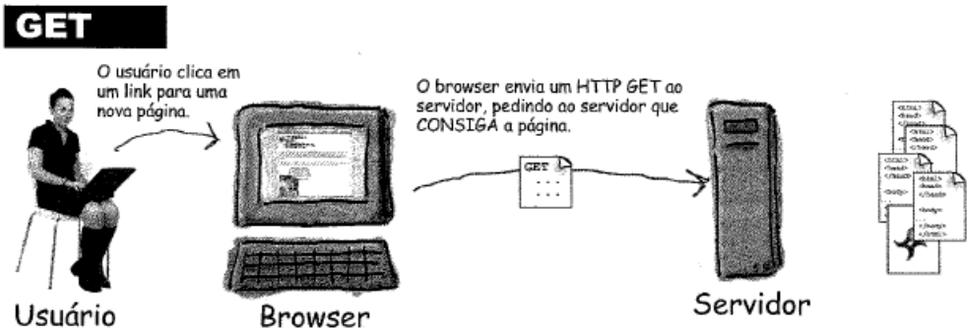
- http://www.exemplo.com/curso.jsp?nome=javaweb

O browser envia a seguinte requisição (resquest)...

```
GET /curso.jsp?nome=javaweb HTTP/1.1
Host: www.exemplo.com
Accept-Encoding: gzip,deflate
```

...e recebe como resposta do servidor (response):

```
HTTP/1.1 200 OK
Date: Mon, 23 May 2005 22:38:34 GMT
Server: Apache/1.3.27 (Unix) (Red-Hat/Linux)
Last-Modified: Wed, 08 Jan 2003 23:11:55 GMT
Etag: "3f80f-1b6-3e1cb03b"
Accept-Ranges: bytes
Content-Length: 438
Connection: close
Content-Type: text/html; charset=UTF-8
```



POST

Método semelhante ao GET, mas com a diferença que este encapsula os parâmetros.

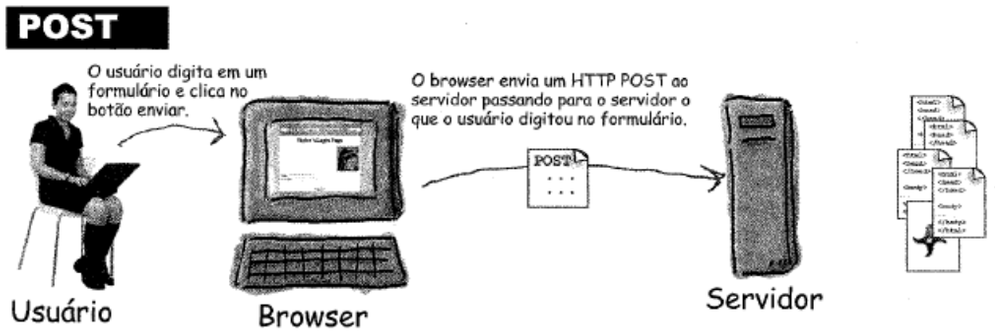
Usado fundamentalmente em formulários.

```
<html>
<head>
<title>Formulário de teste</title>
</head>
<body>
<form name="contato" action="enviaContato" method="post" >
      Nome: <input type="text" name="nome" /><br/>
      Mensagem: <input type="text" name="mensagem" /><br/>
      <input type="submit" value="Enviar" />
</form>
</body>
</html>
```

O request do exemplo anterior (URL GET) via método POST:

```
POST /curso.jsp HTTP/1.1
Host: www.exemplo.com
Accept-Encoding: gzip,deflate

nome=javaweb
```



GET x POST

Ambos podem enviar e receber parâmetros, porem o método GET possui um número limitado de caracteres a ser colocados no Request. No GET, os parâmetros são adicionados à URL, no POST, ao corpo da requisição. Isto permite que via GET seja feita a adição de bookmark (Favoritos) da pagina requisitada.

Convenção: GET = getting / POST = update

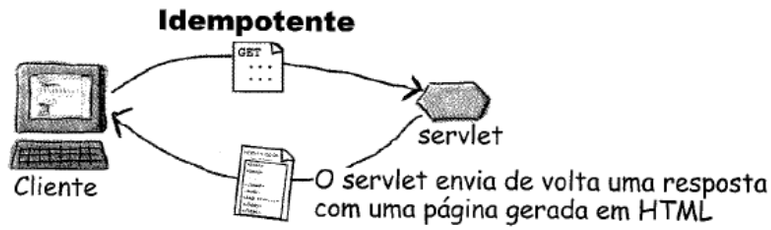
O total de caracteres no GET é realmente limitado (dependendo do servidor). Se o usuário digitar, um trecho extenso em uma caixa de entrada do tipo "Procurar", o GET pode não funcionar.

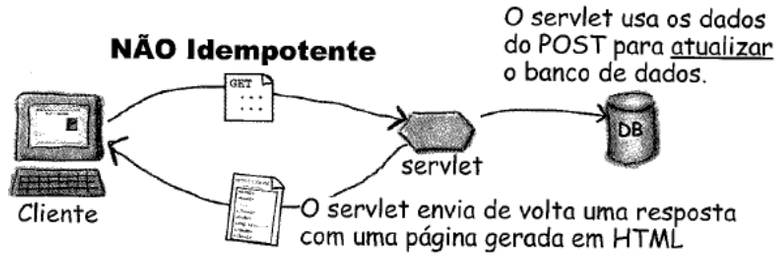
Os dados que você envia através do GET são anexados à URL, lá em cima, na barra de endereços do browser; portanto, seja o que for, ficará exposto. Melhor não colocar a senha ou outra informação sensível como parte de um GET.

O HTTP GET é usado apenas para conseguir coisas e não deve mudar nada no servidor. Então um GET é, por definição (e de acordo com a especificação HTTP), idempotente. Ele pode ser executado mais de uma vez, sem causar qualquer efeito colateral danoso.

O POST não é idempotente. Os dados submetidos no corpo de um POST podem ser destinados a uma transação que não pode ser desfeita. Portanto, tenha cuidado com a funcionalidade do seu doPost().

Cuidado! O conceito de idempotência é referente à especificação HTTP dos métodos GET e POST, não significa que os métodos doGet () e doPost() dos servlet's seguem este conceito, isto depende de sua implementação.





Exercícios:

1 - GET ou POST? Preencha as lacunas com o nome do método correto.

_____ Um usuário digitando um login e uma senha.

_____ Um usuário solicitando uma nova página via hyperlink.

_____ Um usuário em uma sala de bate-papo enviando uma resposta.

_____ Um usuário dica no botão "next" para ver a próxima página.

_____ Um usuário dica no botão de "logout" num site seguro de um banco.

_____ Um usuário dica em "Voltar" no browser.

_____ Um usuário envia um formulário com nome e endereço para o servidor.

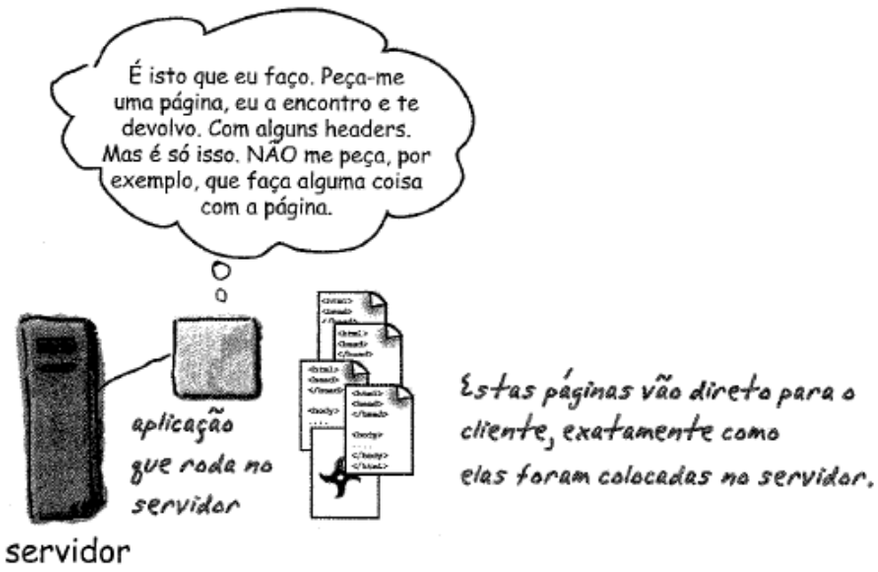
_____ Um usuário faz uma escolha em um botão de seleção.

2 – Crie o formulário abaixo, alterne entre os métodos GET e POST e descreva as diferenças.

```
<html>
<head>
<title>Formulário de teste</title>
</head>
<body>
<form name="contato" action="enviaContato" method="post" >
      Nome: <input type="text" name="nome" /><br/>
      Mensagem: <input type="text" name="mensagem" /><br/>
      <input type="submit" value="Enviar" />
</form>
</body>
</html>
```

HTML e Servlet

Páginas HTML são estáticas (o seu conteúdo é estático, não muda) e os servidores web adoram páginas estáticas!



Duas coisas que o servidor não faz sozinho:

1 – Conteúdo dinâmico:

A aplicação que roda no servidor disponibiliza somente páginas estáticas, porém outra aplicação "assistente", com a qual o servidor pode se comunicar, pode construir páginas dinâmicas instantaneamente.

2 – Salvando dados no servidor:

Quando o usuário envia os dados em um formulário, o servidor analisa os dados e pensa: "E aí? Eu devo me preocupar?". Para processar esses dados, salvá-los em um arquivo ou banco de dados, ou até mesmo para usá-los na criação da página de resposta, você precisa de outra aplicação.

Quando a Web surgiu, seu objetivo era a troca de conteúdos através, principalmente, de páginas HTML estáticas. Eram arquivos escritos no formato HTML e disponibilizados em servidores para serem acessados nos

navegadores. Imagens, animações e outros conteúdos também eram disponibilizados.

Mas logo se viu que a Web tinha um enorme potencial de comunicação e interação além da exibição de simples conteúdos. Para atingir esse novo objetivo, porém, páginas estáticas não seriam suficientes. Era preciso servir páginas HTML geradas dinamicamente baseadas nas requisições dos usuários.

Hoje, boa parte do que se acessa na Web (portais, blogs, home bankings etc.) é baseado em conteúdo dinâmico. O usuário requisita algo ao servidor que, por sua vez, processa essa requisição e devolve uma resposta nova para o usuário.

Uma das primeiras ideias para esses “geradores dinâmicos” de páginas HTML foi fazer o servidor Web invocar um outro programa externo em cada requisição para gerar o HTML de resposta. Era o famoso CGI que permitia escrever pequenos programas para apresentar páginas dinâmicas usando, por exemplo, Perl, PHP, ASP e até C ou C++.

Servlet

Na plataforma Java, a primeira e principal tecnologia capaz de gerar páginas dinâmicas são os Servlets, que surgiram no ano de 1997. Hoje, a versão mais encontrada no mercado é baseada nas versões 2.x, mais especificamente a 2.4 (parte do J2EE 1.4) e a 2.5 (parte do Java EE 5). A última versão disponível é a versão 3.0 lançada em Dezembro de 2009 com o Java EE 6, mas que ainda não tem adoção no mercado.

A função de um servlet é receber uma solicitação do cliente e devolver uma resposta. A solicitação talvez seja simples: "traga-me a página de Boas-vindas". Ou pode ser complexa: "Finalize o processo do meu carrinho de compras." A solicitação traz consigo dados cruciais e o código do seu servlet tem que saber como encontrá-las e utilizá-las.

A resposta leva a informação que o browser precisa para montar uma página (ou baixar alguns dados) e o código do seu servlet tem que saber como enviá-las. Ou não... Em vez disso, seu servlet pode decidir encaminhar a solicitação adiante (para outra página, servlet ou JSP). Com esta função de receber chamadas HTTP, processá-las e devolver uma resposta ao cliente, podemos traduzir o nome Servlet como “Servidorzinho”.

O comportamento dos servlets que vamos ver neste capítulo foi definido na classe HttpServlet do pacote javax. servlet.

A interface Servlet é a que define exatamente como um servlet funciona, mas não é o que vamos utilizar, uma vez que ela possibilita o uso de qualquer protocolo baseado em requisições e respostas, e não especificamente HTTP.

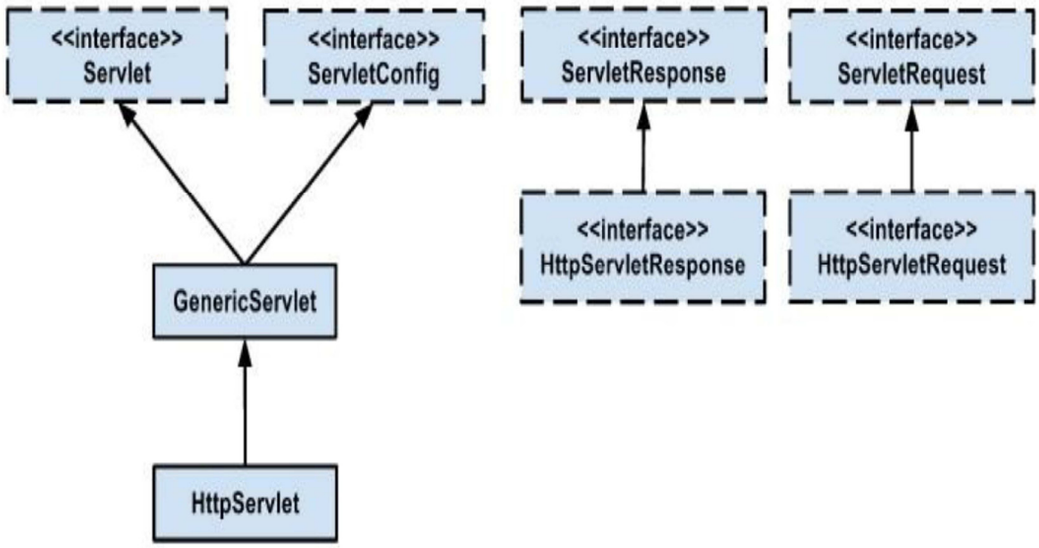
Para escrevermos um servlet, criamos uma classe Java que estenda HttpServlet e sobrescreva um método chamado service. Esse método será o responsável por atender requisições e gerar as respostas adequadas. Sua assinatura:

```
protected void service (HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
    ...
}
```

Repare que o método recebe dois objetos que representam, respectivamente, a requisição feita pelo usuário e a resposta que será exibida no final. Veremos que podemos usar esses objetos para obter informações sobre a requisição e para construir a resposta final para o usuário.

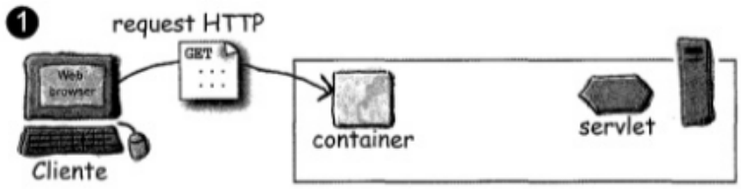
Usando a interface HttpServletRequest, escrever o código que retira da solicitação os parâmetros do formulário HTML, a informação do header da solicitação HTTP. Usando a interface HttpServletResponse, escrever o código que cria um header para a resposta HTTP, configura o tipo de conteúdo da resposta, recebe um stream de texto para a resposta, recebe um stream binário para a resposta, redireciona uma solicitação HTTP para outra URL.

Estrutura de classes

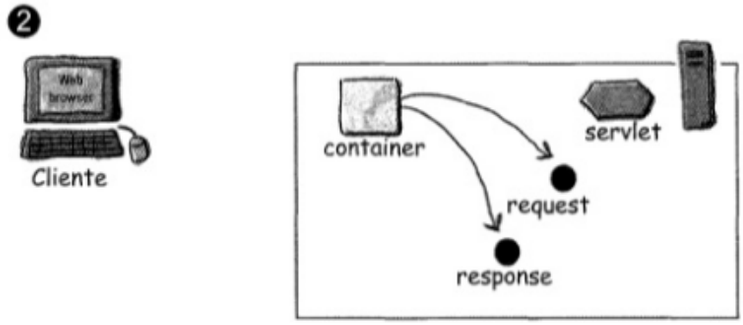


Os Servlets são controlados pelo Container

O Servlet Container cria os objetos request e response, cria ou aloca uma nova thread para o servlet e chama o método serviceO do servlet, passando as referências de request e response como argumentos.

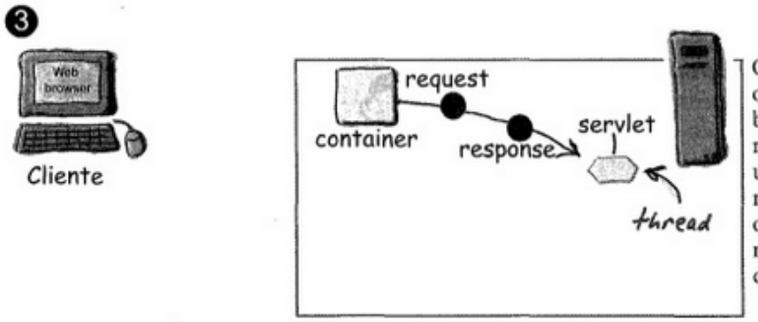


O usuário clica em um link que contém uma URL para um servlet, em vez de uma página estática.

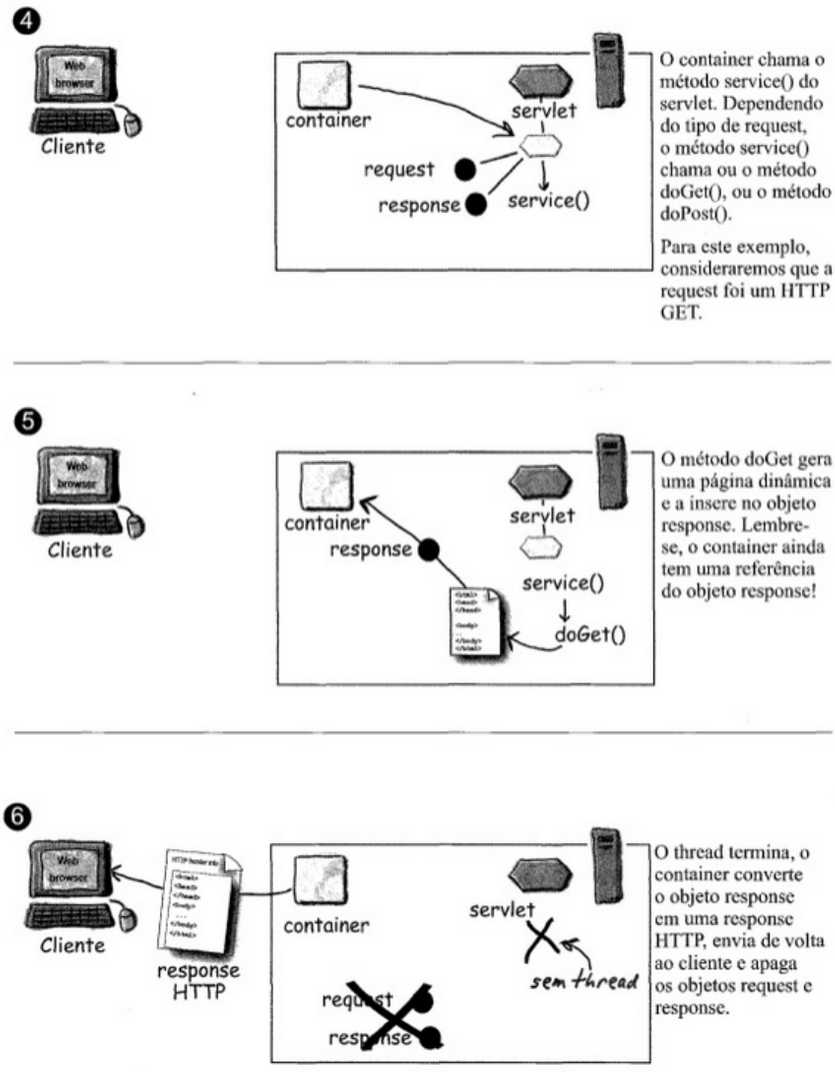


O container "vê" que a request é para um servlet e então ele cria dois objetos:

1) HttpServletResponse
2) HttpServletRequest



O container encontra o servlet correto baseado na URL da request, cria ou aloca uma thread para essa request, e passa os objetos request e response para a thread do servlet.



A solicitação do cliente sempre inclui um Método HTTP específico. Se o Método HTTP for um

GET, o método service () chama o doGet (), Se for um POST, o método service () chama o doPost ().

O método service aceita todos os métodos HTTP, portanto, tanto o método GET quanto o POST. Para especificarmos como trataremos cada método, temos que escrever os métodos doGet e/ou doPost no nosso servlet:

void do Get(HttpServletRequest req, HttpServletResponse res);

void doPost(HttpServletRequest req, HttpServletResponse res);

Ciclo de Vida de um Servlet

O ciclo de vida de um Servlet é determinado pela execução dos seguintes métodos:

- init (): É executado somente uma única vez após a criação do Servlet. É utilizado para iniciar o Servlet e criar suas configurações;
- service (): É executado sempre antes do Servlet responder a uma requisição. É sucedido por um dos métodos doGet() / doPost();
- Destroy(): É executado pelo container no seu fechamento ou quando um Servlet é reiniciado.

Todo servlet deve possuir um construtor sem argumentos para que o container possa criá-la. Após a criação, o servlet container inicializa a servlet com o método init(ServletConfig config) e a usa durante todo o seu período ativo, até que vai desativá-la através do método destroy(), para então liberar o objeto.

É importante perceber que a sua Servlet será instanciada uma única vez pelo container e esse único objeto será usado para atender a todas as requisições de todos os clientes em threads separadas. Aliás, é justo isso que traz uma melhoria em relação aos CGI comuns que disparavam diversos processos.

Na inicialização de uma servlet, quando parâmetros podem ser lidos e variáveis comuns a todas as requisições devem ser inicializadas. É um bom momento, por exemplo, para carregar arquivos diversos de configurações da aplicação:

void init (ServletConfig config);

Na finalização, devemos liberar possíveis recursos que estejamos segurando:

void destroy();

Os métodos init e destroy, quando reescritos, são obrigados a chamar o super. init() e super.destroy() respectivamente. Isso acontece, pois um método é diferente de um construtor. Quando estendemos uma classe e criamos o nosso próprio construtor da classe filha, ela chama o construtor da classe pai sem argumentos, preservando a garantia da chamada de um construtor. O mesmo não acontece com os métodos.

Web.xml

Definido o Servlet, surgem alguns questionamentos: Como vamos acessá-la pelo navegador? Qual o endereço podemos acessar para fazermos com que ela execute? O container não tem como saber essas informações, a não ser que digamos isso para ele. Para isso, vamos fazer um mapeamento de uma URL específica para um servlet através do arquivo web. xml, que fica dentro do WEB-INF.

Quando você for distribuir seu servlet em seu Container, você criará um documento XML, razoavelmente simples, chamado Deployment Descriptor (DD), para informar ao Container como executar seus servlets e JSPs. Embora você use o DD para outras tarefas além de simples mapeamento de nomes, você usará dois elementos XML para mapear as URLs aos servlets - um para mapear o nome público de URL conhecido pelo cliente para o seu próprio nome interno, e o outro para mapear seu próprio nome interno para um nome de classe completamente qualificado.



ServletConfig

- Um objeto ServletConfig por servlet
- Use-o para passar informações de tempo de distribuição para o servlet (um banco de dados ou a pesquisa do nome de um enterprise bean, por exemplo) que você não queira fazer hardcode no servlet (parâmetros init do servlet).
- Use-o para acessar o ServletContext.
- Os parâmetros são configurados no Deployment Descriptor.

ServletContext

- Um ServletContext por aplicação. (Eles deveriam tê-lo chamado de AppContext.).
- Use-o para acessar parâmetros da aplicação (também configurado no Deployment Descriptor).
- Use-o como se fosse um quadro de avisos da aplicação, onde você pode escrever mensagens (conhecidas como atributos) que as outras

partes da aplicação possam acessar (mais sobre isto no próximo capítulo).

- Use-o para obter informações do servidor, incluindo o nome e a versão do Container e a versão da API que é suportada.

Alguns métodos da interface HttpServletRequest

- String [] getParameterValues (String s):
  - Retorna múltiplos valores de um parâmetro que possui mais de um valor;
- Enumeration getParameterNames ():
  - Retorna os nomes de todos os parâmetros enviados;
- String getHeader (String s):
  - Retorna o valor do parâmetro do cabeçalho enviado;
- Enumeration getHeaders (String s):
  - Retorna todos os valores do cabeçalho;
- Enumeration getHeaderNames ():
  - Retorna todos os nomes de cabeçalho enviado

Parâmetros da requisição

Parâmetros são pares nome=valor que são enviados pelo cliente concatenados em strings separados por & e podem ser passados na requisição de duas formas:

- Se o método for GET, os parâmetros são passados em uma única linha na query string, que estende a URL após um"?":

  GET /servlet/Teste?id=agente007&acesso=3 HTTP/1.0

- Se o método for POST, os parâmetros são passados como um stream no corpo na mensagem (o cabeçalho Content-length, presente em requisições POST informa o tamanho):

  POST /servlet/Teste HTTP/1.0 Content-length: 21 Content-type: x-www-form-urlencoded id=agente007&acesso=3

Como ler os parâmetros da requisição

Seja o método POST ou GET, os valores dos parâmetros, podem ser recuperados pelo método getParameter () de ServletRequest, que recebe seu nome. Exemplo:

String parâmetro = request. getParameter("nome");

Parâmetros de mesmo nome podem ser repetidos. Neste caso getParameter() retornará apenas a primeira ocorrência. Para obter todas use String [] getParameterValues (). Exemplo:

String [] params = request. getParameterValues("nome");

Alguns métodos da interface HttpServletResponse

- addHeader(String nome, String valor):
  - adiciona cabeçalho HTTP
- setContentType (tipo MIME):
  - define o tipo MIME que será usado para gerar a saída (text/html, image/gif, etc.).
- sendRedirect (String location):
  - envia informação de redirecionamento para o cliente (Location: url)
- Writer getWriter ():

- obtém um Writer para gerar a saída. Ideal para saída de texto.

Como gerar uma resposta

Para gerar uma resposta, primeiro é necessário obter, do objeto HttpServletResponse, um fluxo de saída, que pode ser de caracteres (Writer) ou de bytes (OutputStream). Exemplo:

```
Writer out = response.getWriter(); // ou

OutputStream out = response.getOutputStream();
```

Apenas um deve ser usado. Os objetos correspondem ao mesmo stream de dados. Deve-se também definir o tipo de dados a ser gerado. Isto é importante para que o cabeçalho Content-type seja gerado corretamente e o browser saiba exibir as informações

```
response.setContentType ("text/html");
```

Depois, podem-se gerar os dados, imprimindo-os no objeto de saída (out) obtido anteriormente. Muitas vezes precisamos que o nosso servlet redirecione o usuário para uma URL absoluta, ou externa. Nestas situações podemos usar o método sendRedirect () da interface HttpServletResponse. Exemplo:

```
response.sendRedirect (“minhaPagina.html”);

response. sendRedirect(“meuServlet”);
```

Mais sobre os parâmetros da requisição

Dentro do método service do nosso Servlet para adição de contatos, vamos buscar os dados que foram enviados na requisição. Para buscarmos esses dados, precisamos utilizar o parâmetro request do método service

chamando o método getParameter (“nomeDoParametro”), onde o nome do parâmetro é o mesmo nome do input que você quer buscar o valor. Isso vai retornar uma String com o valor do parâmetro. Caso não exista o parâmetro, será retornado null:

String valorDoParametro = request. getParameter("nomeDoParametro");

EXERCÍCIO:

Meu primeiro Servlet

Código do servlet



Para acessar o Servlet acima, iremos criar um formulário HTML:

O HTML do formulário da página inicial

O HTML é simples – ele coloca o cabeçalho, a lista drop-down de onde o usuário seleciona uma cor para a cerveja e o botão Submit.

```
<html><body>
<h1 align="center">Beer Selection Page</h1>
<form method="POST"
  action="SelectBeer.do">
  Select beer characteristics<p>
  Color:
  <select name="color" size="1">
    <option value="light"> light </option>
    <option value="amber"> amber </option>
    <option value="brown"> brown </option>
    <option value="dark"> dark </option>
  </select>
  <br><br>
  <center>
    <input type="SUBMIT">
  </center>
</form></body></html>
```

Por que escolhemos POST em vez de GET?

O HTML pensa que o servlet tem este nome. Não há NADA na sua estrutura de diretórios chamado SelectBeer.do! Isto é um nome lógico...

Foi assim que criamos o menu pull-down, mas suas opções podem variar. (Você entendeu o size="1"?)

Faça o mapeamento adequado para o Servlet e teste sua aplicação no Tomcat.

JSP – Java Server Pages

Até agora, vimos que podemos escrever conteúdo dinâmico através de Servlets. No entanto, se toda hora criarmos Servlets para fazermos esse trabalho, teremos muitos problemas na manutenção das nossas páginas e também na legibilidade do nosso código, pois sempre aparece código Java misturado com código HTML.

Imagine todo um sistema criado com Servlets fazendo a geração do HTML. Para não termos que criar todos os nossos conteúdos dinâmicos dentro de classes, misturando fortemente HTML com código Java, precisamos usar

uma tecnologia que podemos usar o HTML de forma direta, e que também vá possibilitar a utilização do Java.

Desenvolvido para facilitar a criação de paginas feitas pelos Web Designer;

- Separação de papéis: Programador e Web Designer;

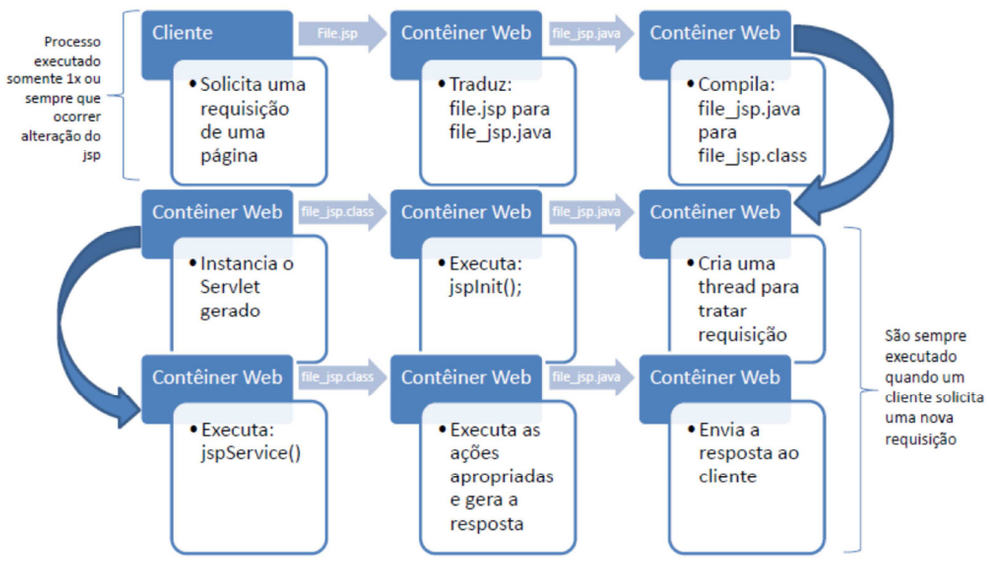
É um documento texto que contém dois tipos de conteúdo:

- Estáticos: HTML e XML;
- Dinâmico: Elementos JSP;

O JSP é baseado em Servlet

O JSP toma-se um servlet completo rodando em sua aplicação. E muito parecido com qualquer outro servlet, exceto pelo fato de que a classe do servlet é escrita para você - pelo Container.

O Container utiliza o que você escreveu no seu JSP, o traduz para um arquivo-fonte da classe servlet (java) e o compila em uma classe servlet Java. Depois disso, ele será só um servlet e rodará exatamente do mesmo jeito, como se você mesmo tivesse escrito e compilado o código. Ou seja, o Container carrega a classe servlet, instancia-a e inicializa-a, cria uma thread para cada solicitação e chama o método service() do servlet.



O contêiner gerencia as duas fases:

- Tradução: o contêiner valida a sintaxe da pagina e determina uma classe para implementá-la;
- Execução: o contêiner gerencia uma ou mais instâncias em função do evento que requisitou;

Elementos JSP

Dentro de um arquivo JSP podemos escrever também código Java, para que possamos adicionar comportamento dinâmico em nossas páginas, como declaração de variáveis, condicionais (if), loops (for, while) entre outros. Para escrever código Java na sua página, basta escrevê-lo entre as tags < % e %>.

Esse tipo de código é chamado de scriptlet:

```
<%
        String mensagem = "Hello World!";
%>
```

Os scriplets tem acesso direto as variáveis do servlet;

final javax. servlet.jsp.PageContext **pageContext**;

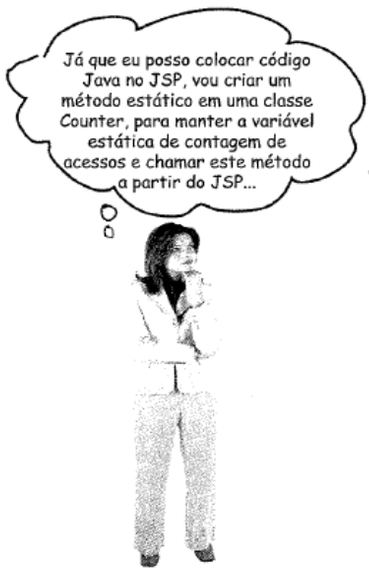
javax.servlet.http.HttpSession **session** = null;

final javax.servlet.ServletContext **application**;

final javax. servlet.ServletConfig **config**;

javax.servlet.jsp.JspWriter **out** = null;

final java.lang.Object **page** = this;



BasicCounter.jsp

```
<html>
<body>
The page count is:
<%
  out.println(Counter.getCount());
%>
</body>
</html>
```

O objeto "out" está implícito lá. Tudo entre <% e %> é um scriptlet, que é apenas Java simples.

Counter.java

```
package foo;

public class Counter {
   private static int count;
   public static synchronized int getCount()
{
       count++;
       return count;
   }
}
```

Classe assistente Java simples.

A classe Counter está no pacote foo, mas não há nada no JSP capaz de reconhecê-la. É idêntico ao que acontece com você em qualquer outro código J ava. E você conhece a regra: importe o pacote ou use o nome da classe totalmente qualificado no seu código.

O código JSP *era*:

```
<% out.println(Counter.getCount()); %>
```

O código JSP *deveria ser*:

```
<% out.println(foo.Counter.getCount()); %>
```

Agora vai funcionar.

A diretiva é um recurso que você tem para dar instruções especiais ao Container no momento da tradução da página. As diretivas vêm em três sabores: page, include e taglib.

Para importar um *único* pacote:

```
<%@ page 'import="foo.*" %>

<html>
<body>
The page count is:
<%
  out.println(Counter.getCount());
%>
</body>
</html>
```

Esta é uma diretiva page com um atributo import. (Observe que não há nenhum ponto-e-vírgula no final da diretiva).

Os scriptlets são Java normais; logo, todas as declarações em um scriptlet devem terminar com um ponto-e-vírgula!

Para importar *múltiplos* pacotes:

```
<%@ page import="foo.*,java.util.*" %>
```

Use uma vírgula para separar os pacotes. A lista inteira de pacotes vem entre aspas!

O código Java vem entre os símbolos <% e %>. Mas a diretiva acrescenta um caractere a mais no começo do elemento - o símbolo @ (arroba) Se você vir um código JSP que começa com <%@, você já sabe que se trata de uma diretiva.

Grande parte da razão do JSP é evitar o println(). É por isso que existe um elemento expressão no JSP - ele exibe automaticamente aquilo que você colocou entre as tags.

O código *scriptlet*:

```
<%@ page import="foo.*" %>
<html>
<body>
The page count is:
<% out.println(Counter.getCount()); %>
</body>
</html>
```

O código *expressão*:

```
<%@ page import="foo.*" %>
<html>
<body>
The page count is now:
<%= Counter.getCount() %>
</body>
</html>
```

A expressão é mais curta – nós não precisamos escrever o print...

O código scriptlet vem entre os sinais de porcentagem <% e %>. Já as expressões recebem um caractere adicional no início do elemento: um sinal de igual (=).

Este JSP:

```
<html><body>
<% int count=0; %>
The page count is now:
<%= ++count %>
</body></html>
```

Torna-se este servlet:

```
public class basicCounter_jsp extends
SomeSpecialHttpServlet {

public void _jspService(HttpServletRequest request,
   HttpServletResponse response)throws java.
   io.IOException,
            ServletException {

   PrintWriter out = response.getWriter();
   response.setContentType("text/html");
   out.write("<html><body>");
   int count=0;
   out.write("The page count is now:");
   out.print( ++count );
   out.write("</body></html>");

   }
}
```

O Container coloca todo o código em um método service genérico. Pense nele como um pacotão doGet/doPost que pega tudo.

TODOS os códigos do scriptlet e das expressões ficam em um método service.

Isto significa que as variáveis declaradas em um scriptlet são sempre variáveis LOCAIS!

Declarar a variável contadora no scriptlet nos mostrou que a variável foi reinicializada cada vez que o método de serviço rodou. Quer dizer que ela foi reiniciada para Oa cada solicitação. De alguma forma, precisamos tomar count uma variável de instância.

```
<%! int count=0; %>
```

Coloque um ponto de exclamação (!) depois do sinal de porcentagem (%).

Isto não é uma expressão - você PRECISA do ponto-e-vírgula aqui!

As declarações JSP servem para declarar membros da classe do servlet gerado. Isto significa as variáveis e os métodos! Portanto, tudo que estiver entre a tag <%! e %> é adicionado à classe fora do método service. Logo, você pode declarar os métodos e as variáveis estáticas e de instância.

A diretiva page

<%@ page import= “foo.*" session= “false" %>

Define propriedades específicas da página, como códigos de caracteres, o tipo de conteúdo para a resposta e se esta página deveria possuir o objeto implícito session. Uma diretiva page pode utilizar até 13 atributos diferentes (como o atributo import), embora apenas quatro sejam cobrados no exame.

A diretiva taglib

<%@ taglib tagdir="/WEB-INF/tags/cool" prefix="cool" %>

Define as bibliotecas de tags disponíveis para o JSP.

A diretiva include

<%@ include file= “wickedHeader.html" %>

Define os textos e os códigos que são acrescentados na página atual no momento da tradução. Isto lhe permite construir pedaços reutilizáveis (como um título-padrão para a página ou a barra de navegação), que podem ser acrescentados a cada página, sem precisar repetir todo aquele código em cada JSP.

Veremos mais sobre JSP em breve...

Sessões e Cookies

Os servidores web não têm memória curta. Assim que eles lhe enviam uma resposta, eles esquecem quem você é. Na próxima vez que você fizer uma solicitação, eles não o reconhecerão. Em outras palavras, eles não se recordam do que você solicitou no passado e nem do que eles enviaram como resposta. Nada. Algumas vezes isso é bom.

Porém, algumas vezes você precisa manter o estado de conversação com o cliente durante várias solicitações. Um carrinho de compras não funcionaria se o usuário tivesse que escolher seus produtos e finalizar a compra de uma única vez. Você vai encontrar uma solução extremamente simples na API Servlet.

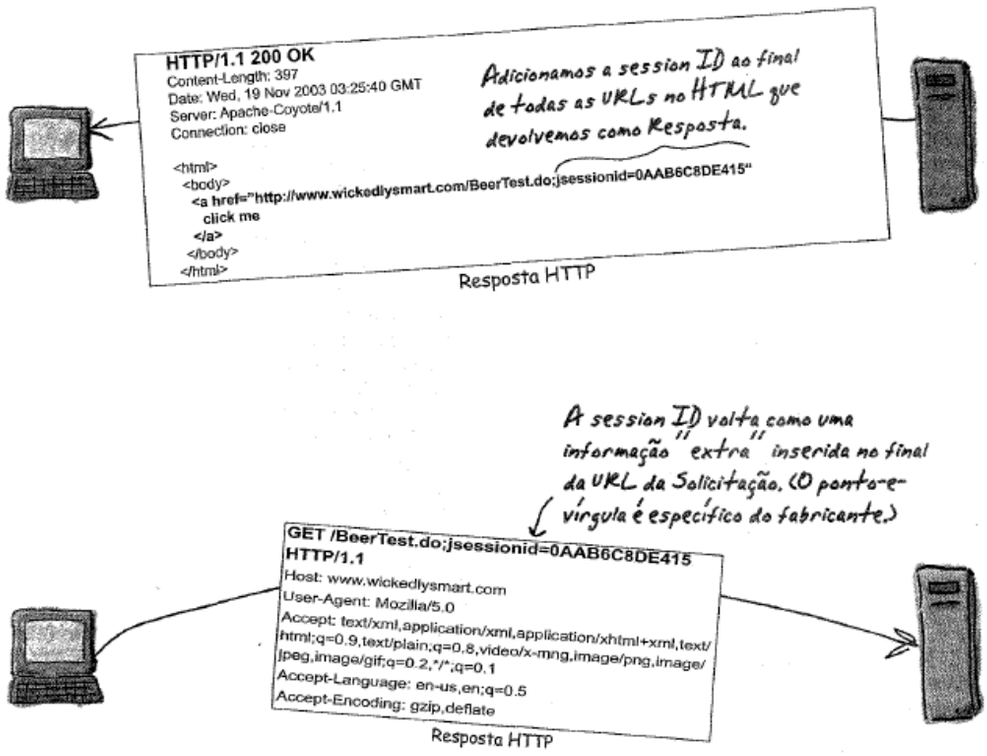
O protocolo HTTP usa conexões stateless. O browser do cliente faz uma conexão para o servidor, envia a solicitação, obtém a resposta e fecha a conexão. Ou seja, a conexão existe apenas para uma única solicitação/resposta.

Devido à conexão não persistir, o Container não reconhece que o cliente que fez a segunda solicitação é o mesmo de uma solicitação anterior. Para o Container, cada solicitação é de um novo cliente.

Técnicas para controle de Sessão

Campo oculto (type=hidden): O campo oculto é enviado junto com a requisição.

Reescrita de URL: A cada conexão é acrescentado um parâmetro com identificação. A reescrita de URL carrega a session ID e a encaixa bem no final de cada URL que a aplicação recebe. O Container simplesmente retira a parte extra da URL e a usa para encontrar a sessão correspondente.



Cookies

Conjunto de pares nome-valor, enviados pelo servidor e armazenado no cliente e retornado ao servidor a cada requisição. Usar Cookies parece facilitar muito a vida, mas através de um cookie não é possível marcar um cliente com um objeto, somente com Strings. Imagine gravar os dados do usuário logado através de cookies. Seria necessário um cookie para cada atributo: usuário, senha, id, data de inscrição, etc. Sem contar a falta de segurança.

O Cookie também pode estar desabilitado no cliente, sendo que não será possível lembrar nada que o usuário fez. O Container que gera a session ID, criando um novo objeto Cookie, inserindo a session ID dentro do cookie e configurando o cookie como parte da resposta. E nas solicitações subsequentes, o Container recebe a session ID de um cookie da solicitação, compara-a com uma sessão existente e associa essa sessão com a solicitação atual.



Tipos de cookies

- Sessão: Permanecem no computador enquanto estiver conectado e interagindo com o servidor web;
- Persistente: São armazenados em um arquivo texto no computador do cliente e podem durar dias, meses ou até anos.

Session

A API Servlet fornece a classe HttpSession encapsulada nas requisições, e são utilizadas para transportar informações entre paginas.

Uma sessão facilita a vida de todos por permitir atrelar objetos de qualquer tipo a um cliente, não sendo limitada somente à strings e é independente de cliente.

A abstração da API facilita o trabalho do programador, pois ele não precisa saber como é que a sessão foi implementada no servlet container, ele simplesmente sabe que a funcionalidade existe e está lá para o uso.

A sessão nada mais é que um tempo que o usuário permanece ativo no sistema. A cada página visitada, o tempo de sessão é zerado. Quando o tempo ultrapassa um limite demarcado no arquivo web.xml, o cliente perde sua sessão.

As sessões são mantidas por um objeto HttpSession. Os métodos getSession () do objeto request é utilizado para acessar um objeto HttpSession.

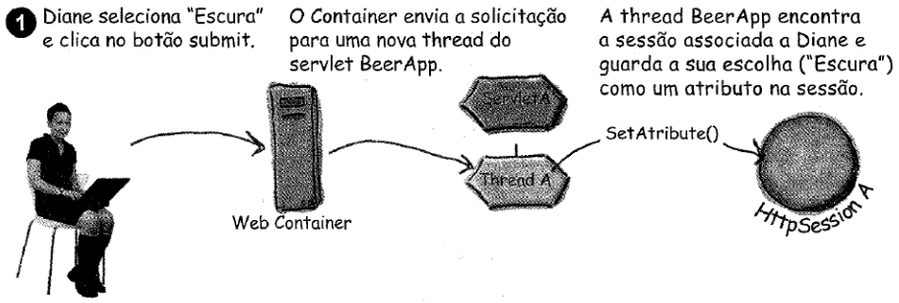
Principais métodos

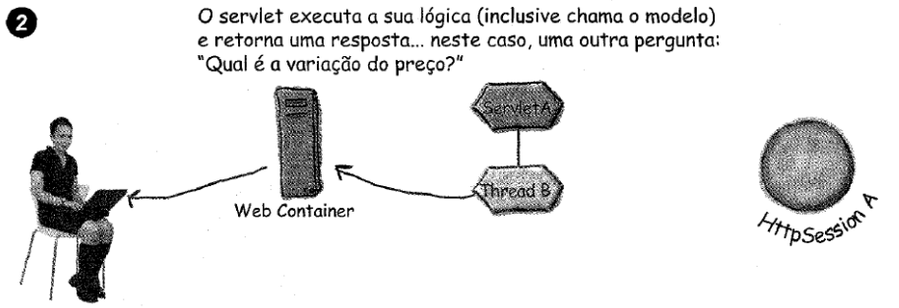
- getCreationTime (): retorna a data de criação da sessão;
- getLastAccessedTime (): retorna a data do ultimo acesso;
- setMaxInactiveIntervals (): seta o valor de inatividade para encerrar a sessão;
- getMaxInactiveIntervals (): retorna o valor de inatividade que encerra a sessão;
- invalidate (): encerra uma sessão ativa explicitamente;
- isNew (): verifica se é uma nova sessão.

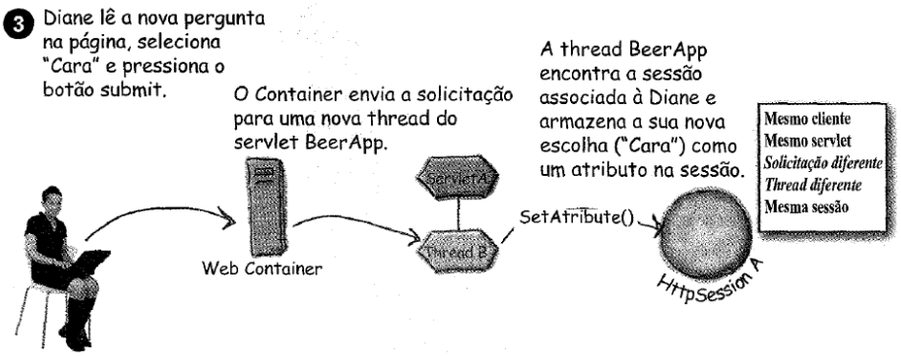
Web. xml – configurando tempo de inatividade para encerrar sessão

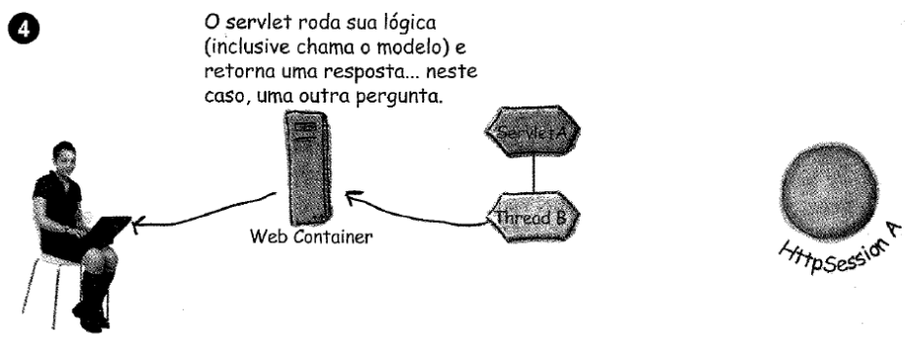
```
<session-config>
    <session-timeout>5</session-timeout>
</session-config>
```

- Tempo em minutos;
- Utilizado para todas as sessões criadas na aplicação.


1 Diane seleciona "Escura" e clica no botão submit.
O Container envia a solicitação para uma nova thread do servlet BeerApp.
A thread BeerApp encontra a sessão associada a Diane e guarda a sua escolha ("Escura") como um atributo na sessão.
ServletA
Thread A
SetAtribute()
Web Container
HttpSession A


2
O servlet executa a sua lógica (inclusive chama o modelo) e retorna uma resposta... neste caso, uma outra pergunta: "Qual é a variação do preço?"
ServletA
Thread B
Web Container
HttpSession A


3 Diane lê a nova pergunta na página, seleciona "Cara" e pressiona o botão submit.
O Container envia a solicitação para uma nova thread do servlet BeerApp.
A thread BeerApp encontra a sessão associada à Diane e armazena a sua nova escolha ("Cara") como um atributo na sessão.
Mesmo cliente
Mesmo servlet
Solicitação diferente
Thread diferente
Mesma sessão
ServletA
Thread B
SetAtribute()
Web Container
HttpSession A


4
O servlet roda sua lógica (inclusive chama o modelo) e retorna uma resposta... neste caso, uma outra pergunta.
ServletA
Thread B
Web Container
HttpSession A

E se outro cliente acessa a aplicação? Como o container vai saber quem é o cliente?

A ideia é simples: na primeira solicitação do cliente, o Container gera uma única session ID e a devolve para o cliente juntamente com a resposta. O cliente envia de volta a session ID com cada solicitação subsequente. O Container verifica o ID, encontra a sessão correspondente e a associa à solicitação.

Você pode se deparar com uma situação na qual o servlet quer usar apenas uma sessão já criada. Pode não fazer sentido para o servlet responsável pelo checkout, por exemplo, iniciar uma nova sessão.

Então, existe um método sobrecarregado getSession (booleano) apenas para esse propósito. Se você não quer criar uma sessão nova, chame o getSession (false) e você obterá ou um HttpSession nulo, ou um já existente.

O padrão arquitetural MVC – Model View Controller

Servlet ou JSP?

Colocar todo HTML dentro de um Servlet realmente não nos parece a melhor ideia. O que acontece quando precisamos mudar o design da página? O designer não vai saber Java para editar o Servlet, recompilá-la e colocá-la no servidor.

Imagine usar apenas JSP. Ficaríamos com muito scriptlet, que é muito difícil de dar manutenção. Uma ideia mais interessante é usar o que é bom de cada um dos dois:

- JSP – Apresentação de resultados
- Não acessar dados, instanciar objetos...
- Código Java deve ser tratado pelo Servlet
- Servlet – Lógica de negócios
- JSP – Lógica de apresentação

O ideal então é que o Servlet faça o trabalho árduo, a tal da lógica de negócio. E o JSP apenas apresente visualmente os resultados gerados pelo Servlet. O Servlet ficaria então com a lógica de negócios (ou regras de negócio) e o JSP tem a lógica de apresentação.

Retirando o código Java dos JSP's:

EL – Expression Language

O código abaixo:

```
out.println("<html>");
out.println("<body>");
out.println("Contato " + contato.getNome() + " adicionado com sucesso");
out.println("</body>");
out.println("</html>");
```

Pode ser substituído pelo simples código no JSP:

```
<html>
    <body>
        Contato ${param.nome} adicionado com sucesso
    </body>
</html>
```

Uso da variável "param" responsável pelos parâmetros enviados na requisição. Estamos utilizando no exemplo acima asLanguage, A languagem desenvolvida pela Sun e interpretada pelo Servlet Container. Oficialmente, tomou-se parte da especificação a partir da especificação ISP 2.0. A EL é quase sempre o jeito mais simples de se fazer algo que você normalmente faria com scriptlets e expressões.

Exemplo:

utramig.br/cadastro.jsp?nome="marcelo"&idade=30

- Meu nome é ${param.nome}
- Minha idade é $ {param.idade}

Request Dispatcher

Uma forma de redirecionar as requisições entre os recursos do servidor é realizando o dispatch para renderizar o JSP após e execução das regras de negócio dentro do Servlet.

A API Servlets nos permite fazer tal redirecionamento. Basta conhecermos a URL que queremos acessar e podemos usar um objeto RequestDispatcher para acessar outro recurso Web, seja esse recurso uma página JSP ou um servlet:

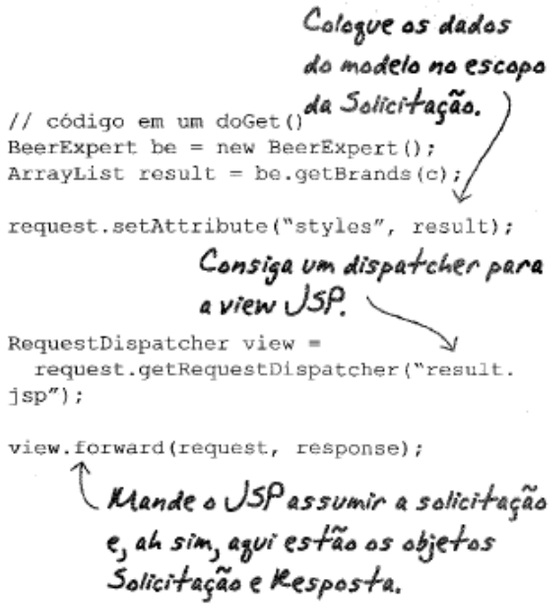
Exemplo:

RequestDispatcher rd = request.getRequestDispatcher(“/recurso.jsp");

rd.forward(request,response);

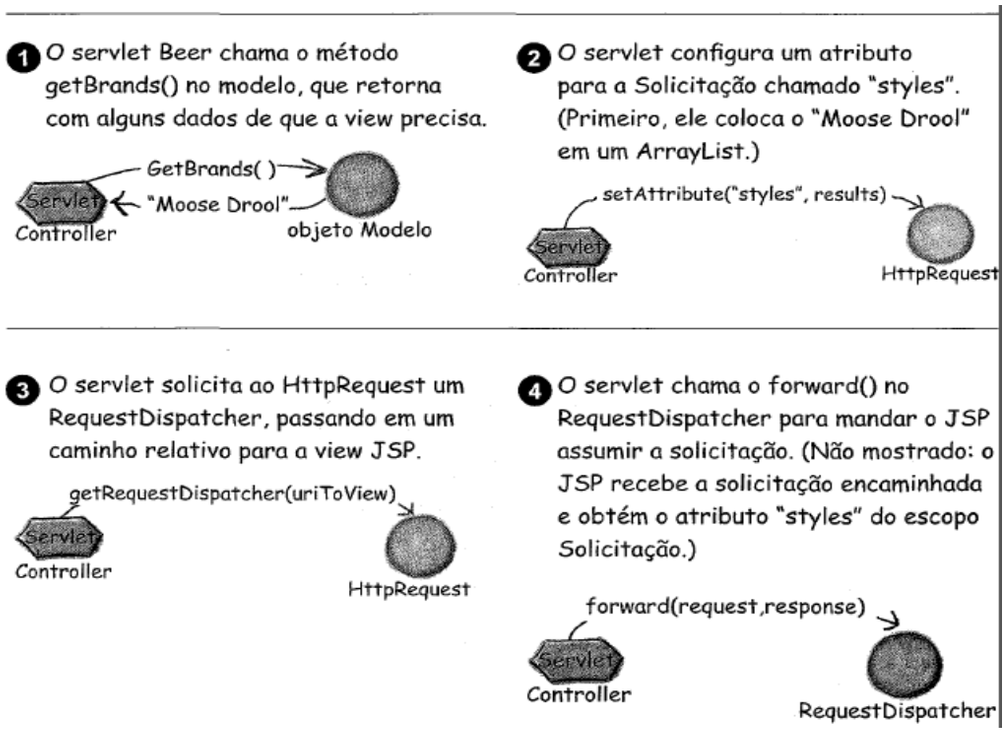
Podemos facilmente executar a lógica de nossa aplicação Web em um servlet e então redirecionar para uma página JSP, onde você possui seu código HTML e tags que irão manipular os dados trazidos pelo servlet.

Os atributos da solicitação fazem sentido quando você quiser que algum outro componente da aplicação assuma o controle de toda a solicitação ou de parte dela. Nosso exemplo simples e característico é uma aplicação MVC que inicia com um servlet controlador, mas termina em uma view JSP. O controlador comunica-se com o modelo e obtém os dados que a view



necessita para construir a resposta.

Não há nenhuma razão para colocar os dados em um contexto ou atributo de sessão, já que só vale para esta solicitação. Desta forma, o colocamos no escopo da solicitação. Então, como fazemos para que outra parte do componente assuma a solicitação? Com um RequestDispatcher.



O método forward só pode ser chamado quando nada foi ainda escrito para a saída. No momento que algo for escrito, fica impossível redirecionar o usuário, pois o protocolo HTTP não possui meios de voltar atrás naquilo que já foi enviado ao cliente.

Existe outro método da classe RequestDispatcher que representa a inclusão de página e não o redirecionamento. Esse método se chama include e pode ser chamado a qualquer instante para acrescentar ao resultado de uma página os dados de outra.

O método include () envia a solicitação para outro lugar (geralmente outro servlet), para realizar alguma tarefa e então voltar para o remetente! Em outras palavras, significa que o include () pede por ajuda no tratamento da

solicitação, mas não chega a ser uma ajuda completa. É apenas uma transferência de controle temporária, e não, permanente. Com o forward (), você está dizendo: "É isso aí, eu não vou fazer mais nada para processar esta solicitação e esta resposta." Porém, com o include(), você está dizendo: "Eu quero que outra pessoa faça algumas coisas com a solicitação e/ou resposta, mas quando ela terminar, eu mesmo quero acabar de tratar a solicitação e a resposta.

JSTL JavaServer Pages Standard Tag Library

A JSTL é a API que encapsulou em tags simples toda a funcionalidade que diversas páginas Web precisam, como controle de laços (fors), controle de fluxo do tipo if else, manipulação de dados XML e a internacionalização de sua aplicação.

Instalação

A JSTL 1.1 NÃO faz parte da especificação JSP 2.0! Ter acesso às APIs dos Servlets e JSP não significa que você tem acesso à JSTL. Antes de poder usar a JSTL, você precisa instalar o arquivo 'jstl.jar' no diretório: **WEBINF/lib** do seu aplicativo web. Isso significa que cada aplicativo web precisa de uma cópia. Para instalar a implementação mais famosa da JSTL basta baixar a mesma no site https://jstl.java.net/

API JSTL core

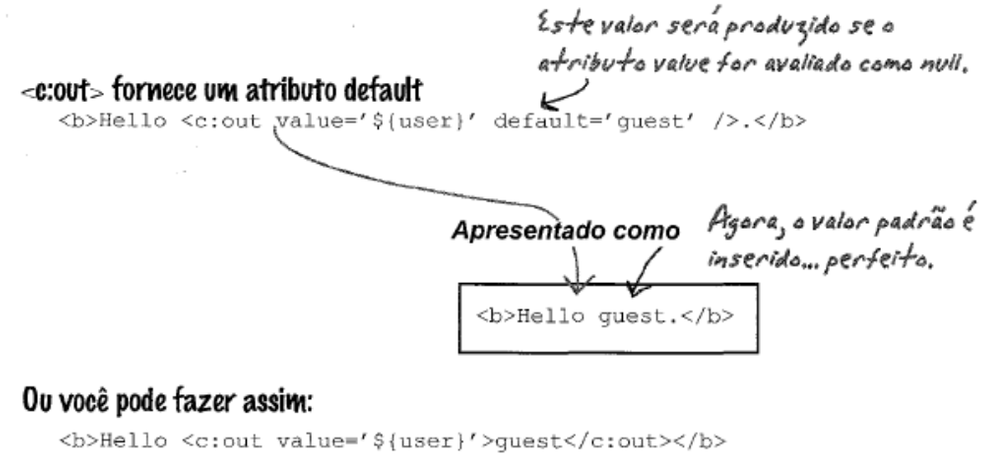
Sempre que vamos utilizar uma taglib devemos primeiro escrever um cabeçalho através de uma tag JSP que define qual taglib vamos utilizar e um nome, chamado prefixo. Esse prefixo pode ter qualquer valor, mas no caso da taglib core da JSTL o padrão da Sun é a letra c. Já a URI (que não deve ser

decorada) é mostrada a seguir e não implica em uma requisição pelo protocolo http e sim uma busca entre os arquivos. jar no diretório lib.

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
```

Tag <c:out>

Suponha que você queira mostrar a esses usuários anônimos uma mensagem que diz: "Hello guest" (Olá convidado). Este é um lugar perfeito para usar um valor padrão com a tag c: out. Simplesmente adicione um atributo default e forneça o valor que deseja imprimir, caso sua expressão seja avaliada como null:



Ou você pode fazer assim:

```
<b>Hello <c:out value='${user}'>guest</c:out></b>
```

Tag <c:forEach>

Imagine que você deseja algo que faça um loop através de uma coleção (por exemplo, um array de itens de um catálogo), que retire um elemento de cada vez, e que exiba esse elemento em uma linha de uma tabela gerada dinamicamente. Não é possível programar de antemão a tabela completa - você não tem ideia de quantas linhas existirão no momento da execução e, é claro, você não sabe quais são os valores correspondentes.

A resposta é a tag <c:forEach>. Isto requer um conhecimento bastante ligeiro sobre as tabelas HTML, mas incluímos aqui notas para os leitores que não estiverem familiarizados com o tópico.

Em um JSP, *com* scripting

```
<table>
<% String[] items = (String[]) request.getAttribute("movieList");
   String var=null;
   for (int i = 0; i < items.length; i++) {
      var = items[i];
 %>
  <tr><td><%= var %></td></tr>
  <% } %>
</table>
```

Código JSP

(Falaremos sobre esta diretiva taglib mais adiante no capítulo.)

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<html><body>
<strong> Movie list:</strong>
<br><br>

<table>
  <c:forEach var="movie" items="${movieList}" >
     <tr>
        <td>${movie}</td>
     </tr>
  </c:forEach>
</table>

</body></html>
```

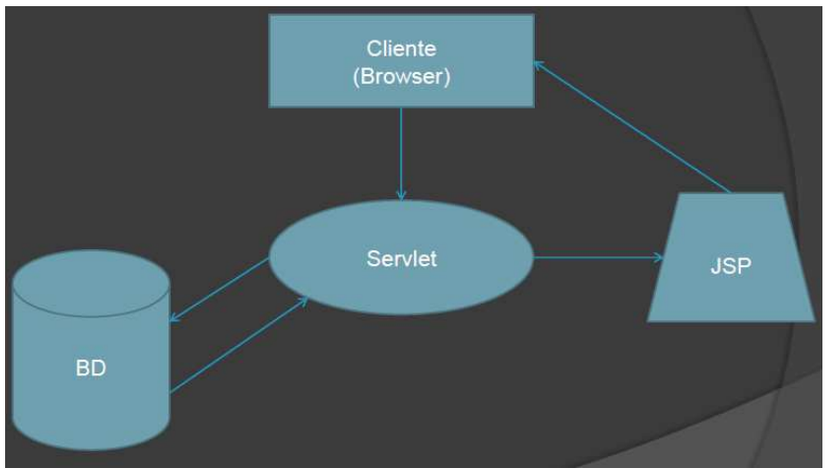
Faz um loop através do array inteiro (o atributo "movieList") e exibe cada elemento em uma nova linha. (Esta tabela tem apenas uma coluna por linha.)

Outras Tags

- <c:catch> - bloco do tipo try/catch
- <c:forTokens> - for em tokens (ex: "a,b,c" separados por vírgula)
- <c:param> - parâmetro
- <c:redirect> - redirecionamento
- <c:remove> - remoção de variável

- <c:set> - criação de variável

Separando a apresentação da camada de negócios



O esboço acima já representa tudo que precisamos no nosso modelo MVC?

Não, até o momento apenas separamos a camada de apresentação (View) da camada de negócios. É o começo, mas e se precisarmos migrar nossa aplicação para desktop, utilizando componentes gráficos em Swing?

Não podemos ter lógica de negócios diretamente nos Servlets.

Utilizando MVC, a camada de negócios não somente deve ser separada da camada de apresentação... Ela nunca deve saber da sua existência. Devemos criar uma camada de modelo, onde a lógica de negócios fique em classes java próprias, reutilizáveis, independentes do tipo de visualização.

A essência do MVC é que você pode separar a lógica de negócio da apresentação, mas ponha algo entre elas para que a lógica de negócio possa agir sozinha como uma classe Java reutilizável, sem precisar saber nada sobre a view.

No nosso esboço acima, separamos a lógica de negócio da apresentação, mas a lógica ainda tem uma forte ligação com a view. Ou seja, colocamos a lógica de negócio no servlet, o que significa que não podemos reutilizar a lógica em outros tipos de view (como um GUI Swing). A lógica está

presa em um servlet, quando deveria estar em uma classe Java pura reutilizável.

Camada de Modelo (Model)

É a combinação entre os dados de negócio e as regras que operam estes dados. Esta camada sabe as regras para “pegar” e alterar o estado dos dados de negócio.

Exemplo:

- Caso a data de validade do produto seja menor que a data atual, o produto tem o estado alterado para vencido.
- Não listar os produtos vencidos.

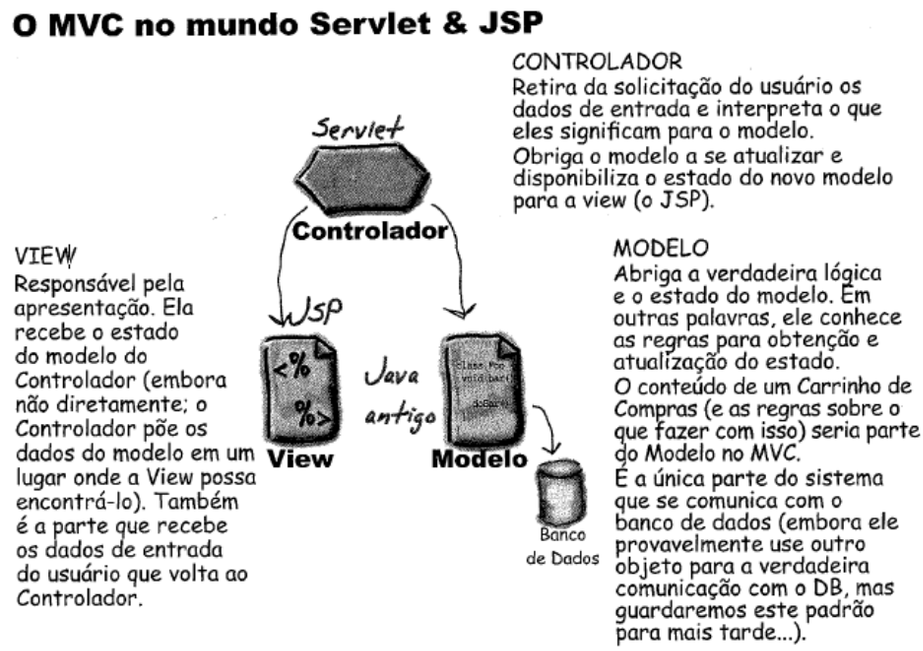
Esta é a única camada que deve falar com a base de dados.

Camada de Controle (Controller)

Recupera a requisição do cliente e traduz de maneira que a camada de modelo entenda. Aciona o modelo para uma eventual alteração de estado e disponibiliza esta atualização para a camada de visão.

Camada de Visão (View)

Responsável pela apresentação do estado do modelo através do controle. Esta apresentação não deve ser direta, o controle deve intermediar esta alteração de estado e torná-la disponível a esta camada.



Padrão DAO – Data Access Object

Nosso esboço foi melhorado, mas podemos ainda isolar todo o acesso a banco em classes bem simples, cuja instância é um objeto responsável por acessar os dados. Da responsabilidade deste objeto surgiu o nome de Data Access Object ou simplesmente DAO, um dos mais famosos padrões de projeto (design pattern).

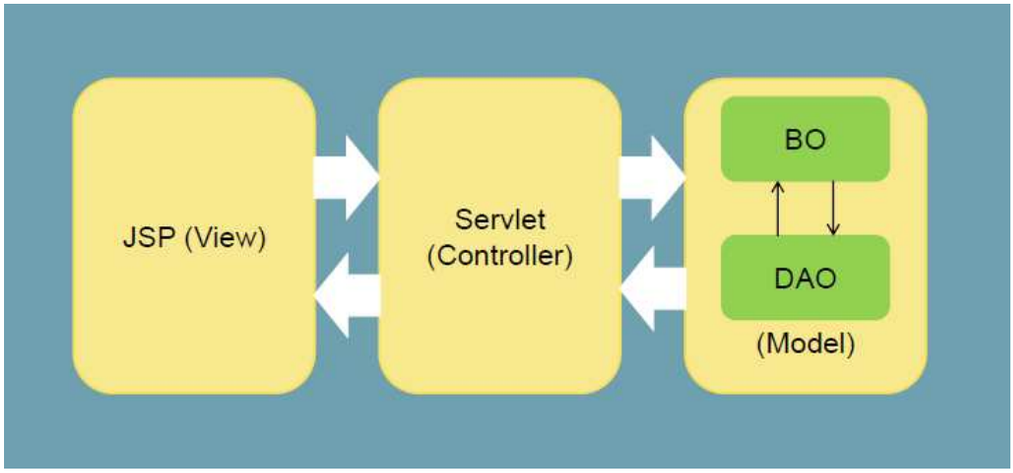
O padrão DAO é um padrão de projeto que abstrai e encapsula os mecanismos de acesso a dados escondendo os detalhes da execução da origem dos dados. É um dos muitos padrões que podemos usar para desacoplar o código de acesso e persistência dos dados da lógica da aplicação.

O objetivo do padrão é tornar o código mais organizado, fácil de manter e entender e de poder ser reutilizado em outras aplicações. Este padrão permite criar as classes de dados independentemente da fonte de dados ser um BD relacional, um arquivo texto, um arquivo XML, etc.

Padrão BO – Business Object

É empregado para separar dados e lógica de negócio usando um modelo orientado a objeto. A classe BO fará, portando, o encapsulamento das regras de negócio e do gerenciamento da camada de persistência (dao).

Chegamos então ao nosso modelo final:



Referências Bibliográficas

Head First Servlets and JSP – Basham, Bryan / Sierra, Kathy / Bates, Bert – O'Reilly.

http://www.caelum.com.br/curso/fj-21-java-web/

http://www.dicionarioweb.com.br

http://www.slideshare.net/GabrielFaustino/parte9-fundamentos-de-sistema-de-informao

http://www.slideshare.net/artinfo/segurana-em-banco-de-dados

MURPHY, C. Tying it all together. Information Week, Manhasset, n. 931, p. 34-38, mar. 2003.

Carvalho, J, 1996, Sistemas de Informação e Desenvolvimento de Sistemas de Informação, Cap. 3, Universidade do Minho.

MORESI, E. A. D. Delineando o valor do sistema de informação de uma organização. **Ciência da Informação**, Brasília, v. 29, n. 1, p. 14-24, jan./abr. 2000.

CRONIN, Blaise. Esquemas conceituais e estratégicos para a gerência da informação. Revista da Escola de Biblioteconomia da UFMG, v.19, n. 2, p. 195-220, Set 1990.

WETHERBE, James C. Análise de sistema para sistemas de informação por computador. 3a Ed. Rio de Janeiro: Campus, 1987.

BAND, William A. Competências críticas: dez novas ideias para revolucionar a empresa. Rio de Janeiro: Campus, 1997.

NETO, Roberto C. Regras de Negócio-Por que você deveria se importar com isso? http://www.linhadecodigo.com.br/artigo/3205/regras-de-negocio-por-que-voce-deveria-se-importar-com-isso.aspx

http://www.liber.ufpe.br/teses/arquivo/20050228173419.pdf

SILVA, Carla T. L. L. Silva Detalhando o projeto arquitetural no desenvolvimento de software orientado a agentes: O caso Tropos, 2003.

http://www.slideshare.net/GabrielFaustino/parte9-fundamentos-de-sistema-de-informao

http://www.slideshare.net/artinfo/segurana-em-banco-de-dados

CURSO TÉCNICO EM

INFORMÁTICA

ETAPA 3

MARKETING EM SISTEMAS DE INFORMAÇÃO

Sumário

1- DEFINIÇÃO DE MARKETING 374
O MARKETING COMO FILOSOFIA 374
AFINAL, O QUE É MARKETING? 375
2- ALGUNS CONCEITOS SOBRE MARKETING 375
3 - VARIÁVEIS UTILIZADAS EM UMA ANÁLISE DO AMBIENTE DE MARKETING 379
4 – QUANTAS DIFERENÇAS PROMOVER? 380
5 – MIX DE MARKETING 380
6 – GERENCIAMENTO E ESTRATÉGIA PARA PRODUTOS E SERVIÇOS 382
7- ESTRATÉGIAS DE PREÇO 382
8 – MÍDIA PARA DIVULGAÇÃO 383
9 – COMPOSTO PROMOCIONAL 384
9.1 – PROPAGANDA 384
9.2 - PROMOÇÃO DE VENDAS 384
9.3 - RELAÇÕES PÚBLICAS E PUBLICIDADE 384
9.4 – PUBLICIDADE 384
9.5 - FORÇA DE VENDAS 385
9.6 - MARKETING DIRETO 385
10- CARACTERÍSTICAS E IMPLICAÇOES DO SERVIÇO PARA A PRÁTICA
DA ADMINISTRAÇÃO DE MARKETING 386
ADMINISTRAÇÃO DE MARKETING 387
ESTADOS DA DEMANDA..... 388
11 – WEB MARKETING 389
MILHARES DE NOVOS CANAIS DE COMUNICAÇÃO 389
A TRANSFORMAÇÃO DO TANGÍVEL NO INTANGÍVEL 390
MARKETING DE PERMISSÃO 390
CONVENIÊNCIA AO CONSUMIDOR 391
CUSTOMIZAÇÃO: PALAVRA DE ORDEM EM WEB MARKETING 391
O NOVO MUNDO DA CONCORRÊNCIA 392
REDUÇÃO DE CUSTOS 393
UM CONSUMIDOR DIFERENTE, EXIGENTE, PORÉM ATRAENTE 393
12 – MARKETING E ÉTICA 394
CÓDIGO DE ÉTICA DA ABEMD 394
13 – TÉCNICAS DE APRESENTAÇÃO 404
BIBLIOGRÁFIA SUGERIDA 420

UTRAMIG

Fundação de Educação para o Trabalho de Minas Gerais

Disciplina: Marketing em SI

Carga horária semanal: 2

Carga horária total: 40

Objetivos Gerais:

- Conhecer e utilizar o marketing e suas ferramentas como estratégia de maximizar a satisfação de seus porvindouros stakeholder;

Avaliação:

1º Bimestre

Critérios de distribuição de pontos

Atividades	Valor
Discussão em sala sobre os conceitos vistos no filme.	3,0
Apresentação dos projetos para a Mostra para avaliação das técnicas de apresentação.	10,0
Apresentação de trabalho interdisciplinar (Marketing e Aplicativos II) em grupo – prévia da Mostra.	7,0
Avaliação bimestral	20,0

2º Bimestre

Critérios de distribuição de pontos

Atividades	Valor
Desenvolvimento de mídias para divulgação na Mostra	30,0
Participação em sala	10,0
Rodada de Palestras	20,0

Total de pontos: 100

1- DEFINIÇÃO DE MARKETING

No Brasil, o conceito de marketing encontra-se, ainda hoje, bastante desfocado. Muitos o associam com a venda de produtos de qualquer modo, mesmo que as pessoas não os desejem. Outros acreditam tratar-se de uma maneira de fazer com que as pessoas comprem o que não precisam, com um dinheiro que não têm. Muito dessa distorção se deve ao fato de a aplicação do marketing ter ocorrido no Brasil quando ainda tínhamos uma economia composta por monopólios e oligopólios não competitivos (década de 60), em que o governo tinha uma função muito mais de gestão do que de tutela da economia.

Na verdade, ele é fruto de um estudo baseado em diversas ciências (Sociologia, Psicologia, Matemática, Antropologia, Estatística, Filosofia, entre outras), tendo como objetivo conhecer o comportamento das pessoas e, a partir disso, satisfazer às necessidades e desejos de cada uma.

O marketing tornou-se uma força difundida e influente em todos os setores da economia. Em poucos anos despojou-se de sua antiga imagem de algo antiético e desnecessário e passou a ser visto como um instrumento essencial para a formação e manutenção de diversos negócios, tendo inclusive seus conceitos aplicados nos mais variados tipos de organização, desde times de futebol a igrejas, passando por governos e organizações não governamentais.

O marketing tornou-se uma força difundida e influente em todos os setores da economia

O marketing como filosofia

Ao considerar a verdadeira visão do conceito de marketing dentro das organizações, verifica-se que, em essência, ela deve se estender por praticamente toda a organização, principalmente para aquelas diretamente relacionadas ao mercado. Neste sentido, Raimar Richers, uma das maiores autoridades em marketing no Brasil, define sua função como sendo simplesmente a intenção de entender e atender o mercado.

Pensando desta forma, todas as atividades relacionadas com a busca da satisfação de clientes, sejam eles internos ou externos, têm uma relação direta com os responsáveis pelo marketing. Se se concentrar nos clientes a principal razão de as empresas existirem, as ações que influenciarão negativa ou positivamente na sua satisfação têm, nos executivos de marketing, seus responsáveis.

Sob essa perspectiva, veremos que o marketing representa muito mais do que ferramentas de promoção e vendas; trata-se de uma filosofia dentro das organizações, filosofia esta que tem no cliente a principal razão da existência da organização.

Afinal, o que é marketing?

De uma forma geral e simplista pode-se afirmar, de acordo com Kotler (2000), que marketing é um processo social por meio do quais pessoas e grupos de pessoas obtêm aquilo de que necessitam e desejam por meio da criação, oferta e troca de produtos e serviços.

Marketing é uma via de duas mãos entre o mercado e as organizações, em que estas buscam no mercado informações sobre seus desejos e necessidades, recebendo como retorno, nesta primeira fase, as informações.

Como passo seguinte, as organizações passam a oferecer ao mercado os produtos e serviços de acordo com os desejos e necessidades dos clientes, tendo como retorno recursos financeiros e clientes satisfeitos.

2- ALGUNS CONCEITOS SOBRE MARKETING

PRODUTO: do ponto de vista do marketing, produto é tudo aquilo que pode satisfazer os desejos e necessidades de um cliente. Também se pode chamar o produto de oferta ou solução. O produto é constituído de 3 componentes: bem físico, serviço e ideia. Quando se compra um sanduíche, por exemplo, se está comprando um bem físico (o sanduíche propriamente dito), um serviço (a preparação e o lugar para comê-lo) e uma ideia (comida rápida, economia de tempo). Quando se compra um carro, o bem físico é o carro, o serviço (transporte) e a ideia (autonomia, individualidade, praticidade).

NECESSIDADE: Em 1943, o psicólogo Abraham Harold Maslow publicou um trabalho que ficou conhecido no mundo inteiro que descrevia as necessidades humanas por meio de uma pirâmide. Maslow definiu as necessidades fisiológicas (fome, sede, abrigo), de segurança (sobrevivência física), as de relacionamento (a aceitação pelo meio e o sentido de importância), as de estima e status (relevância, domínio, reputação, prestígio) e, por fim, as de auto realização (desejo de conhecer, compreender, sistematizar, organizar e construir um sistema de valores). Ele estabeleceu ainda que as necessidades fisiológicas (a base da pirâmide) dominam o comportamento da pessoa até que sejam satisfeitas, em seguida a segunda torna-se preponderante até que seja satisfeita, e nesta ordem até a última. As necessidades não são criadas por empresas ou pela sociedade, elas são inerentes à condição humana.

DESEJOS: Desejos são carências por satisfações específicas para atender às necessidades. Assim, se uma pessoa está com fome, a sua necessidade é de apenas alimentos. Mas um americano típico, por exemplo, quando está com fome, pode pensar em um hambúrguer. Já um monge tibetano pensa em algumas frutas, arroz e lentilha. Quando compra um carro, pode-se pensar que a pessoa está suprindo uma necessidade de se transportar, mas, na verdade ela também quer ser aceita por um determinado grupo, quer demonstrar o seu status social, quer ser admirada. As necessidades do ser humano são poucas, porém, seus desejos são muitos. E eles são influenciados com toda a força pelas instituições sociais: cultura, religião, família, escola, amigos, etc. O marketing trabalha principalmente com desejos.

DEMANDAS: Demandas são desejos por produtos específicos respaldados pela habilidade e disposição em comprá-los. Assim, em vez de querer um simples tênis para praticar esportes, uma pessoa quer um Nike. Muitas pessoas desejam um Mercedes, mas muito poucas podem m realmente comprá-lo. Assim, a empresa deve calcular a demanda por seu produto baseada no número e no perfil de pessoas que desejam os seus produtos e que podem comprá-los.

COMPRA

Racional

- Essencial, antes da aquisição avaliam-se os prós e os contras do produto/serviço. É PENSADA.

Emocional

- Desnecessária, adquire-se sem levar em consideração os aspectos negativos do produto/serviço.

MERCADOS: O mercado consiste em todos os consumidores potenciais que compartilham de uma determinada necessidade ou desejo e possuem condições de adquirir um produto que as satisfaçam. Assim, para ser considerado mercado, um grupo de consumidores precisa ter o desejo, as condições financeiras e a disposição em adquirir o produto. Assim, podemos falar no mercado de necessidades (ex: mercado de dietéticos), produtos (ex: mercado de calçados), demográficos (ex: mercado de jovens) e geográficos (ex: mercado brasileiro).

PÚBLICOS-ALVO: Nenhuma empresa consegue satisfazer todos os desejos e necessidades em todos os mercados. Assim, para conhecer bem o consumidor e adivinhar o que ele gostaria de ter, satisfazendo seus desejos e necessidades, é necessário que se delimite os mercados. Só assim a comunicação atingirá o seu objetivo, usando a linguagem que aquele consumidor entende e atendendo aos seus desejos particulares. Assim, públicos-alvo são sempre delimitados e têm um perfil bem definido. Ex: a nova linha de batons foi concebida para mulheres acima de 30 anos, independentes, bom nível cultural e financeiro. As bermudas da marca tal atendem a jovens rapazes adolescentes, de classe média alta e praticantes de skatismo.

VALOR: O valor tem a ver com o modo como o consumidor avalia os benefícios que um determinado produto poderá lhe trazer. Uma pessoa precisa se vestir para se agasalhar, então dispõe de várias opções de escolha: camisetas, blusas, jaquetas, etc. Mas ela gostaria também de satisfazer outras necessidades como: conforto, economia, beleza, sofisticação, qualidade. Uma camiseta popular

qualquer proporciona economia, mas não beleza e sofisticação. Além disso, por não ter qualidade, a sua durabilidade será reduzida. A pessoa pesará o que é mais importante no momento. Dependendo da sua situação financeira, social e emocional, escolherá a opção que satisfaça as suas exigências ao menor custo possível. Se o mais importante for a sua integração em um grupo social, a qualidade e o status que a camiseta pode lhe proporcionar, a pessoa pode decidir que a camiseta escolhida possui valor, mesmo que aparentemente seu preço seja absurdamente mais caro. Isso explica porque algumas pessoas pagam verdadeiras fortunas por objetos que aparentemente não o valem. É porque o valor é um conceito subjetivo, que depende de cada pessoa e seu contexto.

POSICIONAMENTO: O posicionamento é literalmente a posição que uma empresa ou produto ocupa na mente de um cliente. A palavra foi popularizada com essa conotação por dois executivos da propaganda, Al Ries e Jack Trout. Eles argumentam que, quando uma marca tem uma posição clara na mente do cliente, dificilmente outra consegue roubar o seu lugar. Há várias estratégias para fazer isso e ser lembrado: a primeira é fortalecer a sua posição atual. Assim, por exemplo, a Coca-Cola assume a posição de maior fábrica de refrigerantes do mundo, a Gol como a companhia aérea brasileira com a melhor relação custo-benefício, e assim por diante. A segunda estratégia é buscar uma posição não ocupada, mas valorizada por um nicho de consumidores. Assim, a empresa de cosméticos Natura se diferencia quando se mostra ecológica e socialmente responsável na seleção de seus fornecedores de matéria-prima na selva amazônica. Como nenhuma outra empresa mostra isso, a Natura tornou-se uma referência nesse segmento. Há várias outras estratégias e todas dependem intrinsecamente das características do produto ou empresa e do público-alvo. Uma empresa pode, por exemplo, posicionar-se por um atributo específico (ser a mais antiga, a maior, etc.), por benefício (um batom pode ser o único com gosto de jabuticaba), por usuário (a melhor máquina de lavar roupa para pessoas solteiras ou famílias pequenas), por concorrente (o leite que possui mais vitaminas que todos os seus concorrentes), por categoria (a melhor instituição de ensino de geografia do país), etc.

SATISFAÇÃO: É o desempenho do produto em relação ao **VALOR** relativo às expectativas do comprador.

QUALIDADE: Para o Marketing, QUALIDADE **não é "ausência de defeitos"**, mas a satisfação do cliente.

"AS EMPRESAS INTELIGENTES ENCANTAM OS CLIENTES, PROMETENDO SOMENTE O QUE PODEM OFERECER E OFERECENDO MAIS DO QUE PROMETERAM."

MERCHANDISING: É a propaganda de um produto, fazendo-se menção ao mesmo, dentro de um programa de televisão, de rádio, no teatro ou cinema.

3 - VARIÁVEIS UTILIZADAS EM UMA ANÁLISE DO AMBIENTE DE MARKETING

Em uma análise é importante considerar uma série de variáveis, todas com o mesmo nível de importância, dependendo somente das características do mercado e do produto ou serviço em questão, pois pode variar de um levantamento para outro, de acordo com suas características. A análise ambiental é composta de diversas etapas. A Análise Macroambiente de Marketing envolve variáveis incontroláveis, como as citadas neste capítulo. AAnálise Microambiente de Marketing envolve variáveis controláveis, como fornecedores, consumidores, concorrentes e intermediários de marketing. Segundo Kotler (2000), as principais variáveis a serem consideradas são:

Variáveis Econômicas - São considerados aqui os fatores econômicos que envolvem o mercado em estudo. Deve ser levantado até que ponto as variações na economia podem comprometer positiva ou negativamente o mercado onde a organização atua ou pretende atuar.

Variáveis Demográficas - É importante monitorar a população, pois as pessoas representam o mercado. Podem ser consideradas, por exemplo, variações no número de casamentos e de filhos de uma determinada população, ou ainda o tamanho e a taxa de crescimento da população em diferentes cidades, regiões e nações, distribuição etária e composto étnico, níveis educacionais, faixa etária, etc.

Variáveis Culturais - Dizem respeito ao grau em que a cultura de um mercado pode comprometer a aceitação de um determinado produto ou serviço. Elas envolvem culturas, subculturas, comportamento, influência das religiões, crenças, grau de tecnologia, etc.

Variáveis Tecnológicas - Principalmente nos dias atuais, quando a tecnologia evolui constantemente e quase que diariamente, este setor deve ser considerado em qualquer análise de ambiente. As mudanças e as facilidades de acesso à tecnologia fazem com que um produto se torne obsoleto rapidamente.

Variáveis Político-Legais - as mudanças do ambiente político, que ocorrem constantemente, e das Leis, principalmente no Brasil, devem ser vistas com grande atenção, pois elas podem inviabilizar um produto, serviço ou empreendimento.

4 – QUANTAS DIFERENÇAS PROMOVER?

Muitas empresas defendem a ideia de promover apenas um benefício para o mercado-alvo. Cada marca deve escolher um atributo e promover-se como número um naquele atributo. Os compradores tendem a se lembrar da mensagem número um, principalmente em uma sociedade de comunicação excessiva.

Quais são algumas das posições número um" a promover? As principais são melhor qualidade, melhor serviço, preço mais baixo, maior valor "e tecnologia mais avançada. Se uma empresa assumir um desses posicionamentos e cumpri-los de forma convincente, provavelmente será mais conhecida e lembrada por sua força.

5 – MIX DE MARKETING

Mix de marketing (ou composto de marketing) é o conjunto de ferramentas de marketing que a empresa utiliza para perseguir seus objetivos de marketing no mercado-alvo.

Essas ferramentas são classificadas em quatro grupos amplos, denominados os 4Ps do marketing: produto, preço, praça (ou ponto-de-venda) e promoção (do inglês product, price, place and promotion). As variáveis específicas de marketing sob cada P são mostradas na figura 1, a seguir.

A figura mostra a empresa preparando um mix de ofertas de produtos, serviços e preços, utilizando um mix de promoção, formado por promoção de vendas, publicidade, força de vendas, relações públicas, mala-direta, telemarketing e internet, para alcançar os canais de distribuição e os clientes-alvo.



Traduzidos do inglês, os 4Ps são definidos como:

- Product (Produto): o produto deve, com obrigatoriedade, ser o produto desejado pelo cliente, dentro das suas expectativas e que satisfaçam às suas necessidades.
- Price (Preço): o cliente procurará um preço justo, que não deve ser nem muito elevado – de modo que o cliente considere que não vale a pena comprá-lo nem tão baixo que o leve a pensar que há algo de errado com o produto, a ponto de recusá-lo.
- •Place (Ponto ou Distribuição): o produto desejado com um preço justo deve estar ao acesso do cliente, isto é, num local em que ele possa comprá-lo no momento que desejar.
- Promotion (Composto Promocional): há um provérbio popular que diz: A propaganda é a alma do negócio, e, realmente, ele tem toda a razão, pois se

não divulgarmos o produto aos clientes eles não saberão da sua existência e não poderão adquiri-lo.

Para se obter sucesso com os 4 P´s, existem os 4 A´s.

- Análise – necessidade de se conhecer o mercado.
- Adaptação – acontece logo após a análise e leva em consideração as informações obtidas.
- Ativação – implementação das mudanças necessárias.
- Avaliação – análise da eficiência dos processos e da necessidade de mudanças.

6 – GERENCIAMENTO E ESTRATÉGIA PARA PRODUTOS E SERVIÇOS

De nada adianta tentar vender algum produto ou serviço se ele não estiver em conformidade com os desejos do consumidor.

Mas o que é um produto ou serviço? É algo que pode ser oferecido a um mercado para sua apreciação, aquisição, uso ou consumo, que possa satisfazer um desejo ou uma necessidade. Inclui objetos físicos, serviços, lugares, organizações e ideias.

Para entender melhor o que é um bem, há necessidade de compreender que não se trata somente de algo tangível (palpável ou físico), que compramos e levamos para casa. Muitas vezes estão, junto a ele, serviços, como as férias em um hotel fazenda ou um show de rock.

A Sadia, fabricante de alimentos, conhecedora de que as mulheres atualmente estão muito mais ocupadas (conclusão decorrente de uma efetiva análise ambiental), sabendo que elas estão no mercado de trabalho e têm pouco tempo para se dedicar ao marido e aos filhos, passou a oferecer uma linha de produtos congelados. Mas ela não ofereceu sua nova linha de produtos salientando suas qualidades; ela sugeriu o benefício da liberdade, que é algo de que as mulheres se vangloriam nos dias de hoje. Nesse caso, fica explícito que o conceito

de liberdade é muito mais valorizado do que o produto em si, e as consumidoras que desejam a liberdade em seus casamentos compram-no orgulhosas.

7- ESTRATÉGIAS DE PREÇO

A determinação do preço de um produto ou serviço nem sempre é uma tarefa simples. É preciso considerar que, quando bem posicionado, o preço de um produto ou serviço é fator primordial ao sucesso de um Plano de Marketing.

Ao se elaborar a estratégia do preço de um produto deve-se considerar o seguinte: ele deve ser suficientemente alto, para proporcionar lucro a quem o está produzindo ou comercializando, porém não pode ser tão alto que desestimule a compra. Afinal, sempre se procura comprar produtos mais baratos. Ele também deve ser suficientemente baixo, a fim de que o produto seja atrativo aos clientes. Contudo, não deve ser demasiadamente baixo, pois pode depreciar o produto aos olhos dos clientes, que podem pensar que há algo de errado com o produto ou serviço, além de não ser interessante produzi-lo e comercializá-lo, pois não gerará lucro significativo.

A empresa deve decidir onde posicionar seu produto em termos de qualidade e preço. Pode posicioná-lo no centro do mercado, três níveis acima ou três níveis abaixo.

QUADRO 1 - QUALIDADE DO PRODUTO

	ALTO	PREÇO MÉDIO	BAIXO
Alta	1. Estratégia premium	2. Estratégia de alto valor	3. Estratégia de valor supremo
Média	4. Estratégia de preço alto	5. Estratégia de preço médio	6. Estratégia de valor médio
Baixa	7. Estratégia de desconto	8. Estratégia de falsa economia	9. Estratégia de economia

Dessa forma, é pertinente indagar: Afinal, qual é o preço ideal de um produto? O preço ideal de um produto é simplesmente aquele que o cliente julga justo e que ao mesmo tempo é interessante para a empresa.

8 – MÍDIA PARA DIVULGAÇÃO

Distribuição, em marketing, significa disponibilizar o produto ou serviço ao cliente da forma que seja mais fácil e conveniente adquiri-lo. São várias as formas de distribuição. Dentre as principais estão:

- •Distribuição Direta

Ocorre quando o produtor do serviço ou produto os vende diretamente ao consumidor.

Exemplo: algumas empresas utilizam-se das chamadas lojas da fábrica, em que fabricantes de produtos vendem diretamente aos consumidores. Também pode-se citar os prestadores de serviços, os quais executam, eles mesmos, o serviço para o consumidor, como os dentistas ou os cabeleireiros.

- Distribuição Indireta

Ocorre quando o produtor do produto ou serviço utiliza-se de distribuidores para levar o produto ou serviço até o consumidor.

Exemplo: quase que a totalidade dos produtos encontrados nos supermercados não é fabricado por eles. Os supermercados são intermediários entre o fabricante e o consumidor.

9 – COMPOSTO PROMOCIONAL

Às vezes, propaganda é confundida com marketing, tal é a sua importância. Também deve ser considerado que, pelas suas características, o Composto Promocional é a parte do marketing que mais aparece, é percebida pelas pessoas de uma forma geral e apresenta cinco ferramentas promocionais: propaganda, promoção de vendas, relações públicas e publicidade, força de vendas e marketing direto.

9.1 - Propaganda

É toda comunicação em que se paga pela veiculação, a exemplo de um anúncio classificado.

9.2 - Promoção de vendas

As ferramentas de promoção de vendas são: cupons, concursos, prêmios, entre outras, e possuem três características distintas: de comunicação, que visa atrair a atenção dos consumidores; de incentivo, que visa estimular o consumo; e de convite, que objetiva convidar para uma transação imediata.

9.3 - Relações públicas e publicidade

Trata-se do desenvolvimento de apelos junto ao consumidor utilizando se histórias e dramatizações.

9.4 - Publicidade

É toda comunicação em que não se paga por sua veiculação. Por exemplo, quando um artigo sobre um produto é publicado em um jornal ou revista, as características do produto são comunicadas ao público sem ônus para a empresa.

9.5 - Força de vendas

É a ferramenta mais eficaz em termos de custo nos estágios finais do processo de compra, particularmente no desenvolvimento da preferência, convicção e ação do consumidor. A venda pessoal envolve relacionamento ao vivo, imediato e interativo com o consumidor, permitindo uma relação duradoura.

9.6 - Marketing direto

A identificação da Potencialidade do Marketing Direto é simples. Basta imaginar a quantidade de ações de marketing em massa que se recebe diariamente sem participar da meta (target) a que elas se destinavam. Quantas vezes você ouviu apelos para que comprasse um CD de um certo estilo de música, quando você detesta tal estilo? Ou quantas propagandas incitam-no a consumir cerveja, sendo que você é avesso a bebidas alcoólicas? Este é o grande motivo pelo qual o

marketing direto vem obtendo sucesso e crescimento no Brasil e no mundo, nos últimos anos. Ele identifica quem devemos atingir e direciona a estratégia de marketing até essas pessoas.

As ferramentas do marketing direto são:

- Mala Direta - principal representante do marketing direto é uma forma de comunicação direta dirigida a quem pretendemos informar sobre um produto ou serviço. Quanto mais personalizada a mensagem, maior será o impacto do destinatário ao recebê-la.
- Telemarketing - trata-se da utilização do telefone como ferramenta de marketing. Ele pode ser usado como forma de vender, comunicar, pesquisar ou prospectar clientes.
- Catálogo - ao mesmo tempo em que comunica a existência e os atributos de um produto, o catálogo é também uma forma de distribuição direta.
- Cuponagem - estratégia muito comum em outros países e ainda incipiente no Brasil, consiste em distribuir a pessoas específicas promoções especiais de compra de alguns produtos em determinados pontos de venda.
- Internet - utilização da rede mundial de computadores para comunicação e interligação virtual entre fornecedores e consumidores. Até agora não foi utilizado todo o potencial desta ferramenta, pois há desconhecimento, por parte dos consumidores, sobre o conjunto de recursos que ela pode oferecer. Vem-se utilizando e-mails como forma de comunicação, havendo alguns casos de sucesso.

10- CARACTERÍSTICAS E IMPLICAÇOES DO SERVIÇO PARA A PRÁTICA DA ADMINISTRAÇÃO DE MARKETING

O que é um serviço? Como defini-lo? Como diferenciar serviço e produto?

Uma boa definição pode ser a seguinte: Serviço é um desempenho, essencialmente intangível, que não resulta na propriedade de algo. O serviço pode ou não estar ligado a um produto físico.

Assim, serviços são desempenhos no tempo e espaço que geram valor para o cliente por meio de uma transformação, uma experiência de serviço.

Quando alguém vai ao cabeleireiro, não fica proprietário do corte de cabelo; na verdade, sofre o corte de cabelo. Durante uma aula, um aluno não se torna dono da aula, mas sofre, recebe a experiência da aula, que deve gerar uma transformação, gerando valor. O mesmo ocorre num consultório médico, num restaurante, numa oficina mecânica: o cliente não se torna dono do serviço. O serviço é um desempenho transformador, intangível em essência, mesmo quando ligado a um produto físico. Mesmo em serviços que possuem um forte componente de bens, produtos e máquinas, como o transporte aéreo, o cliente não pode levar a viagem para casa e mostrar aos amigos. Ele pode, sim, viver a experiência transformadora da viagem e, no máximo, mostrar a passagem, as fotos, alguns elementos tangíveis do serviço. Esses elementos inclusive são fundamentais para tangibilizar o serviço, mas essa já é uma outra história, da qual falaremos mais tarde.

Um serviço também precisa ser entendido por suas características: inseparabilidade, variabilidade, intangibilidade e perecibilidade.

O serviço é um desempenho transformador, intangível em essência, mesmo quando ligado a um produto físico

Administração de Marketing

O processo de Administração de Marketing compreende o Planejamento e Operacionalização do Marketing-Mix, a fim de interagir com o mercado, adequando à oferta da organização ou estimulando a demanda.

Análise, planejamento, implementação e controle dos programas destinados a criar, desenvolver e manter os relacionamentos com os clientes.

Além do Marketing por si mesmo, encontram-se na Administração do Marketing:

- Marketing de Relacionamento

"Marketing de relacionamento é o processo por meio do qual ambas as partes – o

"cliente e o fornecedor – estabelecem relações eficazes, eficientes, agradáveis, entusiastas e éticos, isto é, pessoalmente, profissionalmente e proveitosamente recompensadores para ambas as partes." (Michael Porter – 1993).

- Demarketing

O demarketing geralmente visa desencorajar a demanda total através da elevação de preço e redução de promoção e serviços.

O demarketing seletivo visa reduzir a demanda a partir dos segmentos de mercado menos rentáveis ou que precisam de menos serviços. O demarketing não visa destruir a demanda, mas apenas reduzir o seu nível, temporária ou permanente.

- Endomarketing

"Marketing interno, in marketing, endomarketing é estar constantemente atraindo, desenvolvendo, motivando e retendo empregados qualificados através de trabalhos que satisfaçam suas expectativas. É, ainda, a filosofia de tratar empregados como clientes preferenciais." (BEKIN, 2004).

Estados da demanda:

Demanda Inexistente: Os clientes- alvo podem estar desinteressados e indiferentes ao produto, ou seja, agricultores podem estar desinteressados por um novo método agrícola e estudantes universitários podem estar desinteressados por cursos de línguas estrangeiras. A tarefa de marketing é encontrar alternativas de associar os benefícios do produto às necessidades e interesses naturais das pessoas.

Demanda Negativa: Um mercado está em um estado de demanda negativa se a maior parte do mercado não gosta do produto e pode até mesmo pagar um preço para evitá-lo. A pessoa tem demanda negativa por vacinas, tratamento dentário, vasectomia e operações de vesícula. O empregador tem demanda negativa por ex-presos e empregados alcoólicos. A tarefa de marketing é analisar por que o mercado não gosta do produto e desenvolver um programa de marketing, redesenhando o produto, baixando os preços e usando uma promoção muito efetiva a fim de mudar a crença e as atitudes do mercado.

Demanda Latente: Muitos clientes podem ter uma grande necessidade que não pode ser satisfeita por nenhum produto existente: há uma grande demanda latente por cigarros inofensivos, vizinhanças seguras e carros que gastam menos combustíveis. A tarefa do marketing é medir o mercado potencial e desenvolver produtos e serviços eficientes capazes de satisfazer essa demanda.

Demanda Declinante: Toda organização, mais cedo ou mais tarde, tem demanda declinante para um ou mais produtos. Igrejas assistem ao declínio do número de fiéis e faculdades particulares veem diminuídas as matrículas. Cabe ao homem de marketing analisar as causas do declínio do mercado e determinar se a demanda pode ser revigorada através de novos mercados-alvo, mudanças nas características dos produtos ou através de uma comunicação mais eficaz. A tarefa de marketing é reverter à demanda declinante via outro marketing criativo do produto.

Demanda Irregular: Muitas organizações enfrentam demanda que varia por estação, dia ou até mesmo horas causando problemas de capacidade ociosa ou excessiva. No trânsito intenso, a maioria do equipamento fica ocioso fora do período de pico e insuficiente no momento do pico. Museus são pouco visitados durante os dias de semana e superlotados nos fins-de-semana; as salas de operações de hospital são super reservadas no início de semana e praticamente vazias no fim de semana. A tarefa de marketing, chamada marketing de sincronização é encontrar alternativas de alterar o horário da demanda através de preço flexível, promoção e outros incentivos.

Demanda Plena: A organização tem demanda plena quando estão satisfeitas com o volume de negócios. A tarefa de marketing é manter o nível atual de demanda diante da mudança da preferência do cliente e da concorrência acirrada. A organização deve manter ou melhorar a qualidade e avaliar constantemente a satisfação do cliente para ter certeza de que está agindo corretamente.

Demanda Excessiva: Algumas organizações têm um nível de demanda maior do que o normal ou do que podem administrar: a Golden Gate Bridge tem um volume de tráfego maior do que pode aguentar e o Yellowstone National Park é super visitado durante o verão. A tarefa de marketing, chamada demarketing é encontrar alternativas para reduzir a demanda temporária ou permanentemente.

Demanda Indesejada: Produtos indesejados vão atrair esforços organizados para desencorajar o consumo. Campanha de não venda tem sido feitas contra cigarros, álcool, entorpecentes, armas, filmes censurados e famílias numerosas. A tarefa de marketing é fazer com que as pessoas que gostam de alguma coisa deixam de gostar, usando ferramentas, tais como: comunicações que atemorizam, preços elevados e escassez de produto.

11 – WEB MARKETING

É comum as pessoas utilizarem o termo marketing nas mais variadas situações, como na política: Ele está fazendo uma jogada de marketing para conquistar o cargo de vereador; ou na empresa: Você participou daquela reunião? Ele só fez o próprio marketing, não falou de mais ninguém.

A internet está revolucionando a forma de acesso à informação, e, consequentemente, criou mais uma forma de aplicação do marketing: web marketing. Pode-se dividir as ações de marketing em quatro segmentos, conhecidos como composto de marketing: Produto, Preço, Distribuição e Comunicação.

Deste modo, tudo que efetivamente a internet efetuar em termos de melhoria ou influenciar o processo deste composto pode ser tomado como ações de web marketing.

Esta nova definição provinda da internet não é apenas um conceito isolado ou alguma forma a mais de denegrir o verdadeiro marketing, que na sua essência significa propor que o vendedor descubra os desejos dos seus consumidores e, deste modo, ofereça produtos e serviços de real valor, proporcionando, no final deste processo de troca, satisfação para ambas as partes. Marketing não é e nunca será parte integrante da propaganda e publicidade. A propaganda e a publicidade é que fazem parte de um conceito maior chamado marketing, assim como a internet, atualmente, tornou-se mais uma ferramenta essencial para ações vencedoras de marketing. Mas, afinal, o que é web marketing?

Conceitua-se web marketing como uma nova ferramenta de comunicação e distribuição realizada por meio de recursos digitais, em que organizações e consumidores buscam interatividade total nos seus relacionamentos, proporcionando uma troca com satisfação rápida, personalizada e dinâmica. Mas

como a comunicação e distribuição foram afetadas de forma avassaladora pela internet? Não devemos ter a menor dúvida: o mundo mudou depois do advento da web, seja qual for o segmento de mercado ou nível socioeconômico.

Milhares de Novos Canais de Comunicação

Quando se chega a casa a primeira coisa a fazer é ligar a televisão? Nem sempre. Hoje em dia tem-se muitas outras alternativas de comunicação, entre elas a internet. Por que esperar o jornal da noite para acompanhar a cotação do dólar se é possível obtê-la minuto a minuto no computador? Por que esperar a música predileta tocar no rádio, se se pode escutá-la imediatamente no computador? São inúmeros os exemplos, mas o fato é que embora tenhamos mais opções, continuamos a ter apenas 24 horas por dia. Assim, escolheremos os canais de televisão que nos oferecerem maior qualidade, e visitaremos os websites – entre os muitos existentes – que nos agregarem maior valor, ou seja: quem ganha com isto é o consumidor.

A Transformação do Tangível no Intangível

Aproximadamente 60% das compras realizadas pela internet hoje são produtos relacionados à CDs e livros. Mesmo assim, algo de novo está ocorrendo com estes produtos. Eles estão deixando de serem produtos tangíveis para se tornarem arquivos de computador, sem perder sua funcionalidade original: a de tocar música e proporcionar a leitura. O mundo acompanhou o nascimento do formato de música MP3, um arquivo de computador capaz de tocar músicas com qualidade muito próxima à de um CD original, sendo que para adquiri-lo basta ter acesso à internet. Sabemos que ter acesso a uma música sem o pagamento devido é pirataria, mas esta é uma outra questão. O que se quer ressaltar aqui é o fato de que já se pode comprar música sem a necessidade de receber um CD pelo correio, mas sim recebendo um simples arquivo pelo e-mail. O crescimento do mercado de livros eletrônicos também é uma amostra dessa mudança de consumo.

Marketing de Permissão

Quando recebemos uma correspondência em casa oferecendo um produto através de um catálogo ou folder, nossa decisão é simples: abrir a carta, ler e decidir se a jogamos fora ou guardamos para futura consulta, mesmo porque o volume de cartas não é tão grande assim. O problema para o consumidor, com relação à internet, é que provavelmente também receberá, em seu endereço eletrônico, ofertas de produtos ou serviços em um volume muito maior, próximo ao insuportável. Atualmente, quando alguém envia mensagem de ordem comercial a um consumidor e este não autorizou o envio, o primeiro é acusado de invasão de privacidade, havendo, já, no Brasil, alguns casos de reclamação judicial. Nos Estados Unidos, este ato é considerado crime. Diante disso, qual a solução para enviar e-mails para possíveis clientes? A resposta é: marketing de permissão, que consiste em receber de alguma forma, por parte do consumidor, uma autorização para o envio de mensagens de cunho comercial.

Isto pode ser feito de várias formas, seja no momento do cadastro do cliente em uma loja, por telefone, por carta convencional, ou até mesmo estimulando-o, pelos veículos de comunicação de massa, como televisão, jornal e rádio, a visitar determinado website e fazer o seu cadastro, autorizando o envio de mensagens. A partir do momento em que uma empresa possui um banco de dados com clientes que deram a permissão, ela terá um ativo valorosíssimo, pois terá a demanda de cliente sob seu controle, oferecendo produtos e serviços quando o cliente desejar, pois o mínimo que se espera por parte da empresa é que, durante este relacionamento, ela consiga dados como a data de nascimento, as preferências pessoais, o tamanho da família, e características comportamentais que possam sugerir ofertas especiais e individuais a esses clientes. O marketing de permissão representa, sem dúvida, a principal ferramenta de web marketing atualmente.

Conveniência ao Consumidor

Enfrentar filas no banco para realizar um saque, depósito ou pagamento é coisa do passado” para muitos consumidores que possuem acesso à internet. A conveniência nesse campo, trazida pela internet, é surpreendente. Os pais podem

acompanhar o boletim do aluno on-line, assim como o aluno pode fazer seus testes e simulações em casa em websites educacionais. Já existe a possibilidade de fazer pós-graduação de forma semipresencial, ou seja, boa parte do curso é realizada pela internet. Ainda é cedo para contabilizar as mudanças ocorridas em todo este processo, mas tem-se uma certeza: o consumidor está tendo cada vez mais facilidade na busca da informação, e a empresa que não viabilizar este processo terá sua sobrevivência organizacional ameaçada.

Customização: Palavra de Ordem em Web Marketing

Você já comprou um CD de música por causa "daquela"música especial? Normalmente a resposta é sim, e de quebra levou outras músicas porque não era possível comprar somente a música desejada. A internet está possibilitando, a cada dia, a personalização de produtos e serviços aos consumidores. Já é possível comprar pela internet roupas em que especificamos as nossas medidas e o modelo desejado e, em alguns dias, recebemos em casa uma calça, por exemplo, feita de modo exclusivo. Pode-se comprar veículos pela internet escolhendo a cor, os opcionais e o modelo, o que de certa forma é um processo de compra personalizada. Pode-se visitar websites de notícias, em que selecionamos os assuntos que queremos receber diariamente em nossa caixa postal, criando, deste modo, um jornal personalizado. Num mundo onde os produtos estão se tornando cada vez mais parecidos no aspecto técnico, à solução mais viável e eficiente para a diferenciação no mercado é, certamente, a personalização de produtos e serviços.

O Novo Mundo da Concorrência

Se alguém lhe perguntar quem é a sua concorrência, a resposta mais simples e comum encontrada nos livros de marketing tradicional aponta que são todos aqueles que oferecem produtos e serviços similares ao seu, a preços também similares. Com o web marketing é preciso esquecer esta ideia.

Todos conhecem a megaempresa Sony, fabricante do videogame Playstation. Muitos podem imaginar que o concorrente deste videogame sejam os produtos simulares do seu segmento, como videogames da Nintendo e Sega. Não

são. Para desenvolver um videogame são necessários tempo e muito dinheiro. Tudo começa com pesquisa de mercado, tanto para mensurar a demanda de pessoas que desejam comprar um equipamento de entretenimento, como para buscar fornecedores e tecnologia. Após este longo e oneroso processo vem o desenvolvimento do produto, envolvendo milhares de colaboradores, como pesquisadores científicos e engenheiros. A seguir tem início a produção em escala, partindo para ações de marketing, buscando representantes no mundo inteiro, criando programas de comunicação global e uma infinidade de ações comuns ao lançamento de um produto deste porte.

Quem será, então, concorrente deste complexo processo produtivo? Dois meninos de 16 anos. Estes pequenos jovens utilizaram a força da disponibilização de informação da internet para criar um software que, ao ser executado, transforma o seu computador em um potente e maravilhoso videogame Playstation. Esses meninos vendiam o software por US$ 15,00, e por muitas vezes este era disponibilizado de graça na internet. O preço de um videogame Playstation é aproximadamente U$ 130,00. Para criar um software que transforme o seu computador em Playstation esses garotos não precisaram gastar sequer 0,5% do que a Sony gastou para desenvolvê-lo. Sabemos que isto é pirataria também. A Sony processou os dois meninos, que foram obrigados a fechar o seu website. Mas é preciso reconhecer que esses meninos, além de piratas, são gênios. O que se aprende nesta história é o seguinte: não basta ter milhões de dólares para conquistar o mundo, quando dois jovens de 16 anos podem fazer a mesma coisa, sem despender um tostão.

Equipamentos como DVDs e videoquês e ligações internacionais pelo telefone convencional já se tornaram vítimas da concorrência da internet, pois seus produtos estão sendo substituídos de forma rápida por soluções digitais.

Redução de Custos

Qualquer empresa pode utilizar ações de web marketing para obter redução dos seus custos. O envio de documentos pelo correio convencional está diminuindo de forma considerável, pois o e-mail pode realizar este processo com sucesso na maioria das vezes. O departamento de compras alcança muito mais agilidade e

velocidade na aquisição de produtos e serviços, pois os orçamentos são enviados por e-mail e, em alguns websites, o fornecedor cria o orçamento de forma on-line. A contratação de funcionários, por sua vez, exigia uma energia significativa por parte do departamento de recursos humanos. Hoje existem empresas de altíssimo nível que podem fazer este serviço de busca de profissionais de forma rápida e com muita qualidade. A própria criação de material de comunicação acaba tendo uma redução significativa, pois, além de cara, demanda despesas de envio. A criação de uma página na internet para demonstrar produtos e serviços aos seus atuais e futuros clientes, além de proporcionar alta qualidade de imagens, não impõe limitação de tamanho, cor e formato, podendo-se, inclusive, inserir fotos, sons e videos.

Um Consumidor Diferente, Exigente, porém Atraente.

Hoje, no Brasil, existem aproximadamente 19 milhões de usuários de internet. Comparativamente com a população brasileira, este número não é muito significativo. Mas, é preciso ter cuidado com esta visão equivocada, pois são alguns milhões de brasileiros que possuem capacidade de compra real, inseridos na sua maioria entre as classes média e alta, com alto nível de escolaridade. Para alguns produtos o uso da internet como ferramenta de venda pode ser inútil, mas para outros segmentos pode significar sua sobrevivência. O consumidor da internet é diferente, possui um comportamento nada parecido com o do comprador convencional. O consumidor da internet é exigente, não aprecia demora, seja para visualizar uma página ou para receber um produto, quando decide reclamar não poupa palavras, pesquisa muito antes de comprar, busca normalmente o preço mais barato, enfim, é um consumidor extremamente ligado nos seus direitos. Para muitos este consumidor normalmente recebe o estereótipo de chato, mas garanto a vocês que, pessoalmente, queria muitos destes chatos comprando no meu website.

12 – MARKETING E ÉTICA

O marketing ainda é uma área da administração pouco usada nos negócios, no Brasil, no que se refere a seu amplo e complexo escopo de aplicação. Existem na

realidade ações isoladas de algumas das ferramentas, em especial as comunicacionais, propaganda e promoção de vendas. Em parte explica-se tal procedimento porque é mais fácil o entendimento do marketing como filosofia, do que da aplicação de suas técnicas, que em muitas situações envolvem áreas distintas do saber.

Apesar de ainda negligenciada no Brasil, a ética no marketing, em função do amadurecimento e consciência da nossa sociedade, está cada vez mais sendo incorporadas nas práticas empresariais. Ela transcende a uma área específica da empresa e permeia toda a organização, o negócio, estando assim incorporada à ética dos negócios.

Devemos ser conscientes que em uma sociedade exigente e sabedora de seus direitos e deveres, ela cobrará cada vez mais uma postura ética nos negócios e, como consequência, do marketing.

Ações não éticas nos remeterá a julgamento social. Assim, ética nos negócios e no marketing deve ser objeto de estudos acadêmicos, profundos e constantes.

Conheça, na íntegra, o Código Brasileiro de Auto-regulamentação do MARKETING DIRETO, aprovado dia 12 de março de 1997.

Código de Ética da ABEMD

Código Brasileiro de Auto-regulamentação do MARKETING DIRETO

O Código de Ética da ABEMD - Associação Brasileira de Marketing Direto - constitui a auto-regulamentação do setor de Marketing Direto no Brasil, entendendo-se por Marketing Direto um sistema interativo que usa uma ou mais mídias para obter uma resposta ou transação mensurável em algum lugar.

As presentes normas de auto-regulamentação têm por objetivo prover os consumidores e as organizações que atuam em Marketing Direto de princípios de conduta de aceitação nacional e internacional.

Este Código reflete a política da ABEMD de manter um nível de conduta que prioriza a responsabilidade, tanto da entidade quanto dos agentes de Marketing Direto, e de estabelecer com o consumidor relações que tenham por base princípios éticos e justos.

Todos são chamados a apoiar este Código de fato, levando em conta seus princípios e objetivos, e a não considerar suas normas meros obstáculos passíveis de serem contornados por manobras legais.

Por considerar que práticas comerciais abusivas e toda comunicação fraudulenta, enganosa, imoral ou ofensiva são sempre nocivas à publicidade, inclusive no setor de Marketing Direto, a ABEMD determina a todos os seus associados à observância deste Código.

Capítulo 1 - OS TERMOS DA OFERTA

Honestidade

1.1. As ofertas devem ser claras, precisas e completas, para que o consumidor possa saber a natureza exata do que é oferecido, o preço, eventuais taxas extras, as condições de pagamento e as obrigações nas quais incorre ao fazer a solicitação.

1.2. Antes de divulgar uma oferta, os agentes de Marketing Direto devem estar preparados para comprovar todas as afirmações anunciadas. São vedadas publicidades ou alegações inverídicas, enganosas, fraudulentas ou que depreciem outras empresas injustificadamente.

Clareza

1.3. Os materiais de comunicação devem incluir uma descrição clara dos pontos essenciais da oferta e do produto ou serviço. Deve-se deixar claro quando uma oferta mencionar bens que não estão incluídos ou que têm custo extra.

1.4. Não poderá ser utilizado material de comunicação que, por suas dimensões reduzidas, localização ou outra característica visual, possa afetar a clareza da oferta ou de exceções a ela.

1.5. As fotografias, ilustrações e situações que representam devem constituir reproduções fieis dos produtos ou serviços relacionados no material de comunicação.

Condições objetivas

1.6. Todas as descrições e garantias devem estar de acordo com as condições, situações e circunstâncias vigentes no período da oferta. Referências a quaisquer limitações (tais como tempo e quantidade) devem estar expressas no material de comunicação e serem cumpridas.

Discriminação

1.7. É inaceitável a discriminação de qualquer pessoa ou grupo com base em raça, cor, religião, nacionalidade, sexo, idade ou estado civil.
Padrões morais

1.8. As propostas não poderão incluir material ofensivo ou inaceitável para publicidade em termos morais.

Publicidade para crianças

1.9. Ofertas próprias apenas para adultos não poderão ser dirigidas a crianças.

Identificação e objetivo

1.10. Em todos os contatos de Marketing Direto devem ser explicitados os seus objetivos e o nome da empresa, instituição ou organização responsável. Ninguém

poderá fazer ofertas ou solicitações sob pretexto de pesquisa ou similar quando o verdadeiro objetivo for à venda.

1.11. As ofertas e remessas devem apresentar com clareza o nome e o endereço ou telefone do agente de Marketing Direto, para que o consumidor possa entrar em contato.

Proposta em vez de fatura

1.12. Não podem ser usadas propostas que possam ser confundidas com notas, faturas ou outros documentos de cobrança.

Taxa de entrega

1.13. Quando houver cobrança de taxa de entrega, esta deverá ser compatível com os preços vigentes.

Uso da palavra "grátis" e termos semelhantes

1.14. Qualquer produto ou serviço oferecido ao consumidor sem custos ou obrigações pode ser apresentado como “grátis”. Quando um produto ou serviço for oferecido gratuitamente e/ou quando a oferta exigir que o consumidor compre outro produto ou serviço, todos os termos e condições devem ser apresentados junto à palavra “grátis” ou ao termo semelhante.
Sistemas de venda

1.15. Ofertas de opção negativa - aquelas que exigem uma manifestação de não aceitação pelo consumidor - são proibidas pelo Código de Defesa do Consumidor

1.16. Programas de fornecimento automático e continuado de produtos ou serviços deverão obter do consumidor a prévia anuência dos termos iniciais e condições de cancelamento.

Propaganda comparativa

1.17. O material de comunicação que fizer comparação com produtos e serviços da concorrência deverá respeitar os seguintes princípios e limites:

O objetivo maior da comparação deverá ser o esclarecimento ou a defesa do consumidor;

A comparação deverá ter por princípio a objetividade. Dados subjetivos, de fundo psicológico ou emocional não constituem uma base válida de comparação perante o consumidor;

A comparação deverá ser passível de comprovação;

A comparação de preços deverá citar a fonte;

A comparação não poderá ser feita entre produtos ou serviços de épocas diferentes, a menos que se trate de referência para demonstrar evolução entre bens de consumo, o que, nesse caso, deve ser caracterizado;

A comparação não poderá estabelecer confusão entre produtos e marcas concorrentes, caracterizar-se como ato de concorrência desleal ou denegrir a imagem do produto, serviço ou marca de outra empresa.

Garantias

1.18. Quando um produto ou serviço for fornecido com uma garantia (incluindo assistência e suporte técnico), o material de comunicação deverá relacionar seus termos e condições ou informar como o consumidor poderá conhecê-los.

Uso de teste ou pesquisa na comunicação

1.19. Pesquisas ou testes utilizados em material de comunicação devem ter sua fonte e metodologia citadas e provar a afirmação específica a que se referem. O material de comunicação não pode distorcer resultados de testes ou pesquisas nem utilizá-los fora de contexto.

Depoimentos e declarações

1.20. Depoimentos e declarações somente podem ser utilizados quando:

Autorizados pela pessoa citada;

Verdadeiros e relacionados à experiência da pessoa em questão;

Não tomados fora de contexto de forma a distorcer a opinião ou experiência da pessoa com o produto ou serviço.

Capítulo 2 – ATENDIMENTO

Segurança

2.1. Os produtos e serviços devem atender à legislação vigente. As informações fornecidas com o produto devem incluir orientação correta para o uso e instruções completas, inclusive advertência sobre montagem e segurança, sempre que necessário.

Disponibilidade

2.2. Os agentes de Marketing Direto só podem oferecer produtos e serviços quando estes estiverem disponíveis ou quando houver previsão de entrega ou fornecimento.
Prazo de entrega

2.3. Quando o material de comunicação não especificar o prazo de entrega de produtos ou serviços, fica o mesmo estabelecido em no máximo 30 (trinta) dias a

contar da data do recebimento do pedido. Em todos os casos, os agentes de Marketing Direto devem atender aos pedidos com a maior rapidez.

Igual oportunidade de crédito

2.4. Os agentes de Marketing Direto que oferecerem em suas campanhas opção de pagamento parcelado deve apresentar formalmente as razões da recusa de um pedido de crédito, sempre que solicitados pelo consumidor.

Capítulo 3 – LISTAS

3.1. Lista é um conjunto de informações individuais sobre consumidores - pessoas físicas ou jurídicas - utilizado nas atividades profissionais de Marketing Direto, e assim deverá ser definida sempre que divulgada.

Respeito à privacidade do consumidor

3.2. Os agentes de Marketing Direto que abrem a outras empresas a possibilidade de utilizar suas listas de clientes para ações de Marketing Direto devem informar tal prática ao consumidor no momento em que seus dados estiverem sendo coletados.

3.3. Os agentes de Marketing Direto devem oferecer ao consumidor a opção de ter seu nome suprimido de sua lista e providenciar a supressão para os que assim o desejarem.

3.4. Os agentes de Marketing Direto devem restringir a coleta de informações sobre o consumidor e seu compartilhamento com outras empresas aos dados adequados para fins de marketing.

3.5. Critérios de seleção e informações que possam ser consideradas de natureza pessoal ou íntima não poderão ser compartilhadas com outras empresas quando o consumidor acreditar que tais dados serão mantidos em sigilo.

Acordos para uso de listas

3.6. Os agentes de Marketing Direto devem certificar-se da fonte e dos métodos de formação da lista antes de utilizá-la ou tornarem-se seu corretor.

3.7. Os proprietários, responsáveis pela compilação, e corretores de listas, não poderão permitir o compartilhamento de suas listas para ofertas que violem quaisquer das normas deste Código.

3.8. As ofertas devem ser dirigidas para os segmentos do público que mais provavelmente estarão interessados ou que sejam usuários em potencial dos produtos ou serviços oferecidos.

Respeito à propriedade das listas

3.9. Nenhuma lista ou informação nela contida poderá ser usada em violação aos direitos legais do proprietário da lista ou ao acordo entre as partes. Qualquer abuso deverá ser levado ao conhecimento do proprietário legal.

Capítulo 4 – TELEMARKETING

4.1. Nas abordagens de telemarketing e fechamento de negócios através deste canal, o agente de Marketing Direto deverá observar as normas deste Código sobre Oferta, Atendimento e Listas.

Infraestrutura de atendimento receptivo

4.2. Antes de divulgar um número telefônico para atendimento receptivo, o agente de Marketing Direto deverá tomar os cuidados necessários para dimensionar a infraestrutura adequada à demanda estimulada. Horários de chamadas

4.3. Nas ações de telemarketing ativo, aquelas em que o agente de Marketing Direto realiza a chamada telefônica, o contato somente poderá ser feito de segunda a

sexta-feira das 8:00 às 21:30 horas, e aos sábados das 9:00 às 18:00 horas. Aos domingos e feriados não poderá ser realizada nenhuma forma de telemarketing ativo.

Comunicação

4.4. Táticas de coerção ou constrangimento não devem ser usadas de forma alguma.

4.5. O agente de Marketing Direto não poderá realizar chamadas a cobrar para os consumidores, salvo com seu prévio consentimento.

Uso de equipamento automático

4.6. Os equipamentos de discagem automática somente poderão ser utilizados se liberarem a linha telefônica no momento em que a pessoa chamada desligar.

4.7. Toda mensagem gravada deverá ser precedida pelo contato de um operador solicitando autorização para transmissão. É dispensada a intervenção de um operador somente nos casos de mensagens de utilidade pública (campanhas de vacinação, por exemplo).

Gravação de conversas telefônicas

4.8. A gravação de conversa telefônica somente poderá ser realizada com o consentimento do interlocutor.

4.9. A gravação de conversa telefônica, sempre sob consentimento do interlocutor, é recomendada para registro do fechamento de negócios e para o treinamento de operadores.

Abordagem de crianças

4.10. É vedada a formulação de ofertas por telefone e o fechamento de vendas junto a crianças

Veja a íntegra do código no menu Código de Ética de Call Center/Telemarketing

Capítulo 5 - ARRECADAÇÃO DE FUNDOS

Veto à participação nos resultados

5.1. Nas ações de arrecadação de fundos e doações para instituições sem fins lucrativos, os indivíduos, organizações ou empresas que delas participem não poderão ser remunerados de forma proporcional aos recursos captados.

Autenticidade da instituição

5.2. Não poderá ser feita arrecadação de fundos e doações para instituições sem fins lucrativos que não estejam em funcionamento.

Capítulo 6 - RESPEITO À LEGISLAÇÃO

6.1. Os agentes de Marketing Direto devem atuar de acordo com o Código de Defesa do Consumidor e obedecer à legislação vigente sobre publicidade, práticas de marketing e comércio.

(Data de Aprovação: 12/03/97)

13 – TÉCNICAS DE APRESENTAÇÃO

Falar em público é uma habilidade essencial em qualquer organização e fundamental para partilhar o conhecimento adquirido nas áreas específicas de trabalho que cada um ocupa. A comunicação precisa acontecer de maneira clara, objetiva e envolvente. Essas habilidades podem ser desenvolvidas com técnicas específicas e com a prática bem fundamentada.

Além de uma boa desenvoltura ao falar para o público, para transmitir conhecimento é necessário desenvolver técnicas de didática do ensino. Para ensinar não basta ser um especialista no assunto, é preciso saber ouvir, entender as necessidades do seu público, planejar, executar um plano de ensino e avaliar de forma continuada todo o processo. A maneira como executar um plano de ensino também pode ser aprendida e aprimorada com a prática.

Comunicação

Conceito

Comunicação provém do latim communis, que significa “tornar comum”. A comunicação pode ser definida como a transmissão de informações e compreensão mediante o uso de símbolos comuns (verbais e não verbais). Sendo assim, a comunicação não significa apenas enviar uma informação ou mensagem, mas torná-la comum entre as pessoas envolvidas. Para que haja comunicação, é preciso que o destinatário da informação a receba e a compreenda. Comunicar significa tornar comum a uma ou mais pessoas determinada informação ou mensagem (CHIAVENATTO, 2005).

As relações sociais estão sujeitas à influência de um conjunto de variáveis que determinam, ou pelo menos influenciam a condução dos processos de comunicação.

Comunicar é uma arte de bem enviar e receber mensagens. O tempo, espaço, o meio físico envolvente, o clima relacional, o corpo, os fatores históricos da vida pessoal e social de cada indivíduo, as expectativas condicionam e determinam a maneira de se relacionar dos seres humanos. É, portanto, importante se conhecer

alguns dos fatores que podem constituir-se como barreiras à comunicação e fontes no ruído na comunicação, para evitá-los e assim realizar a comunicação de forma mais clara e efetiva.

Para que haja comunicação, é preciso que o destinatário da informação a receba e a compreenda.

Elementos da Comunicação

A comunicação pode ampliar ou limitar conhecimentos, facilitar ou dificultar o desenvolvimento de criatividade e de habilidades inerentes ao melhor desempenho do aluno.

Por meio da comunicação, instrutores e alunos explicitam objetivos, revelam poderes, valores e culturas que norteiam as relações profissionais e interpessoais no ambiente de aprendizagem.

Deve-se, portanto, administrar o processo global de comunicação, que necessariamente abrange os seguintes elementos:

_ Fonte: é o emissor, a pessoa que está tentando enviar uma mensagem (falada, escrita, por meio de sinais ou não verbal) a uma pessoa ou pessoas, por meio da codificação do pensamento;

_ Codificação: é a tradução dos símbolos escolhidos pela fonte para que a mensagem possa ser adequadamente transmitida pelo canal;

_ Mensagem: é o produto físico codificado pelo emissor, e pode ser a fala, o texto escrito, um quadro, uma música, os movimentos de nossos braços e expressões faciais;

_ Canal: é o meio que existe fora do emissor pelo qual é escolhido para conduzir a mensagem. Esse veículo pode ser o discurso oral (audição), a documentação escrita (visão e tato) e a comunicação não verbal (sentidos básicos);

_ Decodificação: é a tradução da mensagem, para que a mesma possa ser compreendida pelo receptor;

_ Receptor: é o sujeito a quem a mensagem se dirige, o destino final da comunicação. Também é chamado de destinatário;

_ Feedback: é o elo final no processo de comunicação. O feedback – ou retorno – determina se a compreensão foi alcançada ou não e,

_ Ruídos: perturbação indesejável que tende a deturpar, distorcer ou alterar, de maneira imprevisível, a mensagem transmitida.

Aspectos de maior impacto na comunicação

Emissor e transmissor são papéis assumidos em uma verdadeira interação dos três tipos de comunicação:

A verbal (por meio da escrita e da fala); a não verbal (as fisionomias, os gestos); e a factual (quando se considera a prática, a habilidade, a experiência passada, o exemplo dado, que levam o profissional a ter fama positiva ou negativa). Então, se quisermos saber o significado da comunicação, devemos observar a reação obtida com os sinais que emitimos, pois quando nos comunicamos não é somente o que falamos que está em jogo, mas a maneira como falamos também é muito representativa e significativa no processo de comunicação.

Estudos mostram que do que se aprende apenas 20% é da mensagem verbal, o resto é não verbal e factual. Segundo O´Connor e Seymour, “as palavras são o conteúdo da mensagem, e a postura, os gestos, a expressão e o tom de voz são o contexto no qual a mensagem está embutida. Juntos eles formam o significado da comunicação” (p. 35, 1995). Ainda para estes autores, a comunicação corporal corresponde a 55% do total das formas como nos comunicamos, as palavras correspondem a 7% e o tom de voz 38%.

Comunicação verbal, não verbal e factual

Na comunicação verbal devem-se levar em conta os seguintes pontos: grau de domínio do assunto; vocabulário ao nível do interlocutor; pontuação clara e variação; articulação de ideias; fluência e ritmo (altos e baixos); uso de audiovisuais.

Na comunicação não verbal alguns cuidados a serem observados: mobilidade da cabeça e rosto; olhar (direcionado a todos, ou a alguns); gestos enriquecedores; voz graduada ao ambiente; respiração e desinibição; postura corporal; andar; roupas; cores; penteados; adornos.

Na comunicação factual existem alguns pontos a considerar: coordenação e domínio das atividades pedagógicas; uso do espaço e administração do tempo;

administração de conflitos em sala de aula; ser exemplo de ação (coerência entre teoria e prática); ser decidido e prático nas propostas de ação; saber usar os recursos instrucionais.

Falar em público – Necessidade e importância

Não há como fugir. O especialista Reinaldo Polito (2006) salienta que qualquer que seja a atividade, daqui em diante você estará sempre 'correndo o risco' de falar em público.

A habilidade em técnicas de comunicação e apresentações passou a ser competência importante para profissionais de todas as atividades. Falar em público seja em pequenas reuniões, diante de poucas pessoas, seja em grandes eventos, diante de plateias numerosas, tornou-se tarefa quase corriqueira, independentemente da posição hierárquica ocupada pelo profissional na organização.

Embora as pessoas estejam frequentemente às voltas com apresentações em público, poucas estão bem preparadas para desempenhar a tarefa com eficiência. Todos os profissionais, por maior que seja sua dificuldade para se apresentar em público, podem, em poucas horas de capacitação criteriosa, expressar-se com eficiência nas mais diversas circunstâncias, diante de qualquer tipo de plateia.

Técnicas de Apresentação e Comunicação

Técnica pode ser definida como maneira, jeito ou habilidade especial de executar ou fazer algo. Para que o instrutor de capacitação possa se comunicar com sucesso existem maneiras apropriadas para tal. As técnicas de apresentação podem ser aprendidas e praticadas visando ao aprimoramento das habilidades de comunicação fazendo assim com que o instrutor possa ter sucesso em suas aulas, palestras, oficinas e outros eventos de capacitação. A seguir, uma série de dicas importantes para o seu aperfeiçoamento:

Respiração – tranquilize-se

O primeiro passo para uma boa apresentação é estar tranquilo. A respiração pode ser uma aliada fundamental para o instrutor. Para controlar a ansiedade, antes de iniciar a apresentação é importante fazer a respiração abdominal – pelo menos três vezes – com o objetivo de oxigenar o cérebro.

Estar calmo perante a plateia faz com que o instrutor transmita segurança a quem assiste à palestra. Durante a apresentação é importante manter uma respiração pausada para que não se atropelem as palavras e assim torne sua aula ininteligível.

A respiração deve ser utilizada nos momentos de tensão ou ansiedade que antecedem uma apresentação ou aula, mas que com o tempo e a prática pode se tornar um processo natural do instrutor.

Posicionamento – o que fazer e o que não fazer

O instrutor deverá encontrar um ponto central que servirá como local de referência. Em uma sala de aula, pode ser entre o quadro e as cadeiras dos alunos. Em um auditório, pode ser num ponto mais próximo à plateia. O importante é que haja mobilidade por parte do instrutor, mas sempre com objetivo e alguma finalidade, por exemplo, se você se desloca de um lado para o outro da plateia para dar ênfase à determinada informação, ou para recuperar a atenção de pessoas que começam a se desconcentrar, estará agindo de maneira correta. Se, todavia, os movimentos ocorrerem apenas porque não se sente confortável, e você se desloca apenas para tentar se sentir mais à vontade, é quase certo que esteja se comportando de forma inadequada.

Não fique de costas para o público, mesmo que para mostrar algo que está na sua apresentação, ao ficar de costas sua voz se perderá e seus ouvintes não o entenderão. Se quiser apontar algo na apresentação, use uma régua, laser pointer ou mostre primeiro e explique depois.

Nunca se posicione na frente da luz do data show, pois isso pode ser um grande incômodo para quem está na plateia tentando fazer anotações ou mesmo

prestar atenção ao que está sendo projetado. Caminhe nas laterais do data show, de maneira a não atravessar na sua frente.

Evite fazer a sua apresentação sentado, a menos que seja estritamente necessário. Permanecer sentado tornará sua apresentação cansativa e monótona para a plateia.

Postura e expressão corporal

De acordo com Polito (2006) há dois erros comuns que precisam ser evitados na gesticulação. O primeiro deles é a ausência de gestos, pois os movimentos do corpo são importantes no processo de comunicação da mensagem. O outro, geralmente mais grave do que o primeiro é o exagero de gestos. Provavelmente o resultado da sua apresentação será muito mais positivo se você for comedido nos gestos do que se você exagerar na gesticulação.

As pernas

Procure se posicionar com equilíbrio sobre as duas pernas, evitando ficar apoiado ora sobre a perna esquerda, ora sobre a direita observe que elas não fiquem demasiadamente abertas, para que não tornem sua postura deselegante, ou, ao contrário, que não permaneçam muito juntas, para que sua mobilidade não seja prejudicada.

Os braços e as mãos

Não carregue caneta, papel e outros objetos, pois eles podem muito mais atrapalhar do que ajudar. Se você estiver visivelmente nervoso, tremerá a mão ao segurar um papel ou o laser, e isso ficará claro para a plateia. Deixe suas anotações sobre a mesa e verifique-as quando necessário.

Laser pointer

É um ótimo instrumento para apontar dados na apresentação. O seu uso só deve ser feito pelo instrutor que possui a mão firme, porque do contrário, o *laser* estará trêmulo e não servirá para o seu propósito que é indicar alguma informação, além de deixar claro para os participantes que você está nervoso.

Gestos

Ao gesticular, é recomendável fazer os movimentos acima da linha da cintura e alternar a posição de apoio dos braços, deixando-os às vezes ao longo do corpo, em outros momentos gesticulando apenas com um deles, para em seguida atuar com os dois. Dessa forma evitará a rigidez da postura e tornará os gestos mais elegantes e harmoniosos.

Evite cruzar os braços, posicioná-los nas costas ou colocar as mãos nos bolsos, procure fazer os repousos de mãos a frente ou deixar os braços soltos. Se agir assim com frequência e por tempo prolongado, talvez demonstre desnecessariamente aos ouvintes que está inseguro e intranquilo.

Evite também esfregar ou apertar nervosamente as mãos, principalmente no princípio, quando essas atitudes são mais comuns. Tome cuidado ainda para não coçar a cabeça o tempo todo, ou ficar segurando sucessivamente a gola da blusa ou do paletó. Quase sempre os ouvintes têm a impressão de que, quando o orador domina o assunto de sua apresentação, não fica nervoso diante da plateia.

Portanto, mesmo que essa interpretação do público não seja verdadeira, essas demonstrações de hesitação podem comprometer o resultado de sua exposição.

Olhar para a plateia

Olhar na direção das pessoas fará com que elas se sintam incluídas no ambiente, por isso não esquive o seu olhar. Caso tenha dificuldades em olhar para seu público, você pode utilizar a Técnica da Triangulação. Escolha três pontos no auditório ou sala de aula, três rostos familiares ou três pessoas que lhe deem

feedback positivo durante sua fala. Olhe para a primeira pessoa do lado esquerdo, vagarosamente olhe para a pessoa no centro, e vá movendo seu olhar para a pessoa no lado direito. Dessa maneira você estará percorrendo o auditório ou sala de aula com seu olhar e de maneira segura. Quando estiver se sentindo mais a vontade, olhará para todos e verá muitos rostos interessados no seu assunto.

A voz

É importante diferenciar o tom da sua voz para lugares grandes e pequenos. Você deverá falar mais alto ou mais baixo dependendo da acústica do local, da distância em que se encontrar do último ouvinte da plateia, dos ruídos que possam interferir na compreensão do público.

Evite falar alto quando estiver próximo de poucas pessoas em uma pequena reunião, e utilize o microfone diante de plateias numerosas.

Caso você tenha de usar o microfone fique atento para a sensibilidade do mesmo. A boca deve ficar distante do microfone aproximadamente dez centímetros. Faça o teste antes da palestra para saber se precisa afastar ou aproximar o microfone para que a voz seja bem captada.

O melhor momento para fazer a dosagem da voz é quando você realiza a sua apresentação pessoal, que deve ser feita sempre no início da palestra.

Lembre-se que esse é um momento muito importante da sua fala, a apresentação pessoal deve sempre ser objetiva, não é adequado alongar-se muito nessa hora, mas a fala deve conter as informações essenciais sobre o trabalho que você desempenha e sua formação.

Repetições

Cuidado com repetições de palavras como: né? Ok? Tá? E outras. Esses recursos costumam ser utilizados quando o instrutor quer enfatizar o assunto ou buscar aprovação da plateia. Evite-os, pois em excesso servem para deixar a fala caricata e desagradável de ouvir. Preste atenção ao seu próprio discurso e perceba se existe um chavão ou palavra que costuma repetir. Experimente construir suas frases sem eles e ouça como irá soar de maneira mais agradável.

Pessoas que falam devagar tendem a preencher os espaços de tempo com sons do tipo: ããããã ou ééééé. Fique atento para esse fato e prefira o silêncio entre as frases. Uma técnica utilizada para evitar esse tipo de som é iniciar imediatamente a frase seguinte.

Ritmo

Depois de certo tempo de duração – entre 20 e 30 minutos de palestra ou aula os ouvintes começam a ter sua atenção desviada. Se o palestrante utilizar uma fala monótona e monocórdia se tornará impossível prender a atenção no assunto. Procure mudar sua entonação quando se tratar de um tópico importante, frise com a sua voz aquilo que todos precisam prestar atenção.

Vocabulário

O seu vocabulário deverá estar de acordo com o público alvo da aula ou palestra. No caso do uso das siglas, utilize-as somente se você tiver certeza que seu público está familiarizado com elas. Caso você esteja querendo ensinar conceitos ou utilizar as siglas para abreviar nomes ou expressões muito longas, explique primeiro o que cada sigla significa para depois começar a falar sobre elas.

O vocabulário técnico só deve ser utilizado também para o público que está familiarizado com ele. As regras são as mesmas utilizadas para as siglas. Muitos palestrantes gostam de demonstrar seu conhecimento com vocabulário rebuscado ou com o uso de estrangeirismos. Se esse tipo de fala faz parte da cultura organizacional sua fala estará contextualizada. Caso não faça, sua fala não será compreendida e logo a plateia irá perder o interesse no que você está dizendo.

Domínio do tema

De nada adiantará falar com naturalidade, envolvimento e disposição se não dominar o assunto a ser apresentado. Por isso, estude o assunto com a maior profundidade que puder. Saiba muito mais do que precisará para expor. Deixe sobrar informações. Primeiro, porque esse conhecimento lhe dará mais confiança

diante do público. Depois, porque demonstração de domínio da matéria poderá dar mais credibilidade à sua apresentação.

Caso um de seus alunos ou ouvintes da plateia faça uma pergunta cujo conteúdo você não domina ou você não tenha certeza sobre a resposta, fique calmo diga simplesmente “Não sei a resposta no momento, mas vou pesquisar e trarei a informação amanhã”. Pesquise o tema e de fato traga a resposta no dia seguinte.

Linguagem fática

Linguagem fática, ou função fática da linguagem está “centralizada no canal, tendo como objetivo prolongar ou não o contato com o receptor, ou testar a eficiência do canal” (BECHARA, 2000). É o tipo de linguagem das falas telefônicas, saudações e similares. Numa situação de sala de aula ou palestra é interessante que o instrutor faça uso da linguagem fática como recurso para verificar se o canal de comunicação está funcionando ou se ele está sendo compreendido pelas pessoas que o ouvem.

A seguir alguns exemplos da função fática: (NICOLA, 2004) Ex: “entenderam?”.

É também a função empregada, quando no decorrer de uma conversa, emitimos sons:

Ex: “heim”!”“, “hum hum”!”.

E além de ser utilizada para testar o canal, a função fática ocorre também quando o emissor quer iniciar uma comunicação:

Ex: - olá! Como vai? – perguntou o homem de olhos empapuçados

Ou quando visa prolongar o contato com seu receptor:

Ex: - ela não desanima nunca... Você concorda? Não acha?

Ou quando deseja interromper o ato de comunicação:

Ex: - qualquer dia, amigo, eu volto a te encontrar.

E em alguns casos, percebe- se que a única preocupação do emissor é manter o contato com o destinatário ou testando o canal com frases do tipo:

“veja bem” ou “olha...” ou “compreende?”.

Essa preocupação com o contato caracteriza- se função fática.

Conquistando o público

Conquistar o público é uma habilidade que pode ser desenvolvida, principalmente se você iniciar sua palestra utilizando uma frase que possa provocar impacto. Sua frase deve estar relacionada ao assunto tratado e faça isso de tal forma que as pessoas percebam, pela força da informação que a mensagem é importante.

Dicas para falar bem

Você já passou pelas técnicas mais utilizadas para realizar uma apresentação eficaz, agora sugerimos algumas dicas para que você possa falar com desenvoltura e naturalidade sobre os assuntos que deseja abordar.

Continue sendo você mesmo

Parece ironia, mas somente depois que estiver muito bem preparado para falar em público você conseguirá se expressar diante da plateia mantendo seu estilo e suas características. E quanto mais espontâneo for, melhores serão os resultados de suas exposições.

O especialista Reinaldo Polito (2006) recomenda que ao se apresentar diante de um grupo de pessoas, reflita se está conseguindo se comunicar da mesma maneira como faria se estivesse conversando à vontade com amigos, familiares ou colegas de trabalho. Se a resposta for negativa, significa que provavelmente sua maneira de falar está artificial. Quando tiver condições de levar para frente do público a forma natural como se comunica com as pessoas de seu relacionamento mais próximo, terá resolvido uma importante etapa de seu desenvolvimento como comunicador.

Por isso, sempre que tiver que fazer uma apresentação, procure conversar bastante sobre o assunto antes de enfrentar o público. Lembre-se de que não basta apenas preparar por escrito o que pretende comunicar. Há muita diferença entre a comunicação escrita e a comunicação oral.

O ritmo, a cadência, as pausas, o vocabulário e tantos outros aspectos da comunicação se mostram distintos na fala e na escrita. Se você pretende falar da mesma maneira como escreve, é quase certo que não fará uma boa apresentação.

Esse exercício simples de conversar com as pessoas a respeito do tema que vai expor fará com que descubra as palavras apropriadas, a velocidade certa, a entonação adequada. Depois, diante da plateia, o comportamento deverá ser praticamente o mesmo: continuar a conversa ensaiada.

Mostre interesse pelo assunto

Embora sua apresentação diante do público deva seguir o estilo conversacional, essa circunstância exigirá que fale com envolvimento e disposição. Isto é, deve ser uma conversa, mas uma conversa bem animada. Não tenha ilusões: se não demonstrar aos ouvintes que o assunto é importante para você, dificilmente fará com que as pessoas se envolvam no tema de sua apresentação. Concentre-se nessa premissa. Antes de envolver a plateia, você precisará mostrar, pelo seu entusiasmo, o interesse que tem pela matéria.

Quando conseguir conjugar esses dois aspectos relevantes da comunicação – espontaneidade e envolvimento –, se aproximará dos ouvintes com segurança e credibilidade.

Não seja prolixo

Quem assiste a uma palestra ou a uma aula deseja objetividade por parte do instrutor. Palestras muito longas ou difusas fazem com que a plateia perca o interesse no assunto. Os exemplos utilizados, as apresentações preparadas para o evento também não precisam ser excessivas.

O instrutor prolixo dá aulas muito longas e enfadonhas. Procure ser o oposto disso, criando atividades adequadas às circunstâncias e voltadas para a realidade prática do grupo.

Preparação do conteúdo

Se você não souber planejar de maneira correta uma apresentação, dificilmente conseguirá transmitir a mensagem com eficiência. Qualquer que seja o tipo de exposição de que venha a fazer precisará ter todas as etapas ordenadas de maneira lógica e concatenadas, desde o princípio até a conclusão.

O primeiro passo é identificar o objetivo. Talvez essa seja a falha mais comum do instrutor: não saber exatamente qual é o objetivo da apresentação.

Depois que você determinar o assunto a ser exposto, analise o que pretende conseguir com a mensagem. Estruture sua apresentação em introdução, desenvolvimento e conclusão. A introdução deve ser curta, com um ou dois slides, contendo conceitos principais. No desenvolvimento, esmiúce os conceitos, utilize imagens que remetam ao assunto tratado. Conclua o assunto sistematizando as principais ideias.

Estimular perguntas e saber ouvir

Perguntas e respostas fazem parte integrante de uma comunicação efetiva entre o instrutor e a plateia. A pergunta é um ótimo instrumento do instrutor, pois a partir dela é possível descobrir o que a sua plateia conhece sobre o assunto, mas é muito importante que você não confunda uma pergunta legítima, que tem o objetivo de realmente saber o que os alunos pensam com uma pergunta retórica, cujo único objetivo é dar continuidade a fala do instrutor.

Uma dica útil é iniciar a aula com perguntas abertas sobre o assunto “O que você faria diante disso?” ou “O que vocês sabem sobre o assunto?”. Ouça o que seus alunos têm a dizer sobre o assunto, faça anotações num flip chart, por exemplo, ou simplesmente utilize a fala dos próprios alunos para iniciar a sua.

Ouvir as perguntas e saber aproveitar as contribuições da plateia faz com que os alunos se envolvam muito mais no processo.

Em diversos momentos você pode pensar que alguma pergunta foi inconveniente, evite demonstrar que achou a pergunta desnecessária ou mal formulada, pois você poderá inibir futuras perguntas. Responda o que for possível ou

diga “esse assunto tratarei mais tarde” ou “esse assunto não é bem o foco da nossa palestra, posso te indicar onde encontrar uma resposta”.

E muito importante: não retarde o progresso do grupo. Quando alguns participantes não conseguem entender o assunto e o instrutor insiste, o restante do grupo se desinteressa. É uma boa prática, marcar um horário especial para explicar o assunto, detidamente, aos mais vagarosos.

Lembre-se:

1. Chegue antes dos participantes – dê exemplo, seja pontual;
2. Prepare o ambiente para receber os participantes;
3. Prepare seu material na véspera e,
4. Verifique todos os equipamentos antes da apresentação.

BIBLIOGRAFIA SUGERIDA

KOTLER, Philip. **Marketing para o século XXI**: como criar, conquistar e dominar mercados. São Paulo: Futura, 1999.

KOTLER, Philip. **Princípios de marketing**. São Paulo: Prentice Hall, 2000.

KOTLER, Philip. **Administração de marketing**: a edição do novo milênio. São Paulo: Prentice Hall, 2000.

Curso Técnico

INFORMÁTICA

Etapa 3

AUTORIDADES UTRAMIG

JOSÉ MURILO RESENDE
Presidente

FELIPE MOREIRA CARRARA
Chefe de Gabinete

MARIA LÚCIA SOARES DE MOURA
Diretora de Planejamento, Gestão e Finanças

JUSSARA SILVA NEGROMONTE
Diretora de Ensino Pesquisa

FILIPE GALGANI GOMES
Diretor de Qualificação e Extensão

VANESSA LOPES TEIXEIRA NEVES
Diretora de Ensino a Distância

DIVA HELENA SALES WARDI SILVA
Diretora da Unidade Nova Lima

JANES MARIA FERREIRA
Diretora da Unidade de Uberlândia

WENDERSON HARLLEY ASSIS CURVELLO
Gerente Responsável pelo Projeto

LÉA VICENTINA COSTA MELO
Supervisora Pedagógica Responsável pelo Projeto

EXPEDIENTE

Impressão
EDITORA AZUL

Projeto Gráfico/Diagramação
GABRIEL SEBASTIAN FLEITAS CORTIGLIA

A Fundação de Educação para o Trabalho de Minas Gerais UTRAMIG

tem a missão de formar profissionais competentes para o mercado de trabalho e promover educação em diversas áreas do conhecimento. Neste âmbito, realiza qualificação profissional, formação técnica, pós-graduação (Lato Sensu) e formação superior especial para professores. É referência na Licenciatura Plena em todo o Estado e busca estar sempre inovando e oferecendo à sociedade respostas educacionais atinentes à realidade atual, como o Ensino a Distância. A Instituição conta com uma equipe altamente qualificada, formada por doutores e mestres, apta a formar profissionais preparados para o mercado de trabalho atual.

Além da sede em Belo Horizonte, a UTRAMIG está presente em Nova Lima, São João Del Rei e Uberlândia onde são ofertados cursos técnicos em Eletrônica, Enfermagem, Informática, Análises Clínicas, Segurança do Trabalho, Telecomunicações, Meio Ambiente, Recursos Humanos e Especialização Técnica de Ensino Médio em Instrumentação Cirúrgica.

UNIDADE UTRAMIG

BELO HORIZONTE SEDE



Endereço: Avenida Afonso Pena, nº 3400
Bairro: Cruzeiro
CEP: 30.130-009
Horário de funcionamento:
07:20 as 22:40
Telefone: (31) 3263.7500
faleconosco@utramig.mg.gov.br

Cursos Oferecidos:
Eletrônica
Informática
Segurança do Trabalho
Enfermagem
Análises Clínicas
Recursos Humanos
Telecomunicações
Meio Ambiente
Especialização em Instrumentação Cirúrgica

UNIDADE UTRAMIG

NOVA LIMA



Endereço: Rua 32, n° 36

Bairro: Oswaldo Barbosa Pena
CEP: 34.000-000
Horário de funcionamento:
07:20 as 22:40
Telefone: (31) 3542.5647
faleconosco@utramig.mg.gov.br

Cursos Oferecidos:
Eletrônica
Informática
Segurança do Trabalho
Enfermagem

UNIDADE UTRAMIG

UBERLÂNDIA



Avenida Vinícius de Moraes, nº 40
Bairro Residencial Liberdade
CEP 38.401-274.
Horário de funcionamento:
07:20 as 22:40
Telefone: (34) 3210-6546
faleconosco@utramig.mg.gov.br

Cursos Oferecidos:
Informática
Segurança do Trabalho
Recursos Humanos

UNIDADE UTRAMIG

SÃO JOÃO DEL REY



Rua Doutor Elói Reis, nº 04
Bairro Matozinhos
CEP 36.300-000
Horário de funcionamento:
07:20 as 22:40
faleconosco@utramig.mg.gov.br

Cursos Oferecidos:
Informática
Segurança do Trabalho
Recursos Humanos
Enfermagem