

## Servidores de Comércio Eletrônico

## Componentes da WWW

- WWW: serviço de acesso a conteúdos disponibilizados remotamente
- Os elementos desse serviço se comportam como um gigantesco sistema distribuído com as seguintes características:
  - arquitetura cliente-servidor
  - operação sobre a Internet (TCP/IP)
  - protocolo de acesso: HTTP
  - apresentação definida por HTML e Javascript
  - capacidade de geração de conteúdo dinâmico
  - recursos para armazenamento de dados

## Clientes WWW

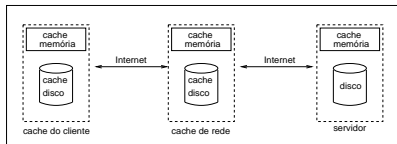
- Programas capazes de obter informações de servidores WWW
- Navegadores (*browsers*): clientes mais comuns
  - Recebem comandos dos usuários
  - Requisitam conteúdo da WWW
  - Exibem documentos recebidos
  - Exemplos de programas:
    - \* Netscape Communicator (múltiplas plataformas)
    - \* Microsoft Internet Explorer
    - \* Opções de software livre: Mozilla, Konqueror, Nautilus
- Robôs, *crawlers*, *spiders*: programas para navegação e processamento automático de informação da WWW

## Servidores WWW

- Sistemas autônomos passivos:
  - armazenam e controlam informações;
  - atendem requisições dos clientes.
- Informações podem ser documentos estáticos ou dinâmicos
- Principais servidores:
  - Apache (Linux):
    - \* software livre
    - \* responsável por cerca de 60% dos sítios WWW
  - Netscape Enterprise Server
  - Sun iPlanet Web Server
  - Lotus Domino
  - Microsoft IIS

## Caches e Proxies

- Tráfego entre clientes e servidores exige grande redundância
- Elementos intermediários podem ser inseridos para concentrar consultas de um grupo de clientes próximos
  - Consultas trafegam a rede externa apenas uma vez
  - Novas consultas idênticas são servidas localmente
  - Organização hierárquica aumenta os ganhos



## Infra-estrutura de rede

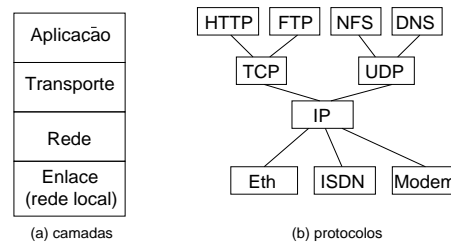
- Recursos básicos envolvidos
  - A Internet
  - A arquitetura TCP/IP
  - O serviço de nomes (DNS)

## A Internet

- Rede mundial de computadores usando o protocolo IP
- Redes locais de tecnologias diversas interligadas por roteadores
- Sistema organizado em camadas (protocolos)
- Serviço básico: entrega de pacotes (mensagens curtas)

## Organização em camadas

Objetivo: reduzir a complexidade do sistema



- Também denominada camada de rede local
- Abrange as diferentes tecnologias de interconexão

Tecnologia	Capacidade nominal	Principal utilização
modem	56 Kbps	acesso doméstico (baixo custo)
cable modem/xDSL	125 Kbps – 2 Mbps	acesso doméstico (banda larga)
T1/T3	1,5/45 Mbps	linhas dedicadas privadas
Ethernet/Fast E.	10/100 Mbps	redes locais
ATM (OC3)	155 Mbps	backbones
Satélite	1 Mbps	áreas de difícil acesso

#### Internet Protocol

- Presente em todos os elementos da Internet
- Protocolo da camada de interligação de redes
- Identificação de máquinas: endereços IP (p.ex.: 192.168.1.113, 150.164.10.1)
- Provê o envio de dados (pacotes) sem garantias
- Caminhos através da rede determinados pelos roteadores

## TCP

#### Transmission Control Protocol

- Adiciona um serviço de entrega confiável sobre IP
- Orientado a conexões, com três fases
  - estabelecimento (*three-way handshake*)
  - transferência de dados
    - \* *slow-start* (aumento gradativo da taxa de envio)
    - \* *delayed-acks* (otimização para redes lentas, atrasa pacotes)
    - \* controle de congestionamento (evita excessos na transmissão)
  - encerramento (liberação de recursos do servidor)

## TCP: impactos na WWW

- Diversas técnicas utilizadas por TCP afetam o comportamento da WWW

Característica	Motivação	Impacto
<i>Three-way handshake</i>	Garantir reconhecimento dos participantes	Introduz atrasos no início da comunicação
<i>Slow-start</i>	Evitar que conexões comecem com taxas muito altas	Taxa de transmissão cresce lentamente
<i>Delayed ACKs</i>	Melhorar a utilização da rede em certos casos	Introduz atrasos durante a comunicação

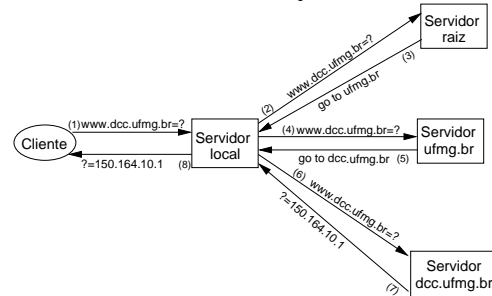
## DNS

#### Domain Name System

- A nível da rede IP, máquinas são conhecidas apenas por números
- O DNS permite que nomes sejam associados a esses números
- Implementado como uma arquitetura cliente-servidor distribuída
  - Cada sítio define um servidor para seus nomes
  - Servidores são organizados hierarquicamente a partir de uma raiz
  - Clientes direcionam consultas inicialmente para um servidor raiz
  - Resposta pode ser o endereço IP associado ao nome ou a identificação de outro servidor da hierarquia mais próximo do sítio procurado

## DNS: resolução de uma consulta

Qual o endereço do servidor [www.dcc.ufmg.br](http://www.dcc.ufmg.br)?



## Infra-estrutura de apresentação

- Nível responsável pela interface com o cliente
- Recebe requisições e compõe os documentos em resposta
- Implementado através dos recursos usuais da WWW
- Servidor WWW controla o processo de comunicação

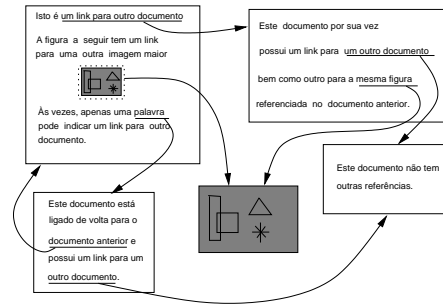
## A World Wide Web

- Sistema distribuído implementado sobre a Internet
- Ambiente dinâmico onde informações são publicadas e compartilhadas
- Documentos (páginas) são interligadas como hipertexto
- Criada inicialmente para troca de informações entre físicos
- Novas utilizações surgem a cada dia
  - Museus digitais
  - Máquinas de busca
  - Automação de processos de documentação
  - Comércio eletrônico

## Hipertexto

- Organização da informação de forma não-linear
- Grafo onde nós são documentos e arestas são relações entre eles
- Normalmente informação engloba texto, imagens, áudio e vídeo (hipermídia)
- Pontos específicos de um documento (âncoras) podem ser usados para remeter o usuário a outros documentos relacionados
- Leitores podem escolher o caminho que mais lhes interesse

## Hipertexto (exemplo)



## URI/URL

### Universal Resource Identifier/Locator

- URI: sequência de caracteres que identifica um recurso da rede
- URL: URI que inclui informação sobre localização do recurso
- Componentes usuais:
  - Protocolo de acesso (HTTP, FTP etc.)
  - Nome/endereço IP do servidor associado
  - Localização do documento dentro do servidor
  - Informações adicionais complementares

## URI/URL: exemplos

Protocolo	URI
NNTP	news://domain.name.com/news.group
HTTP	http://www.dcc.ufmg.br/index.html
FTP	ftp://ftp.w3c.org/pub/
SMTP	mailto://user@host
TELNET	telnet://maquina.dcc.ufmg.br

## HTTP

### HyperText Transfer Protocol

- Protocolo básico usado por clientes e servidores WWW
- Implementado como conjunto de comandos e respostas textuais
  - Comandos incluem a URI do objeto e a operação desejada (método)
  - Respostas são formadas por um código identificador, frase descritiva e conteúdo do documento relacionado, se for o caso
- Pode transportar documentos de tipos/mídias variadas
  - Cliente e servidor devem determinar formatos aceitáveis
  - Identificação de formatos realizada pela codificação MIME
- Duas versões atualmente disponíveis (HTTP 1.0, HTTP 1.1)

## HTTP: principais métodos

- **GET**: cliente deseja receber o objeto identificado (conteúdo e meta-informações relacionadas)
- **IMS**: versão alterada do GET onde o objeto só deve ser retornado se tiver sido modificado após uma certa data (*If-Modified-Since*)
- **HEAD**: semelhante ao GET, porém a resposta deve incluir apenas as meta-informações sobre o objeto
- **POST**: o objeto identificado pela URL é acessado como um programa; informações adicionais da requisição servem como parâmetros
- **PUT** e **DELETE**: definidos para permitir que clientes atualizem o conteúdo do servidor inserindo ou removendo documentos
  - Normalmente não são implementados por razões de segurança

## HTTP: respostas

### Exemplos de códigos de resposta e frases descritivas

Código	Frase	Significado
200	OK	Operação bem-sucedida
304	Not Modified	Resposta a um IMS
400	Bad Request	Requisição mal-formulada
403	Forbidden	Acesso ao URI não é permitido
404	Not Found	URI não corresponde a um objeto

## HTTP: 1.0 x 1.1

- Cada requisição/resposta indica a versão de HTTP usada
- Cliente e servidor podem negociar a versão a ser usada
- Versão 1.1 mantém uma única conexão com um servidor para todos os pedidos de objetos associados a um documento, enquanto HTTP 1.0 cria uma conexão para cada objeto
- HTTP 1.1 define novos métodos para operações especiais

- Em diversas operações na WWW é necessário garantir a segurança dos dados
- Problema especialmente importante na efetivação de transações comerciais através da WWW
- Diferentes aplicações podem ter diferentes necessidades:
  - **Privacidade:** proteção contra "escutas" indevidas
  - **Integridade:** garantia de que a informação não seja alterada
  - **Autenticação:** comprovação da identidade dos participantes
- Soluções podem utilizar protocolos especiais ou recursos do HTTP

### Secure Socket Layer/Transport Layer Security

- SSL foi proposto pela Netscape, estendido e padronizado como TLS
- Independente de HTTP, podendo ser usado por outras aplicações
- Protocolo independente adicionado entre HTTP e TCP (aplicação HTTP inalterada)
  - Nesse caso, o protocolo é denominado HTTP seguro (HTTPS)
- Permite que cliente e servidor negociem algoritmo de segurança
  - Protocolo de negociação (*handshake*)
  - Protocolo de transferência propriamente dito

## SSL/TLS

- Durante a negociação, cliente e servidor selecionam algoritmos a serem utilizados
- Servidor se identifica para o cliente por um certificado digital
- Cliente verifica a validade do certificado de forma independente
- Se o certificado é aceito ele pode ser usado pelo cliente para enviar para o servidor uma chave privada em sigilo
- Comunicação posterior entre os participantes usa chave privada
- Usualmente o cliente não precisa de um certificado para se identificar para o servidor
  - Identificação por outros meios (cartão de crédito, senha etc.)

## Segurança em HTTP

- HTTP por si só pode garantir apenas autenticação simples
- Diálogo desafio/resposta (*challenge/response*)
- Acesso a um documento protegido é respondido com uma mensagem de desafio
- Navegador envia a resposta do cliente como conjunto de credenciais
- Servidor verifica credenciais contra uma base de dados definida
  - Usualmente credenciais são identificador e senha de usuário
- Toda a comunicação é feita por protocolo textual sem criptografia!!!

## Composição da interface

- Elementos discutidos até aqui garantem a transferência de pedidos e de respostas com a informação desejada, porém não definem como essa informação será representada
- Essa é a tarefa da linguagem de representação dos documentos (HTML) e da linguagem de comandos associada (Javascript)

## HTML

### HyperText Markup Language

- Linguagem de formatação de documentos padronizada aberta
- Documentos em forma textual (ASCII) com marcas especiais para controle da formatação
- Marcas (*tags*) definidas em pares, como <H1> e </H1>, identificando o início e fim de um trecho do documento
- Efeito das marcas pode ser qualificado por parâmetros associados
- Apesar de haver um padrão bem definido, grande parte dos documentos da WWW apresenta pequenas violações desse padrão, aceitas pelos navegadores

## HTML: marcas

- Estrutura do documento:
  - <HTML> e </HTML> delimitam o documento;
  - <HEAD> e </HEAD> delimitam o cabeçalho do mesmo;
  - <BODY> e </BODY> delimitam o corpo que será exibido.
- Formatação: p. ex., <H1> e </H1> indicam um texto do tipo 1;
- Elementos especiais:
  - <IMG> define uma figura; parâmetros incluem a URL do arquivo propriamente dito e uma descrição textual:
 

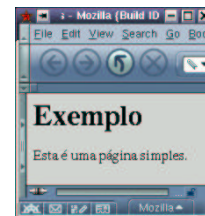
```
<IMG SRC="http://maquina/arquivo.gif" ALT="Descrição">
```
  - <A> e </A> definem uma âncora no texto; um parâmetro obrigatório é a URL associada:
 

```
<A HREF="http://maquina/pagina.html">Texto ancorado</A>
```

## HTML: exemplo

```
<html>
<head>
  <title>Página simples</title>
</head>
<body>
  <H1>Exemplo</H1>
  <p>Esta é uma página simples.
</body>
</html>
```

(a) Conteúdo do arquivo

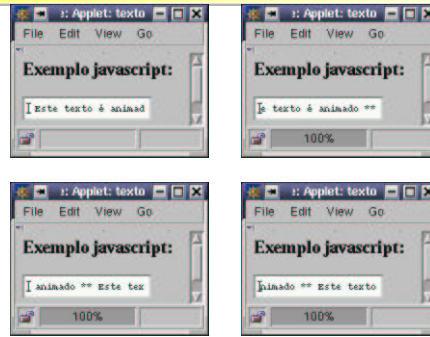


(b) Visualização

## Javascript: exemplo

```
<html>
<head> <title>Applet: texto animado</title> </head>
<body onload="Scroll()">
<SCRIPT LANGUAGE="JavaScript">
<!--
var ScrollString = " ** Este texto é animado"
var timer=0
function Scroll() {
    document.box.scrolltext.value = ScrollString
    ScrollString = ScrollString.substr(1, ScrollString.length)
    + ScrollString.charAt(0)
    timer= setTimeout("Scroll()",150)
}
// ----- -->
</SCRIPT>
<H1> Exemplo javascript:</H1>
<p> <FORM NAME="box"><INPUT name="scrolltext"> </FORM> </p>
</body>
</html>
```

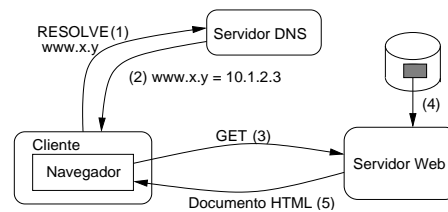
## Javascript: exemplo



## Passos de uma requisição HTTP

1. O cliente faz uma consulta ao serviço de DNS para obter o endereço IP correspondente ao servidor
2. O servidor DNS responde com o endereço IP obtido
3. O navegador do cliente abre uma conexão TCP para o servidor
  - Caso a URL exija HTTPS mensagens são trocadas para estabelecer um canal seguro
  - O navegador envia o método GET e a URL do objeto desejado
4. O servidor extrai da URL o identificador do arquivo na máquina e busca o seu conteúdo no disco
5. O conteúdo HTML é encapsulado em uma resposta HTTP e devolvido para o navegador que o exibe na tela

## Passos de uma requisição HTTP



## Infra-estrutura de lógica de negócio

- Em sistemas de comércio eletrônico o conteúdo das páginas é fortemente dependente da semântica da aplicação
  - Também denominada lógica do negócio
  - Implementação das funções para interação entre cliente e servidor:
    - \* identificação do cliente;
    - \* procura por produtos;
    - \* manutenção do carrinho de compras.
- Para que todos essas operações possam se refletir nas páginas exibidas para o usuário é preciso que estas sejam geradas dinamicamente

## CGI

### Common Gateway Interface

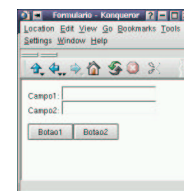
- Interface para execução de programas associados a certas URLs
- Uma URL com certo padrão (p.ex., "cgi-bin") identifica um programa
- Após o nome do programa podem ser adicionados à URL parâmetros para o mesmo
- O programa gera como saída um documento HTML que é entregue ao servidor HTTP ao executar
- Outras formas de execução de comandos/programas incluem:
  - módulos especiais embutidos no código do servidor (Perl, Python etc.)
  - extensões do servidor para interpretar comandos dentro do código HTML (JSP, ASP, PHP)

## Formulários HTML

- Para que os programas geradores de páginas se adaptem às necessidades do usuário eles precisam ser capazes de receber dados
- Linguagem HTML prevê marcas para definição de formulários a serem preenchidos pelo usuário
- Marcas definem botões, campos, valores aceitáveis, associação de variáveis a cada campo etc.
- Ao se selecionar um link/botão na página, valores das variáveis (campos) são adicionados à URL do link como parâmetros
- A URL pode ser definida de forma a acionar um script CGI

## Formulários HTML: exemplo

```
<html>
<head><title>Formulario</title></head>
<body>
<form action="/formulario"
      method="get">
    Campo1: <input name="C1"
              type="text">
    Campo2: <input name="C2"
              type="text">
    <input name="B" type="submit"
            value="Botao1">
    <input name="B" type="submit"
            value="Botao2">
  </form>
</body>
</html>
```



(a) Conteúdo do arquivo.

(b) Visualização.

## Problemas de CGI

- Cada requisição por uma URL que identifica um *script* CGI precisa disparar um novo processo na máquina do servidor
  - Inicialização de processos introduz atrasos e aumenta a carga da máquina
  - Métodos como módulos Perl ou interpretadores JSP/ASP associados ao servidor reduzem esse problema
- Em muitos casos, a parte a ser gerada dinamicamente no documento é pequena
  - Uso de CGI ou módulos exige que o *script* gere o documento completo
  - Extensões como JSP/PHP permite que o documento HTML seja anotado com comandos para gerar dinamicamente apenas trechos

## Manutenção de estado

- Ao contrário da infra-estrutura de apresentação, a lógica de negócios exige que se mantenha um histórico da interação entre cliente e servidor
- Sessão: conjunto das operações executadas por um cliente
  - páginas visitadas durante a interação
  - dados cadastrais do cliente
  - produtos selecionados
  - outras informações fornecidas pelo usuário

## Manutenção de estado

- HTTP não mantém estado e cada requisição é normalmente enviada em uma nova conexão, dificultando a identificação da sessão
- Cada sessão deve receber um identificador único reconhecido pela lógica de negócios
- Servidor deve ser capaz de recuperar identificador na requisição
- Três soluções para o problema:
  - campos invisíveis em formulários HTML;
  - reescrita de URLs;
  - uso de *cookies*.

## Manutenção de estado: campos invisíveis

- Ao selecionar um link em um formulário os valores de todos os campos do mesmo são adicionados à URL enviada na requisição
- HTML permite que campos sejam definidos como “invisíveis”, não sendo apresentados ao usuário
  - Se o formulário é criado com um campo invisível e um valor associado ao mesmo, esse valor será mantido inalterado pelo navegador
  - Qualquer link selecionado pelo usuário naquele formulário fará com que o valor do campo invisível seja enviado na requisição
  - Cada página enviada em uma sessão é gerada dinamicamente com o identificador da seção em um campo invisível

## Manutenção de estado: reescrita de URLs

- Após o acesso a uma página inicial que causa a abertura de uma sessão, todas as páginas servidas têm suas URL alteradas
  - Toda URL em uma página da sessão é reescrita de forma que um trecho da mesma receba o identificador da sessão
  - O servidor é configurado com um *script* CGI que transforma a URL novamente em um identificador de arquivo válido
  - O identificador de sessão retirado da URL passa a ser um parâmetro entregue à lógica de negócios

## Manutenção de estado: Cookies

- *Cookies*: recurso definido pela Netscape para armazenamento de informações (pares nome/valor) no navegador
- HTTP inclui recursos de manipulação entre servidor e navegador:
  - servidor pode depositar um *cookie* (par nome/valor) no navegador
  - servidor pode requisitar que o navegador envie de volta o valor associado a um determinado nome
- Ao abrir uma sessão o servidor deposita um *cookie* com o identificador da mesma no navegador do cliente
- Cada vez que uma página dinâmica dependa de dados da sessão o servidor consulta o navegador para recuperar o *cookie*
- Apesar dos vários usos válidos de *cookies* há também várias questões de segurança e invasão de privacidade associadas a elas

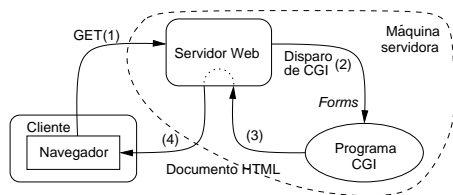
## Implementação da lógica de negócios com CGI

- Toda página com dados da sessão é gerada por um *script* CGI
- Todo *script* usa uma das técnicas de identificação de sessão
- Cada *script* deve recuperar estado da sessão do disco e atualizá-lo
  - Cada requisição dispara um novo CGI
  - Servidor HTTP não mantém qualquer noção de estado
- Acessos a disco podem se tornar um problema de desempenho

## Passos de uma requisição com CGI

1. Servidor HTTP recebe a requisição por uma página dinâmica
2. Com base na URL fornecida um determinado *script* CGI é disparado para processar a requisição da seguinte forma:
  - obtém o identificador da sessão
  - recupera do disco o estado da sessão
  - atualiza o estado no disco com base na nova operação
3. Um novo documento HTML é gerado como resultado do processamento e entregue ao servidor HTTP
4. O servidor HTTP entrega o documento ao navegador do cliente

## Passos de uma requisição com CGI



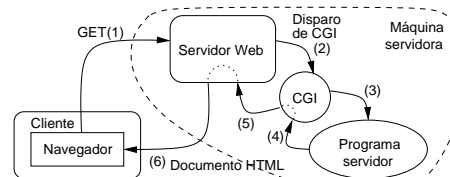
## Implementação da lógica de negócios com servidor de aplicação

- Toda página dinâmica é gerada por um servidor de aplicação
- O *script* CGI apenas formata os dados para o servidor
- O servidor permanece em execução gerenciando todas as sessões
  - O estado das sessões é mantido em memória
  - Nenhum acesso a disco é necessário para esse fim
- Opcionalmente o *script* CGI pode realizar algum processamento final do documento resultante

## Passos de uma requisição com servidor de aplicação

1. Servidor HTTP recebe a requisição por uma página dinâmica
2. Um *script* CGI é disparado para recuperar o identificador da sessão
3. O *script* repassa a requisição, a URL e o identificador para o servidor
  - O programa servidor processa a requisição na memória
4. O documento HTML é montado pelo servidor de aplicação e devolvido ao *script*
5. O *script* repassa o documento para o servidor HTTP
6. O servidor HTTP entrega o documento ao navegador do cliente

## Passos de uma requisição com servidor de aplicação



## Linguagens para páginas dinâmicas

- A princípio, qualquer linguagem pode ser usada para o desenvolvimento de programas para geração de páginas dinâmicas
- Algumas linguagens possuem recursos especiais para uma melhor interface com CGI:
  - processamento de URLs
  - montagem de documentos HTML
  - comandos para controle do servidor
  - extração de *cookies* e campos de formulários
- Outras linguagens foram desenvolvidas especificamente para serem injetadas em documentos HTML e gerar apenas certos trechos dos mesmos

## Linguagens para geração de páginas: Perl, Java etc.

- Perl: uma das linguagens mais difundidas para CGI
- Java: linguagem de muitos recursos e alta portabilidade
- Servidor Apache possui módulos para execução de várias linguagens que dispensam o disparo de novos programas
- Possuem módulos especiais para controle de HTTP, geração de documentos HTML etc.
- Programas/*scripts* devem gerar a página HTML completa

## Linguagens para geração de páginas: Perl

```
#!/usr/bin/perl -w
use strict;
use CGI qw(:standard);
my $value=param('userName');
print header(),
      start_html("Perl"),
      p("<H1>"),
      p("Saudações, ", ($value) ),
      p("</H1>"),
      p("Texto HTML comum"),
      end_html();

# Fim do programa
```

## Linguagens para geração de páginas: Java

```
public void doGet(
    HttpServletRequest rq,
    HttpServletResponse rp )
{
    rp.setContentType("text/html");
    PrintWriter out = rp.getWriter();
    out.println(
        HttpUtil.head("Java") +
        "<H1>Saudações, " +
        HttpUtil.param("userName") +
        "</H1>" +
        "Texto HTML comum" +
        HttpUtil.endBody()
    );
}
```

- JSP: extensão de Java embutida em servidores HTTP (Apache etc.)
- PHP ("PHP: Hypertext Preprocessor"): linguagem desenvolvida especificamente para o inserção em páginas HTML
- Servidores devem ser capazes de interpretar comandos na linguagem escolhida (uso de módulos especiais)
- Páginas podem ser geradas em HTML com comandos da linguagem apenas para os trechos dependentes da sessão
- Usualmente utilizados em páginas com poucos trechos dinâmicos

```
<html>
<head>
<title>Página JSP</title>
</head>
<body>
  <H1>Saudações,
  <%
    out.println(Utils.nameCookie(req));
  %>
  </H1>
  Texto HTML comum aqui.
</body>
</html>
```

```
<html>
<head>
<title>Página PHP</title>
</head>
<body>
  <H1>Saudações,
  <?php
    echo $userName;
  ?>
  </H1>
  Texto HTML comum aqui.
</body>
</html>
```

- Lógica de negócios mantém apenas informações (dinâmicas) sobre sessões: cesta de compras, páginas visitadas etc.
- Informações sobre produtos e cadastro de clientes têm demandas diferentes:
  - armazenamento contínuo
  - garantias de consistência entre diversas sessões
  - disponibilidade para outras aplicações
  - acesso concorrente
- Soluções especialmente desenvolvidas são possíveis, mas usualmente um sistema de gerência de bancos de dados (SGBD) é adotado
  - recursos como tolerância a falhas e consistência já disponíveis
  - papel do projetista é definir as categorias de acesso dos dados

- É importante identificar os tipos de dados envolvidos para se determinar os comportamentos aceitáveis para a aplicação
- Principais características a serem consideradas nesse processo:
  - volatilidade — frequência e forma de alteração dos dados;
  - compartilhamento — tratamento dado a acessos concorrentes.

- Estáticos: informações alteradas apenas para correções eventuais  
exemplo: dados de um livro, tais como autor, título, resumo
- Pouco voláteis: sofrem alterações esporádicas  
exemplo: o preço de um livro pode mudar a cada remessa
  - sub-categoria: dados de alteração definitiva
    - \* Informações atribuídas a um produto apenas uma vez  
exemplo: identidade e endereço do comprador
- Muito voláteis: sofrem alterações constantes durante as sessões  
exemplo: quantidade de um livro muito procurado em estoque.

A volatilidade indica quanta atenção deve ser dada ao armazenamento

- Dados estáticos podem ser tratados com flexibilidade, podendo ser facilmente replicados
- Dados muito voláteis exigem cuidados especiais para garantir sua consistência entre aplicações e mantê-los atualizados
- Dados definitivos devem ser garantidamente alterados uma só vez teoricamente por uma entre diversas aplicações

- Replicáveis: poucas limitações sobre acessos concorrentes
  - várias cópias podem ser feitas e acessadas sem conflitos
- Periodicamente consistentes: vários acessos podem ser combinados
  - acessos e alterações são permitidos sob regras bem definidas
  - a visão global resultante pode ser mantida da combinação das diversas alterações de forma consistente
- Mutuamente exclusivos: apenas um acesso permitido por vez
  - alterações concorrentes não podem ser feitas com garantias
  - algumas combinações concorrentes podem ser possíveis, mas outras podem levar a estados inconsistentes  
exemplo: alterações de saldos bancários ou de estoque



## Categorias de dados

Normalmente algumas combinações em termos de volatilidade e compartilhamento são mais utilizadas na prática:

- dados estáticos são normalmente replicáveis;
- dados muito voláteis tendem a ser mutuamente exclusivos;
- dados pouco voláteis podem ser periodicamente consistentes.

Outras combinações são também possíveis:

- dados muito voláteis podem ser definidos como periodicamente consistentes segundo uma determinada semântica;
- dados pouco voláteis podem ser mutuamente exclusivos, já que por definição as alterações serão pouco frequentes.

Nem todas as combinações são razoáveis: por exemplo, não há sentido em se definir dados estáticos como mutuamente exclusivos.

## ODBC

### Open DataBase Connectivity

- Os diversos SGBD possuem similaridades, mas as interfaces tendem a ser diferenciadas
- Para solucionar esse problema criou-se o padrão ODBC
- Conjunto de comandos estendidos para permitir a integração de aplicações a diversos SGBD de forma padronizada
- Inclui SQL (*Simple Query Language*) para a construção de consultas

## ODBC

- Praticamente todo SGBD disponível exporta uma interface ODBC
- Existem interfaces para integração nas principais linguagens para geração de páginas, tais como Java, Perl e PHP
- Interface define comandos para conexão e configuração
- Início do acesso exige apenas um identificador DSN (*Data Source Name*)
  - identificação do SGBD a ser utilizado
  - identificação para acesso do usuário (*login* etc.)
  - consultas expressas utilizando SQL

## Exemplo: ODBC + PHP

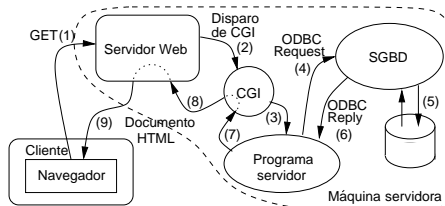
```
<?
// DSN "db", usuario "zeh", senha "maneh"
$conexao = odbc_connect("db", "zeh", "maneh");
// consulta: tabela usuarios, nome e sobrenome
$consulta = "SELECT nome, sobrenome FROM usuarios";
// executa a consulta
$resultado = odbc_exec($conexao, $consulta);
// recupera os dados do bando de dados
while (odbc_fetch_row($resultado)) {
    $nome = odbc_result($resultado, 1);
    $sobrenome = odbc_result($resultado, 2);
    print("$nome $sobrenome");
}
// encerra a conexão
odbc_close($conexao);
?>
```

## Passos de uma requisição com SGBD

No caso de se utilizar um SGBD a utilização de um servidor de aplicação independente se torna mais interessante por permitir a inicialização do SGBD uma única vez.

1. O pedido do cliente é recebido pelo servidor HTTP
2. Um *script* CGI é disparado...
3. ...e transfere a requisição para o servidor de aplicação
4. O servidor monta a consulta ao SGBD e a envia utilizando ODBC
5. O SGBD realiza a consulta sobre a base de dados
6. O resultado é devolvido ao servidor de aplicação, o qual completa o processamento estruturando a resposta em uma página HTML
7. A página é devolvida para o *script* CGI...
8. ...que completa o processamento e a devolve ao servidor HTTP...
9. ...que finalmente a entrega ao navegador do cliente

## Passos de uma requisição com SGBD



## Estruturação do conteúdo

- O uso de SGBDs através da interface ODBC soluciona o problema de armazenamento de dados
- Técnicas de páginas dinâmicas permitem a geração de documentos HTML completos a partir desses dados
- Uma nova tendência é a extração de informações a partir de páginas já estruturadas para exibição
- Essa estruturação de conteúdo é conseguida através do uso de XML

## XML

### eXtended Markup Language

- Derivada do padrão SGML (*Standard Generalized Markup Language*)
- Fortemente baseada no uso de marcas (*tags*)
- Marcas podem ser definidas para cada tipo de documento
- Interpretação das marcas é definida pelo usuário em um documento associado, o DTD (*Document Type Definition*)
- Regras de exibição podem ser definidas para cada tipo de documento e cada tipo de interface de apresentação

```

<TITLE>Comp. usado</TITLE>
<BODY>
  <UL>
    <LI> HAL 9000
    <LI> 9 GHz
    <LI> 128 TeraBytes RAM
    <LI> 20 PetaBytes HD
    <LI> 200 Kg
    <LI> US$ 3.000
  </UL>
</BODY>

```

(a) HTML

```

<COMPUTER CLASS= "Super">
  <MANUFACT> HAL Inc. </MANUFACT>
  <FAMILY> Thinking </FAMILY>
  <LINE> HAL </LINE>
  <MODEL> 9000 </MODEL>
  <SPEED UOM="GHz"> 9 </SPEED>
  <RAM UOM="TB"> 128 </RAM>
  <DISK UOM="PB"> 20 </DISK>
  <WEIGHT UOM="Kg"> 200 </WEIGHT>
  <COST CURR="USD"> 3000 </COST>
</COMPUTER>

```

(b) XML

- O objetivo de XML é definir a estrutura da informação
- Com base nessa definição seria possível processar automaticamente páginas montadas com um certo conjunto de marcas
- A possibilidade de cada usuário definir seu conjunto de marcas é uma grande flexibilidade mas pode causar problemas
  - O uso de conjuntos diferentes de marcas por cada usuário impossibilitaria o processamento automático do conteúdo
- Existem esforços para criar marcas padronizadas
  - Um exemplo importante é ebXML (*electronic business XML*)

## Aspectos de implementação dos servidores

A discussão anterior nos permite identificar alguns elementos fundamentais da organização do sistema de comércio eletrônico:

- O processamento dos protocolos de rede está a cargo do sistema operacional, que oferece a interface *sockets* para a aplicação
- O tratamento de requisições do usuário é responsabilidade do servidor HTTP, que se encarrega da comunicação com os demais
- Páginas dinâmicas são geradas pelo servidor da aplicação, responsável pela implementação da lógica de negócio
- O gerenciamento dos dados de longa duração do serviço é realizado pelo SGBD que é acessado pelo servidor de aplicação

## Aspectos de implementação dos servidores

A implementação mais comum utiliza um sistema de três camadas (*three-tier application*):

- Apresentação: servidor HTTP
  - Interface CGI ou módulos interpretadores do servidor
- Lógica de negócio: servidor de apresentação (loja)
  - Interface ODBC
- Armazenamento de dados: SGBD

## Estruturação dos servidores

Mesmo após a adoção da arquitetura de três camadas, é preciso ainda definir outras características do sistema, como a implementação de cada servidor para lidar com eventos concorrentes:

- servidor totalmente seriado
- servidor monoprocessado compartimentado
- servidor seriado replicado
- servidor *multi-threaded*

## Estruturação dos servidores

Servidor totalmente seriado

- Desenvolvido para atender uma requisição por vez
- Processamento prossegue pelas várias etapas da requisição
- Não há possibilidade de explorar a concorrência entre tarefas
- Solução de implementação mais simples
- Desempenho percebido pelos clientes tende a ser ruim
- Utilizado apenas em protótipos simples

## Estruturação dos servidores

Servidor monoprocessado compartimentado

- Processo único com loop em torno da primitiva *select*
- Processamento é dividido em tarefas não bloqueantes
- Toda vez que o processamento atinge uma operação bloqueante outra tarefa pode ser executada
- Custo para chaveamento de tarefas é o menor possível
- Implementações podem atingir desempenhos bastante elevados
- Código tende a ser complexo para garantir a não-interferência das tarefas executadas em um único processo
- Utilizado no servidor e cache HTTP Squid

## Estruturação dos servidores

Servidor seriado replicado

- Dispara cópias do servidor para atender diferentes requisições
- Custo de chaveamento entre tarefas é relativamente alto
- Implementação simples: cada processo atende uma única requisição
- Solução viável apenas se as tarefas executadas forem independentes
- Desempenho depende do número de cópias do programa em execução e da forma como as mesmas são coordenadas
- O número de processos pode sobrecarregar o sistema operacional
- Utilizado pelas primeiras versões do servidor HTTP Apache

### Servidor *multi-threaded*

- Vários fluxos de execução (*threads*) em um mesmo processo
- Código para cada fluxo de execução é sequencial, mais complexo que um servidor seriado devido aos cuidados na interação entre *threads*
- Custo de chaveamento entre tarefas é maior que no servidor compartimentado, mas menor que no servidor replicado
- Permite a implementação de tarefas onde haja interação entre diferentes requisições utilizando-se a memória compartilhada
- Utilizado pelo SGBD MySQL

- Quais componentes da máquina servidora são mais exigidos?
- Quais elementos do sistema causam as maiores cargas?
- Para a maior parte dos casos podemos considerar três componentes:
  - o conjunto processador/memória,
  - o sistema de discos,
  - a(s) interface(s) de rede,
  - controle de acesso.

## Demandas de processamento: processador/memória

- Processamento aritmético: cálculos de amortização, fretes etc.
- Processamento de caracteres: formatação de texto, buscas, cópias
- Suporte a transações: monitoração de posições de memória
- Armazenamento temporário: replicação de dados, resultados temporários, estruturas de dados com alocação dinâmica
- Ambientes de execução: abstrações oferecidas por linguagens interpretadas e/ou orientadas a objetos (*garbage collection* etc.)

## Demandas de processamento: discos

- Leitura de arquivos: busca de imagens e páginas estáticas
- Escrita de arquivos: alterações de documentos durante uma sessão
- Consultas atributo-valor: grande parte dos acessos em bancos de dados montam listas após busca por uma chave
- Armazenamento persistente: SGBD deve garantir a durabilidade e o isolamento dos dados

## Demandas de processamento: interfaces de rede

- Envio de mensagens: exige que a interface de rede seja capaz de receber e armazenar mensagens da aplicação para envio
- Recebimento de mensagens: retirada de mensagens da rede e entrega para o sistema operacional
  - esse processamento é ditado pela taxa de chegada de dados
  - pode acarretar cargas inesperadas e de difícil caracterização
  - todo servidor na Internet deve ser capaz desse processamento
- Demandas da interface de rede podem levar a demandas sobre o processador e a memória
  - tratamento de interrupções
  - processamento de protocolos da pilha TCP/IP

## Demandas de processamento: controle de acesso

- Acionado sempre que operações de segurança são necessárias:
  - garantia de acesso seguro,
  - garantia de autenticidade e/ou confidencialidade,
  - verificação de identificação de usuários.
- Em última instância se reflete em demandas sobre a interface de rede e processamento aritmético

## Demandas de processamento

Na análise de uma implementação é necessário:

- identificar as demandas de processamento de cada elemento,
- verificar se as soluções privilegiam demandas de menor custo.

Por exemplo, na escolha de uma implementação em Perl, uma linguagem para geração de *scripts*, interpretada:

- Perl é eficiente na manipulação de caracteres e deve ser usada quando a demanda por esse tipo de processamento for alta;
- por outro lado, deve ser evitada em casos com grande volume de processamento aritmético ou onde a memória já estiver sendo muito demandada por outros motivos.

## Conclusões

No desenvolvimento de aplicações de comércio eletrônico devemos considerar os recursos de infra-estrutura em quatro níveis:

- armazenamento — manutenção dos dados de longa duração, sob a responsabilidade de um SGBD;
- lógica da aplicação — implementação dos algoritmos responsáveis pela operação do negócio, a cargo do servidor de aplicação;
- apresentação — interação com o programa navegador, executada pelo servidor HTTP;
- rede — recursos de operação da Internet necessários para garantir a comunicação entre cliente e servidor.