

## Aula 12

Professor:

*Claudio Esperança*

## Autenticação e Segurança

Conteúdo:

- Autenticação via HTTP
- Autenticação com sessões
- Secure Sockets Layer

# Necessidade para Autenticação e Segurança

- Para proteger dados sensíveis
  - ▶ Compras via Internet
  - ▶ Movimentação de contas de banco via Internet
- Serviços web pagos
  - ▶ Apenas usuários pagantes podem acessar
- Impedir estranhos de obterem informações sigilosas
  - ▶ Tráfego de dados deve ser protegido contra estranhos

# Autenticação HTTP

- O protocolo http dispõe de um mecanismo para autenticar e autorizar usuários
- Quando um recurso (uma página, por exemplo) protegido é acessado, o servidor envia uma resposta com código de estado **401 Unauthorized**
  - ▶ O navegador, ao receber esta resposta, tipicamente exibe uma caixa de diálogo pedindo nome e senha:



Abrir editor

Abrir navegador

# Autenticação HTTP

- Uma vez obtidas as credenciais (nome e senha), o navegador pode então pedir de novo o conteúdo, desta vez informando ao servidor as credenciais

Navegador



Servidor



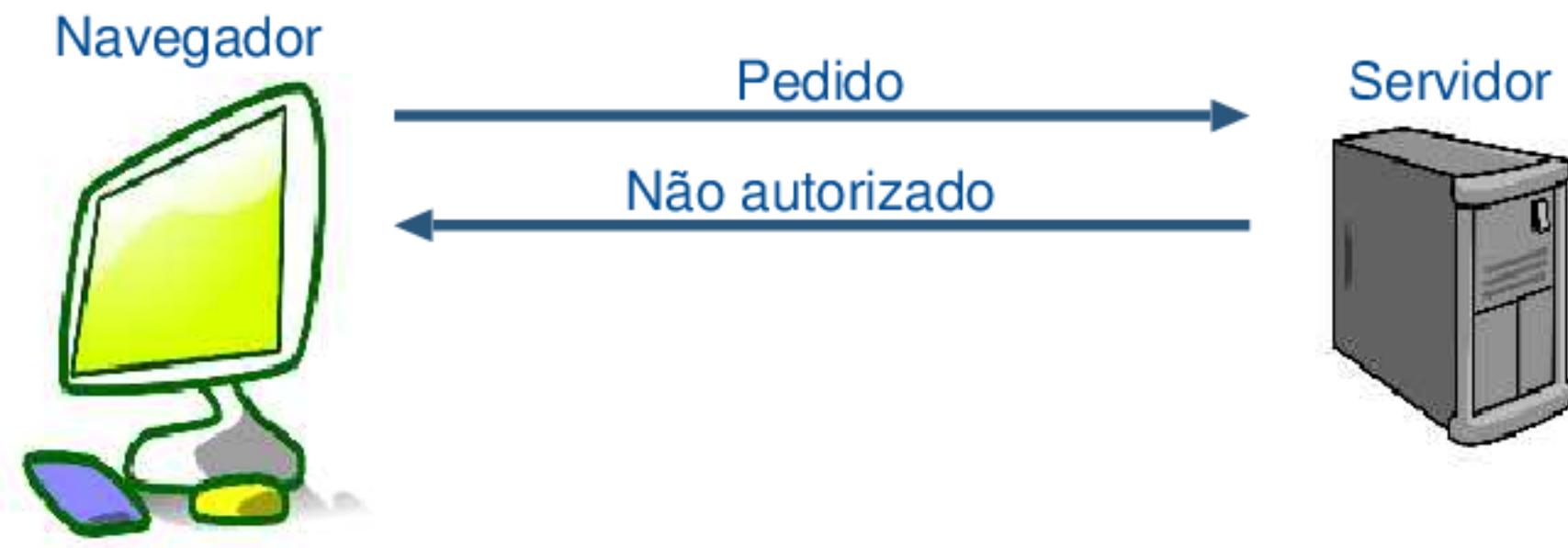
# Autenticação HTTP

- Uma vez obtidas as credenciais (nome e senha), o navegador pode então pedir de novo o conteúdo, desta vez informando ao servidor as credenciais



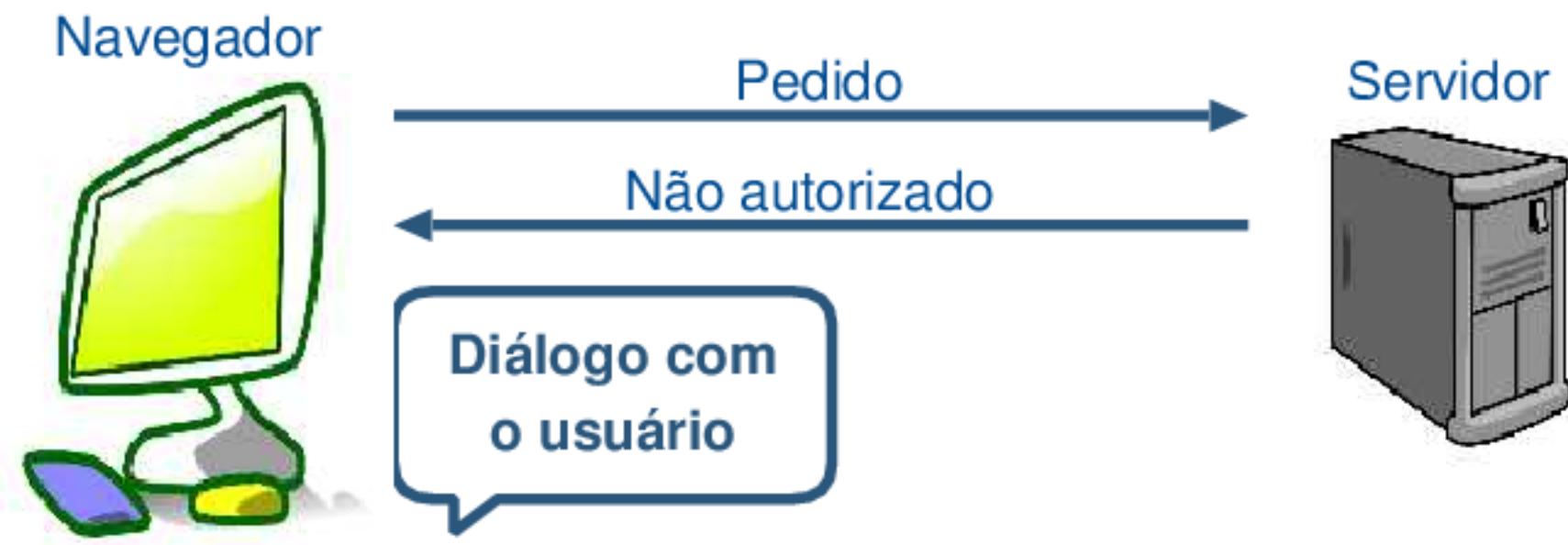
# Autenticação HTTP

- Uma vez obtidas as credenciais (nome e senha), o navegador pode então pedir de novo o conteúdo, desta vez informando ao servidor as credenciais



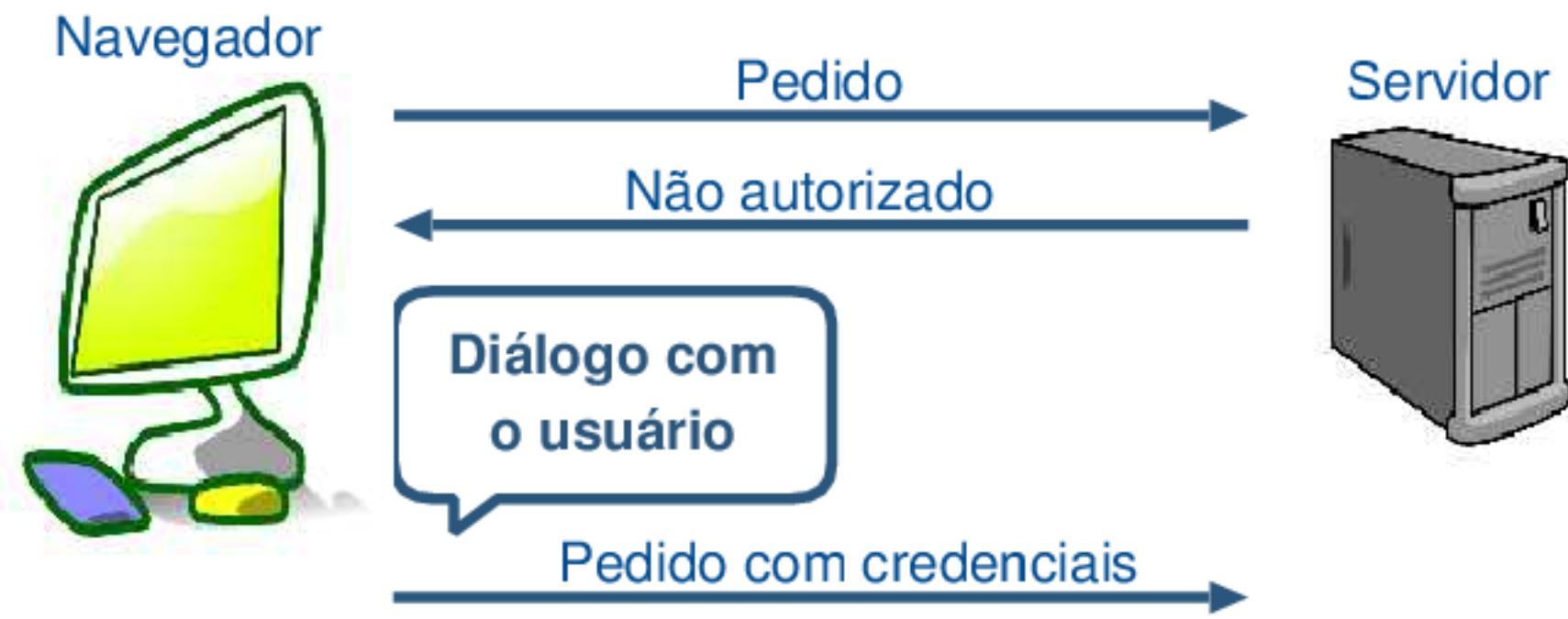
# Autenticação HTTP

- Uma vez obtidas as credenciais (nome e senha), o navegador pode então pedir de novo o conteúdo, desta vez informando ao servidor as credenciais



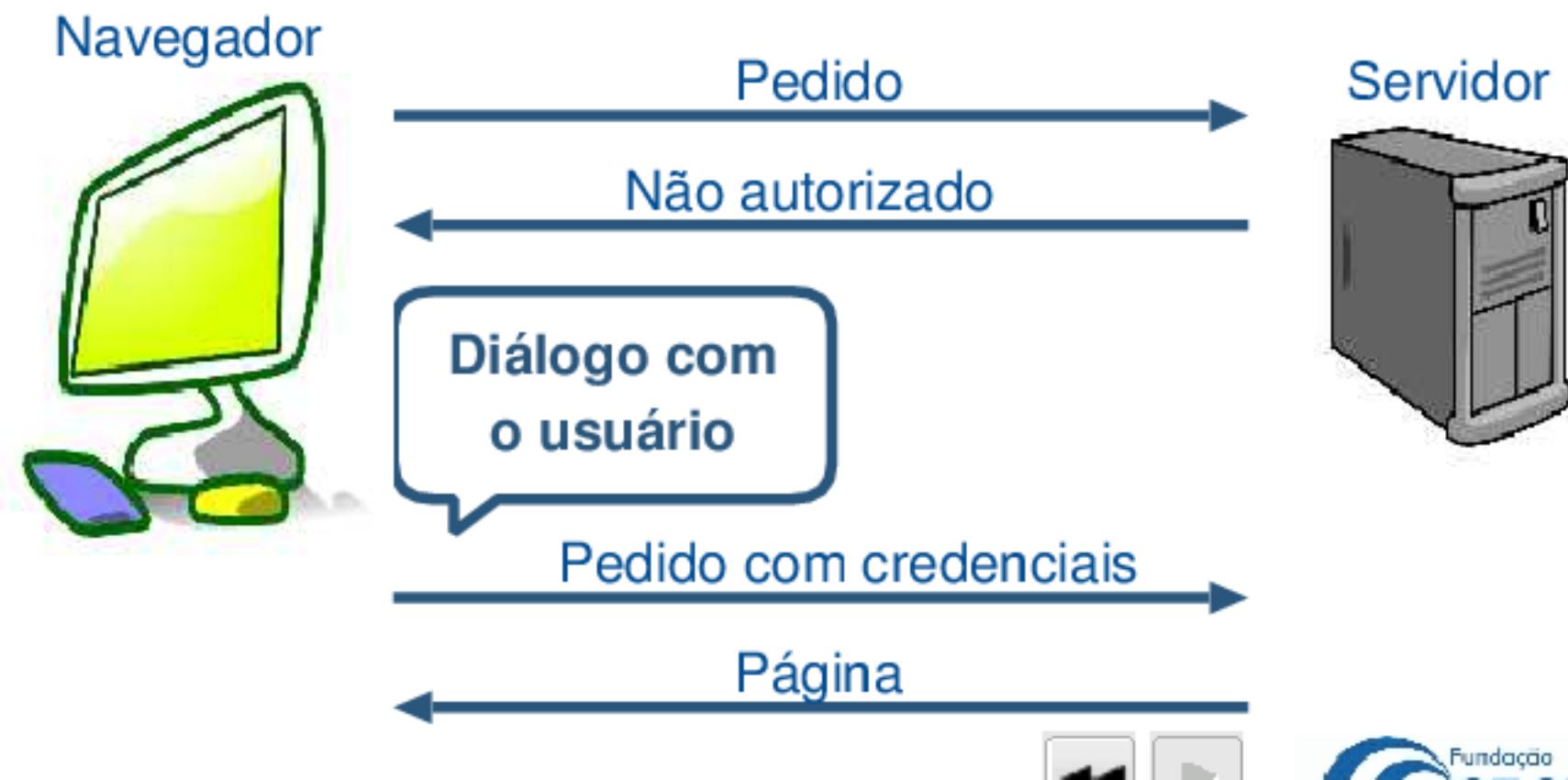
# Autenticação HTTP

- Uma vez obtidas as credenciais (nome e senha), o navegador pode então pedir de novo o conteúdo, desta vez informando ao servidor as credenciais



# Autenticação HTTP

- Uma vez obtidas as credenciais (nome e senha), o navegador pode então pedir de novo o conteúdo, desta vez informando ao servidor as credenciais



# Criando um diretório protegido

- Para criar um diretório protegido pelo mecanismo do HTTP, há duas opções
  - ▶ Configurar o servidor Apache (arquivo httpd.conf)
    - Normalmente requer privilégio de administrador
  - ▶ Criar um arquivo especial com nome .htaccess no diretório a ser protegido. Exemplo:

```
AuthType Basic
AuthName "Diretorio Protegido"
AuthUserFile /home/esperanc/senhas
Require valid-user
```

# Formato do arquivo .htaccess

- **AuthType** *tipoAutenticacao*
  - ▶ Onde *tipoAutenticacao* pode ser **Basic** ou **Digest**
  - ▶ **Basic** codifica o nome e a senha de forma pouco segura
  - ▶ **Digest** codifica de forma mais segura mas incompatível com navegadores antigos
- **AuthName** *nomeRecurso*
  - ▶ *nomeRecurso* é usado pelo navegador ao mostrar a caixa de diálogo
- **AuthUserFile** *arquivoSenhas*
  - ▶ Indica o nome do arquivo com nomes e senhas codificadas
- **Require** *usuários*
  - ▶ Nomes de usuários permitidos separados por vírgula
  - ▶ Use **valid-user** se qualquer usuário do arquivo de senhas pode acessar as páginas do diretório

# Construindo um arquivo de senhas

- O arquivo de senhas pode ser construído com o utilitário **htpasswd** (faz parte do Apache)
- **htpasswd -c arquivo nome**
  - ▶ Cria o *arquivo* de senhas com uma entrada para o usuário *nome*
  - ▶ A *senha* do usuário será pedida interativamente
  - ▶ Se a opção **-c** não estiver presente, o novo par *nome/senha* será acrescentada ao arquivo previamente existente

# Exemplo

```
$ htpasswd -c senhas aluno
New password: *****
Re-type new password: *****
Adding password for user aluno
$ htpasswd senhas fulano
New password: *****
Re-type new password: *****
$ more senhas
aluno:HuQUt3.uKdHgo
fulano:kdewl7J5108bk
```

# Acesso a credenciais via PHP

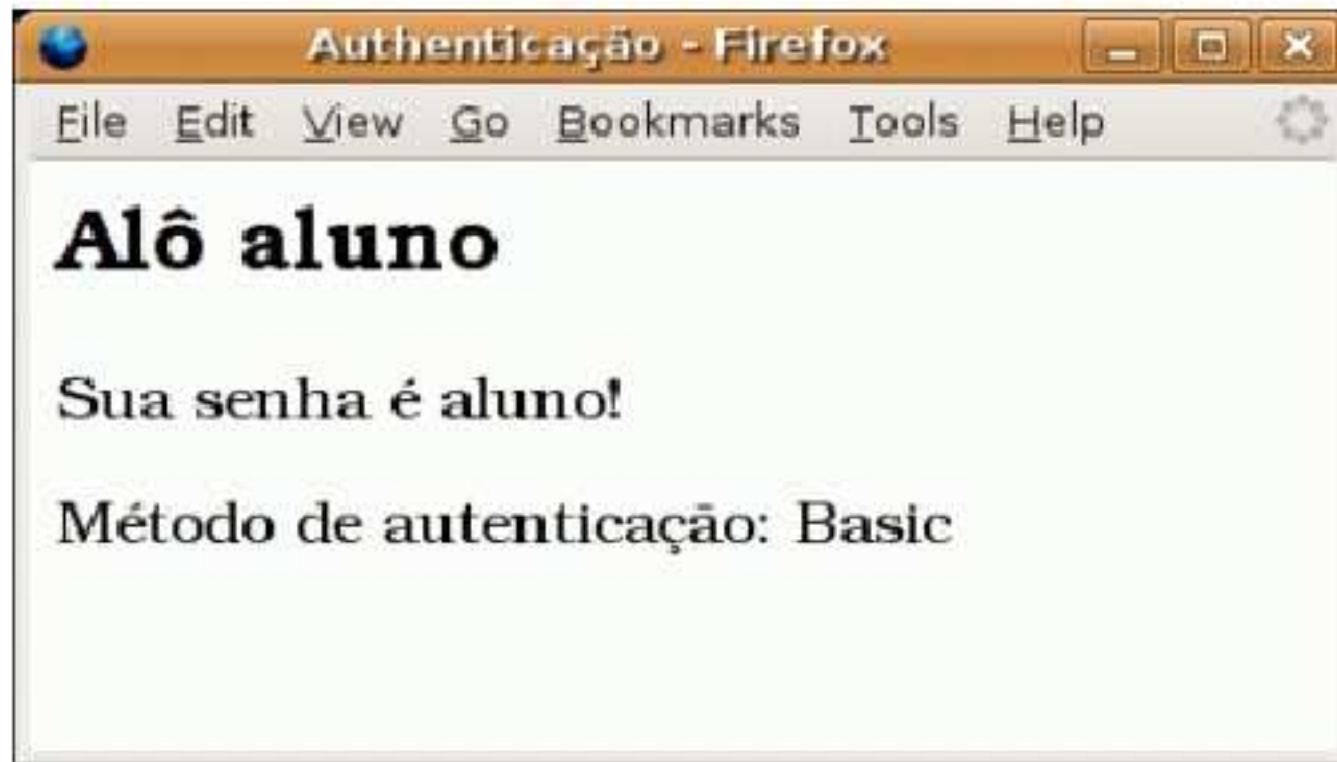
- As variáveis globais `$_SERVER['PHP_AUTH_USER']` e `$_SERVER['PHP_AUTH_PW']` são inicializadas com o nome e senha do usuário autenticado via http
- A variável `$_SERVER['AUTH_TYPE']` é inicializada com o tipo de autenticação usada (Basic por default)
- Essas variáveis podem ser usadas para refinar os procedimentos de segurança da aplicação

[Abrir editor](#)[Abrir navegador](#)

# Exemplo

```
<html>
  <head><title>Autenticação</title></head>
  <body>
    <h2>Alô
      <?php echo $_SERVER['PHP_AUTH_USER']; ?>
    </h2>
    <p>Sua senha é
      <?php echo $_SERVER['PHP_AUTH_PW']; ?>!
    </p>
    <p>Método de autenticação:
      <?php echo $_SERVER['AUTH_TYPE']; ?>
    </p>
  </body>
</html>
```

# Exemplo



# Autenticação http com PHP

- O mecanismo de autenticação http pode ser gerenciado pelo script PHP ao invés de usar os recursos do servidor Apache
- O próprio script envia o cabeçalho com o código de estado **Unauthorized**, solicita autenticação e testa o nome e senha preenchido na caixa de diálogo
- Elimina a necessidade de se criar o arquivo **.htaccess** ou gerar um arquivo de senhas
  - ▶ O script tem que ter seu próprio método de validar nomes e senhas
- Permite maior controle sobre autenticação
  - ▶ Usuários e senhas manuseados pelo próprio script
  - ▶ Pode-se definir grupos de usuários com privilégios distintos

# Pedindo autenticação via http

- Ao detectar que a pedido não vem com credenciais adequadas, o script deve enviar um cabeçalho solicitando autenticação
  - ▶ Usa-se a função `header()` do PHP
- O campo do cabeçalho para solicitar autenticação tem a forma:  
`WWW-Authenticate: tipo realm=nomeRecurso`
  - ▶ `tipo`: Basic ou Digest
  - ▶ `nomeRecurso`: string usada no diálogo de autenticação
- O código de estado 401 também deve ser enviado para indicar acesso não autorizado:  
`HTTP/1.0 401 Unauthorized`

# Exemplo

```
// autentica é uma função que retorna true ou false
if (!autentica ($_SERVER['PHP_AUTH_USER'],
                $_SERVER['PHP_AUTH_PW'])) {
    // Credenciais não encontradas.
    // Enviar cabeçalho pedindo autenticacao
    header("WWW-Authenticate: Basic " .
           "realm='Pagina Protegida'");
    header("HTTP/1.0 401 Unauthorized");
    // Mensagem exibida se usuario cancelar o diálogo
    echo "<H1> Pagina Protegida.</h1>";
    // Abortar processamento
    exit;
}
// Página propriamente dita segue ...
```

# Exemplo de função de autenticação

```
function autentica ($nome, $senha) {  
    // Retorna true apenas se nome e senha validos  
    if (isset($nome) && isset($senha)) {  
        // procurar nome e senha no array usuarios  
        global $usuarios;  
        foreach ($usuarios as $u)  
            if ($u == array($nome, $senha))  
                return true;  
    }  
    // Nao encontrado. Retornar falso  
    return false;  
}  
// Lista de usuarios autorizados  
$usuarios = array (array ("aluno", "aluno") ,  
                  array ("fulano", "xpto") ,  
                  ...);
```

# Armazenando uma tabela de usuários

- A maneira mais sensata de guardar uma tabela de usuários para autenticação é em um banco de dados
  - ▶ Consulta otimizada
  - ▶ Permite manutenção via web usando scripts PHP
- Exemplo:

```
CREATE TABLE usuarios (
    nome varchar(10) not null,
    senha varchar(15) not null,
    PRIMARY KEY (nome)
);
```

[Abrir editor](#)[Cria usuários](#)[Abrir navegador](#)

# Encriptando senhas

- Senhas são informações importantes e devem ser guardadas em forma encriptada
- Há duas maneiras:
  - ▶ Usando a função password do SQL
    - O servidor MySQL se encarrega de encriptar a senha
    - Exemplo:

```
UPDATE usuarios
SET senha = password('xpto')
WHERE nome = 'fulano'
```
  - ▶ Usando a função crypt do PHP
    - Senha é encriptada pelo script antes de ser guardada ou consultada no banco de dados
    - É uma função mais flexível que a função password

# A função crypt

- Formato:

`crypt (senha, sal)`

- ▶ *senha* é a string que se deseja encriptar
- ▶ *sal* é uma string de dois caracteres opcional usada para variar o resultado da encriptação
  - Por exemplo, se dois usuários têm a mesma senha, é desejável que a encriptação da mesma seja diferente para os dois.  
Pode-se usar o primeiros 2 caracteres do nome como sal
- O resultado é uma string de comprimento fixo encriptada usando a cifra configurada no PHP
  - ▶ Normalmente, o algoritmo DES
  - ▶ Em algumas instalações, o algoritmo MD5
- Não é possível decriptar para obter a senha original

# Exemplo: inserindo registros na tabela de usuários

```
function insere_usuario ($nome, $senha, $conexao) {  
    // Computar o sal para o crypt  
    $sal = substr($nome, 0, 2);  
    // Criar senha encriptada  
    $senha_enc = crypt($senha, $sal);  
    // Realizar inserção  
    mysql_query ("insert into usuarios  
        values ('$nome', '$senha_enc')", $conexao);  
}  
  
// Conexão com MySQL e banco de dados prog2  
$servidor = $_SERVER["REMOTE_ADDR"];  
$conexao = mysql_connect($servidor, "aluno", "aluno");  
mysql_select_db("prog2", $conexao)  
  
// Dois usuários inseridos na tabela  
insere_usuario ("aluno", "aluno", $conexao);  
insere_usuario ("fulano", "xpto", $conexao);
```

# Exemplo: autenticação usando tabela de usuários

```
function autentica ($nome, $senha, $conexao) {  
    if (isset($nome) && isset($senha)) {  
        // Computar o sal para o crypt  
        $sal = substr($nome, 0, 2);  
        // Criar senha encriptada  
        $senha_enc = crypt($senha, $sal);  
        // Realizar consulta  
        $consulta = "select * from usuarios where  
            nome = '$nome' and senha = '$senha_enc'";  
        $result = mysql_query ($consulta, $conexao);  
        // Retornar true somente se encontrado 1 nome  
        return (mysql_num_rows ($result) == 1);  
    }  
    return false;  
}
```

# Autenticando sem http

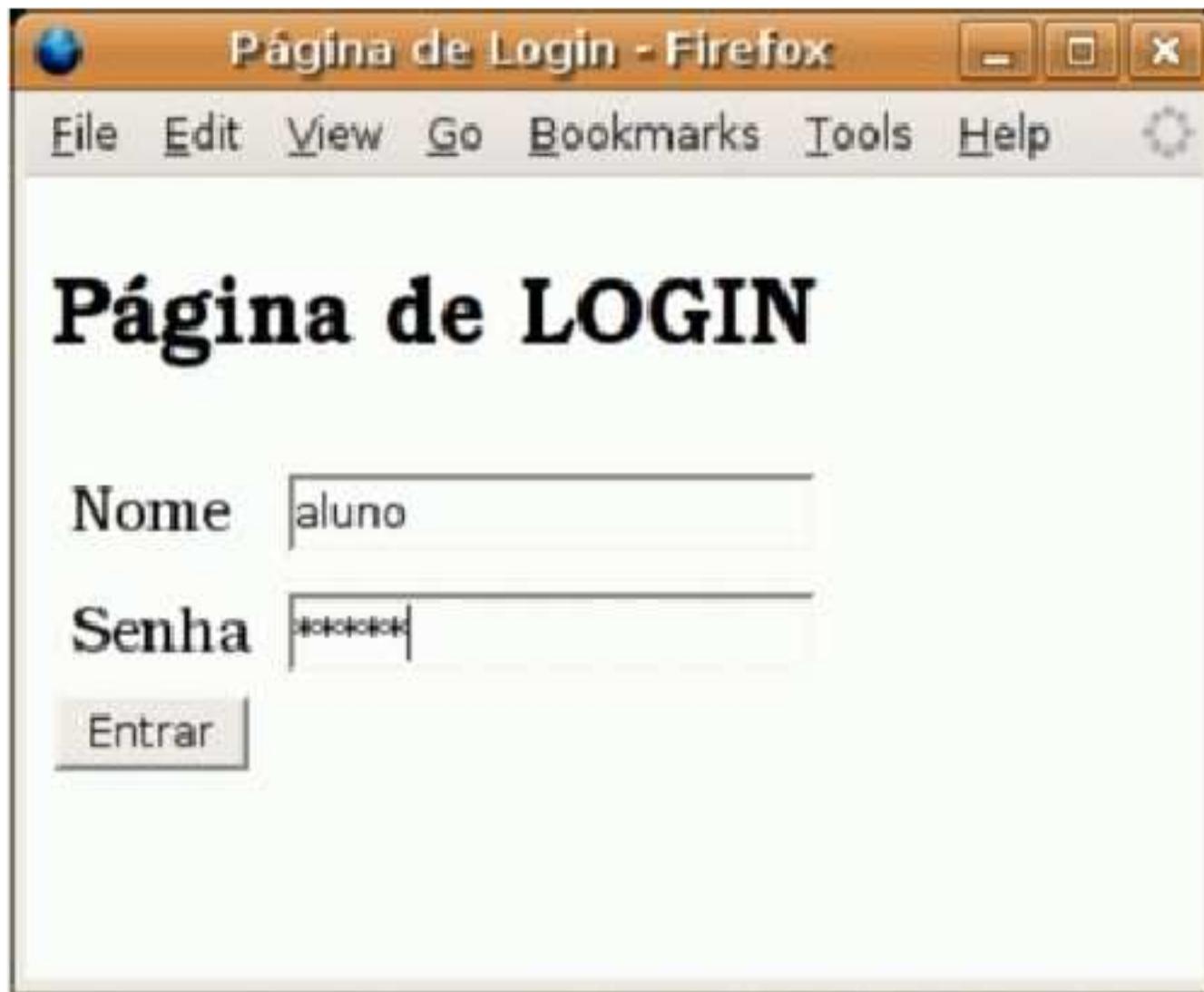
- O mecanismo de autenticação do http é simples mas inadequado em alguns casos
  - ▶ O navegador lembra das credenciais até ser terminado
    - Um outro usuário usando o mesmo navegador poderá acessar a página sem se autenticar
  - ▶ A caixa de diálogo de autenticação tem formato fixo
    - É impossível acrescentar outro campo que não nome/senha
    - Não se pode aproveitar a página de login para exibir outras informações (instruções, anúncios, etc)
- Uma solução é realizar a autenticação usando variáveis de sessão

[Abrir editor](#)[Abrir navegador](#)

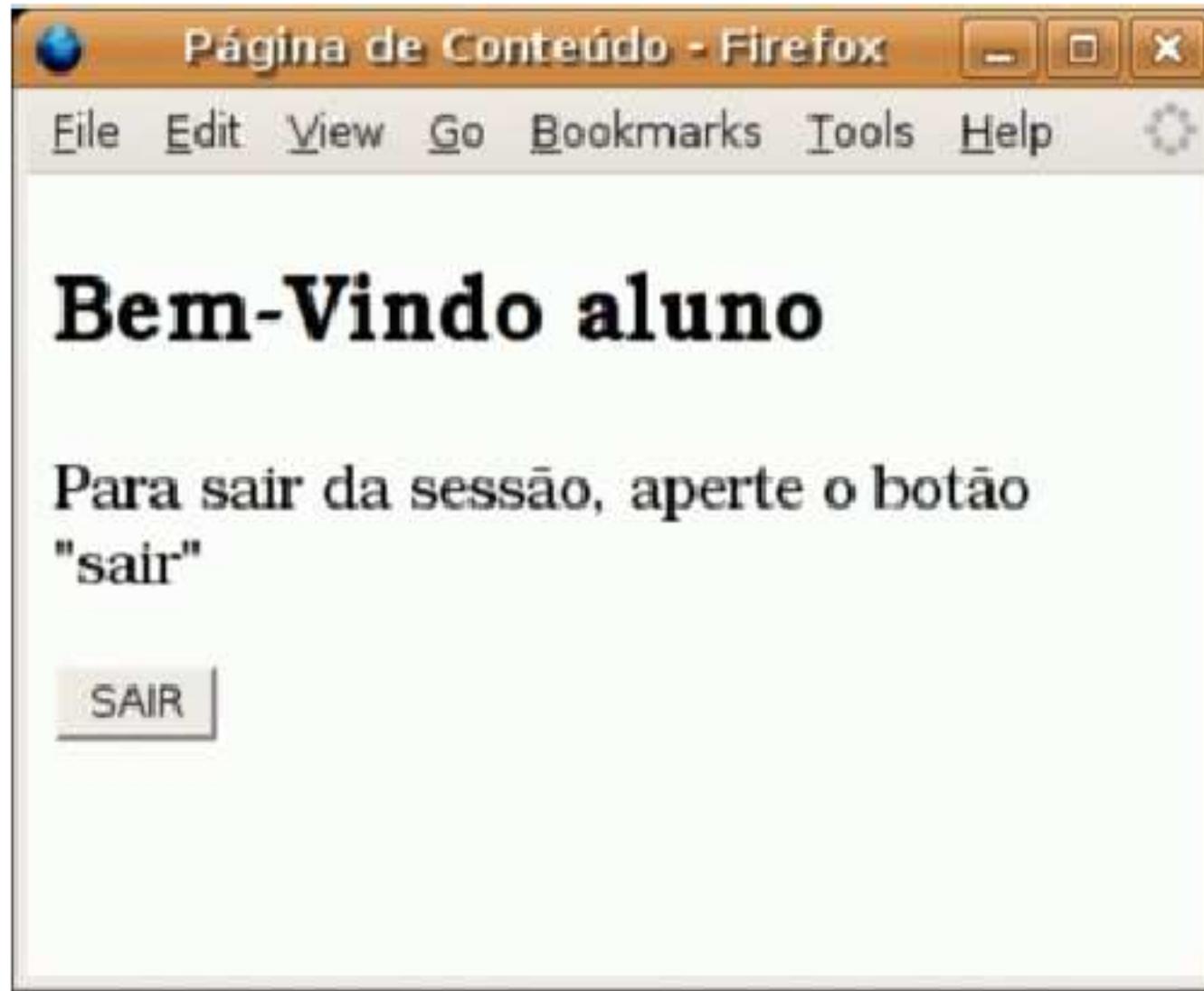
# Autenticando com sessões

- Usa-se uma variável de sessão para registrar o usuário autenticado
- Se essa variável está definida, mostra-se o conteúdo protegido
  - ▶ Um botão ou um link pode ser exibido para permitir encerrar a sessão
- Caso contrário, exibe-se uma tela de login com um formulário contendo campos para nome e senha
  - ▶ Ao submeter o formulário, o código de autenticação visto anteriormente é usado para verificar as credenciais
  - ▶ Caso as credenciais estejam corretas, exibe-se o conteúdo protegido

# Exemplo



# Exemplo



# Exemplo



# Exemplo

```
session_start ();
include "autenticacao.inc"; //código de autenticação
if (isset ($_POST["logout"])) {
    // Usuário apertou o botão logout
    session_destroy ();
    pagina_logout ();
} elseif (isset ($_SESSION["usuario"])) {
    // Usuário já autenticado
    pagina_conteudo ();
} elseif (autentica ($_POST["nome"],
                     $_POST["senha"], $conexao)) {
    // Lembrar usuário e exibir conteúdo
    $_SESSION["usuario"] = $_POST["nome"];
    pagina_conteudo ();
} else {
    // Usuário não autenticado: exibir login
    pagina_login ();
}
```

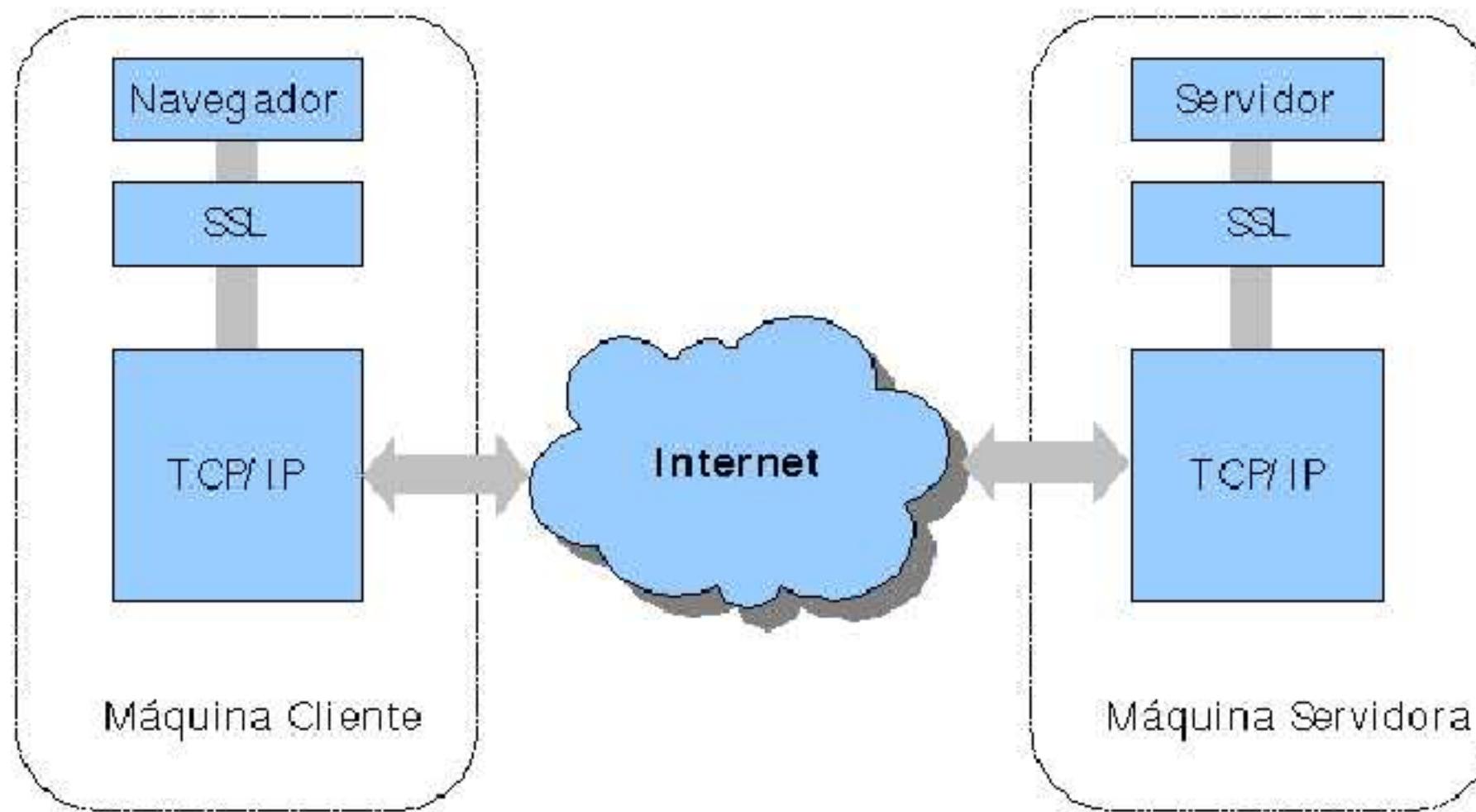
# Exemplo

- As funções `pagina_login()`, `pagina_logout()` e `pagina_conteudo()` exibem os conteúdos correspondentes às fases da autenticação:
  - ▶ `pagina_login()` exibe um formulário com os campos nome e senha
  - ▶ `pagina_conteudo()` exibe o conteúdo que só é visível para usuários autenticados
    - Um formulário com um único botão "SAIR" é exibido para permitir encerrar a sessão
  - ▶ `pagina_logout()` exibe uma mensagem de finalização da sessão

# O protocolo SSL

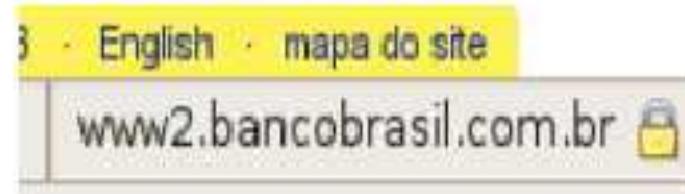
- Tráfego entre o servidor e o navegador pode em algumas circunstâncias ser capturado por um *hacker*
- O protocolo **Secure Sockets Layer** permite a encriptação de todo o tráfego de dados entre servidor e navegador
  - ▶ Mesmo que o *hacker* consiga capturar o tráfego, as informações estarão ilegíveis
- O protocolo SSL consiste de uma camada entre o navegador/servidor e os protocolos de comunicação TCP/IP que efetivamente são encarregados de transportar os dados entre as máquinas

# O protocolo SSL



# Servindo páginas seguras com Apache

- Páginas encriptadas têm URLs iniciando com `https://`
- Navegadores identificam páginas encriptadas com um cadeado na barra de estado:



- Diferentemente do protocolo http normal que é transmitido por default pela porta 80, páginas encriptadas com SSL são normalmente servidas pela porta 443
- Para servir páginas encriptadas com SSL, o Apache tem que ser configurado
  - ▶ (Está fora do escopo deste curso - veja em <http://www.apache.org>)

# Certificados Digitais

- Um passo importante da configuração do SSL é o estabelecimento de um *certificado digital*
- Um certificado digital contém uma assinatura de autenticidade, isto é, um código emitido por uma *autoridade certificadora*
  - ▶ Existem várias!
  - ▶ Funcionam como "cartórios digitais"
  - ▶ Dão fé pública de que o certificado é original e identifica o sítio como autêntico
- Pode-se usar um certificado digital assinado por qualquer um, mas não há garantia contra impostores