DESENVOLVIMENTO DE SOFTWARE PARA WEB

CAPÍTULO 3 - COMO VINCULAR BANCOS DE DADOS ÀS PÁGINAS?

Fernando Skackauskas Dias

Introdução

É praticamente impossível imaginar a maioria das atividades da sociedade atual sem páginas *web*. É pelos *sites* que fazemos as matrículas nas instituições de ensino, pagamos as contas, acessamos o saldo bancário, buscamos informações, marcamos uma consulta médica, agendamos viagens, compramos qualquer tipo de produto, entre diversos outros serviços possíveis. Uma página *web*, na atualidade, é quase uma extensão das nossas vidas. Então, é possível perceber a quantidade enorme de informações que são transacionadas entre os diversos servidores espalhados pelo mundo, como também a quantidade crescente de informações que estão armazenadas nos bancos de dados.

Mas, no início da popularização da *web*, não era bem assim. As páginas serviam somente para exibir as informações e produtos, e o usuário via as páginas como um simples emissor de informações, não existindo nenhuma interação com o sistema. Ou seja, não era possível comprar, cadastrar ou realizar qualquer tipo de troca de informações entre o usuário e a organização que oferecia o sistema pela internet.

Passada esta época, hoje realizamos quase tudo por meio de um site e nele, as informações são gravadas, recuperadas e exibidas. Para que tudo isso ocorra, é necessário que os dados estejam disponíveis em bancos de dados. Mas como é possível realizar os processos de ler, gravar e recuperar informações das páginas *web*? Quais são os comandos? Como é possível manipular e gerenciar as informações nos bancos de dados?

Após a leitura deste capítulo você será capaz de responder estas questões e criar páginas *web* realizando processos de gravação e recuperação de dados por um *site.*

Bons estudos!

3.1 Desenvolvimento *web*: cadastrar registro no banco de dados

Ao iniciar nossos estudos sobre os cadastros de registros em bancos de dados na *web*, é necessário compreender alguns conceitos fundamentais. A primeira distinção necessária é saber que existem páginas estáticas e páginas dinâmicas. As páginas estáticas são consideradas aquelas páginas que não se alteram quando recarregadas automaticamente, mesmo quando o usuário executa alguma função. Páginas dinâmicas são as páginas em que as informações são carregadas automaticamente sem a intervenção do usuário, como também quando existe uma interação entre o usuário e o sistema, como, por exemplo, cadastro de informações, recuperação de informações, etc. Na figura a seguir vemos o exemplo de uma página dinâmica, pois, mesmo não existindo interação com o usuário, ela é recarregada automaticamente quando uma nova informação é inserida.

Figura 1 - Exemplo de uma página dinâmica de um jornal dos Estados Unidos, na qual as informações são carregadas automaticamente.
Fonte: LEMAY; COLBURN; TYLER, 2002, p. 9.

Na figura a seguir vemos um exemplo de uma página dinâmica que possui um nível de interação com o usuário, por meio de inserção e recuperação de informações.

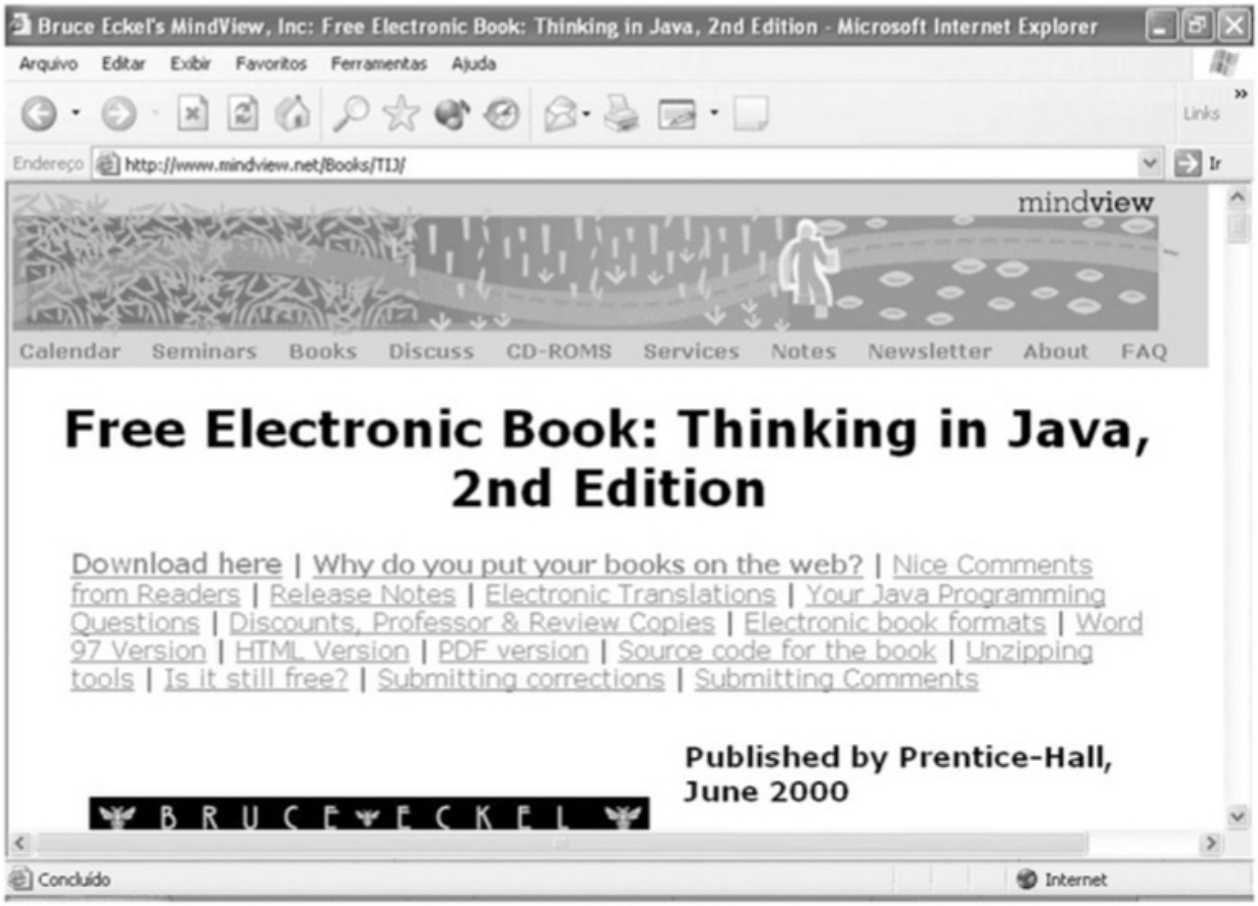

Figura 2 - Exemplo de uma página dinâmica de uma empresa de revisão de textos e livros eletrônicos, na qual existe uma interação com o usuário.
Fonte: LEMAY; COLBURN; TYLER, 2002, p. 8.

Mas, como desenvolver uma página *web,* que permita inserir informações, como, por exemplo, um cadastro de cliente? Quais são os comandos fundamentais? Como é possível interagir em uma mesma página HTML com os comandos de inserção em um banco de dados? Veremos a seguir.

3.1.1 Insert

O primeiro passo a ser dado é compreender como podemos obter os dados de uma página HTML e salvar em um banco de dados. Para isso, vamos entender o que é e como funciona uma página HTML. A linguagem *Hyper Text Markup Language* (HTML), segundo Giroldo e Fressati (2015, p. 3) é "baseada em *tags* ou marcações que teve sua criação em 1990 por Tim Berners-Lee. Essas páginas ou textos são interpretadas por um navegador ou browser também chamado de cliente".
Portanto, HTML é uma linguagem que tem o objetivo de criar sites. Esses *sites* podem ser visualizados por qualquer navegador conectado à internet. As marcações (*tags*) são comandos que permitem a criação de páginas HTML. A linguagem HTML consiste em uma série de códigos curtos em um arquivo de texto, marcados por tags. Portanto, HTML é uma linguagem, composta por um código e tem uma sintaxe como qualquer outra linguagem. A seguir é apresentado um exemplo de código HTML. Vamos analisá-lo.

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
<title>Desenvolvimento de Software para Web</title>
</head>
```

<body>
<h1>Aqui terá um texto grande</h1>
Aqui vai o código que fará seu site aparecer
</body>
</html>

É possível compreender o documento em HTML como sendo uma divisão de blocos de *tags*, conforme a seguinte estrutura (W3SCHOOLS, 2019):

- <!DOCTYPE html> declaração que define este documento como HTML;
- <html> é o elemento raiz de uma página HTML. lang="pt-br" define o idioma;
- <head> elemento que contém as informações sobre o documento;
- <title> elemento que especifica um título para o documento;
- <body> elemento que contém todo o conteúdo da página *web*;
- <h1> elemento que define um cabeçalho de tamanho grande;
- elemento que insere um parágrafo.

Na figura a seguir tem uma demonstração de como seria esta página HTML em um navegador do código acima.

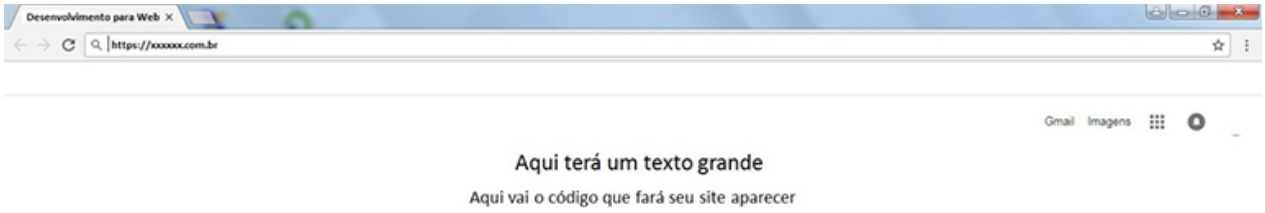

Figura 3 - Demonstração de uma tela no navegador com o código HTML básico e estático, sem interação do usuário com a página.
Fonte: Elaborada pelo autor, 2019.

Este código HTML pode ser considerado um código estático, pois não existe nenhuma interação com o usuário e nem atualização dinâmica. Ou seja, não recebe nenhuma informação e não executa nenhuma interação com banco de dados. Para que possamos iniciar a criação de uma página dinâmica, é necessário que haja uma entrada de dados no formulário. Para isso, vamos criar uma página HTML de Cadastro de Cliente de uma loja fictícia que chamaremos de "Lojas Super". Nela teremos a inclusão do nome do cliente, endereço, idade e sexo. Para isso é necessário conhecer algumas *tags*, segundo Lemay e Tyler (2002, p. 142). Clique para conhecê-las.

Define uma descrição de um texto e sua configuração.

<input></input>
Especifica um campo de entrada no qual o usuário pode inserir dados.
Neste caso, como iremos criar um cadastro de clientes, vamos modificar os elementos do exemplo anterior e incluir as tags para a entrada de dados, ficando da seguinte maneira:

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
<title>Lojas Super</title>
</head>
<body>
<h1>Cadastro de Clientes</h1><p></p>
<p>Inclua as informações do cliente abaixo</p>
<font> Nome do cliente: </font><p></p>
<input type=”text” id=”nome”></input><p></p>
<font> Endereço do cliente: </font><p></p>
<input type=”text” id=”endereco”></input><p></p>
<font> Idade do cliente </font><p></p>
<input type=”text” id=”idade”></input><p></p>
<font> Sexo do cliente </font><p></p>
<input type=”radio” name=“sexo” id=”masculino” checked="checked"></input><p></p>
<input type=”radio” name = “sexo” id=”feminino”></input><p></p>
<input type="button" name="botao-ok" value="Cadastrar" id=”botao_cadastra”>
</body>
</html>
```

Vamos analisar as *tags* do código acima. Clique nas abas e acompanhe essa análise.

- <title> Lojas Super </title>

  É o texto que será exibido na aba do navegador.
- <h1> Cadastro do Cliente <h1>

  Vai aparecer como um cabeçalho.
- Inclua as informações do cliente

  Será um parágrafo com a descrição inserida.
- Nome do Cliente

  Será um rótulo da entrada de dados.
- <input type=“text” id=”nome”></input>

Vai aparecer na tela do navegador como um campo para entrada de dados. O atributo type="text" indica que na entrada do campo pode ser digitado qualquer caractere (letras, números e caracteres especiais).

- Endereço do cliente

  Terá o mesmo comportamento do *input* "nome".

- <input type="text" id="idade"></input>

  Terá o mesmo comportamento do *input* "nome".

- <input type="radio" name = "sexo" id="masculino" checked="checked"></input>
  <input type="radio" name = "sexo" id="feminino"></input>

  Este *input* terá um comportamento diferente. O atributo type="radio" exibirá no navegador uma opção a ser escolhida (Masculino ou Feminino). Para que, quando o usuário clicar em uma opção, e a outra opção seja desabilitada, é necessário colocar a *tag* "name", com a mesma descrição para ambos. Outra informação importante é a atributo "checked", que irá colocar a opção "Masculino" como previamente marcado.

- <input type="button" id="button-ok" value="Cadastrar"></input>

  Este *input* será o botão que deverá ser acionado quando o cliente for efetuar o cadastro. Perceba que o type="button" fará com que a *tag* seja um botão e o atributo value="Cadastrar" será o texto exibido no botão.

É importante notar que, em todos os *inputs*, existe o atributo "id". Este atributo é fundamental para designar que aquele campo é único no código HTML e serve para identificá-lo ao acessar o seu conteúdo, quando necessário.
Na figura a seguir tem uma demonstração de como seria esta página HTML em um navegador com o código acima.

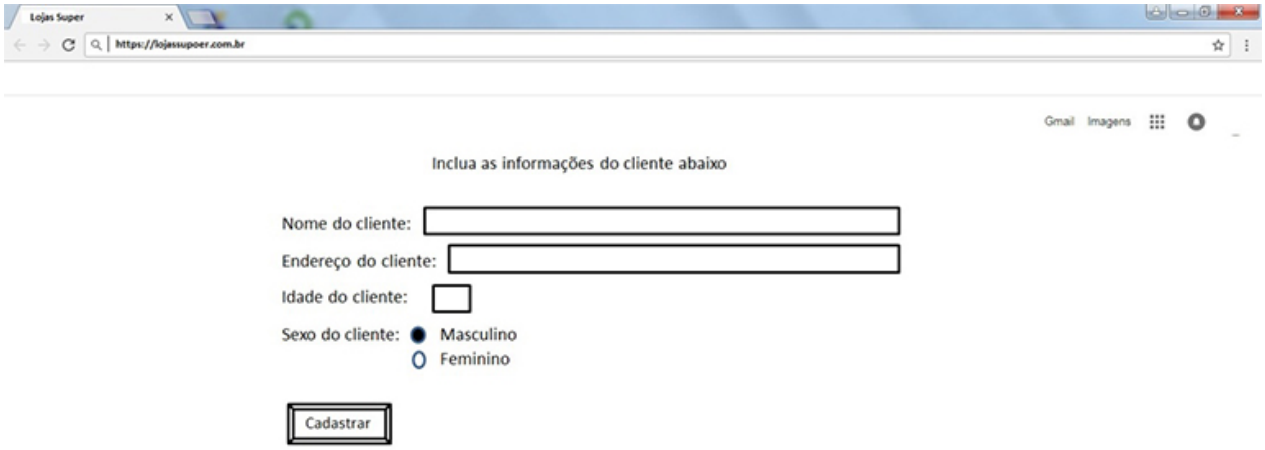

Figura 4 - Demonstração de uma tela de cadastro de clientes de uma loja desenvolvida com o código HTML.
Fonte: Elaborada pelo autor, 2019.

Continuando a elaboração do nosso cadastro de cliente, após o usuário efetuar a entrada dos dados necessários, é preciso fazer a inclusão de um comando para que o programa "leia" as informações cadastradas e as inclua no banco de dados. Para isto, na *tag* do "button-ok" deve ser incluído o atributo "onClick" direcionando para uma função que recupere as informações digitadas. Esta *tag* ficaria, então:

<input type="button" name="botao-ok" value="Cadastrar" id="botao_cadastra" onClick="Cadastro()">

Observe que o atributo se refere a uma descrição de uma função a ser incluída no código com abre-e-fecha parênteses.

Agora, vamos compreender como uma página HTML pode interagir com o <script>. A <script> *tag* é usada para definir um script do lado do cliente (JavaScript). O <script> elemento contém instruções de *script* ou aponta para um arquivo de *script* externo por meio do *src* atributo. Usos comuns para JavaScript são manipulação de imagem, validação de formulário e alterações dinâmicas de conteúdo (W3SCHOOLS, 2019).

Ou seja, haverá uma interação entre o código HTML e comandos JavaScript para que uma página estática se torne uma página dinâmica. O *script* é escrito pela sintaxe <script> comandos JavaScript </script> e deverá residir na área do <head> do código HTML.

Para que o código consiga acessar as informações inseridas pelo usuário, é necessário utilizar o comando: document.getElementById("id do campo").value. Este comando em partes. Clique para conhecê-las.

document

Se refere à página onde estão contidas as informações.

getElementById(id do campo")

Indica que as informações serão recuperadas pelo "id" dos campo contido no document (HTML).

.value

Indica que será recuperado o conteúdo do campo que foi referenciado.

É importante ressaltar que o resultado deste comando deverá ser armazenado em uma variável declarada como "var" que poderá ser declarada junto ao comando. No código a seguir, temos o nosso cadastro de cliente com o *script* de acesso às informações:

<!DOCTYPE html>

```
<html lang="pt-br">
<head>
<title>Lojas Super</title>
<script>
function Cadastro() {
var nome_cliente = document.getElementById("nome").value;
var endereco_cliente = document.getElementById("endereco").value;
var idade = document.getElementById("idade").value;
if (document.getElementById("masculino").checked)
{
var sexo_cliente = “M”;
}
else
{
var sexo_cliente = “F”;
}
}
</script>
</head>
<body>
<h1>Cadastro de Clientes</h1>
<p>Inclua as informações do cliente abaixo</p>
<font> Nome do cliente: </font><p></p>
<input type=”text” id=”nome”></input><p></p>
<font> Endereço do cliente: </font><p></p>
<input type=”text” id=”endereco”></input><p></p>
<font> Idade do cliente </font><p></p>
<input type=”text” id=”idade”></input><p></p>
<font> Sexo do cliente </font><p></p>
<input type=”radio” name=“sexo” id=”masculino” checked="checked"></input><p></p>
<input type=”radio” name = “sexo” id=”feminino”></input><p></p>
<input type="button" name="botao" value="Cadastrar" id=”botao_cadastra” onClick=”Cadastro()>
</body>
</html>
```

Agora, vamos analisar o código. Os conteúdos dos campos nome, endereço e idade foram acessados diretamente pelo seu valor (*value*), sendo armazenado nas variáveis para posterior inclusão no banco de dados.

CASO

Com o propósito de atender à diversidade de preferências e necessidades de seus clientes, uma grande instituição bancária no Brasil foi, nos últimos anos, disponibilizando serviços financeiros por meio de vários canais de comunicação. Ocorria um problema no sentido de que nem todos os canais disponibilizavam todos os tipos de serviços, além do que, existiam diferentes agrupamentos em um mesmo canal, em função do tipo de cliente.

Para solucionar este problema, com o esforço para integrar múltiplos sistemas de informação a múltiplos canais, foi implantada uma solução de arquitetura de *site* específico, promovendo a integração entre o negócio e a TI por meio de conjunto de interfaces de serviços acoplados. A implantação de uma arquitetura de *site* específica para atender aos negócios da empresa adicionou valor aos negócios da instituição bancária, sendo um recurso estratégico para o aumento da competitividade da organização.

Mas o campo sexo, como era um tipo "*radio*", não teria como saber diretamente se é "Masculino" ou "Feminino". No código, então, verificamos se um deles (no caso o masculino) estava marcado pelo atributo "*checked*". Ou seja, se o campo com o id="masculino" retornou "*true*" para o comando "*if*", então é porque ele estava marcado. Dessa forma, guardamos o conteúdo "M" em uma variável criada, "sexo_cliente". Se não estiver marcado, então o que estaria "*checked*", seria, portanto, o sexo feminino.

Bom, para que possamos prosseguir com o desenvolvimento do nosso código é necessário acessar o banco de dados e gravar as informações que foram lidas do formulário HTML.

Neste caso, como estamos diante de um processamento síncrono, ou seja, os comandos seguem determinada ordem, precisamos utilizar algum recurso para a gravação no banco de dados, pois este é um processamento assíncrono (JQUERY, 2019). Neste caso, será necessário utilizar um recurso chamado "Ajax" (*Asynchronous JavaScript and XML*). O Ajax utiliza o JavaScript, XML e HTML de forma dinâmica. Uma característica do Ajax é que as solicitações são assíncronas. Isso quer dizer que o Javascript de um aplicativo Ajax se comunica assincronamente com o servidor. No nosso código, vamos inserir o comando Ajax.

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
<title>Lojas Super</title>
<script>
function Cadastro() {
var nome_cliente = document.getElementById("nome").value;
var endereco_cliente = document.getElementById("endereco").value;
var idade = document.getElementById("idade").value;
if (document.getElementById("masculino").checked)
{
var sexo_cliente = "M";
}
else
{
var sexo_cliente = "F";
}
```

```
var dadosajax = {
'nome_cliente': nome_cliente,
‘endereco_cliente’: endereço_cliente,
‘idade_cliente’: idade_cliente,
‘sexo_cliente’: sexo_cliente
};
var pageurl = grava_cliente.php';
$.ajax({
url: pageurl,
data: dadosajax,
type: 'POST',
cache: false,
error: function(XMLHttpRequest, textStatus, errorThrown)
{
alert(“Erro Gravação Cliente!!”);
},
success: function(result)
{
alert("Gravação Cliente efetuada com sucesso");
return;
}
}
</script>
</head>
<body>
<h1>Cadastro de Clientes</h1>
<p>Inclua as informações do cliente abaixo</p>
<font> Nome do cliente: </font><p></p>
<input type="text" id="nome"></input><p></p>
<font> Endereço do cliente: </font><p></p>
<input type="text" id="endereco"></input><p></p>
<font> Idade do cliente </font><p></p>
<input type="text" id="idade"></input><p></p>
<font> Sexo do cliente </font><p></p>
<input type="radio" name="sexo" id="masculino" checked="checked"></input><p></p>
<input type="radio" name = "sexo" id="feminino"></input><p></p>
<input type="button" name="botao" value="Cadastrar" id="botao_cadastra" onClick="Cadastro()>
</body>
</html>
```

Agora, vamos analisar a parte do código referente aos comandos Ajax. Clique e acompanhe a análise.

Inicialmente declaramos uma variável “dadosajax”, que contém todas as informações do cliente recuperadas do formulário HTML.

A seguir declaramos uma variável “pageurl” que vai indicar o nome do código que efetua a gravação dos dados. Neste caso, criamos um código chamado “grava_cliente.php”.

A seguir fazemos a declaração da ação Ajax, colocando o símbolo “$.” que indica ser um comando JQuery.

O jQuery é considerado uma biblioteca eficaz e com muitos recursos disponíveis. A manipulação dos dados dos documentos HTML é executada de forma simples, como também a manipulação de eventos, sendo uma API que funciona em vários navegadores (JQUERY, 2019).

No corpo do comando JQuery inserimos o código PHP que executará a gravação, os dados que serão gravados, o tipo de processamento (*post*) e a indicação que não há cache de processamento.
Na sequência destes comandos, temos as duas condições de sucesso ou de fracasso da execução do programa "grava_cliente.php".
E assim criamos o nosso código em php que executará a gravação do cliente.

PHP

O PHP (*Hypertext Preprocessor*) é considerado uma linguagem de *script* de código aberto para ser utilizada em diversas plataformas, sendo indicada principalmente para o desenvolvimento de páginas *web* (PHP, 2019).
Ou seja, o PHP é executado no servidor e é processado na hora em que a página é executada, que é o que precisamos para gravar os dados dos nossos clientes. O que caracteriza o PHP é o fato de ser executado no lado do servidor, e o JavaScript é executado no lado do cliente. O navegador, que disparou o processamento do PHP, recebe como retorno os resultados da execução do seu *script.* Este é, portanto, um processamento assíncrono. Os comandos do JavaScript, como são executados no cliente, é um processamento síncrono. Estes são os fundamentos do sincronismo e assincronismo das páginas *web* e o documento HTML. Agora, vamos criar o nosso código PHP (grava_cliente.php) e analisar cada comando inscrito nele.

```
<?php
$host = "servidor_banco_de_dados.com.br";
$user = "usuario_banco_de_dados";
$pass = "senha_banco_de_dados";
$db = "banco_de_dados";
$conecta = mysql_connect($host, $user, $pass) or print (
mysql_select_db($db, $conecta);
$nome_cliente = $_REQUEST['nome_cliente'];
$endereco_cliente = $_REQUEST['endereco_cliente'];
$idade_cliente = $_REQUEST['idade_cliente'];
$sexo_cliente = $_REQUEST['sexo_cliente'];
$sql = "INSERT INTO Cliente
(NomeCliente,
EnderecoCliente,
IdadeCliente,
SexoCliente,
EnderecoCliente,)
VALUES
('$nome_cliente',
'$endereco_cliente',
'$idade_cliente',
'$sexo_cliente');
$query = mysql_query($sql);
?>
```

Então, agora vamos fazer uma análise detalhada de cada comando deste código. Todo código PHP deve iniciar com o símbolo "**<?php**" e terminar como símbolo "**?>**". Tudo o que estiver inserido entre estes dois símbolos será interpretado como um comando PHP.
Em seguida são declaradas as variáveis que farão acesso ao banco de dados. Toda variável em PHP é declarada pelo símbolo "**$**". Neste caso, declaramos as quatro variáveis necessárias para se acessar o banco de dados: o servidor ($host), o usuário do banco de dados ($user), a senha para acessar o banco ($pass) e a base de dados a ser manipulada ($db). Portanto, é necessário ter um cadastro em um servidor de banco de dados que repassa as informações reais para acesso.

VOCÊ QUER VER?

A palestra *PHP & Segurança: Blindando Aplicações Web* proferida por Rafael Jaques (2017), na iMasters PHP Experience 2017, em São Paulo, demonstra a importância da segurança do PHP e das aplicações web atualmente, uma época repleta de ataques de *hackers*, invasões e roubo de informações. As empresas precisam estar preparadas para prevenir danos em ataques, pois as informações são uma propriedade de alto valor. Assista: https://www.youtube.com/embed/KmShf5VDGUM

O próximo comando é o acesso efetivo ao banco de dados. Para isto, o PHP disponibiliza o comando "mysql_connect". Este comando, juntamente com os parâmetros já declarados anteriormente, fará a conexão com o banco. Caso ocorra algum erro, ele será retornado pela função do PHP "my_sql_error()". Percebam que o retorno do sucesso (ou do fracasso) da conexão com o banco de dados estará armazenado na variável "$conecta". Em seguida é realizado o comando de acesso à base de dados, feito pelo comando "mysql_select_db", e nele são passados como parâmetros a base de dados e a conexão efetuada anteriormente.
O próximo passo é recuperar as informações do cliente que foram passadas pelo comando Ajax (dadosajax). O comando $_REQUEST[] contém as informações que foram passadas. O nome contido no $_REQUEST deve ser idêntico ao que foi informando pelo Ajax. Novamente, guardamos estas informações em uma variável. Por exemplo, $nome_cliente = $_REQUEST['nome_cliente'].

VOCÊ QUER LER?

O livro "Use a Cabeça! SQL" (LYNN, 2016) ensina SQL de forma prática e de fácil leitura. Este livro aborda de forma prática e simples os principais comandos do SQL, sua aplicabilidade e recursos. É recomendado para os iniciantes em desenvolvimento para *web*.

Seguindo o entendimento do código, agora iremos criar o código que contém o comando SQL para inserção das informações no banco de dados. SQL é uma linguagem de programação designada para manipular dados em um banco com sintaxe própria. SQL é sigla inglesa de "*Structured Query Language*" que significa, uma linguagem de consulta estruturada. Esta linguagem tem como objetivo executar o gerenciamento e manipulação das informações contidas em banco de dados relacionais (MYSQL, 2019).
Neste caso, estamos usando o comando **INSERT.** Este comando tem a seguinte sintaxe:
INSERT INTO <nome da tabela> (<campos da tabela onde serão gravados os dados separados por ",">) **VALUES** (<valores que serão gravados nos campos separados por ",">); .
É importante ressaltar que os campos da tabela onde serão gravados os dados e os valores a serem gravados deverão estar na mesma ordem.
Conforme visto, este comando é armazenado em uma variável, que, no nosso exemplo, chamamos de $sql. Por fim, usamos o comando "mysql_query()" para, efetivamente, executar o INSERT do sql que foi montado. Neste ponto encerramos todos os procedimentos para se efetuar a inclusão de registro de informações em um banco de dados por uma página HTML.

Agora, veremos como listar no *site* as informações contidas em um banco de dados.

3.2 Desenvolvimento *web*: listar registro no banco de dados

Quando temos uma série de informações gravadas em algum banco de dados, geralmente, estas informações necessitam ser exibidas em forma de consulta. Portanto, após a inclusão das informações, como é possível recuperá-las e exibi-las no navegador? Como seria uma estrutura HTML e quais os comandos PHP e SQL que fariam esta função? Vamos compreender a partir de agora.

3.2.1 Select

Para que possamos listar uma série de informações a partir de um banco de dados e exibir no navegador pelo código HTML, é necessário compreender como recuperar as informações no banco de dados. Para isso, vamos continuar no nosso sistema de clientes, agora listando todos os clientes cadastrados e exibi-los na tela. Vamos criar um código HTML que execute esta função, conforme mostrado abaixo.

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
<title>Lojas Super</title>
</head>
<body>
<h1>Listagem de Clientes</h1>
<p>Inclua as informações do cliente abaixo</p>
<input type="button" name="botao-ok" value="Listar" id=“botao_cadastra”>
<table>
<tr>
<th>Nome</th>
<th>Endereço</th>
<th>Idade</th>
<th>Sexo</th>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
</tr>
</table>
</body>
</html>
```

Na figura a seguir temos uma demonstração de como seria esta página HTML com o código acima.

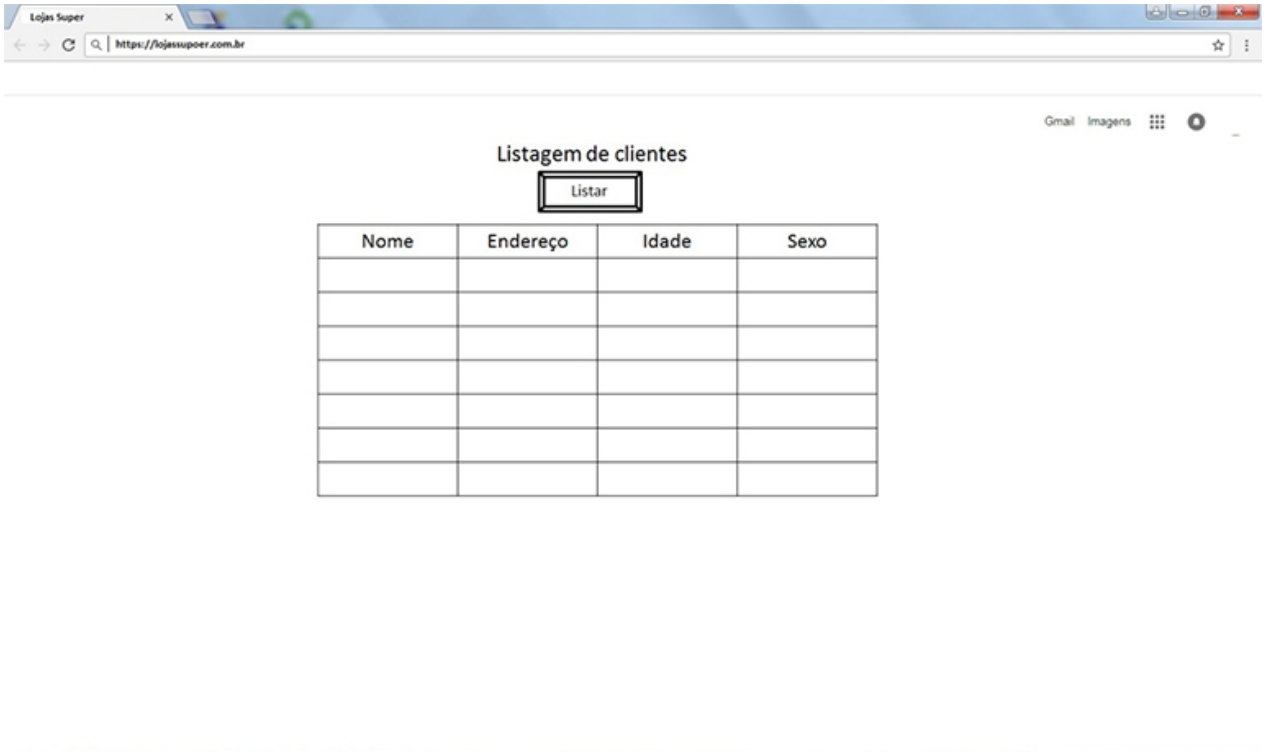

Figura 5 - Demonstração de uma tela de listagem de clientes de uma loja desenvolvida com o código HTML.
Fonte: Elaborada pelo autor, 2019.

Vamos, agora, analisar o código HTML. Como podemos ver, teremos no código somente o botão que dispara a pesquisa dos clientes no banco de dados e irá exibir as informações destes clientes na tabela, logo abaixo do botão. A tabela, em HTML, é inserida entre as *tags* <table></table>. Os cabeçalhos estão entre as *tags* <th></th>. Por fim, as células onde estarão contidas as informações recuperadas, estarão entre as *tags* <td></td>.

VOCÊ QUER LER?

O livro "Javascript e Jquery - Desenvolvimento de Interfaces Web Interativas - Guia Prático" (DUCKETT, 2015) ensina de forma clara e concisa como utilizar JavaScript e Jquery no desenvolvimento de páginas *web*. O JavaScript e o jQuery possibilitam executar funções assíncronas em páginas *web* desenvolvidas em HTML. Somente desta forma é possível ter acesso aos dados inseridos e bancos de dados e exibi-los em tempo real nas páginas web. Este livro aborda de forma acessível e de fácil entendimento os principais recursos JavaScript e jQuery.

Portanto, para que se processe a recuperação das informações no banco de dados, é necessário incluir o <script> e a função Ajax para acesso e recuperação das informações, conforme está descrito no código a seguir.

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
<title>Lojas Super</title>
<script>
```

```
function Lista() {
var pageurl = lista_cliente.php';
$.ajax({
url: pageurl,
type: 'POST',
cache: false,
error: function(XMLHttpRequest, textStatus, errorThrown)
{
alert("Erro Gravação Cliente!!");
},
success: function(result)
{
var tabelas_cliente = document.getElementById("tabela_coordenadas");
var i = result.lenght;
for(i=0;i<final;i++)
{
var nome = result(i,0);
tabela_cliente.cell = nome;
var endereco = result(i,1);
tabela_cliente.cell = endereco;
var idade = result(i,2);
tabela_cliente.cell = idade;
var sexo = result(i,3);
tabela_cliente.cell = sexo;
}
}
}
</script>
</head>
<body>
<h1>Listagem de Clientes</h1>
<p>Inclua as informações do cliente abaixo</p>
<input type="button" name="botao-ok" value="Listar" id=”botao_cadastra” onClick=”Lista()”>
<table id=”tabela”>
<tr>
<th>Nome</th>
<th>Endereço</th>
<th>Idade</th>
<th>Sexo</th>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
</tr>
</table>
```

```
</body>
</html>
```

O código referente ao processo PHP é o que segue:

```
<?php
$host = "servidor_banco_de_dados.com.br";
$user = "usuario_banco_de_dados";
$pass = "senha_banco_de_dados";
$db = "banco_de_dados";
$conecta = mysql_connect($host, $user, $pass) or print (mysql_error());
mysql_select_db($db, $conecta);
$nome_cliente = $_REQUEST['nome_cliente'];
$sql = "SELECT * FROM Cliente";
$query = mysql_query($sql);
$conteudo = mysql_fetch_array($query);
echo($conteudo);
?>
```

Agora, vamos analisar os códigos. O código lista_cliente.php vai ler todos os dados de todos os clientes por meio do comando:

SELECT * FROM <nome da tabela>;

Acompanhe essa análise clicando nas abas a seguir.

- O nome da tabela será Cliente.
- O resultado deste comando armazenará na variável "$conteudo" todos os dados de todos os clientes.
- O comando PHP "echo" irá retornar para o código HTML pelo comando Ajax no retorno "result".

É necessário ter uma atenção especial aqui: o PHP disponibiliza a função "mysql_fetch_array()" que desmembra as informações recuperadas em $query. Isto faz com que tenhamos cada campo separado em um vetor (iniciando por 0). Neste caso, teremos $contedudo[0] = dados do primeiro cliente; $conteudo[1] = dados do segundo cliente, etc.

VOCÊ SABIA?

As funções *call* e *apply* do JavaScript permitem invocar métodos como se estivéssemos no contexto de um outro objeto. A importância desses dois métodos é porque *apply* e *call* permitem acessar métodos emprestados, reduzindo, assim, a quantidade total de código gerada.

Conheça mais sobre as funções do JavaScript clicando a seguir.

O JavaScript disponibiliza funções que tem como objetivo invocar métodos com um comportamento como se estivesse em outro objeto. Estes comandos são o *call* e o *apply*.

Estes métodos permitem acessar outros métodos emprestados. A vantagem é reduzir o tamanho do código, sendo isto uma característica de código bem estruturado e coeso.
No código HTML, usamos o comando result.lenght que retorna o tamanho do vetor, ou seja, a quantidade de clientes recuperados. Iremos armazenar este valor na variável "i".
A partir daí, fazemos um "*loop*" com o comando for(). A sintaxe do comando for é: *for*(índice inicial, índice final, incremento).
Então iremos ler cada cliente dentro do laço *for*. Para cada cliente, recuperamos cada valor contido no conteúdo de cada vetor. Então, acessamos como sendo um vetor bidimensional, sendo que o primeiro índice representa o cliente e o segundo índice o campo.
Por exemplo, result[i,0], teremos o nome do cliente, result[i,1] o endereço do cliente, etc. Neste momento inserimos na tabela recuperada pela variável "tabela_cliente" pelo comando tabela_cliente.cell = <informação recuperada>.
A seguir iremos verificar como é executado e filtro na recuperação de informações no banco de dados e exibir no código HTML.

3.3 Desenvolvimento *web*: filtrar e mostrar registro no banco de dados

Em diversos momentos, quando usamos uma página web, pode ser que seja necessário executar algum tipo de filtro. Por exemplo, se você está adquirindo uma peça de roupa, é bem provável que queira que seja exibida a cor da sua preferência, modelo e tamanho disponíveis. Portanto, a página deve estar preparada para esta função. Os comandos do SQL SELECT têm como parâmetros que possibilitam recuperar informações do banco de dados por meio de filtros, retornando aquilo que realmente seja necessário para o usuário. Neste sentido, quando a página faz a chamada de um programa PHP, por meio do *script* JavaScript, é executado de assíncrona uma pesquisa SQL. O resultado do processamento PHP é retornado por meio do script evocado anteriormente. Quando este resultado é recuperado, então é possível exibir no navegador pelo documento HTML a pesquisa realizada.

3.3.1 Select

Como é possível filtrar a recuperação de registros em um banco de dados? Para que possamos estudar a recuperação dos registros gravados em um banco de dados e exibir em uma página HTML utilizando um filtro, vamos voltar ao nosso cadastro de clientes, agora criando uma página HTML de consulta. Para tal, vamos criar um código HTML para consulta de cliente. Veja o código a seguir:

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
<title>Lojas Super</title>
</head>
<body>
<h1>Consulta de Clientes</h1>
<p>Preencha o nome do cliente para efetuar a consulta</p>
<font> Nome do cliente: </font><p></p>
<input type="button" name="botao-ok" value="Consultar" id="botao_consulta">
<input type="text" id="nome"></input><p></p>
<font> Endereço do cliente: </font><p></p>
<input type="text" id="endereco"></input><p></p>
<font> Idade do cliente </font><p></p>
```

<input type="text" id="idade"></input>
 Sexo do cliente
<input type="radio" name="sexo" id="masculino" checked="checked"></input>
<input type="radio" name = "sexo" id="feminino"></input>
</body>
</html>

Vamos analisar a estrutura desta página, comparando-a com a página de cadastro. Além das alterações relativas aos rótulos, é solicitado o nome do cliente e o botão com a opção “Consultar”. Abaixo, temos a demonstração da tela.

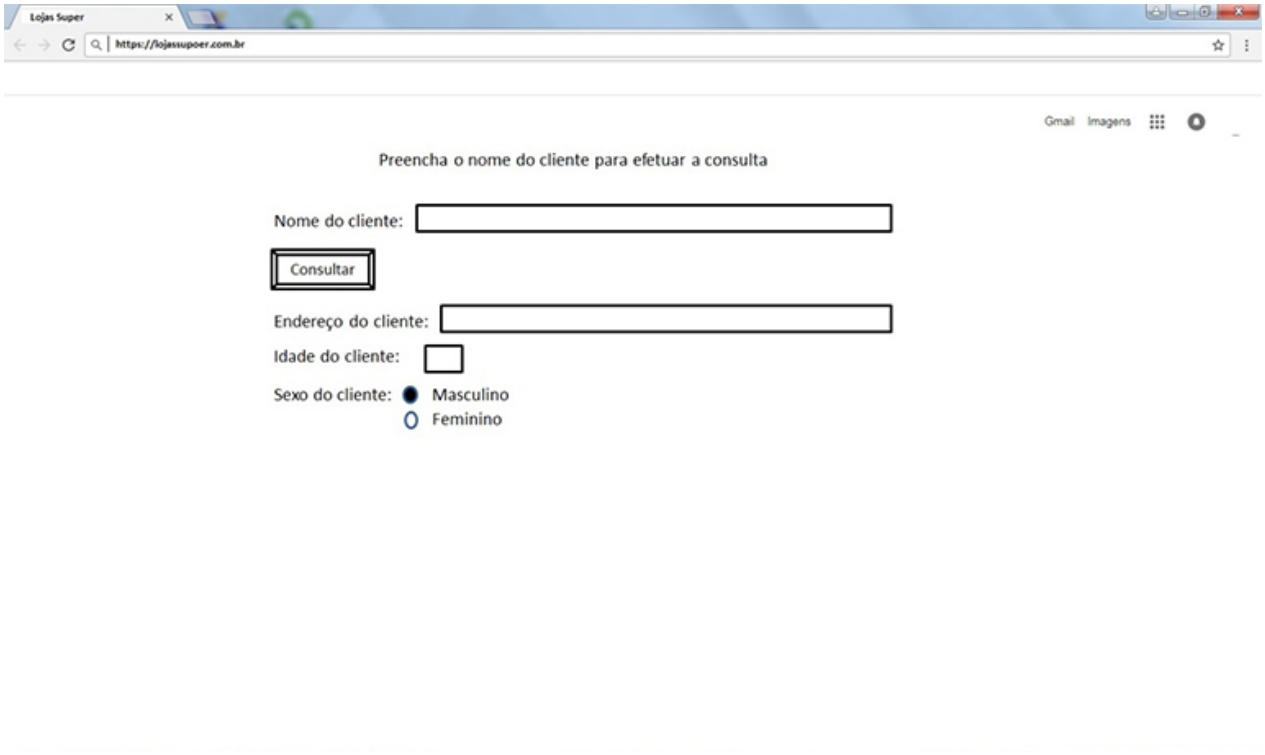

Figura 6 - Demonstração de uma tela de consulta dos clientes de uma loja desenvolvida com o código HTML.
Fonte: Elaborada pelo autor, 2019.

Inicialmente, é necessário incluir no código HTML os comandos do <script> que executem a recuperação do nome do cliente e processe a pesquisa no banco de dados. O código ficaria da seguinte forma:

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
<title>Lojas Super</title>
<script>
fuction Consulta() {
var nome_cliente = document.getElementById("nome").value;
var dadosajax = {
'nome_cliente': nome_cliente
};
var pageurl = consulta_cliente.php';
$.ajax({
url: pageurl,
```

```
data: dadosajax,
type: 'POST',
cache: false,
error: function(XMLHttpRequest, textStatus, errorThrown)
{
alert(“Erro Gravação Cliente!!”);
},
success: function(result)
{
document.getElementById("nome").value = result(0);
document.getElementById("endereço").value = result(1);
document.getElementById("idade").value = result(2);
if (result(3) == “M”)
{
document.getElementById("masculino").checked = True;
}
else
{
document.getElementById("feminino").checked = True;
}
}
}
</script>
</head>
<body>
<h1>Consulta de Clientes</h1>
<p>Preencha o nome do cliente para efetuar a consulta</p>
<font> Nome do cliente: </font><p></p>
<input type="button" name="botao-ok" value="Consultar" id="botao" onClick=Consulta()">
<input type="text" id="nome"></input><p></p>
<font> Endereço do cliente: </font><p></p>
<input type="text" id="endereco"></input><p></p>
<font> Idade do cliente </font><p></p>
<input type="text" id="idade"></input><p></p>
<font> Sexo do cliente </font><p></p>
<input type="radio" name="sexo" id="masculino" checked="checked"></input><p></p>
<input type="radio" name = "sexo" id="feminino"></input><p></p>
</body>
</html>
```

Agora, vamos analisar o <script> do código. Neste caso, o Ajax passa como parâmetro somente o nome do cliente a ser consultado no banco de dados, que será executado no programa PHP “consulta_cliente.php”. Após a execução do programa php, as informações recuperadas estarão contidas na variável “*return*”, caso o processamento tenha sido executado com sucesso pela função function(return).

VOCÊ SABIA?

Em JavaScript, a combinação de uma função e a referência ao seu estado externo é uma *closure. Closure* permite a criar uma função dentro de outra. A capacidade de esconder informações também é conhecida como *data privacy*. Isso é fundamental para que seja possível esconder informações classificadas como privadas.

Neste caso, o retorno é um vetor em que cada item (iniciando por 0) contém os dados que foram recuperados pelo programa "consulta_cliente.php". Percebam que o retorno do sexo ("M" ou "F") deve ser convertido para o campo "radio". Se for "M", o campo "*radio*", onde o identificador for "masculino", deverá ser indicado como "*True* " para o atributo "*checked*". E o mesmo para o caso de ser feminino. Agora, vamos verificar como ficaria o código do programa "consulta_cliente.php".

```
<?php
$host = "servidor_banco_de_dados.com.br";
$user = "usuario_banco_de_dados";
$pass = "senha_banco_de_dados";
$db = "banco_de_dados";
$conecta = mysql_connect($host, $user, $pass) or print (mysql_error());
mysql_select_db($db, $conecta);
$nome_cliente = $_REQUEST['nome_cliente'];
$sql = "SELECT * FROM Cliente WHERE NomeCliente = '$nome_cliente'";
$query = mysql_query($sql);
$conteudo = mysql_fetch_array($query);
echo($conteudo);
?>
```

Os comandos fundamentais são os mesmos do cadastro de cliente, ou seja, o acesso ao banco de dados e a declaração das variáveis. O que podemos analisar com detalhes é o comando SQL SELECT com o filtro. Este comando terá a seguinte sintaxe:

SELECT * FROM <nome da tabela> **WHERE** <condições de pesquisa>;

Analisando a sintaxe temos que, após o comando **SELECT** podemos utilizar o símbolo de asterisco (*) para selecionar todos os campos da tabela indicada por **FROM**, ou indicar qual o campo que se deseja recuperar, caso não seja necessário recuperar todos os campos. Após esta cláusula, temos o comando **WHERE** que indica o filtro da seleção. No nosso caso, queremos selecionar todos os campos da tabela Cliente, onde será filtrado pelo nome do cliente.

Confira, no vídeo a seguir, um caso prático que exemplifica a sintaxe e lógica do comando SELECT, demonstrando uma situação na qual um desenvolvedor necessita implementar cláusulas avançadas deste comando.

https://cdnapisec.kaltura.com/p/1972831/sp/197283100/embedIframeJs/uiconf_id/30443981/partner_id/1972831?iframeembed=true&playerId=kaltura_player_1548870288&entry_id=1_31n6br7p

Igualmente ao cadastro, executamos o comando "mysql_query()". Neste caso, será armazenado na variável $conteudo todas as informações retornadas pelo "mysql_query". Mas, é necessário ter uma atenção especial aqui: o PHP disponibiliza a função "mysql_fetch_array()" que desmembra as informações recuperadas em $query. Isto faz com que tenhamos cada campo separado em um vetor (iniciando por 0). Neste caso, teremos $conteudo[0] = nome do cliente; $conteudo[1] = endereço do cliente, etc.

VOCÊ O CONHECE?

O matemático britânico Edgar Frank Codd (1923-2003) foi o criador da linguagem SQL, nos anos 1970, nos laboratórios da IBM em San Jose, parte do projeto System R, que tinha por objetivo demonstrar a viabilidade da implementação do modelo relacional.

Finalmente, retornamos para o código que disparou este programa PHP (consulta_cliente), por meio do comando "*echo*" o campo $conteúdo, sendo que, no programa consulta_cliente, será retornado em *return,* conforme observado no código descrito.
No próximo tópico, vamos analisar como é possível editar os registros em um banco de dados por um programa PHP, executado pela página HTML.

3.4 Desenvolvimento *web:* editar registro no banco de dados

Quando inserimos qualquer informação em um banco de dados, pode ocorrer que seja necessário alterar esta informação em um determinado momento. Ou seja, as informações gravadas também podem ser alteradas ou excluídas. Os sistemas devem estar preparados para estes comandos fundamentais de processamento de dados, que são a inserção, alteração, exclusão e recuperação. Com relação aos sistemas desenvolvidos em uma página *web,* a lógica é a mesma. A página deve estar preparada para qualquer situação onde o usuário necessite manipular as informações, excluí-las ou consultá-las, como também, poder alterar estas informações. Veremos a seguir como é possível, por meio do comando Update do SQL, editar e alterar os registros em um banco de dados.

3.4.1 Update

Após a inclusão de um registro em um banco de dados, quase sempre o usuário pode querer alterá-lo. Como isto pode ser feito? Para que possamos exibir as informações em uma página HTML e efetuar a alteração dos registros gravados em um banco de dados, vamos voltar ao nosso cadastro de clientes, agora criando uma página HTML de alteração. Para isso, vamos criar um código HTML para alteração de cliente. Veja o código a seguir:

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
<title>Lojas Super</title>
</head>
<body>
<h1>Alteração de Clientes</h1>
<p>Preencha o nome do cliente para efetuar a alteração</p>
<font> Nome do cliente: </font><p></p>
<input type="button" name="botao" value="Consultar" id="botao_consulta" onClick="Pesquisa()”>
<input type="text" id="nome"></input><p></p>
<font> Endereço do cliente: </font><p></p>
<input type="text" id="endereco"></input><p></p>
```

 Idade do cliente
<input type="text" id="idade"></input>
 Sexo do cliente
<input type="radio" name="sexo" id="masculino" checked="checked"></input>
<input type="radio" name = "sexo" id="feminino"></input>
<input type="button" name="botao" value="Consultar" id="botao_consulta" onClick="Altera()">
</body>
</html>
Na figura a seguir vemos como será exibida a tela no navegador.

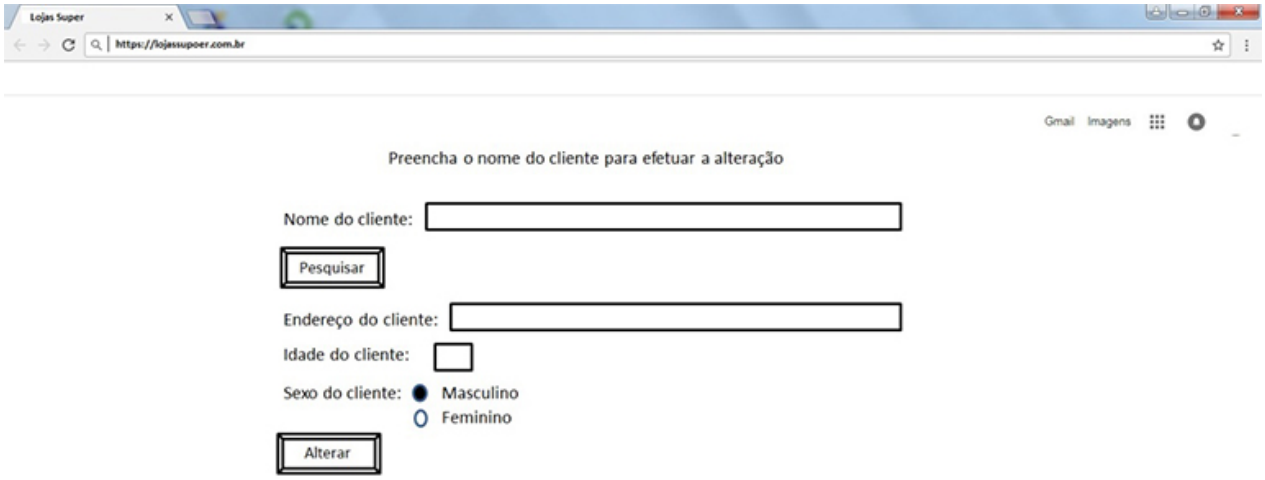

Figura 7 - Demonstração de uma tela de alteração de clientes de uma loja desenvolvida com o código HTML.
Fonte: Elaborada pelo autor, 2019.

Neste código, haverá duas funções do <script>. Uma para a recuperação das informações do cliente no banco de dados e outra para efetuar as alterações no banco. Portanto, haverá uma função “Pesquisa()” que executará o programa PHP “pesquisa_cliente()” e outra função “Altera()” que executará o programa PHP “altera_cliente()”. O código completo ficará, portanto, da seguinte maneira:
<!DOCTYPE html>
<html lang="pt-br">
<head>
<title>Lojas Super</title>
<script>
fuction Pesquisa() {
var nome_cliente = document.getElementById("nome").value;
var dadosajax = {
'nome_cliente': nome_cliente
};
var pageurl = pesquisa_cliente.php';

```
$.ajax({
url: pageurl,
data: dadosajax,
type: 'POST',
cache: false,
error: function(XMLHttpRequest, textStatus, errorThrown)
{
alert(“Erro Gravação Cliente!!”);
},
success: function(result)
{
document.getElementById("nome").value = result(0);
document.getElementById("endereço").value = result(1);
document.getElementById("idade").value = result(2);
if (result(3) == “M”)
{
document.getElementById("masculino").checked = True;
}
else
{
document.getElementById("feminino").checked = True;
}
}
}
fuction Altera() {
var nome_cliente = document.getElementById("nome").value;
var endereco_cliente = document.getElementById("endereco").value;
var idade = document.getElementById("idade").value;
if (document.getElementById("masculino").checked)
{
var sexo_cliente = “M”;
}
Else
{
var sexo_cliente = “F”;
}
}
var dadosajax = {
'nome_cliente': nome_cliente,
‘endereco_cliente’: endereço_cliente,
‘idade_cliente’: idade_cliente,
‘sexo_cliente’: sexo_cliente
}
};
var pageurl = altera_cliente.php';
$.ajax({
url: pageurl,
data: dadosajax,
type: 'POST',
```

```
cache: false,
error: function(XMLHttpRequest, textStatus, errorThrown)
{
alert(“Erro Alteração Cliente!!”);
},
success: function(result)
{
alert(“Alteração efetuada com sucesso”);
}
}
</script>
</head>
<body>
<h1>Consulta de Clientes</h1>
<p>Preencha o nome do cliente para efetuar a consulta</p>
<font> Nome do cliente: </font><p></p>
<input type="button" name="botao" value="pesquisa" id="botao" onClick=Pesquisa()">
<input type="text" id="nome"></input><p></p>
<font> Endereço do cliente: </font><p></p>
<input type="text" id="endereco"></input><p></p>
<font> Idade do cliente </font><p></p>
<input type="text" id="idade"></input><p></p>
<font> Sexo do cliente </font><p></p>
<input type="radio" name="sexo" id="masculino" checked="checked"></input><p></p>
<input type="radio" name = "sexo" id="feminino"></input><p></p>
<input type="button" name="botao" value="altera" id="botao" onClick=Altera()">
</body>
</html>
```

O código PHP para executar a pesquisa do cliente seria:

```
<?php
$host = "servidor_banco_de_dados.com.br";
$user = "usuario_banco_de_dados";
$pass = "senha_banco_de_dados";
$db = "banco_de_dados";
$conecta = mysql_connect($host, $user, $pass) or print (mysql_error());
mysql_select_db($db, $conecta);
$nome_cliente = $_REQUEST['nome_cliente'];
$sql = "SELECT * FROM Cliente WHERE NomeCliente = '$nome_cliente'”;
$query = mysql_query($sql);
$conteudo = mysql_fetch_array($query);
echo($conteudo);
?>
```

O código PHP para executar a alteração do cliente seria:

```
<?php
$host = "servidor_banco_de_dados.com.br";
$user = "usuario_banco_de_dados";
$pass = "senha_banco_de_dados";
$db = "banco_de_dados";
$conecta = mysql_connect($host, $user, $pass) or print (mysql_error());
```

```
mysql_select_db($db, $conecta);
$nome_cliente = $_REQUEST['nome_cliente'];
$endereco_cliente = $_REQUEST[' endereco_cliente '];
$idade_cliente = $_REQUEST['idade_cliente'];
$sexo_cliente = $_REQUEST['sexo_cliente'];
$sql = "UPDATE Cliente SET
EnderecoCliente = $endereco_cliente,
IdadeCliente = $idade_cliente,
SexoCliente = $sexo_cliente,
WHERE NomeCliente = $nome_cliente.
$query = mysql_query($sql);
$conteudo = mysql_fetch_array($query);
echo($conteudo);
?>
```

O que teremos de inédito neste código de alteração é o comando **UPDATE**, que tem a seguinte sintaxe:
UPDATE <tabela> **SET** <campo> = <valor> **WHERE** <condição>
Então, é possível executar a alteração pelo comando **UPDATE**, referenciando cada campo da tabela com o novo valor pelo comando **SET,** considerando a condição **WHERE**, no nosso caso, pelo nome do cliente.
Assim, foi possível compreender como é a sintaxe de alteração no banco de dados por meio do comando SQL inserido no código PHP que foi evocado pelo código HTML. Esta chamada foi realizada pelo Ajax do JQuery, permitindo um processamento assíncrono da página do *site* com o código PHP.

Síntese

Neste capítulo foi possível aprender como fazer as inclusões no banco de dados, recuperar informações e alterar por meio de uma página *web*. As páginas *web* são escritas na linguagem HTML, que sozinhas não conseguem executar nenhuma função, senão exibir as informações na tela. Todas as funções que fazem a interação com o banco de dados são realizadas por meio dos comandos JavaScript e Ajax do JQuery, sendo este um processo assíncrono. Por outro lado, o Ajax chama um programa PHP, que consegue, este sim, interagir com o banco de dados por meio dos comandos SQL.
Podemos verificar que, quando acessamos uma página na internet, na realidade, existem uma série de linguagens que trabalham de forma coordenada e assíncrona para que se possa realizar todas as transações que necessitamos, como consultas, recuperar informações e alterar dados.
As potencialidades das linguagens HTML, JavaScript, PHP e SQL são enormes, permitindo que os desenvolvedores criem sites cada vez mais interativos, intuitivos e de alto nível de qualidade.
Neste capítulo, você teve a oportunidade de:

- conhecer os comandos fundamentais de inserção, recuperação e atualização das informações em banco de dados por meio de uma página *web*;
- entender o comportamento de sincronismo e assincronismo das páginas disponíveis na *web;*
- estudar a relação entre a página HTML, sua sintaxe e a forma de evocação das *scripts* de comandos;
- compreender a inserção dos comandos JavaScript no corpo do documento HTML para a execução da recuperação dos conteúdos da página e a evocação do programa PHP para manipulação nos bancos de dados;
- estudar o comportamento de páginas estáticas e dinâmicas na *web*;
- entender os comandos fundamentais de SQL e de acesso aos bancos de dados pelos comandos PHP.

Bibliografia

DUCKETT, J. **Javascript e Jquery - Desenvolvimento de Interfaces Web Interativas - Guia Prático**. São Paulo: Alta Books, 2015.

GIROLDO, B. C.; FRESSATI, W. Evolução no Desenvolvimento com o HTML 5. In: XVII Seinpar – in... **Anais**... Semana de Informática e XIV Mostra de Trabalhos de Iniciação Científica de Paranavaí. Paranavaí, 2015.

LEMAY, L.; COLBURN, R.; TYLER, D. **Aprenda a criar páginas *web* com HTML e XHTML em 21 dias.** São Paulo: Pearson Education do Brasil, 2002. Disponível em: <http://aulaaberta.bv3.digitalpages.com.br/users/publications/9788534614283>. Acesso em: 13/1/2019.

JAQUES, R. PHP & Segurança: **Blindando Aplicações *Web*** - Rafael Jaques. Canal iMasters, YouTube, publicado em: 2 de jun. de 2017. Disponível em: < https://www.youtube.com/watch?v=KmShf5VDGUM&t=36s>. Acesso em: 14/1/2019.

JQUERY. **jQuery API**. Portal jQuery, The jQuery Foundation, 2019. Disponível em: <http://api.jquery.com/>. Acesso em: 13/1/2019.

LYNN, B. **Use a Cabeça. SQL.** São Paulo: Alta Books, 2016.

MySQL. **MySQL Documentation**. Portal MySQL, Oracle Corporation, 2019. Disponível em: <https://dev.mysql.com/doc/>. Acesso em: 07 out. 2017.

PHP. **Documentation.** Portal PHP, 2019. Disponível em: <http://php.net/docs.php>. Acesso em: 13/1/2019.

W3SCHOOLS. **HTML 5 Tutorial.** Portal W3Schools, 2019. Disponível em: <https://www.w3schools.com/html/>. Acesso em: 13/1/2019.