



Fundamentos de Programação Web

UNIDADE 08

Desenvolvimento full stack (front-end + back-end)

Olá!

Nesta unidade, vamos concentrar nossos esforços em complementar as práticas de desenvolvimento já experimentada nas unidades anteriores.

Neste ponto, já estamos considerando que todos têm um **projeto** de **aplicação web**, no qual estão definidos:

1. **A especificação da aplicação:** os atores, ou personas, da aplicação e seus requisitos funcionais.
2. **A estrutura de dados (*back*):** os dados utilizados e que precisam ser mantidos em BD.
3. **A interface de usuário (*front*):** telas para interação entre o usuário e aplicação.
4. **A integração *back* e *front-end*:** código PHP para interagir com o usuário, enviando e recuperando dados do BD.

Agora, vamos complementar nossa **aplicação web**, para que ela apresente as características de tratamento de *login* a fim de permitir **acesso a usuários cadastrados e autenticados**. Nos exemplos apresentados na VIDEOAULA, já começamos a demonstrar como o controle de acesso é realizado, o que veremos em mais detalhes nas seções seguintes.

Em resumo, nesta unidade, você trabalhará para:

1. Definir e criar uma estrutura para manter dado de uma aplicação *web*.
2. Integrar ***front-end*** e ***back-end*** de uma aplicação *web*, para realização de CRUD completo.
3. Garantir acesso à aplicação *web* apenas de usuários cadastrados e autenticados.

Ao final da implementação desta **2ª *sprint***, nossa aplicação estará configurada como na Figura a seguir.

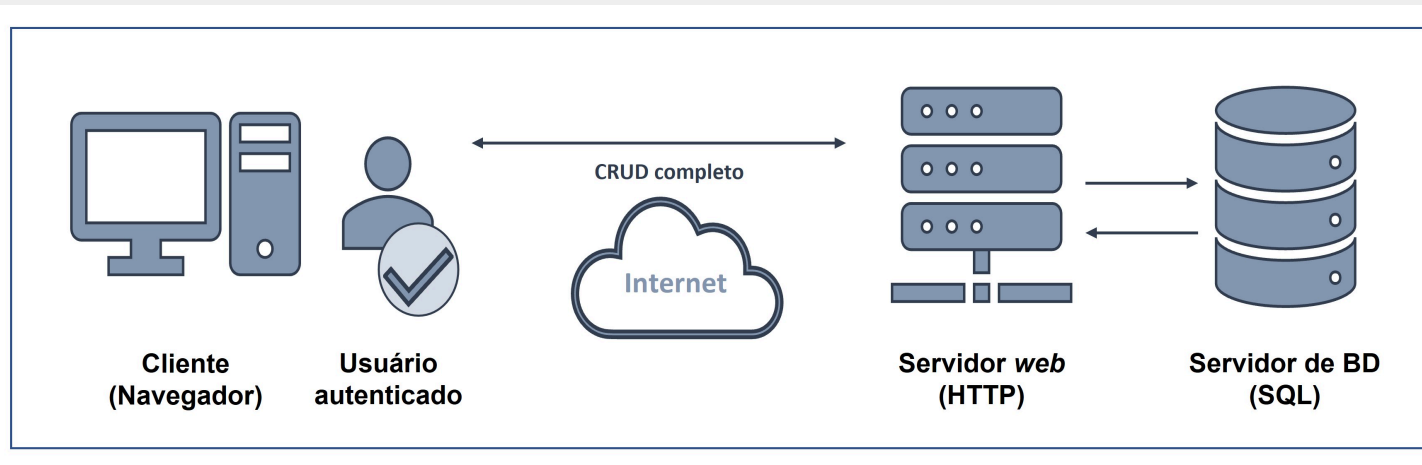


Figura 1: Aplicação web: usuário autenticado. Fonte: A autora (2023).



SAIBA MAIS

Aplicação PHP com autenticação de usuário

- *How to – login form*: https://www.w3schools.com/howto/howto_css_login_form.asp.
- MILETTO, Evandro Manara; BERTAGNOLLI, Silvia de Castro. **Desenvolvimento de Software II: introdução ao desenvolvimento web com html, css, javascript e php**. Porto Alegre: Bookman, 2014. 276 p.
 - Capítulo 8 – Integração de PHP e MySQL
 - Capítulo 10 – Introdução à segurança em sistemas web
 - Criptografia de dados em trânsito com o protocolo HTTPS – página 258.
 - Autenticação de usuários – página 263.

Antes de explorarmos os temas de estudo e os exemplos desta unidade, na presente VIDEOAULA, vamos verificar alguns passos básico de como **recuperar do BD** os dados para **tratamento de *login*** e garantia de **acesso** apenas a **usuários cadastrados e autenticados**. Também veremos como manter **senha criptografada** em banco de dados, além de completarmos os exemplos de **alteração** (*UPDATE*) e **exclusão** (*DELETE*) de dados.

Desenvolvimento full stack (front-end + back-end)



| Aplicação web exemplo: ConsultorioLogin

Para esta prática, daremos continuidade ao exemplo realizado na unidade anterior. Agora, nossa **aplicação web PHP** é o exemplo “**ConsultorioLogin**”, já apresentada de forma geral na VIDEOAULA desta unidade. Esta aplicação está disponibilizada no material de apoio, arquivo compactado **ConsultorioLoginSemana8.zip**.



DOCUMENTOS

Material de apoio:

[Material Semana 8 ConsultorioLogin](#)



EXPERIMENTE

PRÁTICA: Aplicação web exemplo “ConsultórioLogin”

Para acompanhar esta aplicação-exemplo, siga os passos:

1. Descompacte o arquivo **ConsultorioLoginSemana8.zip** na sua pasta de trabalho: **C:/xampp/htdocs**.
2. Inicialize os servidores **Apache** e **MySQL** do **XAMPP**.
3. Exclua a base de dados **ClinicaA**, criada para as práticas anteriores, pois utilizaremos mais uma tabela às originais. No **phpMyAdmin**, selecione a opção SQL do *menu* superior, e execute o comando a seguir para eliminar a base de dados e todas as suas tabelas:

```
1 | DROP DATABASE `ClinicaA`
```

4. Em seguida, ainda na opção SQL do menu superior, crie novamente a base de dados ClinicaA, vazia, sem tabelas ainda, executando o comando:

5. Ainda no **phpMyAdmin**, com a nova base de dados **ClinicaA** selecionada, execute o *script* SQL: **C:/xampp/htdocs/consultorioLogin/bd/bd_clinica.sql**. Este *script* **cria** e **preenche** as novas tabelas necessárias para a aplicação.

6. Após a criação da base, tabelas e preenchimento das tabelas, aplicação já deve estar funcional. Então, tente acessar a aplicação, digitando na barra de endereços do navegador a URL: <http://localhost/consultorioLogin>.

7. A tela do navegador deve apresentar a tela inicial, em que é possível realizar o *login*, ou se cadastrar na aplicação, como na figura a seguir.



Figura 2: Interface inicial da aplicação web exemplo: ConsultórioLogin. Fonte: A autora (2023).

Vamos continuar o exemplo apresentado na VIDEOAULA. Para isso, precisamos entender quais são os dados mantidos na base utilizada pela aplicação, que tem as tabelas **Médico** e **Especialidade**, que se relacionam como exibido na Figura a seguir.

#ParaTodosVerem ⓘ

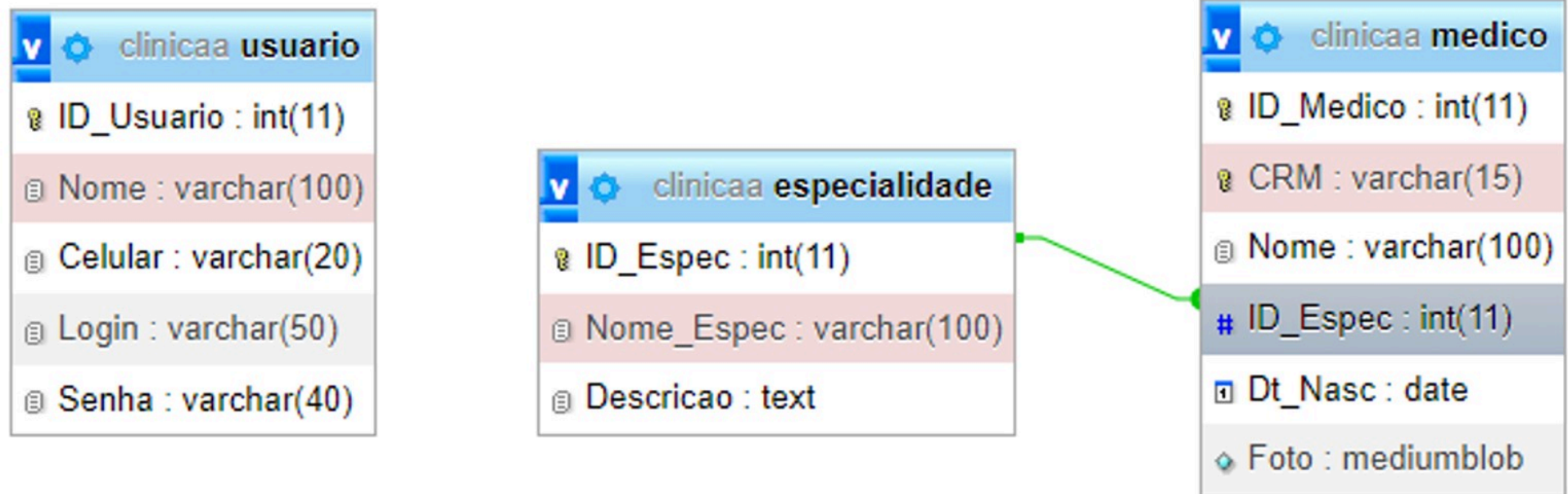


Figura 3: Tabelas utilizadas pela aplicação ConsultorioLogin. Fonte: A autora (2023).

Tabelas:

- Médico.
- Especialidade.

- **Usuário.**

Após executar o *script* SQL **bd_clinica.sql** para criar e preencher as tabelas, podemos verificar as tabelas preenchidas com os comandos SQL:

```
1 | SELECT * FROM `medico`
2 | SELECT * FROM `especialidade`
3 | SELECT * FROM `usuário`
```

O conteúdo das tabelas deve ser:

Especialidade		
ID_Espec	Nome_Espec	Descricao
1	Dermatologista	Médico especialista no diagnóstico, tratamento e p...
2	Endocrinologista	Médico especialista no diagnóstico, tratamento e p...
3	Pediatra	Médico especializado na assistência a crianças e a...

Médico					
ID_Medico	CRM	Nome	ID_Espec	DT_Nasc	Foto
1	CRM/SP 3456-89	Sílvia Moreira	1	1965-09-01	[BLOB - 9.9 KB]
2	CRM/PR 3896-55	Giovane Brito	2	1985-05-16	[BLOB - 10.4 KB]
5	CRM/RS 4587-15	Ana Carolina	3	1989-12-10	<i>NULL</i>

Médico					
ID_Medico	CRM	Nome	ID_Espec	DT_Nasc	Foto
12	CRM/PR 1234-89	Oli Oli Oli	1	2000-10-20	[BLOB - 10.2 KB]
13	CRM/SP 0456-89	Jose da Silva	3	1950-03-05	[BLOB - 9.6 KB]

Usuário				
ID_Usuario	Nome	Celular	Login	Senha*
1	José da Silva	(41)99999-9999	jose.silva	70b4269b412a8af42b1f7b0d26eceff2

*** Senha do usuário é gravada criptografada**

A senha no usuário, gravada no BD na tabela **usuário**, vem do seu cadastro, realizado na página inicial da aplicação *web*, no arquivo **index.php**:

...

```
pattern="(?.*\d) (?.*[a-z]) (?.*[A-Z]).{6,8}" placeholder="sua senha" required>
```

...

A senha é validada por uma **expressão regular** (`pattern`), e deve conter, ao menos:

- Um número.
- Uma letra maiúscula.
- Uma letra minúscula.
- Um caractere especial.
- Ter de 6 a 8 caracteres.

Quando o cadastro do usuário José da Silva foi realizado, foram digitados os seguintes dados:

- Login = `'jose.silva'` e com
- Senha = `'Abc@123'`.

Contudo, ...

- A **senha gravada** na tabela **usuário** é uma sequência ilegível de caracteres, muito diferente de `'Abc@123'`.
- Isso acontece porque, antes de ser gravada, a senha foi criptografada, com uma função de **HASH**.

HASH MD5

Uma função de **hash criptográfico**, muitas vezes conhecida apenas como **hash**, é um algoritmo matemático que transforma qualquer bloco de dados em uma série de caracteres de comprimento fixo.

O **MySQL** possui essa função de **hash**, chamada **md5**. Ela recebe a cadeia de caracteres da senha e retorna 32 dígitos hexadecimais, que corresponde à **senha criptografada**.

Para testar a função de **hash md5 do MySQL**, execute o comando SQL a seguir. Verifique que o resultado dele corresponde ao valor gravado para a senha do 'José da Silva'.

```
SELECT md5 ('Abc@123');
```

Resultado:

md5('Abc@123')
70b4269b412a8af42b1f7b0d26eceff2

```
SELECT senha FROM usuario WHERE login = 'jose.silva';
```

Resultado:

Senha
70b4269b412a8af42b1f7b0d26eceff2



SAIBA MAIS

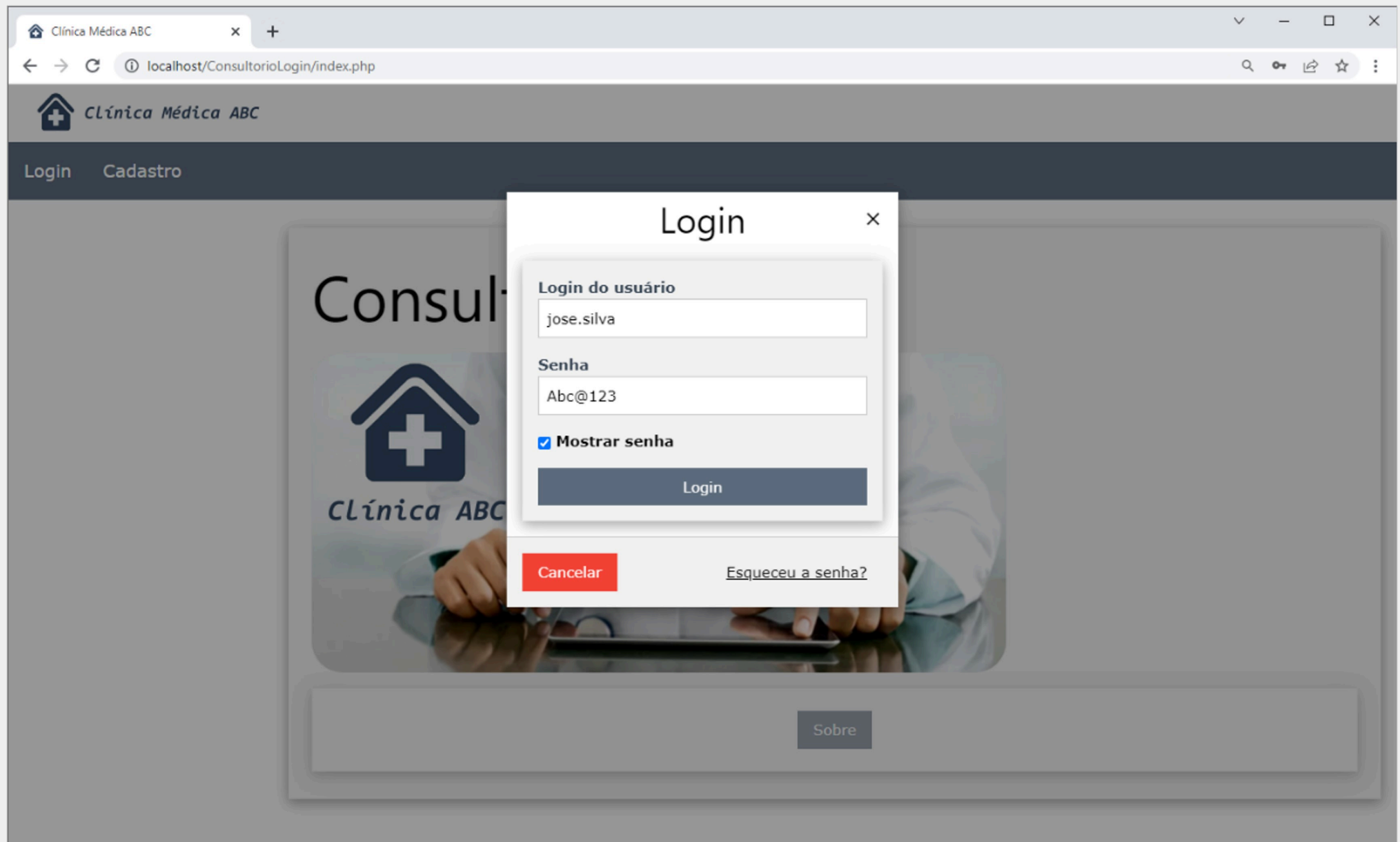
MD5

- **MySQL MD5 function:** <https://www.techonthenet.com/mysql/functions/md5.php>
- **MySQL 12.14 encryption and compression functions:** <https://dev.mysql.com/doc/refman/8.0/en/encryption-functions.html>
- **Criptografia MD5:** <https://www.devmedia.com.br/criptografia-md5/2944>
- MILETTO, Evandro Manara; BERTAGNOLLI, Silvia de Castro. **Desenvolvimento de Software II:** introdução ao desenvolvimento web com html, css, javascript e php. Porto Alegre: Bookman, 2014. 276 p. (*linkdo livro acima*).
 - **Capítulo 10 – Introdução à segurança em sistemas web**
 - Criptografia de dados em trânsito com o protocolo HTTPS – página 258.

- Autenticação de usuários – página 263.

Acrescentando atualização e exclusão de dados (UPDATE + DELETE)

Para habilitarmos a **edição** (*UPDATE*) e **exclusão** (*DELETE*) de registros da tabela **médico**, precisamos realizar o *login* na aplicação. Fazemos esse procedimento como demonstrado na Figura a seguir.



Acesso com o usuário já cadastrado **'José da Silva'**, que tem a senha **'Abc@123'**, visível após acionamento do botão **'Mostrar senha'**.

Figura 4: Realização de login com o usuário 'José da Silva'. Fonte: A autora (2023).

Uma vez autenticado, o usuário poderá realizar o CRUD completo da aplicação.



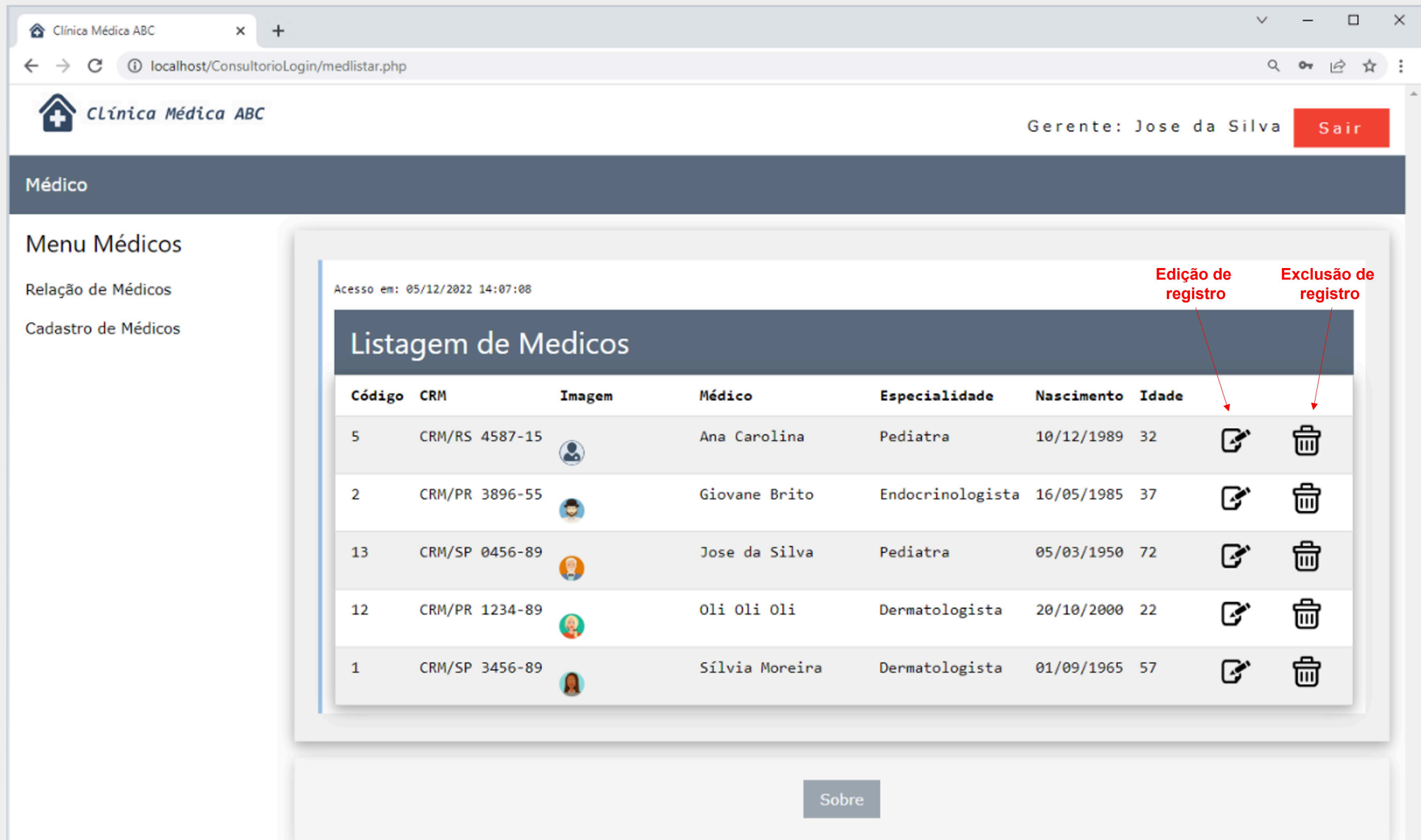
IMPORTANTE

Já realizamos, na unidade anterior, as opções (que não foram alteradas na presente unidade) de:

- *Create* (SQL INSERT) – na aplicação, é acionada na opção de *menu* lateral “**Cadastro de Médicos**”.
 - Arquivo **medIncluir.php** – apresenta formulário para obtenção dos dados para inclusão de registro de médico, que são enviados via *POST* para o arquivo **medIncluir_exe.php**.
 - Arquivo **medIncluir_exe.php** – recebe os dados do arquivo **medIncluir_exe.php** e realiza o comando de *INSERT* na base de dados.
- *Read* (SQL SELECT) – na aplicação, é acionada na opção de menu lateral “**Relação de Médicos**”.
 - Arquivo **medListar.php** – apresenta a relação de médicos, obtida por um comando de *SELECT* na base de dados.

Agora, vamos verificar como realizar o *update* (SQL *UPDATE*) e *delete* (SQL *DELETE*), que são executados a partir de *links* indicados como na Figura a seguir, respectivamente para:

- Edição de registro 📄✎ (SQL *UPDATE*).
- Exclusão de registro 🗑 (SQL *DELETE*).



Clínica Médica ABC

Gerente: Jose da Silva [Sair](#)

Médico
















Menu Médicos

Relação de Médicos

Cadastro de Médicos

Acesso em: 05/12/2022 14:07:08







Listagem de Medicos

Código	CRM	Imagem	Médico	Especialidade	Nascimento	Idade	Edição de registro	Exclusão de registro
5	CRM/RS 4587-15		Ana Carolina	Pediatra	10/12/1989	32		
2	CRM/PR 3896-55		Giovane Brito	Endocrinologista	16/05/1985	37		
13	CRM/SP 0456-89		Jose da Silva	Pediatra	05/03/1950	72		
12	CRM/PR 1234-89		Oli Oli Oli	Dermatologista	20/10/2000	22		
1	CRM/SP 3456-89		Sílvia Moreira	Dermatologista	01/09/1965	57		

Sobre

Figura 5: Relação de médicos, com opções para edição e exclusão de registro. Fonte: A autora (2023).

Listagem de Medicos

Código	CRM	Imagem	Médico	Especialidade	Nascimento	Idade	
5	CRM/RS 4587-15		Ana Carolina	Pediatra	10/12/1989	32	<div><div><div>Editar Médico</div></div></div>
2	CRM/PR 3896-55		Giovane Brito	Endocrinologista	16/05/1985	37	<div><div></div></div>

localhost/ConsultorioLogin/medAtualizar.php?id=5

Figura 6: Detalhe do envio de dados para atualização de registro identificado de médicos. Fonte: A autora (2023).

1. Atualizar registro de médico. Uma vez que selecionamos o registro a editar, como indicado na Figura anterior, esse identificador único (PK) é enviado via URL (método *GET*) para o arquivo `medAtualizar.php` – o detalhe da Figura anterior indica a URL do link no **rodapé do navegador**. No exemplo, estamos atualizando o registro com `ID_Medico = 5`: `http://localhost/medAtualizar.php?id=5`.

Trecho do arquivo `c:\xampp\htdocs\consultorio\bd\medAtualizar.php` (sem as classes CSS para facilitar a leitura)







```
1  ...
2  $id = $_GET['id']; // Obtém PK do Médico que será atualizado
3  ...
4  // Faz Select na Base de Dados
5  $sql = "SELECT ID_Medico, Nome, CRM, Dt_Nasc, ID_Espec, Foto FROM Medico
6         WHERE ID_Medico = $id";
7  ...
8  if ($result = $conn->query($sql)) {    // Consulta ao BD ok
9      if ($result->num_rows == 1) {      // Retorna 1 registro que será atualizado
10         $row = $result->fetch_assoc();
11         $especialidade = $row['ID_Espec'];
12         $id_medico     = $row['ID_Medico'];
13         $nome          = $row['Nome'];
14         $CRM           = $row['CRM'];
15         $dataNasc      = $row['Dt_Nasc'];
16         $foto          = $row['Foto'];
17     ...
18     // Preenche um formulário com os dados do registro a ser editado
19     // Ação do form = medAtualizar_exe.php, com dados enviados via POST
20     <form action="medAtualizar_exe.php" method="post" enctype="multipart/form-data">
21     ...
22     <input type="hidden" id="Id" name="Id" value=" echo $id_medico; ?>">
23     ...
24     <input name="Nome" type="text" pattern="[a-zA-Z\u00C0-\u00FF ]{10,100}$" title="Nome entre 10 e 100 letras." va
25     ...
26     <input name="CRM" id="CRM" type="text" maxlength="15" placeholder="CRM/UF XXXX-XX" title="CRM/UF XXXX-XX" value
27     ...
28     <input name="DataNasc" type="date" placeholder="dd/mm/aaaa" title="dd/mm/aaaa" title="Formato: dd/mm/aaaa" valu
29     ...
30     <input type="submit" value="Alterar" class="w3-btn w3-red">
31     ...
```

2. Aciona o *UPDATE* dos dados no *form*. A ação do formulário indica a página medAtualizar_exe.php, que tenta executar o SQL *UPDATE* no MySQL.b

Trecho do arquivo c:\xampp\htdocs\consultorio\medAtualizar_exe.php

```
1  ...
2  // Recebe os dados que foram preenchidos no formulário, com os valores que serão atualizados
3  $id      = $_POST['Id']; // identifica o registro a ser alterado
4  $nome    = $_POST['Nome'];
5  $CRM     = $_POST['CRM'];
6  $dtNasc  = $_POST['DataNasc'];
7  $espec   = $_POST['Especialidade'];
8  ...
9  // Faz Update na Base de Dados
10 if ($_FILES['Imagem']['size'] == 0) { // Não recebeu uma imagem binária
11     $sql = "UPDATE Medico SET Nome = '$nome', CRM = '$CRM', Dt_Nasc = '$dtNasc'
12           WHERE ID_Medico = $id";
13 }else{ // Prepara imagem para ser salva em BD
14     $imagem = addslashes(file_get_contents($_FILES['Imagem']['tmp_name']));
15     $sql = "UPDATE Medico SET Nome = '$nome', CRM = '$CRM', Dt_Nasc = '$dtNasc',
16           Foto = '$imagem' WHERE ID_Medico = $id";
17 }
18 ...
19 if ($result = $conn->query($sql)) {
20     echo " Registro alterado com sucesso! ";
21 } else {
22     echo "Erro executando UPDATE: " .
23         $conn->connect_error . " ";
24 }
25 ...
```

Listagem de Medicos

Código	CRM	Imagem	Médico	Especialidade	Nascimento	Idade	
5	CRM/RS 4587-15		Ana Carolina	Pediatra	10/12/1989	32	  <div>Excluir Médico</div>
2	CRM/PR 3896-55		Giovane Brito	Endocrinologista	16/05/1985	37	 

localhost/ConsultorioLogin/medExcluir.php?id=5

Figura 7: Detalhe do envio de dados para exclusão de registro identificado de médicos. Fonte: A autora (2023).

3. Excluir registro de médico. Uma vez que selecionamos o registro a excluir, como indicado na Figura anterior, esse identificador único (PK) é enviado via URL (método GET) para o arquivo medExcluir.php – o detalhe da Figura anterior indica a URL do *link* no rodapé do navegador. No exemplo, estamos excluindo o registro com ID_Medico = 5: <http://localhost/medExcluir.php?id=5>.

Trecho do arquivo c:\xampp\htdocs\consultorio\medExcluir.php (sem as classes CSS para facilitar a leitura)

```
1  ...
2  $id = $_GET['id']; //Obtém a PK do médico que será excluído
3  ...
4  // Faz Select na Base de Dados
5  $sql = "SELECT ID_Medico, CRM, Nome, Nome_Espec AS Especialidade, Foto, Dt_Nasc
6         FROM Medico AS M INNER JOIN Especialidade AS E ON
7         (M.ID_Espec = E.ID_Espec) WHERE ID_Medico = $id;";
8  ...
9  if ($result = $conn->query($sql)) {    // Consulta ao BD ok
10     if ($result->num_rows == 1) {        // Retorna 1 registro que será deletado
11         $row = $result->fetch_assoc();
12         $especialidade = $row['ID_Espec'];
13         $id_medico      = $row['ID_Medico'];
14         $nome           = $row['Nome'];
15         $CRM            = $row['CRM'];
16         $dataNasc       = $row['Dt_Nasc'];
17         $foto           = $row['Foto'];
18     ...
19     // Preenche um formulário com os dados do registro a ser deletado
20     // Ação do form = medExcluir_exe.php, com dados enviados via POST
21     <form action="medExcluir_exe.php" method="post">
22     <input type="hidden" id="Id" name="Id" value=" echo $row['ID_Medico']; ?>">
23     ...
24     <label><b>Nome: b> echo $row['Nome']; ?> label>
25     ...
26     <label><b>CRM: b> echo $row['CRM']; ?>label>
27     ...
28     <label><b>Data de Nascimento: b> echo $nova_data;?> label>
29     ...
30     <label><b>Especialidade: b> echo $row['Especialidade'];?>label>
31     ...
32
```

```
33 | <input type="submit" value="Confirma exclusão?" class="w3-btn w3-red">
    | ...
```

4. **Aciona o *DELETE* dos dados no *form*.** A ação do formulário indica a página `medExcluir_exe.php`, que tenta executar o SQL *DELETE* no MySQL.

Trecho do arquivo `c:\xampp\htdocs\consultorio\medlExcluir_exe.php`

```
1 | ...
2 | // ID do registro a ser excluído
3 | $id = $_POST['Id'];
4 | ...
5 | // Faz DELETE na Base de Dados
6 | $sql = "DELETE FROM Medico WHERE ID_Medico = $id";
7 |
8 | ...
9 | if ($result = mysqli_query($conn, $sql)) {
10 |     echo " Registro excluído com sucesso! ";
11 | } else {
12 |     echo " Erro executando DELETE: " . mysqli_error($conn . "");
13 | }
```

Acrescentando permissão de acesso

O tratamento de *login* é bem básico, sendo que o controle é feito no arquivo `index.php`. Ele tem a seguinte lógica de programação:

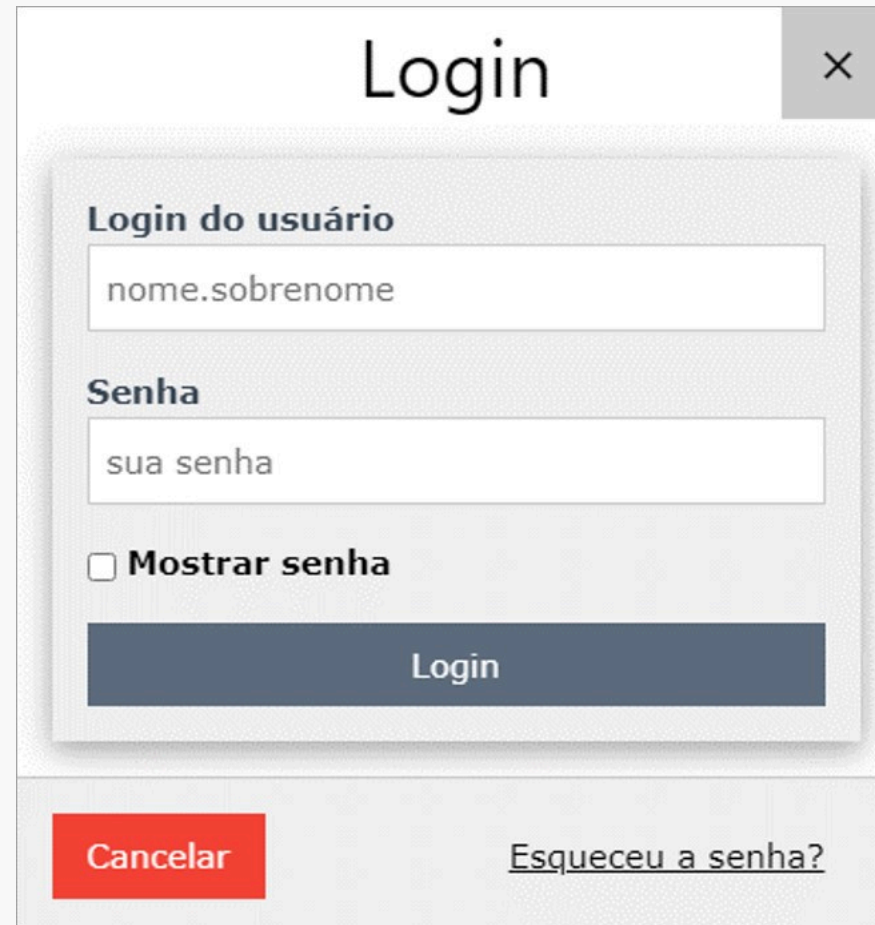
- **PHP define uma estrutura HTML geral.** A página apenas aparece se não houver usuário logado:
 - **Usuário logado** à redireciona para `"location: /ConsultorioLogin/medlistar.php"`.
 - **Usuário NÃO logado** à Nesse caso, são definidas várias **divisões** (elementos HTML

) para organizar em blocos os elementos visíveis, como mostrado a seguir.

Trecho do arquivo c:\xampp\htdocs\consultorio\index.php

```
1  ...
2
3  session_start(); // Cria uma sessão ou retoma a sessão atual
4
5  if (isset($_SESSION ['login'])) { // Verifica se existe usuário  logado
6      header("location: /ConsultorioLogin/medlistar.php"); // Vai para a aplicação web
7      exit();
8  }
9  ?>
10
11
12  <div class="w3-top" id="LoginCadastro" >
13      ...
14  div>
15
16  <div class="w3-top">
17      ...
18  div>
19
20  <div class="w3-sidebar w3-bar-block w3-collapse w3-animate-left" style="z-index:3;width:270px" id="mySidebar" >
21      ...
22  div>
23      ...
24
25  <div class="w3-main w3-container" style="margin-left:270px;margin-top:130px;">
26      ...
27  div>
```

- PHP define uma estrutura HTML para tratamento de *login*, cadastro e falha. Dentro da divisão para o Conteúdo PRINCIPAL, existem outras três divisões para apresentar as janelas de modal¹ para:
 - **Tratamento de *login*:** apresenta o formulário para receber os dados para *LOGIN*, e tentar autenticar usuário a partir de um registro já gravado na tabela **usuário**.

A screenshot of a login modal window. The window has a title bar with the word "Login" and a close button (X). Inside the modal, there is a section titled "Login do usuário" with a text input field containing the placeholder "nome.sobrenome". Below this is a section titled "Senha" with a text input field containing the placeholder "sua senha". Under the password field, there is a checkbox labeled "Mostrar senha". At the bottom of the modal, there is a dark blue button labeled "Login". Below the modal, outside the modal itself, there is a red button labeled "Cancelar" and a link labeled "Esqueceu a senha?".

Login

Login do usuário

nome.sobrenome

Senha

sua senha

☐ Mostrar senha

Login

Cancelar

[Esqueceu a senha?](#)

¹ **Janela de modal:** tipo de **caixa de diálogo** que aparece quando você clica ou toca em algo em sua tela atual. Também são conhecidas como “janelas *pop-up*”.

- **Tratamento de cadastro:** apresenta o formulário para receber os dados para realizar *INCLUDE* de novo registro na tabela **usuário**.

Cadastrar

Nome de usuário*

Nome

Login*

nome.sobrenome

Celular*

(XX)XXXXXX-XXXX

Senha*

Confirma Senha*

☐ **Mostrar senha**

Enviar

Cancelar

- **Tratamento de falha:** aparece se houver erro de login ou de tentativa de cadastro.



Vamos analisar como essas janelas de modal são acionadas: basta alterar o **CSS** (com tratamento de evento **onclick**) para exibir a divisão (**style.display='block'**), onde os links para essas ações estão na divisão do Menu Superior.

Trecho do arquivo c:\xampp\htdocs\consultorio\index.php (sem as classes CSS para facilitar a leitura)

```
1 <div class="w3-top" id="LoginCadastro" >
2 ...
3 <a onclick="document.getElementById('id0L').style.display='block'" >Login</a>
4 <a onclick="document.getElementById('id0C'). style.display='block'">Cadastro</a>
5 ...
6 </div>
```

Cada divisão de janela de modal tem um elemento **HTML span**, em que é possível alterar o **CSS** (também com tratamento de evento **onclick**) para parar de exibir a divisão (**style.display='none'**).

Trecho do arquivo c:\xampp\htdocs\consultorio\index.php (sem as classes CSS para facilitar a leitura)

```
1  ...
2  <span onclick="document.getElementById('id0L').style.display='none'" title="Close Modal">x</span>
3  ...
4  <span onclick="document.getElementById('LF').style.display='none'" title="Close Modal">x</span>
5  ...
6  <span onclick="document.getElementById('id0C').style.display='none'" title="Close Modal">x</span>
```

Quando não há usuário logado, é possível visualizar o **menu superior**, com as opções *login* e cadastro.

- **Index.php: Opção *login*:** formulário de *login* tem . A página login.php: recebe **\$login** e **\$senha**, criptografa a **\$senha** com a função **SQL MD5** e compara com o que há na base de dados.

Trecho do arquivo c:\xampp\htdocs\consultorio\login.php

```
1  ...
2  // prepara a strings recebidas para ser utilizada em comando SQL
3  $usuario = $conn->real_escape_string($_POST['Login']);
4  $senha    = $conn->real_escape_string($_POST['Senha']);
5
6  // Faz Select na Base de Dados, criptografando a senha recebida para verificar se
7  // igual à mantida na tabela:
8  $sql = "SELECT ID_Usuario, nome FROM Usuario
9        WHERE login = '$usuario' AND senha = md5('$senha')";
10  if ($result = $conn->query($sql)) {
11      if ($result->num_rows == 1) {          // Deu match: login e senha combinaram
12          $row = $result->fetch_assoc();
13          $_SESSION ['login']          = $usuario; // Ativa as variáveis de sessão
14          $_SESSION ['ID_Usuario']     = $row['ID_Usuario'];
15          $_SESSION ['nome']           = $row['nome'];
16          unset($_SESSION['nao_autenticado']);
17          unset($_SESSION ['mensagem_header'] );
18          unset($_SESSION ['mensagem'] );
19          // Redireciona para a página de funcionalidades.
20          header('location: /ConsultorioLogin/medlistar.php');
21          exit();
22
23      }else{ // Não deu match: login ou senha incorretos.
24          $_SESSION ['nao_autenticado'] = true; // Ativa ERRO nas variáveis de sessão
25          $_SESSION ['mensagem_header'] = "Login";
26          $_SESSION ['mensagem']        = "ERRO: Login ou Senha inválidos.";
27          // Redireciona para página inicial
28          header('location: /ConsultorioLogin/index.php');
29          exit();
30      }
31  }
32  else {
```

```
33         echo "Erro ao acessar o BD: " . mysqli_error($conn);  
34     }  
35     ...
```

- **Index.php: Opção cadastro:** formulário de Cadastro . A página cadastro.php recebe **\$nome**, **\$login**, **\$celular** e **\$senha** e cria as respectivas **variáveis de sessão** que serão mantidas enquanto o usuário não **solicitar *logout***. São as variáveis de sessão que permitem ao **PHP** identificar se há usuário autenticado.

Trecho do arquivo c:\xampp\htdocs\consultorio\cadastro.php

```
1  ...
2  // prepara a strings recebidas para ser utilizada em comando SQL
3  $nome      = $conn->real_escape_string($_POST['nome']);
4  $login     = $conn->real_escape_string($_POST['Login']);
5  $celular   = $conn->real_escape_string($_POST['Celular']);
6  $senha     = $conn->real_escape_string($_POST['Senha1']);
7
8  // Faz Insert na Base de Dados
9  $sql = "INSERT INTO Usuario (Nome, Celular, Login, Senha)
10         VALUES ('$nome','$celular','$login',md5('$senha'))";
11
12  if ($result = $conn->query($sql)) {
13      $msg = "Registro cadastrado com sucesso! Você já pode realizar login.";
14      $_SESSION ['nao_autenticado'] = true;
15      $_SESSION ['mensagem_header'] = "Cadastro";
16      $_SESSION ['mensagem']       = $msg;
17      header('location: /ConsultorioLogin/index.php');
18      exit();
19  } else {
20      $msg = "Erro executando INSERT: " . $conn->connect_error .
21          " Tente novo cadastro.";
22      $_SESSION ['nao_autenticado'] = true;
23      $_SESSION ['mensagem_header'] = "Cadastro";
24      $_SESSION ['mensagem']       = $msg;
25      header('location: /ConsultorioLogin/index.php');
26      exit();
27  }
28  //Redireciona para página inicial, para tentativa de login com o
29  // usuário recém cadastrado
30      header('location: index.php');
31
32
```

```
33 | $conn->close(); //Encerra conexao com o BD
    ...
```

- **Menu.php** à São as variáveis de sessão que permitem ao **PHP** identificar se há usuário autenticado. Esse tratamento é feito no arquivo menu.php, inserido em todas as demais páginas. Logo, todas as páginas verificam se há **usuário autenticado logado** para **permitir acesso** aos seus conteúdos.

Trecho do arquivo c:\xampp\htdocs\consultorio\geral\menu.php

```
1 | ...
2 |
3 |     session_start();
4 |     if(!isset($_SESSION ['login'])){                // Se ainda não houve login
5 |         unset($_SESSION ['nao_autenticado']);
6 |         unset($_SESSION ['mensagem_header'] );
7 |         unset($_SESSION ['mensagem'] );
8 |         header('location: /ConsultorioLogin/index.php'); // Desvia para a página
9 |         exit();                                          // inicial.
10 |     }
11 | ...
```

Experimente executar todo o exemplo no seu equipamento individual:

1. **Cadastre mais um usuário.** Como a **senha ficou gravada** na base de dados? Foi possível **realizar o login** com esse **novo usuário**?
2. O que acontece se, ainda **não estando logado**, tentarmos acessar uma página qualquer da aplicação *web*, como:
http://localhost/ConsultorioLogin/medlistar.php? Que **mensagem** foi exibida?

Esta atividade pode ser utilizada para completar a **avaliação somativa 2**, iniciada com a **Atividade formativa** na unidade anterior.

Acrescentando as funcionalidades para **completar** o **CRUD** (acrescentaremos o *DELETE* e o *UPDATE*) e as funcionalidades para realizar um **login** para permitir o acesso às páginas da aplicação *web*.

Na sequência, temos a descrição do que é esperado para a **avaliação somativa 2**. Bom trabalho!

| Conclusão

Olá, estudante!

Esta foi nossa última unidade, em que conseguimos unir todas as tecnologias estudadas e praticadas na disciplina. Iniciamos com o básico da *web*, que é o **HTML**, progredimos para a construção de interfaces mais elaboradas, com o **HTML** e o **CSS**, para então juntar programação à interface, com o **JavaScript**. Com isso, completamos nosso roteiro para o *front-end* da aplicação, o lado **cliente**.

Na sequência, trabalhamos com a programação *back-end*, utilizando a linguagem **PHP**, para após, praticarmos a **persistência** em banco de dados, utilizando a linguagem **SQL**. Juntos, o **PHP** e o **SQL** permitem um tratamento completo das **regras de negócio** de uma aplicação, seus **requisitos funcionais**, com o **controle de dados** sendo realizado em um **servidor de BD**. Essa união no lado **servidor** permite que a aplicação *web* seja mais **robusta**¹ e tenha **escalabilidade**², pois ela tem a sustentação de servidores de mercado (**HTTP** e **SGBD**), preparados para esse tipo de demanda. Por último, também, tivemos a oportunidade de exercitar **mecanismos de autenticação**, inclusive com uso de **senhas criptografadas** mantidas no **SGBD**, para **permitir acesso** à nossa aplicação *web*.

Todas as nossas práticas envolveram tecnologias muito comuns de mercado, que também são muito didáticas para apresentar os **mecanismos essenciais que uma aplicação web necessita ter**. Mesmo que você não trabalhe, futuramente, exatamente com essas tecnologias, ao desenvolver todas as atividades propostas nesta disciplina, com certeza você terá uma base sólida de experiências que permitirão sua progressão na utilização de outras tecnologias, com segurança e conhecimento.

Parabéns pela dedicação! Espero que você continue a trabalhar com muito empenho e curiosidade. A área TI tem um universo de oportunidades e inovações, e precisa sempre de profissionais capacitados, autodidatas e esforçados!

Então, bom trabalho para atender por completo a todas as próximas entregas!

Até mais!

¹**Aplicação robusta:** pouco propensa a erros ou problemas, pois estes já estão previstos e tratados nos seus componentes.

Capacidade que a aplicação tem de estar preparada para tratar de erros e problemas, podendo continuar a funcionar mesmo em condições adversas.

² **Aplicação escalável:** capacidade que uma aplicação tem de continuar atendendo a um aumento considerável de requisições simultâneas, mantendo a performance das respostas. Habilidade para funcionar bem em situações redimensionadas.

| Referências Bibliográficas

ALVES, W. P. **Desenvolvimento e design de sites**. São Paulo: Erica, 2014.

MILETTO, E. M.; BERTAGNOLLI, S. C. **Desenvolvimento de software II: Introdução ao desenvolvimento web com HTML, CSS, Java Script e PHP**. Porto Alegre: Bookman, 2014.

TERUEL, E. C. **HTML 5: Guia prático**. 2. ed. Porto Alegre: Bookman, 2014.

