**Uma ação de login**

Bem vindo a segunda parte do nosso curso PHP Mysql e fundamentos da Web. Neste curso continuaremos a criação de um sistema de cadastro, listagem, atualização e remoção de produtos para adicionar uma camada de autenticação e autorização da minha loja: somente quem está cadastrado pode adicionar produtos. Veremos também como enviar email e boas práticas de código e da web, como por exemplo evitar concatenar strings (evitando SQL injection).

Se você não fez a primeira parte do curso verá no primeiro exercício como baixar o projeto e configurar o banco para continuar a partir deste ponto.

No nosso sistema a última coisa que fizemos foi deployar o mesmo na internet. Por isso temos que alterar no nosso arquivo de conexão com o banco e alterar os dados para abrir a conexão com nosso banco local. No arquivo*conecta.php* alteramos para comentar a linha de conexão remota e usar a local:

# $conexao = mysqli\_connect("mysql.hostinger.com.br", "u548680052\_loja", "lojaloja", "u548680052\_loja");

$conexao = mysqli\_connect("localhost", "root", "", "loja");

Quero me logar no meu sistema, mas para me logar tenho que criar uma tabela no banco de dados e para isso vou ao mysql no phpMyAdmin:

create table usuarios (id integer auto\_increment primary key, email varchar(255), senha varchar(255));

Preciso inserir também o meu usuário, o administrador do sistema:

insert into usuarios (email,senha) values ('guilherme.silveira@alura.com.br', '123456');

Mas poxa, se eu inserir esta senha, olha que bagunçado! Minha senha ficará armazenada solta, clara, em forma de texto em um banco de texto de uma empresa. Qualquer pessoa que tenha acesso a este banco terá acesso a senha pura de todos os usuários!

Na prática criptografaremos esta senha com o algoritmo MD5. Entramos em um site qualquer que faça encryption de md5 como md5encryption.com. Digito a minha senha '123456' e ele retorna a minha senha criptografada: 'e10adc3949ba59abbe56e057f20f883e'. Esta é minha senha que será armazenada no banco.

insert into usuarios (email,senha) values ('guilherme.silveira@alura.com.br', 'e10adc3949ba59abbe56e057f20f883e');

Quando eu me logar, o programa criptografará a senha '123456' novamente e terá um select que buscará 'e10adc3949ba59abbe56e057f20f883e'. O programa sempre pega a senha original e criptografa para conferir se as senhas são as mesmas.

Agora que temos nosso usuário vamos criar nosso formulário de login. No arquivo **index.php** adicionaremos ele, com os campos email e senha. Note que continuamos usando o bootstrap:

<h2>Login</h2>

<form action="login.php" method="post">

<table class="table">

<tr>

<td>Email</td>

<td><input class="form-control" type="email" name="email"></td>

</tr>

<tr>

<td>Senha</td>

<td><input class="form-control" type="password" name="senha"></td>

</tr>

<tr>

<td><button type="submit" class="btn btn-primary">Login</button></td>

</tr>

</table>

</form>

Testamos nossa aplicação e o formulário aparece bonito:



Vamos agora criar nossa lógica, a página **login.php** que recebe os dados do formulário, criamos esse arquivo e acessaremos o banco de dados:

<?php

include("conecta.php");

include ("banco-usuario.php");

Desejamos buscar um único usuário através de seu email e senha:

$usuario = buscaUsuario($conexao, $\_POST["email"], $\_POST["senha"]);

Queremos ter uma noção se funcionou ou não:

echo $usuario;

Vamos adicionar o arquivo **banco-usuario.php** com a função que desejamos, a que recebe a conexão, o email e a senha:

<?php

function buscaUsuario($conexao, $email, $senha) {

}

Desejamos fazer um select com o email e a senha:

<?php

function buscaUsuario($conexao, $email, $senha) {

$query = "select \* from usuarios where email='{$email}' and senha='{$senha}'";

}

Queremos agora executar essa query e retornar o primeiro elemento, usando as funções de **mysqli** que já conhecemos:

<?php

function buscaUsuario($conexao, $email, $senha) {

$query = "select \* from usuarios where email='{$email}' and senha='{$senha}'";

$resultado = mysqli\_query($conexao, $query);

$usuario = mysqli\_fetch\_assoc($resultado);

return $usuario;

}

Perfeito, o primeiro usuário será retornado. Se eu passar o email 'guilherme.silveira@alura.com.br' e senha '123456' ele fará a query:

"select \* from usuarios where email='guilherme.silveira@alura.com.br' and senha='123456'";

Mas não é isso que queremos, é? Desejamos criptografar a senha! Rodar o algortimo de md5, e o php já tem a função md5:

function buscaUsuario($conexao, $email, $senha) {

$senhaMd5 = md5($senha);

$query = "select \* from usuarios where email='{$email}' and senha='{$senhaMd5 }'";

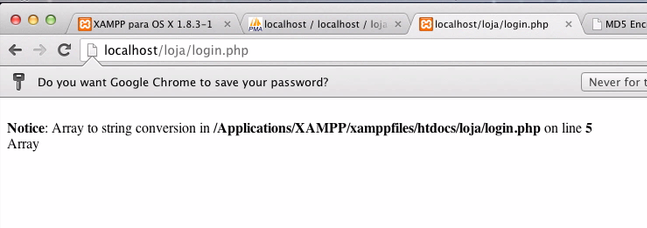
$resultado = mysqli\_query($conexao, $query);

$usuario = mysqli\_fetch\_assoc($resultado);

return $usuario;

}

Perfeito, agora sabemos que nossa senha será sempre criptografada antes de ser salva. Vamos testar? Vamos no navegador, colocamos os dados corretos de usuário e senha e temos:



Que estranho. Ele mostrou que tentou converter uma array em uma string. Faz sentido, queremos saber os valores de nossa array **usuario**, mas estamos tentando convertê-la em string.. e isso não é feito automaticamente pelo**echo**. Vamos então chamar a função que imprime a **array** com todos os seus valores, a **var\_dump**:

<?php

include("conecta.php");

include ("banco-usuario.php");

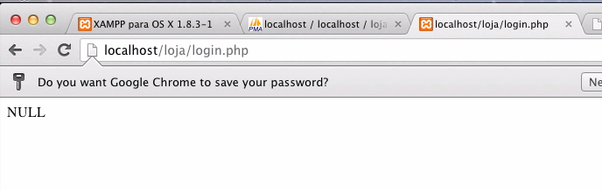
$usuario = buscaUsuario($conexao, $\_POST["email"], $\_POST["senha"]);

var\_dump($usuario);

Testamos novamente o formulário e agora temos o resultado positivo. Podemos ver o usuário e senha em nossa**array**:



Vamos tentar agora com um usuário inválido?



O resultado é NULL, vazio. Legal mas eu gostaria de mostrar uma mensagem de sucesso ou de fracasso. Vou então no **login.php** e adiciono uma condição de **sucesso** ou **fracasso**:

$usuario = buscaUsuario($conexao, $\_POST["email"], $\_POST["senha"]);

if($usuario == null) {

echo "Fracasso";

} else {

echo "Logado";

}

Maravilha, testamos e funciona. Mas ao invés de só mostrar uma mensagem seca, gostaria de voltar para a página principal com a mensagem de sucesso ou fracasso. Já sabemos fazer isso com o **redirect**, usando o **HeaderLocation**. Passamos também um parâmetro dizendo se o login foi um sucesso ou não:

$usuario = buscaUsuario($conexao, $\_POST["email"], $\_POST["senha"]);

if($usuario == null) {

header("Location: index.php?login=false");

} else {

header("Location: index.php?login=true");

}

die();

Vamos agora no index.php. Desejamos mostrar agora uma mensagem. Como checar se o parâmetro **login** existe na array **\_GET**?

isset($\_GET["login"])

Perfeito, se tem o parâmetro **login** **E** ele é true, então mostra a mensagem de sucesso:

<?php

if(isset($\_GET["login"]) && $\_GET["login"]==true) {

?>

<p class="alert-success">Logado com sucesso!</p>

<?php

}

?>

<?php

if(isset($\_GET["login"]) && $\_GET["login"]==false) {

?>

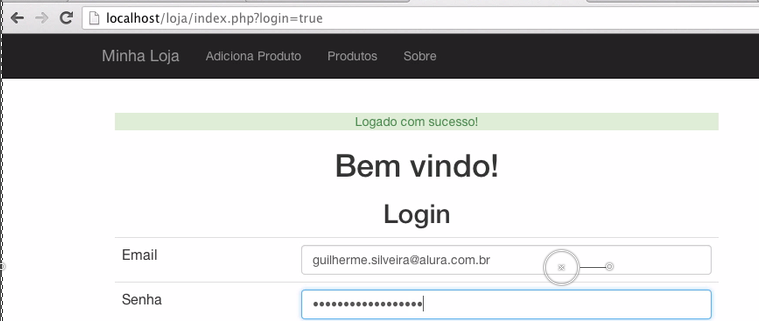
<p class="alert-danger">Usuário ou senha inválida!</p>

<?php

}

?>

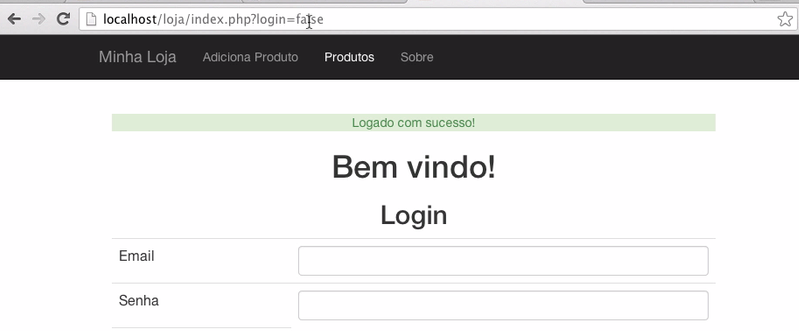
Testamos com um usuário válido e temos sucesso!



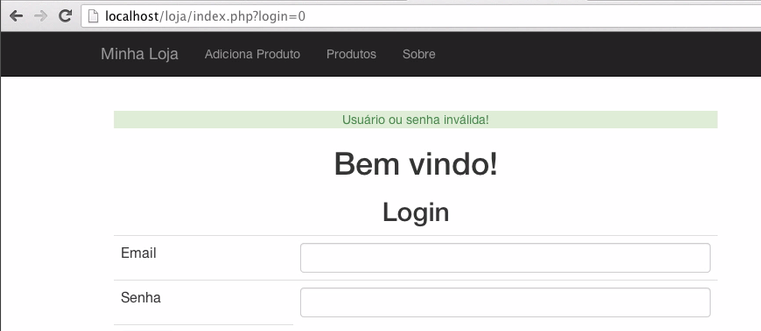
Testamos agora com uma senha inválida:



Não deu certo! Ele mostrou a mensagem de sucesso. Como assim? Vamos olhar nosso código. Sabemos que o parâmetro login é "false". Mas a string **false** vale **true**.... No php o **==** faz a conversão automática lembra? E nele, toda string é **true**. Somente duas strings valem false. Uma delas é **"0"** e a outra é a string vazia **""**. Vamos testar? Na URI testamos com **index.php?login=0**:



E com vazio:



Ambos agora funcionam. Então temos que passar **0** no nosso redirecionamento. **0** quando dá erro e **1** quando sucesso:

$usuario = buscaUsuario($conexao, $\_POST["email"], $\_POST["senha"]);

if($usuario == null) {

header("Location: index.php?login=0");

} else {

header("Location: index.php?login=1");

}

die();

Mas ainda falta um passo importante. Queremos que quando o usuário se logue, nós possamos lembrar quem ele é. Queremos mostrar um botão de logout. Queremos esconder o formulário de login quando o usuário já está logado, mostrando algo como "Bem vindo guilherme.silveira@alura.com.br". Queremos também esconder esses parâmetros 0 e 1 que aparecem na URI. Tem muita coisa que desejamos fazer ainda e veremos tudo isso nos próximos capítulos.