02

Implementando o Login

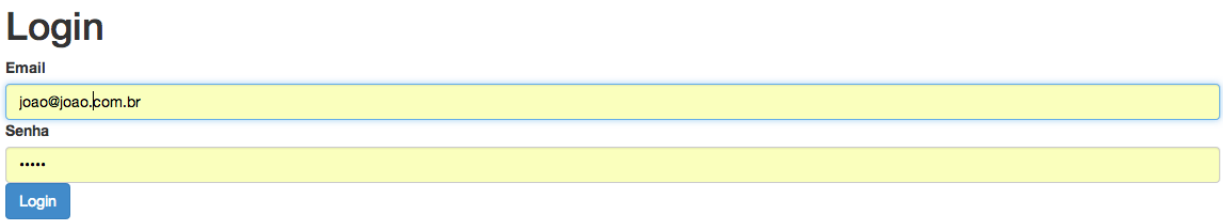
Vamos evoluir a aplicação, implementando a funcionalidade de login, no nosso `index.php` do produtos. Começamos com um formulário de Login, antes do nosso formulário de cadastro, vamos lá, este formulário ira acessar a uri `login/autenticar` :

```
<h1>Login</h1>
<?php
        echo form_open("login/autenticar");
        echo form_close();
        ?>
```

Quais campos iremos ter no formulário de login? E-mail, senha e o botão de login, esse botão é igual de cadastro, mas com o content diferente, vamos adicioná-los:

```
<?php
    echo form_open("login/autenticar");

    echo form_label("Email", "email");
    echo form_input(array(
        "name" => "email",
        "id" => "email",
        "class" => "form-control",
        "maxlength" => "255"
    ));

    echo form_label("Senha", "senha");
    echo form_password(array(
    "name" => "senha",
        "id" => "senha",
        "class" => "form-control",
        "maxlength" => "255"
    ));

    echo form_button(array(
        "class" => "btn btn-primary",
        "content" => "Login",
        "type" => "submit"
    ));


echo form_close();
?>
```

Acesse e veja como ficou:



Agora vamos criar um controller chamado `login` com o método `autenticar`, assim como definimos na uri do nosso formulário:

```
<?php if ( ! defined('BASEPATH')) exit('No direct script access allowed');

class Login extends CI_Controller {
public function autenticar() {
    }
}
```

Precisamos acessar o banco na tabela `usuarios` então vamos fazer o load:

```
public function autenticar() {
    $this->load->model("usuarios_model");
    }
```

Para fazer o login, precisamos buscar um usuário por email e pela senha. Esses dados vem para nós via post, através do formulário, então vamos pegar eles, lembre-se que a senha que temos no banco está criptografada através do `md5()`, então precisamos também chamar o `md5()`, para na hora que formos buscar do banco, encontrarmos o usuário correto:

```
public function autenticar() {
        $this->load->model("usuarios_model");
        $email = $this->input->post("email");
        $senha = md5($this->input->post("senha"));
    }
```

Agora precisamos de um método que faça essa busca para nós, lembrem-se os códigos que acessam o banco ficam no nosso modelo, se este método ira buscar um usuário no banco, então, ficará no `usuarios_model`, vamos fazer a chamada dele aqui:

```
public function autenticar() {
        $this->load->model("usuarios_model");
        $email = $this->input->post("email");
        $senha = md5($this->input->post("senha"));
$usuario = $this->usuarios_model->buscaPorEmailESenha($email, $senha);
    }
```

Agora iremos implementar o método `buscaPorEmailESenha`, no nosso `usuarios_model`, lembrem-se, quando queremos pegar ago do banco, usamos o `get()`, mas agora ao invés de pegarmos todas as linhas, pegaremos apenas a primeira, com o método `row_array()`, depois retornamos esse usuário:

```
public function buscaPorEmailESenha($email, $senha) {
    $usuario = $this->db->get("usuarios")->row_array();
    return $usuario;
    }
```

Legal, agora vamos restringir a nossa busca, trazendo os usuarios somente onde atendem as condições, onde o email e senha são iguais aos passados pelo formulário, para isso vamos adicionar o `where` :

```
public function buscaPorEmailESenha($email, $senha) {
    $this->db->where("email", $email);
    $this->db->where("senha", $senha);
    $usuario = $this->db->get("usuarios")->row_array();
    return $usuario;
}
```

Agora vamos voltar ao controller para verificar se o usuário realmente foi retornado, o código `if($usuario)` verifica se o usuário foi carregado, vamos colocar essa condição no nosso controller. página:

```
public function autenticar() {
        $this->load->model("usuarios_model");
        $email = $this->input->post("email");
        $senha = md5($this->input->post("senha"));
        $usuario = $this->usuarios_model->buscaPorEmailESenha($email, $senha);
        if($usuario) {

        } else {

        }
    }


}
```

Vamos sempre redirecionar o usuário para uma visualização que confirme a ação dele, neste caso, se ele conseguiu ou não se logar, o usuário ira para essa página independente do sucesso do login, então, o nosso redirecionamento ficára fora do `if` .

```
public function autenticar() {
        $this->load->model("usuarios_model");
        $email = $this->input->post("email");
        $senha = md5($this->input->post("senha"));
        $usuario = $this->usuarios_model->buscaPorEmailESenha($email, $senha);
        if($usuario) {

        } else {

        }

        $this->load->view('login/autenticar');

    }
```

```
    }
```

E agora vamos adicionar um mensagem para ser enviada para a view, assim como adicionamos os produtos, criamos um array, e passamos ele como segundo parâmetro do método que faz o redirecionamento.

```
public function autenticar() {
        $this->load->model("usuarios_model");
        $email = $this->input->post("email");
        $senha = md5($this->input->post("senha"));
        $usuario = $this->usuarios_model->buscaPorEmailESenha($email, $senha);
        if($usuario) {
            $dados = array("mensagem" => "Logado com sucesso");
        } else {
            $dados = array("mensagem" => "Usuário ou senha inválida.");
        }

        $this->load->view('login/autenticar',$dados);

    }


    }
```

Vamos agora criar a view `autenticar` na pasta `views/login` . Ela será simples e só ira mostrar o resultado:

```
<html>
    <body>
        <?=$mensagem?>
    </body>
</html>
```

Teste no seu navegador com usuário válido e um inválido.

Veja que no inválido tivemos um problema de encoding:

UsuÃ¡rio ou senha invÃ¡lida.

Vamos corrigir, avisando o HTML para usar o encoding UTF-8:

```
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" >
    </head>
    <body>
        <?=$mensagem?>
    </body>
</html>
```

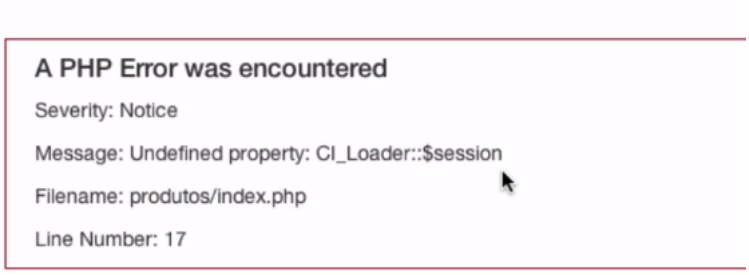
Só que ainda precisamos realmente logar este usuário, marcar o computador, como o computador do usuário, para que na próxima requisição a nossa aplicação já saiba quem é o usuário. Vamos usar a sessão para marcar o usuário como logado, para fazer isso no nosso controler vamos pegar a sessão e adicionar o usuário através da metódo `set_userdata()`

```
if($usuario) {
    $dados = array("mensagem" => "Logado com sucesso");
    $this->session->set_userdata("usuario_logado" , $usuario);
} else {
    $dados = array("mensagem" => "Usuário ou senha inválida.")
}
```

Agora na nossa view de index, vamos fazer assim, só vamos mostrar o formulário de se o usuário ainda não estiver logado, vamos envolver os dois formulários por um if:

```
<?php if(!$this->session->userdata("usuario_logado")) : ?>
formulários aqui
<?php endif ?>
```

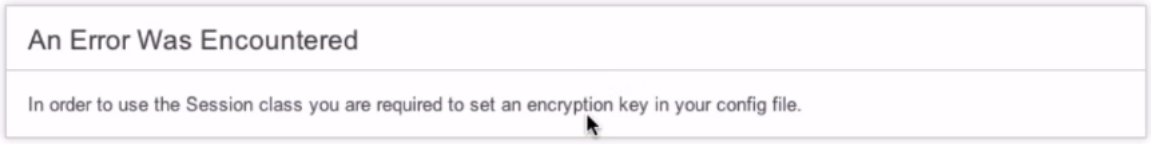
Agora teste no seu navegador, após se logar, o resultado, devera acontecer um erro, pois a session, assim como o banco de dados, precisa ser carregada.



Vamos carregá-la, no `autoload.php` vamos acrescentar o load da `session` :

```
$autoload['libraries'] = array('database', 'session');
```

Ao testarmos, receberemos mais um erro:



Pois para usar a sessão precisamos configurar uma chave de criptografia, vamos fazer isto no arquivo `config.php` , dentro da pasta `config` , adicionando uma `encryption_key` , que por padrão, é vazia, podemos usar números e letras aleatórias, ficara assim:

```
$config['encryption_key'] = '9843hufrh7n7983f443';
```

Agora sim podemos testar que o formulário não aparecerá, lembre-se de recarregar a página.