Instruções Básicas de Como Instalar e Rodar em Ambiente local o TesteFullStackPleno.

Esdras Fragoso da Silva Neto*

2019

Resumo

Este documento tem o objetivo de fornecer instruções de como instalar e rodar em ambiente local o TesteFullStackPleno, conforme o requisitado pela empresa Rockbuzz como requisito de admissão na vaga de emprego Desenvolvedor(a) Full Stack.

Palavras-chaves: PHP7.2; HTML5; CSS3; JavaScript:ES6;

Introdução

O teste consistiu em cria uma api restfull com recurso de posts e as ações de listar, adicionar e atualizar baseada no diagrama da figura 1.

Foi desenvolvida uma interface de autenticação dos usuários, além de uma interface restrita a usuários autenticados com características administrativas com formulário de postagem e edição de conteúdo. Similarmente foi desenvolvida uma interface com características de um blog para exibir as postagens.

Todo código do projeto está disponível no repositório do Github <https://github.com/fsdrasfragoso/TesteFullStackPleno>, com instruções básicas de como instalar e rodar em ambiente local.

[1].

1 Descrição do Código-fonte

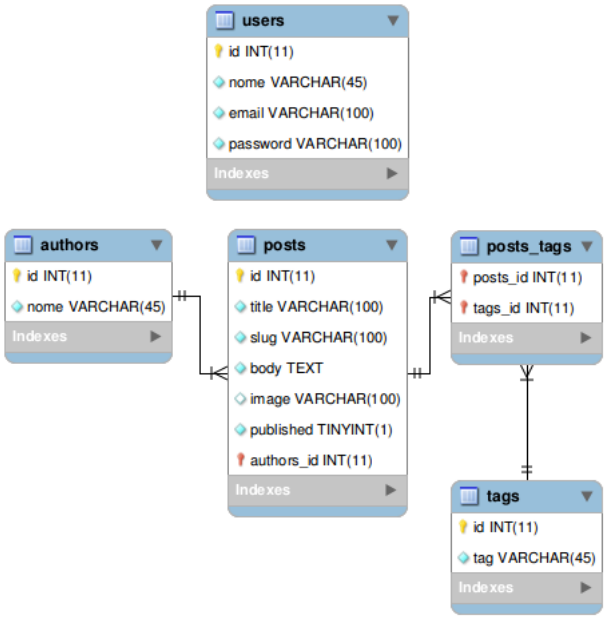
1.1 DB.class.php

O arquivo 'DB.class.php', é responsável pela conexão com Banco de Dados. Para fazer esta conexão a Class DB utiliza dos recursos do PHP Data Objects - PDO. A utilização do mesmo fornece uma camada de abstração em relação a conexão com o banco de dados visto que o PDO efetua a conexão com diversos bancos de dados da mesma maneira, modificando apenas a sua string de conexão.

*esdrasfragoso@gmail.com

[1] <https://github.com/fsdrasfragoso/TesteFullStackPleno>

Figura 1 – Diagrama do Banco dados



O código entre aspas a seguir: "'mysql:host=localhost;dbname=fullstack','root','ecioj'", é a String de conexão a ser modificada para instalar o sistema localmente. Primeiro passo é informar qual o tipo de banco de dados o usuário está utilizando, nesse caso como se pode notar é o Mysql. Em seguida se informa o endereço do servidor do Banco de Dados, que neste cenário é o endereço local. O passo seguinte é informar o nome do Banco de dados, seguido do usuário e senha que acessaram o servidor. Mais Abaixo poderá ser notado o código detalhado da Class DB.

```
  class DB{
private static $conn;
static function getConn(){
if(is_null(self::$conn)){
self::$conn = new PDO('mysql:host=localhost;dbname=fullstack','root','ecioj');
self::$conn->setAttribute(PDO::ATTR_ERRMODE,PDO::ERRMODE_EXCEPTION);
}
return self::$conn;
}
}
```

1.2 Login.class.php

A class Login é responsável pela autenticação dos usuários do sistema, utilizando do conceito de herança ela extends da class DB, herdando a conexão com o Banco de dados e os recursos do PDO. Alterando apenas três constantes é possível fazer a configuração dessa class, permitindo que ela se adeque a qualquer aos mais variados tipos de registro de usuários. As linhas a serem alteradas podem ser vista logo abaixo.

```
  class Login extends DB{
```

```
private $tabela = 'users';
private $prefix = 'fullstack_';
private $cookie = true;
public $erro = '';
```

A constante $tabela é responsável por receber o nome da tabela onde está sendo armazenados os usuários. A $prefix armazena o prefixo dos cookies de registro de login e login e senha, para salvar no navegador os endereço de email e a senha do usuario. Através desse prefixo é verificado se este um cookie do sistema salvo no navegador permitindo o mesmo lembrar de um ou mais dados de logins.

1.3 seguranca.php

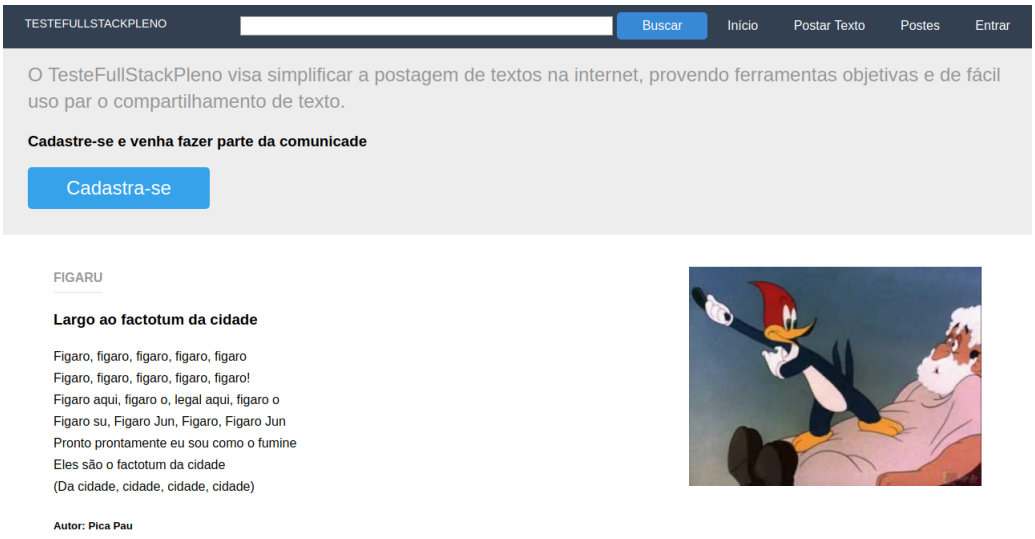
O arquivo seguranca.php é a API de segurança do sistema, ela fornece o serviço de instância de usuários e proteção das páginas. Quando ela é incluída na página a mesma passa a ser protegida podendo ser acessada somente por usuários logadas no sistema.

2 Interface com Caracteristicas de um Blog

2.1 index.php

A página index.php é responsável por exibir todas as postagens dos usuários, deixando-as visível para qualquer um que acessar o endereço da página. A interface da página como ilustrado pela figura 2 tem características comuns a de um blog. No topo o menu, no centro o conteúdo da página, seguido pelo rodapé.

Figura 2 – Pagina Index.php

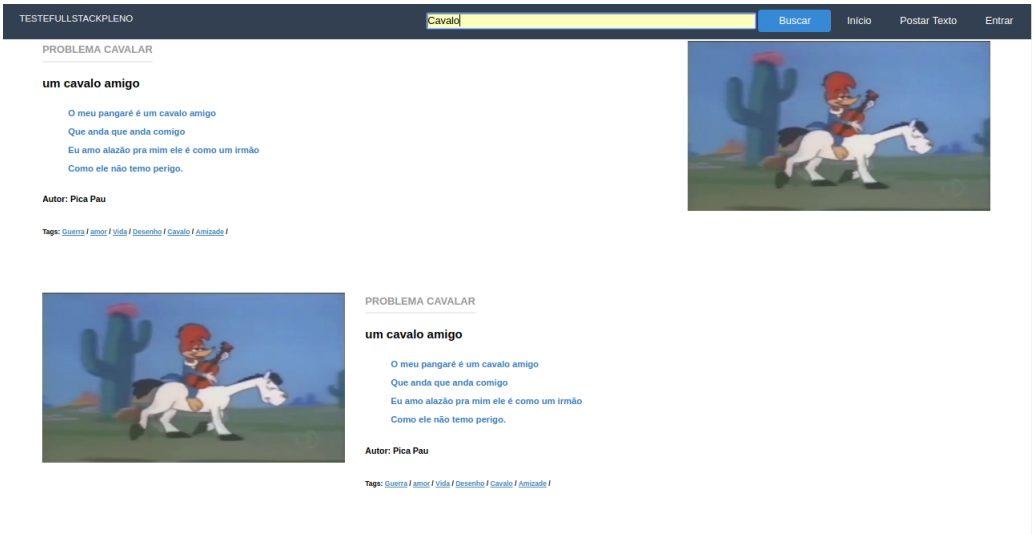


2.2 buscar.php

A página buscar.php é responsável exibir os resultados das buscas feitas pelos clientes que do blog. O resultado é exibido em duas partes primeiro o resultado da busca: pelo nome do autor do texto, título do texto, slug do texto e corpo do texto. depois é

exibido as postagens que possuem tag igual a palavra digitada no formulário de busca. Essa estrutura é ilustrada pela figura 3.

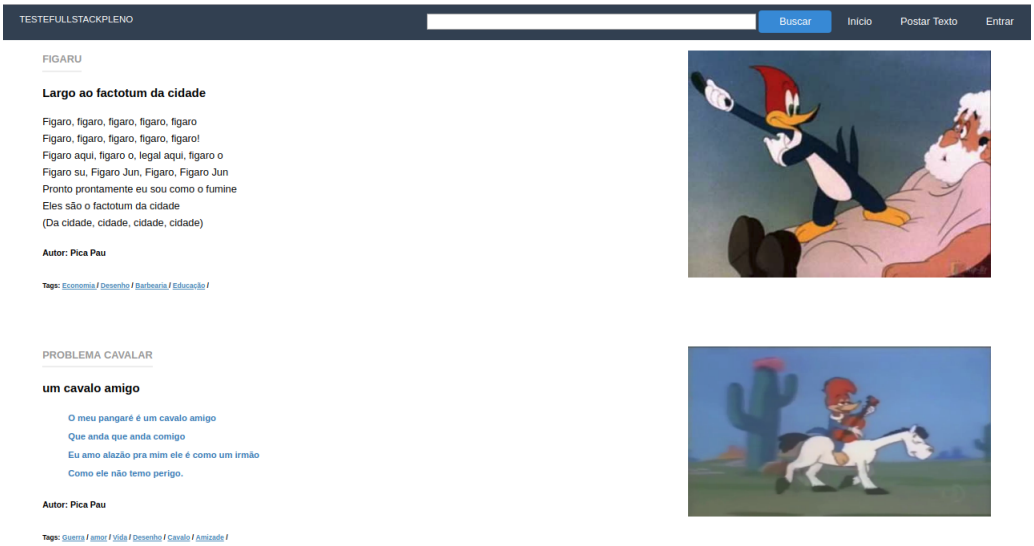
Figura 3 – Página buscar.php



2.3 tag.php

A página tag.php lista todos os postes que possuem a tag clicada. Cada tag exibida nas postagem é também um link que envia via $_GET para a página tag.php o id da tag em questão e através do mesmo é feito uma filtragem na tabela posts_tags para listar os postes que a possuem. A estrutura de como é feita essa listagem pode ser notado pela figura 4.

Figura 4 – Página tag.php




```
//código responsável por exibir as tags dos posts e gerar os links
 while($tag = $tags->fetch(PDO::FETCH_ASSOC)){
echo ' <a href="tag.php?id='.$tag['id'].'">'.$tag['tag'].'</a> / ';
```

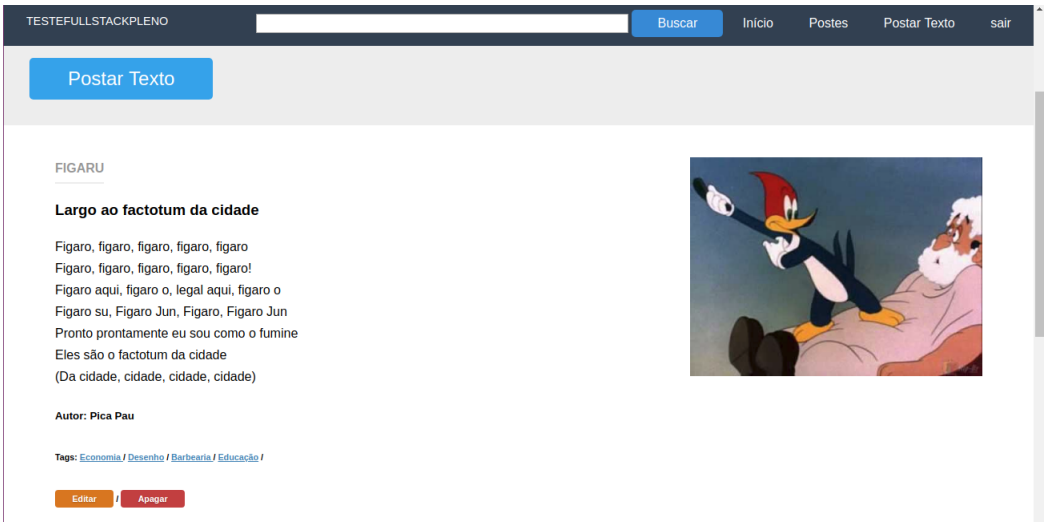
```
                }

/*Linha responsável por identificar o ID da tag
que de ser usada como filtro de listagem dos posts. */

if(isset($_GET['id'])){
            $tag_id = $_GET['id'] ;
```

3 Interface Restrita a Usuários Autenticados

3.1 editar.php

A página editar.php aproveita muito do design da página index.php como pode ser notado na figura 5, mas com o diferencial da inclusão dos botões: postar texto, editar e apagar. Onde os usuários devidamente logados podem tanto editar, como apagar e postar textos.

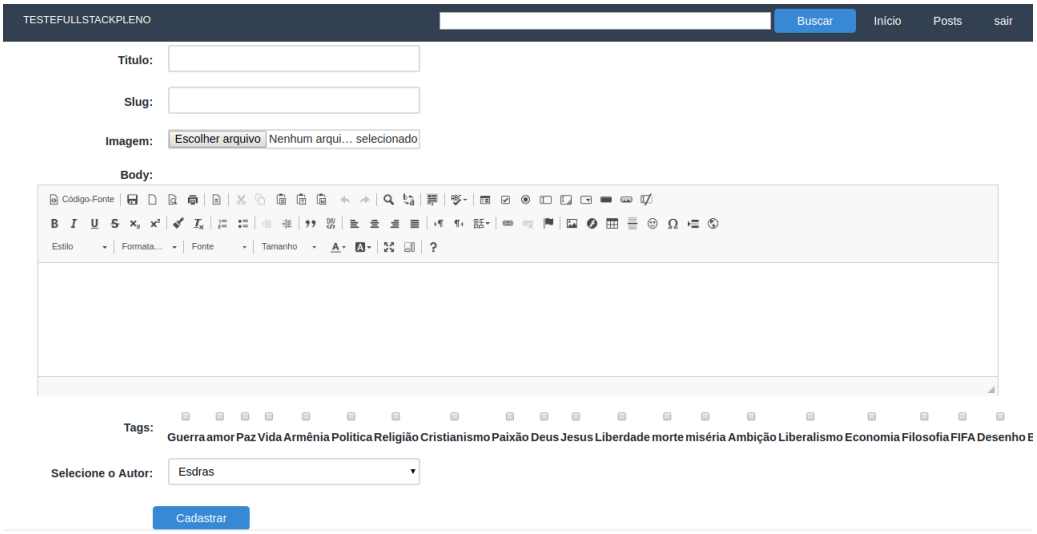
Figura 5 – Página editar.php



4 cadastroPoste.php

A página cadastroPoste.php é um formulário de cadastro de posts Onde o usuário deve escrever o título, depois o slug, escrever o corpo do texto no ckeditor, selecionar as tagas, caso não exista as tags que necessita basta clicar na opção outra que acionará o javascript e o jequery, que invocará uma API de cadastro de tag, para que a teg seja cadastra e já prontamente selecionada na página. Por fim é selecionado o autor do texto, se não tiver na lista do select basta selecionar a opção outro que novamente o jquery e entra em ação para requisitar uma API de cadastro de autores para já incluir na página o novo autor.

Figura 6 – Página CadastroPoste.php



Considerações finais

O Teste foi desenvolvido com boa estrutura e organização, usar bibliotecas específicas e não frameworks full stack's, o código inteligível, escalável, totalmente versionado do código pelo GIT, possui filtragem de posts pelas tags, pesquisas internas. O tempo de conclusão foi razoável. Todo código está disponível no Github pelo link: <https://github.com/fsdrasfragoso/TesteFullStackPleno>, e este documento detalha muito mais do que simplesmente instruções básicas de como instalar e rodar em ambiente local.

Por conta de todos estes fatores apresentados acima é que acredito merecer uma chance de integrar a equipe da Rockbuzz.