PEDRO ANTÔNIO DE SOUZA

DESENVOLVIMENTO DE SISTEMA WEB PARA CADASTRO DE NOTÍCIAS

LAVRAS-MG
2020

SUMÁRIO

1 INTRODUÇÃO

1.1 CONTEXTUALIZAÇÃO E MOTIVAÇÃO

Na disciplina de Programação WEB ofertada no segundo semestre de 2020, foram abordados os seguintes temas: programação front-end e back-end, integração com API e implementação de testes para aplicação. Ao final do curso, foi requerido, aos alunos, o desenvolvimento de um sistema web que utilizasse todos os quatro tópicos citados anteriormente e, também, persistência em banco de dados (BD).

Pensando nisso, foi proposto o desenvolvimento de um site de notícias. Em sistemas desse tipo, é comum que as notícias sejam persistidas em um BD para que os usuários administradores possam cadastrar, consultar, editar e excluir notícias. Portanto, para que os requisitos do trabalho sejam cumpridos, adotou-se a seguinte estratégia: o back-end foi desenvolvido em forma de API REST; o front-end foi integrado à essa API; e, por último, foram implementados testes unitários do Controller da API e testes automatizados dos tipos end-to-end e integração com API.

1.2 PROBLEMAS E OBJETIVOS

A Associação Atlética Acadêmica de Ciência da Computação e Sistemas de Informação Imperial, organização estudantil que tem como objetivo a integração de alunos da Universidade Federal de Lavras através do esporte, possui apenas as redes sociais como forma de comunicação com seus associados. Assim, nesse trabalho fez-se a elaboração do site **Imperial Press** no formato de noticiário para auxiliar a associação na difusão de seus informes.

2 LEVANTAMENTO DE REQUISITOS

A seguir são apresentados os requisitos do sistema.

REQUISITO 001: CADASTRAR NOTÍCIA

Efetuar o cadastro de uma notícia no sistema.

Prioridade: Essencial.

Entradas e pré-condições: O usuário deve estar logado no sistema.

Saídas e pós-condições: O sistema exibe que foi cadastrado com sucesso ou mensagem de erro caso ocorra.

REQUISITO 002: VISUALIZAR NOTÍCIA

Visualiza detalhes de uma determinada notícia previamente cadastrada.

Prioridade: Essencial.

Entradas e pré-condições: Não há.

Saídas e pós-condições: Não há.

REQUISITO 003: EDITAR NOTÍCIA

Editar informações de uma aventura previamente cadastrada no sistema.

Prioridade: Essencial.

Entradas e pré-condições: O usuário deve estar logado no sistema.

Saídas e pós-condições: O sistema exibe que foi atualizado com sucesso ou mensagem de erro caso ocorra.

REQUISITO 004: REMOVER NOTÍCIAS

Remover notícia previamente cadastrada no sistema.

Prioridade: Essencial.

Entradas e pré-condições: O usuário deve estar logado no sistema.

Saídas e pós-condições: O sistema exibe que foi excluída com sucesso ou mensagem de erro caso ocorra.

3 ESTRATÉGIAS DE CODIFICAÇÃO

3.1 BACK-END

O back-end foi arquitetado no estilo Representational State Transfer (REST). Assim, a criação, consulta, alteração e exclusão de notícias do banco de dados pode ser feita através das requisições HTTP do tipo POST, GET, PUT e DELETE, respectivamente.

Para o desenvolvimento, foi utilizada linguagem Java 11 e o Maven para gerenciar as dependências. O editor escolhido foi o Apache Netbeans 12.0. As principais dependências do projeto são as relacionadas ao Spring Framework. Esse framework traz uma série de funções implementadas para manipulação de dados com Java Persistence API (para transação de dados) e Hibernate (para mapeamento objeto-relacional), além de possuir um servidor Apache Tomcat integrado para rodar a aplicação.

A classe-entidade `Noticia` possui os atributos título, lide, corpo, data de criação e data de atualização. A classe-controladora `NoticiaController` é a responsável pelas lógicas de negócio. Essa classe recebe requisições HTTP e as mapeia para suas respectivas funções de serviço relacionadas ao banco de dados.

Para iniciar o servidor, no terminal navegue até o diretório `noticiario_rest/` e digite o comando `mvn spring-boot:run`. O servidor rodará em `http://localhost:8080/`.

3.2 FRONT-END

No front-end utilizou-se o HTML5, CSS3 e JavaScript, além do framework Vue.js. Esse framework possibilita a criação de componentes visuais que podem ser utilizados a qualquer momento e em qualquer página. Para rodar a aplicação, é necessário o uso do *node.js* e as dependências são gerenciadas com o *npm*. O Virtual Studio Code foi a IDE utilizada nessa parte do desenvolvimento.

Para iniciar a aplicação, no terminal navegue até o diretório `noticiario/` e digite o comando `npm run serve`. O servidor rodará em `http://localhost:4545/`.

3.3 INTEGRAÇÃO COM BANCO DE DADOS

O projeto é integrado com o banco de dados MySQL. No back-end, o Spring Framework e o Hibernate fazem todo o acesso e manipulação de dados no BD.

3.4 INTEGRAÇÃO COM API

Para integrar o front-end com a API do back-end, foi utilizada a biblioteca do *axios*, um cliente HTTP para navegadores que possibilita executar requisições.

4 TESTES DA APLICAÇÃO

4.1 TESTES UNITÁRIOS

Na realização dos testes unitários da classe controladora `NoticiaController`, foram utilizadas as bibliotecas Rest Assured MockMvc, Mockito e jUnit. Os testes tiveram como objetivo verificar o código HTTP das respostas das requisições. A tabela a seguir mostra a relação dos nove testes implementados.

Funcionalidade testada	Tipo de requisição	Detalhes da requisição	Resultado esperado	Resultado obtido
Consultar todas notícias	GET	Corpo vazio.	200 OK	200 OK
Cadastrar uma notícia	POST	Corpo com notícia válida.	200 OK	200 OK
Cadastrar uma notícia	POST	Corpo com notícia sem título.	400 Bad Request	400 Bad Request
Consultar uma notícia	GET	Corpo vazio para URL com id de notícia válido.	200 OK	200 OK
Consultar uma notícia	GET	Corpo vazio para URL com id de notícia inexistente.	404 Not Found	404 Not Found
Alterar uma notícia	PUT	Para URL com id de notícia válido.	200 OK	200 OK
Alterar uma notícia	PUT	Para URL com id de notícia inválido.	404 Not Found	404 Not Found
Excluir uma notícia	DELETE	Para URL com id de notícia válido.	200 OK	200 OK
Excluir uma notícia	DELETE	Para URL com id de notícia inválido.	404 Not Found	404 Not Found

4.2 TESTE DE INTEGRAÇÃO E PONTA A PONTA

Os testes de integração e de ponta a ponta foram realizados simultaneamente. Foi desenvolvido um programa em Python3.8, utilizando a biblioteca Selenium para automatizar o acesso à aplicação e testar as funções de inserção de notícia, visualização de

notícia todas as notícias, edição e remoção pela página de visualização de detalhes da notícia e edição e remoção pela página principal. Caso todas as etapas ocorram sem falhas, pode-se concluir que a integração com a API também ocorreu com sucesso.

Para iniciar o teste, no terminal navegue até o diretório do `test_end-to-end/`, digite o comando `python3.8 test_end-to-end.py` e siga as instruções.

5 RESULTADOS ALCANÇADOS

Apesar de não ter sido implementada a função de login de usuários administradores, o sistema desenvolvido simula bem a utilização de um sistema completo. O CRUD de notícias funcionou perfeitamente e todos os testes unitários da API tiveram resultado positivo.

No front-end foi aplicado o conceito *Single Page Application*, demandando menos recursos do usuário já que apenas módulos da página são atualizados a cada nova requisição. O site possui estilo recursivo, se adaptando para cada dispositivo que o acessa.

6 CONCLUSÃO

No desenvolvimento do projeto, percebeu-se a necessidade de refatoração do back-end e front-end. A princípio, ambos eram feitos em Java utilizando a arquitetura Model-View-Controller (MVC) com o framework Spring MVC. Assim, no mesmo projeto haviam arquivos do back-end e do front-end. Isso fez com que a complexidade de integração entre as camadas crescesse de forma assustadora conforme o sistema era incrementado. O mecanismo Thymeleaf era a solução escolhida para criação dos templates. No entanto, esse mecanismo cria templates do lado do servidor, o que não facilita o trabalho de desenvolvimento da interface do usuário. Para solucionar o problema, alterou-se a arquitetura do back-end para REST API.

Como esperado, o desenvolvimento do front-end se tornou independente do back-end. A fim evitar a replicação de código HTML na elaboração das páginas, foi estudada a biblioteca Vue.js. Esse recurso também possibilitou o desenvolvimento de uma *Single Page Application* para diminuir a utilização de recursos do cliente.

A utilização de APIs foi o grande facilitador do projeto. Não foi necessária a escrita de nenhum comando em SQL mesmo o sistema possuindo funções de inserção, consulta, alteração e remoção de informações em um banco de dados MySQL. Com a infinidade de frameworks disponíveis, na maioria dos projetos deve-se procurar utilizá-los com o propósito de favorecer a construção e manutenção de códigos.