Desafio Dev Full Stack – Coco Bambu

Por Lucas Monteiro Freitas

lucas.mont.freitas@gmail.com

www.linkedin.com/in/lucas-monteiro-freitas-299b8131b

1. Introdução

Este documento apresenta detalhes sobre o planejamento, implementação e testes para o Desafio Dev Full Stack (Estagiário) do Laboratório de tecnologia Coco Bambu.

2. Requisitos do sistema

Requisito 1 (RF01): **Permitir ao usuário realizar buscas por título ou autor, exibindo uma lista de livros correspondentes**

Foi feita uma interface capaz de pesquisar por livros através da Open Library API. Esse foi o Minimum Viable Product (MVP) inicial do projeto.

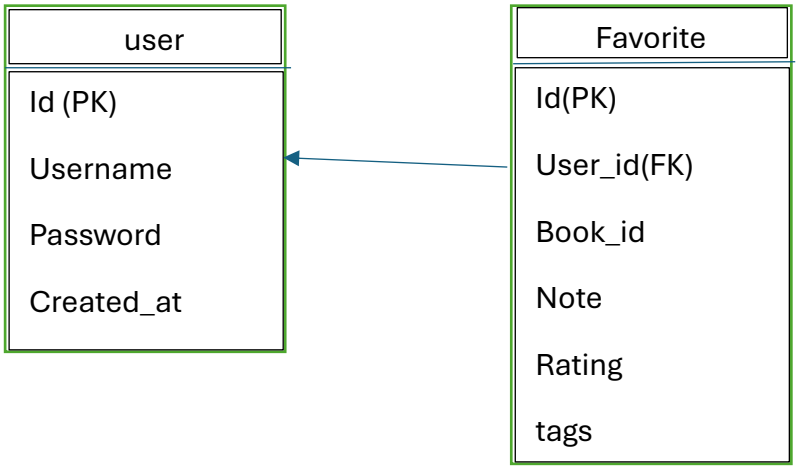
Requisito 2 (RF02): **Para cada livro listado, exibir informações relevantes como título, autor(es), descrição e capa.**

Foi acrescentado ao MVP a capacidade de apresentar após a pesquisa informações relevantes como título, autor(es), descrição e capa.

Requisito 3 (RF03): **Permitir ao usuário favoritar livros e adicionar notas pessoais, uma avaliação (nota de 1 a 5) e tags ao livro favoritado.**

Para permitir que o usuário o favoritar livros e adicionar notas pessoais, uma avaliação (nota de 1 a 5) e tags ao livro favoritado, foi necessário criar um banco de dados local para permitir que criassem um perfil e os livros favoritos persistissem em cada perfil.

O modelo ER do banco está na figura abaixo:



Requisito 4 (RF04): **Listar todos os livros favoritados, permitindo que o usuário edite suas anotações e avaliação.**

Com o banco de dados já estabelecido contendo as informações dos livros favoritos de cada usuário, foi criada uma área para listar todos os livros favoritados. Nessa área, o usuário pode editar suas anotações e avaliações

Requisito 5(RF05): **Implementar uma funcionalidade de filtro nas notas favoritas por tags.**

Com a implementação do uso de tags personalizadas pelos próprios usuários, o último passo foi desenvolver uma funcionalidade de filtro nas anotações favoritas por tags.

3. Critérios de Aceitação

3.1 Utilize a Google Books API ou outra API pública de livros para buscar as informações necessárias.

Foi utilizado o Open Library API disponível publicamente em https://openlibrary.org/dev/docs/api/books com acesso gratuito.

3.2 Utilize Angular.

Foi utilizada a versão 18.2.4 do angular.

3.3 Utilize ao menos 3 operadores RxJS na solução.

Os operadores *RxJS* utilizados no projeto foram: *From ele foi utilizado para transformar a lista de favoritos em uma sequência de valores a serem manipulados de forma assíncrona, mergeMap* foi usado para fazer requisições HTTP para buscar os detalhes dos livros favoritos, *map* foi usado para transformar os dados obtidos pela API Open Library, permitindo a adição de título e capa ao objeto *Favorite e toArray* foi usado para converter de volta os dados transformados em uma lista de favoritos.

3.4 Crie um README.md detalhado com instruções sobre o projeto, como executá-lo e como contribuir.

Readme do projeto disponível para acesso no repositório do projeto https://github.com/LucasMF1/BuscaLivro.git.

4. Diferenciais

4.1 Implementação de testes unitários.

Foram feitos testes padrões no uso do sistema, mas não foram formalizados.

4.2 Sugestões de melhorias ou críticas ao desafio, com argumentação embasada.

1. *Aprimorar o requisito de "implementação de testes unitários":*

O desafio menciona testes unitários como um diferencial, mas não especifica quais áreas devem ser prioritárias ou como a cobertura de testes será avaliada. Seria interessante definir uma meta de cobertura de código ou destacar partes críticas da aplicação que deveriam ser testadas, como os componentes Angular ou os serviços de backend. Isso evitaria uma implementação mínima de testes apenas para cumprir o critério.

2. *Critérios de avaliação não são claros:*

Não é claro como o projeto será avaliado. Por exemplo, a quantidade mínima de funcionalidades implementadas é suficiente? A qualidade do código será avaliada em conjunto com a performance? Um detalhamento maior nos critérios de aceitação e avaliação ajudaria os candidatos a focarem nos aspectos mais importantes.

4.3 Uso de ferramentas externas que facilitem seu trabalho, como IDEs, plugins e frameworks adicionais.

Uso do Visual Studio Code (VSCode) como a IDE, conjuntamente com suas extensões, tais como o Prettire, permitaram automatizar a formatação do código, garantindo padrões consistentes.

4.4 Otimizações e padrões de código.

Uso do ESLint para garantir que o código siga padrões de qualidade, evitando bugs comuns e melhorando a legibilidade. Uso do RxJS para melhorar o fluxo de requisições HTTP do angular.

4.5 Scripts que simplifiquem a execução do projeto, como comandos npm scripts.

Dois Scripts NPM foram criados para facilitar a execução do projeto, um para o *frontend* e outro para o *backend*.

Exemplos dos comandos npm:

1. cd BuscadorLivros.app (Navega para o diretório do frontend:)
2. npm install (Instala as dependências do frontend:)

3. cd backend (Navega para o diretório do backend:)
4. npm install (Instala as dependências do backend:)
5. npm run init-db (Inicializa o banco de dados:)