Departamento de Ciência da Computação

Profa. Ana Paula Piovesan Melchiori

DOCUMENTAÇÃO: minhaplay

Aluno: Mateus Carvalho Gonçalves
Matrícula: 201810245

Documentação entregue como parte das exigências para finalização da disciplina GCC 209 - Programação WEB

LAVRAS - DEZEMBRO/2020

1 INTRODUÇÃO

Este documento descreve o projeto da aplicação web **minhaplay**, uma ferramenta para gerência de uma playlist virtual.

O aplicativo minhaplay foi desenvolvido como parte das exigências para conclusão da disciplina eletiva GCC209 - Programação Web, ofertada pelo Departamento de Ciência da Computação da Universidade Federal de Lavras. Os requisitos básicos solicitados foram: desenvolvimento do front-end e back-end de um CRUD qualquer, com integração com banco de dados e API, e testes automatizados. A escolha das ferramentas foi encarregada aos alunos.

Dessa forma, o minhaplay é uma playlist web em que se é capaz de adicionar, mostrar, editar e excluir faixas, que podem ser redirecionadas para reprodução, e também é capaz de mostrar a letra da música - quando há.

2 REQUISITOS

Esta seção apresenta os requisitos idênticos ao projeto do trabalho, mudando apenas o tempo verbal quando necessário.

2.1 Requisitos Funcionais

RF01 - Adicionar faixa

Para adicionar uma faixa à playlist o usuário deverá informar o nome da faixa, o artista e uma URL para reprodução em qualquer plataforma de sua preferência.

RF02 - Mostrar playlist

O sistema deverá apresentar uma lista com todas as faixas da playlist.

RF03 - Editar faixa

O usuário poderá editar qualquer informação de uma música.

RF04 - Deletar faixa

Deve ser possível excluir uma faixa da playlist.

RF05 - Buscar por nome da faixa

O usuário deve poder buscar uma faixa por nome e o sistema deve retornar uma lista com todos os resultados da string inserida.

RF05 - Buscar por artista

O usuário deve poder buscar um nome de artista e o sistema deve retornar uma lista com todos os resultados da string inserida.

2.2 Requisitos de API

O sistema deve integrar a Vagalume API (disponível em: https://api.vagalume.com.br/) para que uma faixa também apresente sua letra, quando possível.

2.3 Requisitos de testes automatizados

O sistema deve conter pelo menos um teste unitário, um de integração (será feito funcional) e um ponta-a-ponta utilizando Selenium.

3 ESTRATÉGIAS DE CODIFICAÇÃO

Nesta seção serão discutidas as ferramentas e o passo-a-passo para preparação do ambiente para desenvolvimento do aplicativo, no sistema operacional Ubuntu 20.04.1 LTS.

3.1 Descrição das ferramentas

O ambiente de programação utilizado no projeto foi o VSCode, que suporta inúmeras linguagens de programação e fornece extensões para melhorar o desempenho e produtividade do programador.

Para o front-end foi utilizado HTML5 e CSS integrado nos arquivos HTML, com estilizações simples quando necessário. Isso porque a principal ferramenta de estilização utilizada foi o framework Bootstrap, que além de implementar algumas estilizações básicas também ajuda na responsividade. Todas essas ferramentas possuem uma documentação rica na internet, incluindo a da W3C. Além disso, o framework AJAX, desenvolvido a partir da linguagem Javascript, foi responsável pela integração do aplicativo com a Vagalume API com o HTML por meio do JQuery. A troca de mensagens é feita a partir de arquivos JSON. A integração do AJAX e Bootstrap com o HTML é dada por meio de tags script.

Já para o back-end, foi escolhido o framework Django v3.1.3, escrito em Python, por ser simples, rápido e fácil de programar. Possui sistema próprio de validação de campos de formulário, além de implementar protocolos de segurança como o Cross-Site Request Forgery (CSRF), e também possuir uma plataforma de auxílio para testes automatizados.

Além disso, o banco de dados integrado ao sistema foi o SGBD PostgreSQL v12.5, por ser mais completo que o SQLite, que vem por padrão em projetos Django.

Finalmente, como mencionado anteriormente, o Django fornece uma plataforma integrada de testes. Os testes devem ser desenvolvidos no arquivo ‘tests.py’ e toda função deve iniciar com a string test*. Por meio dela foram feitos testes unitários e testes funcionais simples de requisição. O Selenium foi utilizado para desenvolver testes funcionais e ponta-a-ponta, simulando o caminho percorrido pelo usuário, com o WebDriver do Chrome.

3.2 Preparação do ambiente e comandos importantes

A preparação do ambiente pode ser separada em 3 etapas: instalação do Django, criação do banco de dados, e criação do projeto. Todos os passos são utilizando o terminal. O guia completo é feito para que uma pessoa consiga começar um projeto django do início, caso queira apenas executar o presente projeto, é necessário filtrar as informações.

3.2.1 Instalação do Django

Para instalação do Django é necessário ter o Python3 instalado na máquina. A partir do Python, uma VirtualENV foi utilizada para criar um ambiente de programação isolado na máquina, ou seja, quando a venv está ativa, tudo que será instalado pertencerá apenas àquele ambiente, além de instalar alguns pacotes a mais do Python por padrão, como o pacote pip.

```
$ python3 -m venv <nome_venv>          // cria venv
$ source <nome_venv>/bin/activate      // ativa venv
$ pip install django                   // instala django
```

$ pip install psycpg2-binary // instala dependências do PostgreSQL

3.2.2 Criação do banco de dados

Nesta seção, será passado o passo-a-passo para criação e setup do banco de dados.

$ sudo su - postgres // entrou no terminal do postgres
postgres@<nome_maquina>:~$ // prompt ficará assim
$ createdb <nome_db> // criar e deletar um banco, respectivamente
$ dropdb <nome_db>
$ createuser -P <user> // cria usuário
$ psql <nome_db> // acessa terminal do banco
=> GRANT ALL PRIVILEGES ON DATABASE <nome_db> TO <user>;

O último comando define os direitos de acesso ao novo usuário.

Para sair do banco, executa-se o comando *\q*.

O banco está criado e funcionando, para sair do terminal do postgres, pressionar *ctrl+d*.

3.2.3 Criação do projeto e comandos importante

Nesta seção, será passado o passo-a-passo para criação do projeto Django, alguns passos iniciais e comandos importantes.

Primeiro, é necessário criar o projeto e a app, mudar o banco de dados e adicionar a app no arquivo 'settings.py'. Com isso, pode-se usar o 'migrate' para atrelar o banco ao projeto. Assim, é necessário apenas criar um superusuário do projeto para acessar o administrador web do banco e já é possível executá-lo no navegador.

$ django-admin startproject <nomeprojeto>
$ python manage.py startapp <nomeapp>
$ python manage.py migrate
$ python manage.py createsuperuser

O arquivo 'models.py' modela a tabela do banco. Assim, toda vez que esse arquivo é editado é necessário executar os comandos abaixo.

$ python manage.py makemigrations
$ python manage.py migrate

Finalmente, o comando runserver executa o servidor, permitindo que a aplicação rode em um navegador.
$ python manage.py runserver

Por fim, a plataforma de testes do Django oferece uma gama de comandos para executar os testes implementados, que pode ser consultado na documentação (https://docs.djangoproject.com/en/3.1/topics/testing/overview/#running-tests). O framework

cria um banco apenas para os testes, e ao final o destrói, para isso também é necessário executar o seguinte comando dentro do banco do projeto.

=> ALTER USER <username> CREATEDB;

4 RESULTADOS E DISCUSSÕES

Ao fim do desenvolvimento foi possível atender a todos os requisitos da disciplina e os requisitos levantados na seção 2 deste documento.

O aplicativo **minhaplay** pode ser executado localmente no navegador com um servidor rodando no terminal. É possível adicionar, listar, editar e excluir faixas de uma playlist.

Também é possível ser redirecionado para outro aplicativo para ouvir uma faixa e ver sua letra, quando possível. Houve dificuldade em imprimir a letra com as quebras de linha, uma vez que a API retorna a string com a quebra, mas ao colocar o resultado no input do formulário elas são ignoradas. Para solucionar esse problema, todos os '\n' foram trocados pela tag
 antes de atribuir a string ao input e assim foi salvo no banco, e quando o HTML utiliza o atributo o filtro de template "safe" é adicionado, para fazer com que o HTML também execute as tags presentes no atributo.

O mecanismo de busca consegue retornar resultados de buscas por nome da faixa e por artista. Porém, ele só retorna objetos com atributos exatamente iguais à string buscada. Outro problema é quando uma faixa tem nome e artista com o mesmo valor e uma busca por ela é feita, isso retorna uma lista com o objeto duplicado.

5 CONCLUSÃO

Conclui-se, portanto, que o desenvolvimento foi um sucesso considerando o objetivo técnico a ser cumprido.

Recomenda-se, como continuidade do projeto, buscar meios de resolver os problemas do mecanismo de busca citados na seção 4; fazer um estudo mais aprofundado sobre a forma utilizada para solucionar o problema da quebra de linha na letra e, se caso não for adequada, buscar outra solução; melhorar a aplicação, buscando um design e usabilidade de melhor qualidade, responsividade, acessibilidade; expandir o conceito do projeto, possibilitando criar várias playlists, fazer uma biblioteca de álbuns, etc.

Por fim, o objetivo educacional também foi alcançado. Com o projeto foi possível aprofundar na utilização das ferramentas em várias áreas do desenvolvimento web e recobrar todos os conceitos previamente estudados na disciplina.