PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL
ESCOLA POLITÉCNICA
CURSO DE BACHARELADO EM ENGENHARIA DE SOFTWARE

GUILHERME FARIAS STEFANI
HENRIQUE BAPTISTA DE OLIVEIRA
HENRIQUE CARDOSO ZANETTE
JHONATA SARAIVA PERES
LEONARDO SILVEIRA BERLATTO

SISTEMA DE CONTROLE DE TRÁFEGO AÉREO

Porto Alegre
2022

1. INTRODUÇÃO

Este trabalho visa apresentar e instruir sobre o funcionamento da aplicação do sistema de controle de tráfego aéreo, por meio da explicação dos *endpoints* criados e seus parâmetros. E também como utilizar a plataforma *Postman* para operar e testar o sistema.

2. *ENDPOINTS*

Os *endpoints* desenvolvidos para a aplicação de sistema de tráfego aéreo foram os seguintes:

2.1 */rotas/codigoAeroportuario/codigoAeroportuario2*

Este *endpoint* retorna uma lista de rotas entre dois aeroportos. É necessário passar dois parâmetros: o código aeroportuário 1, que será o aeroporto de um extremo inicial, e um código aeroportuário 2 para o aeroporto do extremo final. Um exemplo de parâmetros que podem ser passados é GRU, que representa o código aeroportuário do Aeroporto Internacional de Guarulhos, e POA que representa o Aeroporto Internacional Salgado Filho.

2.2 */planos-de-voo*

O *endpoint /*planos-de-voo é utilizado para cadastrar um plano de voo na aplicação, por meio de um *POST*. O *endpoint* não recebe nenhum parâmetro, entretanto é necessário passar um *body,* em formato de DTO com as informações do plano de voo para que o cadastro seja efetuado.

2.3 */planos-de-voo/verificar*

O *endpoint* /planos-de-voo/verificar pode ser utilizado para verificar se um plano de voo é válido em relação a sua altitude, velocidade de cruzeiro e se a aerovia da altitude passada está ocupada ou não. O endpoint não recebe nenhum parâmetro, porém é necessário passar um *body,* com o plano de voo as informações do plano em formato de DTO para que seja verificado se ele é válido. A resposta dessa requisição será uma lista com os problemas encontrados e um *boolean* se o plano é válido ou não.

2.4 */planos-de-voo/idPlanoDeVoo*

O *endpoint* /planos-de-voo/idPlanoDeVoo é utilizado para cancelar um plano de voo. Ele recebe um id como parâmetro para que a aplicação saiba qual plano cancelar. Ao cancelar um plano desocupa-se a rota que ele está ocupando, e o define como cancelado.

2.5 */aerovias/slots-livres/aeroviaId*

O *endpoint* /aerovias/slots-livres é utilizado para consultar os slots de horários livres de uma aerovia dada uma aerovia, um horário de partida e a velocidade de cruzeiro. Este *endpoint* não possui parâmetros, porém é necessário enviar um DTO com estas informações para que seja retornada uma lista dos slots livres da aerovia.

2.6 */aerovias/ocupacao/nome*

O *endpoint* /aerovias/ocupacao/nome serve para gerar o relatório de ocupação de uma aerovia para uma determinada data. O *endpoint* recebe o nome da aerovia como parâmetro e retorna um de relatório que contém o percentual de ocupação de cada altitude da aerovia ao longo do dia. O valor do percentual total é em relação ao número 1, ou seja, 1 representaria 100% de ocupação.

3. INSTRUÇÕES DE OPERAÇÃO DO *POSTMAN*

Para rodar a implementação do sistema, é necessário possuir a instalação do JDK 19, junto com o Maven. Com estas pré-configurações pode-se abrir e rodar o projeto para que ele então seja testado no *Postman,* seguindo os seguintes passos:

1. O passo inicial que deve ser realizado é importar a coleção da aplicação de sistema de controle de tráfego aéreo no *Postman*. O arquivo está localizado dentro da pasta *resources* da aplicação.

2. Com a coleção importada, é possível popular o banco de dados com planos de voo. Para fazer isso, deve-se rodar a requisição “Popular banco”, dentro do espaço SCTA.

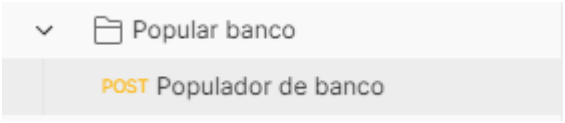

Imagem 1 - Pasta dentro da coleção do Postman.

3. Ao clicar na pasta, a seguinte tela é mostrada:

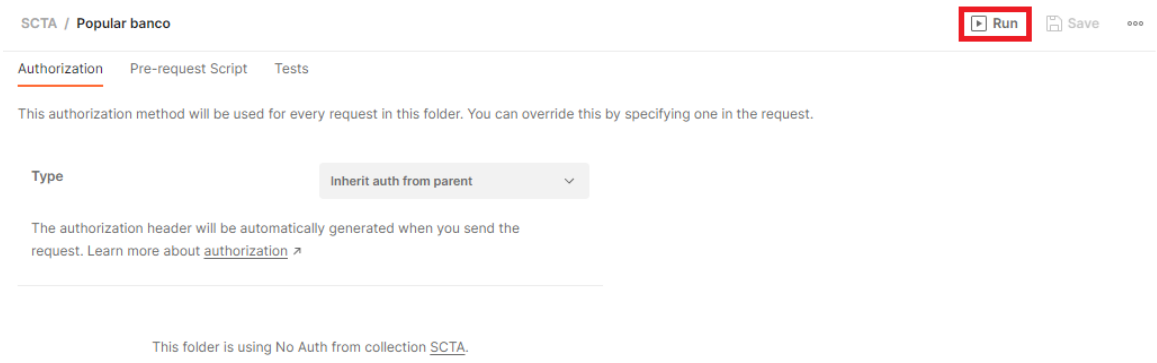

Imagem 2 - Tela da pasta do Popular banco.

Nesta tela, deve-se escolher a opção *Run*, para rodar o *script*.

4. Ao clicar no botão *Run*, a tela de execução é exibida.

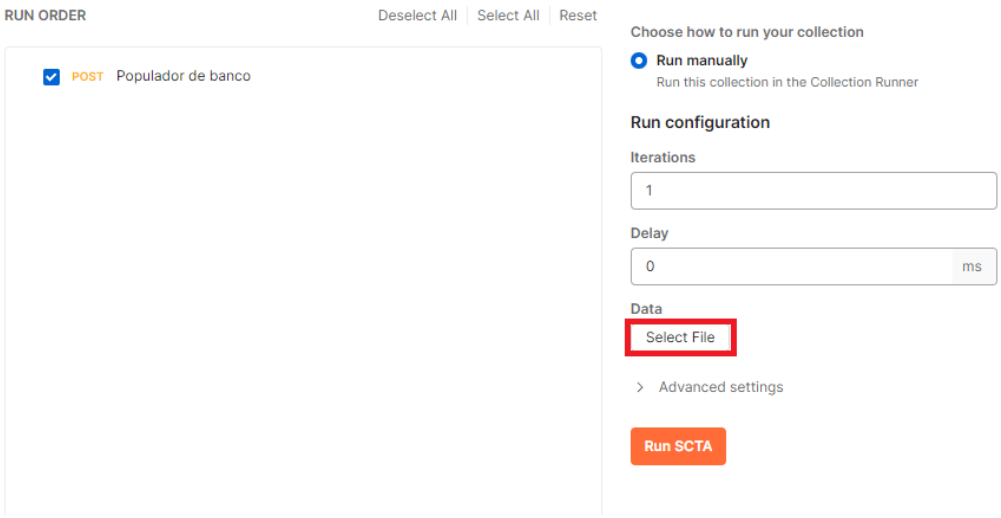

Imagem 3 - Tela de execução de *script.*

A partir disso, pode-se realizar o upload do arquivo JSON, que contém os planos de voo que irão popular o banco de dados.

5. Clicando na opção *Select File*, a opção de escolher arquivos será aberta.

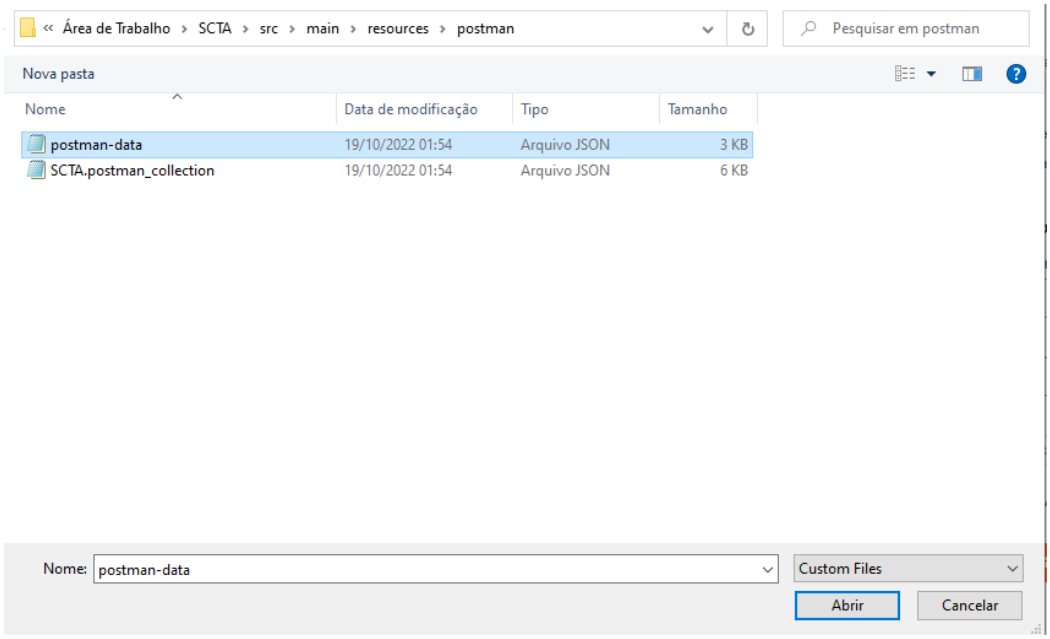

Imagem 4 - Seleção de arquivos.

O arquivo que contém os dados é nomeado de *postman-data* e está armazenado dentro da pasta *resources* da aplicação, junto ao *import* da coleção do *Postman.*

6. Com o arquivo carregado, as iterações apareceram como carregadas e pode-se rodar o *script*.

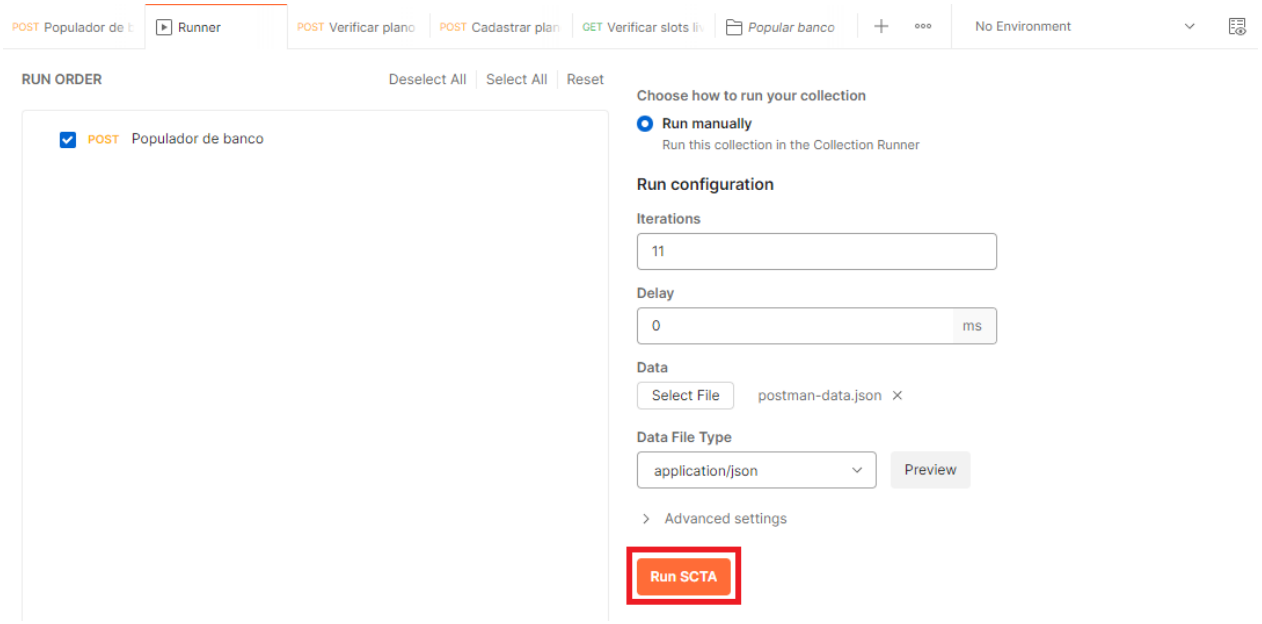

Imagem 5 - *Script* de popular banco carregado.

7. Com o banco de dados populado, pode-se rodar as outras requisições da aplicação, como validar um plano de voo ou checar as possíveis rotas entre dois aeroportos.