

Manual de Utilização - DEIJet

API

Pré-requisitos:

- IDE ou editor de texto (simplificar a execução de código);
- Python 3.11+ (versão utilizada para desenvolver a aplicação)
- Conta no Postman (execução de *requests* e verificação de erros);
- PostgreSQL + PgAdmin 4 (criação da base de dados).

Preparar a base de dados:

1. Criar uma base de dados no PgAdmin 4 e um utilizador;
 - a. *Database*: clicar sobre o servidor e seleccionar “Create “ --> “Database”;
 - b. Utilizador: “Login/ Group Roles”, ativar a opção “Can login?”
2. Configurar a ligação ao servidor com os dados do utilizador através do item “Register - Server”:
 - a. Configurar o acesso com o utilizador e a base de dados criados no passo anterior
3. Copiar o código de criação das tabelas e de procedimentos/funções necessários (deijet2024_data.sql), colar no *Query Tool* e executar.

Setup:

1. Instalação do ambiente virtual:

Recomenda-se o uso de um ambiente virtual que ajuda a gerir as dependências do projeto.

- a) Criar um ambiente virtual:

```
$ python3 -m venv deijet-api
```

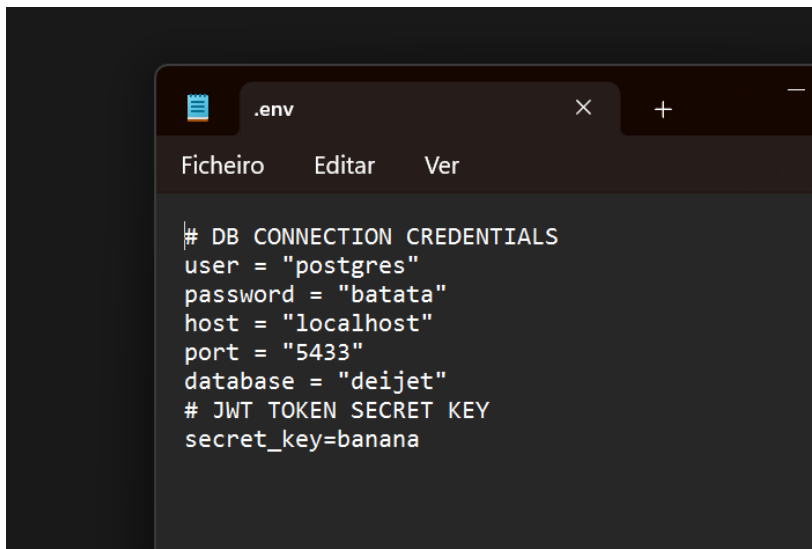
- b) - Ativá-lo:

Windows: executar: `$./deijet-api/Scripts/activate`

MacOS/Linux: executar: `$ source/deijet-api/bin/activate`

2. Variáveis de ambiente:

- Criar um ficheiro `.env`, com a estrutura do `.env.example`, localizado no path “python\app”, que tem os valores das variáveis de acesso definidos na ligação ao servidor e a chave que vai ser usada para codificar os *tokens*. Este ficheiro tem de ser inserido no *path* “python\app”.

A screenshot of a code editor window titled ".env". The window has a menu bar with "Ficheiro", "Editar", and "Ver". The code inside the editor is as follows:

```
# DB CONNECTION CREDENTIALS
user = "postgres"
password = "batata"
host = "localhost"
port = "5433"
database = "deijet"
# JWT TOKEN SECRET KEY
secret_key=banana
```

3. Instalação das bibliotecas necessárias:

- Para instalar as bibliotecas necessárias basta executar o seguinte comando no diretório *root* do projeto:
`$ pip install -r requirements.txt`

Utilização da API:

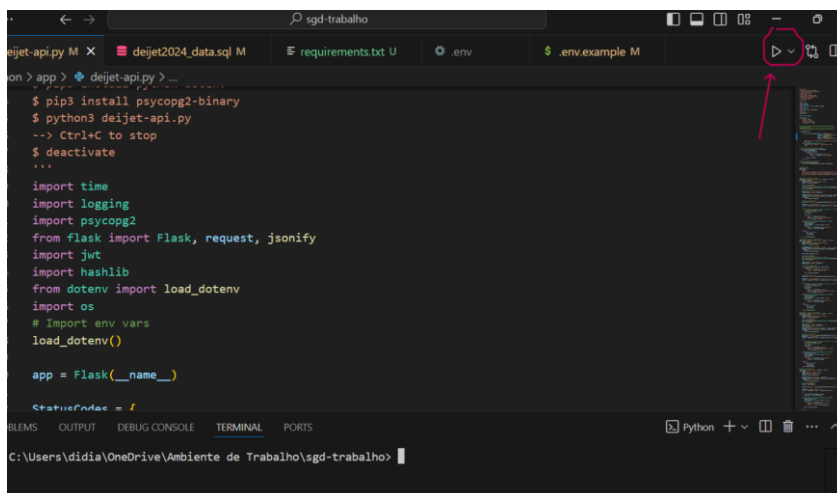
1. EndPoints:

- Existem 17 *endpoints* :
 - o <http://127.0.0.1:5000/>;
 - o <http://127.0.0.1:5000/sgdproj/register/client>; [POST]
 - o <http://127.0.0.1:5000/sgdproj/register/crew>; [POST]
 - o <http://127.0.0.1:5000/sgdproj/register/admin>; [POST]
 - o <http://127.0.0.1:5000/sgdproj/login>; [PUT]
 - o <http://127.0.0.1:5000/sgdproj/airport>; [POST]
 - o <http://127.0.0.1:5000/sgdproj/flight>; [POST]
 - o <http://127.0.0.1:5000/sgdproj/schedule>; [POST]
 - o <http://127.0.0.1:5000/sgdproj/assento>; [POST]
 - o http://127.0.0.1:5000/sgdproj/check_routes; [GET]
 - o http://127.0.0.1:5000/sgdproj/check_seats; [GET]
 - o http://127.0.0.1:5000/sgdproj/book_flight; [POST]
 - o <http://127.0.0.1:5000/sgdproj/bilhetes>; [GET]

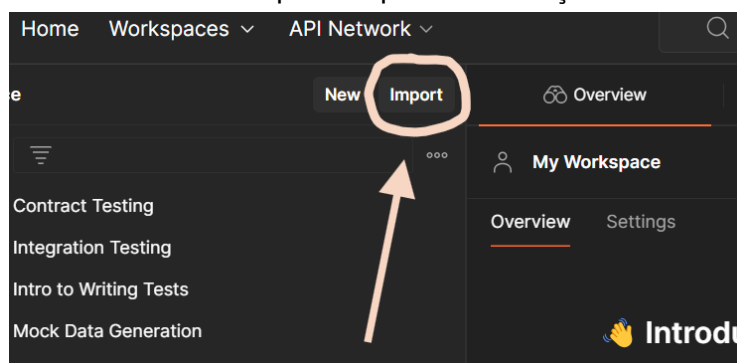
- [http://127.0.0.1:5000/sgdproj/report/topDestinations/<int:n>/\[GET\]](http://127.0.0.1:5000/sgdproj/report/topDestinations/<int:n>/[GET])
- [http://127.0.0.1:5000/sgdproj/report/topRoutes/<int:n>/\[GET\]](http://127.0.0.1:5000/sgdproj/report/topRoutes/<int:n>/[GET])
- [http://127.0.0.1:5000/sgdproj/payment/\[POST\]](http://127.0.0.1:5000/sgdproj/payment/[POST])
- [http://127.0.0.1:5000/sgdproj/report/financial_data/\[GET\]](http://127.0.0.1:5000/sgdproj/report/financial_data/[GET])

2. Testagem:

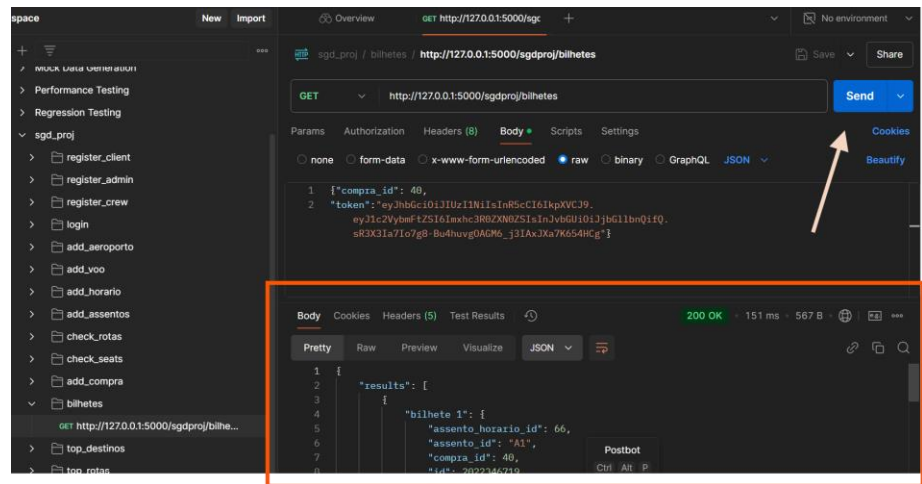
1. Executar o ficheiro “deijet-api.py”, localizado no *path* “python\app”:
 - a. Através do comando: `$ python3 deijet-api.py`
 - b. Através do IDE, clicando onde a seta aponta:



- c. Para parar de correr, fazer “Ctrl+C” no terminal.
2. Recorrer ao Web Browser (primeiro *endpoint*) ou ao Postman (restantes).
 3. Utilizar a coleção de *requests* do Postman que se encontra na pasta “postman”. A coleção está dividida em subpastas de acordo com a operação relacionada com o *request*, sendo que, cada um, inclui os parâmetros necessários para a interação com a API. A figura seguinte mostra onde clicar para importar a coleção:



4. Exemplo de um pedido:
 - A seta rosa aponta para onde clicar para enviar o pedido e o retângulo laranja indica o espaço onde a resposta aparece:



Notas:

- O PgAdmin, o Postman e o código da API têm de estar abertos para se conseguir testar;
- É importante ter em atenção que, inicialmente, a base de dados está, praticamente, vazia. Isso implica que se deve testar os pedidos da coleção pela ordem em que estão colocados uma vez que existem consultas que dependem de dados inseridos em pedidos anteriores e, portanto, se testadas primeiro, vão retornar vazio.
- Uma vez que apenas um administrador pode criar outro, o primeiro administrador é inserido na respetiva tabela quando se executa o código da criação das tabelas, tem como *username* “admin_root” e como *password* “senha_segura”.