



Projeto Final

Um pequeno resumo sobre psycopg, flask e flask-restful

O projeto a seguir trata-se do início da construção de um gerenciador de documentos, uma api a ser utilizada por um projeto no qual, documentos serão submetidos para treinamento de uma rede neural capaz de fazer a classificação dos documentos utilizando uma rede neural profunda.

O projeto é desenvolvido em dois grandes grupos: front-end e back-end e faz uso no front end do VueJs, uma biblioteca escrita em JavaScript para construção de interfaces, no back end faz uso de duas bibliotecas principais **psycopg2** e **Flask**, a primeira trata-se de um conector para postgres escrito em python3, a segunda é responsável em conjunto à biblioteca auxiliar **flask-restful** pela criação de um servidor restful que servirá de api a ser consumida pelo front-end.

Estrutura de pastas

```

projeto
|--- app.py
└── config-
    |--- database.py
└── models
    |--- Auth.py
    |--- Pdf.py
    |--- Person.py
    |--- User.py
└── ui
    └── node_modules
        └── dist
        └── test
            └── public
                └── src
                    |--- main.js
                    |--- router.js
                    |--- store.js
                    └── views
                        |--- Home.vue
                        |--- Dashboard.vue
                        |--- Register.vue
                    └── stores
                        |--- userStore.js

```

Back-end

Em sua criação, um novo usuário foi definido para a realização das consultas e a ele foi dado privilégios nas tabelas pertencentes às aplicações. Foram habilitadas opções de log na configuração assim como foram alterados os modos de acesso no arquivo pg_hba.conf.

Temos na pasta o arquivo app.py que é nosso arquivo de entrada para o back end, assim como a definição das rotas a serem utilizadas pelo front end. Nele são importadas as classes contidas na pasta model, que mapeiam diretamente para tabelas no banco de dados.

Essas classes possuem por padrão 4 métodos de conexão http, get, post, put e delete e cada método realiza uma conexão com o banco de dados. Essa conexão é realizada através de uma conexão padrão contida no arquivo config/database.py, que contém dois métodos init e cursor que retornam a conexão e o cursor respectivamente, para uso nas classes. Para se realizar uma query no banco, um método exposto pelo

cursor, chamado execute é utilizado; o método em si não realiza a query sozinha, após a chamada do execute no cursor dois caminhos podem ser tomados para que um commit ocorra no banco. Para selects pode ser utilizado o método fetchall e para as outras consultas o ato de fechar a conexão é o suficiente para tal através da chamada conn.commit().

Após finalizada a comunicação com o banco, os dados são processados para facilidade de utilização no front end pela função p, contida em todos as classes que cria um dicionário que quando usado como retorno da função é convertido automaticamente para o formato JSON que então será consumido pelo front-end.

Front-end

A interface utiliza a biblioteca **axios**, escrita em javascript, para realizar suas requisições na api exposta na porta **5000** através da execução do arquivo **app.py**.

Após o desenvolvimento completo da interface em VueJs, os arquivos .vue podem ser compilados para arquivos simples html e servidos em qualquer servidor php simples, no momento configurado para ser servido em uma subpasta chamada dist para que os testes possam ser feitos de forma menos disruptiva

