Continuação da Blueprint e Iniciando organização para utilizar SQLAlchemy

DEVinHouse

Parcerias para desenvolver a sua carreira

SENAI

<LAB365>

AGENDA

- Continuação da aula passada
- Criar instância de um banco em JSON
- Criar 3 endpoints para trabalhar com esse banco
- Introdução ao SQLAlchemy
- Modelagem de dados

Continuação da aula passada

- Baseado no código da aula passada (27/07/2022), dentro da pasta src/app iremos criar uma pasta chamada **db**, que será responsável por persistir os dados salvos num banco de dados criado em JSON e dentro desta pasta, iremos criar um arquivo chamado __init__.py e nele iremos criar 2 funções no momento.

Criar instância de um banco em JSON

```
from flask import json

def save(data):
  json_object = json.dumps(data, indent=4)
  with open("src/app/db/technologies.json", "w") as outfile:
      outfile.write(json_object)
def read():
  try:
    with open('src/app/db/technologies.json', 'r') as openfile:
      json_object = json.load(openfile)
      return json_object
  except:
    return None
```

Criar 3 endpoints para trabalhar com esse banco

- Nesse momento, iremos refatorar o endpoint de POST do technologies, para que de fato, ele possa salvar dados na nossa aplicação.
- Para tal, iremos criar uma pasta nova chamada Utils, e dentro dela iremos adicionar um arquivo __init__.py para a aplicação e escrever o seguinte método

```
def exist_key(json):
  if "id" in json and "tech" in json:
    return json
  else:
    None
```

- O propósito desse método é para verificar se todos os dados necessários já estão na requisição

Criar 3 endpoints para trabalhar com esse banco

- Nesse momento, iremos refatorar o endpoint de POST do technologies:

```
from src.app.utils import exist_key #Junto com as importações
from src.app.db import read, save  #Junto com as importações
  data = exist_key(request.get_json()) #Esse bloco até a ultima linha, adicionar na
função add_new_technology
  if data == None:
    return jsonify({"error": "Está faltando algum dos campos obrigatórios"}), 400
  techs = read()
  if techs == None:
    save([data])
    return jsonify([data]), 200
  techs.append(data)
  save(techs)
  return jsonify(techs), 200
```

Criar 3 endpoints para trabalhar com esse banco

- Nesse momento, iremos criar o endpoint de DELETE do technologies:

```
@technology.route('/<int:id>', methods = ["DELETE"])
def delete_technology(id):
  techs = read()
  if techs == None or len(techs) == 0:
    return jsonify({"error": "Não é possível excluir, pois não tem dados para
apagar"}), 400
  onlyTechExistents = []
  for i in techs:
    if i['id'] != id:
      onlyTechExistents.append(i)
  save(onlyTechExistents)
  return jsonify({"message": "Foi deletado com sucesso"})
```

Introdução ao SQLAlchemy

É um ORM (Object-relational mapping) que basicamente permite mapear as tabelas do banco em classes e objetos de forma fácil e prática. Para exemplificar vamos continuar a usar nosso exemplo anterior da tabela de usuários, primeiro vamos deletar a tabela.

DEVinHouse

Parcerias para desenvolver a sua carreira

OBRIGADO!

SENAI

<LAB365>