



Cloud

Nome do Aluno

Breno Schneider Salles de Oliveira

Descrição do Projeto

Este projeto é uma **API dockerizada** integrada com um banco de dados **PostgreSQL**, desenvolvida para **gerenciar usuários** com funcionalidades de registro, login e autenticação utilizando **JWT**. Utilizando tecnologias como **Python**, **FastAPI**, **Docker** e **SQLAlchemy**, a aplicação realiza **web scraping** no serviço **OpenWeatherMap** para obter a previsão do clima da cidade de **São Paulo** nos próximos **5 dias**. A API permite que os usuários autenticados consultem informações detalhadas como temperatura prevista, condições climáticas, umidade e velocidade do vento. A integração com o banco de dados assegura o armazenamento seguro das credenciais dos usuários e a gestão eficiente das sessões autenticadas, proporcionando uma experiência segura e confiável para acessar dados meteorológicos atualizados.

Funcionalidades Implementadas:

- **Scraping de Dados:** Coleta de informações específicas de **[OpenWeatherMap]**.
- **API RESTful:** Fornece endpoints para acessar os dados coletados.
- **Autenticação JWT:** Segurança para proteger os endpoints da API.
- **Containerização com Docker:** Facilita a implantação e execução da aplicação em qualquer ambiente.

Como Executar a Aplicação

Pré-requisitos

- [Docker](#) instalado na sua máquina.
- [Git](#) instalado.

Passo a Passo

1. Baixe o arquivo `docker-compose.yaml`:

- [docker-compose.yaml](#)

2. Certifique-se de que o Docker está rodando no seu computador

3. Iniciar os Contêineres com Docker Compose:

```
docker compose up -d
```

4. Acessar a Aplicação:

- A API estará disponível em <http://localhost:8000/docs>.

📋 Documentação dos Endpoints da API

Autenticação

- **Registrar Usuário**

- **URL:** /register/
- **Método:** POST
- **Body:**

```
{  
    "nome": "seu nome",  
    "email": "seu email",  
    "senha": "sua senha"  
}
```

- **Descrição:** Registra um novo usuário e retorna um token JWT.

- **Login**

- **URL:** /login/
- **Método:** POST
- **Body:**

```
{  
    "email": "seu email",  
    "senha": "sua senha"  
}
```

- **Descrição:** Autentica um usuário e retorna um token JWT.

Previsão do Tempo

- **Consultar Clima**

- **URL:** /consultar/
- **Método:** GET
- **Headers:**

```
Authorization: Bearer <seu_token_jwt>
```

- **Descrição:** Retorna a previsão do tempo baseada nos dados coletados pelo web scraping.
- **Resposta:**

```
{  
    "previsao": [  
        {  
            "tempo": "tempo 1",  
            "temperatura": 25,  
            "icone": "icon 1"  
        },  
        {  
            "tempo": "tempo 2",  
            "temperatura": 28,  
            "icone": "icon 2"  
        }  
    ]  
}
```

```
{  
    "data": "2024-10-15",  
    "temperatura_minima": 16.55,  
    "temperatura_maxima": 26.89,  
    "descricao": "nublado"  
},  
{  
    "data": "2024-10-16",  
    "temperatura_minima": 19.94,  
    "temperatura_maxima": 30.03,  
    "descricao": "nublado"  
}  
// Mais previsões...  
]  
}
```

📸 Screenshots dos Endpoints Testados

The screenshot shows the Postman application interface. The left sidebar has sections for Home, Workspaces, API Network, Collections, Environments, and History. The main area shows a POST request to `localhost:8000/registrar/`. The Body tab is selected, showing a JSON payload:

```
1 {  
2     "nome": "breno",  
3     "email": "exemplo@email.com",  
4     "senha": "senha"  
5 }
```

The response section shows a green **200 OK** status with a response time of 16 ms and a size of 253 B. The response body is a JSON token:

```
1 {  
2     "jwt": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJleGVtcGxvQGVtYWlsLmNvbSJ9.S0p2q-XP-Eamt_wQzAJTcJVz98wzjejJ36aIxjg4dFw"  
3 }
```

At the bottom, there are buttons for Postbot, Runner, Capture requests, Auto-select agent, Cookies, Vault, Trash, and Help.

Descrição da screenshot: Mostrando a resposta do endpoint `/registrar/`.

The screenshot shows a POST request to `localhost:8000/login`. The request body contains:

```

1 {
2   "email": "exemplo3@email.com",
3   "senha": "senha"
4 }

```

The response is a 200 OK status with a token in the JSON body:

```

1 {
2   "jwt": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJleGVtcGxvM0B1bWFpbC5jb20ifQ.tSwhnW7wIXKDabkn6nAZBqiiqx1iKFEPoFmMs6DBQXs"
3 }

```

Descrição da screenshot: Mostrando a resposta do endpoint `/Login/`.

The screenshot shows a GET request to `localhost:8000/consultar/`. The response is a 200 OK status with a JSON body containing weather forecasts:

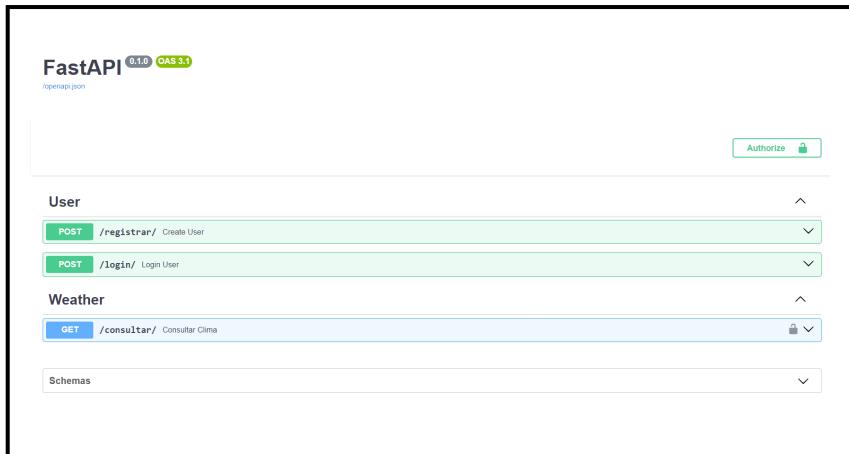
```

1 {
2   "previsao": [
3     {
4       "data": "2024-10-17",
5       "temperatura_minima": 26.34,
6       "temperatura_maxima": 26.34,
7       "descricao": "céu limpo"
8     },
9     {
10       "data": "2024-10-18",
11       "temperatura_minima": 18.11,
12       "temperatura_maxima": 30.56,
13       "descricao": "nublado"
14     },
15     {
16       "data": "2024-10-19",
17       "temperatura_minima": 15.36,
18       "temperatura_maxima": 19.51,
19       "descricao": "chuva leve"
20     },
21     {
22       "data": "2024-10-20",
23       "temperatura_minima": 14.43,
24       "temperatura_maxima": 17.09,
25       "descricao": "chuva leve"
26     },
27     {
28       "data": "2024-10-21",
29     }
30   ]
31 }

```

Descrição da screenshot: Mostrando a resposta do endpoint `/consultar/` com a previsão do tempo.

🎥 Vídeo de Execução da Aplicação



Descrição: Vídeo demonstrando a execução da aplicação e a interação com os endpoints da API.

📦 Link para o Docker Hub do Projeto

[Docker Hub Repository](#)

Descrição: Link para a imagem Docker do projeto no Docker Hub.