📌 Documentação da API - Serviço de Autenticação (WSO2)

    Projeto: Sistema de Autenticação com OAuth 2.0 e WSO2
    Tecnologias: Flask, OAuth2, PostgreSQL
    Objetivo: Autenticar utilizadores da Universidade de Aveiro através do WSO2 e armazenar informações de login.

📌 1. Instalação e Configuração
1.1. Pré-requisitos

Antes de executar o serviço, é necessário garantir que o ambiente está corretamente configurado:

    Python 3.x instalado
    Virtual Environment (venv)
    PostgreSQL configurado e acessível
    WSO2 Gateway configurado para OAuth2
    Kafka instalado (se for necessário para logs)

1.2. Instalar dependências

Dentro do diretório do projeto, execute:

```
python3 -m venv venv
source venv/bin/activate  # No Windows use venv\Scripts\activate
pip install -r requirements.txt
```

1.3. Configuração do Banco de Dados

Antes de iniciar a API, configure a base de dados PostgreSQL:

    Criar a base de dados:

```
CREATE DATABASE PointSystemEGS;
```

Configurar a URL da base de dados no .env:

```
    DATABASE_URL=postgresql://postgres:password@localhost:5432/PointSystemEGS
```

📌 2. Endpoints da API

A API expõe endpoints para autenticação via OAuth2, acesso protegido e logout.
◇ 2.1. Login

    Endpoint: GET /login
    Descrição: Redireciona o utilizador para a página de login do WSO2.
    Resposta: Redireciona para http://localhost:5000/callback após autenticação.

📌 Exemplo de Requisição:

```
GET http://localhost:5000/login
```

◇ 2.2. Callback

Endpoint: GET /callback
Descrição: Processa a resposta do WSO2 e obtém o token de acesso.
Parâmetros:
code: Código de autorização retornado pelo WSO2.
state: Estado da requisição (opcional).
session_state: Estado da sessão do WSO2 (opcional).
Resposta: Retorna um JSON com o token de acesso.

📌 Exemplo de Requisição:

```
GET http://localhost:5000/callback?code=123456&state=abc123
```

📌 Exemplo de Resposta (sucesso):

```
{
  "message": "Login bem-sucedido!",
  "access_token": "eyJhbGciOiJIUz..."
}
```

📌 Exemplo de Resposta (erro):

```
{
  "error": "Authorization failed"
}
```

◇ 2.3. Rota Protegida

Endpoint: GET /protected
Descrição: Permite acesso a um recurso protegido, apenas para utilizadores autenticados.
Cabeçalhos:
Authorization: Bearer <access_token>

📌 Exemplo de Requisição:

```
GET http://localhost:5000/protected
Authorization: Bearer eyJhbGciOiJIUz...
```

📌 Exemplo de Resposta (sucesso):

```
{
  "message": "Bem-vindo, Gabriel!"
}
```

📌 Exemplo de Resposta (erro):

```
{
  "message": "Token inválido!"
}
```

◇ 2.4. Logout

Endpoint: GET /logout
Descrição: Remove a sessão do utilizador.

Resposta: Redireciona para a página inicial.

📌 Exemplo de Requisição:

```
GET http://localhost:5000/logout
```

📌 3. Fluxo de Autenticação

O utilizador acede a /login → Redirecionado para WSO2.
Após autenticação, WSO2 redireciona para /callback com um code.
O backend troca esse code por um access_token.
O utilizador pode agora aceder a endpoints protegidos usando esse token.

📌 4. Configuração do Proxy (se necessário)

Se for necessário um proxy para redirecionar as chamadas do WSO2 para o Flask, configure um .env no proxy:

```
FLASK_RUN_PORT=5000
FLASK_RUN_HOST="0.0.0.0"

IDP_REDIRECT_URI="http://localhost:5000"
IDP_BASE_URL="https://wso2-gw.ua.pt"
CLIENT_ID="agh44RajMJcYvCIq3lSMrutfPJ0a"
CLIENT_SECRET="tMd7PPpzIR2JaY4u_dWEhr9kW9Ya"
```

📌 5. Solução de Problemas (FAQ)
❓ O login redireciona para "Authorization Failed"

☑ Solução:

Verifique se o client_id e client_secret estão corretos.
Teste a API manualmente com:

```
curl -X POST "https://wso2-gw.ua.pt/token" \
-H "Content-Type: application/x-www-form-urlencoded" \
-d "grant_type=authorization_code&code=SEU_CODIGO&redirect_uri=http://localhost:5000&client_id=SEU_CLIENT_ID&client_secret=SEU_CLIENT_SECRET"
```

❓ O WSO2 rejeita a requisição com "invalid_client"

☑ Solução:

Use Basic Authentication no cabeçalho:

```
import base64

auth_header = base64.b64encode(f"{client_id}:{client_secret}".encode()).decode()
headers = {'Authorization': f'Basic {auth_header}', 'Content-Type': 'application/x-www-form-urlencoded'}
```

❓ O servidor Flask não inicia porque a porta 5000 está ocupada

☑ Solução:

    Identifique o processo que está ocupando a porta:

```
sudo lsof -i :5000
```

Mate o processo:

```
sudo kill -9 <PID>
```

Reinicie o Flask em outra porta:

```
    flask run --host=0.0.0.0 --port=5001
```

📌 6. Conclusão

Esta API permite autenticação segura via WSO2, armazenamento de utilizadores no PostgreSQL e acesso a rotas protegidas via token.
Caso existam problemas, consulte a seção de Solução de Problemas ou contacte o suporte técnico do projeto.