### **1. Configurar o Projeto da API**

**Objetivo**: Criar uma API simples com Flask.

**Crie uma pasta para o projeto**:  
bash  
  
mkdir lab\_api && cd lab\_api

Texto

O conteúdo gerado por IA pode estar incorreto.

**Crie o arquivo principal da API (app.py)**:  
python  
  
from flask import Flask, jsonify

from flask\_swagger\_ui import get\_swaggerui\_blueprint

from flask\_jwt\_extended import JWTManager, create\_access\_token, jwt\_required

app = Flask(\_\_name\_\_)

# Configuração do JWT

app.config['JWT\_SECRET\_KEY'] = 'your\_secret\_key'

jwt = JWTManager(app)

### Swagger UI ###

SWAGGER\_URL = '/swagger'

API\_DOC\_URL = '/static/swagger.json'

swaggerui\_blueprint = get\_swaggerui\_blueprint(SWAGGER\_URL, API\_DOC\_URL)

app.register\_blueprint(swaggerui\_blueprint, url\_prefix=SWAGGER\_URL)

@app.route('/')

def home():

return jsonify(message="API is running")

@app.route('/items', methods=['GET'])

def get\_items():

return jsonify(items=["item1", "item2", "item3"])

@app.route('/login', methods=['POST'])

def login():

access\_token = create\_access\_token(identity="user")

return jsonify(access\_token=access\_token)

@app.route('/protected', methods=['GET'])

@jwt\_required()

def protected():

return jsonify(message="Protected route")

if \_\_name\_\_ == '\_\_main\_\_':

app.run(host='0.0.0.0', port=1313)

**Crie o arquivo requirements.txt com as dependências**:  
txt  
  
Flask==2.3.2

Flask-JWT-Extended==4.4.4

Flask-Swagger-UI==4.11.1

**Instale as dependências**:  
bash  
  
pip install -r requirements.txt

### **2. Criar o Dockerfile**

**Objetivo**: Containerizar a API para replicação e deploy.

**Crie o arquivo Dockerfile**:  
dockerfile  
  
# Usar uma imagem base oficial do Python

FROM python:3.9-slim

# Definir o diretório de trabalho

WORKDIR /app

# Copiar dependências

COPY requirements.txt requirements.txt

# Instalar dependências

RUN pip install -r requirements.txt

# Copiar os arquivos da API para o container

COPY . .

# Expor a porta 1313

EXPOSE 1313

# Comando para rodar a aplicação

CMD ["python", "app.py"]

Tela de celular com letras

O conteúdo gerado por IA pode estar incorreto.

### **3. Criar o Docker Compose**

**Objetivo**: Facilitar a execução e gerenciamento dos containers.

**Crie o arquivo docker-compose.yml**:  
yaml  
  
version: '3'

services:

api:

build: .

ports:

- "1313:1313"

volumes:

- .:/app

environment:

- FLASK\_ENV=development

**Rodar o Docker Compose**:  
bash  
  
docker-compose up

docker-compose down

docker-compose build

Texto

O conteúdo gerado por IA pode estar incorreto.

### **4. Testar a API**

1. **Verificar se a API está funcionando**:
   * Abra o navegador e vá para http://localhost:1313/ para verificar se a API está ativa.
2. **Consultar itens disponíveis**:
   * Vá para http://localhost:1313/items para ver a lista de itens.

Interface gráfica do usuário, Aplicativo, chat ou mensagem de texto

O conteúdo gerado por IA pode estar incorreto.

### **5. Adicionar Swagger para Documentação**

**Objetivo**: Documentar e testar a API via Swagger UI.

**Crie o arquivo swagger.json no diretório static/**:  
json  
  
{

"swagger": "2.0",

"info": {

"title": "InovaTech API",

"version": "1.0",

"description": "Documentação da API do laboratório de criação de APIs"

},

"host": "localhost:1313",

"schemes": [

"http"

],

"paths": {

"/": {

"get": {

"summary": "Endpoint principal",

"description": "Verifica o status da API",

"responses": {

"200": {

"description": "Mensagem de status",

"schema": {

"type": "string"

}

}

}

}

},

"/items": {

"get": {

"summary": "Obter lista de itens",

"description": "Retorna uma lista de itens disponíveis.",

"responses": {

"200": {

"description": "Lista de itens",

"schema": {

"type": "array",

"items": {

"type": "string"

}

}

}

}

}

},

"/login": {

"post": {

"summary": "Login para gerar token JWT",

"description": "Gera um token JWT para autenticação.",

"responses": {

"200": {

"description": "Token de autenticação",

"schema": {

"type": "object",

"properties": {

"access\_token": {

"type": "string"

}

}

}

}

}

}

},

"/protected": {

"get": {

"summary": "Rota protegida",

"description": "Acessa uma rota protegida. Requer token JWT.",

"security": [

{

"Bearer": []

}

],

"responses": {

"200": {

"description": "Acesso permitido à rota protegida",

"schema": {

"type": "string"

}

},

"401": {

"description": "Autenticação falhou ou token ausente"

}

}

}

}

},

"securityDefinitions": {

"Bearer": {

"type": "apiKey",

"name": "Authorization",

"in": "header",

"description": "Token JWT a ser informado no formato: Bearer {token}"

}

}

}

1. **Acesse a documentação**:
   * Abra http://localhost:1313/swagger no navegador para visualizar a documentação interativa.

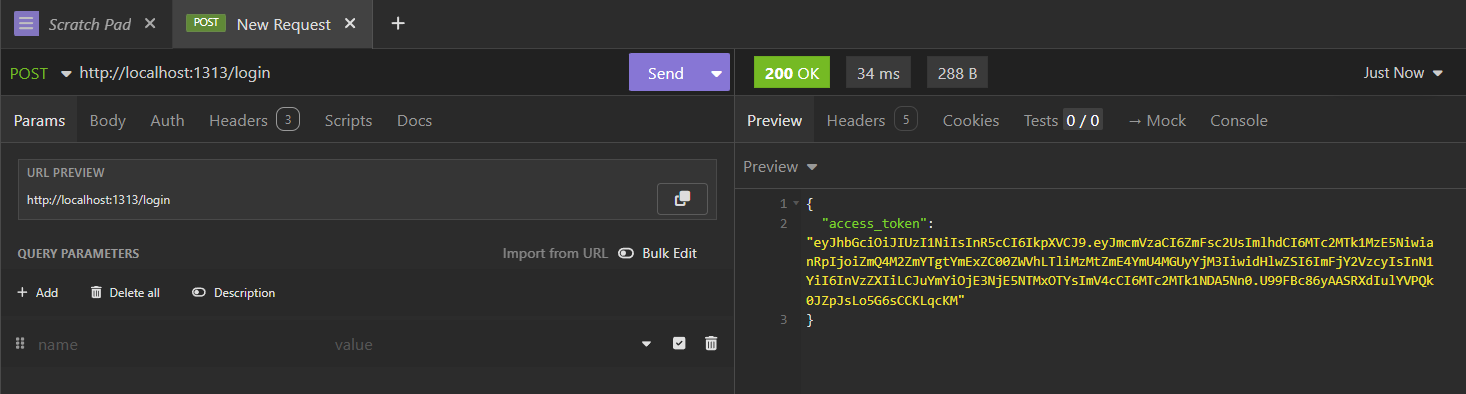
Interface gráfica do usuário, Aplicativo

O conteúdo gerado por IA pode estar incorreto.

### **6. Usar autenticação via JWT**

**Objetivo**: Proteger endpoints com autenticação JWT.

1. **Obtenha o token JWT via /login**:
   * Faça uma requisição POST para http://localhost:1313/login e obtenha o token no campo access\_token.



1. **Acesse a rota protegida /protected**:
   * Envie uma requisição GET para http://localhost:1313/protected com o token no cabeçalho Authorization:  
     bash  
       
     Authorization: Bearer <seu\_token\_jwt>

Tela de celular com aplicativo aberto

O conteúdo gerado por IA pode estar incorreto.