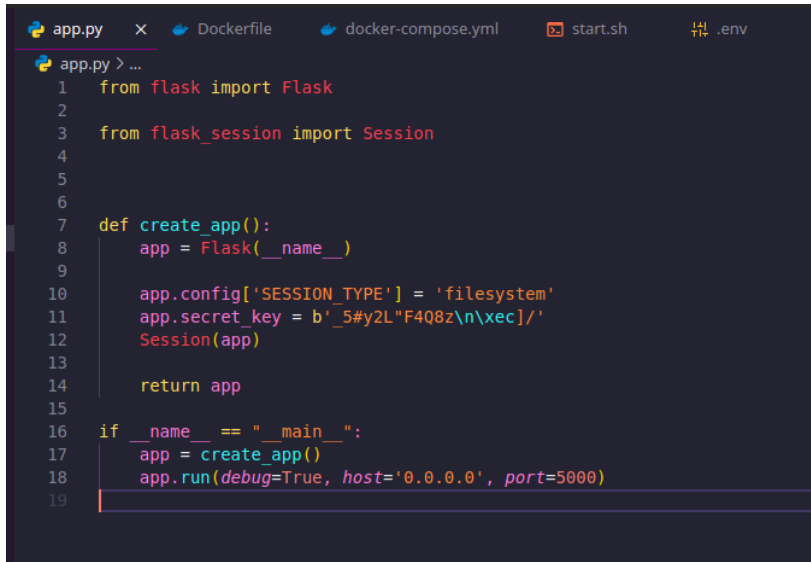


# Informações da estrutura do APP

- Primeiramente, eu crio a aplicação.

A screenshot of a code editor with a dark theme. The editor has several tabs at the top: 'app.py', 'Dockerfile', 'docker-compose.yml', 'start.sh', and '.env'. The 'app.py' tab is active, showing Python code for a Flask application. The code includes imports for Flask and Flask-Session, a function to create the app, and a main block to run the app on port 5000.

```
app.py > ...
1  from flask import Flask
2
3  from flask_session import Session
4
5
6
7  def create_app():
8      app = Flask(__name__)
9
10     app.config['SESSION_TYPE'] = 'filesystem'
11     app.secret_key = b'_5#y2L"F4Q8z\n\xec)/'
12     Session(app)
13
14     return app
15
16 if __name__ == "__main__":
17     app = create_app()
18     app.run(debug=True, host='0.0.0.0', port=5000)
19
```

- Após a configuração e os testes da aplicação, defino o serviço da aplicação no arquivo `docker-compose.yml`. Primeiro, atribuo o nome “WEB” e incluo um arquivo `.env`, que pode ser utilizado para carregar as variáveis necessárias. Também defino o nome do meu container.
- Em seguida, especifico o contexto de build, que será responsável por “montar” o container. O contexto é definido como o diretório atual, com o target “development” especificado no Dockerfile. Abaixo, faço algumas configurações para o volume e o diretório onde a aplicação será executada.

```

docker-compose.yml > {} services > {} web > [ ] ports
docker-compose.yml - The Compose specification establishes a standard
1  version: '3.8'
2  services:
3    web:
4      env_file:
5        - .env
6      restart: always
7      container_name: e-store
8      build:
9        context: .
10       target: development
11       working_dir: /app
12       volumes:
13         - ./app
14     ports:
15       - "5000:5000"
16     networks:
17       - flask_network
18     # command: ls -l
19 networks:
20   flask_network:
21
22

```

- No Dockerfile, estou definindo a versão do Python, juntamente com o target “development”. Após essa etapa, copio o arquivo de dependências do Flask e realizo a instalação dessas dependências.
- Concluído esse processo, é chamado o arquivo `start.sh`, que inicia nosso aplicativo. Em um contexto de aplicação real, esse script pode ser utilizado para executar comandos de migrações e outras tarefas de inicialização.

```

#!/bin/bash
flask db init
flask db migrate
flask db upgrade
exec gunicorn -b 0.0.0.0:5000 "app:create_app()"

```

- Por fim o App pode ser executado com o comando “docker compose up --build”