terça-feira, 1 de abril de 2025 07:32

- CppRestSDK is not an option anymore, no support for years and the github page recommend not to use it
   Pistache

- May lack documentation.... but it's something I can contribute... they are looking for voluntiers
  - Ceems like a good option, has support (github page s active) https://github.com/CrowCpp/Crow
    Looks like there are some documentation arround it https://crowcpp.org/master/getting\_started/setup/linux/
    GPT gave a solid answer... that is nice

### 2 Structuring the C++ API Project

A well-structured project makes it easier to maintain and expand. Below is the recommended folder structure:



#### 3. Running the API in Development Mode

Now, you can run your API without rebuilding the image every time.

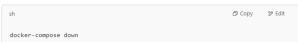
#### Start the API

sh	∜ Edit
docker-compose upbuild	

#### This will-

- Build the container (if needed).
- Mount your local source code inside the container.
- Compile the code ( cmake . && make ).
- Run the API.

#### Stop the API



#### **Recompile Without Restarting**

If you change your C++ code, you don't need to restart Docker. Instead, inside the running container, recompile:

```
docker exec -it cpp-api-dev bash # Open a shell inside the container
cmake . && make
                               # Recompile the API
./api_server
                                # Restart it
```

# 🔥 Summary: Fast Development Workflow

Action	Command
Start API (auto-recompiling)	docker-compose upbuild
Stop API	docker-compose down
Recompile code inside container	docker exec -it cpp-api-dev bash $\rightarrow$ cmake . && make && ./api_server
Restart API after code changes	./run_dev.sh

### 1 Build & Start Container

docker-compose up --build

#### 2 Manually Rebuild (if needed)

docker exec -it cpp-api-dev bash cd build && cmake .. && make -j\$(nproc) ./api\_server # Run manually

# 3 Stop Container

docker-compose down

## Definição

└─ .dockerignore

#### Brainstorm file will havelle both containers -DAML Implementation/ - docker/ - dev.Dockerfile # Image for development (build tools, live of └── prod.Dockerfile # Image for production (run only) # Multi-container setup (dev + prod) — docker-compose.yml - (Makelists tyt - src/ └─ main.cpp - build/ # Ignored in .gitignore, generated during bu

Just a draft

#### 1. Development Dockerfile (docker/dev.Dockerfile)

Dockerfile

Dockerfile
CopyEdit
FROM ubuntu:22.04
# Install build dependencies
RUN apt\_eet update && apt\_get install -y\
g++ cmake git libboost-all-dev libasio-dev\
&& rm -ff /xar/libpat/rlists/\*
# Install Crow
RUN git clone --branch v1.2.1.2 https://sithub.com/CrowCpp/Crow.git /Crow\
&& cd /Crow && mkdir build && cd build\
&& cmake ... -DBUILD\_SHARED\_UBS=ON && make -jS(nproc) && make install

of tooks on the motelation

# 3. Instalar GoogleTest e GoogleMock # 3. Instalar Google lest e GoogleMock RUN git clone https://github.com/google/googletest.git /tmn/poopletest && \

```
- src/
   └─ main.cpp
   build/
                               # Ignored in .gitignore, generated during bu

    dockerignore

L TEST L MY-TEST-CAP
```

```
yaml
                                                          🗗 Сору
                                                                  ⁰ Edit
     dockerfile: docker/dev.Dockerfile
    container_name: cpp-api-dev
   working_dir: /app
   ports:
      - "8080:8080"
     bash -c "mkdir -p build && cd build && cmake .. && make -j$(nproc) &&
     dockerfile: docker/prod.Dockerfile
    container_name: cpp-api-prod
      - "8080:8080"
    depends_on:
```

#### 2. Production Dockerfile (docker/prod.Dockerfile)

Dockerfile CopyEdit FROM ubuntu:22.04 FROM ubuntu:22.04
# Install runtime-only dependencies
RUN apt-get update && apt-get install -y \
libboost-all-dev libasio-dev \
&& rm -ff /var/lib/apt/lists/\*
COPY build/api\_server /app/api\_server
WORKDIR /app
EXPOSE 8080 CMD ["./api\_server"]

# 3. Instalar GoogleTest e GoogleMock RUN git clone nun gir cione
https://github.com/google/googletest.git
/tmp/googletest && \
cd /tmp/googletest && \
cmake - 5. - B build && \
cmake - -build build - j (Inproc) && \
cmake --install build && \
cmake --install build && \ rm -rf /tmp/googletest

יי שמשים --

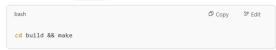
g++ cmake git libboost-all-lev libasio-dev\
&& rm -rf /var/lib/apt/lists/\*
# install Crow
RUN git clone -branch v1.2.1.2 https://github.com/CrowCpp/Crow.git /Crow\
&& cd /Crow && mkdir build && cd build\
&& cmake .- DBUILD\_SHARED\_LIBS=ON && make -j\$(nproc) && make install
WORKDIR/app

#### 4. Development Workflow

**Run Dev Environment:** docker-compose up --build ☐ Copy 🍪 Edit bash docker-compose up cpp-dev

#### Recompile after code change (automatically with volume):

- Change code in src/main.cpp
- Inside the container ( cpp-dev ), rerun:



docker-compose up --build

# 5. Running the API (Dev or Prod)

• From host (Windows or WSL), call:



#### Or test with Postman:

- POST URL: http://localhost:8080/api/analyze
- Body (JSON):

```
% Edit
 "code": "class A {}"
}
```



# Summary

Goal	Solution
Dev on Linux	Use cpp-dev container with volume mapping
Run API	Auto-run in dev or manually in cpp-api
CI/CD	GitHub Actions with docker build
Installer	Use cpack Or makeself after build

- Usar o tunder client ou postman?
  Pensar no versionamento das apis e como testar elas
  O Rodar uma batería de testes na API a cada versão
  Uma aplicação que comunca com uma API e faz avaliação das respostas eseradas
  Camada de segurança (não vou precoupar com isso agora
  Permito aquela origem
  Rcebe um endereço no github, baixa os arquivos e processa eles