Relatório de Entrega: Orquestração de Microsserviços com Docker

Aluno: José Marcos da Silva Lima
Professor: Thiago Ozores
Escola Avanti | Bootcamp Avanti | Trilha DevOps
Data: 28 de Fevereiro de 2026
Tecnologias: Docker, Docker Compose, Nginx (Alpine), Python 3.13 (Slim), Go 1.24 (Multi-stage).

---

1. Introdução

Este documento detalha a implementação da arquitetura de microsserviços conteinerizados. A solução foca em otimização de camadas, segurança através de imagens 'slim' e 'alpine', e o uso de **Multi-stage Builds** para garantir imagens de produção leves e eficientes.

---

2. Dockerfiles do Projeto

2.1. Frontend (Gerador de Saudações)

Utiliza Nginx Alpine para servir o arquivo estático da aplicação.

```
# --- Estágio 1: Definir a imagem base ---
FROM nginx:alpine

# --- Estágio 2: Copiar os arquivos do projeto ---
COPY index.html /usr/share/nginx/html/index.html

# --- Estágio 3: Expor a porta ---
EXPOSE 80
```

2.2. Microsserviço de Pessoas (API Python)

Implementação com foco em cache de camadas e separação de build/runtime

```
# --- Estágio 1: Builder ---
FROM python:3.13-slim AS builder
WORKDIR /app
RUN pip install --upgrade pip
COPY requirements.txt .
RUN pip wheel --no-cache-dir --wheel-dir /app/wheels -r
requirements.txt

# --- Estágio 2: Runtime ---
FROM python:3.13-slim
WORKDIR /app
COPY --from=builder /app/wheels /wheels
RUN pip install --no-cache /wheels/*
COPY . .
EXPOSE 8000
CMD ["python", "app.py"]
```

2.3. Microsserviço de Saudações (API Go)

Uso de Multi-stage Build completo com suporte a CGO e SQLite.

```
# --- Estágio de Build ---
FROM golang:1.24-alpine AS builder
RUN apk add --no-cache build-base gcc
WORKDIR /app
COPY go.mod go.sum ./
RUN go mod download
COPY . .
RUN CGO_ENABLED=1 GOOS=linux go build -a -installsuffix cgo -o
/app/main .

# --- Estágio Final ---
FROM alpine:latest
WORKDIR /app
COPY --from=builder /app/main .
```

```
EXPOSE 8080
CMD ["./main"]
```

3. Orquestração (Docker Compose)

Configuração integrando os serviços em uma rede bridge isolada.

```
version: '3.8'

services:
  site:
    image: jlimacloud/gerador-saudacoes:1.0
    container_name: site-gerador
    ports:
      - "80:80"
    depends_on:
      - ms-pessoas-aleatorias
      - ms-saudacoes-aleatorias
    networks:
      - backend

  ms-pessoas-aleatorias:
    image: jlimacloud/ms-pessoas-aleatorias:1.0
    container_name: api-pessoas
    ports:
      - "8000:8000"
    networks:
      - backend

  ms-saudacoes-aleatorias:
    image: jlimacloud/ms-saudacoes-aleatorias:1.0
    container_name: api-saudacoes
    ports:
      - "8080:8080"
    networks:
      - backend

networks:
  backend: {}


---

## 4. Evidências de Validação

### 4.1. Status dos Containers (docker ps)
```

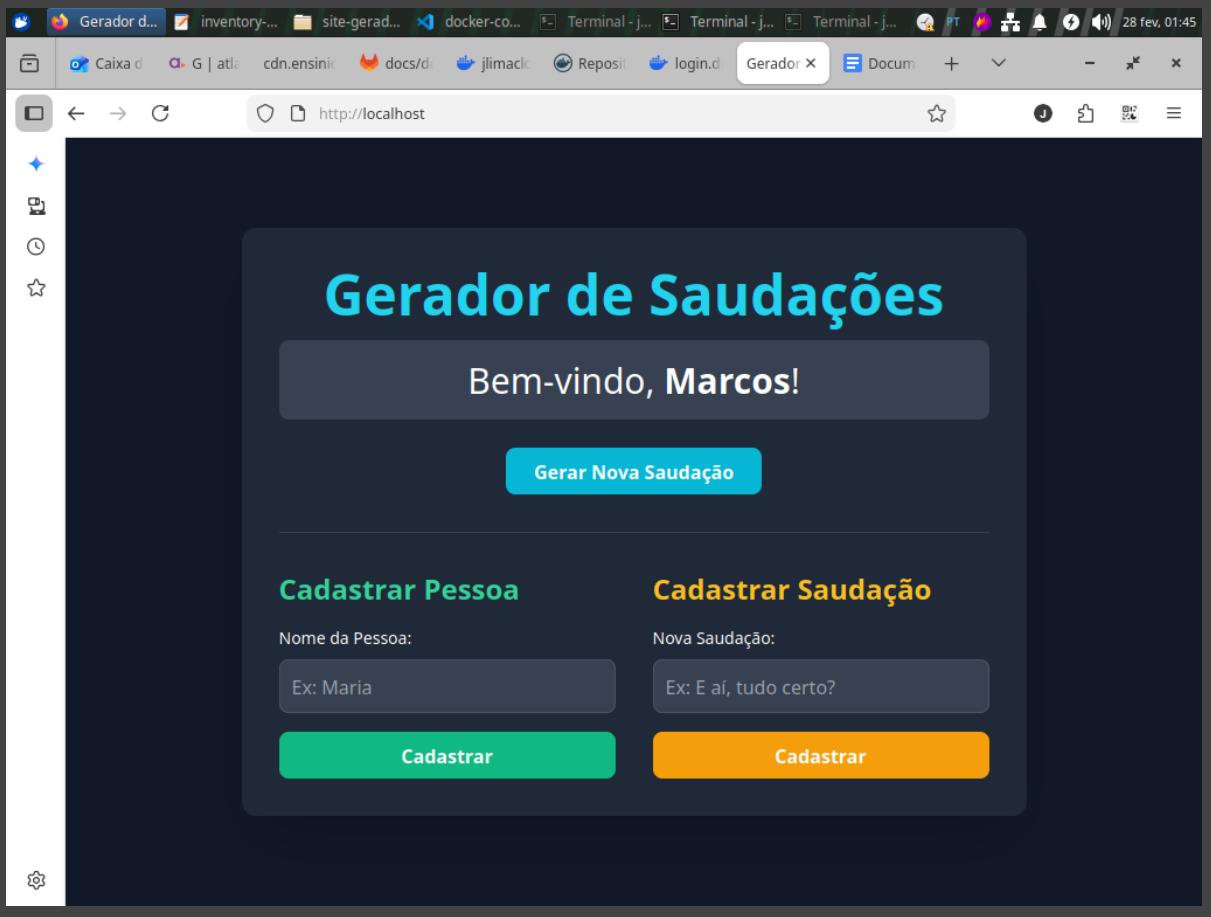
```
18    networks:
19      - backend
20
      ▷Run Service
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS        bash - desafio-docker

jlima@jlima-System:~/desafio-docker$ docker ps
CONTAINER ID   IMAGE                                  COMMAND                  CREATED
  STATUS          PORTS                                          NAMES
73b81d1e6582   jlimacloud/gerador-saudacoes:1.0        "/docker-entrypoint.…"   55 minutes ago
  Up 55 minutes   0.0.0.0:80->80/tcp, [::]:80->80/tcp            desafio-docker-site-1
429bd77b187b   jlimacloud/ms-pessoas-aleatorias:1.0    "uvicorn main:app --…"   55 minutes ago
  Up 55 minutes   0.0.0.0:8000->8000/tcp, [::]:8000->8000/tcp   desafio-docker-ms-pessoas-aleato
rias-1
a0c056490a45   jlimacloud/ms-saudacoes-aleatorias:1.0   "./main"                 55 minutes ago
  Up 55 minutes   0.0.0.0:8080->8080/tcp, [::]:8080->8080/tcp   desafio-docker-ms-saudacoes-alea
torias-1
jlima@jlima-System:~/desafio-docker$
```

```
main*  0 0  -- VISUAL --   Ln 20, Col 1 (32 selected)   Spaces: 2   UTF-8   LF   { } Compose   Go Live
```

4.2. Interface da Aplicação (Browser)



---

5. Conclusão

O projeto demonstra o domínio de conceitos fundamentais de DevSecOps:
- **Imagens Otimizadas:** Redução de tamanho via Alpine e Multi-stage.

- **Orquestração Declarativa:** Deploy simplificado com apenas um comando.
- **Segurança de Rede:** Serviços isolados em rede interna do Docker.