# SeisLabData

Documentação de desenvolvimento

*Equipa NaturalGIS*

# Índice

# 1. Desenvolvimento

Este projeto é composto por múltiplos serviços, que são orquestrados com `docker compose`. O ficheiro `docker/compose.dev.yaml` contem as instruções adequadas para desenvolvimento.

> **Dica**
>
> Quando a *stack* de desenvolvimento do Docker estiver ativa e em execução, execute os comandos docker compose com esta incantação:
>
> ```
> docker compose -f docker/compose.dev.yaml <docker-command> <service-name>
> ```
>
> Isto facilita o ajuste dos comandos ao âmbito deste projeto.

Os serviços mais relevantes são:

- `webapp` – a aplicação web principal, implementada com starlette, sqlmodel, jinja e datastar.
- `processing-worker` – serviço executa a maior parte do processamento. É um *worker* dramatiq.
- `message-broker` – uma instância redis que gere a passagem de mensagens entre a webapp e o processing worker.
- `web-gateway` – uma instância traefik que atua como *reverse proxy* para o sistema.
- `auth-webapp` – uma instância authentik que trata da autenticação de utilizadores.
- `caddy-file-server` – uma instância caddy que serve datasets locais via HTTP.

## 1.1 Configuração do ambiente

Comece por obter os datasets de exemplo que foram disponibilizados pelo cliente. Estes encontram-se no ficheiro `ipma/2025-marine-data-catalog/sample-data/20251125_datasample01_restored_data.tar.gz`, que está disponível na plataforma interna de base de conhecimento.

Crie uma diretoria base para os datasets do projeto (por exemplo em `~/data/seis-lab-data`), obtenha o arquivo e extraia-o dentro desta diretoria:

```
mkdir -p ~/data/seis-lab-data
cd ~/data/seis-lab-data

# obtenha o arquivo tar para esta dir e extraia-o
tar -xvf 20251125_datasample01_restored_data.tar.gz

# remova o arquivo após a extração
rm 20251125_datasample01_restored_data.tar.gz
```

Deverá obter algo semelhante a isto (listagem abreviada):

```
ricardo@tygra:~/data/seis-lab-data/$ tree -L 4
.
└── prr_eolicas
    └── base-final
        └── surveys
            └── owf-2025
```

Agora, clone este repositório localmente:

```
cd ~/dev  # ou onde preferir guardar o código

git clone [https://github.com/NaturalGIS/seis-lab-data.git](https://github.com/NaturalGIS/seis-lab-data.git)
cd seis-lab-data
```

Para simplificar a montagem da diretoria de dados dentro dos serviços docker, o projeto assume que existe uma diretoria `sample-data` na raiz do repositório. Como tal, crie um link simbólico (symlink) a apontar para a diretoria de dados que criou acima:

```
# assumindo que a sua diretoria sample-data está em `~/data/seis-lab-data`
ln -s ~/data/seis-lab-data sample-data
```

Certifique-se de que tem o docker e o uv instalados na sua máquina. Utilize o uv para instalar o projeto localmente:

```
uv sync --group dev --locked
```

Instale os hooks de pre-commit incluídos:

```
uv run pre-commit install
```

Efetue o *pull* das imagens docker do projeto dos respetivos registos (poderá precisar de fazer login em ghcr.io):

```
docker compose -f docker/compose.dev.yaml pull
```

De seguida, lance a stack:

```
docker compose -f docker/compose.dev.yaml up -d
```

Deverá agora conseguir aceder à webapp em http://localhost:8888. Continue para a secção de arranque de uma instalação nova.

Pode fazer login no sistema usando as credenciais:

- utilizador: `akadmin@email.com`
- password: `admin123`

## 1.2 Arranque de uma instalação nova

O processo de arranque (*bootstrapping*) consiste em :

- Criar/atualizar a base de dados,
- Carregar as variáveis predefinidas
- Opcionalmente, adicionar dados de exemplo.

O *bootstrapping* é feito utilizando a aplicação de linha de comandos (CLI) `seis-lab-data`, que está disponível no serviço docker compose chamado `webapp` :

```
docker compose -f docker/compose.dev.yaml exec -ti webapp uv run seis-lab-data --help
```

Execute os seguintes comandos:

```
# inicializar a BD
docker compose -f docker/compose.dev.yaml exec -ti webapp uv run seis-lab-data db upgrade

# adicionar dados predefinidos
docker compose -f docker/compose.dev.yaml exec -ti webapp uv run seis-lab-data bootstrap all

# opcionalmente, carregar registos de exemplo
docker compose -f docker/compose.dev.yaml exec -ti webapp uv run seis-lab-data dev load-all-samples

# opcionalmente, gerar registos sintéticos (útil para a UI)
docker compose -f docker/compose.dev.yaml exec -ti webapp uv run seis-lab-data dev generate-many-projects --num-projects=50
```

## 1.3 Notas adicionais

A imagem docker de desenvolvimento usa a tag `latest` e é reconstruída em cada commit no ramo chamado `main` do repositório. Assim sendo, sempre que houver modificações no código, antes de iniciar a stack de serviços, é aconselhável puxar a versão mais recente da imagem docker:

```
docker compose -f docker/compose.dev.yaml pull webapp
docker compose -f docker/compose.dev.yaml up -d
```

**Criar a imagem docker localmente**

Se adicionar uma nova dependência Python, deverá criar a imagem localmente:

```
docker build \
  --tag ghcr.io/naturalgis/seis-lab-data/seis-lab-data:$(git branch --show-current) \
  --file docker/Dockerfile \
  .
```

Depois, reinicie a stack:

```
CURRENT_GIT_BRANCH=$(git branch --show-current) docker compose -f docker/compose.dev.yaml up -d --force-recreate
```

**Traduções**

Como a diretoria `src` é montada via usando um bind mount, ficheiros `*.mo` da imagem são mascarados pelos que estiverem presentes no disco local. Como tal, de modo a que as traduções funcionem corretamente, será necessário correr o commando:

```
seis-lab-data translations compile
```

Em seguida, reinicie o serviço `webapp`.

# 2. Execução de testes

Os testes normais correm dentro do contentor webapp:

```
docker compose --file docker/compose.dev.yaml exec webapp uv sync --locked --group gdal --group dev
docker compose --file docker/compose.dev.yaml exec webapp uv run pytest
```

Testes de integração:

```
docker compose --file docker/compose.dev.yaml exec webapp uv run pytest -m integration
```

## 2.1 Testes end-to-end (E2E)

A execução dos testes E2E utiliza o playwright. Como tal, a sua execução requer a instalação de dependências adicionais:

```
uv run playwright install --with-deps chromium
```

Os testes podem ser executados com o comando:

```
uv run pytest tests/e2e/ \
    -m e2e \
    --confcutdir tests/e2e \
    --user-email akadmin@email.com \
    --user-password admin123 \
    --base-url http://localhost:8888
```

**Dica**

Para correr os tests E2E no modo *headed* (*i.e.* com execução de uma interface gráfica), adicione os parâmetros `--headed` e `--slowmo 1500`.

```
uv run pytest tests/e2e/ \
    -m e2e \
    --confcutdir tests/e2e \
    --user-email akadmin@email.com \
    --user-password admin123 \
    --base-url http://localhost:8888 \
    --headed \
    --slowmo 1500
```