Projeto Gestão de Horários Escolares — Parte 2

Aluno: Igor Carlos Santos Cruz (A043765)
Grupo: 10
Unidade Curricular: Desenvolvimento Web I
Parte: 2 — API REST (Design-First & Code Generation)

---

1. Introdução

Esta fase do projeto consistiu no desenvolvimento de uma **API REST** seguindo a abordagem **Design-First**. O processo iniciou-se com a escrita manual da especificação **OpenAPI 3.0 (Swagger)**, que serviu de base para a **geração automática** da estrutura do servidor.

Este fluxo garante que a implementação final respeita rigorosamente os contratos definidos na fase de planeamento.

O sistema permite a gestão administrativa de:

- **Alunos**
- **Professores**
- **Salas**
- **Aulas**
- **Horários**

Foi implementada uma relação obrigatória **1:N**, onde:

- **Um Professor pode estar associado a vários Horários**

---

2. Arquitetura da Solução

A solução utiliza uma **arquitetura multi-container**, orquestrada através do **Docker**, garantindo isolamento entre a API e a Base de Dados.

- **Design:** Especificação manual em YAML (OpenAPI 3.0)
- **Servidor:** Node.js com Express (gerado automaticamente)
- **Base de Dados:** MySQL persistente
- **Orquestração:** Docker Compose

---

3. Tecnologias Utilizadas

Tecnologia	Função
OpenAPI 3.0	Definição Design-First da API
OpenAPI Generator	Geração automática do servidor
Node.js / Express	Backend da API

Tecnologia	Função
MySQL	Sistema de Base de Dados Relacional
Docker	Containerização e ambiente multi-container

4. Estrutura do Projeto

A organização do projeto segue a estrutura gerada pelo **OpenAPI Generator** para Node.js/Express, com adaptações para integração da base de dados e execução em ambiente Docker.

```
PARTE2-M2/
│
├── express-server/
│   ├── .openapi-generator/
│   ├── controllers/
│   ├── services/
│   ├── utils/
│   ├── uploaded_files/
│   ├── node_modules/
│   │
│   ├── .eslintignore
│   ├── .eslintrc.json
│   ├── .openapi-generator-ignore
│   ├── combined.log
│   ├── config.js
│   ├── expressServer.js
│   ├── index.js
│   ├── logger.js
│   ├── Dockerfile
│   ├── docker-compose.yml
│   ├── package.json
│   ├── package-lock.json
│   └── README.md
│
├── mysql-db/
│   ├── Dockerfile
│   └── school.sql
│
├── openapi/
├── openapitools.json
├── .gitignore
└── package-lock.json
```

5. Base de Dados

A base de dados **school** foi concebida para suportar todas as funcionalidades da aplicação de gestão de horários escolares.

Foram definidas entidades para:

- Alunos
- Professores
- Salas
- Aulas
- Horários

Volume de Dados

Cada tabela contém aproximadamente **30 registos**, garantindo massa de dados suficiente para:

- validação das cardinalidades
- testes funcionais

Relações e Integridade

- Implementação de uma relação **1:N** entre **Professores** e **Horários**
- Utilização de **Chaves Estrangeiras (Foreign Keys)** para garantir integridade referencial entre:
  - Professores e Horários
  - Salas e Aulas
  - Aulas e Horários

⚙️ 6. Processo de Geração (Workflow)

O desenvolvimento da API seguiu a abordagem **Design-First**, iniciando-se pela definição manual do contrato da API.

Etapas do Processo

1. Escrita manual do ficheiro `openapi.yaml` no formato **OpenAPI 3.0**
2. Validação da especificação no **Swagger Editor**
3. Geração automática da estrutura do servidor Node.js/Express
4. Adaptação manual do código gerado
5. Integração da base de dados MySQL

Comando de Geração

```
npx openapi-generator-cli generate -i openapi.yaml -g nodejs-express-server -o
./express-server
```

Após a geração:

- Foram ajustadas as rotas e controladores
- Foi criada a ligação à base de dados através do ficheiro mysql-db
- O código foi organizado de acordo com os requisitos do enunciado

7. Endpoints Principais

A API disponibiliza um conjunto de endpoints REST para a gestão completa das entidades do sistema.

Recurso	Métodos HTTP	Descrição
`/alunos`	GET, POST, PUT, DELETE	Gestão de alunos
`/professores`	GET, POST, PUT, DELETE	Gestão de professores
`/salas`	GET, POST, PUT, DELETE	Gestão de salas
`/aulas`	GET, POST, PUT, DELETE	Gestão de aulas
`/horarios`	GET, POST, PUT, DELETE	Gestão de horários

8. Execução com Docker

A aplicação foi containerizada utilizando **Docker**, permitindo a execução do sistema em ambiente isolado e reproduzível.

Imagens Docker

As imagens encontram-se publicadas no **Docker Hub**, com a tag:

- `api-horarios:m2`
- `mysql-db:m2`

Inicialização do Sistema

Para colocar o sistema em funcionamento, deve ser executado o seguinte comando na raiz do projeto:

```
cd express-server
docker compose up -d
```

9. Utilização de Inteligência Artificial

Foi utilizada uma ferramenta de Inteligência Artificial como apoio ao desenvolvimento do projeto, respeitando os princípios de transparência académica.

Âmbito da Utilização

A IA foi utilizada exclusivamente como suporte técnico e conceptual, nomeadamente para:

- esclarecimento de conceitos
- validação de sintaxe
- apoio na estruturação inicial de código

Integração no Projeto

Todo o código gerado foi:

- analisado

- compreendido
- adaptado manualmente
- integrado pelo aluno

Tarefas com Apoio de IA

- Geração de massa de dados SQL (30 registos por tabela)
- Apoio na definição do schema.sql
- Validação da especificação OpenAPI (YAML)
- Auxílio na configuração dos ficheiros Dockerfile e docker-compose.yml

10. Conclusão

Este projeto demonstra a aplicação prática de metodologias modernas de desenvolvimento de software, nomeadamente a abordagem Design-First aliada à Geração Automática de Código.

A utilização do OpenAPI Generator permitiu manter consistência entre o design e a implementação, enquanto o Docker garantiu um ambiente controlado e replicável.

O resultado final é uma API REST funcional, escalável, devidamente documentada e preparada para execução em ambiente multi-container.