Desenvolvimento de uma API REST com Node.js, MySQL e Docker

Trabalho – Parte 2 (Biblioteca)

Rodrigo Vieira nº048063

João Mendes nº047275

Introdução

Este trabalho teve como objetivo desenvolver uma API REST própria utilizando Node.js (com o framework Express) e MySQL, executados em ambiente Docker multi-container. A API representa um sistema simples de Biblioteca, permitindo gerir autores, editoras, categorias e livros, com funcionalidades completas de CRUD e documentação em OpenAPI 3.0. Foram também realizados testes funcionais utilizando o Postman.

Tecnologias Utilizadas

• Node.js + Express – Implementação do servidor e das rotas da API REST.

• MySQL – Persistência e modelação dos dados.

• Docker Compose – Orquestração dos containers (API + base de dados).

• Postman – Testes e verificação dos endpoints.

• OpenAPI 3.0 – Documentação da API através do ficheiro openapi.yaml.

Modelo de Dados (MySQL)

A base de dados foi estruturada com os seguintes recursos principais: authors, publishers, categories e books.

Relações: 1 Autor → N Livros (books.author_id); 1 Editora → N Livros (books.publisher_id). Estas relações caracterizam um modelo 1:n.

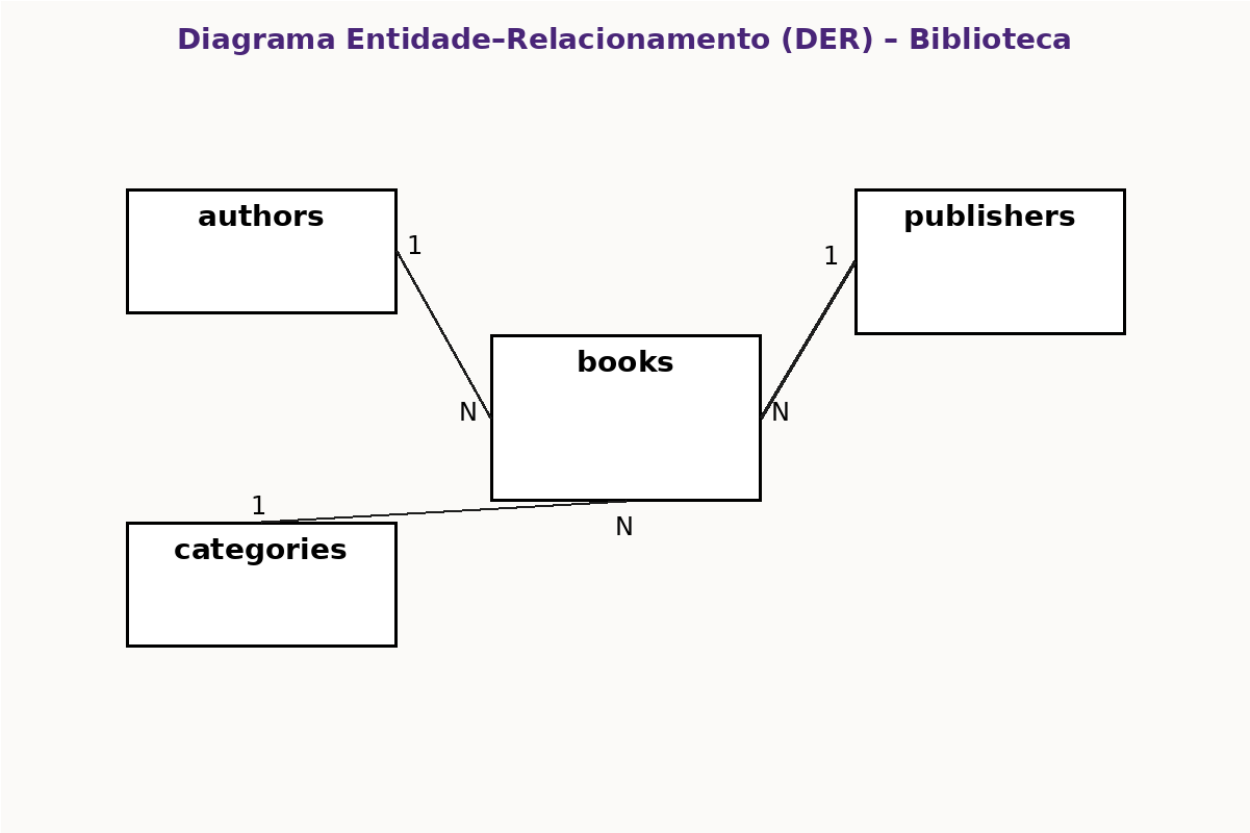

Figura 1 – Diagrama Entidade–Relacionamento (DER) do sistema Biblioteca.

Execução com Docker

A aplicação é composta por dois containers: db (MySQL) e api (Node.js/Express).

Comando utilizado para construir os serviços:

docker compose up –build

Endpoints da API (CRUD)

Base URL: http://localhost:3000

Authors:

GET /api/authors

GET /api/authors/{id}

POST /api/authors

PUT /api/authors/{id}

DELETE /api/authors/{id}

Publishers:

GET /api/publishers

GET /api/publishers/{id}

POST /api/publishers

PUT /api/publishers/{id}

DELETE /api/publishers/{id}

Categories:

GET /api/categories

GET /api/categories/{id}

POST /api/categories

PUT /api/categories/{id}

DELETE /api/categories/{id}

Books:

GET /api/books

GET /api/books/{id}

POST /api/books

PUT /api/books/{id}

DELETE /api/books/{id}

Testes no Postman

Os testes realizados no Postman permitiram validar: (i) o funcionamento completo do CRUD para todos os recursos; os filtros de pesquisa (especialmente em livros); e o tratamento apropriado de erros.

A coleção utilizada foi exportada para o ficheiro:
Parte2_Biblioteca.postman_collection.json.

Tratamento de Erros e Validações

A API implementa mecanismos de validação e integridade, por exemplo: 409 Conflict ao tentar criar uma categoria repetida (campo name UNIQUE) e impedimento de excluir autores com livros associados (restrições de integridade referencial no MySQL).

Documentação OpenAPI

A documentação foi elaborada com a especificação OpenAPI 3.0 no ficheiro openapi.yaml, descrevendo endpoints, parâmetros, modelos de dados e respostas.

Conclusão

A solução desenvolvida cumpre os requisitos, disponibilizando uma API REST funcional com persistência em MySQL, execução em Docker multi-container, documentação em OpenAPI e testes no Postman. O resultado é um sistema modular, escalável e documentado.