

Trabalho Prático #1

Consumo e Implementação de APIs RESTful

Objetivo Geral



Consolidar os conhecimentos em desenvolvimento web com foco na criação, consumo e implementação de APIs RESTful utilizando tecnologias do ecossistema JavaScript:

- Node.js + Express
- MongoDB / MongoDB Atlas
- JSON-Server
- Fetch API
- Swagger (opcional)


O projeto simula o ciclo completo de desenvolvimento de uma aplicação web com front-end e back-end separados, incluindo testes e deploy.


Partes do Trabalho

Parte 1: Estruturação da Base de Dados (JSON)



- Criar um ficheiro `bd.json` com:
 - Lista de alunos: `nome`, `apelido`, `curso`, `anoCurricular`
 - Lista de cursos: `nomeDoCurso`
 -  Diretório sugerido: `/mock-data/`
 -  Entregável: `bd.json`
-

Parte 2: API Simulada com JSON-Server + Testes



- Configurar e iniciar `json-server` com `bd.json`
- Testar os endpoints com Postman (CRUD de alunos, leitura de cursos)
- Exportar a coleção de testes
-  Diretório sugerido: `/mock-server/`

-  Entregáveis:
 - Código de configuração (`package.json` , script json-server)
 - Coleção `.json` do Postman em `/tests/`
-

Parte 3: Interface Web (CRUD de Alunos)

- Desenvolver uma página web funcional para gerir alunos:
 - Ver alunos
 - Adicionar aluno
 - Editar aluno
 - Apagar aluno
 - Utilizar `Fetch API` e programação assíncrona
 -  Diretório sugerido: `/frontend/`
 -  Entregável: Página funcional conectada à API simulada
-


Parte 4: API RESTful real (Node.js + Express + MongoDB Atlas)

- Migrar os dados para o MongoDB Atlas
 - Implementar a API Express com endpoints equivalentes ao JSON-server
 - Manter a estrutura RESTful
 - Sugestão : usar mongoose a abordagem MVC (bónus 5%)
 -  Diretório sugerido: `/backend/`
 -  Entregável: Código funcional da API com instruções
-

Parte 5: Deploy da Aplicação

- Fazer deploy do front-end no [Vercel](#)
- (Opcional) Fazer deploy da API no [Render](#)
- Adaptar o front-end para consumir a nova API

 Incluir no `README.md` :

- URL pública do front-end
 - URL da API real
 -  Entregável: Links funcionais no repositório
-

Parte 6 (Bonificação): Documentação da API

- Utilizar Swagger para documentar os endpoints da API
- Incluir rota `/api-docs` na aplicação
- 📁 Diretório sugerido: `/backend/docs/`
- 📄 Entregável: Swagger funcional e acessível

Organização do Projeto

```
projeto-raiz/  
├── /frontend/ ← Interface web (HTML/CSS/JS)  
├── /backend/ ← API RESTful com Node.js + MongoDB  
├── /mock-server/ ← JSON-server configurado  
├── /mock-data/ ← Base de dados JSON original  
├── /tests/ ← Coleção de testes Postman  
├── README.md ← Instruções, links e notas  
└── .gitignore, etc.
```

Sugestão de Branches

Branch	Descrição
<code>main</code>	Versão estável e final
<code>dev</code>	Desenvolvimento geral
<code>frontend</code>	Interface e interação do usuário
<code>api</code>	API real (Node + MongoDB)
<code>deploy</code>	Adaptações para Vercel/Render

Critérios de Avaliação

Critério	Peso
Base de dados JSON correta	10%
API simulada e testada (Postman)	10%
Funcionalidade do front-end	30%
Qualidade da API real (Node.js)	30%
Integração front-end/backend	10%
Deploy funcional	10%
Bonificação (MVC)	+5%
Bonificação (Swagger)	+5%



Entrega

- Entrega via **GitHub Classroom**.
- O repositório deve conter:
 - Código funcional
 - README.md com instruções claras
 - Links de deploy (front e opcionalmente back)



Repositório Base

Usa o repositório template inicial fornecido no GitHub Classroom.

TWT1RESTAPI