Relatório – Trabalho 3: Web Services (API REST)

O objetivo deste trabalho foi reimplementar o serviço remoto criado no Trabalho 2 utilizando o paradigma de **API REST**, eliminando o uso de sockets ou RMI. O sistema cliente-servidor foi implementado com um servidor Java (Spring Boot) e clientes em diferentes linguagens (JavaScript e Python), conforme exigido pelo enunciado.

Estrutura do Projeto

```
Final/
├── servidor-biblioteca/          # Back-end com Spring Boot (Java)
│   ├── model/Item.java
│   ├── controller/ItemController.java
│   ├── repository/ItemRepository.java
│   └── ServidorBibliotecaApplication.java
│
├── cliente-js/                   # Cliente web com HTML, JS e CSS
│   ├── index.html
│   └── script.js
│
├── cliente-python/               # Cliente CLI com Python
│   └── cliente.py
```

1. Servidor – Spring Boot

Funcionalidade

O servidor expõe uma **API REST** acessível via HTTP com as seguintes rotas:

Método	Endpoint	Função
GET	`/itens`	Lista todos os itens
GET	`/itens/{id}`	Busca item por ID
POST	`/itens`	Cria um novo item
PUT	`/itens/{id}`	Atualiza um item

DELETE	`/itens/{id}`	Remove um item

Modelo de dados (Item.java)

```
@Entity
public class Item {
    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String titulo;
    private String autor;
    private String tipo;
}
```

Banco de Dados

O projeto utiliza um banco **H2 em memória**, que é iniciado automaticamente com o Spring Boot. Os dados são perdidos ao encerrar o servidor, mas permitem testes rápidos.

2. Cliente Web (JavaScript)

Tecnologias

- HTML + CSS
- JavaScript (Fetch API)
- SweetAlert2 para notificações

Funcionalidade

O formulário da interface permite:

- Cadastrar itens com título, autor e tipo (livro, revista, publicação)
- Listar itens em cards coloridos
- Editar itens com popup SweetAlert
- Excluir itens com confirmação

Como executar

1. Abra o arquivo `index.html` com um navegador.
2. Certifique-se de que o servidor Java está rodando (`localhost:8080`).
3. A interface se conectará automaticamente à API `/itens`.

3. Cliente Python (linha de comando)

Como executar

```
cd cliente-python
python cliente.py
```

Testes em rede

Foi possível compartilhar a API para outro computador da mesma rede utilizando o IP local:

```
const API_URL = "http://172.25.212.17:8080/itens";
```

O colega pôde acessar o front-end remoto e interagir com o servidor.

Conclusão

Este trabalho demonstrou a capacidade de construir um sistema distribuído real utilizando **tecnologias modernas e interoperáveis**. O servidor em Java com Spring Boot foi integrado com clientes web e em Python, facilitando testes, reutilização e integração futura com qualquer sistema que entenda HTTP + JSON.