Relatório - Trabalho 3: Web Services (WS) ou API

Disciplina: Sistemas Distribuídos
Professor: Rafael Braga
Curso: Engenharia de Software
Alunos: Marcelo Mikael e Luis Felippe

---

1. Objetivo

O objetivo deste trabalho é reimplementar o serviço remoto criado no Trabalho 2, que utilizava RMI (Java Remote Method Invocation), utilizando agora um modelo baseado em Web Services (WS) ou API HTTP REST. A comunicação deve seguir o modelo cliente-servidor via requisições HTTP, sem uso de sockets ou RMI.

---

2. Descrição da Solução Implementada

2.1. Serviço (Servidor)

O serviço foi implementado como uma API REST utilizando Java com o framework **Spring Boot**, mantendo a lógica de venda de imóveis do Trabalho 2. Os dados foram armazenados em **memória**, como coleções Java (`Map` e `List`), para manter a simplicidade e foco na comunicação HTTP.

Funcionalidades da API:

- Cadastrar imóveis (residenciais ou rurais);
- Vender imóveis;
- Listar imóveis disponíveis para venda;
- Listar imóveis vendidos.

Estrutura de rotas:

Método HTTP	Endpoint	Descrição
POST	`/residencial/imoveis`	Cadastrar imóvel residencial
POST	`/residencial/imoveis/{id}/vender`	Vender imóvel residencial
GET	`/residencial/imoveis`	Listar imóveis disponíveis

GET	`/residencial/vendas`	Listar imóveis vendidos
POST	`/rural/imoveis`	Cadastrar imóvel rural
POST	`/rural/imoveis/{id}/vender`	Vender imóvel rural
GET	`/rural/imoveis`	Listar imóveis rurais disponíveis
GET	`/rural/vendas`	Listar vendas rurais

2.2. Clientes

Cada aluno foi responsável por implementar um cliente utilizando linguagens diferentes de Java, conforme exigido no enunciado.

Cliente 1: Node.js (JavaScript)

- Implementado com o módulo `axios`.
- Consumiu a API `/residencial` para cadastrar, listar e vender imóveis residenciais.
- Testado com sucesso em ambiente local.

Cliente 2: Python

- Desenvolvido com a biblioteca `requests`.
- Consumiu a API `/rural` para cadastrar, listar e vender imóveis rurais.
- Também testado localmente e apresentou funcionamento correto.

---

3. Considerações Finais

A reimplementação do serviço em formato REST demonstrou a versatilidade da arquitetura cliente-servidor baseada em HTTP. O uso do Spring Boot simplificou a criação de rotas e manipulação de dados em memória, enquanto os clientes em Node.js e Python permitiram testar a interoperabilidade do sistema de forma clara.

O trabalho cumpre todos os requisitos propostos:

- Utiliza API HTTP no lugar de RMI;

- Apresenta comunicação entre cliente e servidor;
- Utiliza duas linguagens distintas nos clientes;
- Inclui código funcional, bem como este relatório descritivo.

---

4. Repositório

Link para o repositório contendo o código fonte do servidor e dos clientes:

https://github.com/MarceloMikael/Sistemas-Distribuidos/tree/main/Trabalho-3-SistemasDistribuidos

---

5. Vídeo disponível em:

https://youtu.be/HMiXAl2-E4g