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

Disciplina: Sistemas Distribuídos

Professor: Rafael Braga

Curso: Engenharia de Software

Alunos: Renan Victor & Maely Vitória

1. Objetivo

O objetivo deste trabalho é reimplementar o serviço remoto criado no Trabalho 2 (anteriormente baseado em RMI em Java), utilizando agora uma arquitetura baseada em Web Services (WS), mais especificamente uma API REST desenvolvida com FastAPI em Python. A comunicação entre cliente e servidor deve seguir o modelo de requisições e respostas HTTP, sem o uso de sockets ou RMI.

2. Descrição da Solução Implementada

2.1. Serviço (Servidor)

O serviço foi reimplementado como uma API REST utilizando Python com FastAPI, simulando um sistema de controle de produtos veterinários. A lógica da aplicação foi mantida em memória e implementada com classes e estruturas de dados em Python, representando um estoque de produtos.

Principais funcionalidades:
- Cadastrar produtos quimioterápicos e biológicos (perecíveis e não perecíveis)
- Listar todos os produtos cadastrados
- Listar apenas produtos perecíveis
- Verificar a quantidade total de itens no estoque

Estrutura de rotas da API:

Método HTTP	Endpoint	Descrição
POST	/produtos/quimioterapico	Cadastrar produto quimioterápico
POST	/produtos/vacina_perecivel	Cadastrar vacina perecível
POST	/produtos/vacina_nao_perecivel	Cadastrar vacina não perecível
GET	/produtos/	Listar todos os produtos

GET	/produtos/pereciveis	Listar apenas produtos perecíveis
GET	/estoque/quantidade_total	Exibir quantidade total de produtos

2.2. Clientes

Foram implementados dois clientes, cada um utilizando uma linguagem diferente de Python, conforme exigência do enunciado. Ambos os clientes se comunicam com a API via HTTP e consomem os serviços expostos em JSON.

Cliente 1: Java
- Desenvolvido com a biblioteca HttpURLConnection e HttpClient da JDK padrão.
- Permite ao usuário cadastrar produtos e listar todos os produtos cadastrados por meio de um menu interativo.
- Resposta da API formatada sem bibliotecas externas de JSON, utilizando tratamento manual da string JSON.

Cliente 2: JavaScript (Node.js)
- Implementado com o módulo axios.
- Também apresenta um menu de interação via terminal, utilizando readline.
- Permite cadastrar produtos e listar todos os produtos com facilidade.

Ambos os clientes foram testados localmente e também com o servidor rodando em rede local (acessível via IP 192.168.x.x), demonstrando comunicação bem-sucedida entre diferentes dispositivos e linguagens.

3. Considerações Finais

O trabalho demonstrou com sucesso os conceitos fundamentais de sistemas distribuídos e arquitetura cliente-servidor. A escolha pela API REST com FastAPI mostrou-se eficiente e de fácil integração com diferentes linguagens.

A interoperabilidade foi um dos pontos fortes do projeto, possibilitando a comunicação entre o servidor em Python e clientes implementados em Java e JavaScript sem a necessidade de bibliotecas externas complexas.

O trabalho atende integralmente aos requisitos do enunciado:
- ✅ Substituição do RMI por API HTTP
- ✅ Comunicação cliente-servidor via HTTP
- ✅ Implementação de clientes em duas linguagens distintas
- ✅ Sistema funcional e bem documentado

4. Repositório

Link para o repositório contendo o código-fonte do servidor e dos clientes:
https://github.com/renanalmeida2801/trabalhoSD/tree/main/Trabalho%203

5. Vídeo disponível em:

https://youtu.be/lAVLBhZtGIk