Universidade Federal do Ceará – Campus Quixadá

Disciplina: Sistemas Distribuídos
Professor: Rafael Braga
Curso: Ciência da Computação
Trabalho: Trabalho 3 – Web Services (WS) / API

Autor: Nícolas Ferreira Leite - 552425 / Patrick de Farias Ramos - 556711

---

1. Introdução

Este trabalho tem como objetivo reimplementar o serviço remoto desenvolvido no Trabalho 2 utilizando uma arquitetura cliente-servidor baseada em Web Services (WS) ou Application Programming Interface (API). A comunicação entre cliente e servidor ocorre por meio do protocolo HTTP, seguindo o modelo de requisição e resposta, sem o uso de sockets ou RMI, conforme especificado pelo enunciado da disciplina.

A aplicação desenvolvida simula um sistema de farmácia, permitindo o gerenciamento de medicamentos, controle de estoque e realização de vendas. Diferentemente de uma implementação simples com requisições fixas, o sistema evoluiu para oferecer **interfaces gráficas interativas**, permitindo ao usuário selecionar produtos dinamicamente, caracterizando um cenário mais próximo de aplicações reais distribuídas.

---

2. Tecnologias Utilizadas

- **Linguagem do Servidor:** Python
- **Framework Web (Servidor):** FastAPI
- **Linguagens dos Clientes:**
    - C# (Windows Forms)
    - JavaScript (HTML, CSS e Fetch API)
- **Formato de Comunicação:** JSON sobre HTTP
- **Padrão Arquitetural:** Cliente-Servidor / API REST

---

3. Arquitetura do Sistema

O sistema foi desenvolvido seguindo o modelo cliente-servidor, no qual um único servidor central disponibiliza serviços remotos por meio de uma API REST, enquanto múltiplos clientes, desenvolvidos em linguagens diferentes, consomem esses serviços.

Componentes do Sistema:

- **Servidor (API REST em Python):** Responsável por concentrar toda a lógica de negócio, armazenamento dos dados em memória e disponibilização das rotas HTTP.
- **Cliente Desktop (C#):** Simula um caixa de farmácia (PDV), permitindo ao operador visualizar o estoque e realizar vendas diretamente por meio de uma interface gráfica.
- **Cliente Web (HTML/CSS/JavaScript):** Interface web para gerenciamento do estoque, possibilitando adicionar, remover e visualizar medicamentos em tempo real.

Essa arquitetura demonstra claramente a interoperabilidade entre sistemas distribuídos desenvolvidos em tecnologias distintas.

---

4. Modelagem Orientada a Objetos

4.1 Modelo de Dados

O sistema utiliza a entidade **Medicine** como principal modelo de dados, contendo os seguintes atributos:

- `id`: identificador do medicamento
- `name`: nome do medicamento
- `price`: preço unitário
- `quantity`: quantidade em estoque
- `category`: categoria do medicamento

Esse modelo é compartilhado conceitualmente entre o servidor e os clientes, garantindo consistência na comunicação via JSON.

4.2 Organização em Camadas

- **Camada de Domínio:** Representada pelas classes de modelo que definem a estrutura dos dados.
- **Camada de Serviço:** Implementada no servidor Python, responsável pelas regras de negócio (cadastro, venda, remoção e listagem).
- **Camada de Apresentação:** Representada pelas interfaces gráficas em C# e Web.

---

5. Serviços Remotos Implementados

5.1 Listagem de Medicamentos

- **Rota:** `GET /medicines`
- **Descrição:** Retorna todos os medicamentos cadastrados no estoque.

5.2 Cadastro de Medicamentos

- **Rota:** `POST /medicines`
- **Descrição:** Adiciona um novo medicamento ao estoque da farmácia.
- **Entrada:** Dados do medicamento em formato JSON.

5.3 Venda de Medicamentos

- **Rota:** `POST /sell/{med_id}/{quantity}`
- **Descrição:** Realiza a venda de um medicamento específico, reduzindo a quantidade disponível em estoque.

5.4 Remoção de Medicamentos

- **Rota:** `DELETE /medicines/{med_id}`
- **Descrição:** Remove um medicamento do sistema.

---

6. Clientes da Aplicação

6.1 Cliente Desktop (C# – Windows Forms)

O cliente em C# simula um sistema de ponto de venda (PDV) de uma farmácia. Ele apresenta uma tabela com os medicamentos disponíveis, permitindo ao usuário selecionar um item e realizar a venda com apenas um clique. Todas as operações são realizadas por meio de requisições HTTP à API Python.

Esse cliente evidencia a integração entre linguagens distintas e o consumo de serviços REST em aplicações desktop.

6.2 Cliente Web (HTML, CSS e JavaScript)

O cliente web fornece uma interface gráfica acessível via navegador, permitindo o gerenciamento do estoque de medicamentos. A interface possibilita adicionar novos produtos, visualizar o estoque atualizado em tempo real e remover medicamentos.

A comunicação com a API ocorre por meio da Fetch API, reforçando o uso de padrões modernos de desenvolvimento web.

---

7. Execução do Sistema

1. O servidor FastAPI é iniciado na porta 8000.
2. Os clientes (C# e Web) realizam requisições HTTP para a API.
3. A API processa as requisições, aplica as regras de negócio e retorna respostas em formato JSON.
4. As interfaces gráficas refletem as alterações no estoque em tempo real.

---

8. Conclusão

O desenvolvimento deste trabalho permitiu aplicar, de forma prática, os conceitos fundamentais de sistemas distribuídos, como APIs REST, comunicação cliente-servidor, interoperabilidade entre linguagens e separação de responsabilidades.

A utilização de múltiplos clientes, incluindo uma aplicação desktop e uma aplicação web, demonstra a flexibilidade e o poder das APIs baseadas em HTTP, atendendo integralmente aos requisitos propostos pela disciplina.