Sistema de Gerenciamento de Chocolataria

Trabalho A3 - UC Sistemas Distribuídos e Mobile

Prof.: Adailton Cerqueira Jr.

- **André Vinícius**
- **Nayara Lorena**
- **Wendel Alves**
- **Victor Sobral**

---

1. Introdução

O presente trabalho tem como objetivo desenvolver um sistema distribuído para gerenciamento de uma rede de chocolaterias, atendendo aos requisitos específicos do trabalho A3 da disciplina de Sistemas Distribuídos e Mobile. A escolha do tema chocolataria foi motivada pela necessidade de criar um cenário real e atrativo que demonstre a aplicação prática dos conceitos de sistemas distribuídos em um ambiente comercial.

O sistema foi projetado para simular operações reais de uma rede de chocolaterias, incluindo gestão de clientes, vendedores, estoque, processamento de vendas e geração de relatórios estatísticos. A abordagem metodológica adotada baseia-se na arquitetura de microserviços, permitindo escalabilidade, manutenibilidade e separação clara de responsabilidades entre os diferentes componentes do sistema.

Os objetivos principais deste projeto incluem: (1) implementar um sistema distribuído funcional utilizando tecnologias modernas como Node.js, PostgreSQL e Docker; (2) demonstrar a comunicação eficiente entre microserviços independentes.

2. Fundamentação Teórica

2.1 Sistemas Distribuídos

Sistemas distribuídos são coleções de componentes de software independentes que aparecem aos usuários como um sistema único e coerente. Segundo Tanenbaum e Van Steen, essas arquiteturas oferecem vantagens como escalabilidade, tolerância a falhas e

compartilhamento de recursos. No contexto deste projeto, implementamos uma arquitetura distribuída onde diferentes serviços operam independentemente, comunicando-se através de APIs RESTful.

2.2 Arquitetura de Microserviços

A arquitetura de microserviços é um padrão arquitetural que estrutura uma aplicação como uma coleção de serviços pequenos, autônomos e focados em negócios específicos. Cada microserviço é implantado independentemente e se comunica através de APIs bem definidas. Esta abordagem oferece benefícios como:

- **Escalabilidade independente**: Cada serviço pode ser dimensionado conforme sua demanda específica
- **Tecnologia heterogênea**: Diferentes serviços podem utilizar tecnologias distintas
- **Resiliência**: Falhas em um serviço não comprometem todo o sistema
- **Desenvolvimento paralelo**: Equipes podem trabalhar independentemente em diferentes serviços

2.3 Tecnologias Utilizadas

Node.js e Express.js

Node.js é um ambiente de execução JavaScript server-side construído sobre o motor V8 do Google Chrome. Sua arquitetura orientada a eventos e não-bloqueante torna-o ideal para aplicações distribuídas que requerem alta concorrência. O Express.js é um framework web minimalista e flexível que fornece recursos robustos para desenvolvimento de APIs RESTful.

PostgreSQL

PostgreSQL é um sistema de gerenciamento de banco de dados relacional objeto-relacional de código aberto, conhecido por sua confiabilidade, robustez de recursos e performance. Oferece suporte completo a ACID (Atomicidade, Consistência, Isolamento, Durabilidade) e é amplamente utilizado em aplicações empresariais.

Docker e Containerização

Docker é uma plataforma de containerização que permite empacotar aplicações e suas dependências em containers portáveis. Os containers oferecem isolamento, consistência entre ambientes e facilidade de deployment. Docker Compose permite orquestrar múltiplos containers, definindo e gerenciando aplicações multi-container.

2.4 Padrões de Design Implementados

Padrão DAO (Data Access Object)

O padrão DAO abstrai e encapsula todo o acesso à fonte de dados, fornecendo uma interface uniforme para operações de persistência. Isso permite separar a lógica de negócio da lógica de acesso a dados, facilitando manutenção e testes.

Padrão MVC (Model-View-Controller)

- **Models**: Representação das entidades de domínio
- **Controllers**: Lógica de negócio e processamento de requisições
- **Routes**: Definição de endpoints e roteamento

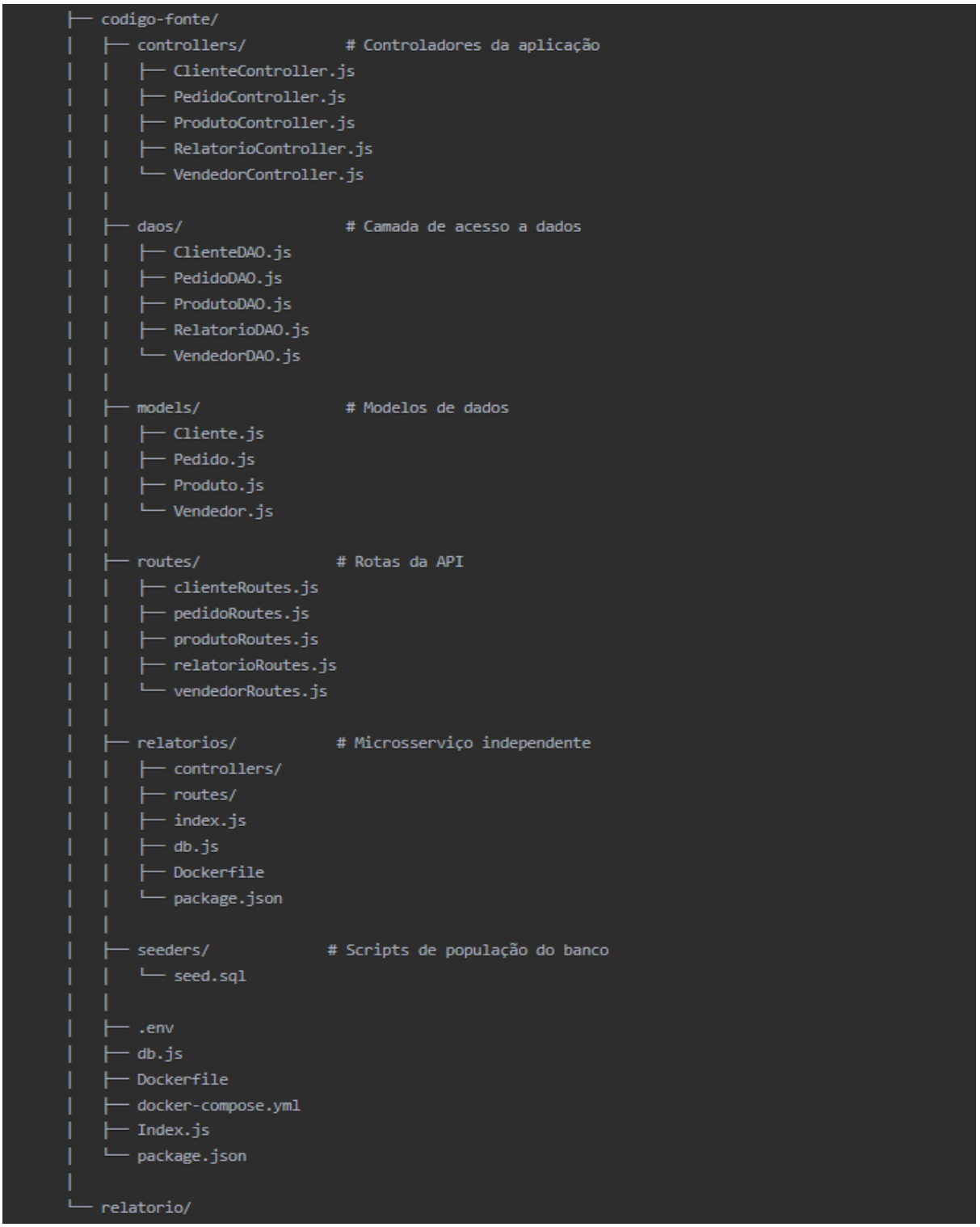
3. Projeto de Implementação

3.1 Arquitetura Geral do Sistema

O sistema foi estruturado seguindo uma arquitetura de microserviços composta por:

1. **API Principal** (Porta 3000): Responsável pela gestão de clientes, vendedores, produtos e pedidos
2. **Microserviço de Relatórios** (Porta 3001): Serviço independente para geração de relatórios estatísticos
3. **Banco de Dados PostgreSQL** (Porta 5432): Armazenamento relacional compartilhado
4. **Orquestração Docker**: Gerenciamento de containers e rede isolada

3.2 Estrutura do Projeto



3.3 Modelagem de Dados

O sistema utiliza um modelo relacional com as seguintes entidades principais:

Entidade Cliente

- `id` (Chave Primária)
- `nome` (VARCHAR)
- `email` (VARCHAR UNIQUE)
- `telefone` (VARCHAR)
- `endereco` (TEXT)
- `cpf` (VARCHAR UNIQUE)

Entidade Produto

- `id` (Chave Primária)
- `nome` (VARCHAR)
- `descricao` (TEXT)
- `preco` (DECIMAL)
- `estoque` (INTEGER)
- `categoria` (VARCHAR)

Entidade Vendedor

- `matricula` (Chave Primária)
- `nome` (VARCHAR)
- `email` (VARCHAR UNIQUE)
- `telefone` (VARCHAR)
- `data_admissao` (DATE)
- `salario` (DECIMAL)

Entidade Pedido

- `id` (Chave Primária)
- `cliente_id` (Chave Estrangeira → Cliente)
- `vendedor_id` (Chave Estrangeira → Vendedor)
- `data_pedido` (TIMESTAMP)
- `valor_total` (DECIMAL)
- `status` (VARCHAR)

Entidade Item_Pedido

- `id` (Chave Primária)
- `pedido_id` (Chave Estrangeira → Pedido)
- `produto_id` (Chave Estrangeira → Produto)
- `quantidade` (INTEGER)
- `preco_unitario` (DECIMAL)

3.4 Comunicação Entre Serviços

Os microserviços comunicam-se através de:

1. **APIs RESTful**: Endpoints HTTP com JSON
2. **Banco de Dados Compartilhado**: PostgreSQL como fonte única de verdade
3. **Rede Docker Isolada**: Subnet customizada (172.23.0.0/24)
4. **Resolução de Nomes**: Docker DNS para comunicação inter-container

Funcionalidades Implementadas Conforme A3

Gerenciar Cliente

- CRUD Completo: Create, Read, Update, Delete
- Endpoints:
    - GET /clientes - Listar todos os clientes
    - GET /clientes/:id - Buscar cliente específico
    - POST /clientes - Criar novo cliente
    - PUT /clientes/:id - Atualizar cliente
    - DELETE /clientes/:id - Remover cliente

Gerenciar Vendedor

- CRUD Completo: Create, Read, Update, Delete
- Endpoints:
    - GET /vendedor - Listar todos os vendedores
    - GET /vendedor/:id - Buscar vendedor específico
    - POST /vendedor - Criar novo vendedor
    - PUT /vendedor/:id - Atualizar vendedor
    - DELETE /vendedor/:id - Remover vendedor

Gerenciar Estoque/Vendas

- CRUD Estoque: Gestão completa de produtos
- Receber Pedido: Sistema de criação de vendas
- Cancelar Pedido: Funcionalidade de cancelamento

- Endpoints:
    - GET /produtos - Listar produtos (estoque)
    - POST /produtos - Cadastrar produto
    - PUT /produtos/:id - Atualizar produto/estoque
    - POST /pedidos - Receber pedido de compra
    - DELETE /pedidos/:id - Cancelar pedido

Geração de Relatórios Estatísticos (Serviço Separado)

- Produtos Mais Vendidos: GET /relatorios/mais_vendido
- Produto por Cliente: GET /relatorios/cliente/total/:id
- Consumo Médio do Cliente: GET /relatorios/cliente/media/:id
- Produto com Baixo Estoque: GET /relatorios/produto_estoque_baixo

4. Considerações Finais

O desenvolvimento do Sistema de Gerenciamento de Chocolataria representou uma experiência enriquecedora na implementação prática dos conceitos de sistemas distribuídos. Os resultados obtidos demonstram a viabilidade e eficiência da arquitetura de microserviços para aplicações comerciais.

5. Bibliografia

- FOWLER, M. **Microservices: a definition of this new architectural term**. Disponível em: https://martinfowler.com/articles/microservices.html. Acesso em: 2025.
- DOCKER INC. **Docker Documentation**. Disponível em: https://docs.docker.com/. Acesso em: 2025.
- POSTGRESQL GLOBAL DEVELOPMENT GROUP. **PostgreSQL Documentation**. Disponível em: https://www.postgresql.org/docs/. Acesso em: 2025.
- NODE.JS FOUNDATION. **Node.js Documentation**. Disponível em: https://nodejs.org/docs/. Acesso em: 2025.

- Tutoriais sobre RESTful APIs e microserviços da DigitalOcean