

**Relatório**

**Loja Esportiva**

Caique Silva dos Reis

1272322080

Salvador-BA

25.1-2025

### **1. Introdução**

Este relatório descreve o desenvolvimento de um sistema distribuído voltado para a simulação das operações de uma rede de lojas especializada na venda de roupas esportivas. Dada a crescente valorização do mercado fitness e o foco da população em bem-estar e saúde, esse setor mostra-se oportuno para a aplicação de conceitos de sistemas distribuídos e análise de dados.

O sistema desenvolvido visa cobrir as operações principais de uma loja de roupas esportivas: cadastro de clientes e vendedores, controle de produtos e estoque, processamento de vendas e geração de relatórios analíticos. Para isso, foi adotada uma arquitetura baseada em microsserviços, em que cada funcionalidade é dividida em serviços independentes, executados em containers Docker e se comunicando via APIs RESTful. Todos os serviços compartilham um banco de dados PostgreSQL centralizado.

Além das funcionalidades básicas, o sistema implementa controle rigoroso de estoque, recebimento de pedidos e possibilidade de cancelamento de compras, além de um serviço específico para geração de relatórios. Essa abordagem simula um ambiente real de varejo, onde decisões estratégicas dependem diretamente da análise de dados operacionais.

### **2. Fundamentação Teórica**

Sistemas distribuídos são compostos por múltiplos serviços independentes que interagem para compor um sistema coeso. Esse modelo permite escalabilidade, isolamento de falhas e facilidade de manutenção. A arquitetura utilizada no projeto segue o padrão de microsserviços, onde cada serviço tem responsabilidades bem definidas.

A implementação utilizou JavaScript com o framework Express (Node.js) para construção das APIs REST, enquanto o banco de dados PostgreSQL foi adotado pela sua confiabilidade e suporte a transações. A orquestração dos serviços foi feita com Docker, utilizando o docker-compose para facilitar o gerenciamento dos containers.

Dentre os serviços implementados, destaca-se o módulo de relatórios analíticos, que realiza consultas agregadas sobre os dados de vendas e produtos, gerando insights como os produtos mais vendidos, o consumo médio dos clientes e itens com baixo estoque. Esta separação permite a escalabilidade independente do serviço de análise, de acordo com a demanda.

### **3. Projeto de Implementação**

#### **3.1 Estrutura Geral da Aplicação**

A aplicação é composta pelos seguintes serviços:

* **clientes-service**: gerenciamento de clientes (CRUD)
* **vendedores-service**: gerenciamento de vendedores (CRUD)
* **produtos-service**: gerenciamento de produtos e controle de estoque
* **vendas-service**: processamento de pedidos de compra e cancelamentos
* **relatorios-service**: geração de relatórios estatísticos e operacionais

Cada serviço roda de forma isolada em containers Docker e se comunica via HTTP. Todos os serviços acessam um banco de dados PostgreSQL comum, o que garante integridade dos dados e consistência entre os módulos.

#### **3.2 Modelagem dos Dados**

As principais tabelas utilizadas no projeto são:



Obs: O sistema também foi estruturado para permitir geração de um diagrama entidade-relacionamento (ER), caso desejado.

#### **3.3 Tecnologias Utilizadas**

* **Node.js** e **Express**: para construção das APIs REST
* **PostgreSQL**: banco de dados relacional
* **Docker** e **Docker Compose**: containerização dos serviços
* **Axios**: comunicação entre serviços (quando aplicável)
* **Jest/Vitest** (opcional): estrutura para testes automatizados

#### **3.4 Funcionalidades Implementadas**

* **Gerenciamento de Estoque**: o módulo de produtos permite cadastrar, editar e consultar produtos com atualização em tempo real da quantidade em estoque.
* **Recebimento de Pedidos de Compra**: o módulo de vendas verifica o estoque antes de registrar a venda e atualiza a quantidade disponível. Cada venda é registrada junto com seus itens, cliente e vendedor responsáveis.
* **Cancelamento de Pedido**: uma rota específica foi implementada para cancelar uma venda. Ao cancelar, os itens vendidos são excluídos e os produtos têm o estoque restaurado.
* **Geração de Relatórios**: o serviço de relatórios disponibiliza as seguintes análises:  
  + Produtos mais vendidos
  + Produtos comprados por cliente
  + Consumo médio por cliente
  + Produtos com baixo estoque
  + Total de vendas por cliente
  + Total de vendas por vendedor
  + Total de vendas por produto

#### **3.5 Fluxo de uma Venda**

1. O cliente escolhe os produtos e envia um pedido.
2. O serviço de vendas valida o estoque disponível.
3. A venda é registrada no banco de dados.
4. O estoque dos produtos vendidos é atualizado.
5. O serviço de relatórios coleta as informações para geração de estatísticas.

### **4. Considerações Finais**

O sistema desenvolvido simula com fidelidade as operações essenciais de uma loja de roupas esportivas, adotando boas práticas de modularidade, escalabilidade e comunicação entre microsserviços. A utilização de Docker e PostgreSQL proporcionou estabilidade e flexibilidade ao ambiente, enquanto as APIs REST possibilitaram integrações fáceis entre os módulos.

Como diferencial, foi implementado um módulo robusto de relatórios e uma funcionalidade de cancelamento de vendas com recuperação de estoque, alinhando o projeto com situações reais de um ambiente de varejo.

Entre os desafios enfrentados, destacam-se a orquestração de containers, o gerenciamento transacional entre tabelas relacionadas (especialmente na venda e cancelamento), e a padronização de rotas(foi insuportável fazer isso, professor) e respostas para integração com o serviço de relatórios. Superados esses pontos, a aplicação mostrou-se funcional e eficiente.

Como melhorias futuras, poderíamos implementar autenticação JWT para proteger os serviços, uma interface gráfica para facilitar o uso da aplicação, e até uma camada de cache com Redis para otimizar o tempo de geração de relatórios. Apesar dos desafios, a solução atendeu todos os requisitos propostos e simula com fidelidade uma aplicação real para o mercado de roupas esportivas. (Confesso que fazer esse trabalho me deu vontade de trabalhar em uma loja só para aplicar, pena que o mercado de trabalho é difícil, mas vai dar certo)

### **5. Bibliografia**

BROWN, Ethan Wilson. *Desenvolvendo com Node.js*. São Paulo: Novatec Editora, 2015.

COULOURIS, George; DOLLIMORE, Jean; KINDBERG, Tim. *Sistemas Distribuídos: Conceitos e Projeto*. 5. ed. Porto Alegre: Bookman, 2012.

Docker Inc. *Docker Documentation*. Disponível em:<https://docs.docker.com/>. Acesso em: 11 jun. 2025.

Express.js. *Express – API Reference*. Disponível em:<https://expressjs.com/>. Acesso em: 11 jun. 2025.

FERREIRA, Iago. *Aprenda Cloud e DevOps do ZERO*. [S.l.]: YouTube, 2023. Disponível em:<https://www.youtube.com/watch?v=N3tNmoLbKwg>. Acesso em: 11 jun. 2025.

GODOY, Leandro. *PostgreSQL: Guia do Programador*. São Paulo: Novatec Editora, 2019.

KIPPER, Fernanda. *Docker e Docker Compose na Prática*. [S.l.]: YouTube, 2022. Disponível em:<https://www.youtube.com/watch?v=DdoncfOdru8>. Acesso em: 11 jun. 2025.

NEWMAN, Sam. *Building Microservices*. 2nd ed. Sebastopol: O’Reilly Media, 2021.

Node.js Foundation. *Node.js Documentation*. Disponível em:<https://nodejs.org/en/docs>. Acesso em: 11 jun. 2025.

OPENAI. *ChatGPT*. São Francisco, 2025. Disponível em:<https://chat.openai.com>. Acesso em: 11 jun. 2025. Utilizado como apoio para sugestões de escrita e aprimoramento textual durante o desenvolvimento do relatório.

PostgreSQL Global Development Group. *PostgreSQL Documentation*. Disponível em:<https://www.postgresql.org/docs/>. Acesso em: 11 jun. 2025.

Postman, Inc. *Postman Learning Center*. Disponível em:<https://learning.postman.com/>. Acesso em: 11 jun. 2025.

RICHARDSON, Leonard; RUBY, Mike. *RESTful Web Services*. Sebastopol: O’Reilly Media, 2007.

TANENBAUM, Andrew S.; VAN STEEN, Maarten. *Sistemas Distribuídos: Princípios e Paradigmas*. 2. ed. São Paulo: Pearson, 2007.

TURNBULL, James. *The Docker Book: Containerization is the New Virtualization*. Independently published, 2014.