

Documentação do Projeto: TechStore

v1.1.0

Versão: 1.1.0

Projeto: Sistema de Loja Online (Full-Stack)

GitHub/Relatório: <https://github.com/Kauaog13/tech-store-java?tab=readme-ov-file>

Stack: Java (Backend) + React/TypeScript (Frontend)

1. Introdução

Bem-vindo à documentação oficial da versão 1.0.0 do Sistema de Loja Online ("TechStore"). Este documento descreve o escopo, as funcionalidades e os casos de uso da plataforma.

O projeto consiste em um sistema de e-commerce full-stack focado na venda de componentes e periféricos de PC. Esta versão inclui **autenticação de usuários**, **gerenciamento de perfis de cliente** e um **painel de administração** completo para gerenciamento da loja.

A arquitetura do sistema é desacoplada, composta por:

- **Backend:** Uma API RESTful robusta construída em **Java** (Spring Boot) e gerenciada com **Maven**.
- **Frontend:** Uma interface de usuário (UI) moderna e reativa construída em **React** com **TypeScript**.
- **Banco de Dados:** **MySQL** para persistência de dados.

2. Objetivo

O objetivo principal da versão 1.1.0 é estabelecer um ecossistema de e-commerce completo e gerenciável, fornecendo experiências distintas e seguras para Clientes e Administradores.

Objetivos do Cliente (Fluxo 1):

1. Criar uma conta e gerenciar seu próprio perfil.
2. Navegar pelo catálogo de produtos.
3. Adicionar e gerenciar itens em um carrinho de compras.
4. Finalizar um pedido com sucesso, vinculado à sua conta.

Objetivos do Administrador (Fluxo 2):

1. Analisar a saúde da loja através de um Dashboard.
2. Ter controle total (CRUD) sobre o catálogo de produtos (Estoque).

3. Gerenciar e monitorar todos os pedidos (Vendas) realizados pelos clientes.

3. Funcionalidades (Escopo v1.0.0)

A versão 1.0.0 inclui as seguintes funcionalidades:

3.1. Módulo de Autenticação e Contas

- **Login:** Permite que Clientes e Administradores acessem o sistema com credenciais únicas (email e senha). O sistema deve diferenciar os papéis (**ROLE_CLIENTE**, **ROLE_ADMIN**).
- **Registro de Cliente:** Novos usuários podem criar uma conta de cliente.
- **Logout:** Encerramento seguro da sessão do usuário.

3.2. Módulo do Cliente (Fluxo 1)

- **Gerenciamento de Perfil:** Após o login, o cliente tem uma página própria onde pode visualizar e **editar suas informações pessoais** (nome e endereço).
- **Listagem de Produtos:** O frontend exibe todos os produtos disponíveis.
- **Detalhes do Produto:** O usuário pode clicar em um produto para ver detalhes.
- **Módulo de Carrinho:**
 - Adicionar ao Carrinho: O cliente pode adicionar um **Produto** ao seu carrinho.
 - Gerenciar Quantidade: O cliente pode alterar a quantidade de um **ItemPedido**.
 - Remover do Carrinho: O cliente pode remover um item do carrinho.
- **Módulo de Checkout:**
 - Formulário de Cliente: Os dados do cliente (nome, endereço) são **preenchidos automaticamente ou manualmente** a partir do seu perfil, podendo ser confirmados ou alterados para aquele pedido específico.
 - Finalização de Pedido: Envio dos dados para o backend.

3.3. Módulo de Pedidos e Regras de Negócio (Backend)

- **Criação de Pedido:** O backend cria um novo **Pedido** associado ao **Cliente** (logado).
- **Validação Crítica de Estoque:** A lógica da **EstoqueInsuficienteException** é mantida para bloquear pedidos sem estoque.

3.4. Módulo de Administração (Painel Admin - Fluxo 2)

Após o login, o Administrador é redirecionado para uma página própria de gerenciamento com 3 sub-páginas:

- **3.4.1. Dashboard (Principal)**
 - Exibe relatórios visuais (gráficos e KPIs) sobre a saúde do e-commerce.
 - Indicadores de vendas totais, ticket médio e pedidos recentes.
 - Relatórios de produtos mais vendidos e com baixo estoque.
- **3.4.2. Gerenciamento de Estoque**
 - Permite ao administrador gerenciar todo o ciclo de vida dos produtos.
 - **Criar:** Adicionar um novo produto ao catálogo.
 - **Listar/Editar:** Visualizar todos os produtos em uma tabela, com opções para editar preço, nome, descrição, categoria e quantidade em estoque.

- **Excluir:** Remover um produto da loja.
- **3.4.3. Gerenciamento de Vendas**
 - Permite ao administrador gerenciar os pedidos dos clientes.
 - **Listar:** Visualizar todos os pedidos realizados no sistema.
 - **Detalhar:** Acessar informações detalhadas de um pedido (quem comprou, o que comprou, endereço de entrega, valor total).
 - **Editar:** Alterar o status de um pedido ("PENDENTE", "ENVIADO", 'PAGO', CANCELADA).
 - **Excluir:** Cancelar/excluir um pedido.

4. Casos de Uso (Use Cases)

Abaixo estão os casos de uso para a v1.0.0.

Ator: Usuário (Qualquer)

| ID | Caso de Uso | Descrição da Ação |
|--------|-----------------|--|
| UC-001 | Realizar Login | Como um usuário, eu quero inserir meu email e senha, para que eu possa acessar as funcionalidades protegidas do sistema (seja como Cliente ou Admin). |
| UC-002 | Registrar Conta | Como um novo visitante, eu quero criar uma conta de cliente, para que eu possa salvar meus dados e realizar compras. |

Ator Principal: Cliente

| ID | Caso de Uso | Descrição da Ação |
|--------|-------------------|---|
| UC-101 | Visualizar Perfil | Como um cliente logado, eu quero acessar minha página de perfil, para que eu possa ver meus dados cadastrados e meus pedidos criados/cancelados. |

| | | |
|--------|-------------------------------------|--|
| UC-102 | Editar Perfil | Como um cliente logado, eu quero editar meu nome e endereço no meu perfil, para que minhas compras futuras já tenham esses dados corretos. |
| UC-103 | Navegar pelo Catálogo | Como um cliente, eu quero ver a lista de todos os produtos disponíveis. |
| UC-104 | Adicionar ao Carrinho | Como um cliente, eu quero adicionar produtos ao meu carrinho. |
| UC-105 | Ajustar Carrinho | Como um cliente, eu quero alterar a quantidade de itens no meu carrinho. |
| UC-106 | Remover do Carrinho | Como um cliente, eu quero remover um item do carrinho. |
| UC-107 | Finalizar Pedido (Sucesso) | Como um cliente logado, eu quero finalizar minha compra, para que o sistema crie um pedido associado à minha conta. |
| UC-108 | Finalizar Pedido (Falha de Estoque) | Como um cliente, eu quero ser impedido de comprar um item se a quantidade solicitada for maior que o estoque. |
| UC-109 | Cancelar Pedido | O cliente pode cancelar seu próprio pedido através da aba "Meus Pedidos", desde que o status seja "PENDENTE". Isso aciona o estorno automático do estoque. |
| UC-110 | Consultar Histórico de Pedidos | O cliente acessa a aba " Meus Pedidos " no perfil para ver o histórico completo e status de compras anteriores. |

Ator Secundário: Administrador

| ID | Caso de Uso | Descrição da Ação |
|--------|-------------------------|--|
| UC-201 | Ver Dashboard | Como um administrador, eu quero acessar o Dashboard, para que eu possa ver relatórios de vendas e produtos. |
| UC-202 | Criar Produto | Como um administrador, eu quero acessar a página "Estoque" e adicionar um novo produto. |
| UC-203 | Editar Produto | Como um administrador, eu quero editar um produto existente, para que eu possa atualizar seu preço, nome, categoria, imagem ou estoque. |
| UC-204 | Excluir Produto | Como um administrador, eu quero excluir um produto, para que ele não seja mais vendido. |
| UC-205 | Listar Vendas | Como um administrador, eu quero acessar a página "Vendas", para que eu possa ver a lista de todos os pedidos feitos. |
| UC-206 | Gerenciar Pedido | Como um administrador, eu quero abrir um pedido específico e editar seu status (ex: "PAGO" para "ENVIADO"). |
| UC-207 | Excluir/Cancelar Pedido | Como um administrador, eu quero excluir um pedido (ex: duplicado ou fraudulento). |

5. Ferramentas de I.A Utilizadas

Este projeto foi acelerado com ferramentas de Inteligência Artificial para design, arquitetura e geração de código:

- **Gemini 2.5 PRO**
 - Utilizado para: Engenharia de prompt, Design da arquitetura do sistema (Front-end), Desenvolvimento do Backend, Definição das classes Java e suas responsabilidades, Geração de documentação.
- **Bolt.new**
 - Usado para *scaffolding* do frontend, com: Estrutura de pastas e componentes React, Criação de interfaces TypeScript, Camada de serviço (Axios) integrada ao backend Java.

6. Principais Prompts Utilizados

- Bolt.new:

Bolt.new Optimized Prompt (TechStore v1.0.0 Frontend)

1. Persona (Role)

You are a Senior Frontend Software Engineer and UI/UX expert. Your specialty is building modern, responsive, and scalable Single Page Applications (SPAs) using React (Vite) + TypeScript. You excel at translating backend documentation into clean, functional, and production-ready user interfaces.

2. Primary Goal

Your goal is to generate the complete initial codebase and file structure for the "TechStore v1.0.0" frontend. This application will consume a pre-defined Java API. The generated code must be modern, strictly-typed, and fully implement all user flows defined in the project documentation (Auth, Client Flow 1, Admin Flow 2).

3. Core Specifications & Constraints

- Framework: React (Vite) + TypeScript
- UI Library: **Shadcn/UI + Tailwind CSS** (You must use **Shadcn/UI** components like **Button**, **Input**, **Card**, **Table**, **Dialog**, etc., for a modern feel).
- Routing: **react-router-dom** (v6+)
- State Management: **Zustand** (Use this for global state, specifically for Auth and Cart).
- API Client: **axios**.

!!! CRITICAL LANGUAGE CONSTRAINT !!!

- All user-facing text in the UI (labels, buttons, placeholders, titles, error messages, etc.) **MUST** be in Brazilian Portuguese (pt-BR).
- (Code, comments, and file names should remain in English).

4. Backend API Contract (The "Source of Truth")

The frontend will consume a Java (Spring Boot) / Maven API connected to a MySQL database. This dictates our data models: all **id** keys will be of type **number** (mapping from MySQL **BIGINT**).

Generate the following interfaces in **/src/types/models.ts**:

TypeScript

```
// /src/types/models.ts
```

```
// Maps the 'usuarios' table (id: BIGINT)
```

```
export interface User {
```

```
  id: number;
```

```
  nome: string;
```

```
  email: string;
```

```
  endereco: string;
```

```
  role: 'CLIENTE' | 'ADMIN';
```

```
}
```

```
// Maps the 'produtos' table (id: BIGINT)
```

```
export interface Product {
```

```
  id: number;
```

```
  nome: string;
```

```
  preco: number;
```

```
  estoque: number;
```

```
  imageUrl: string;
```

```
}
```

```
// Logical interface for the Zustand cart store
```

```
export interface CartItem {
```

```
  product: Product;
```

```
  quantidade: number;
```

```
}
```

```
// Maps the 'pedidos' table (id: BIGINT)
```

```
export interface Order {
```

```
  id: number;
```

```
  cliente: User; // Backend will populate this
```

```
  itens: CartItem[]; // Backend will populate this
```

```
  status: 'PENDENTE' | 'PAGO' | 'ENVIADO' | 'CANCELADO';
```

```
  valorTotal: number;
```

```
  dataPedido: string; // Mapped from MySQL TIMESTAMP
```

```
}
```

5. Required File Structure

```
/src
```

```
  /components
```

```
    /layout    # AdminLayout.tsx, ClientLayout.tsx
```

```
    /ui        # Shadcn components (Button, Input, etc.)
```

```
  /pages
```

```
    /auth      # LoginPage.tsx, RegisterPage.tsx
```

```
    /client    # HomePage.tsx, CartPage.tsx, ProfilePage.tsx
```


`/admin` `# DashboardPage.tsx, AdminEstoquePage.tsx, AdminVendasPage.tsx`

`/store` `# useAuthStore.ts, useCartStore.ts`

`/services` `# apiService.ts`

`/hooks` `# (custom hooks)`

`/lib` `# (utils, shadcn helper)`

`/types` `# models.ts`

`/routes` `# AppRoutes.tsx`

`App.tsx`

`main.tsx`

6. Action: Generate All Features & Flows

6.1. Global State (Zustand)

- `useAuthStore.ts`: Create a Zustand store to manage `user: User | null`, `token: string | null`, and actions: `login`, `register`, `logout`.
- `useCartStore.ts`: Create a Zustand store to manage `items: CartItem[]` and actions: `addToCart(product: Product)`, `removeFromCart(productId: number)`, `updateQuantity(productId: number, quantity: number)`.

6.2. Core & Auth Flow (Public)

- `AppRoutes.tsx`: Configure all routes using `react-router-dom`. Implement a `ProtectedRoute` component for clients and an `AdminRoute` component (checking for `role === 'ADMIN'`).
- `LoginPage.tsx`: Generate a centered form with `Input` (Shadcn) for "Email" and "Senha". Include a "Entrar" `Button` and a link: "Não tem uma conta? Registre-se".
- `RegisterPage.tsx`: Generate a form with `Input` for "Nome", "Email", "Endereço", and "Senha". Include a "Criar Conta" `Button` and a link back to Login.

6.3. Client Flow (Fluxo 1 - Client Protected Route)

- **ClientLayout.tsx**: A layout with a Header/Navbar. The Navbar must display the "TechStore" logo, a Cart icon (with item count from **useCartStore**), and a user profile dropdown (with links to "Meu Perfil" and a "Sair" button).
- **HomePage.tsx**: Display a grid of products using a reusable **ProductCard** component.
- **ProductCard.tsx** (in **/components**): Must be a **Card** (Shadcn) showing product **imageUrl**, **nome**, and **preco**. Must have an "Adicionar ao Carrinho" **Button**.
- **CartPage.tsx**: Display a list of items from **useCartStore**. Allow quantity updates (using an **Input**) and item removal. Show "Valor Total". Include a "Finalizar Compra" **Button**.
- **ProfilePage.tsx**: "Meu Perfil" page. Display a form (pre-filled from **useAuthStore**) allowing the user to edit "Nome" and "Endereço". Include a "Salvar Alterações" **Button**.

6.4. Admin Flow (Fluxo 2 - Admin Protected Route)

- **AdminLayout.tsx**: A layout with a persistent Sidebar (Menu Lateral) and a main content area. The Sidebar must have links with icons for: "Dashboard", "Estoque", and "Vendas".
- **DashboardPage.tsx**: Main page. Use **Card** (Shadcn) components as placeholders for "Vendas Totais", "Pedidos Recentes", and "Produtos Mais Vendidos". (e.g., **<h2>Vendas Totais</h2>**).
- **AdminEstoquePage.tsx**: "Gerenciar Estoque" page. Must include a "Adicionar Novo Produto" **Button** (which opens a **Dialog** (Shadcn) with a creation form). Must display a **DataTable** (Shadcn) listing all products.
 - Table Columns: ID, Nome, Preço, Estoque, Ações.
 - Actions Column: Must have "Editar" and "Excluir" buttons (or a dropdown menu) that trigger respective **Dialogs**.
- **AdminVendasPage.tsx**: "Gerenciar Vendas" page. Must display a **DataTable** listing all **Orders**.
 - Table Columns: ID Pedido, Cliente (nome), Data, Status, Valor Total, Ações.
 - Actions Column: Must have a "Ver Detalhes" button (opens a **Dialog** to show **items**) and a **Select** (Shadcn) to allow editing the **status** (e.g., 'PENDENTE' -> 'ENVIADO').