**Serviço Nacional de Aprendizagem Industrial**

**Gabriel Barbosa Magalhães**

**José Vitor Masiero da Silva**

**Samara de Souza Bento**

**Thiago Elias Antunes**

**RELATÓRIO SISTÊMICO**

**VAMO.COME**

**TUBARÃO**

**2025**

**Gabriel Barbosa Magalhães**

**José Vitor Masiero da Silva**

**Samara de Souza Bento**

**Thiago Elias Antunes**

**RELATÓRIO SISTÊMICO**

**VAMO.COME**

Relatório apresentado à disciplina Desenvolvimento de Sistemas, do curso de Técnico de Análise e Desenvolvimento de Sistemas do SENAI como primeira atividade prática, sob a orientação do prof.º Matheus da S. Carvalho.

TUBARÃO

2025

**RESUMO**

Este relatório apresenta a documentação técnica de uma plataforma de delivery responsiva, inspirada no modelo do iFood, desenvolvida para integrar clientes e estabelecimentos alimentícios em um ambiente digital eficiente. O sistema foi projetado para atender às principais demandas de uma aplicação de pedidos online, incluindo cadastro e login de usuários, gerenciamento de produtos, realização e acompanhamento de pedidos, avaliação de serviços e controle de status de entrega. Utilizando uma arquitetura cliente-servidor, com **Spring Boot** no backend e **HTML/CSS/JavaScript/TypeScript** no frontend, o sistema oferece uma experiência fluida e responsiva, acessível em diferentes dispositivos. A documentação contempla a descrição da arquitetura, estrutura dos módulos, organização das rotas, funcionamento da comunicação entre as camadas e análise das principais interfaces de usuário. Os resultados demonstram que a plataforma desenvolvida é capaz de simular com fidelidade um ambiente de delivery real, sendo aplicável a pequenos e médios negócios interessados em autonomia tecnológica e agilidade no atendimento ao cliente.

**Palavras-chave:** Plataforma de Delivery, Spring Boot, JavaScript, Vite, Pedidos Online, Responsividade.

**LISTA DE ILUSTRAÇÕES**

[Figura 1 - Tela Inicial Cliente 10](#_Toc199708762)

[Figura 2 - Tela de login para fornecedor ou cliente 11](#_Toc199708763)

[Figura 3 - Tela de Cadastro de Cliente 11](#_Toc199708764)

[Figura 4 - Tela de Visualização de Fornecedor e seus Produtos 12](#_Toc199708765)

[Figura 5 - Tela do carrinho 13](#_Toc199708766)

[Figura 6 - Tela de Formas de Pagamento 13](#_Toc199708767)

[Figura 7 - Tela de Formas de Pagamento 2 14](#_Toc199708768)

[Figura 8 - Tela Status do Pedido 14](#_Toc199708769)

[Figura 9 - Tela de Avaliação de Pedido 15](#_Toc199708770)

[Figura 10 - Tela de Edição de Cliente 16](#_Toc199708771)

[Figura 11 - Tela Inicial Fornecedor 17](#_Toc199708772)

[Figura 12 - TELA DE LOGIN PARA FORNECEDOR OU CLIENTE 17](#_Toc199708773)

[Figura 13 - Tela de Cadastro de Fornecedor 18](#_Toc199708774)

[Figura 14 - Tela de Listagem de Produto 18](#_Toc199708775)

[Figura 15 - Tela de Cadastro de Produto 19](#_Toc199708776)

[Figura 16 - Tela de Edição de Produto 19](#_Toc199708777)

[Figura 17 - Tela de Listagem de Pedidos para Fornecedor 20](#_Toc199708778)

[Figura 18 - Tela de Edição de Dados do Fornecedor 20](#_Toc199708779)

[Figura 19 - Modelagem do Banco de Dados 21](#_Toc199708780)

[Figura 20 - Tela Backend Pedido Controller 24](#_Toc199708781)

[Figura 21 - Tela Backend Pedido Model 25](#_Toc199708782)

[Figura 22 - Tela Backend Pedido Repository 26](#_Toc199708783)

[Figura 23 - Tela Backend Pedido DTO 26](#_Toc199708784)

[Figura 24 - Tela Backend Pedido Response DTO 27](#_Toc199708785)

[Figura 25 - Tela Backend Pedido Service 29](#_Toc199708786)

Sumário

[1. INTRODUÇÃO 5](#_Toc199709458)

[1.1 Contextualização 5](#_Toc199709459)

[1.2 Objetivos 5](#_Toc199709460)

[1.3 Justificativa 6](#_Toc199709461)

[2. FUNDAMENTAÇÃO TEÓRICA 7](#_Toc199709462)

[2.1 Sistemas de Delivery Online 7](#_Toc199709463)

[2.2 Arquitetura Cliente-Servidor 7](#_Toc199709464)

[2.3 Tecnologias Utilizadas 7](#_Toc199709465)

[2.3.1 Backend 7](#_Toc199709466)

[2.3.2 Frontend 8](#_Toc199709467)

[3. DESCRIÇÃO DO SISTEMA 9](#_Toc199709468)

[3.1 Visão Geral 9](#_Toc199709469)

[3.2 Arquitetura 9](#_Toc199709470)

[3.3 Funcionalidades 10](#_Toc199709471)

[3.3.1 Área do Cliente 10](#_Toc199709472)

[3.3.2 Área do Fornecedor 16](#_Toc199709473)

[3.4 Modelo de Dados 20](#_Toc199709474)

[4. IMPLEMENTAÇÃO 23](#_Toc199709475)

[4.1 Backend 23](#_Toc199709476)

[4.1.1 Estrutura do Projeto 23](#_Toc199709477)

[4.1.2 Controllers 23](#_Toc199709478)

[4.1.3 Models 24](#_Toc199709479)

[4.1.5 DTOs 26](#_Toc199709480)

[4.2 Frontend 30](#_Toc199709481)

[4.2.1 Estrutura do Projeto 30](#_Toc199709482)

[4.2.2 Interfaces 30](#_Toc199709483)

[4.2.3 JavaScript/TypeScript 30](#_Toc199709484)

[4.2.4 Estilos 31](#_Toc199709485)

[5. CONCLUSÃO 32](#_Toc199709486)

[5.1 Considerações Finais 32](#_Toc199709487)

# 1. INTRODUÇÃO

## 1.1 Contextualização

O setor de serviços de entrega por aplicativo tem se consolidado como uma das principais formas de consumo alimentício no Brasil, impulsionado pela praticidade, velocidade e ampla disponibilidade de restaurantes. Grandes plataformas, como o iFood, popularizaram essa modalidade, mas muitos estabelecimentos ainda enfrentam desafios para ingressar nesse mercado devido à falta de soluções acessíveis e personalizadas. Pequenos e médios empreendedores, em especial, carecem de sistemas próprios que integrem seus processos de pedidos, gerenciamento de produtos e atendimento ao cliente.

Neste contexto, o sistema de delivery desenvolvido neste projeto surge como uma solução responsiva e escalável, inspirada nos modelos existentes, mas com foco na independência tecnológica de restaurantes locais. A aplicação contempla tanto o lado do cliente quanto o do fornecedor, oferecendo uma experiência digital completa desde o cadastro até a entrega e avaliação do pedido.

## 1.2 Objetivos

O objetivo geral deste trabalho é documentar e analisar o desenvolvimento de uma plataforma de delivery responsiva que permita a integração eficiente entre consumidores e estabelecimentos, simulando o funcionamento de grandes sistemas de pedidos online.

Os objetivos específicos incluem:

* Descrever a arquitetura e os componentes técnicos da aplicação;
* Apresentar as funcionalidades implementadas para os perfis de cliente e fornecedor;
* Detalhar os modelos de dados e o fluxo de comunicação entre frontend e backend;
* Avaliar os aspectos visuais, responsividade e usabilidade do sistema;
* Identificar limitações atuais e propor melhorias para versões futuras.

## 1.3 Justificativa

A criação de um sistema de delivery próprio justifica-se pela necessidade de:

* Permitir que estabelecimentos alimentícios ofereçam seus serviços digitalmente sem depender de grandes plataformas;
* Automatizar o processo de pedidos, controle de produtos e avaliação de serviços;
* Melhorar o atendimento ao cliente por meio de uma plataforma moderna, responsiva e intuitiva;
* Proporcionar maior autonomia e visibilidade para negócios locais;
* Estimular o aprendizado técnico prático em desenvolvimento web full-stack, aplicando conceitos de autenticação, consumo de APIs, controle de estado e arquitetura de sistemas.

O sistema busca suprir essas necessidades de forma integrada, oferecendo uma solução tecnológica viável, com potencial de aplicação real e personalização para diferentes contextos de negócio.

# 2. FUNDAMENTAÇÃO TEÓRICA

## 2.1 Sistemas de Delivery Online

Os sistemas de delivery online se consolidaram como ferramentas essenciais para o setor alimentício nos últimos anos, especialmente com a popularização de plataformas como iFood, Uber Eats e Rappi. Esses sistemas têm como principal objetivo conectar clientes a estabelecimentos de forma digital, permitindo pedidos rápidos, acompanhamento em tempo real e avaliação da experiência de consumo. Segundo Gomes (2021), um sistema de delivery eficiente deve contemplar, no mínimo, os seguintes módulos: cadastro de usuários e fornecedores, gerenciamento de produtos, carrinho de compras, controle de status de pedidos, avaliações e formas de pagamento.

No desenvolvimento da plataforma de delivery apresentada neste projeto, todos esses aspectos foram considerados, com foco especial na responsividade, usabilidade e separação clara de perfis de usuário (cliente e fornecedor), permitindo que ambos os lados do processo tenham controle e autonomia sobre suas interações na aplicação.

## 2.2 Arquitetura Cliente-Servidor

A aplicação foi desenvolvida com base na arquitetura cliente-servidor, modelo amplamente adotado em sistemas web, no qual o cliente (frontend) é responsável pela interface com o usuário e o servidor (backend) gerencia a lógica de negócio e o acesso aos dados. Segundo Tanenbaum e Van Steen (2007), esse tipo de arquitetura apresenta benefícios como escalabilidade, manutenção simplificada e centralização dos dados.

No caso desta plataforma, o **servidor (backend)** foi construído com a linguagem Java utilizando o framework Spring Boot, sendo responsável por operações como autenticação com JWT, gerenciamento de entidades (clientes, fornecedores, produtos, pedidos), e comunicação com o banco de dados relacional. O **cliente (frontend)** foi desenvolvido utilizando tecnologias modernas de construção de interfaces web, sendo responsável por exibir os dados ao usuário, enviar requisições ao servidor e garantir uma experiência fluida tanto em dispositivos móveis quanto em desktop.

## 2.3 Tecnologias Utilizadas

### 2.3.1 Backend

O backend do sistema foi desenvolvido com as seguintes tecnologias:

* **Java**: Linguagem orientada a objetos utilizada pela sua robustez e amplo uso em aplicações empresariais.
* **Spring Boot**: Framework que simplifica o desenvolvimento de aplicações Java, oferecendo configuração automática, integração com bibliotecas e facilidade na criação de APIs REST.
* **Spring Security + JWT**: Módulo de segurança utilizado para autenticação e autorização de usuários, garantindo acesso restrito a determinadas funcionalidades.
* **Spring Data JPA**: Abstração sobre o JPA (Java Persistence API) que facilita a criação de repositórios e operações de banco de dados.
* **Hibernate**: Framework ORM utilizado para mapear objetos Java em tabelas relacionais do banco de dados.
* **MySQL**: Sistema de gerenciamento de banco de dados relacional utilizado para persistência de dados do sistema.

### 2.3.2 Frontend

O frontend do sistema foi desenvolvido utilizando uma combinação de tecnologias modernas:

* **HTML5**: Linguagem de marcação utilizada para a estruturação dos elementos da página.
* **CSS3**: Linguagem de estilo utilizada para definir a aparência visual e responsividade da interface.
* **JavaScript**: Linguagem de programação utilizada para implementar funcionalidades interativas.
* **TypeScript**: Superset de JavaScript que adiciona tipagem estática ao código, aumentando a robustez e legibilidade do frontend.
* **Vite**: Ferramenta de build moderna utilizada para desenvolvimento rápido e eficiente de aplicações front-end.
* **Fetch API**: Interface nativa do JavaScript utilizada para realizar requisições HTTP entre frontend e backend.
* **Font Awesome**: Biblioteca de ícones vetoriais amplamente utilizada para enriquecer a interface gráfica.

# 3. DESCRIÇÃO DO SISTEMA

## 3.1 Visão Geral

A plataforma de delivery desenvolvida neste projeto foi criada com o objetivo de oferecer uma solução digital completa para o atendimento de pedidos online, inspirada nos modelos adotados por aplicativos como o iFood. O sistema permite a interação entre dois perfis de usuários: **clientes**, que realizam pedidos, e **fornecedores (restaurantes)**, que cadastram seus produtos e gerenciam os pedidos recebidos.

A aplicação conta com uma interface web responsiva, compatível com dispositivos móveis e desktops, possibilitando o uso prático e fluido em diferentes contextos. O sistema foi estruturado com foco na clareza das funcionalidades, na separação das responsabilidades entre as camadas e na entrega de uma experiência simples, intuitiva e eficaz tanto para consumidores quanto para estabelecimentos.

## 3.2 Arquitetura

A arquitetura do sistema segue o padrão **MVC (Model-View-Controller)**, promovendo a separação entre apresentação, lógica de negócio e acesso a dados.

O sistema é composto pelos seguintes componentes principais:

* **Frontend (Camada de Apresentação):** Desenvolvido com HTML5, CSS3, JavaScript e TypeScript utilizando Vite, responsável pela interface com o usuário.
* **Backend (Camada de Lógica de Negócio):** Desenvolvido com Java e Spring Boot, responsável pelas regras de negócio, autenticação e APIs REST.
* **Repositórios (Camada de Acesso a Dados):** Utiliza Spring Data JPA para facilitar a comunicação com o banco de dados.
* **Banco de Dados:** Utiliza MySQL para o armazenamento estruturado e persistente dos dados.

A comunicação entre o frontend e o backend é realizada por meio de uma **API RESTful**, garantindo desacoplamento entre as camadas e facilitando a manutenção e expansão futura da aplicação.

## 3.3 Funcionalidades

A plataforma de delivery implementa as seguintes funcionalidades principais:

### 3.3.1 Área do Cliente

* Cadastro e login e tela inicial de cliente;

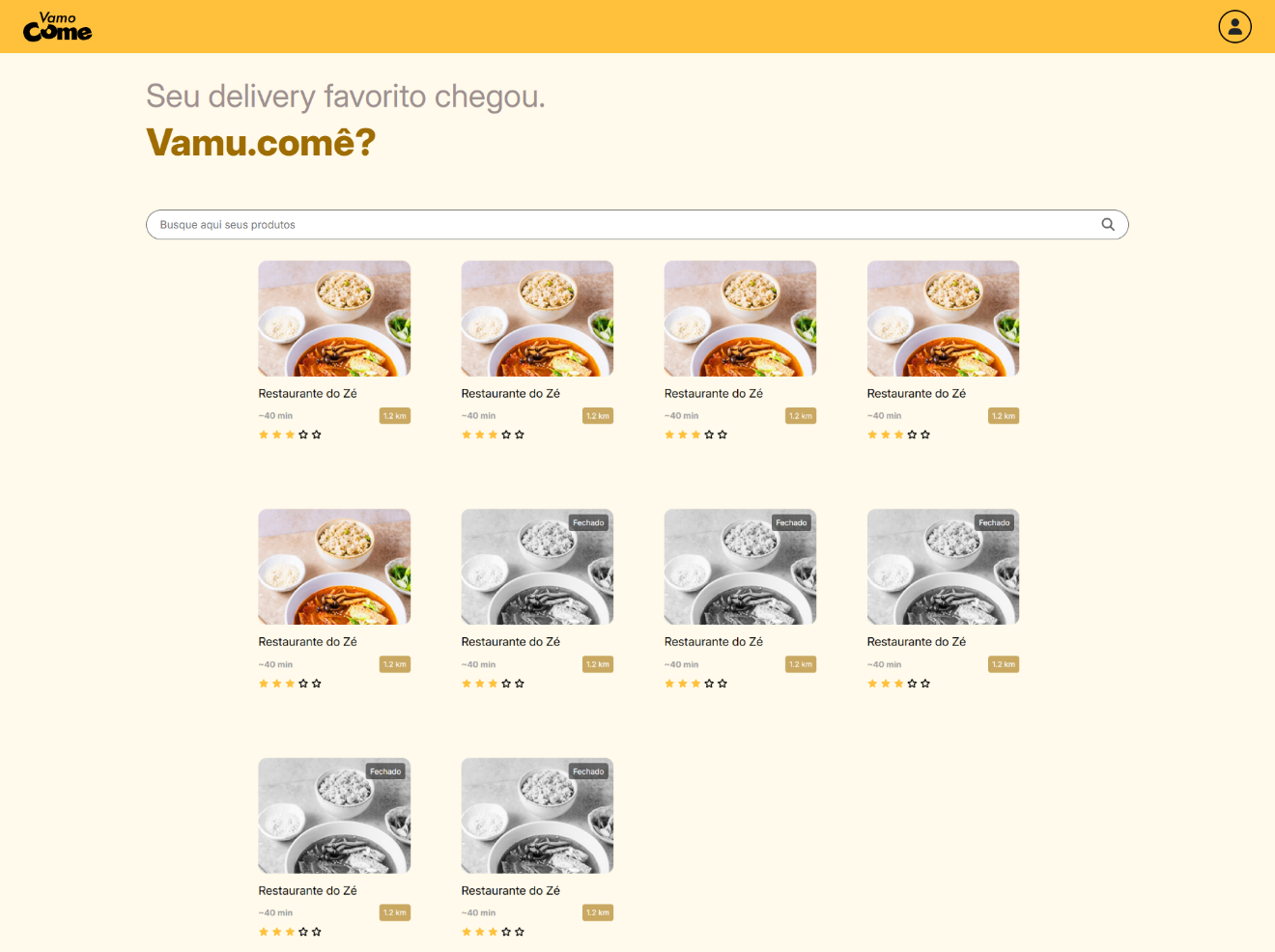


Figura 1 - Tela Inicial Cliente



Figura 2 - Tela de login para fornecedor ou cliente

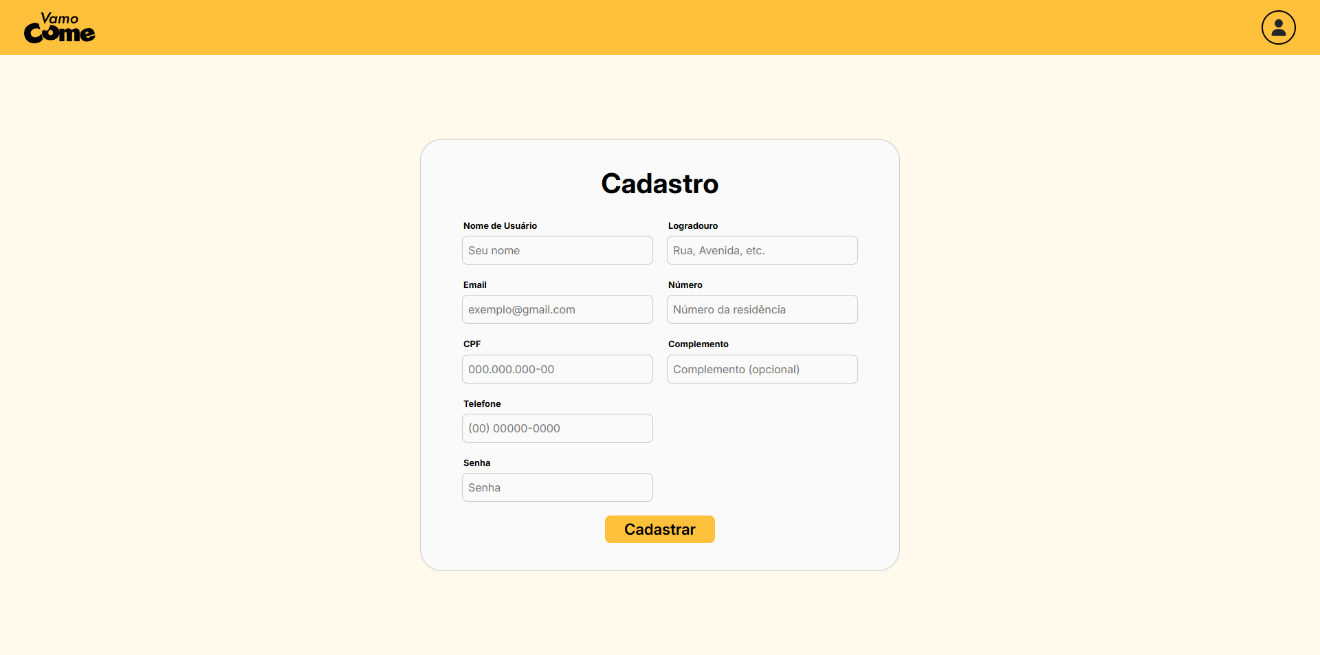


Figura 3 - Tela de Cadastro de Cliente

* Visualização de restaurantes e seus produtos;

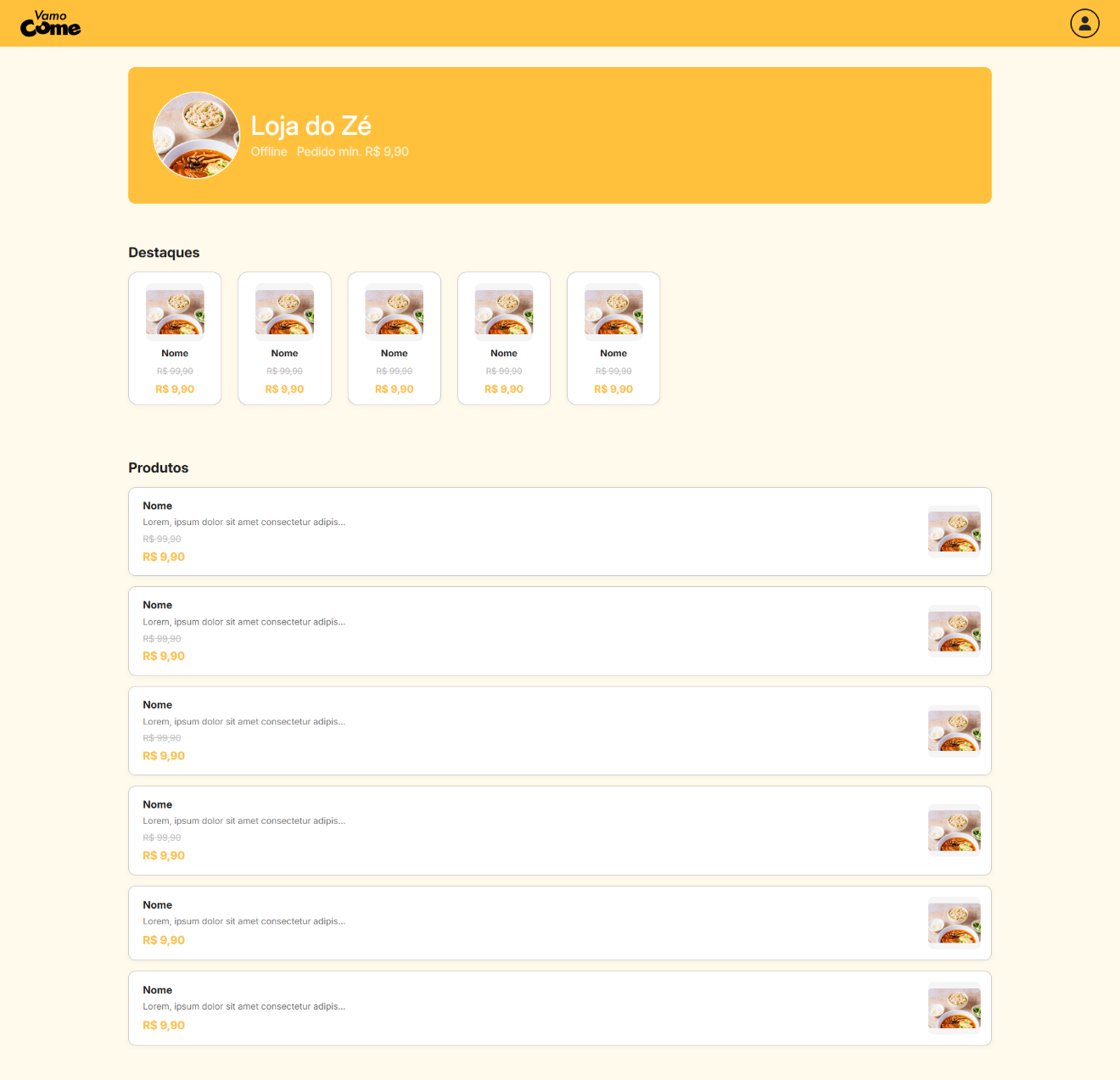


Figura 4 - Tela de Visualização de Fornecedor e seus Produtos

* Carrinho;

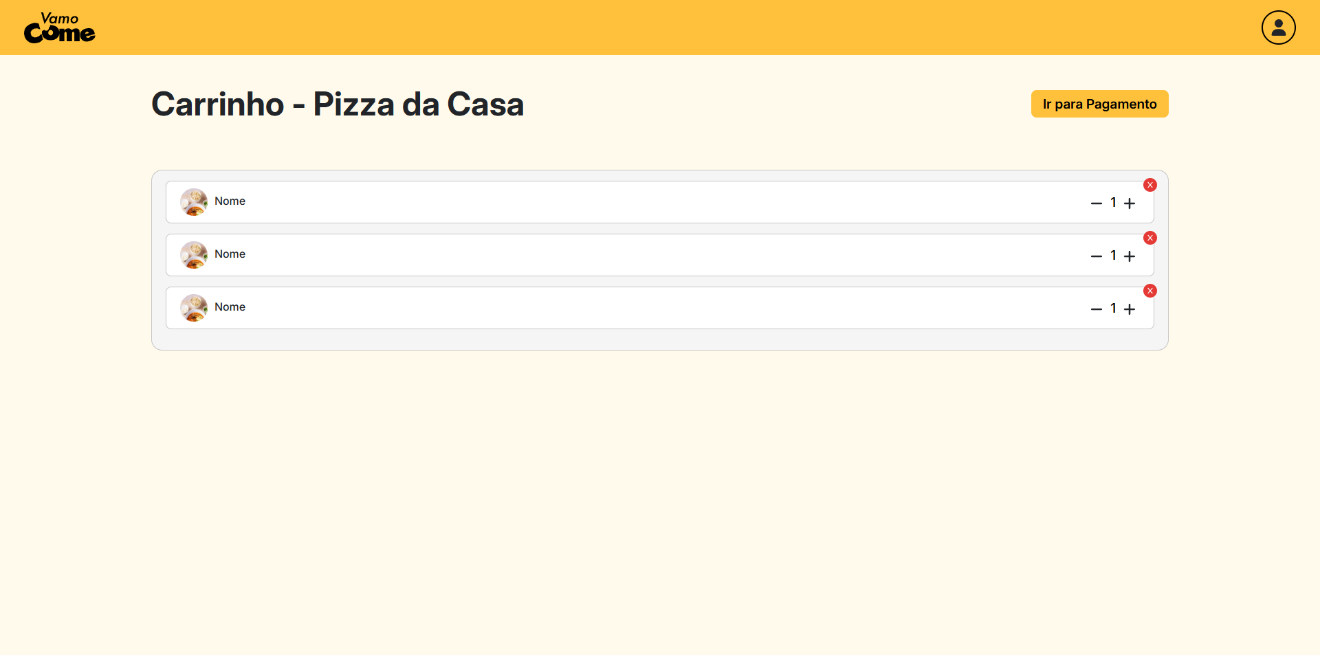


Figura 5 - Tela do carrinho

* Finalização de pedidos com escolha de forma de pagamento;

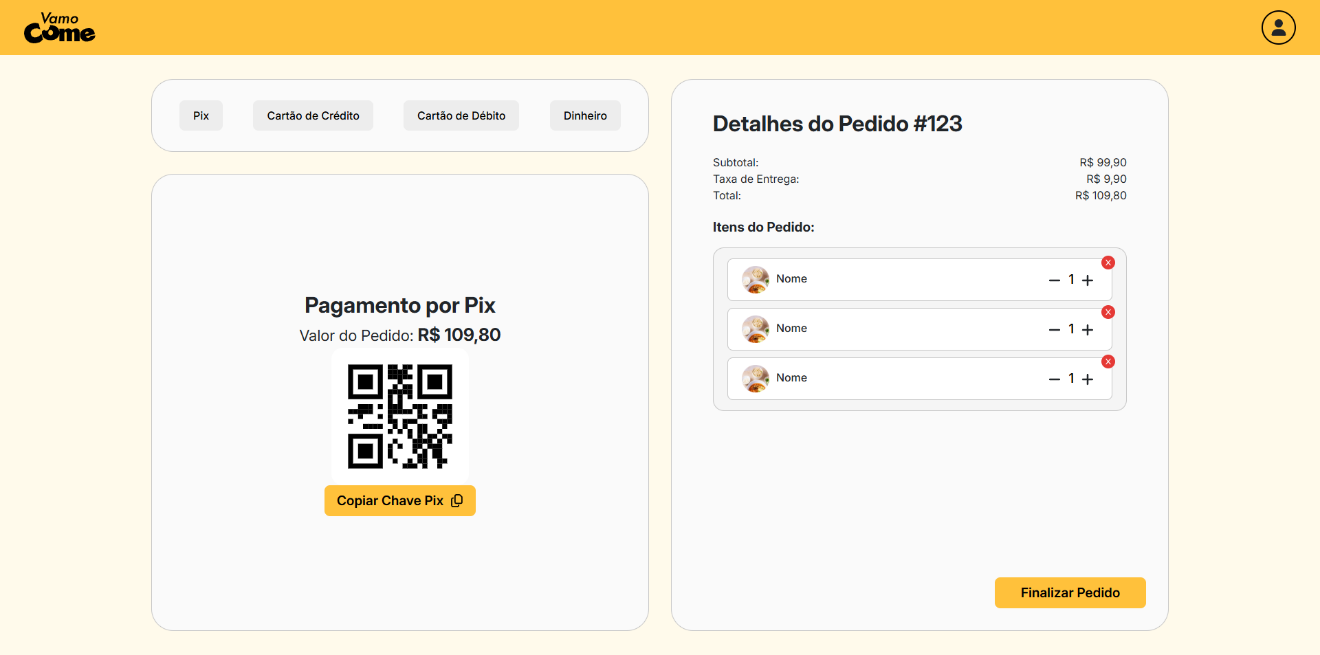


Figura 6 - Tela de Formas de Pagamento

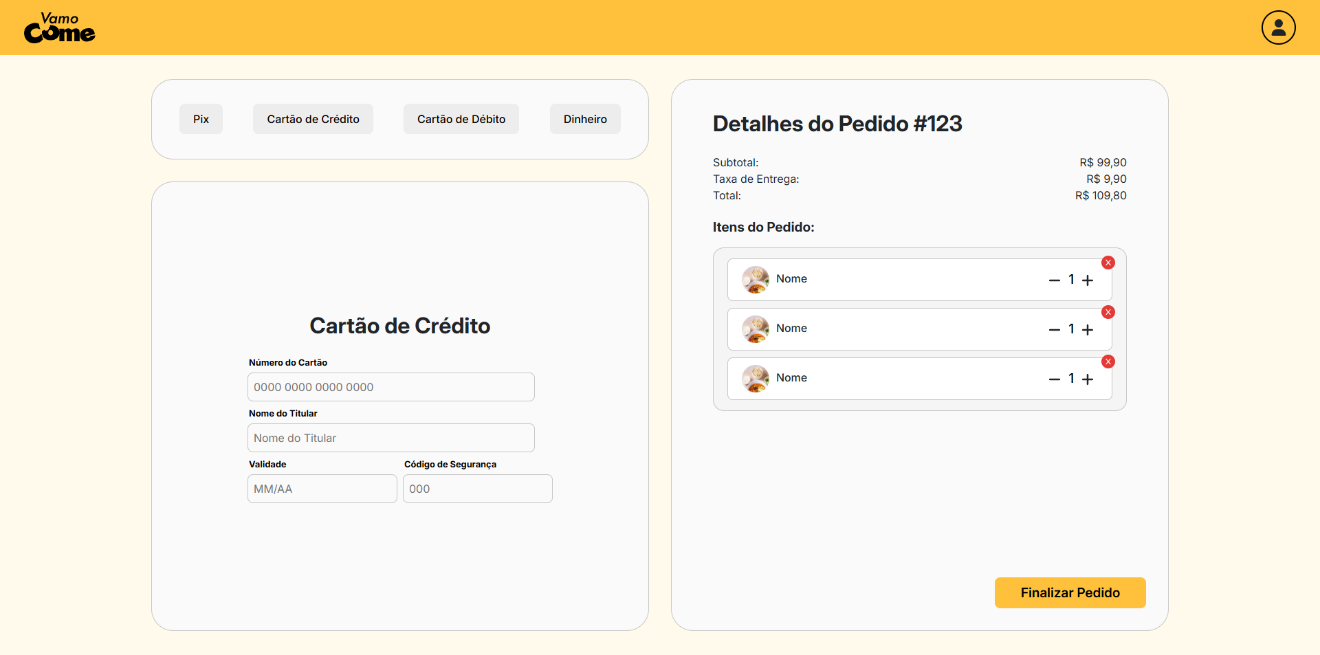


Figura 7 - Tela de Formas de Pagamento 2

* Acompanhamento do status do pedido em tempo real;

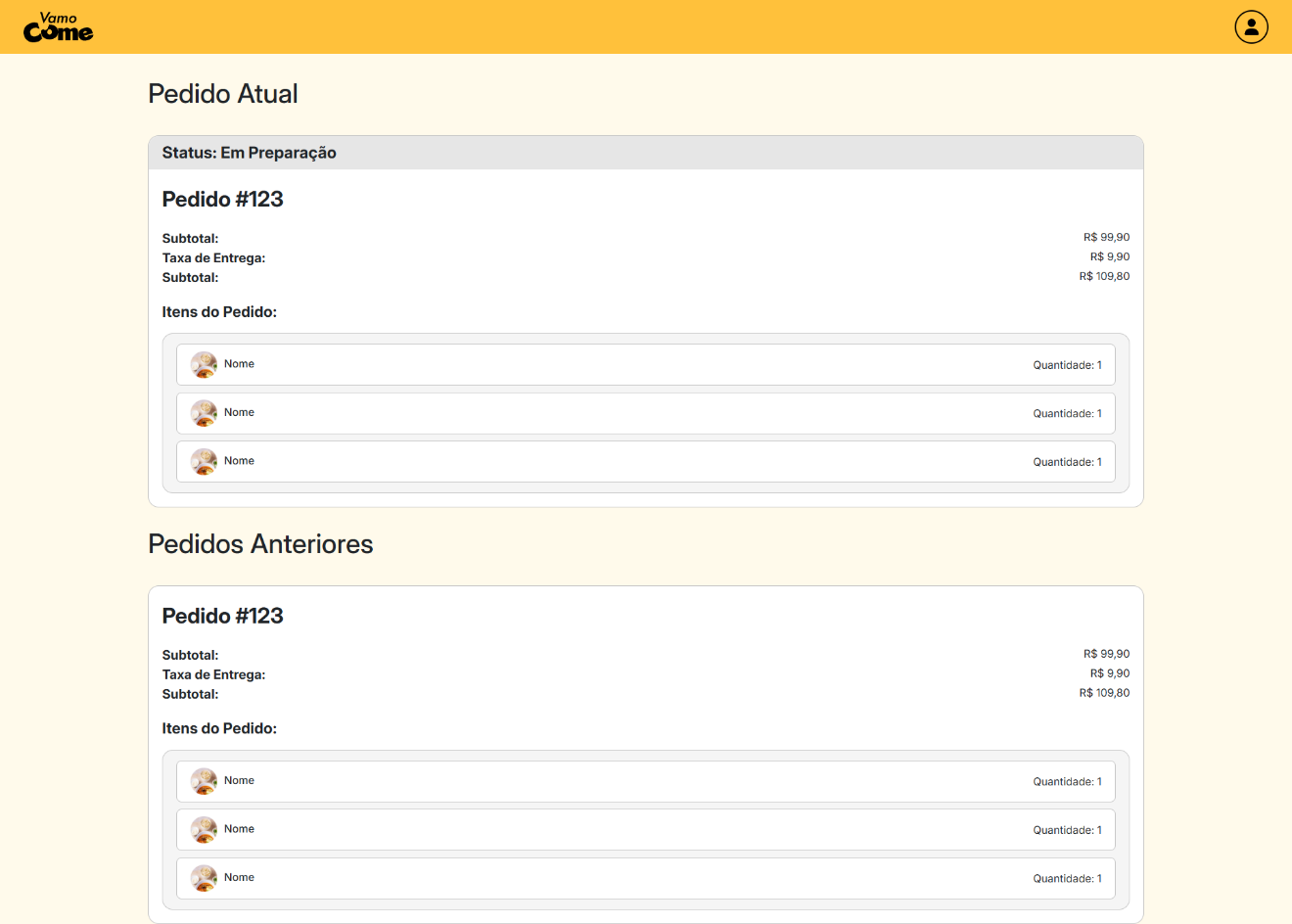


Figura 8 - Tela Status do Pedido

* Avaliação do pedido;

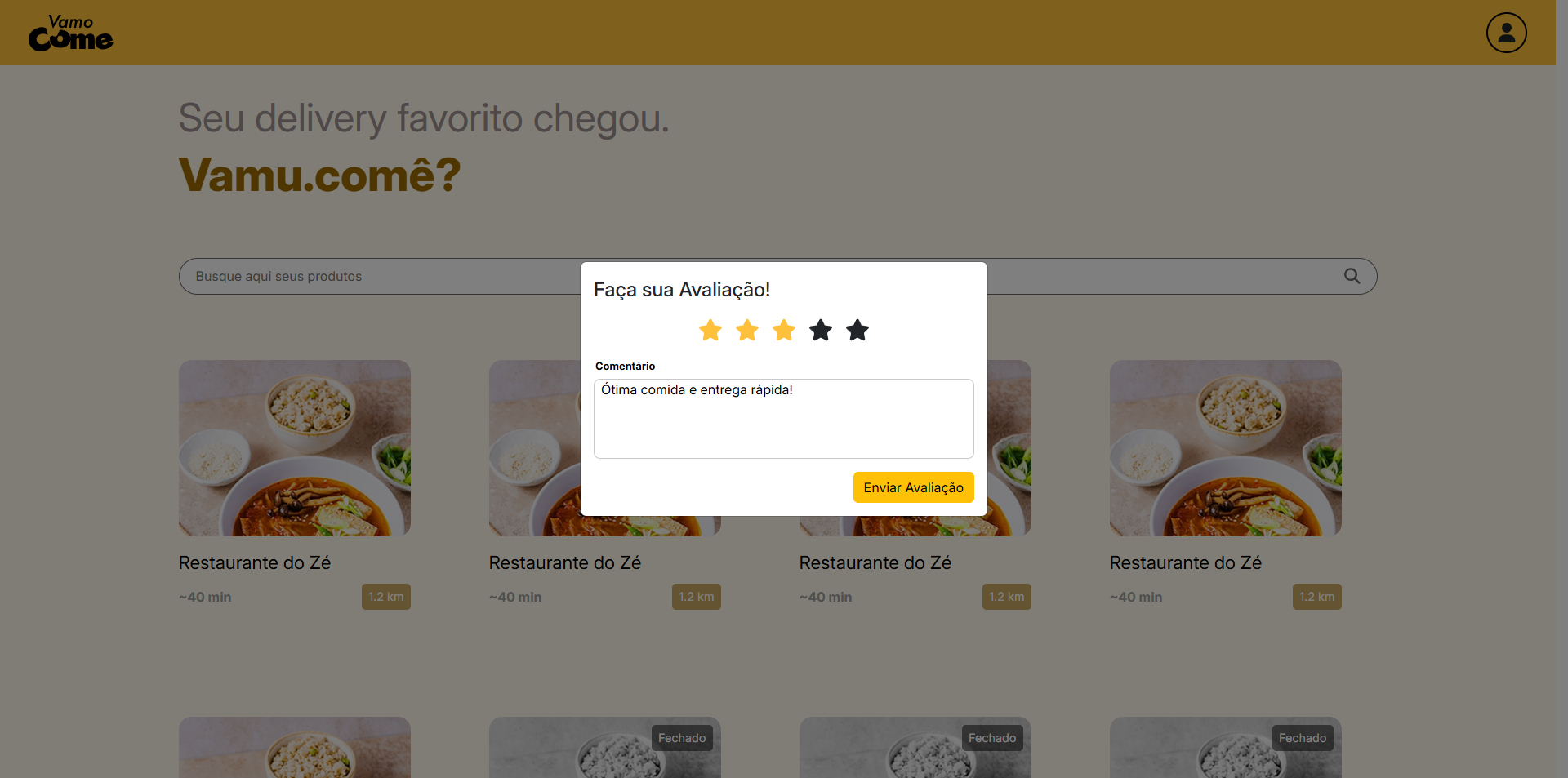


Figura 9 - Tela de Avaliação de Pedido

* Edição de perfil e endereço.

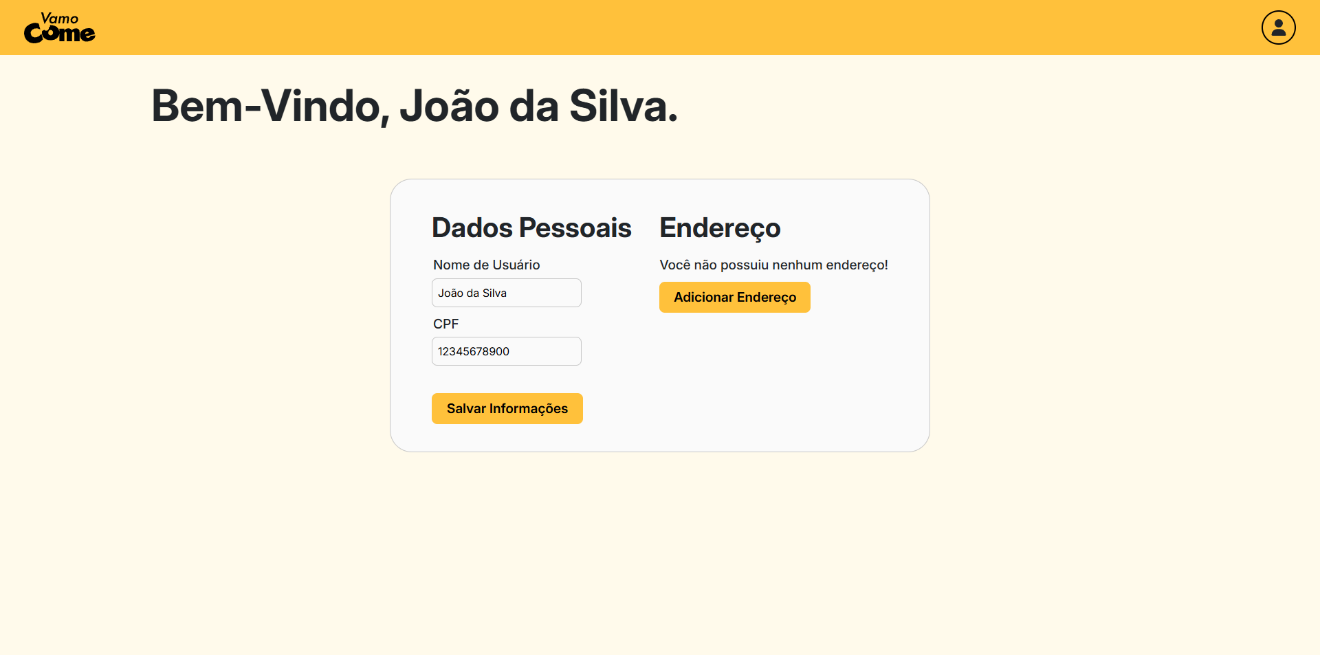


Figura 10 - Tela de Edição de Cliente

### 3.3.2 Área do Fornecedor

* Cadastro e login e tela inicial de fornecedor;

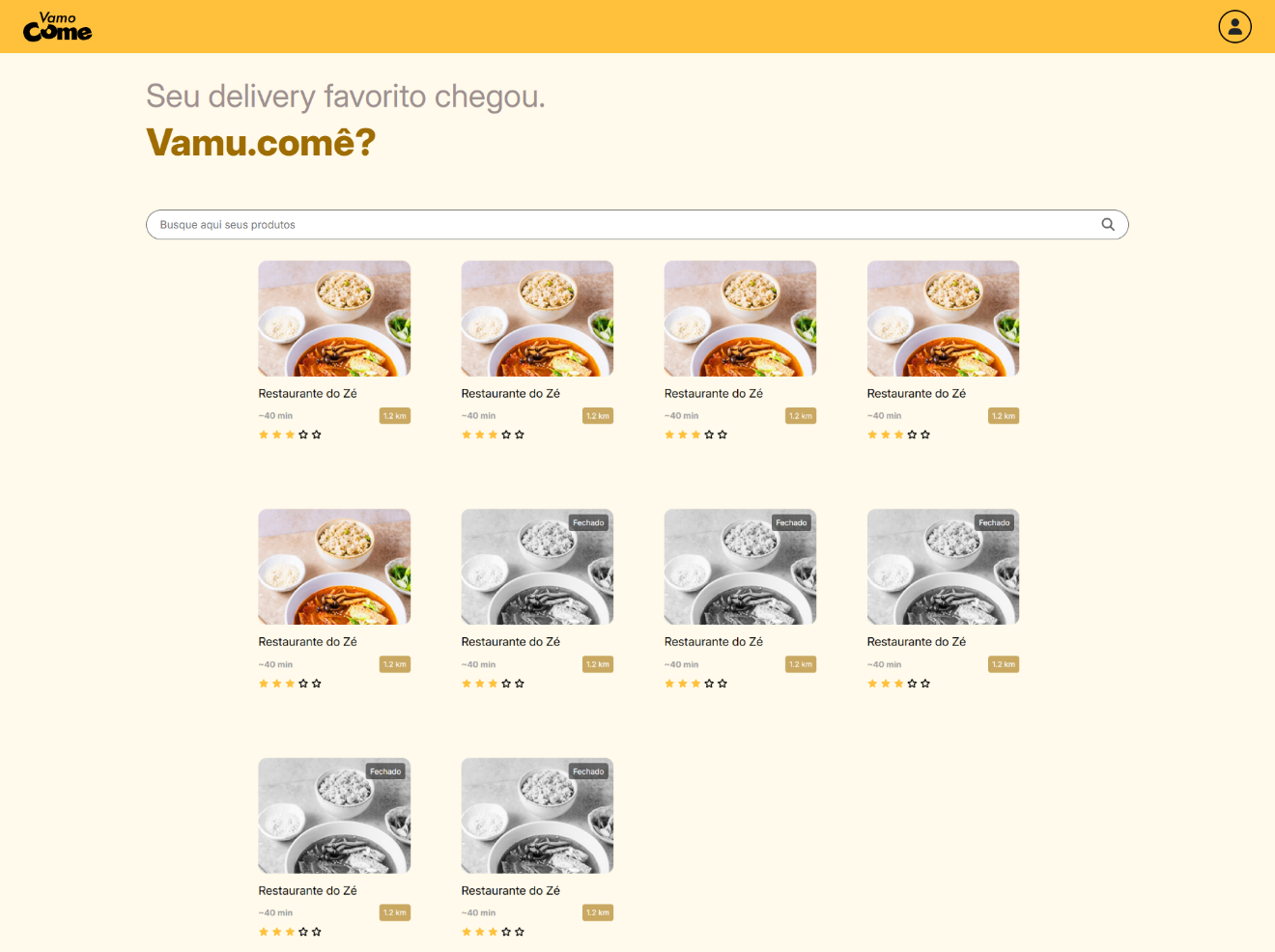


Figura 11 - Tela Inicial Fornecedor



Figura 12 - TELA DE LOGIN PARA FORNECEDOR OU CLIENTE

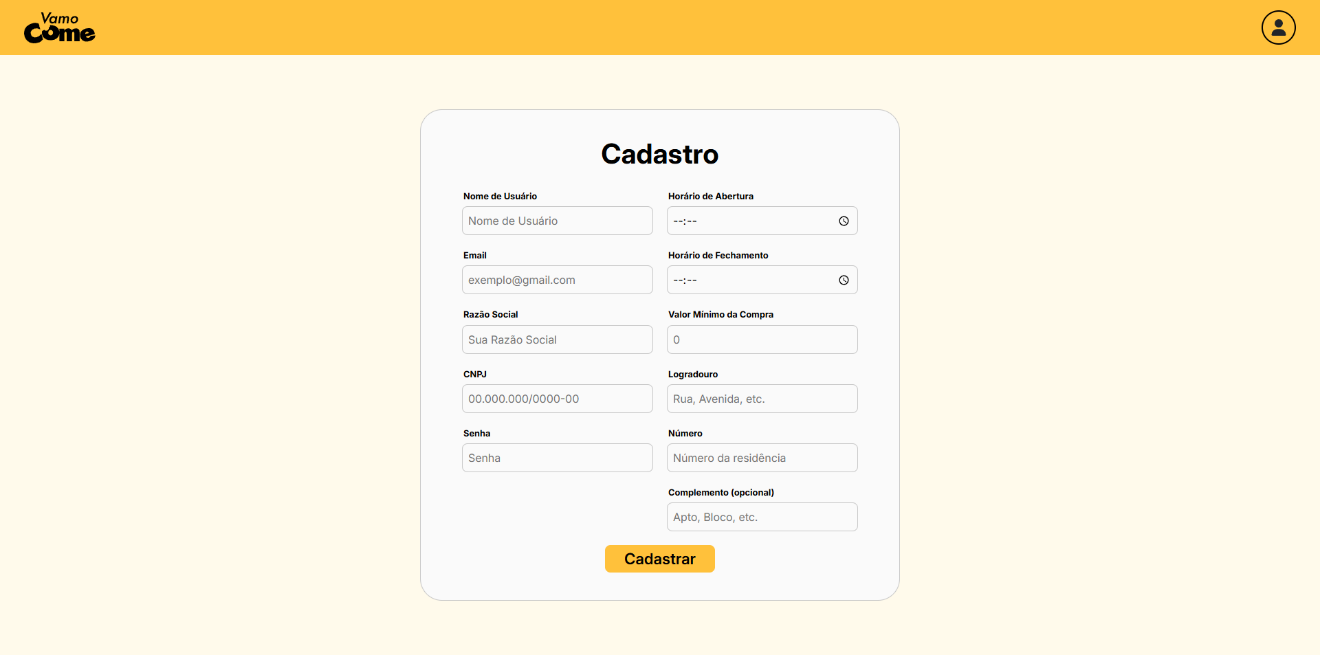


Figura 13 - Tela de Cadastro de Fornecedor

* Cadastro, edição e exclusão de produtos no cardápio;

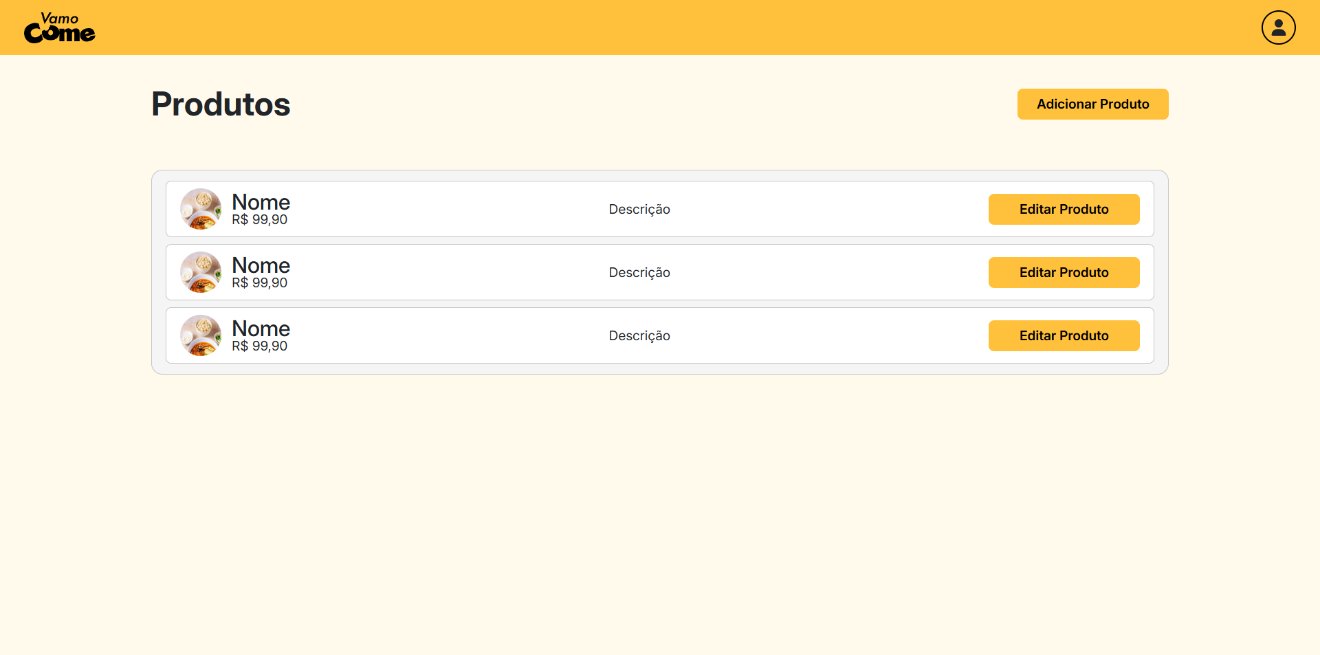


Figura 14 - Tela de Listagem de Produto

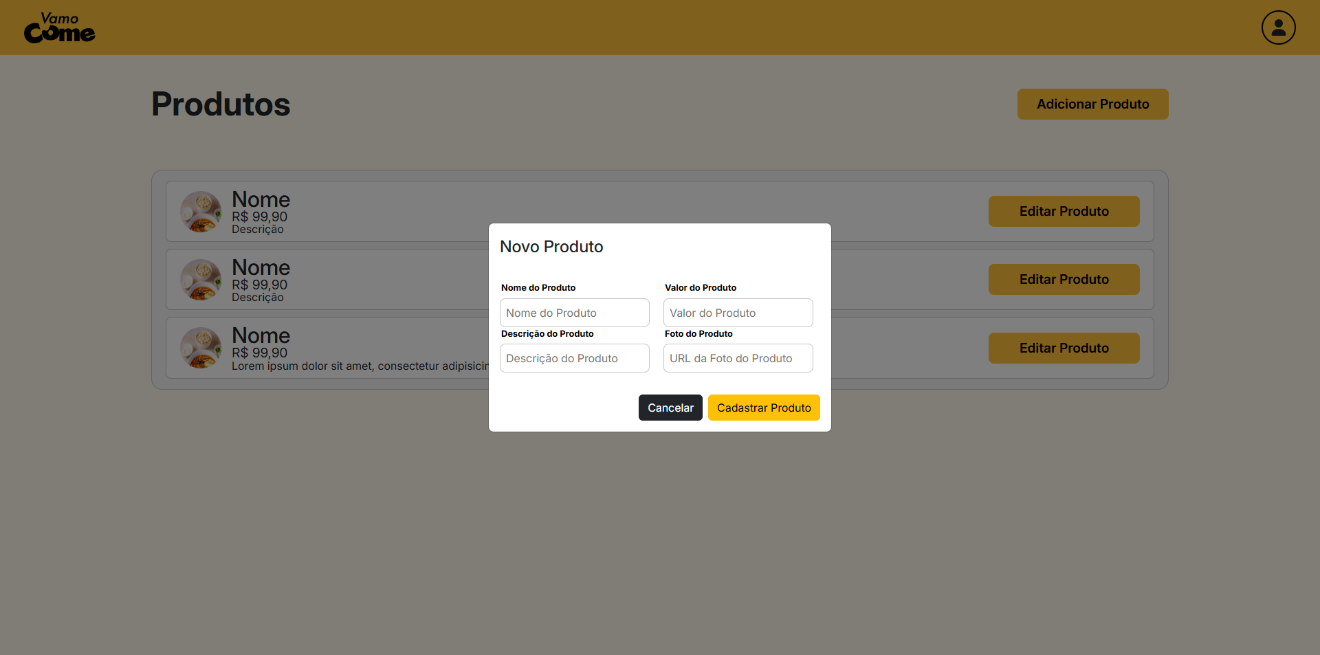


Figura 15 - Tela de Cadastro de Produto

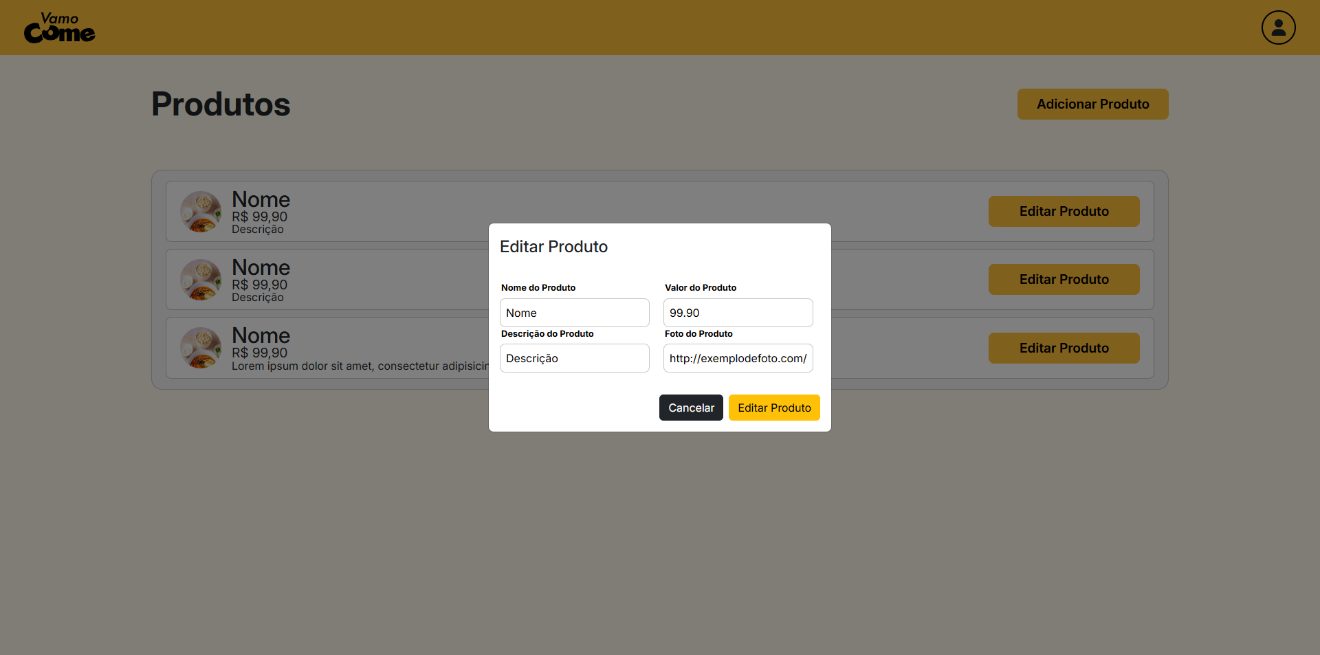


Figura 16 - Tela de Edição de Produto

* Visualização dos pedidos recebidos;

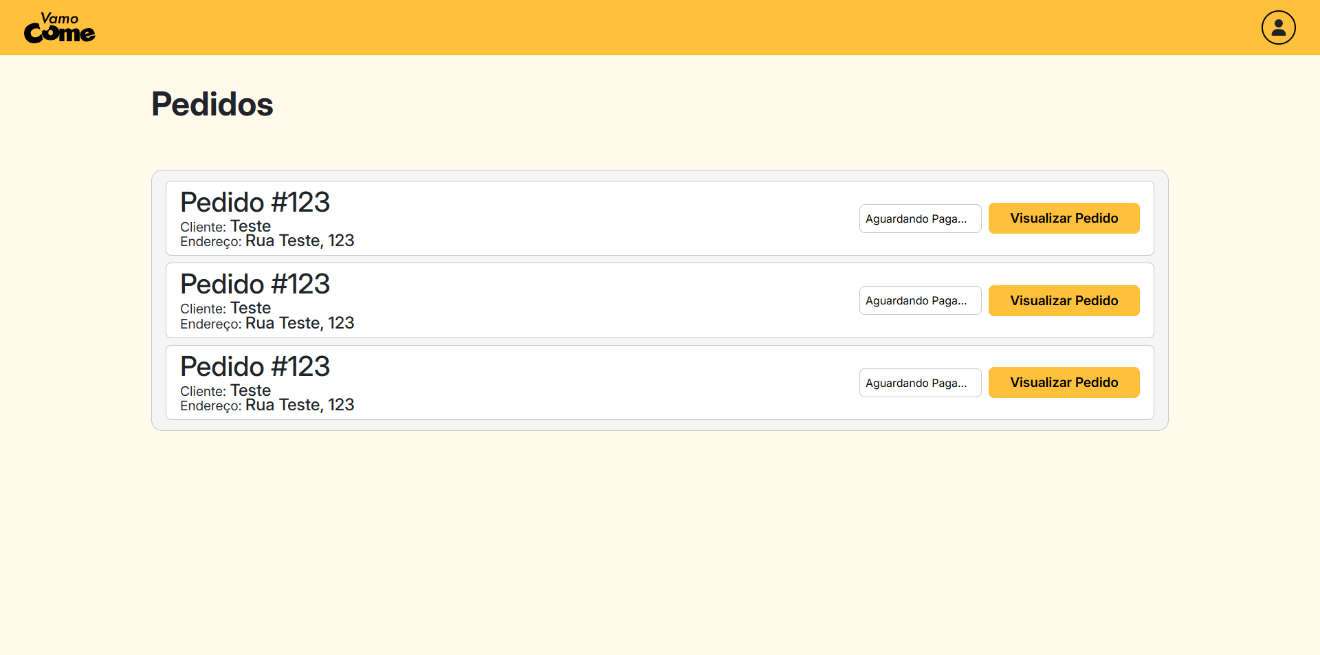


Figura 17 - Tela de Listagem de Pedidos para Fornecedor

* Edição de dados do estabelecimento.

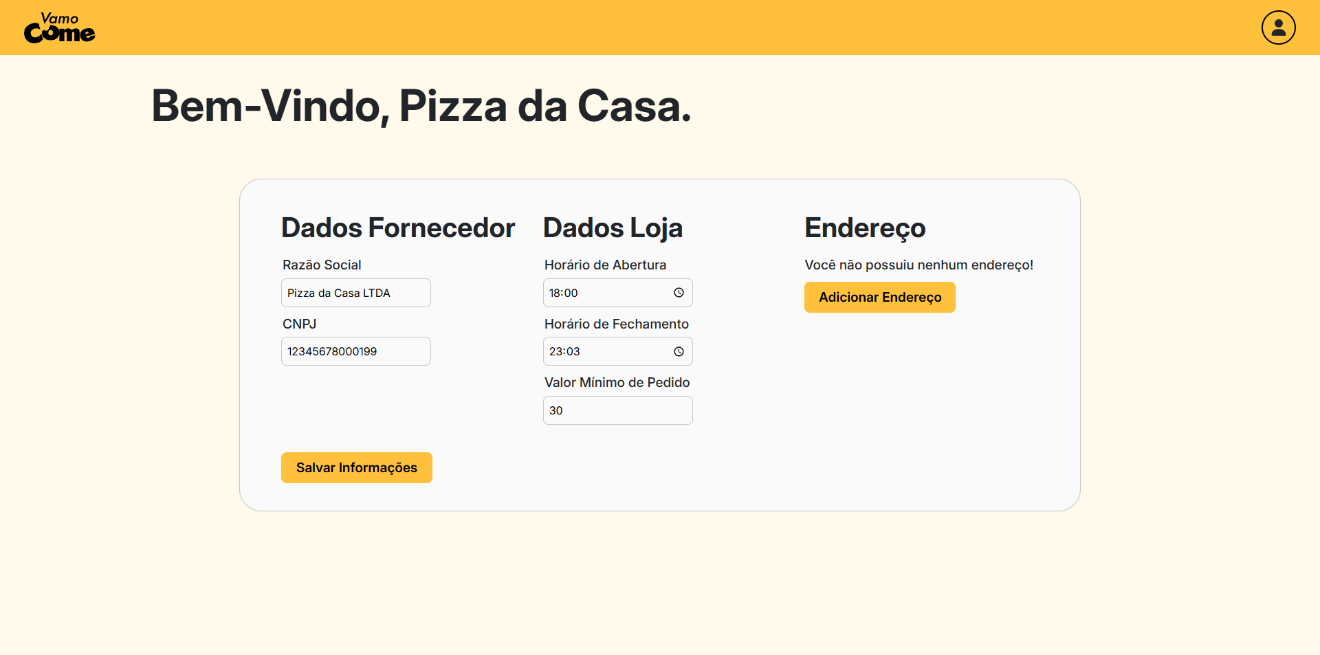


Figura 18 - Tela de Edição de Dados do Fornecedor

## 3.4 Modelo de Dados

O modelo de dados do sistema é composto por seis entidades principais: **Cliente**, **Fornecedor**, **Produto**, **Pedido**, **ItemPedido** (associativa) e **Avaliação**. A Figura X apresenta o diagrama de entidade-relacionamento do sistema.

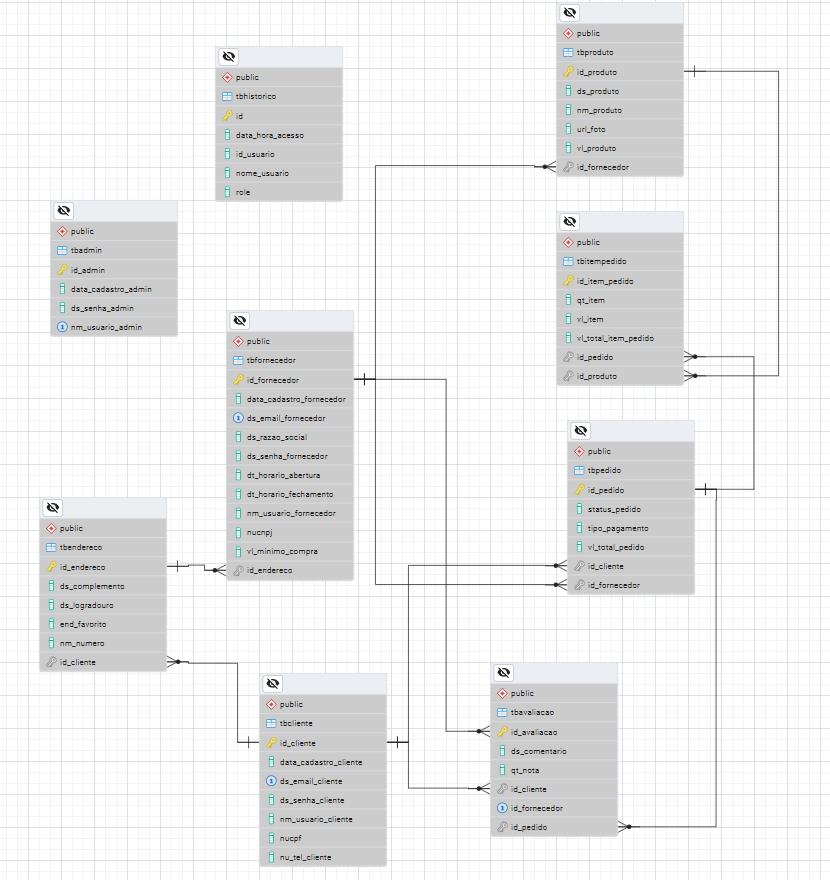


Figura 19 - Modelagem do Banco de Dados

As principais entidades e seus relacionamentos são:

* **Cliente:** Armazena informações dos consumidores da plataforma, incluindo nome, e-mail, senha criptografada, número de telefone e endereço completo.
* **Fornecedor:** Representa os estabelecimentos cadastrados na plataforma. Armazena dados como razão social, nome do restaurante, categoria, contato, endereço e credenciais de acesso.
* **Produto:** Contém os itens ofertados por cada fornecedor, com informações como nome, descrição, preço, imagem e vínculo com o restaurante responsável.
* **Pedido:** Registra cada solicitação de compra realizada por um cliente. Inclui informações como data e hora, status (ex: Recebido, Em Preparo, Entregue), forma de pagamento e referência ao cliente e ao fornecedor.
* **ItemPedido:** Entidade associativa entre Pedido e Produto, que registra os produtos escolhidos em cada pedido, incluindo quantidade e subtotal.
* **Avaliação:** Permite que o cliente avalie o pedido após a entrega. Armazena nota (ex: 1 a 5 estrelas), comentário, data da avaliação e está vinculada ao pedido correspondente.

# 4. IMPLEMENTAÇÃO

## 4.1 Backend

### 4.1.1 Estrutura do Projeto

O backend do sistema foi implementado utilizando **Spring Boot**, com a estrutura organizada em pacotes conforme as boas práticas de separação de responsabilidades do framework:

* com.example.login\_auth\_api: Pacote raiz do projeto, contendo a classe principal de inicialização (LoginAuthApiApplication.java).
* com.example.login\_auth\_api.controllers: Contém os controladores REST que expõem as APIs do sistema.
* com.example.login\_auth\_api.controllers.auth: Reúne os controladores responsáveis pelas rotas de autenticação para diferentes perfis de usuário (Cliente, Fornecedor e Admin).
* com.example.login\_auth\_api.domain: Contém as classes de modelo (entidades) que representam os objetos persistentes no banco de dados.
* com.example.login\_auth\_api.repositories: Contém as interfaces de repositório que herdam de JpaRepository, permitindo acesso direto aos dados.
* com.example.login\_auth\_api.dto: Agrupa os DTOs utilizados para comunicação entre frontend e backend, organizados em request e response.
* com.example.login\_auth\_api.service: Contém os serviços que concentram a lógica de negócio das operações do sistema.
* com.example.login\_auth\_api.infra: Contém as configurações da aplicação, incluindo segurança, tratamento de exceções e conexão com o banco de dados.

### 4.1.2 Controllers

Os controllers implementam a API REST do sistema, expondo endpoints para operações **CRUD** (Create, Read, Update, Delete) e outras operações específicas, como autenticação, atualização de status e filtros.

Principais controllers:

* ClienteController: Gerencia cadastro, atualização, visualização e exclusão de clientes.
* FornecedorController: Responsável pelo controle de dados dos fornecedores e seus produtos.
* ProdutoController: Permite a criação, edição, listagem e exclusão de produtos vinculados a um fornecedor.
* PedidoController: Lida com a criação de pedidos, alteração de status, e exibição dos pedidos por cliente ou fornecedor.

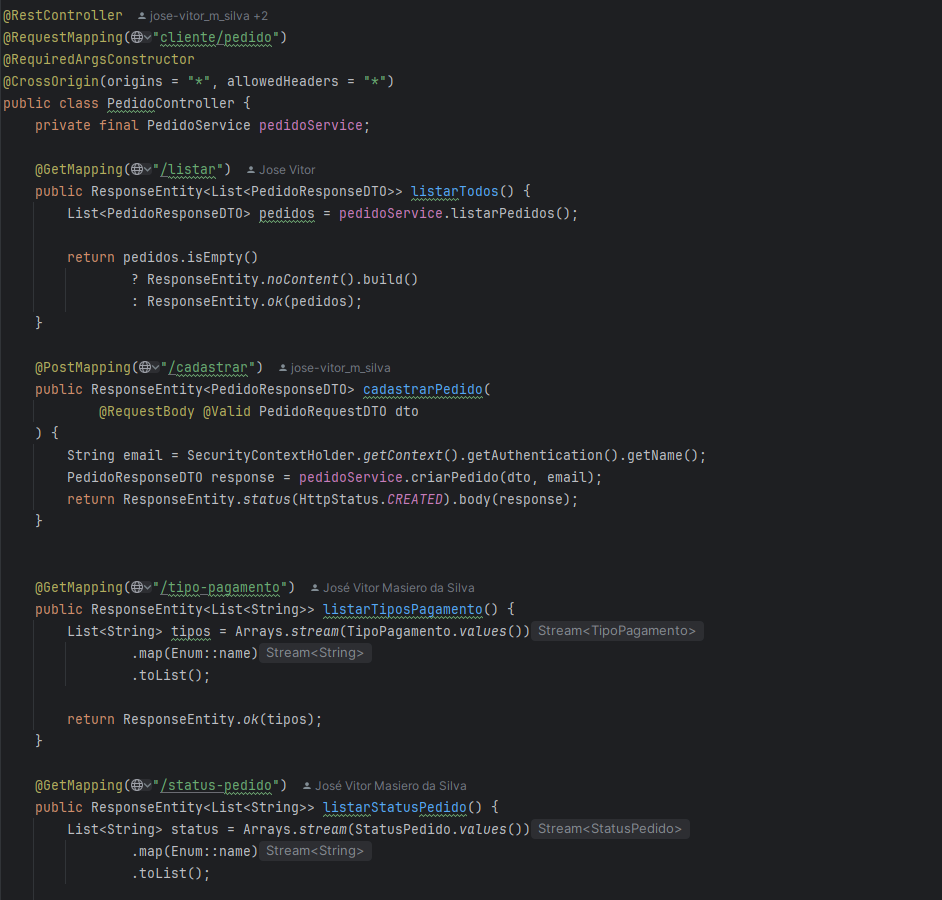


Figura 20 - Tela Backend Pedido Controller

* AvaliacaoController: Trata do registro de avaliações feitas pelos clientes após o recebimento do pedido.
* RelatorioController: Fornece informações para análise administrativa do sistema.
* AdminAuthController, ClienteAuthController, FornecedorAuthController: Responsáveis pelos fluxos de login, registro e recuperação de senha, conforme o tipo de usuário.

### 4.1.3 Models

As entidades (models) representam os dados do sistema e são mapeadas com anotações JPA para integração com o banco de dados. As principais são:

* Cliente: Representa o usuário consumidor, com dados pessoais e endereço.
* Fornecedor: Representa os estabelecimentos que oferecem produtos.
* Produto: Armazena as informações dos itens disponíveis para venda.
* Pedido: Representa os pedidos realizados na plataforma.

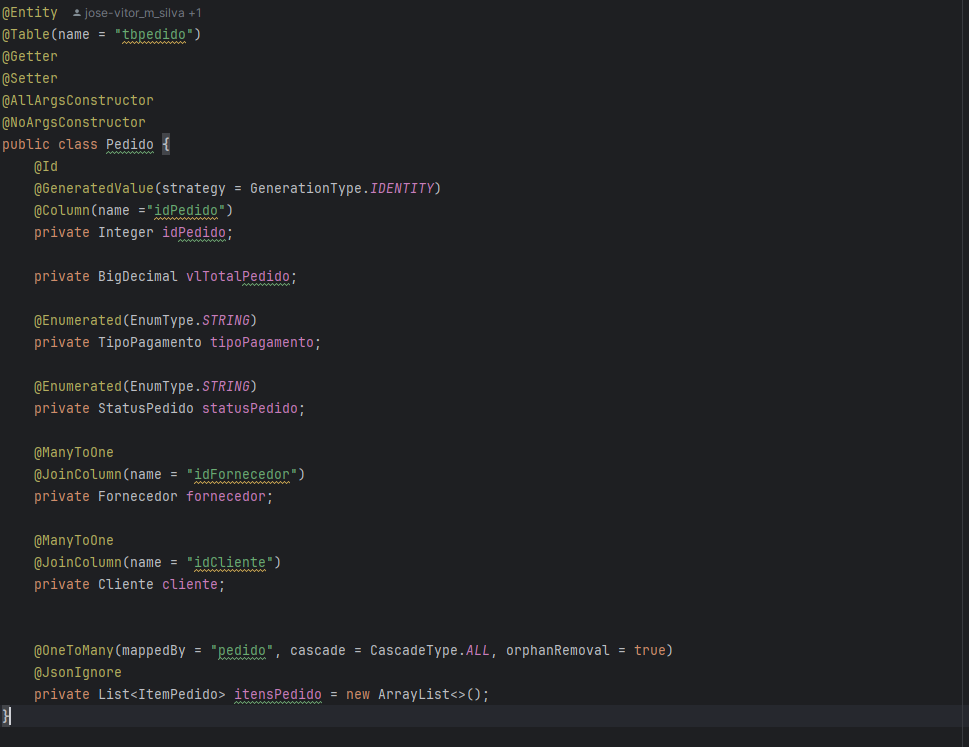


Figura 21 - Tela Backend Pedido Model

* ItemPedido: Entidade associativa entre Pedido e Produto, contendo quantidade e subtotal.
* Avaliacao: Registro da opinião do cliente após a entrega.
* Endereco: Entidade de apoio com os dados de localização.
* Admin: Representa usuários com privilégios administrativos.

#### 4.1.4 Repositories

Os repositórios estendem JpaRepository e permitem o acesso eficiente às entidades persistidas. Principais repositórios:

* ClienteRepository: Operações com clientes.
* FornecedorRepository: Acesso a dados dos fornecedores.
* ProdutoRepository: Manipulação de produtos cadastrados.
* PedidoRepository: Controle e consulta de pedidos.

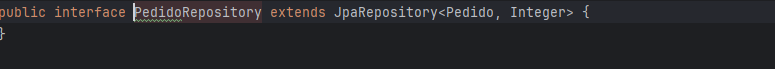


Figura 22 - Tela Backend Pedido Repository

* ItemPedidoRepository: Controle dos itens contidos em cada pedido.
* AvaliacaoRepository: Gerencia as avaliações dos pedidos.

### 4.1.5 DTOs

Os DTOs (Data Transfer Objects) são usados para transportar dados entre as camadas do sistema, organizando e limitando os atributos expostos. Entre os principais:

* ClienteRequestRegisterDTO / ClienteResponseDTO
* FornecedorRequestRegisterDTO / FornecedorResponseDTO
* ProdutoRequestDTO / ProdutoResponseDTO
* PedidoRequestDTO / PedidoResponseDTO

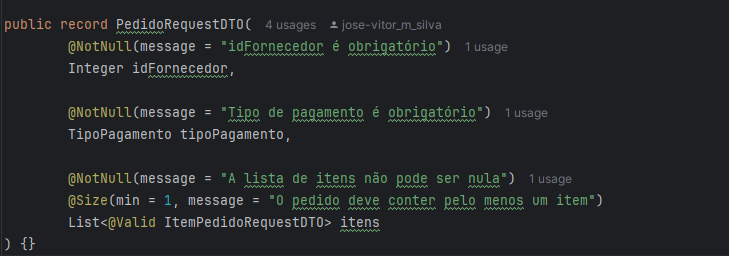


Figura 23 - Tela Backend Pedido DTO

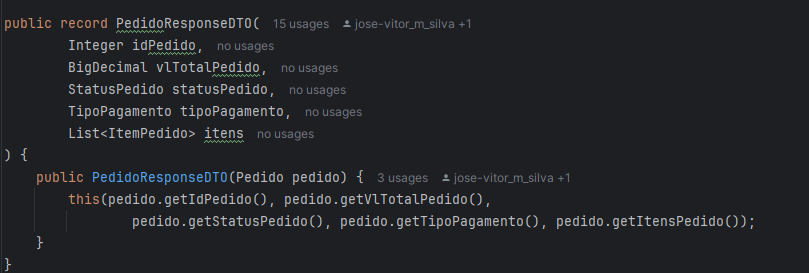


Figura 24 - Tela Backend Pedido Response DTO

* AvaliacaoRequestDTO / AvaliacaoResponseDTO
* LoginResponseDTO: Usado no retorno da autenticação com JWT
* RecSenhaClienteRequestDTO / RecSenhaFornecedorRequestDTO: DTOs para redefinição de senha

#### 4.1.6 Service

Os services implementam a lógica de negócio do sistema, funcionando como a camada intermediária entre os controllers e os repositórios. São responsáveis por validar dados, aplicar regras específicas e executar os fluxos das funcionalidades principais da aplicação.

Principais services:

* **ClienteService:** Responsável pela lógica de atualização de dados do cliente, recuperação de informações, redefinição de senha e suporte às operações do ClienteController.
* **FornecedorService:** Gerencia as operações relacionadas aos dados do fornecedor, como atualização de perfil e interação com produtos e pedidos vinculados ao restaurante.
* **ProdutoService:** Implementa as regras para criação, edição, exclusão e listagem de produtos, garantindo a associação correta com o fornecedor responsável.
* **PedidoService:** Executa a lógica de criação de pedidos, controle e atualização de status (ex: Recebido, Em preparo, Entregue), e vinculação entre cliente, fornecedor e itens do pedido.

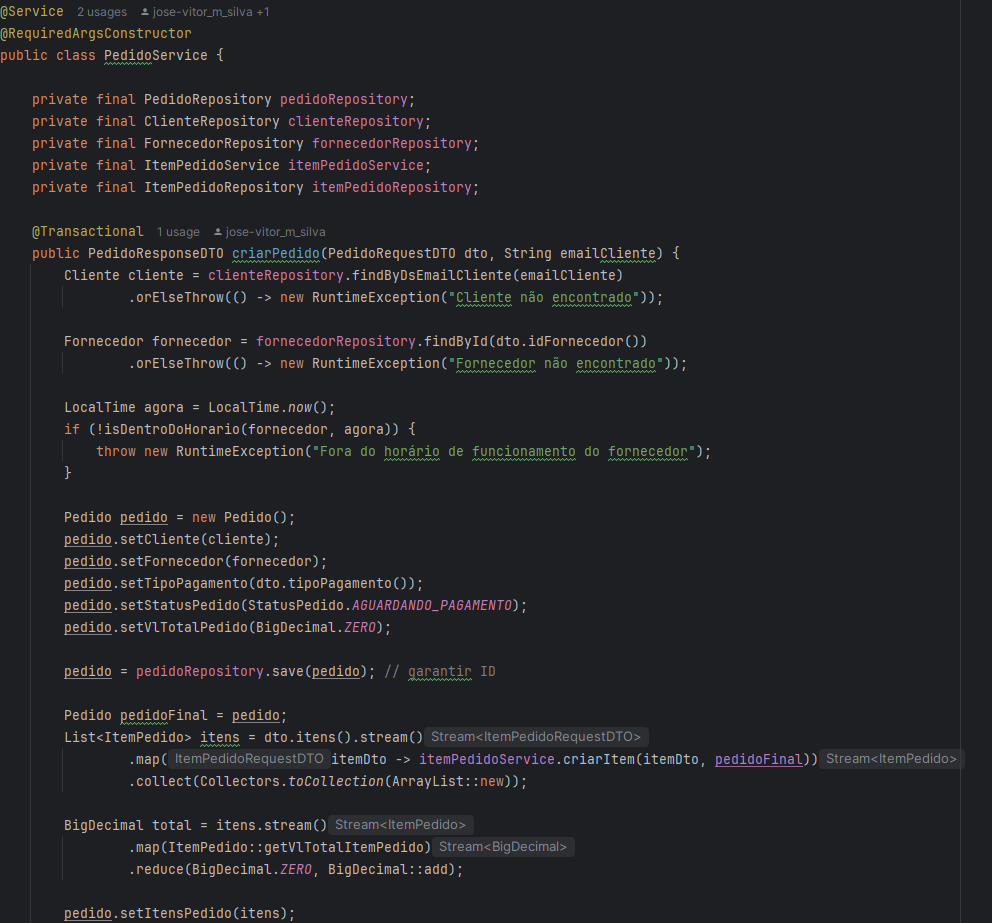


Figura 25 - Tela Backend Pedido Service

* **ItemPedidoService:** Trata da lógica de composição dos pedidos, registrando os itens escolhidos, quantidades e subtotais.
* **AvaliacaoService:** Responsável por registrar e recuperar as avaliações feitas pelos clientes após a finalização dos pedidos, garantindo que cada pedido possa ser avaliado apenas uma vez.
* **TokenService:** Gera, valida e decodifica os tokens JWT utilizados na autenticação, sendo essencial para a segurança da aplicação.
* **ClienteAuthService, FornecedorAuthService, AdminAuthService:** Gerenciam o fluxo de autenticação (login), registro e recuperação de senha, respeitando as particularidades de cada tipo de usuário.

Todos os services seguem uma organização coesa, promovendo o reaproveitamento de lógica e a manutenção do princípio da separação de responsabilidades entre as camadas do sistema.

## 4.2 Frontend

### 4.2.1 Estrutura do Projeto

O frontend foi desenvolvido utilizando **HTML5**, **CSS3**, **JavaScript** e **TypeScript** com o framework **Vite**. A estrutura foi organizada da seguinte forma:

* public/: Armazena imagens e recursos estáticos utilizados na aplicação.
* src/assets/: Contém ícones, logotipos e arquivos de estilo (CSS).
* src/pages/: Contém as páginas principais da aplicação (ex: Home, Carrinho, Login, Cadastro, Perfil).
* src/components/: Contém os componentes reutilizáveis da interface, como Header, CardProduto, ModalPedido, entre outros.
* src/services/: Contém arquivos responsáveis pela comunicação com a API REST (fetch, métodos POST, GET, PUT e DELETE).
* src/routes/: Define as rotas da aplicação com base nos perfis de usuário e nas páginas acessíveis.

### 4.2.2 Interfaces

A aplicação conta com múltiplas interfaces, cada uma projetada para um perfil e funcionalidade:

* index.html: Página de carregamento e redirecionamento de login.
* login.html: Tela de autenticação para cliente, fornecedor ou administrador.
* home.html: Tela principal para o cliente navegar pelas categorias e restaurantes.
* restaurante.html: Tela com cardápio e informações do restaurante.
* carrinho.html: Interface para visualização e finalização do pedido.
* pedidos.html: Tela de acompanhamento de pedidos.
* cadastro.html: Formulários de registro para novos usuários.

Cada interface foi desenvolvida de forma responsiva e modular, priorizando a usabilidade e o acesso por dispositivos móveis.

### 4.2.3 JavaScript/TypeScript

A lógica do frontend é modularizada e dividida por funcionalidade. Os principais arquivos JavaScript/TypeScript são:

* auth.ts: Gerencia a autenticação e o armazenamento do token JWT.
* cliente.ts: Trata das interações com os dados do cliente.
* fornecedor.ts: Responsável pelas ações de gerenciamento do fornecedor.
* produto.ts: Controla o carregamento e exibição dos produtos.
* pedido.ts: Gerencia a criação, alteração e visualização dos pedidos.
* avaliacao.ts: Controla o envio e carregamento das avaliações.

### 4.2.4 Estilos

Os estilos foram organizados em arquivos CSS específicos para cada seção:

* main.css: Estilos globais da aplicação.
* login.css: Estilo das páginas de autenticação.
* home.css: Layout e responsividade da tela principal.
* carrinho.css: Estilo visual da página de pedidos.
* fornecedor.css: Estilo específico para a área administrativa do fornecedor.

# 5. CONCLUSÃO

## 5.1 Considerações Finais

O desenvolvimento da plataforma de delivery apresentada neste relatório demonstrou a aplicação prática de tecnologias modernas e boas práticas de engenharia de software no contexto de um sistema web completo e responsivo. Inspirado em grandes soluções do mercado, como o iFood, o projeto foi concebido para simular de forma realista o funcionamento de uma aplicação de pedidos online, atendendo tanto ao público consumidor quanto aos estabelecimentos comerciais.

A arquitetura cliente-servidor adotada, com backend em **Spring Boot** e frontend desenvolvido com **HTML, CSS, JavaScript e TypeScript**, mostrou-se eficaz para garantir a separação de responsabilidades, a escalabilidade e a manutenção contínua do sistema. A utilização de APIs REST, autenticação com JWT e organização modular por responsabilidades facilitou o desenvolvimento e a integração entre as camadas.

O modelo de dados implementado reflete com precisão os relacionamentos entre clientes, fornecedores, produtos, pedidos, itens e avaliações, permitindo o registro consistente e a recuperação eficiente das informações ao longo de todo o processo de compra.

As funcionalidades desenvolvidas cobrem os principais fluxos esperados em uma aplicação de delivery, incluindo autenticação por tipo de usuário, gestão de produtos, realização e acompanhamento de pedidos, avaliação de serviços e controle de status em tempo real. Com isso, o sistema se posiciona como uma solução viável e adaptável para pequenos e médios empreendedores do setor alimentício que desejam oferecer serviços de entrega por meio de uma plataforma digital própria.