**Documentação do Sistema de Pizzaria**



Sistema de Gerenciamento de Pizzaria

Lucas Terminiello  
1 DEV – tarde  
Prof.º Bruno

SENAI – Luiz Varga  
LImeria-SP  
Junho de 2025

**Sumário**

1.Introdução

2.Metodologia de Desenvolvimento

3. Levantamento de requisitos

4.FLUXOGRAMAS E DIAGRAMAS

5.Empatia no Projeto

6.Análise dos Códigos

•Página de Login

•Página do Cliente

•Página do Administrador

•Popups

7.Relatório de Conclusão

8. LINKS

**1. Introdução**

Este documento apresenta a documentação técnica e funcional do sistema de gerenciamento de pizzaria "Tratoria Rocca". O sistema otimiza pedidos de clientes e a administração interna, oferecendo interfaces intuitivas e funcionalidades essenciais. O objetivo é fornecer uma compreensão clara da estrutura, funcionamento e tecnologias empregadas, em linguagem simples e objetiva.

**2. Metodologia de Desenvolvimento**

O desenvolvimento do sistema utilizou a metodologia ágil Kanban, focando na otimização e entrega contínua. A aplicação do Kanban incluiu:

•Visualização do Trabalho: Tarefas em quadros (A Fazer, Em Andamento, Concluído) para clareza do progresso.

•Limitação do Trabalho em Andamento (WIP): Controle de tarefas simultâneas para evitar sobrecarga.

•Gestão do Fluxo: Foco na movimentação suave das tarefas, identificando e removendo gargalos.

•Melhoria Contínua: Ajustes constantes no processo para maior eficiência.

Essa abordagem garantiu adaptabilidade, entregas frequentes e feedback contínuo.

## 3. Levantamento de Requisitos

### 3.1 Requisitos Funcionais (RF)

3.1.1 O sistema deve permitir que usuários façam login com credenciais válidas.

3.1.2 O sistema deve validar as credenciais e redirecionar o usuário para a interface correta, seja Cliente ou Administrador.

3.1.3 O sistema deve possibilitar login via conta Google como opção alternativa.

3.1.4 O cliente deve poder visualizar o cardápio completo com imagens, descrições e preços dos produtos.

3.1.5 O cliente deve poder acessar promoções e ofertas disponíveis no momento.

3.1.6 O cliente deve poder montar sua própria pizza selecionando ingredientes disponíveis.

3.1.7 O cliente deve poder adicionar pizzas pré-montadas ou personalizadas ao carrinho de compras.

3.1.8 O cliente deve poder visualizar e modificar os itens do carrinho, incluindo alterar quantidades ou remover produtos.

3.1.9 O administrador deve poder cadastrar novas pizzas, incluindo nome, descrição, preço e imagem.

3.1.10 O administrador deve poder consultar, alterar e excluir pizzas já cadastradas.

3.1.11 O administrador deve poder gerenciar os ingredientes disponíveis para a opção "Monte Sua Pizza".

3.1.12 O administrador deve poder registrar vendas realizadas no sistema.

3.1.13 O administrador deve poder gerar relatórios de vendas para análise do desempenho.

3.1.14 O sistema deve apresentar interface responsiva, funcionando adequadamente em desktops e dispositivos móveis.

3.1.15 O sistema deve fornecer feedback visual ao usuário, por meio de popups ou mensagens, informando sucesso, erros ou outras informações relevantes.

3.1.16 O sistema deve armazenar dados temporariamente no localStorage do navegador para persistência entre sessões.

### 3.2 Requisitos Não Funcionais (RNF)

3.2.1 A interface deve ser simples, clara e intuitiva para facilitar a navegação dos usuários.

3.2.2 O sistema deve carregar e responder rapidamente às ações do usuário, sem necessidade de recarregar páginas.

3.2.3 Apenas usuários autenticados devem ter acesso às funcionalidades específicas de seus perfis (cliente ou administrador).

3.2.4 O sistema deve ser compatível com os principais navegadores da web (Chrome, Firefox, Edge e Safari).

3.2.5 O design deve ser responsivo, adaptando-se a diferentes tamanhos de tela, incluindo smartphones e tablets.

3.2.6 O código deve ser modular e bem documentado para facilitar manutenções futuras.

### 3.3 Restrições do Sistema

3.3.1 O sistema será desenvolvido usando tecnologias web front-end: HTML, CSS e JavaScript.

3.3.2 O armazenamento dos dados será realizado exclusivamente no localStorage do navegador, não utilizando banco de dados servidor.

3.3.3 O sistema será utilizado preferencialmente em ambiente local ou intranet, sem integrações externas neste momento.

### 3.4 Requisitos Desejáveis (Futuras versões)

3.4.1 Implementar sistema de pagamento online integrado ao carrinho.

3.4.2 Enviar notificações por email ou SMS para confirmação de pedidos.

3.4.3 Integrar o sistema com plataforma de delivery para rastreamento dos pedidos.

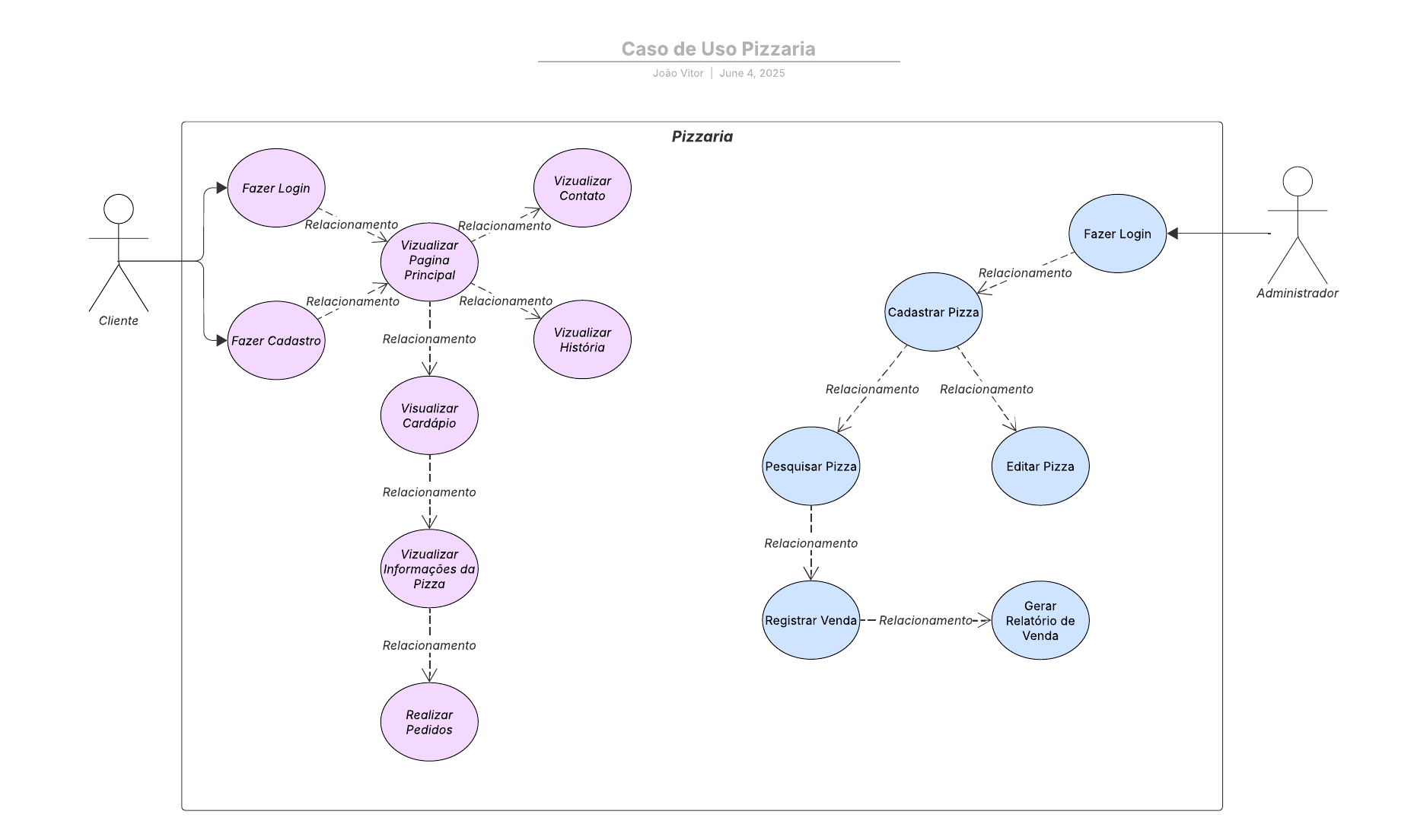
3.4.4 Permitir histórico de pedidos para clientes e administradores.

3.4.5 Possibilitar perfis de usuário com níveis de acesso variados.

# **4 Diagramas**

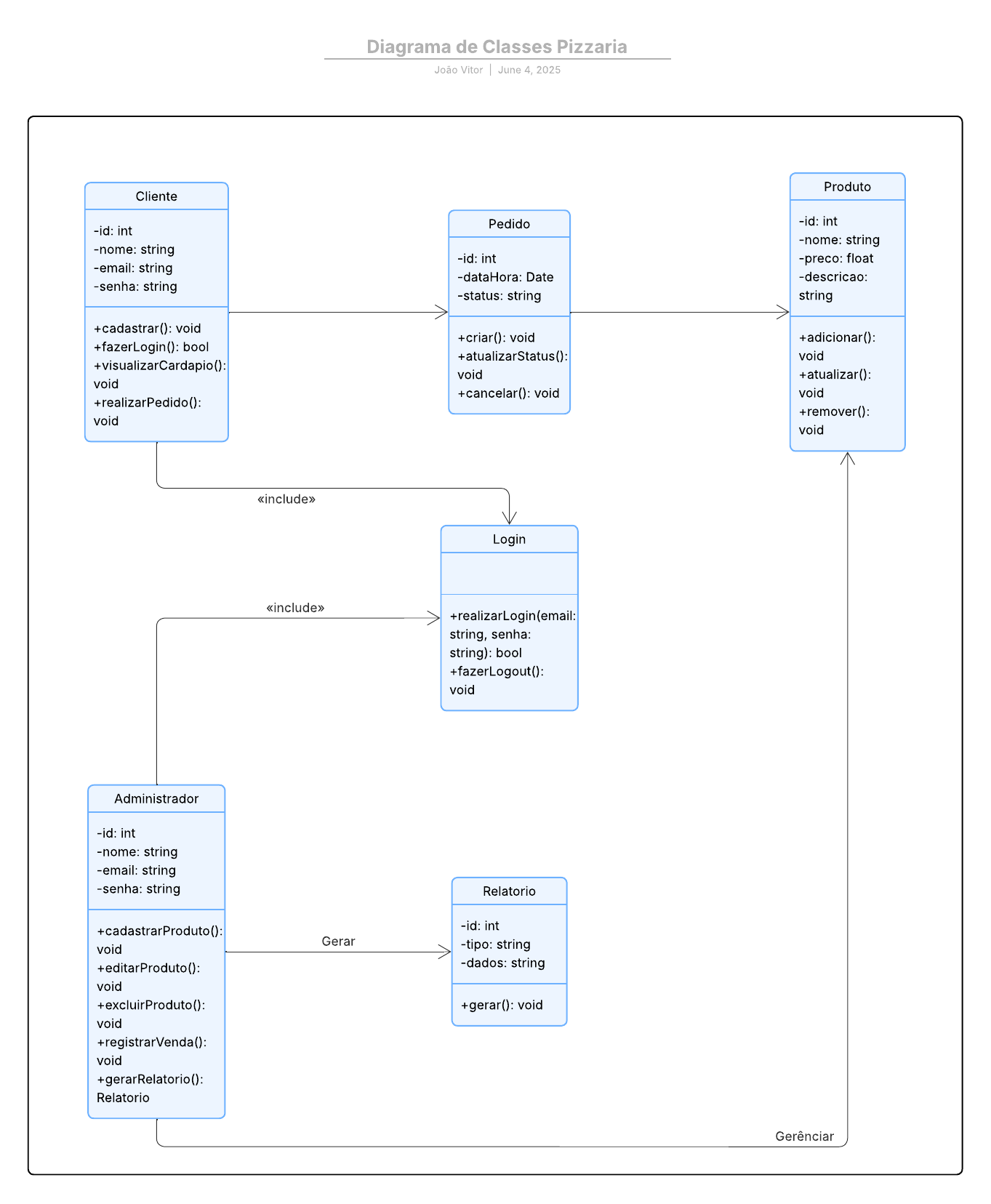
# **Casos de Uso**

O diagrama de casos de uso tem como objetivo representar de forma visual as principais interações entre os atores do sistema e suas funcionalidades. No projeto *Pizzaria Mônaco*, os dois atores principais são o **Administrador** e o **Cliente**. Cada um interage com o sistema por meio de diferentes funcionalidades, como cadastro de produtos, geração de relatórios e realização de pedidos. Esse diagrama ajuda a entender o escopo funcional do sistema e os papéis de cada tipo de usuário.



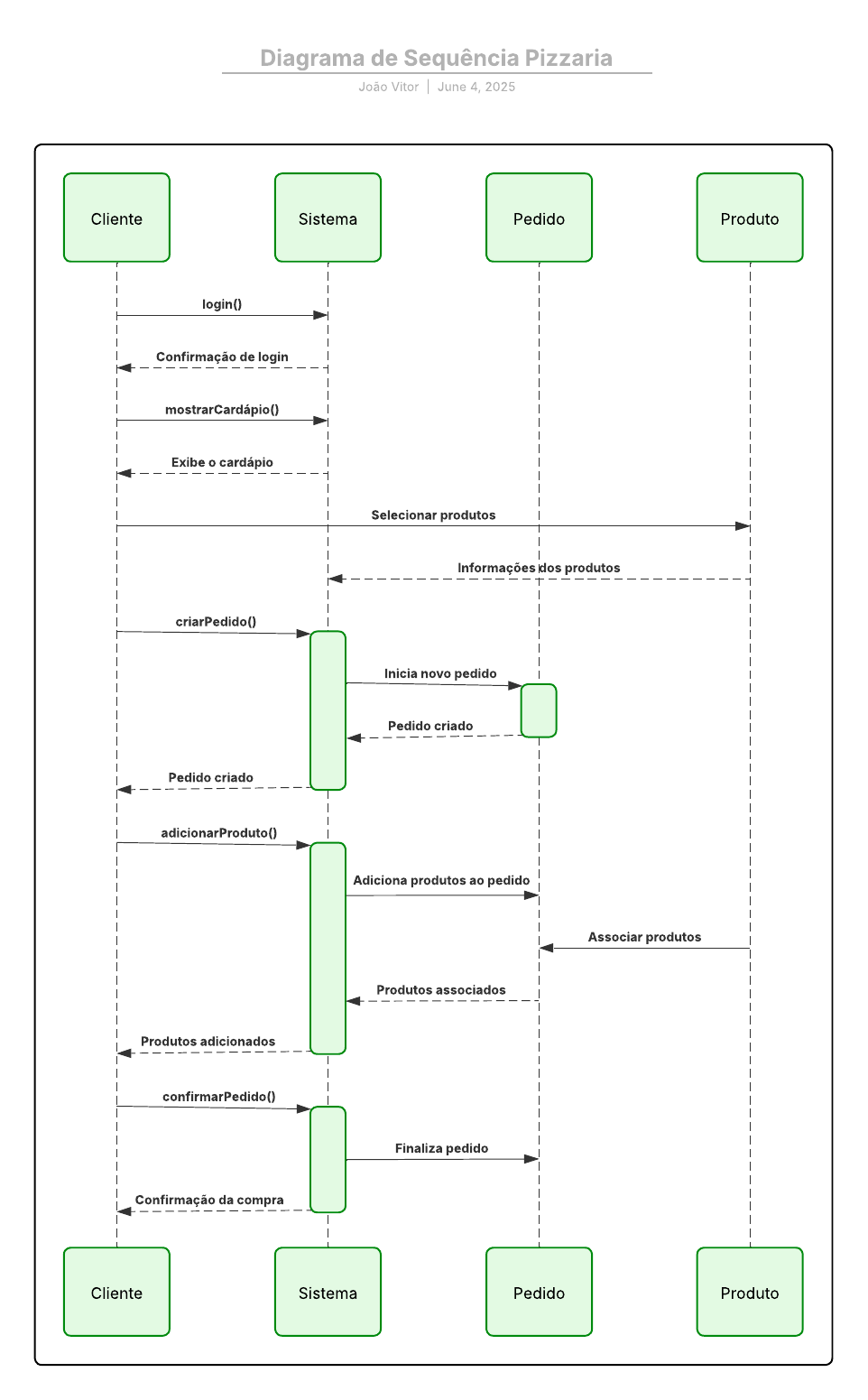
# **Classe**

O diagrama de classe apresenta a estrutura lógica do sistema, representando as principais classes envolvidas, seus atributos e métodos, além dos relacionamentos entre elas. Para o sistema da *Pizzaria Mônaco*, são destacadas classes como **Produto**, **Pedido**, **Cliente**, **Administrador** e **Relatório**. Esse diagrama auxilia no planejamento da lógica de programação e na organização dos dados ao longo do desenvolvimento.



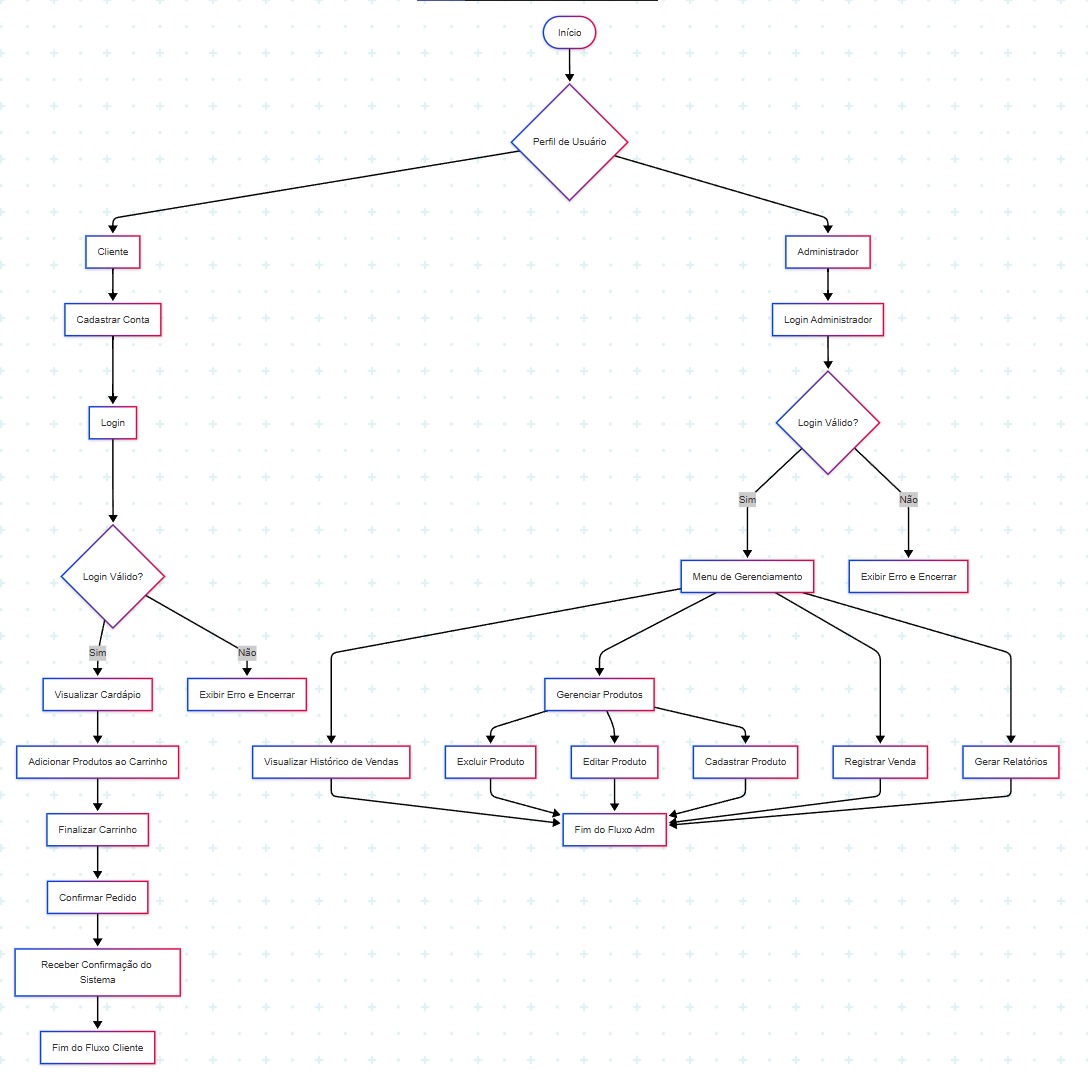
# **Sequência**

O diagrama de sequência demonstra a ordem das interações entre os objetos do sistema durante a execução de uma funcionalidade específica. Nesse projeto, ele pode representar, por exemplo, o fluxo de um pedido realizado pelo cliente, desde a seleção do produto até a confirmação da compra. Esse tipo de diagrama é útil para visualizar como os dados circulam entre as partes do sistema e garantir que os processos estejam bem estruturados.



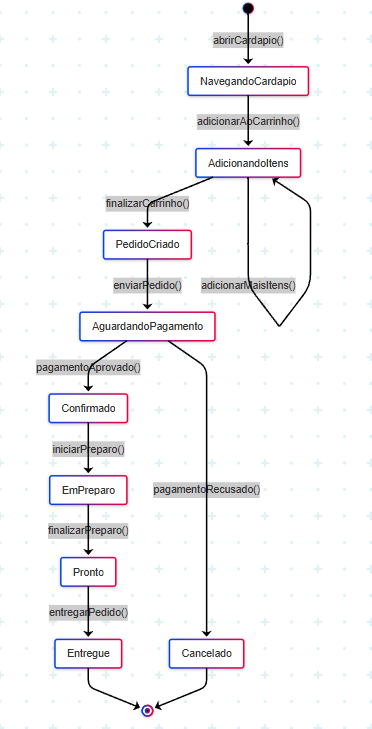
# **Funcionalidades**

O fluxograma de funcionalidades tem como finalidade ilustrar o fluxo lógico das ações realizadas no sistema, tanto pelo cliente quanto pelo administrador. Por meio de símbolos padronizados, são apresentados os caminhos percorridos durante o uso das principais funções, como login, cadastro de produtos, realização de pedidos e geração de relatórios. Esse diagrama contribui para a análise da experiência do usuário e para a clareza dos processos internos.

****

# **Estado**

O diagrama de estados representa os diferentes estados possíveis de um objeto ou entidade dentro do sistema, assim como as transições entre esses estados. No contexto da *Pizzaria Mônaco*, ele pode ser aplicado, por exemplo, ao status de um **Pedido**, que pode passar por estados como: *Pendente*, *Preparando*, *Pronto*, *Finalizado*. Esse diagrama é importante para compreender o ciclo de vida de determinadas entidades dentro do sistema e garantir consistência nas transições.



**Protótipo no Figma**

Para uma visualização interativa das interfaces, acesse o protótipo no Figma:

[Protótipo Pizzaria - Figma](https://www.figma.com/proto/16fEbQj7ct7RvFQsHAHOdG/Prot%C3%B3tipo-Pizzaria--Community-?node-id=6-223&t=4a0uR6W4seMbtKNa-1)

**5. Empatia no Projeto**

O desenvolvimento do sistema foi guiado pela empatia, buscando compreender as necessidades de clientes e administradores. Isso se manifestou em:

•Foco no Usuário Final (Cliente): Interface intuitiva, cardápio claro, promoções visíveis e ferramenta "Monte Sua Pizza" flexível para uma experiência de compra fluida.

•Otimização para o Administrador: Interface eficiente e completa para simplificar tarefas diárias, economizar tempo e reduzir erros.

•Consideração da Experiência Mobile: Design responsivo (CSS) para acessibilidade em diferentes dispositivos.

•Feedback e Melhoria Contínua: Metodologia Kanban permitiu evolução baseada em feedback, garantindo aprimoramento constante.

O sistema busca melhorar a experiência geral de todos os envolvidos.

**6. Análise dos Códigos**

Esta seção explora os arquivos de código, explicando seu papel e destacando as principais tecnologias e conceitos.

**Página de Login**

•PIZZARIA\_login.html: Estrutura da página de login (campos, botões). Usa id e onclick.

•PIZZARIA\_login.css: Estilização (cores, fontes, layout, responsividade). Usa variáveis CSS (:root) e @media.

•PIZZARIA\_login.js: Lógica de validação de login (validarLogin(), loginComGoogle()). Redireciona (window.location.href) e usa showPopup() e setTimeout().

**Página do Cliente**

•PIZZARIA\_cliente.html: Estrutura da interface do cliente (cabeçalho, navegação, banner, seções de cardápio, promoções, "Monte Sua Pizza", modal de carrinho). Usa data-target.

•PIZZARIA\_cliente.css: Estilização (design responsivo, botões, cabeçalho, cards, modal). Usa grid e flexbox.

•PIZZARIA\_cliente.js: Interatividade (carrinho, opções de montagem via localStorage). Funções como carregarCardapioCliente(), adicionarAoCarrinho(), atualizarCarrinhoUI(), atualizarResumoMontagem(). Usa event listeners.

**Página do Administrador**

•PIZZARIA\_admin.html: Estrutura da página de administração (menu, seções ocultas para cadastro, consulta, alteração, vendas, relatório). Usa tabelas.

•PIZZARIA\_admin.css: Estilização (menu, formulários, tabelas). Mantém identidade visual.

•PIZZARIA\_admin.js: Lógica de gerenciamento (cardápio, opções de montagem, vendas via localStorage). Funções como mostrarSecao(), adicionarPizza(), excluirPizza(), registrarVenda(), gerarRelatorioVendas().

**Popups**

•popup\_styles.css: Estiliza popups de mensagem (aparência, cores para sucesso, erro, info).

•popup\_function.js: Lógica para criar, exibir e remover popups (showPopup()). Usa setTimeout() para controle de tempo.

**7. Relatório de Conclusão**

O sistema "Tratoria Rocca" foi um projeto bem-sucedido, utilizando Kanban para um fluxo de trabalho organizado. Ele oferece interfaces de login, cliente e administrador para gerenciamento completo. Construído com HTML, CSS e JavaScript, e usando localStorage para persistência de dados, é leve e eficiente. Este projeto é um excelente exemplo prático de integração de tecnologias web, fornecendo uma base sólida para estudantes e profissionais. O sistema está pronto para futuras expansões e melhorias.

**8. LINKS**

[FIGMA - PIZZARIA --> CLIQUE AQUI](PIZZARIA_documentação.docx)

[TRELLO - PIZZARIA --> CLIQUE AQUI](https://trello.com/b/pruCoT3R/pizzaria)