ESCOLA SENAI “HENRIQUE LUPO”
TÉCNICO EM DESENVOLVIMENTO DE SISTEMAS

GABRIEL CARLOS DE ALMEIDA
GUILHERME LOPES LOURENÇO
KAUAN OLIVEIRA DA SILVA
KELVYN NERIS DE SENA SILVA
MATHEUS SOUZA DE MATTOS
VITOR HUGO DE MELO

CANTINA VIRTUAL

ARARAQUARA
2024

GABRIEL CARLOS DE ALMEIDA
GUILHERME LOPES LOURENÇO
KAUAN OLIVEIRA DA SILVA
KELVYN NERIS DE SENA SILVA
MATHEUS SOUZA DE MATTOS
VITOR HUGO DE MELO

CANTINA VIRTUAL

Trabalho de Conclusão de Curso apresentado como requisito parcial para a obtenção do certificado de Técnico em Desenvolvimento de Sistemas, da Escola SENAI “Henrique Lupo”.

Orientadores:

Prof. Alex Fernando Stocco

Prof. Ivo Conceição Neto

ARARAQUARA

2024

AGRADECIMENTOS

Agradecemos, em primeiro lugar, a Deus, pela dádiva da vida e pela força que nos proporcionou para vencer os desafios que surgiram ao longo desta jornada. Nossa gratidão se estende aos nossos pais, irmãos e amigos, cuja presença constante, apoio incondicional e compreensão foram pilares essenciais para a realização deste projeto.

Expressamos nossa sincera gratidão aos professores Alex Fernando Stocco e Ivo Conceição Neto, cujas orientações, sabedoria e dedicação moldaram nosso desenvolvimento acadêmico e profissional, e cuja contribuição foi indispensável para a concretização deste trabalho.

Aos educadores e à instituição SESI e SENAI, agradecemos por proporcionarem um ambiente de aprendizado inspirador, que nos preparou com conhecimento e ferramentas essenciais para enfrentar o mercado de trabalho com confiança.

Por fim, nossos agradecimentos vão para todos aqueles que, de alguma forma, contribuíram para o sucesso do projeto "Cantina Virtual". Seja com uma palavra de incentivo, uma sugestão valiosa ou um gesto de colaboração, sua contribuição foi vital para que chegássemos até aqui. A todos que estiveram ao nosso lado, direta ou indiretamente, expressamos nossa mais profunda gratidão.

"A tecnologia tornou possível a grandeza da comunicação, mas nada substitui a profundidade da conexão humana."

(Carl Sagan).

RESUMO

O presente trabalho teve como objetivo o desenvolvimento do sistema Cantina Virtual, uma plataforma voltada para a otimização do processo de pedidos na cantina escolar, com o intuito de melhorar a organização, reduzir filas e otimizar o tempo de espera tanto para os alunos quanto para as funcionárias responsáveis pelo atendimento. A metodologia adotada envolveu análise de sistemas existentes, entrevistas com funcionárias da cantina e a criação de protótipos que foram testados por meio de testes de usabilidade. O sistema foi desenvolvido utilizando tecnologias como HTML, CSS, JavaScript, SQL e Bootstrap, proporcionando uma interface intuitiva e funcionalidades práticas, como o controle de pedidos, cadastro de produtos e a visualização do cardápio. Durante os testes, as funcionárias aprovaram a plataforma, destacando a melhoria no fluxo de pedidos e a redução do tempo de espera, além de sugerirem melhorias que podem ser implementadas em versões futuras. O sistema atingiu seus objetivos ao promover a otimização do trabalho das funcionárias da cantina, reduzindo a sobrecarga de tarefas e garantindo uma experiência mais eficiente e satisfatória para os usuários. Como sugestão para aprimoramentos futuros, a inclusão de um sistema de pagamento digital poderia contribuir ainda mais para a agilidade nas transações, melhorando a experiência geral tanto para os alunos quanto para as funcionárias da cantina. Esse projeto permitiu ao grupo adquirir novas habilidades técnicas e fortalecer competências socioemocionais, como trabalho em equipe e resiliência, durante seu desenvolvimento.

Palavras-chave: sistema; cantina; otimização; funcionalidade; usabilidade.

ABSTRACT

This work aimed at the development of the Cantina Virtual system, a platform designed to optimize the ordering process at the school canteen, with the goal of improving organization, reducing queues, and optimizing wait times for both students and the staff responsible for service. The methodology involved analyzing existing systems, interviewing canteen workers, and creating prototypes that were tested through usability tests. The system was developed using technologies such as HTML, CSS, JavaScript, SQL, and Bootstrap, providing an intuitive interface and practical functionalities, such as order control, product registration, and menu visualization. During the tests, the staff approved the platform, highlighting improvements in the order flow and reduced wait times, as well as suggesting improvements that could be implemented in future versions. The system achieved its goals by promoting the optimization of the work for the canteen staff, reducing task overload and ensuring a more efficient and satisfactory experience for users. A suggestion for future improvements is the inclusion of a digital payment system, which could further enhance transaction speed, improving the overall experience for both students and staff. This project allowed the team to acquire new technical skills and strengthen socio-emotional competencies, such as teamwork and resilience, throughout its development.

Keywords: system; canteen; optimization; functionality; usability.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

SENAI	Serviço Nacional de Aprendizagem Industrial
TCC	Trabalho de Conclusão de Curso
AI	Inteligência Artificial
IDE	Ambiente de Desenvolvimento Integrado
HTML	HyperText Markup Language
CSS	Cascading Style Sheets
JS	JavaScript
SQL	Structured Query Language
API	Application Programming Interface
PK	Primary Key (Chave Primária)
FK	Foreign Key (Chave Estrangeira)
DER	Diagrama de Entidade-Relacionamento
MER	Modelo Entidade-Relacionamento
JSON	JavaScript Object Notation
UI	User Interface (Interface do Usuário)
UX	User Experience (Experiência do Usuário)

Sumário

1 INTRODUÇÃO

O avanço da tecnologia tem revolucionado diversos aspectos do dia a dia, e uma das áreas que mais se beneficia desse progresso é a gestão de serviços. No ambiente escolar, a cantina desempenha um papel fundamental no atendimento às necessidades alimentares de alunos, professores e funcionários.

Contudo, a gestão habitual das cantinas, com filas longas e atendimento manual, pode ser ineficiente e desgastante, tanto para os clientes quanto para a equipe de trabalho. Então foi pensado, quais soluções podem ser implementadas na Cantina da LU para melhorar a experiência do cliente e a gestão operacional?

Pensando nisso, este trabalho de conclusão de curso propõe o desenvolvimento do sistema Cantina Virtual, um sistema versátil projetado para otimizar o processo de compras na cantina do SENAI, permitindo que os usuários realizem pedidos de forma online e acompanhem o status de suas refeições em tempo real.

Este projeto é uma aplicação prática dos conhecimentos adquiridos durante o curso de Análise e Desenvolvimento de Sistemas, além de contribuir para o uso de soluções tecnológicas que impactem positivamente o cotidiano escolar.

1.1 Objetivo Geral

O projeto tem como objetivo desenvolver um aplicativo para a gestão de compras na cantina escolar do SENAI, permitindo que alunos, professores e funcionários realizem pedidos de forma online, visando otimizar o processo de compra, reduzir filas e melhorar a experiência dos usuários dentro do ambiente escolar.

1.2 Objetivos Específicos

- Entrevistar as responsáveis da cantina, para entender seu funcionamento

e suas necessidades de melhoria.

- Realizar uma reunião para discussão dos requisitos do sistema e seu funcionamento em geral.

- Estruturar um diagrama das funcionalidades principais do sistema, para ter uma visão mais organizada do funcionamento da Cantina Virtual.

- Formatar e programar a base do sistema, exemplo das páginas principais
e as funcionalidades básicas do sistema.

- Testar a usabilidade de sistema com testes, apresentações e questionários de avaliação do público.

- Solucionar problemas que possam existir no sistema, e finalizar detalhes.

1.3 Justificativa

A crescente demanda por praticidade e agilidade em serviços de alimentação, especialmente em ambientes educacionais, justifica a necessidade de um sistema como a Cantina Virtual. As filas extensas e a falta de um controle mais eficiente no atendimento geram insatisfação e atrasos, impactando diretamente o tempo disponível para os alunos e profissionais se alimentarem. Com o desenvolvimento de uma plataforma de pedidos online, espera-se reduzir as filas, otimizar o processo de compra e entrega de produtos, além de oferecer uma experiência mais personalizada e prática para os usuários.

Além disso, o projeto reflete a tendência de digitalização dos serviços, que vem sendo amplamente adotada em diversos setores, incluindo o educacional. O desenvolvimento deste sistema contribuirá para o aprimoramento das habilidades

técnicas adquiridas ao longo do curso e proporcionará uma solução prática e aplicável à realidade das cantinas escolares. Automatizar esse tipo de tarefa faz parte de toda a ascensão tecnológica que o mundo vem passando, assim facilitando processos que antes eram demorados e hoje podem ser muito mais rápidos.

2 DESENVOLVIMENTO

Nesta seção, será detalhado o processo de criação do sistema Cantina Virtual, desde a análise dos requisitos até a implementação e testes. O desenvolvimento foi dividido em etapas, visando garantir a organização e a clareza durante a construção do projeto.

Primeiramente, estabelecemos como objetivo achar alguma área do SENAI que precisasse de algum apoio ou melhoria, e que necessitasse de algum auxílio tecnológico. Assim poderíamos criar um projeto interessante para a conclusão de curso, e que também fosse benéfico para a instituição.

Então, tivemos um período de discussão para que conseguíssemos definir algum setor que exigisse algum tipo de renovação. Assim chegamos na cantina, um local que é extremamente essencial para a estrutura do SENAI, mas que precisava de ajustes na questão do tamanho das filas e na venda de marmitas. Por isso, decidimos trabalhar nessa adversidade.

Posteriormente, realizamos um levantamento de requisitos para entender as necessidades dos usuários e da equipe da cantina. A partir dessa análise, foi definido o escopo do sistema, contemplando as principais funcionalidades, como cadastro de usuários, visualização do cardápio, pedidos online e notificações em tempo real.

Além disso, discutimos sobre a arquitetura do sistema, as tecnologias utilizadas no desenvolvimento, e as metodologias empregadas para garantir a qualidade e eficiência do projeto. As etapas de implementação foram realizadas de maneira organizada para assegurar que o sistema atendesse às expectativas finais.

2.1 Entrevista

Como parte do levantamento de requisitos, foi realizada uma entrevista com as responsáveis pela cantina para entender detalhadamente as operações do local e identificar os principais desafios enfrentados no dia a dia. A entrevista abordou questões como:

- Informações gerais sobre o funcionamento da cantina;
- Produtos oferecidos e a gestão do estoque;
- Processos internos de atendimento e preparação de refeições;
- Feedback recebido dos clientes (alunos, professores e funcionários);
- Tecnologias já presentes na operação da cantina;
- Desafios enfrentados, especialmente durante os horários de maior movimento.

A partir dessa análise, os principais problemas identificados foram o tamanho das filas durante os horários de pico e a dificuldade na dinâmica dos pedidos de marmitas realizados nas salas de aula. Esses fatores contribuíam para atrasos e insatisfação dos clientes, além de sobrecarregar a equipe da cantina. Com base nessas constatações, o sistema proposto foi desenhado para resolver esses problemas, priorizando a otimização do tempo de espera e a gestão dos pedidos.

Durante a entrevista, as responsáveis pela cantina explicaram como ocorre o funcionamento diário do estabelecimento. A cantina atende a alunos, professores e funcionários, operando durante os intervalos e horários de almoço. O fluxo de pessoas é maior nos horários de pico e aos finais de semana, geralmente durante o intervalo entre aulas, o que gera filas longas e aumenta a pressão sobre o atendimento. A equipe é composta por um pequeno número de funcionárias, o que torna o atendimento mais lento nos momentos de maior movimento.

A cantina oferece uma variedade de produtos, incluindo salgados, sucos, lanches rápidos e marmitas. Os produtos são adquiridos de fornecedores externos e armazenados de acordo com suas respectivas condições de conservação. O

gerenciamento do estoque é feito manualmente, apenas com o auxílio de um pequeno sistema automatizado. Isso gera dificuldades em controlar o nível de estoque, levando a problemas como falta de produtos em horários de alta demanda e desperdício de itens que não foram vendidos.

O atendimento na cantina é realizado por meio de pedidos feitos diretamente no balcão. Após o pedido, os alimentos são preparados, o que leva um tempo de espera significativo, especialmente para pratos como as marmitas, que precisam ser aquecidas antes de serem entregues ao cliente. Esse processo não apenas contribui para a formação de filas, mas também sobrecarrega a equipe nos horários de maior movimento.

As responsáveis pela cantina mencionaram que os clientes, em sua maioria, reclamam das longas filas e do tempo de espera para receberem seus pedidos, especialmente nos horários de intervalo. Outro ponto levantado foi a dificuldade dos alunos em fazer pedidos de marmitas diretamente das salas de aula, o que acaba resultando em pedidos feitos em cima da hora, dificultando o atendimento eficiente. No entanto, também houveram elogios à qualidade dos alimentos.

Figura 1 - Fila da Cantina

Fonte: Autoria Própria (2024)

Atualmente, a cantina opera com um sistema de caixa eletrônico para o recebimento dos pagamentos, e utiliza uma tecnologia básica. Os pedidos são feitos diretamente no balcão, de forma manual, sem o uso de aplicativos. Isso limita a eficiência no controle dos processos e dificulta a comunicação entre os clientes e a equipe da cantina.

Os maiores desafios enfrentados pela cantina ocorrem durante os horários de maior movimento, como os intervalos entre as aulas. Nesse período, a formação de filas extensas é inevitável, o que gera insatisfação entre os clientes. Outro desafio é o tempo de preparo das marmitas, que frequentemente precisam ser aquecidas após o pedido, resultando em um processo lento e ineficiente. A simplicidade do sistema digital atual para gerenciar esses pedidos também contribui para a sobrecarga da equipe, que precisa lidar com pedidos acumulados e falta de meios para solucionar esse problema.

2.2 Sistemas de Inspiração

Ainda no processo de levantamento de requisitos para a Cantina Virtual, foram analisados sistemas já estabelecidos no mercado, como o Anota AI, iFood e Rappi, que serviram de inspiração para a criação da plataforma. Esses sistemas oferecem soluções digitais para otimizar o processo de pedidos de alimentos, focando em praticidade e agilidade para os usuários.

O Anota AI é um sistema de gestão de pedidos voltado para pequenos e médios negócios do setor de alimentação. Ele se destaca por oferecer uma plataforma simplificada que permite que restaurantes e cantinas recebam pedidos diretamente pelo WhatsApp. A sua facilidade de uso e a integração com um canal de comunicação amplamente acessível foram aspectos inspiradores para o nosso sistema. No desenvolvimento da Cantina Virtual, buscamos adotar a mesma simplicidade e eficiência, mas com a proposta de criar uma interface web dedicada para atender às necessidades específicas da cantina escolar do SENAI.

O iFood é uma plataforma líder em delivery de alimentos no Brasil, oferecendo uma ótima experiência para os usuários com funcionalidades como acompanhamento de pedidos em tempo real, diversas opções de pagamento e um extenso catálogo de estabelecimentos. A rapidez no processo de pedidos e a eficiência no atendimento ao cliente foram aspectos inspiradores para a Cantina Virtual. Embora o foco do iFood seja o delivery, o conceito de facilitar o processo de pedidos e melhorar a experiência do usuário foi utilizado de inspiração para o ambiente escolar, com o objetivo de reduzir filas e melhorar a gestão de pedidos.

A plataforma do Rappi segue a mesma linha. Uma rede de delivery que disponibiliza o acesso a mercados e redes de restaurante, o que não é o foco do sistema da Cantina Virtual, mas serviu de grande inspiração para o desenvolvimento da interface e design do projeto.

3 LEVANTAMENTO DE REQUISITOS E REGRAS DE NEGÓCIO

Para iniciar o desenvolvimento da Cantina Virtual, começamos montando o levantamento de requisitos e identificando as principais regras de negócio.

3.1 Requisitos Funcionais

Requisitos funcionais descrevem as funcionalidades que um sistema deve oferecer para atender às necessidades específicas dos usuários e aos objetivos do projeto. Eles detalham todas as funções do sistema, assim garantindo que todas as funcionalidades sejam implementadas de maneira clara.

3.2 Requisitos Não-Funcionais

Requisitos não funcionais apresentam as características e restrições que o sistema deve possuir para garantir a qualidade de seu desempenho, segurança, usabilidade e confiabilidade. Eles estão relacionados a forma de como o sistema deve se comportar enquanto realiza as funcionalidades.

3.3 Regras de Negócio

Regras de negócio são diretrizes específicas que definem como o sistema deve operar dentro do contexto do negócio. Elas estabelecem restrições, condições e políticas que precisam ser seguidas para que o sistema funcione de acordo com as necessidades do serviço.

4 FERRAMENTAS E LINGUAGENS DE PROGRAMÇÃO

4.1 Visual Studio Code

O Visual Studio Code foi o ambiente de desenvolvimento integrado (IDE) escolhido para a codificação do projeto. Ele oferece suporte a várias linguagens de programação, como HyperText Markup Language (HTML), *Cascading Style Sheets* (CSS), JavaScript e *Structured Query Language* (SQL), que foram fundamentais para o desenvolvimento da aplicação. Além disso, suas extensões facilitam o gerenciamento de bibliotecas e frameworks utilizados no sistema.
Remessa Online (2021)

4.1.1 Editor de Código-Fonte

O editor de código-fonte é a ferramenta onde o desenvolvedor escreve e edita o código do sistema. Ele facilita a programação, destacando diferentes partes do código com cores diferentes, o que ajuda a melhorar a visualização e o entendimento do que está sendo escrito. Além disso, o editor sugere partes do código automaticamente enquanto o programador digita, economizando tempo e ajudando a evitar erros.
Remessa Online (2021)

4.1.2 Depurador (Debugger)

O depurador é uma função que permite "pausar" o código em pontos específicos para ver o que está acontecendo naquele momento. Com ele, o programador pode verificar o valor das variáveis e entender o que o sistema está fazendo. Isso é muito útil para encontrar erros, pois também permite executar o código uma linha de cada vez, observando exatamente como ele está se comportando.
Remessa Online (2021)

4.1.3 Terminal Integrado

O terminal integrado é uma ferramenta dentro do ambiente de desenvolvimento onde o programador pode digitar comandos e realizar tarefas diretamente. Isso é útil para executar programas, rodar testes ou verificar erros sem precisar sair da ferramenta de desenvolvimento, deixando tudo mais prático.
Remessa Online (2021)

4.2 Git e GitHub

Para o controle de versão e trabalho em equipe, utilizamos o Git em conjunto com o GitHub. O Git é uma ferramenta que registra todas as alterações feitas no código, permitindo que os desenvolvedores voltem a versões anteriores caso algo

dê errado. Já o GitHub é uma plataforma online onde o código do projeto pode ser armazenado e compartilhado. No contexto da Cantina Virtual, isso foi fundamental para manter o desenvolvimento organizado, já que diferentes desenvolvedores puderam trabalhar em partes do sistema ao mesmo tempo, sem causar conflitos no código. Além disso, o GitHub também facilitou a colaboração entre membros da equipe, permitindo que cada um contribuísse e revisasse o trabalho dos outros.
Camila Fernanda (2024), Vinícius Louzada (2024)

4.3 Google Keep

O Google Keep é uma ferramenta de organização e produtividade desenvolvida pelo Google, que permite criar e gerenciar notas, listas e lembretes de forma prática e colaborativa. Ele também possibilita uma sincronização automática entre dispositivos e compartilhamento com outros usuários.

No desenvolvimento do projeto Cantina Virtual, o Google Keep desempenhou um papel essencial no gerenciamento das ideias do projeto, e no compartilhamento de informações entre os membros do grupo.

4.4 Back-End

Back-End (também escrito em outras grafias “backend”), refere-se à parte da aplicação que opera no servidor e é responsável por gerenciar a lógica de negócios, o acesso a banco de dados e a autenticação de usuários. Para o Cantina Virtual, o back-end foi desenvolvido utilizando uma linguagem como Python e um sistema de gerenciamento de banco de dados SQL. O back-end foi crucial para a operação da aplicação, pois trata de todas as solicitações feitas pelo front-end e garante que os dados sejam processados e armazenados corretamente.
Mario Souto (2024)

4.4.1 Python

Python é a principal linguagem de programação que dita o funcionamento do backend. Oferece suporte para várias funções, como manipulação de dados,

controle de fluxo, e integração com outras tecnologias, o que a torna muito útil para o desenvolvimento do sistema.

Na Cantina Virtual, Python é responsável por toda a lógica que acontece no backend, como o gerenciamento de pedidos, a conexão com o banco de dados, e o processamento das informações do usuário. Isso permite que o sistema funcione de forma organizada, cuidando de tarefas como atualizar o estoque de produtos e verificar o status dos pedidos.
AWS (2024)

4.4.1.1 Flask

Flask é um *microframework* em Python usado para o aprimoramento de aplicações web. Ele é conhecido por sua simplicidade e flexibilidade, permitindo a criação de APIs e páginas web de forma rápida. No projeto da Cantina Virtual, o Flask foi utilizado para gerenciar as rotas do servidor e facilitar a comunicação entre o frontend e o backend, além de integrar a aplicação com o banco de dados.

4.4.1.1.1 Jsonify

O método Jsonify é uma função do framework Flask, em Python, que facilita a conversão de dados em formato Python (como listas e dicionários) para o formato JSON (JavaScript Object Notation), que é amplamente usado para transferir dados entre aplicações web.

Em nosso projeto, o uso do método jsonify foi essencial para implementar uma comunicação eficiente entre o back-end e o front-end (usado pelos alunos e administradores). O jsonify permite que dados estruturados sejam enviados em formato JSON, facilitando a integração e o funcionamento dinâmico do sistema.

4.4.2 API

Application Programming Interface (API) é um conjunto de definições e protocolos que permite que diferentes sistemas de software se comuniquem entre si. Elas foram fundamentais para a integração com sistemas externos, como o sistema de SMS, que foi utilizado em nosso projeto para enviar códigos de confirmação aos clientes.

4.4.3 Banco de Dados

Um banco de dados é um sistema organizado para armazenar, gerenciar e recuperar informações de forma eficiente. Ele funciona como um repositório digital onde os dados são estruturados, permitindo acesso, consulta, atualização e exclusão desses elementos.

4.4.3.1 MySQL Workbench

O MySQL Workbench foi utilizado para modelagem e gerenciamento do banco de dados. Ele proporcionou uma interface gráfica para visualizar as tabelas e criar consultas SQL, facilitando a administração dos dados armazenados no sistema, como os produtos da cantina, pedidos e usuários.
Awari (2023)

4.4.3.2 SQL

SQL (Structured Query Language) é uma linguagem usada para gerenciar e manipular dados em bancos de dados. Em outras palavras, é o que permite acessar, organizar, modificar e gerenciar informações em uma base de dados de maneira eficiente.

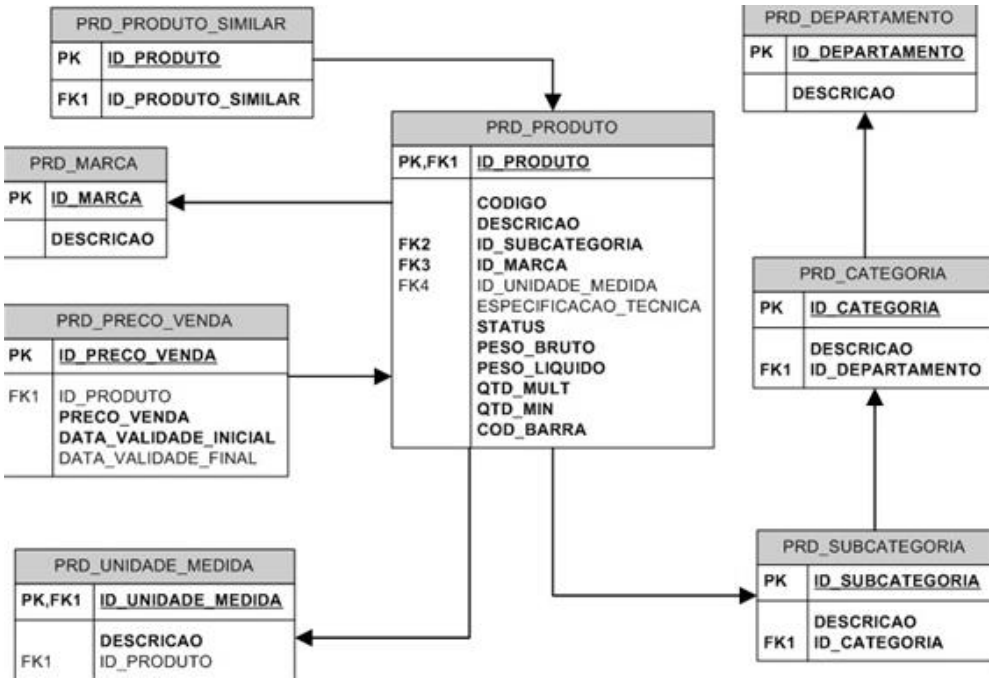
Para entender de forma simples, pense em um banco de dados como uma grande tabela ou planilha onde você armazena informações. O SQL é a ferramenta utilizada para puxar informações da tabela.
Awari (2023)

4.4.3.3 Diagrama do Banco de Dados

O diagrama de banco de dados é uma representação das relações entre os dados armazenados em um sistema, facilitando o entendimento de como as informações são conectadas. No desenvolvimento da Cantina Virtual, adotamos dois modelos para a criação dos diagramas do banco de dados. O Diagrama de Entidade-Relacionamento (DER) e o Modelo Entidade-Relacionamento (MER). Esses modelos foram muito importantes para organizar as informações que o sistema precisaria gerenciar.

Usamos o DER para mapear as principais entidades do sistema em tabelas com todas suas informações, utilizando o nome em que elas estão definidas no banco de dados, e separando-as pela sua importância, Chave-primária (PK) e Chave-Estrangeira (FK). Esse modelo ajudou a estruturar as tabelas do banco de dados, separando todos os atributos em cada tabela e fazendo suas devidas ligações, como por exemplo, a relação entre o cliente e seus pedidos ou entre o pedido e os itens selecionados.

Figura 2 - Exemplo Diagrama DER

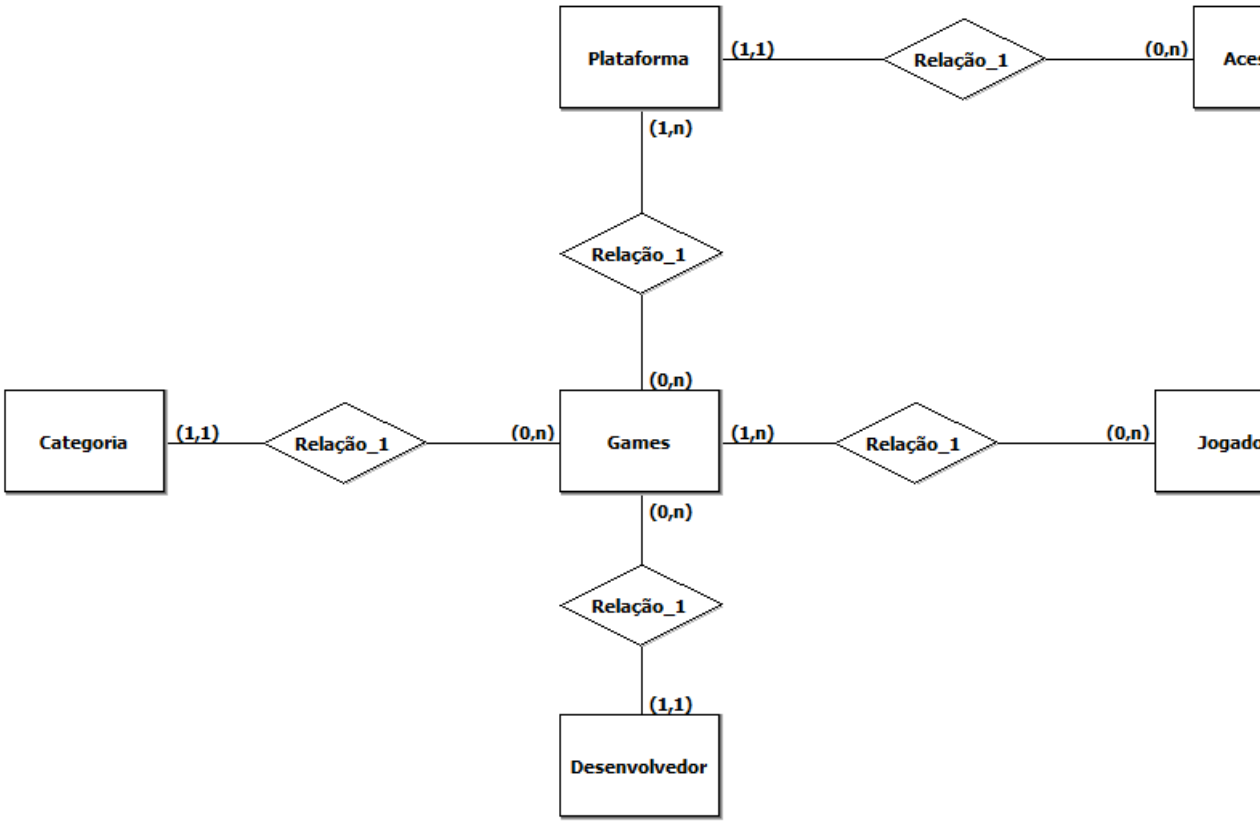


Fonte: Wellington Balbo (2010)

Já o MER apresenta uma visão mais simples do sistema, destacando os relacionamentos e ajudando a definir as regras que cada entidade deve seguir. Com o MER, conseguimos definir, por exemplo, que um cliente pode fazer múltiplos pedidos e que cada pedido pode conter vários produtos, além de especificar como cada um desses dados se relaciona.

No MER, a proporção N/1 representa uma relação entre entidades, onde o "N" simboliza a parte que pode ter muitas possibilidades, enquanto o "1" refere-se a uma entidade que possui apenas uma possibilidade associada. Por exemplo, no sistema da Cantina Virtual, um cliente pode fazer vários pedidos, mas cada pedido pertence a um único cliente, também representando uma relação de muitos para um.

Figura 3 - Exemplo Diagrama MER



Fonte: Gabriel de Barros Pontes (2018)

Danielle Oliveira (2023)

4.5 Front-End

Front-End (também escrito em outras grafias “frontend”), refere-se à parte da aplicação web que interage diretamente com o usuário. É a interface que os usuários visualizam e utilizam, composta por elementos como layout, design e interação. Para o projeto Cantina Virtual, o front-end foi desenvolvido utilizando tecnologias como HTML, CSS e JavaScript. Essas ferramentas permitem a criação de uma interface visualmente atraente, que melhora a experiência do usuário ao navegar pelo sistema.

Mario Souto (2024)

4.5.1 HTML

HTML é a linguagem básica de marcação utilizada para criar o conteúdo e a estrutura de páginas da web. Ela é responsável por organizar o conteúdo presente em um sistema, como textos, imagens, links e vídeos. HTML é fundamental para qualquer sistema, porque ela organiza o conteúdo de maneira que os navegadores da web possam entender e exibir corretamente. Sem HTML, não haveria uma forma de estruturar as informações que você deseja mostrar.
Awari (2023)

4.5.2 CSS

CSS é a linguagem usada para definir o visual de um sistema. O CSS é responsável por decorar o conteúdo do HTML. O CSS faz com que o sistema fique visualmente agradável e organizado. Ele é o que transforma uma página simples em algo visualmente atraente e fácil de usar. Então toda estilização de cores, barras de navegação, botões e seções, vem do CSS.
Awari (2023)

4.5.3 JavaScript

JavaScript é uma linguagem de programação muito importante para o desenvolvimento de sistemas. Ele permite que o sistema responda de forma imediata às ações dos usuários, proporcionando uma experiência de navegação fluida. No contexto do Cantina Virtual, o JavaScript desempenha várias funções importantes, como a atualização dinâmica dos itens no carrinho de compras, o cálculo automático dos valores de pedidos e a validação de formulários de forma rápida.
Awari (2023)

Além disso, o JavaScript facilita a interação do usuário com o sistema sem a necessidade de recarregar a página inteira. Por exemplo, ao adicionar guarnições a um pedido ou remover itens do carrinho, essas ações podem ser realizadas

instantaneamente com o auxílio de JavaScript, mantendo o fluxo de compra contínuo.

4.5.4 Wireframe e Prototipagem

O wireframe é uma representação visual simples da interface do sistema. Ele serve para ilustrar a estrutura e o layout das telas, sem se preocupar com detalhes como cores ou fontes. O objetivo principal do wireframe é organizar a disposição dos elementos da interface, como menus, botões, imagens e campos de entrada, garantindo que a navegação seja clara pro usuário.

No caso do Cantina Virtual, a criação de wireframes foi fundamental para planejar a disposição das telas, como a exibição do cardápio, a adição de produtos ao carrinho e a visualização dos detalhes do pedido. Com o wireframe, foi possível alinhar a organização dos elementos de todo o sistema, deixando o projeto com uma navegação intuitiva.

A prototipagem vai além do wireframe, criando uma versão interativa e mais próxima da versão final do sistema. Ela adiciona cores e elementos visuais que se aproximam da versão final da estrutura do projeto.

Para o Cantina Virtual, a prototipagem foi essencial para implementarmos as cores que iriamos utilizar no sistema e todos os outros elementos que deixariam o projeto mais chamativo para o usuário.

4.5.4.1 Figma

O Figma foi a ferramenta escolhida para o design da interface de usuário (UI) da Cantina Virtual. Com ele, foram criados Wireframes (esboços simples da interface) e protótipos de alta fidelidade (versões mais detalhadas), o que permitiu visualizar e testar o layout do sistema antes de ser desenvolvido. Isso ajudou a garantir que a interface da Cantina Virtual fosse fácil de usar, tornando-a simples para os clientes e para os administradores do sistema, e que funcionasse bem tanto em computadores quanto em dispositivos móveis. O Figma também facilitou a

colaboração entre os desenvolvedores, já que todos podiam acessar e comentar o design em tempo real, ajudando a alinhar a estética e a funcionalidade do sistema. *Mateus Villain, Maria Isabelle Silveira (2024)*

4.5.4.1.1 Figma – Login

Esta tela permite o acesso dos usuários ao sistema da Cantina Virtual, contendo campos simples de e-mail e senha, além do botão de "Entrar". Caso o usuário tenha esquecido sua senha, há um link para recuperação logo abaixo. O layout é clean, focado em facilitar o acesso rápido ao sistema, minimizando distrações e otimizando a usabilidade.

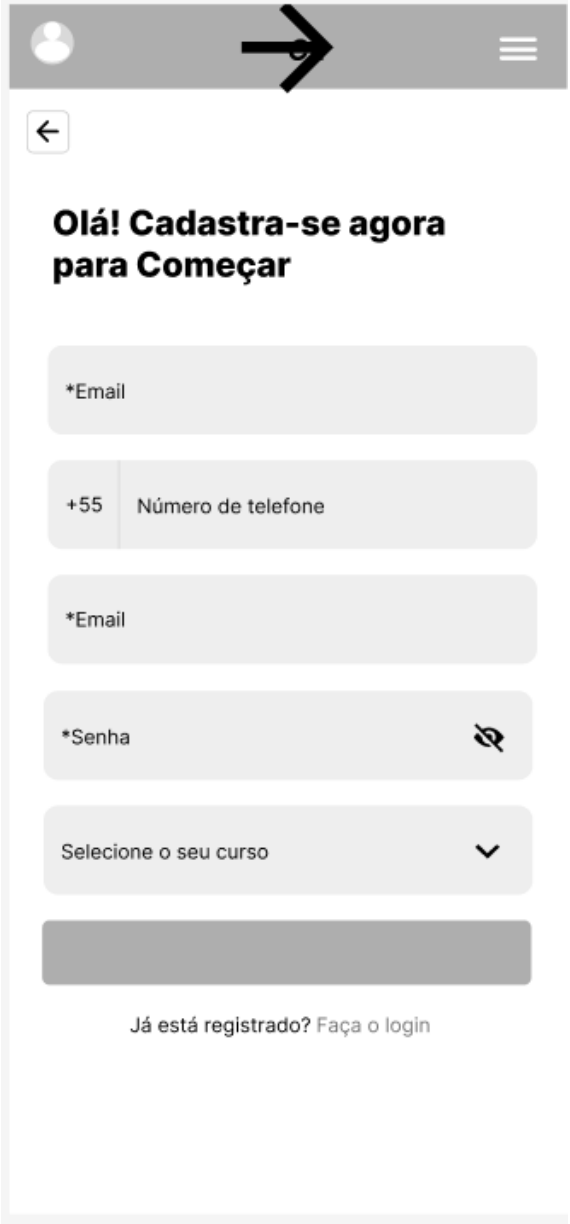
Figura 4 - Tela de Login (Mobile)

Fonte: Autoria Própria (2024)

4.5.4.1.2 Figma – Cadastro

A tela de cadastro foi projetada para que novos usuários possam se registrar no sistema. Ela inclui campos para inserir e-mail, nome completo e senha, além da opção de aceitar os termos e condições de uso. O layout segue a simplicidade da tela de login, priorizando a facilidade no preenchimento dos dados.

Figura 5 - Tela de Cadastro



Fonte: Autoria Própria (2024)

4.5.4.1.3 Figma - Lista de Produtos

Esta tela exibe uma lista de produtos disponíveis na cantina, com cada produto sendo apresentado com uma imagem (*placeholder*) e seu respectivo nome abaixo. O topo da tela inclui um menu de navegação com botões que possibilitam o acesso a diferentes categorias ou funcionalidades. Esta disposição permite que os usuários explorem facilmente os produtos oferecidos pela cantina.

Figura 6 - Lista de produtos (Mobile)

Fonte: Autoria Própria (2024)

4.5.4.1.4 Figma - Detalhes do Produto

Esta tela exibe detalhes específicos de um produto após a seleção do usuário, com nome, descrição, e opções para adicionar diferentes quantidades do item ao carrinho. Ela também permite ao usuário personalizar suas escolhas, como tamanho ou variação do produto, caso aplicável. A interface é focada em uma experiência simples e eficiente, garantindo que os usuários possam realizar suas escolhas de maneira clara.

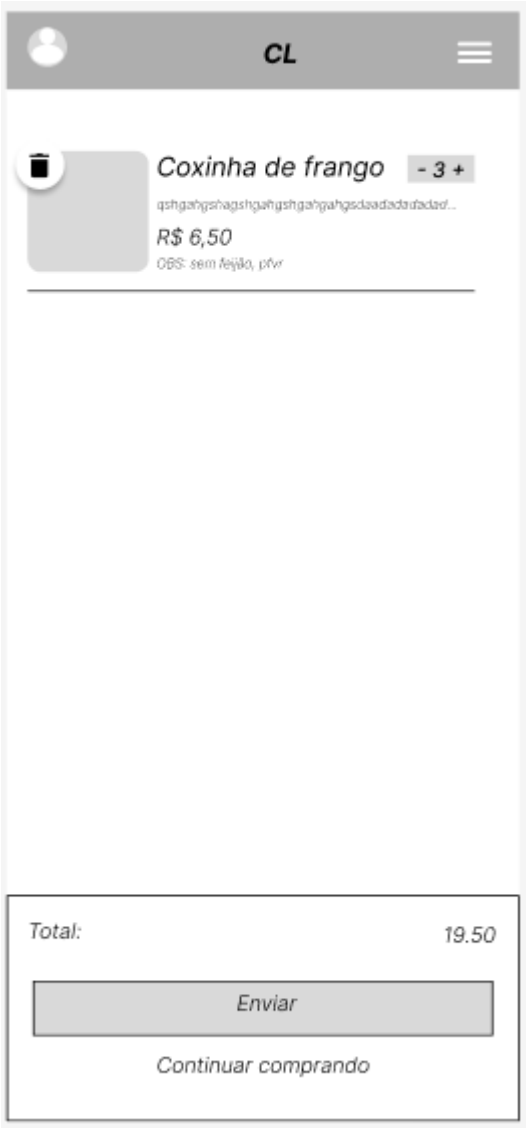
Figura 7 - Detalhes dos produtos

Fonte: Autoria Própria (2024)

4.5.4.1.5 Figma - Carrinho de Compras

O carrinho de compras apresenta os itens selecionados pelo usuário com a quantidade de cada produto. É possível alterar as quantidades diretamente nesta tela, além de visualizar o valor total do pedido. Um botão de "Finalizar Compra" é destacado para concluir a operação. Este layout é pensado para ser funcional e direto, facilitando ajustes e revisões antes da conclusão da compra.

Figura 8 - Carrinho de Produtos



Fonte: Autoria Própria (2024)

4.5.4.1.6 Figma - Adicionar Produto (Adm)

Esta tela foi projetada para que administradores possam adicionar novos produtos ao sistema. Ela contém campos para inserir o nome, preço e categoria do produto, além de um botão para carregar uma imagem representativa do item. A interface simples garante que a equipe administrativa possa adicionar novos itens rapidamente e sem complicações.

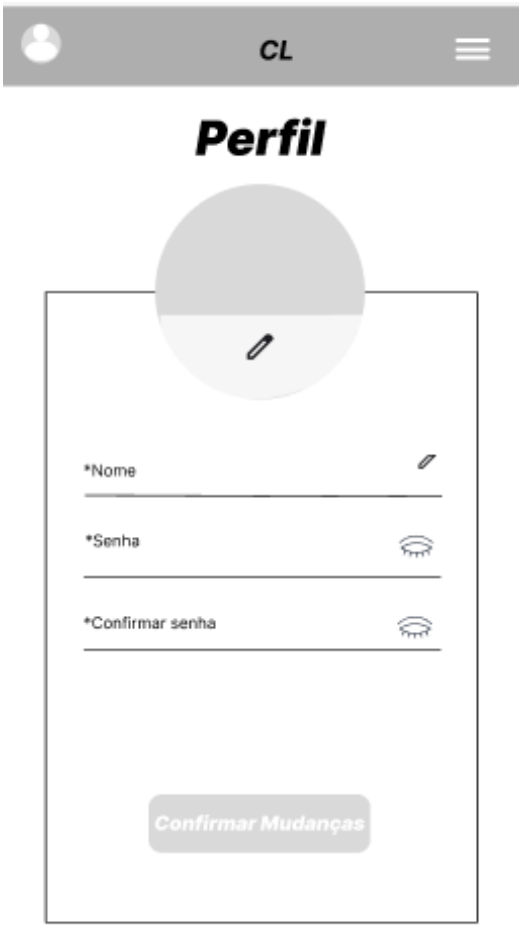
Figura 9 - Adicionar produto (Adm)

Fonte: Autoria Própria (2024)

4.5.4.1.7 Figma - Perfil

A tela de perfil foi desenvolvida no **Figma** com o objetivo de oferecer ao usuário uma maneira clara e intuitiva de visualizar e gerenciar suas informações pessoais, como nome e foto de perfil. Essa funcionalidade foi projetada para aprimorar a experiência do usuário no **Cantina Virtual**, promovendo personalização e controle sobre os dados.

Figura 10 - Perfil



Fonte: Autoria Própria (2024)

4.5.4.2 Implementação da Interface de Usuário

A interface de usuário do sistema Cantina Virtual foi projetada com foco em simplicidade, acessibilidade e facilidade de uso, tanto para os usuários finais (alunos) quanto para os administradores (funcionários da cantina). Durante o desenvolvimento, foram seguidas boas práticas de design, levando em consideração o feedback obtido na entrevista com as responsáveis da cantina.
Marcos Paiva (2024)

4.5.4.2.1 Design Responsivo

O sistema foi projetado para ser responsivo, permitindo que os usuários acessem o aplicativo em diversos dispositivos. Utilizou-se do CSS e media queries para garantir que os elementos da interface se ajustem corretamente em telas de diferentes tamanhos, proporcionando uma experiência consistente em qualquer dispositivo.
Marcos Paiva (2024)

4.5.4.2.2 Media Queries

Media Queries são funcionalidades do CSS que permitem adaptar o layout de um sistema a diferentes tamanhos de tela, como celulares, tablets e desktops. Eles fazem parte do conceito de design responsivo, essencial para oferecer uma ótima experiência de usuário, independentemente do dispositivo usado.

No caso da Cantina Virtual, o uso de Media Queries é crucial, pois o sistema será acessado por alunos, professores e funcionários através de diversos dispositivos. É importante que o sistema ofereça uma navegação eficiente, tanto em dispositivos móveis quanto em telas maiores.
Marcos Paiva (2024)

4.5.4.2.3 Mobile First

O Mobile First refere-se a uma abordagem de design e desenvolvimento onde a versão para dispositivos móveis é criada como prioridade, antes de adaptar o sistema para telas maiores. Esse método é essencial para a Cantina Virtual, pois a maioria dos alunos, professores e funcionários utilizaram celulares como primeira alternativa para acessar o sistema.
Gabriel Serra (2019)

4.5.4.2.4 Navegação Intuitiva

A navegação foi simplificada para que os usuários pudessem concluir suas tarefas em poucos passos. Menus e botões foram posicionados estrategicamente para que as funções principais, como realizar pedidos e visualizar o cardápio estivessem sempre acessíveis com poucos cliques.
Marcos Paiva (2024)

4.5.4.3 Componentes Visuais

A interface da Cantina Virtual foi desenvolvida utilizando bibliotecas visuais como Bootstrap e FontAwesome, garantindo um design moderno e funcional. Elementos como botões, ícones e menus de navegação foram cuidadosamente selecionados para tornar o sistema fácil de usar, mesmo para aqueles com pouca experiência em tecnologia. Promovendo uma melhor experiência para todos os usuários do sistema.

4.5.4.3.1 Bootstrap

O Bootstrap é um dos frameworks front-end mais utilizados para construir sites responsivos e com um design agradável. O Bootstrap oferece uma série de

componentes pré-prontos, como botões, formulários, tabelas e menus de navegação, que foram personalizados para atender às necessidades específicas do sistema da cantina. Isso garantiu uma aparência moderna e facilitou o trabalho dos desenvolvedores na criação de uma experiência simples para os usuários.

4.5.4.3.2 FontAwesome

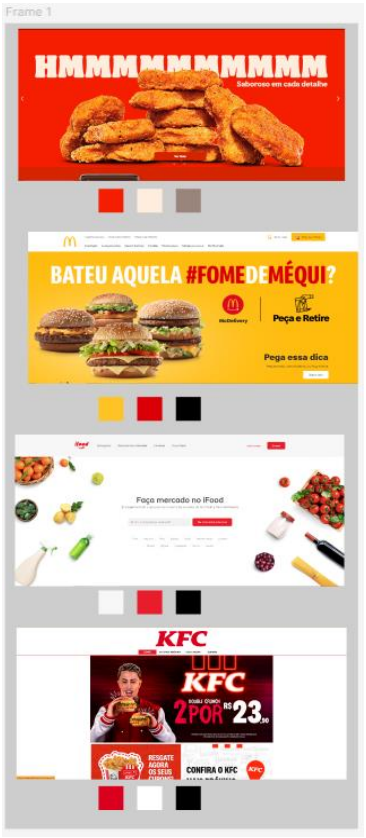
O *FontAwesome* foi utilizado para integrar ícones na interface da Cantina Virtual, melhorando a comunicação visual. Essa biblioteca de ícones foi escolhida devido à sua extensa variedade de opções, desde simples ícones de navegação até símbolos mais específicos como carrinhos de compras ou configurações. Além disso, a flexibilidade do *FontAwesome* em termos de customização permitiu ajustar o estilo dos ícones para harmonizar com o design geral da interface.
Guilherme Lima (2023)

4.5.4.4 Paleta de Cores e Acessibilidade

A escolha da paleta de cores foi baseada tanto em fatores estéticos já estabelecidos originalmente pela Cantina da Lu, quanto na psicologia das cores, buscando criar uma interface que não apenas fosse visualmente agradável, mas que também chamasse a atenção dos usuários no sistema Cantina Virtual.

4.5.4.4.1 Paleta de Cores

Figura 11 - Protótipo com cores



Fonte: Autoria Própria (2024)

4.5.4.4.2 Cores Utilizadas

A seleção de cores na Cantina Virtual segue a lógica utilizada em muitos estabelecimentos de fast food, onde as cores são escolhidas para provocar emoções e comportamentos específicos. Redes de fast food tendem a usar vermelho e amarelo, pois essas cores estão associadas ao apetite e ao desejo de tomar decisões, criando um ambiente onde os consumidores são incentivados a fazer pedidos.

Vermelho (#CF2226): É uma cor que influência e promove a sensação de consumo no público.
Amarelo (#FFBB0E): Estimula a fome e atrai a atenção dos consumidores.
Luciana. (11 de dezembro de 2015). Psicologia das cores.

5 DESCRIÇÃO GERAL DO SISTEMA

5.1 Visão Geral do Sistema

O sistema Cantina Virtual foi desenvolvido para facilitar o processo de pedidos de alimentos e bebidas na cantina da escola SENAI, tanto para os clientes quanto para os administradores. Ele oferece uma experiência organizada e eficiente para a realização de pedidos, além de uma interface administrativa para o gerenciamento de produtos e acompanhamento de pedidos.

Ao acessar o sistema, o cliente tem acesso imediato ao menu de produtos disponíveis, como salgados, bebidas e doces. No entanto, para adicionar produtos ao carrinho, é necessário que o usuário se cadastre e faça login. O cadastro exige informações como nome, telefone, e-mail, senha e curso do SENAI.

Após o login, o usuário pode adicionar produtos ao carrinho. O sistema permite escolher a quantidade desejada de cada item (por exemplo, 3 salgados), combinar diferentes produtos em um único pedido (por exemplo, 1 salgado e 2 sucos), e adicionar comentários específicos (como “1 suco sem açúcar”). Quando todos os produtos desejados forem adicionados ao carrinho, o pedido pode ser enviado.

Assim que o pedido for enviado, o cliente aguarda até que a equipe da cantina confirme o status do pedido (em preparação, pronto ou cancelado). O pagamento é feito no balcão, pois o sistema não oferece a opção de pagamento online. O usuário também pode personalizar seu perfil no sistema, incluindo a troca de foto de perfil, nome e senha.

Para os administradores, existe uma página específica de administração. O login de administrador já está pré-cadastrado no sistema e só pode ser acessado com credenciais específicas. Uma vez logados, os administradores podem visualizar todos os pedidos recebidos e atualizar o status de cada pedido conforme ele avança, podendo marcá-lo como “pendente”, “em processo”, “pronto”, ou cancelar, caso necessário. Além disso, os administradores podem adicionar novos produtos

ao menu, informando nome, descrição, preço, imagem e categoria (salgado, bebida, doce, etc.). Eles também podem editar ou desativar produtos caso um item acabe ou não esteja disponível naquele dia.

O sistema foi projetado para ser prático tanto para os clientes quanto para a administração da cantina, garantindo agilidade e controle nos pedidos.

5.2 Marmitas

Na parte de reservas de marmitas, o usuário terá que realizar um processo um pouco diferente em relação aos outros produtos. Primeiro, o cliente vai ter acesso a todas as opções de marmitas presentes no menu, e depois de logado, ele pode escolher uma marmita específica e personalizá-la, adicionando ou removendo acompanhamentos de acordo com suas preferências e disponibilidade de produtos. Em seguida, o cliente adiciona a marmita ao seu carrinho virtual e, quando estiver pronto, finaliza o pedido, que é enviado diretamente para o sistema da cantina.

Na parte administrativa, as responsáveis pela cantina visualizarão o que foi reservado, incluindo as personalizações escolhidas por cada cliente, o que permite que elas preparem as marmitas de acordo com o pedido específico. Além disso, elas também poderão adicionar novas marmitas ao menu, podendo incluir guarnições em cada nova adição, além das informações básicas presentes nos demais produtos. Dessa forma, o sistema agiliza o atendimento, organizando o processo tanto para o cliente quanto para a administração.

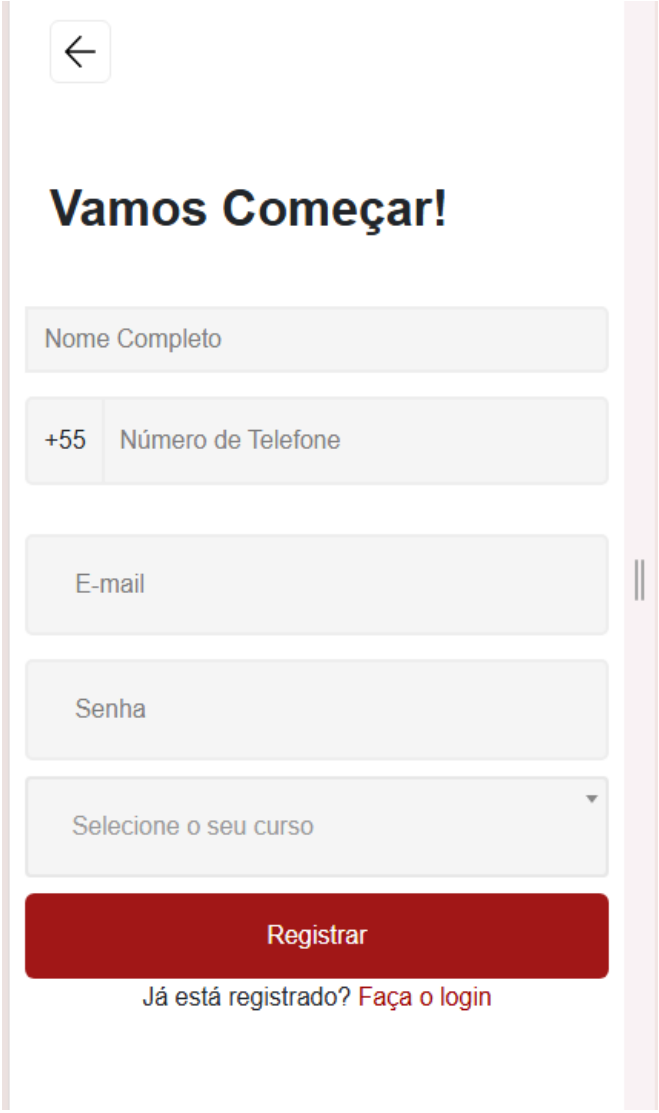
5.3 Funcionalidades Principais

5.3.1 Cadastro

O cadastro de usuário no sistema Cantina Virtual permite que novos clientes registrem suas informações pessoais para acessar as funcionalidades oferecidas. Para isso, o usuário preenche um formulário com nome completo, e-mail, número de telefone, senha e curso. Após a validação desses dados, o sistema os armazena no banco de dados, garantindo a criptografia da senha para maior segurança. O

cadastro habilita o acesso ao sistema, permitindo que o cliente utilize as credenciais fornecidas para login e navegação.

Figura 12 - Cadastro

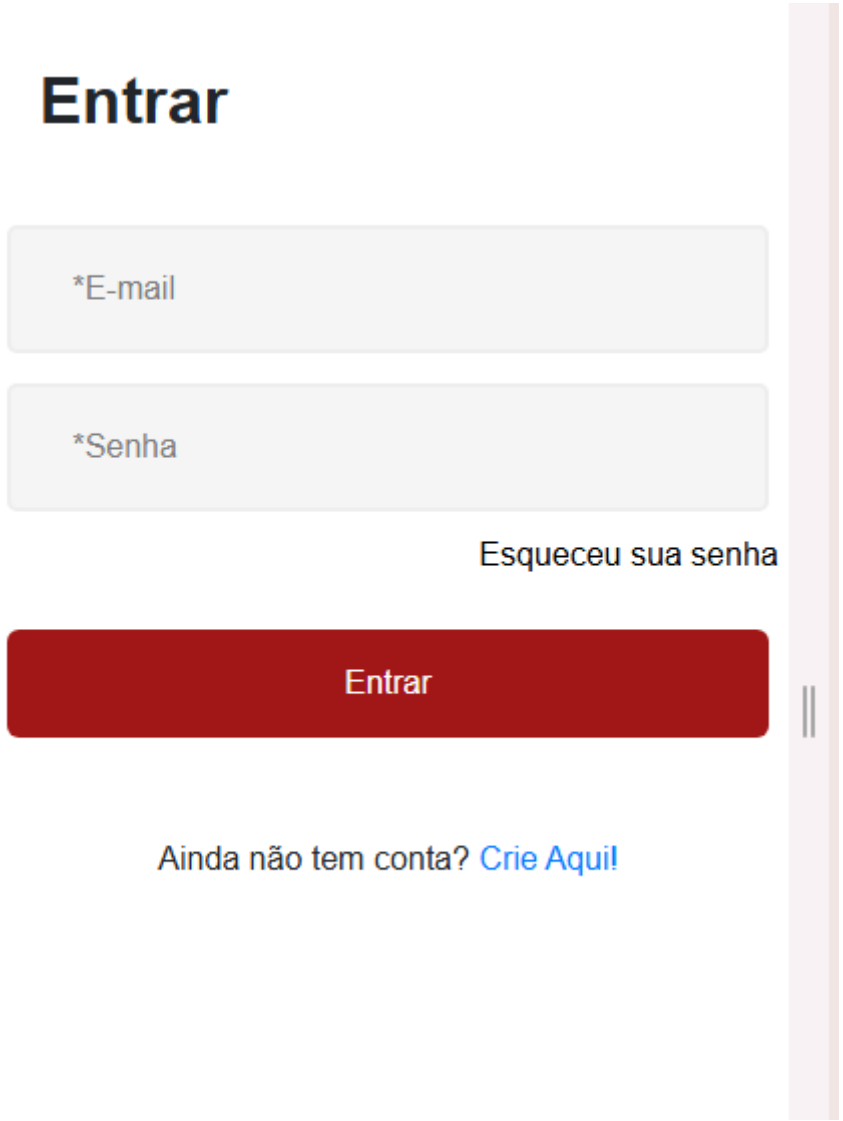


Fonte: Autoria Própria (2024)

5.3.2 Login

O sistema do Cantina Virtual permite que usuários previamente cadastrados acessem suas contas utilizando credenciais válidas. Para realizar o login, o usuário insere o e-mail e a senha cadastrados na tela apropriada. O sistema valida as informações fornecidas e, caso sejam corretas, concede acesso as funcionalidades disponíveis. Após a autenticação, uma mensagem de sucesso é exibida.

Figura 13 - Login



Fonte: Autoria Própria (2024)

5.3.3 Adicionar ao Carrinho

O sistema permite que o usuário adicione produtos ao carrinho de compras. Ao navegar pela lista de produtos ou página de um produto específico, o usuário pode selecionar a quantidade desejada e confirmar a adição. O sistema verifica a disponibilidade do item em estoque e atualiza o carrinho, exibindo os produtos, suas quantidades e o valor total. Essa funcionalidade depende que o usuário esteja logado e o cadastro prévio do produto no sistema.

Figura 14 - Adicionar ao Carrinho

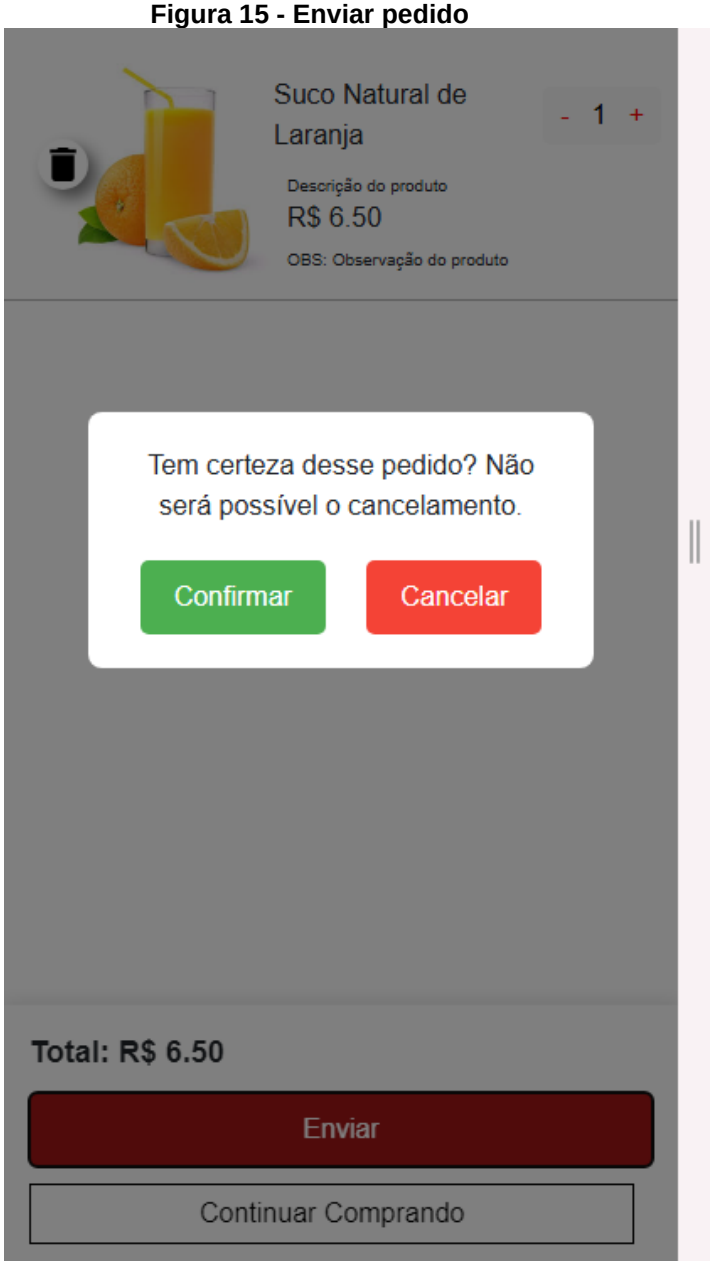
Fonte: Autoria Própria (2024)

5.3.4 Enviar Pedido

O sistema possibilita que o usuário finalize suas compras enviando os itens do carrinho para processamento. Para isso, é necessário que pelo menos um item esteja no carrinho e que o usuário esteja logado. Após a confirmação do pedido, o

sistema registra as informações, como produtos, quantidades e valores. Um número de pedido é gerado com um status de pendente, para que os administradores da cantina possam começar o seu preparo.

Figura 15 - Enviar pedido



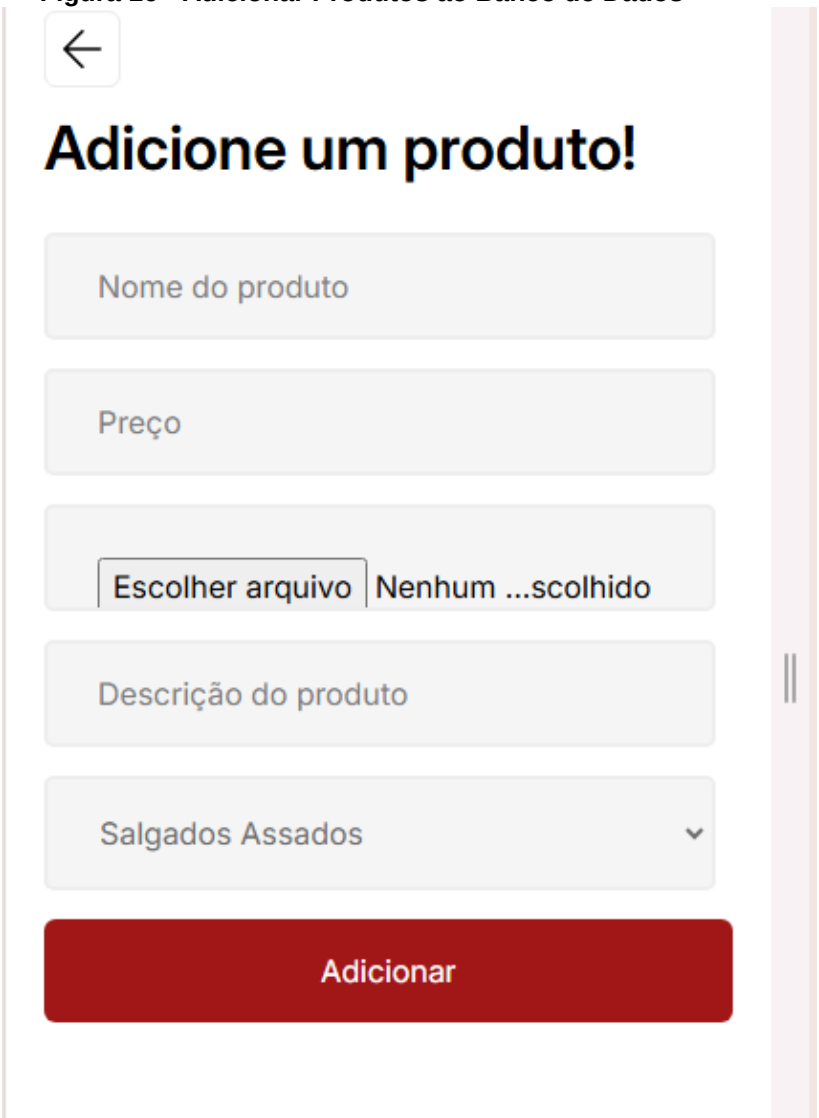
Fonte: Autoria Própria (2024)

5.3.5 Adicionar Produtos ao Banco de Dados

Administradores do sistema podem cadastrar novos produtos no catálogo, informando nome, descrição, preço, categoria e imagem. O sistema armazena essas informações no banco de dados, gerando um ID único para cada item. Após o

cadastro, o produto é exibido no catálogo, disponível para consulta e adição ao carrinho.

Figura 16 - Adicionar Produtos ao Banco de Dados



Fonte: Autoria Própria (2024)

5.3.6 Remover Produtos do Banco de Dados

Administradores podem excluir produtos do catálogo quando necessário. Para isso, devem estar logados no sistema com o perfil adequado. A funcionalidade permite a desabilitação do produto, incluindo imagens e dados associados.

Figura 17 - Remover Produtos do Banco de Dados

Fonte: Autoria Própria (2024)

5.3.7 Status do Pedido

O sistema oferece ao usuário a possibilidade de acompanhar o status dos seus pedidos. O cliente pode consultar detalhes, como data do pedido, produtos, valores e o estado atual (em processamento, enviado, entregue). Essas informações são atualizadas de acordo com o preparo do pedido, garantindo precisão e transparência no acompanhamento.

Figura 18 - Status do Pedido

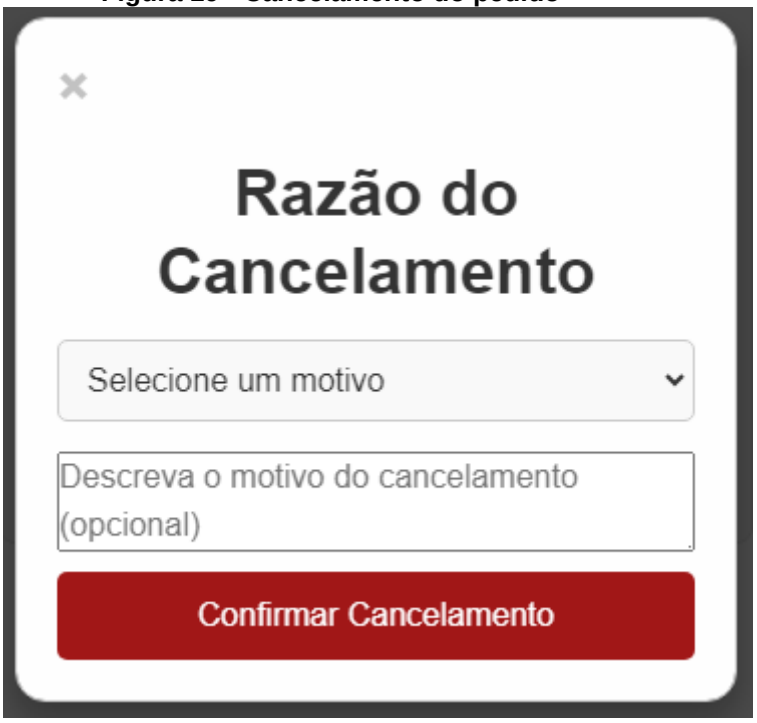
Fonte: Autoria Própria (2024)

5.3.8 Cancelamento de Pedidos

O sistema permite que o administrador cancele pedidos, desde que esteja logado e o pedido não tenha sido concluído. Ele terá que informar o motivo do cancelamento do pedido. Ao confirmar o cancelamento, o status do pedido é

removido da interface do cliente. O banco de dados atualiza as informações do pedido.

Figura 19 - Cancelamento do pedido

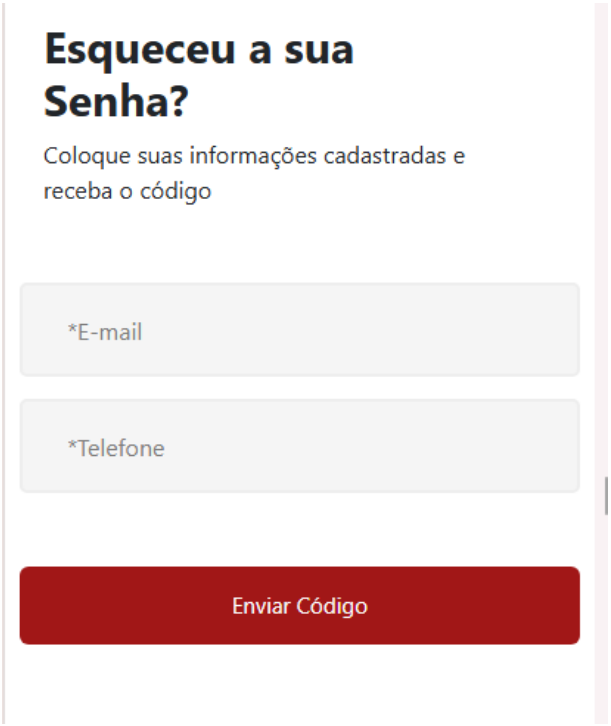


Fonte: Autoria Própria (2024)

5.3.9 Esqueceu Sua Senha

Usuários que esquecerem suas senhas podem alterá-las pelo sistema. Antes do login, o usuário pode acessar a funcionalidade de redefinição, onde colocará o seu e-mail e telefone, assim podendo redefinir sua senha. O sistema valida e atualiza os dados no banco, permitindo que a nova senha seja usada nos próximos acessos.

Figura 20 - Esqueceu sua senha

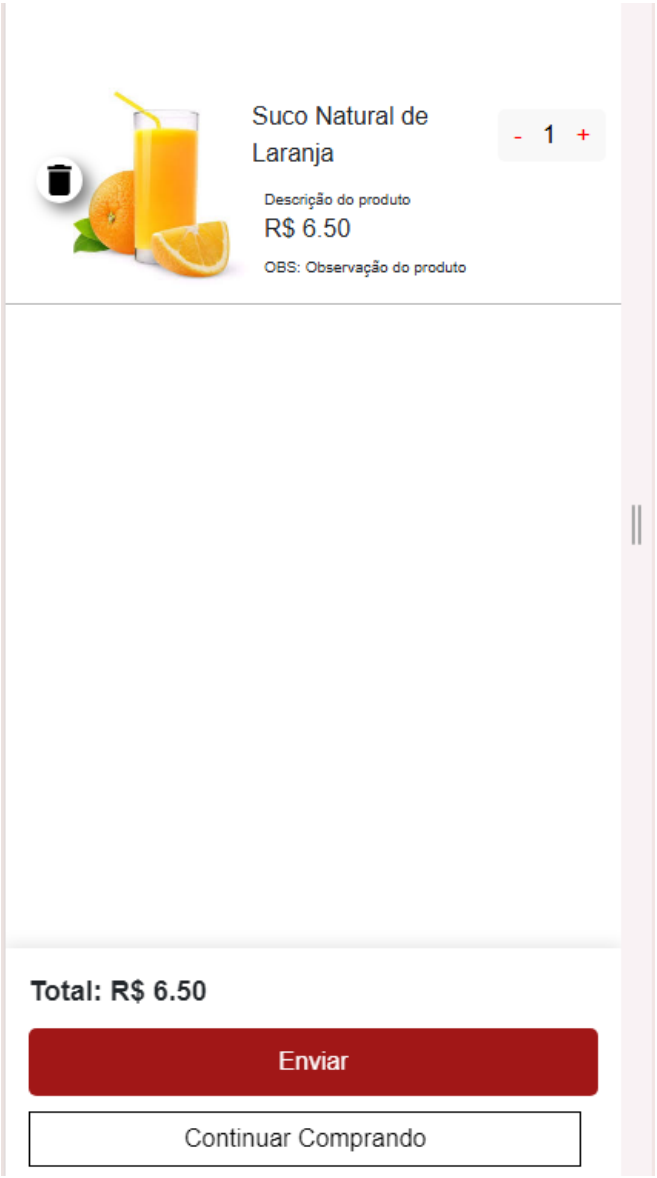


Fonte: Autoria Própria (2024)

5.3.10 Exibir Carrinho

O sistema permite que o usuário visualize o conteúdo do seu carrinho de compras, incluindo itens adicionados, quantidades, valores unitários e o total da compra. O usuário pode acessar essa funcionalidade a qualquer momento para revisar ou ajustar os produtos antes de finalizar o pedido.

Figura 21 - Exibir Carrinho



Fonte: Autoria Própria (2024)

5.3.11 Excluir item do carrinho

O sistema permite que o cliente remova itens específicos de seu carrinho de compras. Essa funcionalidade atualiza automaticamente a exibição do carrinho, ajustando o total de preços para refletir a exclusão.

Figura 22 - Excluir item do carrinho

Fonte: Autoria Própria (2024)

5.3.12 Exibir produtos

Os clientes podem visualizar uma lista de produtos disponíveis no sistema, com informações básicas, como nome, imagem e preço. Essa funcionalidade possibilita que o cliente explore o catálogo de maneira eficiente e selecione itens para mais detalhes ou para adicionar ao carrinho.

Figura 23 - Exibir Produtos

Fonte: Autoria Própria (2024)

5.3.13 Exibir produto único

O cliente pode acessar os detalhes completos de um produto específico ao selecioná-lo. A funcionalidade exibe informações como descrição, imagens e preço, permitindo que o cliente tome decisões informadas antes de realizar a compra.

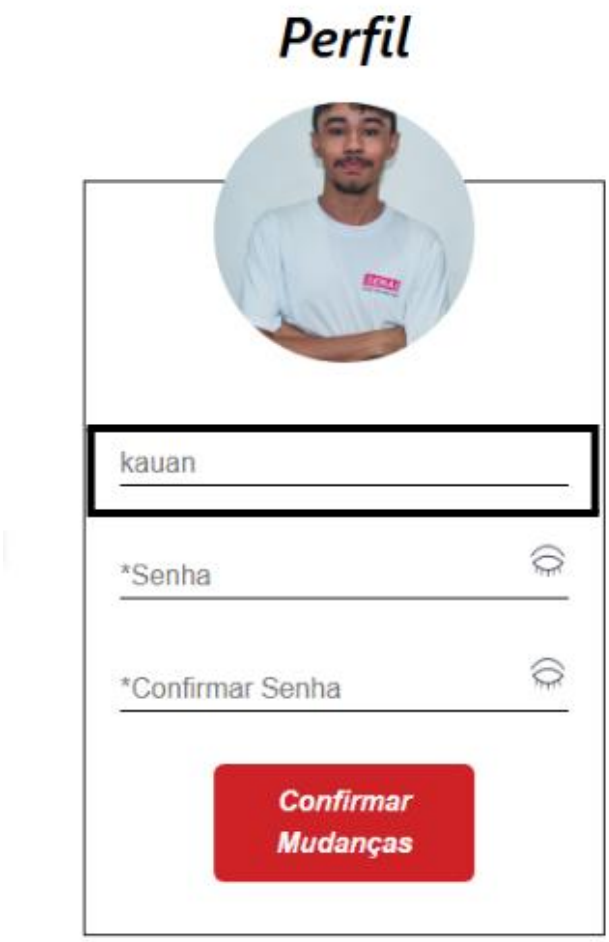
Figura 24 - Exibir produto único

Fonte: Autoria Própria (2024)

5.3.14 Modificação de Nome do Perfil

O sistema oferece ao cliente a opção de alterar o nome associado ao seu perfil. Após a modificação, o novo nome é refletido em todas as seções do sistema onde é exibido, garantindo uma atualização uniforme e personalizada.

Figura 25 - Modificação de Nome do Perfil



Fonte: Autoria Própria (2024)

5.3.15 Confirmação de Senha do Perfil

Os clientes tem que confirmar sua senha se quiserem alterar seu nome ou foto de perfil. Eles devem acessar sua página de perfil, e logo após alterarem seu nome ou foto, devem confirmar sua senha para que a alteração seja efetuada de maneira correta.

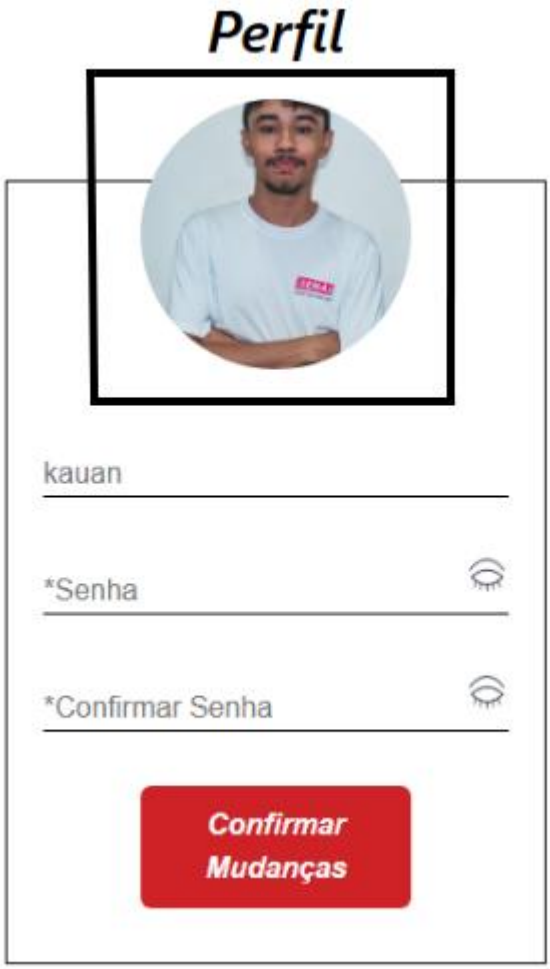
Figura 26 - Confirmação de Senha do Perfil

Fonte: Autoria Própria (2024)

5.3.16 Modificação de Foto do Perfil

O cliente pode atualizar sua foto de perfil fazendo o upload de uma nova imagem. Após a modificação, a nova foto é exibida em todas as áreas do sistema associadas ao perfil, promovendo uma identidade visual atualizada.

Figura 27 - Modificação de Foto do Perfil



Fonte: Autoria Própria (2024)

5.3.17 Histórico de Pedidos

O sistema exibe para o cliente o histórico de pedidos realizados, com detalhes como data, status, produtos adquiridos, total pago e outras informações. Essa funcionalidade permite ao cliente revisar compras passadas e acompanhar o progresso de pedidos em andamento.

Figura 28 - Histórico de Pedidos

Fonte: Autoria Própria (2024)

5.3.18 Exibir pedidos

Os administradores podem visualizar uma lista com todos os pedidos realizados no sistema, incluindo informações resumidas, como nome do cliente, status e valor total.

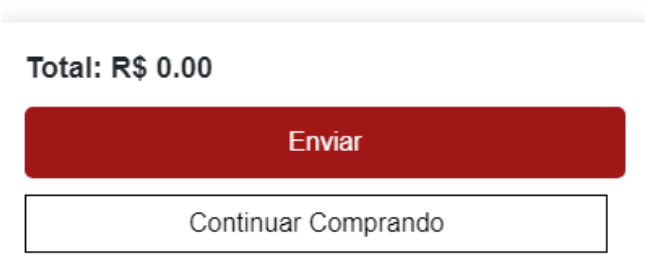
Figura 29 - Exibir pedidos

Fonte: Autoria Própria (2024)

5.3.19 Apagar carrinho após o envio do pedido

Após a finalização de um pedido, o sistema esvazia automaticamente o carrinho de compras do cliente, removendo todos os itens para liberar o espaço para novos pedidos.

Figura 30 - Apagar Carrinho Após o Envio do Pedido

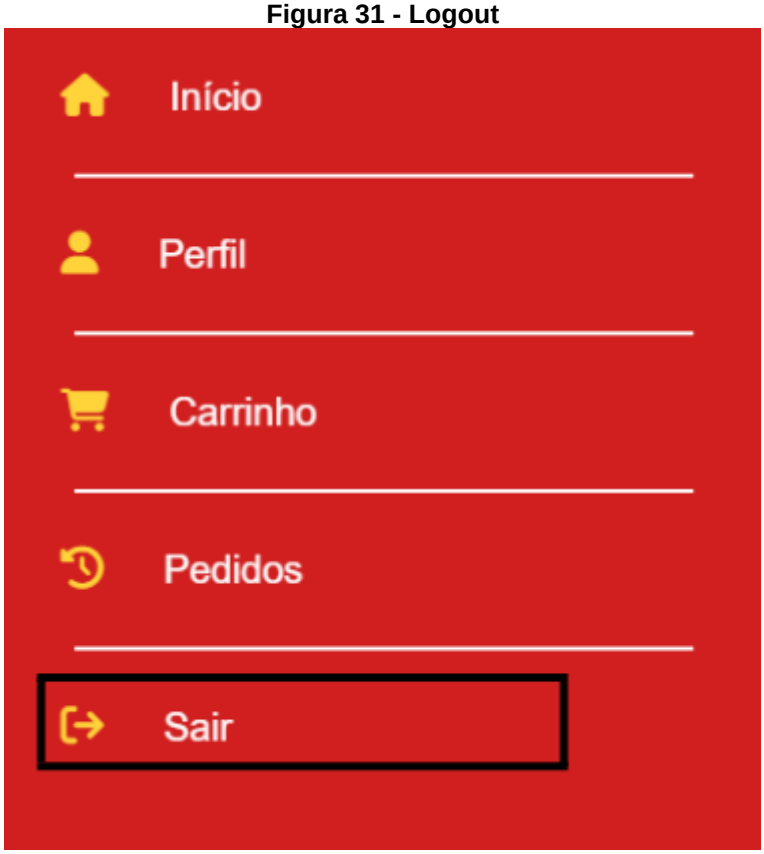


Fonte: Autoria Própria (2024)

5.3.20 Logout

O cliente pode encerrar sua sessão no sistema de forma segura. Após o logout, o cliente precisa fazer login novamente para acessar as funcionalidades de pedido.

Figura 31 - Logout



Fonte: Autoria Própria (2024)

5.3.21 Apagar carrinho após o logout

Ao realizar o logout, o sistema apaga automaticamente o conteúdo do carrinho de compras do cliente. Dessa forma, qualquer item selecionado anteriormente é descartado, garantindo que o carrinho esteja limpo em um novo login.

5.3.22 Editar produto

O administrador pode editar informações de produtos cadastrados, como nome, preço, descrição e imagens. Essa funcionalidade permite atualizações rápidas e eficientes no catálogo, garantindo informações sempre precisas.

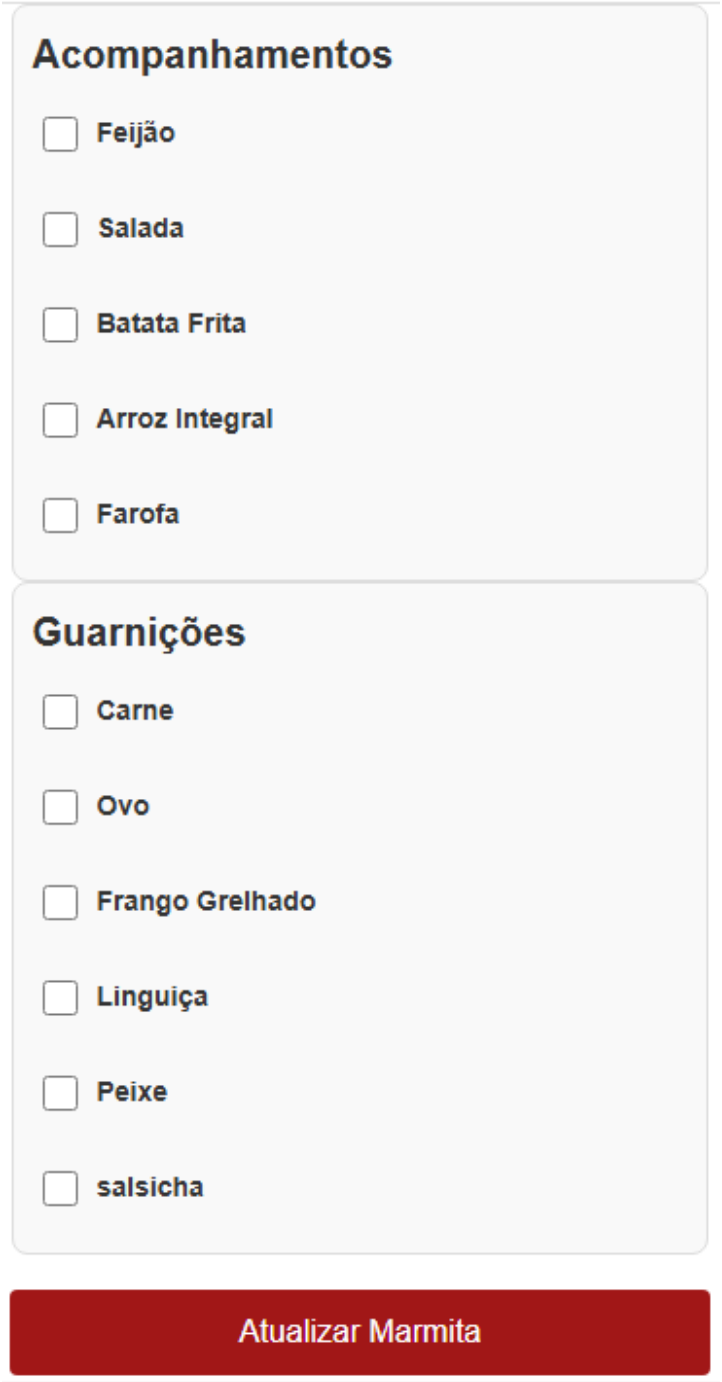
Figura 32 - Editar Produto

Fonte: Autoria Própria (2024)

5.3.23 Habilitar/Desabilitar guarnição

O administrador pode habilitar ou desabilitar guarnições disponíveis no cardápio de marmitas, permitindo maior controle sobre os acompanhamentos que podem ser selecionados pelos clientes no momento do pedido.

Figura 33 - Habilitar/Desabilitar Guarnição



Fonte: Autoria Própria (2024)

5.3.24 Exibir marmita

O cliente pode visualizar as marmitas disponíveis no cardápio, com informações completas, como nome, guarnições, preço e imagens. Essa funcionalidade facilita a escolha da marmita ideal de acordo com as preferências do cliente.

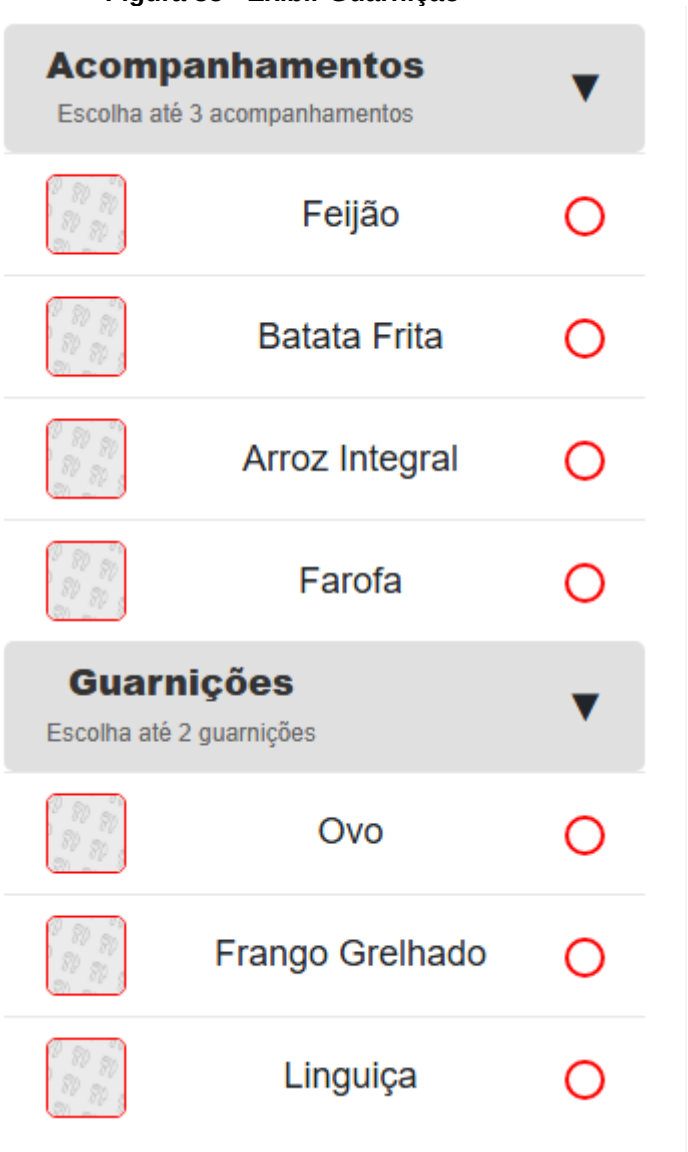
Figura 34 - Exibir Marmita

Fonte: Autoria Própria (2024)

5.3.25 Exibir guarnição

Ao selecionar uma marmita, o cliente pode visualizar a lista de guarnições disponíveis, podendo escolher uma ou mais opções para complementar seu pedido. Isso proporciona maior personalização no momento da compra.

Figura 35 - Exibir Guarnição



Fonte: Autoria Própria (2024)

5.3.26 Exibir relatório

Permite que o administrador gere um relatório detalhado de vendas em um período específico, escolhendo o intervalo de datas desejado.

Figura 36 - Exibir Relatório

Fonte: Autoria Própria (2024)

5.4 Testes de usabilidade

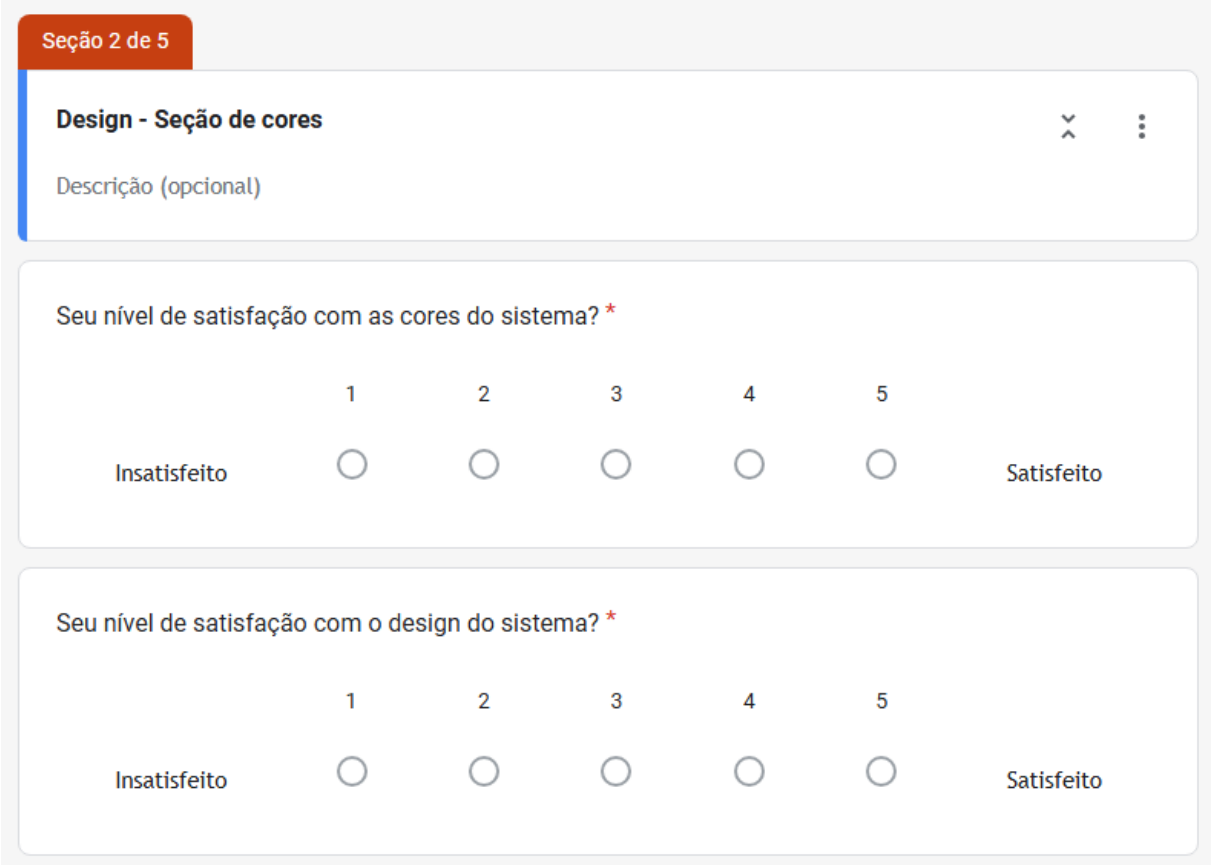
Durante o desenvolvimento do sistema, nós realizamos diversos testes de usabilidade para poder validado por completo. O objetivo foi garantir que a experiência do usuário fosse intuitiva e eficiente, identificando possíveis melhorias tanto no aspecto visual quanto no funcionamento das funcionalidades. Para isso, utilizamos o Google Forms para criar um formulário, que foi enviado aos usuários para coleta de feedback.

O formulário continha perguntas específicas sobre a navegação do sistema, a clareza das informações, a facilidade de uso das funcionalidades, além da satisfação geral com a interface e a experiência proporcionada. As respostas obtidas foram analisadas para realizar ajustes necessários e garantir que o sistema atendesse às expectativas dos usuários de forma otimizada. Com esses testes, conseguimos validar e aprimorar o sistema, tornando-o mais acessível e funcional para os usuários finais.

Figura 37 - Introdução

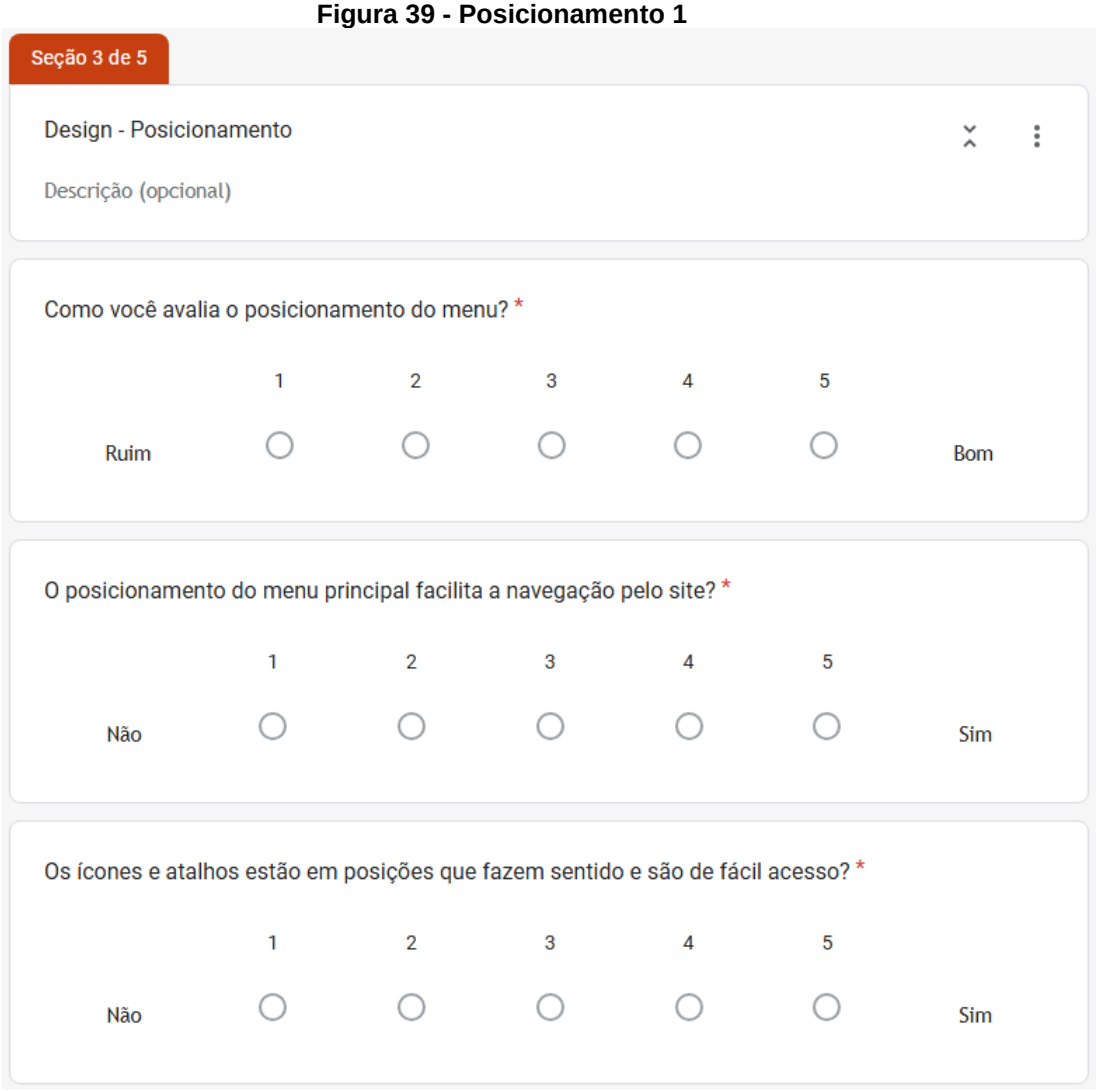
Fonte: Autoria Própria (2024)

Figura 38 - Seção de cores



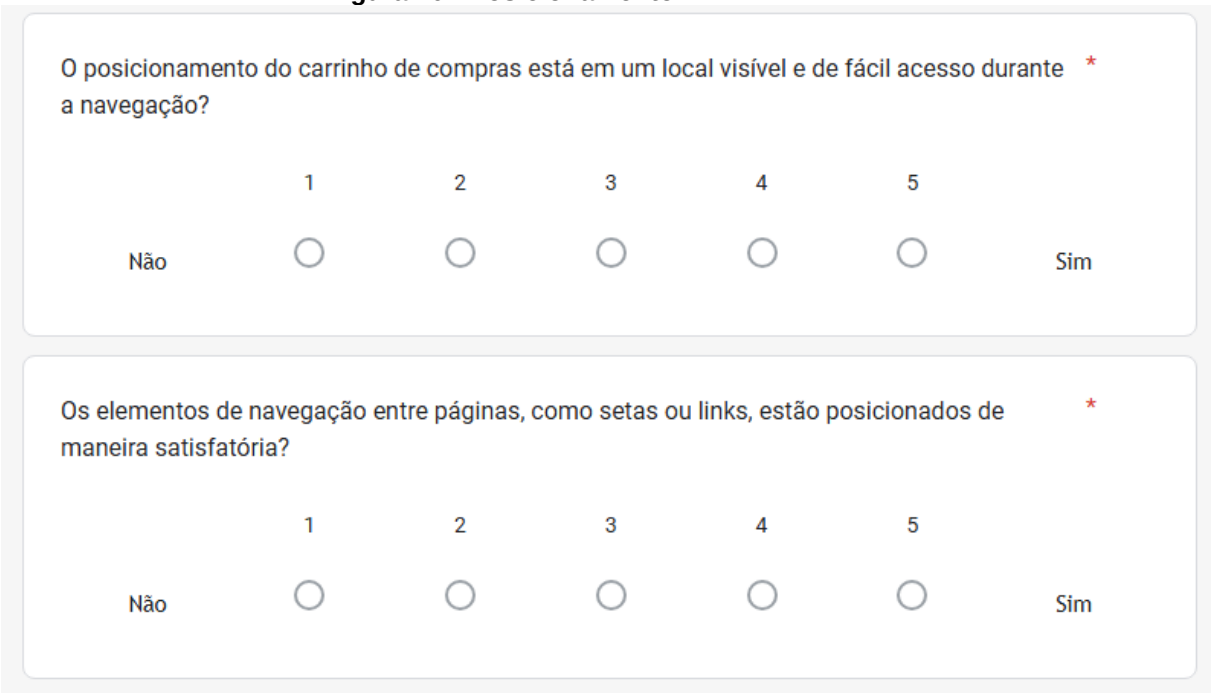
Fonte: Autoria Própria (2024)

Figura 39 - Posicionamento 1



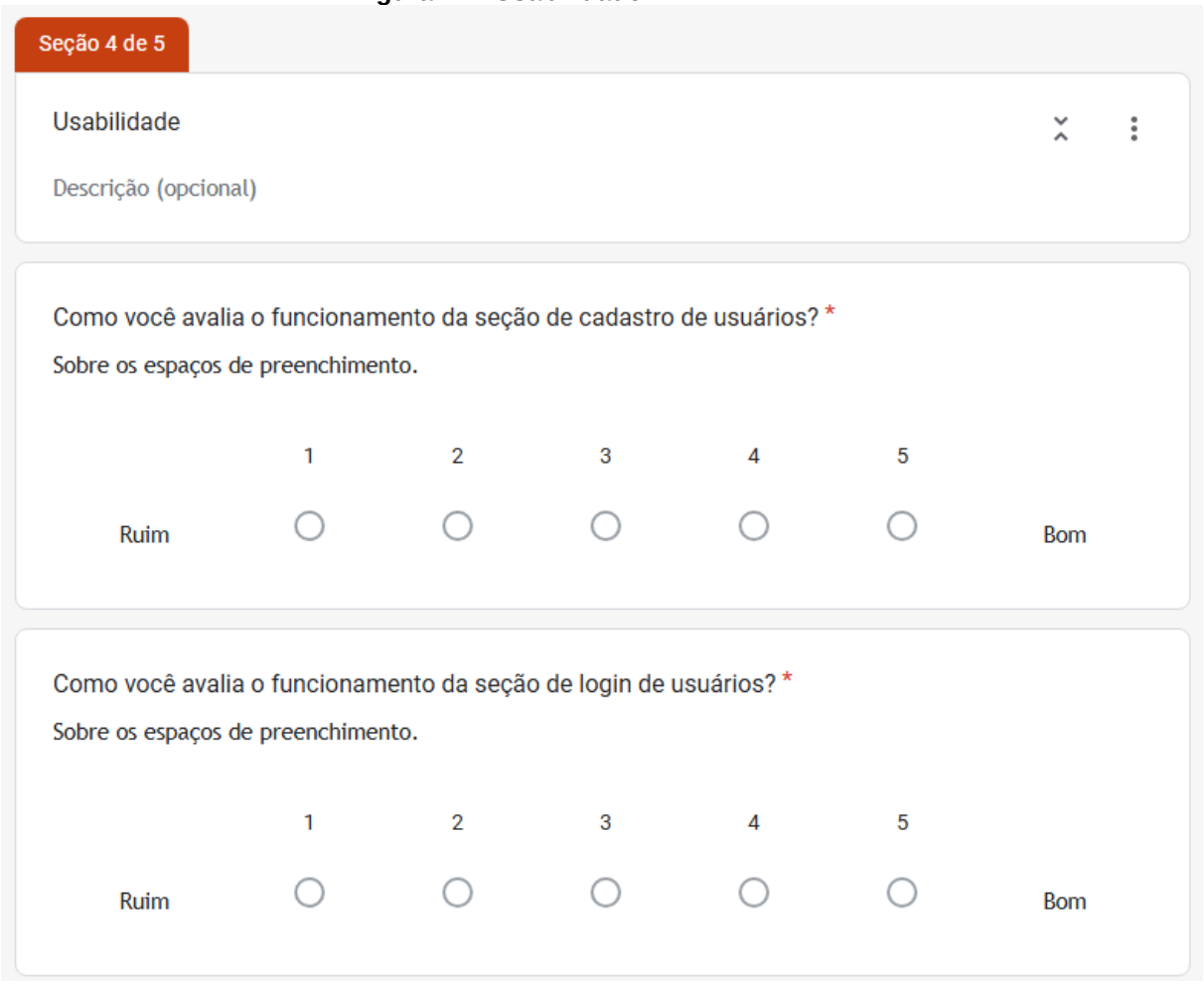
Fonte: Autoria Própria (2024)

Figura 40 - Posicionamento 2



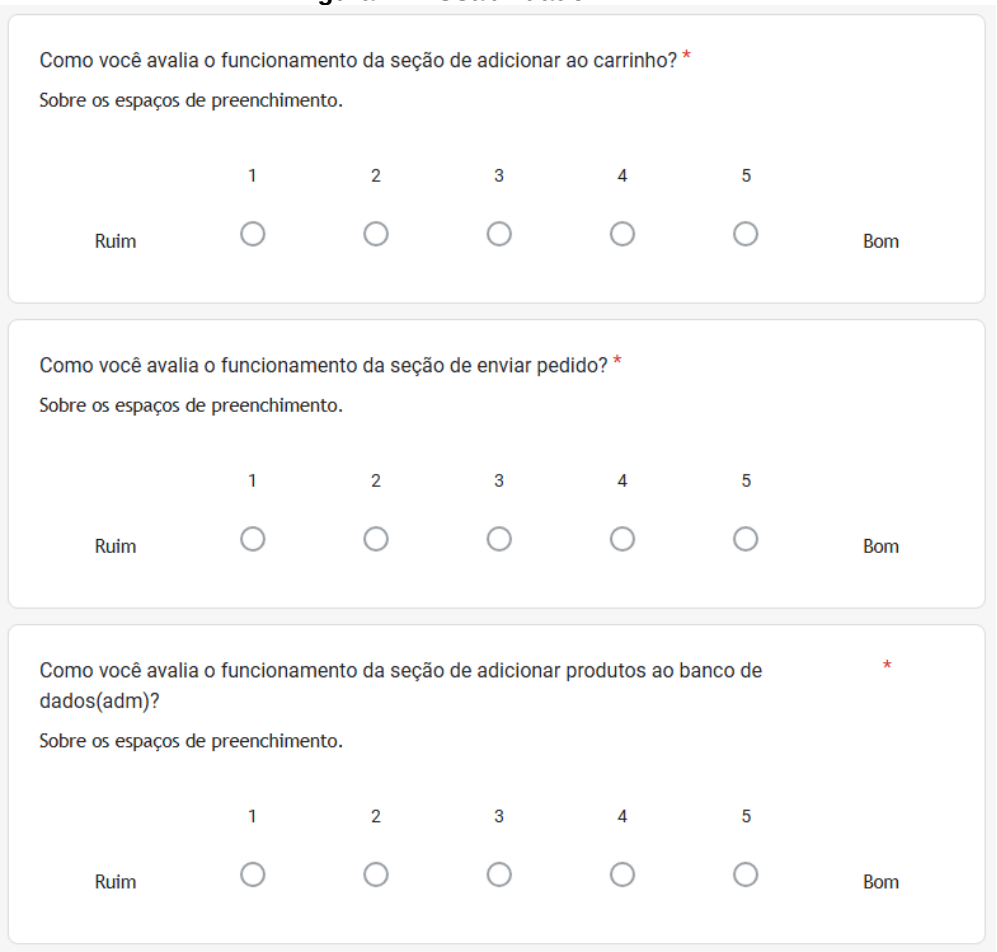
Fonte: Autoria Própria (2024)

Figura 41 - Usabilidade 1



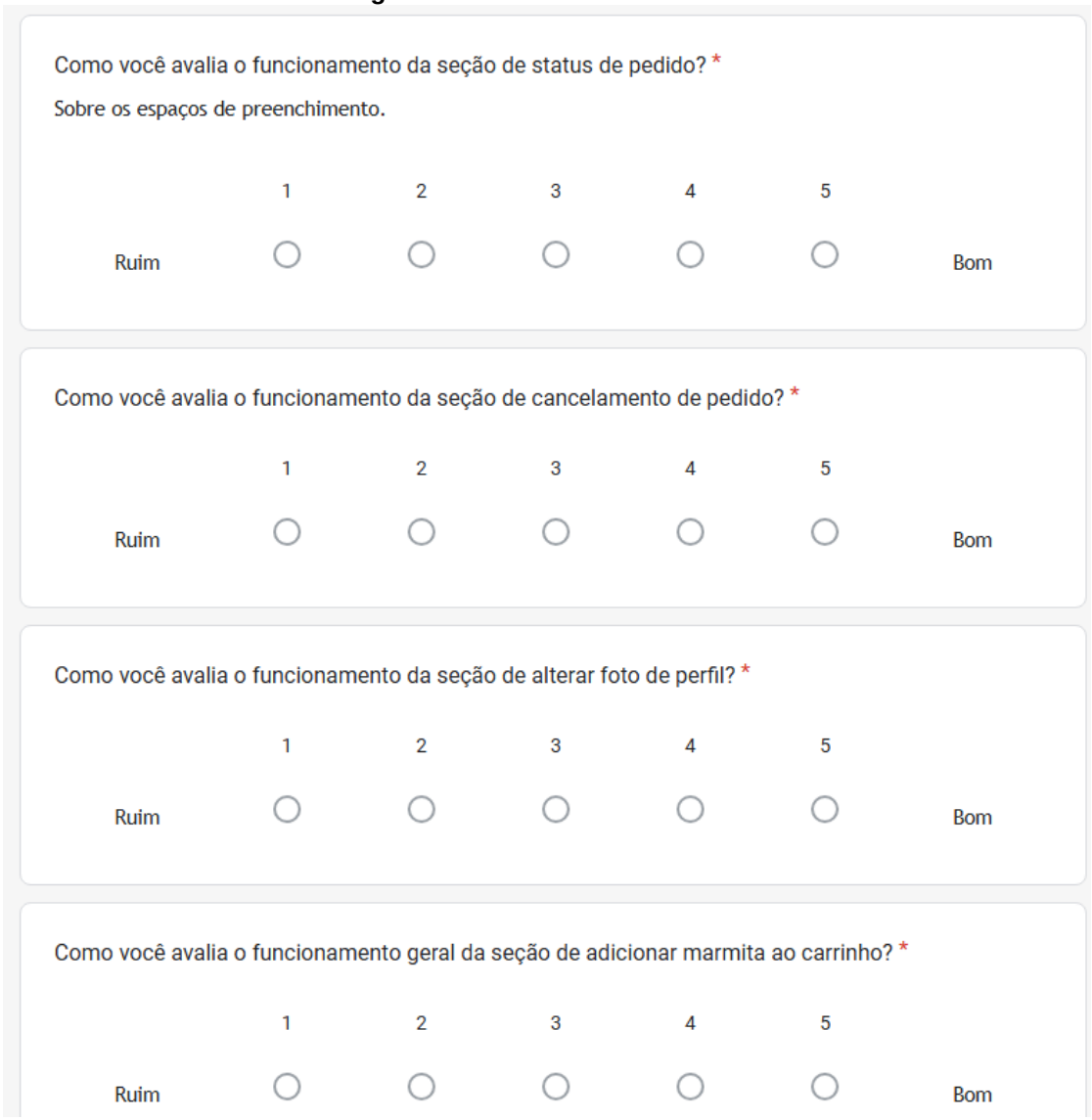
Fonte: Autoria Própria (2024)

Figura 42 - Usabilidade 2



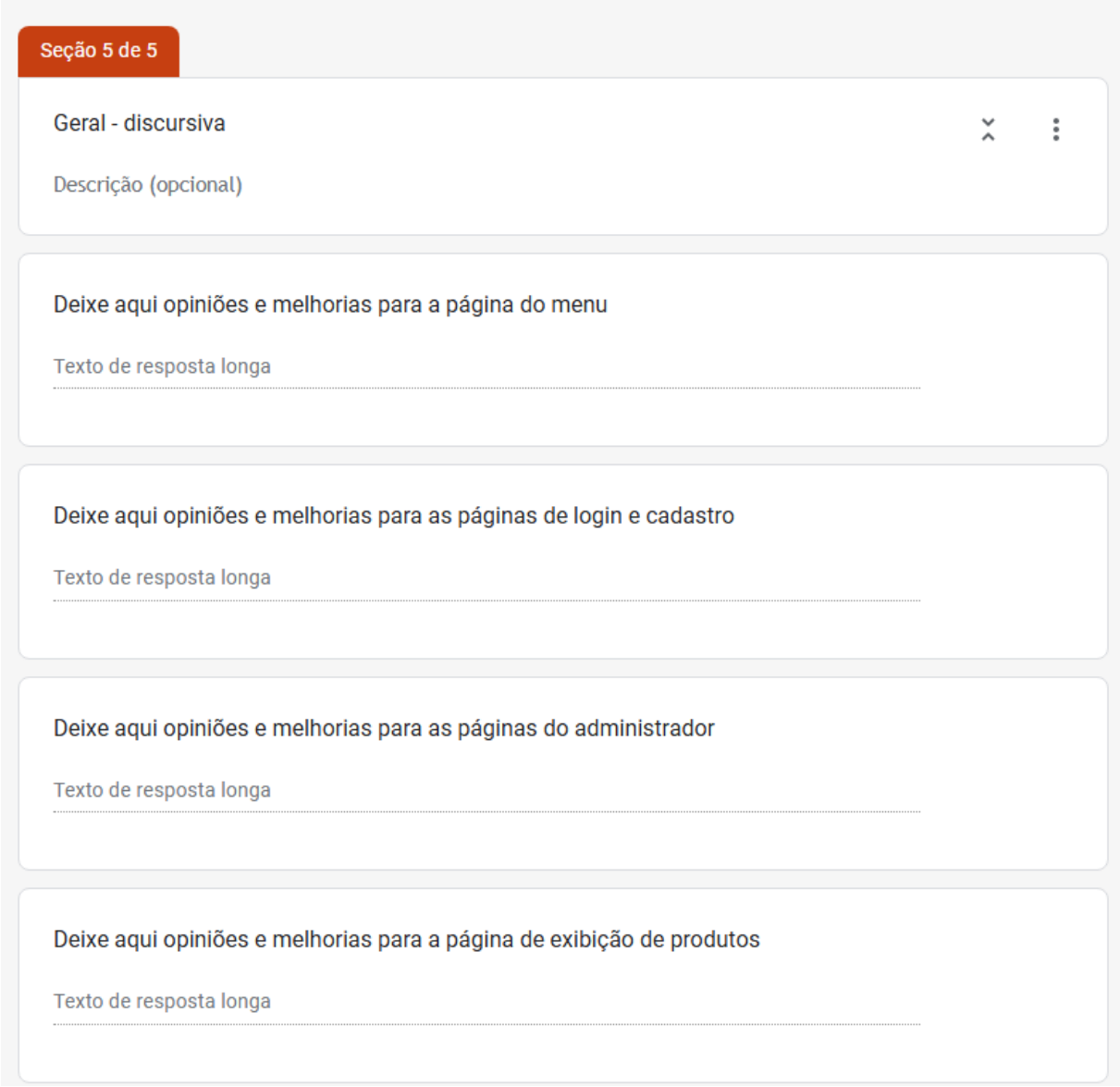
Fonte: Autoria Própria (2024)

Figura 43 - Usabilidade 3



Fonte: Autoria Própria (2024)

Figura 44 - Geral 1



Fonte: Autoria Própria (2024)

Figura 45 - Geral 2

Fonte: Autoria Própria (2024)

6 CONCLUSÃO

A conclusão deste trabalho reflete sobre os objetivos propostos e os resultados alcançados durante o desenvolvimento do sistema da Cantina Virtual. O objetivo geral de criar um sistema capaz de otimizar o trabalho das funcionárias da cantina e reduzir as filas foi cumprido com êxito. O projeto demonstrou eficiência ao proporcionar um sistema organizado, com todas as funcionalidades necessárias para que o usuário possa ter a melhorar experiência possível ao realizar um pedido na Cantina da Lu. O resultado final também foi de grande importância para as funcionárias da cantina, que terão muito mais facilidade em lidar com os pedidos, com um sistema administrativo extremamente eficiente e de fácil usabilidade.

Após a realização dos testes de usabilidade, o sistema apresentou sucesso em sua performance geral, com as funcionalidades desempenhando suas respectivas funções conforme planejado. As funcionárias da cantina avaliaram

positivamente a solução proposta, destacando a praticidade da interface e o impacto na organização dos pedidos. Algumas sugestões foram recebidas, como a inclusão da funcionalidade de exibição de relatório com estatísticas de vendas da cantina, e melhorias nas funções que envolviam a venda de marmitas, algo de muita importância no desenvolvimento do projeto.

O desenvolvimento do Cantina Virtual representou um período de aprendizado significativo para o grupo, tanto em melhoria nas áreas de programação e documentação, quanto nas habilidades de capacidade socioemocionais do integrantes. Além disso, o trabalho em equipe foi essencial para superar desafios, e apesar de algumas discussões, tudo foi resolvido de maneira saudável para que o projeto continuasse adiante. Alguns membros do grupo também aprimoraram suas habilidades de comunicação, por conta das apresentações, que nos ajudaram a desenvolver melhor nossa aptidão para falar em público.

Como proposta para trabalhos futuros, consideramos a possibilidade de incluir um sistema de pagamentos integrado, permitindo que os usuários finalizem suas transações diretamente pelo aplicativo. Além disso, a funcionalidade de relatórios poderia ser otimizada para oferecer mais detalhes sobre as vendas e o comportamento dos usuários. Essas melhorias visam elevar ainda mais a eficiência do sistema, garantindo que ele continue atendendo às necessidades dos usuários e evolua para a constante melhoria do ambiente de cantina escolar do SENAI.

REFERÊNCIAS

(2024), M. S. (29 de Novembro de 2024). *O que é Front-end Back-end e Full Stack - aprenda as diferenças entre essas áreas*. Fonte: Alura: https://www.alura.com.br/artigos/o-que-e-front-end-e-back-end?utm_term=&utm_campaign=%5BSearch%5D+%5BPerformance%5D+-+Dynamic+Search+Ads+-+Artigos+e+Conte%C3%BAdos&utm_source=adwords&utm_medium=ppc&hsa_acc=7964138385&hsa_cam=11384329873&hsa_grp=1640688476

Awari . (29 de Novembro de 2024). *Aprenda Html, Css, Javascript e Sql: os Fundamentos da Programação Web*. Fonte: Awari : https://awari.com.br/aprenda-html-css-javascript-e-sql-os-fundamentos-da-programacao-web-3/

Awari . (29 de Novembro de 2024). *Tutorial Completo do Mysql Workbench 8.0: Aprenda a Utilizar Essa Poderosa Ferramenta de Banco de Dados*. Fonte: Awari : https://awari.com.br/tutorial-completo-do-mysql-workbench-8-0-aprenda-a-utilizar-essa-poderosa-ferramenta-de-banco-de-dados/#:~:text=banco%20de%20dados-,O%20que%20%C3%A9%20o%20Mysql%20Workbench%208.0%20e%20como%20ele,desenvolver%20bancos%20de%20dados%20My

AWS. (29 de Novembro de 2024). *O que é Python?* Fonte: AWS: https://aws.amazon.com/pt/what-is/python/

Balbo, W. (2010). *SQL Server 2008 – Parte 9 – Conceitos de MER e DER, Regras e Tipos de Relacionamentos*. Acesso em 25 de outubro de 2024, disponível em Programando .NET: https://programandodotnet.wordpress.com/2010/11/06/sql-server-2008-%E2%80%93-parte-9-%E2%80%93-conceitos-de-mer-e-der-regras-e-tipos-de-relacionamentos/

Camila Fernanda, V. L. (29 de Novembro de 2024). *O que é Git e Github: os primeiros passos nessas ferramentas*. Fonte: Alura: https://www.alura.com.br/artigos/o-que-e-git-

github?utm_term=&utm_campaign=&utm_source=google&utm_medium=cpc&campaign_id=21946164160_174679622047_722718200760&utm_id=21946164160_174679622047_722718200760&hsa_acc=7964138385&hsa_cam=&hsa_grp=174679622047&hs

Developer Mozilla. (11 de outubro de 2024). *O que é JavaScript?* Fonte: Mozilla: https://developer.mozilla.org/pt-BR/docs/Learn/JavaScript/First_steps/What_is_JavaScript

Felipe Lo Sardo. (2014). *goomer*. Acesso em 25 de outubro de 2024, disponível em goomer: https://goomer.com.br/

Ferraz, U. (s.d.). *Success Check Animation Pure CSS*. Acesso em 25 de outubro de 2024, disponível em codepen.io: https://codepen.io/urak/pen/VOENMx

Goodwin, M. (2024). *O que é uma API?* Acesso em 08 de Novembro de 2024, disponível em ibm: https://www.ibm.com/br-pt/topics/api

Lima, G. (29 de Novembro de 2024). *Bootstrap: O que é, Documentação, como e quando usar*. Fonte: Alura: https://www.alura.com.br/artigos/bootstrap?utm_term=&utm_campaign=%5BSearch%5D+%5BPerformance%5D+-+Dynamic+Search+Ads+-+Artigos+e+Conte%C3%BAdos&utm_source=adwords&utm_medium=ppc&hsa_acc=7964138385&hsa_cam=11384329873&hsa_grp=164068847699&hsa_ad=722752682

Luciana. (2015). *Psicologia das cores: influência no cérebro*. Acesso em 25 de outubro de 2024, disponível em par de ideias: https://www.pardeideias.com/psicologia-das-cores

Mateus Villain, M. I. (29 de Novembro de 2024). *Figma: o que é a ferramenta, design e como usar*. Fonte: Alura: https://www.alura.com.br/artigos/figma?utm_term=&utm_campaign=&utm_source=google&utm_medium=cpc&campaign_id=21946164160_174679622047_72271820076

0&utm_id=21946164160_174679622047_722718200760&hsa_acc=7964138385&hsa_cam=&hsa_grp=174679622047&hsa_ad=72271820

Niehues, F. (2024). *Guia passo a passo para elaborar a documentação de processos*. Acesso em 08 de Novembro de 2024, disponível em Neomind: https://www.neomind.com.br/blog/guia-passo-a-passo-para-elaborar-a-documentacao-de-processos/

Oliveira, D. (29 de Novembro de 2023). *MER e DER: Definições, Banco de Dados e Exemplos*. Fonte: Alura: https://www.alura.com.br/artigos/mer-e-der-funcoes?srsltid=AfmBOoq-s4pnbMKjFMmM5SN9Qjt9K_z1qJhaWmAP-I7je3o_31ReBceY

Paiva, M. (29 de Novembro de 2024). *O que é design responsivo e qual a sua importância?* Fonte: Nuvem Shop: https://www.nuvemshop.com.br/blog/design-responsivo/

Remessa Online. (29 de Novembro de 2024). *Visual Studio Code: confira as principais funções da ferramenta*. Fonte: Remessa Online: https://www.remessaonline.com.br/blog/visual-studio-code-confira-as-principais-funcoes-da-ferramenta/

Serra, G. (29 de Novembro de 2024). *Mobile First: desenvolvimento web em ordem crescente*. Fonte: Tecnospeed: https://blog.tecnospeed.com.br/mobile-first/

Yohansson, A. (2023). *Ferramentas Que todo Front-End Precisa Conhecer*. Acesso em 25 de outubro de 2024, disponível em dio.me: https://www.dio.me/articles/ferramentas-que-todo-front-end-precisa-conhecer

GLOSSÁRIO

Banco de Dados - Um banco de dados armazena informações digitalmente, que permite o acesso e gerenciamento rápido dos dados.

Escopo - Estrutura que ajuda o sistema a atingir seus principais objetivos e funcionalidades.

Framework - Framework é um conjunto de bibliotecas, que permite reaproveitar e adaptar partes de códigos para criar novas funcionalidades de uma maneira mais simples.

Gestão de Estoque - Controle de produtos disponíveis para venda, monitorando a quantidade e o armazenamento adequado.

Horários de Pico - Períodos de maior movimento, com maior volume de clientes simultâneos.

Interface - A interface inclui todos os elementos visuais, como botões, menus e campos de texto, assim facilitando a experiencia do usuário.

Metodologias - Conjuntos de práticas aplicadas para guiar o desenvolvimento do sistema, assegurando qualidade e organização.

Micro-Frameworks - Um microframework é uma versão mais leve de um framework, com foco em fornecer apenas as funcionalidades essenciais para o desenvolvimento de um sistema.

Usabilidade - Qualidade que mede o quão fácil e intuitivo é para um usuário interagir com um sistema, garantindo que ele consiga realizar as ações desejadas sem dificuldades.

Wireframe - É um esboço visual de um sistema, que mostra a estrutura básica das telas, ajudando no planejamento da interface antes da programação.

APÊNDICE A – Código do Cadastro

A estrutura do código de cadastro do nosso sistema, onde existem espaços requisitados a serem preenchidos com informações importantes do usuário, elas sendo, nome completo, número de telefone, e-mail, criar, ou selecionar uma senha de sua preferência, e a seleção do seu curso, e caso já tenha uma conta, você pode clicar em um link que te redireciona para a página de login.

```
<div class="titulo">
            <h2>Olá! Cadastre-Se agora para Começar</h2>
        </div>

        <section class="container-principal">
            <div class="formulario">
                <form id="form-cadastro" action="" method="post">
                    <input id="nome" name="nome" placeholder="Nome Completo"
type="text" required>

                    <div class="form-group">
                        <div class="input-group">
                            <span class="prefix">+55</span>
                            <input id="tel" name="tel" placeholder="Número de
Telefone" type="text"
                                oninput="formatPhone()" required>
                        </div>
                    </div>

                    <input id="email" name="email" placeholder="E-mail"
type="email" required>
                    <input id="senha" name="senha" placeholder="Senha"
type="password" required>

                    <div class="select">
                        <select id="curso" name="curso" required>
```

```
                            <option value="">Selecione o seu curso</option>
                            {% for registro in cursos %}
                            <option value="{{ registro.id_curso }}">{{
registro.curso }}</option>
                            {% endfor %}
                        </select>

                        <button type="submit">Registrar</button>

                        <p>Já está registrado? <a href="/logar" class="login-
btn">Faça o login</a></p>
                    </div>
                </form>
            </div>
        </section>
    </main>
```

APÊNDICE B – Login de usuário

O código para que você entre na sua conta, com um formulário com informações a serem preenchidas, sendo elas, e-mail, senha, um link para recuperação de senha, e um link que redireciona o usuário para a criação de conta.

```
<div class="titulo">
            <h2>Olá! Logue agora para Começar</h2>
        </div>
        <section class="container-principal">

            <div class="formulario">

                <form action="" method="post">
                    <input name="email" id="email" placeholder="*E-mail"
type="text">
```

```
                    <input name="senha" id="senha" placeholder="*Senha"
type="password">
                    <a href="/trocar_senha"><span>Esqueceu sua
senha</span></a>
                    <button type="submit">Entrar</button>
                </form>


            </div>


            <span class="semConta">Ainda não tem conta?<a href="/cadastro">
Crie Aqui!</a></span>


        </section>
```

APÊNDICE C – Visualização do menu

Este código exibe a página principal do site para quem for usuário, tendo nela, um carrossel de imagens onde uma propaganda da cantina é feita, destacando e ressaltando as qualidades da cantina. Os produtos e marmitas aparecem logo abaixo, em formato de cards, com informações relevantes sobre o produto, como a imagem do produto, preço, descrição e uma imagem ilustrativa, caso o usuário clique no card, ele é direcionado para a visualização do produto em específico, com as mesmas informações, e um botão para acrescentar ao carrinho é adicionada.

```
<body>
  <main>
    <section class="carrossel">
      <div id="carouselExampleControls" class="carousel slide" data-
ride="carousel">
        <div class="carousel-inner">
          <div class="carousel-item active">
            <img class="d-block w-100" src="/static/img/test.png"
alt="Primeiro Slide">
```

```
          </div>
          <div class="carousel-item">
            <img class="d-block w-100" src="/static/img/test2.png"
alt="Segundo Slide">
          </div>
          <div class="carousel-item">
            <img class="d-block w-100" src="/static/img/test3.png"
alt="Terceiro Slide">
          </div>
        </div>
        <a class="carousel-control-prev" href="#carouselExampleControls"
role="button" data-slide="prev">
          <span class="carousel-control-prev-icon" aria-hidden="true"></span>
          <span class="sr-only">Anterior</span>
        </a>
        <a class="carousel-control-next" href="#carouselExampleControls"
role="button" data-slide="next">
          <span class="carousel-control-next-icon" aria-hidden="true"></span>
          <span class="sr-only">Próximo</span>
        </a>
      </div>
    </section>

    <!-- Seção de Produtos -->
    <section class="container-produtos">
      <h2>Nossos Produtos</h2>
      <div class="grid-produto">
        <div class="produtos" id="produtos-list">
          {% for produto in lista_produtos %}
          <form action="/produto_unico" method="post">
            <button value="{{ produto.id_produto }}" name="btn-produto"
type="submit">
              <div class="card-produtos">
                <div class="left-lado">
                  <span class="produto-nome">{{ produto.nome_produto }}</span>
                  <span class="produto-desc">{{ produto.descricao }}</span>
                  <span class="produto-preco">R$ {{ produto.preco }}</span>
                </div>
```

```
                <div class="right-lado">
                  <img src="{{ produto.imagem_produto }}" alt="{{
produto.nome_produto }}">
                </div>
              </div>
            </button>
          </form>
          {% endfor %}
        </div>
      </div>
    </section>

    <!-- Seção de Marmitas -->
    <section class="container-produtos">
      <h2>Marmitex</h2>
      <div class="grid-produto">
        <div class="produtos" id="marmitas-list">
          {% for marmita in lista_marmitas %}
          <form action="/marmita_unica" method="post">
            <button value="{{ marmita.id_marmita }}" name="btn-produto"
type="submit">
              <div class="card-produtos">
                <div class="left-lado">
                  <span class="produto-nome">{{ marmita.nome_marmita }}</span>
                  <span class="produto-desc">{{ marmita.descricao }}</span>
                  <span class="produto-preco">R$ {{ marmita.preco }}</span>
                </div>
                <div class="right-lado">
                  <img src="{{ marmita.imagem_marmita }}" alt="{{
marmita.nome_marmita }}">
                </div>
              </div>
            </button>
          </form>
          {% endfor %}
        </div>
      </div>
    </section>
```

```
  </main>
```

APÊNDICE D – Visualização e personalização das marmitas

O código para a exibição particular da marmitex, exibindo as suas informações, sendo elas, imagem, nome, preço e descrição, também tem uma área de selecionar alimentos que estarão presentes na sua marmitex, oferecendo os mais diversos tipos, que se adequem aos seus padrões alimentares, entretanto, existe um limite de guarnições e acompanhamentos que você pode selecionar, isso depende do tamanho da marmitex que você pediu, as menores são limitadas á uma guarnição, a média e grande podem ter até duas guarnições.

Durante a personalização os itens selecionáveis poderão ser vistos pelo usuário, em grande variedade, podendo montar a sua marmitex de acordo com seus gostos pessoais, e podendo concluir o carrinho de compras mais ao final da página.

```
<body>
    <main>
        <img class="ft-item" src="{{ marmita.imagem_marmita }}" alt="Imagem da
Marmita">

        <section class="descricao">
            <span class="nome-produto">{{ marmita.nome_marmita }}</span>
            <span class="descricao-produto">{{ marmita.descricao }}</span>
            <span class="preco-produto">R$ {{ marmita.preco }}</span>

            <div class="opc">

                <form id="form-marmita" class="formulario-marmita">
                    <input type="hidden" name="id_marmita" value="{{
marmita.id_marmita }}"/> <!-- ID da marmita -->
```

```
                    <!-- Campos para Acompanhamentos -->
                    {% if marmita.acompanhamentos and
marmita.acompanhamentos|length > 0 %}
                    <div class="acompanhamentos">
                        <div class="header">
                            <div class="titulosubtitulo">
                                <h3>Acompanhamentos</h3>
                                <p id="acompanhamento-info">Escolha até <span
id="max-acompanhamentos"></span>
                                        acompanhamentos</p>
                            </div>
                            <span class="seta"
onclick="toggleAcompanhamentos(this)">&#9660;</span>
                        </div>
                        <div class="opcoes-acompanhamentos hidden">
                            {% for acompanhamento in marmita.acompanhamentos
%}
                            <div class="acompanhamento">
                                <img
src="https://pedido.anota.ai/assets/item_no_image-a8c57261.png"
                                    alt="{{ acompanhamento.nome }}" />
                                <span>{{ acompanhamento.nome }}</span>
                                <label>
                                    <input type="checkbox"
name="acompanhamento" value="{{ acompanhamento.id }}" />
                                    <span></span>
                                </label>
                            </div>
                            {% endfor %}
                        </div>
                    </div>
                    {% endif %}

                    <!-- Campos para Guarnições -->
                    {% if marmita.guarnicoes and marmita.guarnicoes|length > 0
%}
                    <div class="acompanhamentos">
```

```
                        <div class="header">
                            <div class="titulosubtitulo">
                                <h3>Guarnições</h3>
                                <p id="guarnicao-info">Escolha até <span
id="max-guarnicoes"></span> guarnições</p>
                            </div>
                            <span class="seta"
onclick="toggleGuarnicoes(this)">&#9660;</span>
                        </div>
                        <div class="opcoes-guarnicoes hidden">
                            {% for guarnicao in marmita.guarnicoes %}
                            <div class="acompanhamento">
                                <img
src="https://pedido.anota.ai/assets/item_no_image-a8c57261.png"
                                    alt="{{ guarnicao.nome }}" />
                                <span>{{ guarnicao.nome }}</span>
                                <label>
                                    <input type="checkbox" name="guarnicao"
value="{{ guarnicao.id }}" />
                                    <span></span>
                                </label>
                            </div>
                            {% endfor %}
                        </div>
                    </div>
                    {% endif %}

                    <!-- Botões no final -->
                    <div class="sair">
                        <a href="/">Voltar</a>
                        <button type="button"
id="adicionarCarrinhoBtn">Adicionar ao Carrinho</button>
                    </div>
                </form>
            </div>
        </section>
    </main>
</body>
```

```
<div class="espacamento1"></div>
```

APÊNDICE E – Adição de produtos ao carrinho

Este código é uma página que exibe os itens do carrinho. Cada produto ou marmitex no carrinho tem a sua quantidade, uma foto, nome, preço e pode ser excluído. No final o valor de todos os pedidos do usuário é somados e é mostrado o valor total da compra, com opções para finalizar a compra, e continuar comprando.

```
    <main>

        {% for registro in lista_carrinho.produtos %}
        <div class="form-exclui-carrinho">
            <section class="container-principal">
                <div class="produto-card">
                    <div class="imgdiv">
                        <img class="img-produto" src="{{
registro.imagem_produto }}" alt="Imagem Produto">
                        <button class="button btn-excluir" data-id="{{
registro.id_carrinho }}">
                            <img src="../static/img/lixo.png" alt="Excluir
Produto">
                        </button>
                        <div class="direita-card">
                            <div class="cima">
                                <span>{{ registro.nome_produto }}</span>
                                <div class="qnt-btn">
                                    <button class="menos-btn" type="button"
                                        data-id="{{ registro.id_carrinho }}">-
</button>
                                    <input type="number" id="quantidade-{{
registro.id_carrinho }}"
```

```
                                        value="{{ registro.quantidade }}"
data-id="{{ registro.id_carrinho }}" min="1">
                                    <button class="mais-btn" type="button"
                                        data-id="{{ registro.id_carrinho
}}">+</button>
                                </div>
                            </div>
                            <div class="baixo">
                                <span class="desc-baixo">{{ 'Descrição do
produto' }}</span>
                                <span class="preco-item">R$ {{ registro.preco
}}</span>
                                <span class="obs-item">OBS: {{ 'Observação do
produto' }}</span>
                            </div>
                        </div>
                    </div>
                </section>
            </div>
            {% endfor %}

            {% for registro in lista_carrinho.marmitas %}
            <div class="form-exclui-carrinho">
                <section class="container-principal">
                    <div class="produto-card">
                        <div class="imgdiv">
                            <img class="img-produto" src="{{
registro.imagem_produto }}" alt="Imagem Marmita">
                            <button class="button btn-excluir" data-id="{{
registro.id_carrinho }}">
                                <img src="../static/img/lixo.png" alt="Excluir
Marmita">
                            </button>
                            <div class="direita-card">
                                <div class="cima">
                                    <span>{{ registro.nome_marmita }}</span>
                                    <div class="qnt-btn">
```

```
                                    <button class="menos-btn" type="button"
                                        data-id="{{ registro.id_carrinho }}">-
</button>
                                    <input type="number" id="quantidade-{{
registro.id_carrinho }}"
                                        value="{{ registro.quantidade }}"
data-id="{{ registro.id_carrinho }}" min="1">
                                    <button class="mais-btn" type="button"
                                        data-id="{{ registro.id_carrinho
}}">+</button>
                                </div>
                            </div>
                            <div class="baixo">
                                <span class="desc-baixo">{{ registro.descricao
}}</span>
                                <span class="preco-item">R$ {{ registro.preco
}}</span>

                                <div class="guarnicoes">
                                    <strong>Guarnições:</strong>
                                    <ul>
                                        {% for guarnicao in
registro.guarnicoes %}
                                        <li>{{ guarnicao }}</li>
                                        {% endfor %}
                                    </ul>
                                </div>

                                <div class="acompanhamentos">
                                    <strong>Acompanhamentos:</strong>
                                    <ul>
                                        {% for acompanhamento in
registro.acompanhamentos %}
                                        <li>{{ acompanhamento }}</li>
                                        {% endfor %}
                                    </ul>
                                </div>
```

```
                                <span class="obs-item">OBS: {{ 'Observação da
marmita' }}</span>
                            </div>
                        </div>
                    </div>
                </div>
            </section>
        </div>
        {% endfor %}

        <div class="finalizacao">
            <div class="preco">
                <span class="span1">Total:</span>
                <span id="total-preco" class="span2">R$ {{
lista_carrinho.total_preco }}</span>
            </div>

            <button type="button" onclick="enviarCarrinho()"
class="button1">Enviar</button>
            <a href="/">Continuar Comprando</a>
        </div>

        <span class="span3">.</span>
    </main>
```

APÊNDICE F – Envio de pedidos

O sistema exibe pedidos feitos por clientes de forma dinâmica, mostrando as informações gerais do cliente que fez o pedido, o número e data do pedido.

Estes dados são organizados e tem as suas informações atualizadas via AJAX.

```
    <body>
    <main>
        <section class="container-cards" id="order-container">
            <!-- Os pedidos serão carregados dinamicamente aqui -->
        </section>
    </main>

    <script>
        // Função para exibir os pedidos via AJAX
        function carregarPedidos() {
            fetch('/obter_pedidos')
                .then(response => response.json())
                .then(data => {
                    const orderContainer = document.getElementById('order-
container');
                    orderContainer.innerHTML = ''; // Limpa o conteúdo antes
de carregar os pedidos

                    for (const [id_cliente, cliente] of Object.entries(data))
{
                        let cardPedidos = `<div class="card-pedidos">`;

                        for (const [id_pedido, pedido] of
Object.entries(cliente.pedidos)) {
                            cardPedidos += `
                                <div class="order-card">
                                    <h2>Cliente: ${cliente.nome_cliente}</h2>
                                    <div class="order-header">
                                        <span class="order-
number">#${id_pedido}</span>
                                        <span class="order-
time">${pedido.hora}</span>
                                        <span class="order-
date">${pedido.data_pedido}</span>
                                    </div>
                                    <div class="order-details">
```

```
                                    <h3>Pedidos:</h3>
                                    <ul>`;

                            // Exibe os produtos com quantidade
                            if (pedido.produtos && pedido.produtos.length > 0)
{
                                pedido.produtos.forEach(produto => {
                                    if (produto.nome_produto) {
                                        cardPedidos += `<li>Produto:
${produto.nome_produto} - Quantidade: ${produto.quantidade} - Preço: R$
${produto.preco.toFixed(2)}</li>`;
                                    }
                                });
                            }

                            // Exibe as marmitas com quantidade e associa
guarnições e acompanhamentos
                            if (pedido.marmitas && pedido.marmitas.length > 0)
{
                                pedido.marmitas.forEach(marmita => {
                                    if (marmita.nome_marmita) {
                                        cardPedidos += `<li>Marmita:
${marmita.nome_marmita} - Tamanho: ${marmita.tamanho} - Quantidade:
${marmita.quantidade} - Preço: R$ ${marmita.preco.toFixed(2)}</li>`;

                                        // Guarnições associadas à marmita
                                        if (pedido.guarnicoes &&
pedido.guarnicoes.length > 0) {
                                            cardPedidos +=
`<ul><li><strong>Guarnições:</strong></li>`;

pedido.guarnicoes.forEach(guarnicao => {
                                                cardPedidos +=
`<li>${guarnicao}</li>`;
                                            });
                                            cardPedidos += `</ul>`;
                                        }
```

```
                                    // Acompanhamentos associados à
marmita
                                    if (pedido.acompanhamentos &&
pedido.acompanhamentos.length > 0) {
                                        cardPedidos +=
`<ul><li><strong>Acompanhamentos:</strong></li>`;

pedido.acompanhamentos.forEach(acompanhamento => {
                                            cardPedidos +=
`<li>${acompanhamento}</li>`;
                                        });
                                        cardPedidos += `</ul>`;
                                    }
                                }
                            });
                        }

                        // Mensagem caso não haja produtos, marmitas,
guarnições ou acompanhamentos
                        if (pedido.produtos.length === 0 &&
pedido.marmitas.length === 0 && pedido.guarnicoes.length === 0 &&
pedido.acompanhamentos.length === 0) {
                            cardPedidos += `<li>Nenhum produto, marmita,
guarnição ou acompanhamento</li>`;
                        }
```

APÊNDICE G – Acompanhamento de status do pedido

Este código exibe uma lista de pedidos e permite que o usuário atualize o status de cada pedido para "Preparando", "Pronto", "Entregue" ou "Cancelar". A lista de pedidos é atualizada automaticamente a cada 2 segundos e quando a página é carregada. Cada botão de status envia uma requisição para atualizar o status do pedido no servidor, e o botão de "Cancelar" remove o pedido.

```
cardPedidos += `</ul>
                                    </div>
                                    <div class="order-status">
                                        <span class="order-price">Total: R$
${pedido.total_preco.toFixed(2)}</span>
                                    </div>
                                    <div class="btn-cancelar">
                                        <button class="status-btn" data-
pedido="${id_pedido}" value="preparando">Preparando</button>
                                        <button class="status-btn" data-
pedido="${id_pedido}" value="feito">Pronto</button>
                                        <button class="status-btn" data-
pedido="${id_pedido}" value="entregue">Entregue</button>
                                        <button class="cancel-btn" data-
pedido="${id_pedido}">Cancelar</button>
                                    </div>
                                </div>`;
                        }
                        cardPedidos += `</div>`;
                        orderContainer.innerHTML += cardPedidos; // Adiciona
os pedidos na página
                    }
                })
                .catch(error => console.error('Erro ao carregar os pedidos:',
error));
        }

        // Atualiza a lista de pedidos a cada 2 segundos (tempo ajustável)
        setInterval(carregarPedidos, 2000);

        // Carrega os pedidos quando a página é carregada
        window.onload = carregarPedidos;

        // Função para lidar com cliques nos botões de status e cancelar
        document.addEventListener('click', function (e) {
            // Para os botões de status
```

```
            if (e.target.classList.contains('status-btn')) {
                const idPedido = e.target.getAttribute('data-pedido');
                const novoStatus = e.target.value;

                // Se o status for "entregue", chama a rota específica para
marcar como entregue
                if (novoStatus === 'entregue') {
                    fetch(`/marcar_entregue/${idPedido}`, {
                        method: 'POST',
                        headers: {
                            'Content-Type': 'application/x-www-form-
urlencoded'
                        }
                    })
                    .then(response => response.json())
                    .then(data => {
                        if (data.status === 'sucesso') {
                            alert('Pedido marcado como entregue!');
                            carregarPedidos();  // Atualiza a lista de pedidos
na página
                        } else {
                            alert('Erro ao marcar como entregue: ' +
data.mensagem);
                        }
                    })
                    .catch(error => console.error('Erro ao enviar a
solicitação:', error));
                } else {
                    // Envia uma solicitação POST via AJAX para atualizar o
status do pedido
                    fetch('/atualizar_status_pedido', {
                        method: 'POST',
                        headers: {
                            'Content-Type': 'application/x-www-form-
urlencoded'
                        },
                        body: `id_pedido=${idPedido}&status=${novoStatus}`
                    })
```

```
                    .then(response => response.json())
                    .then(data => {
                        if (data.status === 'sucesso') {
                            alert('Status atualizado com sucesso!');
                            carregarPedidos();  // Atualiza a lista de pedidos
na página
                        } else {
                            alert('Erro ao atualizar o status: ' +
data.mensagem);
                        }
                    })
                    .catch(error => console.error('Erro ao enviar a
solicitação:', error));
                }
            }

            // Para o botão de cancelar pedido
            if (e.target.classList.contains('cancel-btn')) {
                const idPedido = e.target.getAttribute('data-pedido');

                // Envia uma solicitação POST via AJAX para cancelar o pedido
                fetch('/cancelar_pedido', {
                    method: 'POST',
                    headers: {
                        'Content-Type': 'application/x-www-form-urlencoded'
                    },
                    body: `id_pedido=${idPedido}`
                })
                .then(response => response.json())
                .then(data => {
                    if (data.status === 'sucesso') {
                        alert('Pedido cancelado com sucesso!');
                        carregarPedidos();  // Atualiza a lista de pedidos na
página
                    } else {
                        alert('Erro ao cancelar o pedido: ' + data.mensagem);
                    }
                })
```

```
                .catch(error => console.error('Erro ao enviar a solicitação:',
error));
            }
        });
    </script>
```

APÊNDICE H – Personalização do perfil

O código permite o usuário navegar pela página do usuário, one ele pode visualizar e atualizar os dados do seu perfil, sendo eles, nome e imagem.

```
<body>
    <main>
        <section>
            <form action="/atualizar_perfil" method="post"
enctype="multipart/form-data" class="container-principal">
                <h2>Perfil</h2>

                <div class="profile-container" id="profile-container">
                    <div class="profile-image">
                        <img id="profile-img"
                            src="{{ url_for('imagem_perfil',
id_cliente=session['usuario_logado']['id_cliente']) }}"
                            alt="Imagem do Perfil" width="150" height="150"
                            onerror="this.src='../static/img/default-
avatar.png';">
                    </div>

                    <div class="overlay">
                        <input type="file" id="file-upload"
name="imagem_perfil" accept="image/png, image/jpeg"
                               onchange="previewImage(event)">
                        <img src="../static/img/btn-editar.png" alt="Imagem
Sobreposta" class="overlay-image">
                    </div>
                </div>
```

```
                <div class="formulario">
                    <div class="form">
                        <input placeholder="{{ perfil_usuario.nome }}"
type="text" name="nome" value="{{ session['usuario_logado']['nome_comp'] }}">

                        <input type="password" id="password" name="senha"
placeholder="*Senha" required>
                        <button type="button" class="toggle-button"
                                onclick="togglePassword('password', 'toggle-
icon')">
                            <img id="toggle-icon" src="../static/img/olho-
fechado.png" alt="Mostrar Senha"
                                 width="25" height="25">
                        </button>

                        <input placeholder="*Confirmar Senha"
id="passwordConfirmar" name="confirmar_senha"
                               type="password" required>
                        <button type="button" class="toggle-button2"
                                onclick="togglePassword('passwordConfirmar',
'toggle-icon2')">
                            <img id="toggle-icon2" src="../static/img/olho-
fechado.png" alt="Mostrar Senha"
                                 width="25" height="25">
                        </button>

                        <button class="btn-confirmar" type="submit">Confirmar
Mudanças</button>
                    </div>
                </div>
            </form>
        </section>
    </main>

    <script>
        // Função para pré-visualizar a imagem selecionada
```

```
        function previewImage(event) {
            const input = event.target;
            const reader = new FileReader();

            reader.onload = function () {
                const imgElement = document.getElementById('profile-img');
                imgElement.src = reader.result;
            };

            if (input.files && input.files[0]) {
                reader.readAsDataURL(input.files[0]);
            }
        }

        // Funções para exibir e ocultar senha
        function togglePassword(inputId, iconId) {
            const inputField = document.getElementById(inputId);
            const icon = document.getElementById(iconId);

            inputField.type = inputField.type === 'password' ? 'text' :
'password';
            icon.src = inputField.type === 'text' ? '../static/img/olho-
aberto.png' : '../static/img/olho-fechado.png';
        }
    </script>
</body>
```

APÊNDICE I – Desativação de produtos

Este código define funções para habilitar e desabilitar produtos e marmitas em um sistema administrativo. Cada função conecta-se ao banco de dados, atualiza o status do item para habilitado (1) ou desabilitado (0) com base no ID fornecido, confirma a mudança e encerra a conexão. Em seguida, retorna uma mensagem indicando o sucesso da operação.

O código define a função de habilitar e desabilitar produtos e marmitex para o administrador

```
def desabilitar_produto_adm(self, produto_id):
        mydb = Conexao.conectar()
        mycursor = mydb.cursor()

        # Atualizar o status do produto para desabilitado (0)
        sql = "UPDATE tb_produto SET habilitado = 0 WHERE cod_produto = %s"
        mycursor.execute(sql, (produto_id,))

        mydb.commit()
        mydb.close()

        return {"message": "Produto desabilitado com sucesso!"}  # Retorna um
dicionário

    # Modifique a função para habilitar produto
    def habilitar_produto_adm(self, produto_id):
        mydb = Conexao.conectar()
        mycursor = mydb.cursor()

        # Atualizar o status do produto para habilitado (1)
        sql = "UPDATE tb_produto SET habilitado = 1 WHERE cod_produto = %s"
        mycursor.execute(sql, (produto_id,))

        mydb.commit()
        mydb.close()

        return {"message": "Produto habilitado com sucesso!"}  # Retorna um
dicionário




    # Modifique a função para desabilitar marmita
```

```
    def desabilitar_marmita_adm(self, marmita_id):
        mydb = Conexao.conectar()
        mycursor = mydb.cursor()

        # Atualizar o status da marmita para desabilitado (0)
        sql = "UPDATE tb_marmita SET habilitado = 0 WHERE id_marmita = %s"
        mycursor.execute(sql, (marmita_id,))

        mydb.commit()
        mydb.close()

        return {"message": "Marmita desabilitada com sucesso!"}  # Retorna um
dicionário

    # Modifique a função para habilitar marmita
    def habilitar_marmita_adm(self, marmita_id):
        mydb = Conexao.conectar()
        mycursor = mydb.cursor()

        # Atualizar o status da marmita para habilitado (1)
        sql = "UPDATE tb_marmita SET habilitado = 1 WHERE id_marmita = %s"
        mycursor.execute(sql, (marmita_id,))

        mydb.commit()
        mydb.close()

        return {"message": "Marmita habilitada com sucesso!"}  # Retorna um
dicionário
```

APÊNDICE J – Back-End

```
      from flask import Flask, render_template, request, redirect, session,
jsonify, flash, Response, url_for
      from usuario import Usuario
      from sistema import Sistema
```

```
      from carrinho import Carrinho
      from perfil import Perfil
      from adm import Adm
      import random
      from twilio.rest import Client
      import os
      from datetime import datetime
      import pytz
      from relatorio import Relatorio
      from dotenv import load_dotenv




      app = Flask(__name__)
      app.secret_key = 'sua_chave_secreta'  # Chave secreta para gerenciamento
de sessões


      # Configuração do diretório de upload
      app.config['UPLOAD_FOLDER']   =   os.path.join(app.root_path,   'static',
'uploads')

      # Crie o diretório se ele não existir
      os.makedirs(app.config['UPLOAD_FOLDER'], exist_ok=True)




      # Carregar variáveis do arquivo .env
      load_dotenv("sms.env")

      account_sid = os.getenv("TWILIO_ACCOUNT_SID")
      auth_token = os.getenv("TWILIO_AUTH_TOKEN")

      client = Client(account_sid, auth_token)



      def verificar_sessao():
          """
          Função genérica para verificar o estado da sessão do usuário.
          Se o usuário estiver no meio do processo de verificação incompleta,
          a sessão será limpa.
          """
                    if      'usuario_logado'      in      session      and
session.get('verificacao_incompleta'):
              # Limpa a sessão de usuário e a flag de verificação incompleta
              session.pop('usuario_logado', None)
```

```
            session.pop('verificacao_incompleta', None)



    @app.route("/")
    def principal():
        try:
            verificar_sessao()

            # Redirecionar para a página inicial do administrador se o tipo
for 'adm'
                            if     'usuario_logado'     in     session     and
session['usuario_logado'].get('tipo') == 'adm':
                return redirect("/inicialadm")

            # Verifica se o site está aberto, sem a necessidade de login
            timezone_sp = pytz.timezone('America/Sao_Paulo')
            now = datetime.now(timezone_sp)
            hora_atual = now.hour
            minutos_atuais = now.minute
                site_aberto  =  hora_atual  >=  7  and  (hora_atual  <  21  or
(hora_atual == 21 and minutos_atuais < 45))

            # Obter o status de login bem-sucedido para exibir o alerta
            success = session.pop('login_sucesso', False)

            if site_aberto:
                sistema = Sistema()
                produtos_por_categoria = sistema.exibir_produtos()
                lista_marmitas = sistema.exibir_marmitas()
                return render_template(
                    "index.html",
                    produtos_por_categoria=produtos_por_categoria,
                    lista_marmitas=lista_marmitas,
                    site_aberto=site_aberto,
                    success=success,
                )

              return  render_template("index.html",  site_aberto=site_aberto,
success=success)

        except Exception as e:
            print(f"Erro ao verificar o horário de funcionamento: {e}")
            return "Erro ao verificar o horário de funcionamento."
```

```
@app.route("/produtos_json", methods=['GET'])
def produtos():
    sistema = Sistema()  # Cria uma instância da classe Sistema
     produtos_por_categoria = sistema.exibir_produtos()  # Obtém a lista de produtos agrupados

     # Cria uma lista para armazenar todos os produtos com a categoria associada
    lista_produtos = []

    # Itera sobre as categorias e adiciona os produtos à lista
    for categoria in produtos_por_categoria.values():
        for produto in categoria['produtos']:
            produto_com_categoria = {
                'id_produto': produto['id_produto'],
                'nome_produto': produto['nome_produto'],
                'preco': produto['preco'],
                'imagem_produto': produto['imagem_produto'],
                'descricao': produto['descricao'],
                 'nome_categoria': categoria['nome_categoria']  # Inclui o nome da categoria
                }
                lista_produtos.append(produto_com_categoria)

     return jsonify(lista_produtos)  # Retorna a lista de produtos em formato JSON



@app.route("/marmitas_json", methods=['GET'])
def marmitas():
    sistema = Sistema()  # Cria uma instância da classe Sistema
     lista_marmitas = sistema.exibir_marmitas()  # Obtém a lista de produtos
     return jsonify(lista_marmitas)  # Retorna os produtos em formato JSON



@app.route("/inicialadm")
def inicialadm():
    # Verifica e limpa a sessão se necessário
    verificar_sessao()
    # Verifica se o usuário está logado e possui um ID de cliente válido
```

```
          if 'usuario_logado' not in session or session['usuario_logado'] is
None or session['usuario_logado'].get('id_cliente') is None:
              return redirect('/logar')  # Redireciona para a página de login

          # Verifica se o tipo do usuário é 'adm'
          if session['usuario_logado']['tipo'] == "cliente":
              return redirect("/")  # Redireciona para a rota "/"

          # Renderiza a página inicialAdm.html se o usuário não for 'adm'
          return render_template("inicialAdm.html")




      @app.route("/adm")
      def principal_adm():
          # Verifica e limpa a sessão se necessário
          verificar_sessao()
          # Verifica se o usuário está logado e possui um ID de cliente válido
          if 'usuario_logado' not in session or session['usuario_logado'] is
None or session['usuario_logado'].get('id_cliente') is None:
              return redirect('/logar')  # Redireciona para a página de login

          # Verifica se o tipo do usuário é 'adm'
          if session['usuario_logado']['tipo'] == "cliente":
              return redirect("/")  # Redireciona para a rota "/"

          sistema = Sistema()  # Cria uma instância da classe Sistema
          lista_produtos = sistema.exibir_produtos_adm()  # Obtém a lista de
produtos
          lista_marmitas = sistema.exibir_marmitas_adm()
                               return          render_template("index-adm.html",
lista_produtos=lista_produtos, lista_marmitas = lista_marmitas)  # Renderiza a
página inicial com a lista de produtos

      # Rota de cadastro
      @app.route("/cadastro", methods=["GET", "POST"])
      def cadastro():

          # Redireciona para "/" se o usuário já estiver logado
          if 'usuario_logado' in session:
              return redirect("/")

          if request.method == 'GET':
              verificar_sessao()
              usuario = Usuario()
              cursos = usuario.exibir_cursos()
              return render_template("cadastrar.html", cursos=cursos)
          else:
```

```
            # Recebe os dados do formulário
            nome = request.form["nome"]
            telefone = request.form["tel"]
            email = request.form["email"]
            senha = request.form["senha"]
            curso = request.form["curso"]
            tipo = "cliente"

            usuario = Usuario()

            # Verifica se o email ou telefone já estão cadastrados
            if usuario.verificar_duplicidade(email, telefone):
                flash("Email ou telefone já cadastrado.")
                return redirect("/cadastro")

            # Gera o código de verificação e o envia
            verification_code = str(random.randint(1000, 9999)).zfill(4)
            message = client.messages.create(
                to=telefone,
                from_="+13195190041",
                body=f'Seu código é: {verification_code}'
            )
            print(message.sid)

             # Armazena dados temporários na sessão para realizar o cadastro
após a verificação
            session['dados_cadastro'] = {
                "nome": nome,
                "telefone": telefone,
                "email": email,
                "senha": senha,
                "curso": curso,
                "tipo": tipo,
                "verification_code": verification_code
            }
              session['tipo_verificacao'] = "cadastro"  # Define o tipo de
verificação como cadastro

            # Redireciona para a página de verificação
            return redirect("/verificacao")


    @app.route("/verificacao", methods=["GET", "POST"])
    def verificacao():
         if 'dados_cadastro' not in session and 'email_pendente' not in
session:
            # Redireciona para login se faltar informações
            session.pop('usuario_logado', None)
            return redirect("/logar")
```

```
        if request.method == 'GET':
            return render_template("verificacao.html")

        # Código inserido pelo usuário
        codigo_inserido = "".join([request.form["codigo1"],
                                   request.form["codigo2"],
                                   request.form["codigo3"],
                                   request.form["codigo4"]])
        verification_code = session.get('verification_code')
        tipo_verificacao = session.get('tipo_verificacao')

        if codigo_inserido == verification_code:
            # Verificação bem-sucedida
            if tipo_verificacao == "cadastro":
                dados_cadastro = session.pop('dados_cadastro', None)
                if dados_cadastro:
                    usuario = Usuario()
                    usuario.cadastrar(
                        dados_cadastro["nome"],
                        dados_cadastro["telefone"],
                        dados_cadastro["email"],
                        dados_cadastro["senha"],
                        dados_cadastro["curso"],
                        dados_cadastro["tipo"]
                    )

                    # Login automático após o cadastro
                                        usuario.logar(dados_cadastro["email"],
dados_cadastro["senha"])
                    if usuario.logado:
                        session['usuario_logado'] = {
                            "nome": usuario.nome,
                            "email": usuario.email,
                            "tel": usuario.tel,
                            "id_cliente": usuario.id_cliente,
                            "tipo": usuario.tipo
                        }
                session.pop('verification_code', None)
                session.pop('tipo_verificacao', None)
                return redirect("/")

            elif tipo_verificacao == "atualizar_dados_iniciais":
                id_cliente = session['usuario_logado']['id_cliente']
                telefone = session['telefone']
                email = session.pop('email_pendente')
                senha = session.pop('senha_pendente')

                # Atualiza o telefone
```

```
                usuario = Usuario()
                usuario.atualizar_telefone(id_cliente, telefone)  # Atualiza
o telefone no banco

                # Agora atualiza os dados (email e senha)
                 usuario.atualizar_dados(id_cliente, telefone, email, senha)
 # Atualiza o email e senha, mas telefone já foi atualizado

                session.pop('verification_code', None)
                session.pop('tipo_verificacao', None)
                session.pop('verificacao_incompleta', None)

                return redirect("/inicialadm")


        else:
                  return   render_template("verificacao.html",   erro="Código
incorreto. Tente novamente.")









    @app.route("/logar", methods=['GET', 'POST'])
    def logar():
        if request.method == 'GET':
            # Verifica se o usuário já está logado
            if 'usuario_logado' in session:
                usuario = session['usuario_logado']
                # Redireciona conforme o tipo do usuário
                if usuario.get('tipo') == 'adm':
                    return redirect("/inicialadm")
                else:
                    return redirect("/")


            # Renderiza a página de login para usuários não logados
            erro = session.pop('login_erro', False)
            return render_template('login.html', success=False, erro=erro)

        # Processa o formulário de login
        senha = request.form['senha']
        email = request.form['email']
        usuario = Usuario()
        usuario.logar(email, senha)
```

```
        if usuario.logado:
            # Define os dados do usuário na sessão
            session['usuario_logado'] = {
                "nome": usuario.nome,
                "email": usuario.email,
                "tel": usuario.tel,
                "id_cliente": usuario.id_cliente,
                "tipo": usuario.tipo,
                "senha": usuario.senha,
                "primeiro_login": usuario.primeiro_login
            }

            print("Sessão após login:", session)  # Verificação de sessão

            # Redireciona com base no tipo e status do login
            if usuario.tipo != 'cliente' and not usuario.primeiro_login:
                return redirect("/inicialadm")

            if usuario.tipo != 'cliente' and usuario.primeiro_login:
                 session['verificacao_incompleta'] = True  # Marca a sessão
para verificação incompleta
                 return redirect("/atualizar_dados_iniciais")  # Redireciona
para a atualização de dados

            session['login_sucesso'] = True
            return redirect("/")

        # Define erro de login na sessão e redireciona para login
        session['login_erro'] = True
        return redirect("/logar")




    @app.route("/atualizar_dados_iniciais", methods=["GET", "POST"])
    def atualizar_dados_iniciais():
        # Verifica se o usuário está logado e se a verificação está pendente
        if 'usuario_logado' not in session:
            return redirect("/logar")

        if not session.get('verificacao_incompleta'):
            return redirect("/")

        if request.method == 'GET':
            return render_template("atualizar_dados_iniciais.html")

        # Processa a atualização dos dados
        telefone = request.form.get('telefone')
```

```
          email = request.form.get('email')
          senha = request.form.get('senha')

          if not telefone or not email or not senha:
                      return   render_template("atualizar_dados_iniciais.html",
erro="Preencha todos os campos.")

          id_cliente = session['usuario_logado']['id_cliente']
          usuario = Usuario()
          usuario.atualizar_telefone(id_cliente, telefone)

          # Salva os dados pendentes na sessão
          session['telefone'] = telefone
          session['email_pendente'] = email
          session['senha_pendente'] = senha
                session['verification_code']     =     str(random.randint(1000,
9999)).zfill(4)
          session['tipo_verificacao'] = "atualizar_dados_iniciais"

          # Envia o código de verificação
          message = client.messages.create(
              to=telefone,
              from_="+17753707822",
              body=f'Seu código é: {session["verification_code"]}'
          )
          session['verificacao_incompleta'] = True
          return redirect("/verificacao")









      # Rota para logout de usuários
      @app.route('/logout')
      def logout():
          if request.method == 'GET':
                  id_cliente  =  session.get('usuario_logado')['id_cliente']   #
Obtém o ID do cliente da sessão
```

```
            usuario = Usuario()  # Cria uma instância da classe Usuario
            usuario.logout(id_cliente)  # Realiza o logout do usuário
            session.clear()  # Limpa a sessão
            return redirect("/")  # Redireciona para a página inicial


    @app.route('/inserir_produtos', methods=['POST'])
    def inserir_produtos():
        # Verifica se o usuário está logado e possui um ID de cliente válido
        if 'usuario_logado' not in session or session['usuario_logado'] is
None or session['usuario_logado'].get('id_cliente') is None:
            return redirect('/logar')  # Redireciona para a página de login

        # Verifica se o tipo do usuário é 'adm'
        if session['usuario_logado']['tipo'] == "cliente":
            return redirect("/")  # Redireciona para a rota "/"

        nome = request.form['nome']
        preco = request.form['preco']
        descricao = request.form['descricao']
        categoria = request.form['categoria']
          tamanho  =  request.form.get('tamanho')  #  Tamanho  da  marmita
(opcional)

        # Captura guarnições e acompanhamentos
        guarnicoes_existentes = request.form.getlist('guarnicoes')
        acompanhamentos_existentes = request.form.getlist('acompanhamentos')
        novas_guarnicoes = request.form.getlist('nova_guarnicoes')

        # Upload da imagem
        imagem = request.files['img']
        imagem_binaria = imagem.read() if imagem else None

        # Cria uma instância do objeto que contém o método de inserção
        adm = Adm()  # Substitua pelo seu objeto real

        # Verifica se a categoria selecionada é "Marmita"
        id_categoria_marmita  =  7  # Substitua pelo ID real da categoria
"Marmita"

        if int(categoria) == id_categoria_marmita:
            sucesso  =  adm.inserir_marmita(nome,  preco,  imagem_binaria,
descricao,       tamanho,       guarnicoes_existentes,       novas_guarnicoes,
acompanhamentos_existentes)
        else:
            sucesso  =  adm.inserir_produto(nome,  preco,  imagem_binaria,
descricao, categoria, novas_guarnicoes)

        return redirect('/inicialadm')
```

```
@app.route("/exibir_guarnicao")
def exibir_guarnicao():
    # Verifica se o usuário está logado e possui um ID de cliente válido
     if 'usuario_logado' not in session or session['usuario_logado'] is
None or session['usuario_logado'].get('id_cliente') is None:
        return redirect('/logar')  # Redireciona para a página de login

    # Verifica se o tipo do usuário é 'adm'
    if session['usuario_logado']['tipo'] == "cliente":
        return redirect("/")  # Redireciona para a rota "/"
    adm = Adm()
    lista_guarnicao = adm.exibir_guarnicao()
    lista_acompanhamento = adm.exibir_acompanhamento()
    categorias = adm.exibir_categorias()
                        return          render_template("cad-produto.html",
lista_guarnicao=lista_guarnicao,    lista_acompanhamento=lista_acompanhamento,
categorias=categorias)



@app.route('/adicionar_guarnicao', methods=['POST'])
def adicionar_guarnicao():
    nome_guarnicao = request.form.get('nome_guarnicao')
    adm = Adm()
    if nome_guarnicao:
        sucesso, id_guarnicao = adm.inserir_guarnicao(nome_guarnicao)  #
Modifique a função para retornar o ID
        return jsonify(success=sucesso, id_guarnicao=id_guarnicao)
    return jsonify(success=False)




@app.route('/adicionar_acompanhamento', methods=['POST'])
def adicionar_acompanhamento():
    nome_acompanhamento = request.form.get('nome_acompanhamento')
    adm = Adm()
    if nome_acompanhamento:
                                    sucesso,       id_acompanhamento       =
adm.inserir_acompanhamento(nome_acompanhamento)
```

```
                                        return       jsonify(success=sucesso,
id_acompanhamento=id_acompanhamento)
          return jsonify(success=False)




      # Rota para exibição de produtos
      @app.route("/exibir_produtos")
      def compras():
          sistema = Sistema()  # Cria uma instância da classe Sistema
           lista_produtos  =  sistema.exibir_produtos()   #  Obtém  a  lista  de
produtos
                              return           render_template("produto.html",
lista_produtos=lista_produtos)  # Renderiza a página com a lista de produtos


      # Rota para exibir detalhes de um produto único
      @app.route("/produto_unico", methods=['GET', 'POST'])
      def exibir_produto_unico():

          if request.method == 'POST':
               btn_produto = request.form.get('btn-produto')  # Obtém o ID do
produto selecionado
               session['id'] = {'id_produto': btn_produto}  # Armazena o ID na
sessão

          # Recupera o ID do produto da sessão
          id_produto = session['id'].get('id_produto')

          if id_produto is None:
              flash('ID do produto não encontrado.', 'error')
                return  redirect('/')  #  Redireciona  se  o  ID  não  estiver  na
sessão

          sistema = Sistema()  # Cria uma instância da classe Sistema
          lista_prounico = sistema.exibir_produto(id_produto)

          if lista_prounico is None:
              flash('Produto não encontrado.', 'error')
              return redirect('/')  # Ou outra página que faça sentido

          # Renderiza o template com os detalhes do produto
                              return           render_template("produto.html",
lista_prounico=lista_prounico)


      # Rota para exibir detalhes de um produto único
```

```
@app.route("/marmita_unica", methods=['GET', 'POST'])
def exibir_marmita_unica():

    if request.method == 'POST':
        btn_marmita = request.form.get('btn-produto')  # Obtém o ID do produto selecionado
        session['id'] = {'id_marmita': btn_marmita}  # Armazena o ID na sessão

    # Recupera o ID do produto da sessão
    id_marmita = session['id'].get('id_marmita')

    if id_marmita is None:
        flash('ID do produto não encontrado.', 'error')
        return redirect('/')  # Redireciona se o ID não estiver na sessão

    sistema = Sistema()  # Cria uma instância da classe Sistema
    dados_marmita = sistema.exibir_marmita(id_marmita)  # Aqui agora retorna um único item

    if dados_marmita is None:
        flash('Produto não encontrado.', 'error')
        return redirect('/')  # Ou outra página que faça sentido

    # Renderiza o template com os detalhes do produto
    return render_template("marmita.html", marmita=dados_marmita[0])



# Habilitar e desabilitar o produto (adm)
@app.route("/desabilitar_produto_adm", methods=['POST'])
def desabilitar_produto_adm():
    try:
        adm = Adm()  # Cria uma instância da classe Sistema
        btn_desabilitar = request.form.get("btn_desabilitar")  # Obtém o ID do produto
        adm.desabilitar_produto_adm(btn_desabilitar)  # Desabilita o produto
        return jsonify({'status': 'success'})  # Retorna uma resposta de sucesso
    except Exception as e:
        return jsonify({'status': 'error', 'message': str(e)}), 500  # Retorna erro se algo falhar


@app.route("/habilitar_produto_adm", methods=['POST'])
def habilitar_produto_adm():
    try:
```

```
            adm = Adm()  # Cria uma instância da classe Sistema
            btn_habilitar = request.form.get("btn_habilitar")  # Obtém o ID
do produto
            adm.habilitar_produto_adm(btn_habilitar)  # Habilita o produto
            return jsonify({'status': 'success'})  # Retorna uma resposta de
sucesso
        except Exception as e:
            return jsonify({'status': 'error', 'message': str(e)}), 500  #
Retorna erro se algo falhar




    # Habilitar e desabilitar marmita (adm)
    @app.route("/desabilitar_marmita_adm", methods=['POST'])
    def desabilitar_marmita_adm():
        try:
            adm = Adm()  # Cria uma instância da classe Sistema
            btn_desabilitar = request.form.get("btn_desabilitar")  # Obtém o
ID da marmita
            adm.desabilitar_marmita_adm(btn_desabilitar)  # Desabilita a
marmita
            return jsonify({'status': 'success'})  # Retorna uma resposta de
sucesso
        except Exception as e:
            return jsonify({'status': 'error', 'message': str(e)}), 500  #
Retorna erro se algo falhar


    @app.route("/habilitar_marmita_adm", methods=['POST'])
    def habilitar_marmita_adm():
        try:
            adm = Adm()  # Cria uma instância da classe Sistema
            btn_habilitar = request.form.get("btn_habilitar")  # Obtém o ID
da marmita
            adm.habilitar_marmita_adm(btn_habilitar)  # Habilita a marmita
            return jsonify({'status': 'success'})  # Retorna uma resposta de
sucesso
        except Exception as e:
            return jsonify({'status': 'error', 'message': str(e)}), 500  #
Retorna erro se algo falhar




    # ========== Pedidos ==========
```

```
      @app.route("/exibir_pedidos", methods=['GET'])
      def exibir_pedidos_route():
          # Verifica se o usuário está logado e possui um ID de cliente válido
           if 'usuario_logado' not in session or session['usuario_logado'] is
None or session['usuario_logado'].get('id_cliente') is None:
              return redirect('/logar')  # Redireciona para a página de login

          # Verifica se o tipo do usuário é 'adm'
          if session['usuario_logado']['tipo'] == "cliente":
              return redirect("/")  # Redireciona para a rota "/"

          try:
              adm = Adm()  # Cria uma instância da classe Adm
              lista_pedidos = adm.exibir_pedidos()  # Obtém a lista de pedidos
                                  return     render_template('recebePedido.html',
lista_pedidos=lista_pedidos)  # Passa a variável para o template

          except Exception as e:
              print(f"Erro ao exibir pedidos: {e}")  # Log do erro
               return render_template('error.html', message="Erro ao carregar
pedidos.")  # Renderiza uma página de erro



      @app.route("/obter_pedidos", methods=['GET'])
      def obter_pedidos():
           if 'usuario_logado' not in session or session['usuario_logado'] is
None or session['usuario_logado'].get('id_cliente') is None:
               return jsonify({'redirect': '/logar'})  # Redireciona se não
estiver logado

          try:
              adm = Adm()  # Cria uma instância da classe Adm
              lista_pedidos = adm.exibir_pedidos()  # Obtém a lista de pedidos
              print(lista_pedidos)  # Debug: Verifique os dados retornados
               return jsonify(lista_pedidos)  # Retorna a lista de pedidos em
formato JSON

          except Exception as e:
              print(f"Erro ao obter pedidos: {e}")  # Log do erro
                 return jsonify({'error': "Erro ao carregar pedidos."})  #
Retorna mensagem de erro em JSON
```

```
      @app.route('/marcar_entregue/<int:id_pedido>', methods=['POST'])
      def marcar_entregue(id_pedido):
          try:
              adm = Adm()  # Cria uma instância da classe Adm
                  adm.atualizar_status_pedido_entregue(id_pedido)   #  Chama  a
função para atualizar o status
              return jsonify({'status': 'sucesso'})  # Retorna sucesso em JSON

          except Exception as e:
               print(f"Erro ao marcar pedido como entregue: {e}")  # Log do
erro
               return jsonify({'status': 'erro', 'mensagem': str(e)}), 500  #
Retorna mensagem de erro em JSON









      @app.route("/atualizar_status_pedido", methods=['POST'])
      def atualizar_status_pedido():
           if 'usuario_logado' not in session or session['usuario_logado'] is
None or session['usuario_logado'].get('id_cliente') is None:
               return jsonify({'redirect': '/logar'})  # Redireciona se não
estiver logado

          # Obtém o ID do pedido e o novo status enviados pelo AJAX
          id_pedido = request.form.get('id_pedido')
          novo_status = request.form.get('status')

          # Verifica se o ID do pedido e o status são válidos
          if id_pedido and novo_status:
              sistema = Sistema()  # Instância da classe Sistema
                      sucesso   =   sistema.atualizar_status_pedido(id_pedido,
novo_status)  # Atualiza o status do pedido no sistema

              if sucesso:
                    # Se o status for "pronto", envia uma mensagem para o
cliente
                  if novo_status.lower() == "feito":
                      try:
                          # Obtém os dados do cliente pelo ID do pedido
                                                                  cliente     =
sistema.obter_dados_cliente_por_pedido(id_pedido)
                          telefone_cliente = cliente.get('telefone')
                          nome_cliente = cliente.get('nome')
```

```
                        if telefone_cliente:
                            mensagem = f"Olá {nome_cliente}, seu pedido está
pronto! Pode retirar ou aguardar a entrega."
                            message = client.messages.create(
                                body=mensagem,
                                from_="+17753707822",
                                to=telefone_cliente
                            )
                                    print(f"Mensagem enviada com sucesso:
{message.sid}")
                    except Exception as e:
                        print(f"Erro ao enviar mensagem: {e}")

                return jsonify({'status': 'sucesso'})
            else:
                    return jsonify({'status': 'erro', 'mensagem': 'Não foi
possível atualizar o status.'})
        return jsonify({'status': 'erro', 'mensagem': 'Dados inválidos.'})


    @app.route("/cancelar_pedido", methods=['POST'])
    def cancelar_pedido():
        if 'usuario_logado' not in session or session['usuario_logado'] is
None or session['usuario_logado'].get('id_cliente') is None:
            return jsonify({'redirect': '/logar'})  # Redireciona se não
estiver logado

        id_pedido = request.form.get('id_pedido')
         motivo_cancelamento = request.form.get('motivo_cancelamento')  #
Obtém o motivo do cancelamento (select)
         descricao_cancelamento = request.form.get('descricao_cancelamento')
 # Obtém a descrição do cancelamento (textarea)

        # Verifica se o ID do pedido e pelo menos um motivo ou descrição do
cancelamento foram fornecidos
        if id_pedido and (motivo_cancelamento or descricao_cancelamento):
            sistema = Sistema()  # Cria uma instância da classe Sistema
             motivo_final = motivo_cancelamento if motivo_cancelamento else
descricao_cancelamento  # Usa o motivo ou a descrição
             sucesso = sistema.cancelar_pedido(id_pedido, motivo_final)  #
Passa o motivo para a função

            if sucesso:
                return jsonify({'status': 'sucesso'})
            else:
                    return jsonify({'status': 'erro', 'mensagem': 'Não foi
possível cancelar o pedido.'})
        return jsonify({'status': 'erro', 'mensagem': 'Dados inválidos.'})
```

```
@app.route("/enviar_carrinho", methods=['POST'])
def enviar_carrinho():
    try:
        timezone_sp = pytz.timezone('America/Sao_Paulo')
        now = datetime.now(timezone_sp)
        hora_atual = now.strftime("%H:%M:%S")  # Captura a hora atual no formato HH:MM:SS
        print(f"Hora atual: {hora_atual}")

         # Verifica se está dentro do horário de funcionamento (entre 7h00 e 21h45)
        site_aberto = (now.hour > 7 or (now.hour == 7 and now.minute >= 0)) and (now.hour < 21 or (now.hour == 21 and now.minute <= 45))

        if not site_aberto:
            return jsonify({"error": "Site offline. Não é possível enviar pedidos neste horário."}), 403

        if 'usuario_logado' in session:
            id_cliente = session['usuario_logado']['id_cliente']
            data = request.get_json()

            # Verifica se os dados foram enviados corretamente
            if data is None:
                return jsonify(success=False, message="Erro ao obter dados do carrinho."), 400

            itens = data.get('itens', [])

            # Verifica se os itens estão vazios
            if not itens:
                return jsonify(success=False, message="Carrinho está vazio."), 400

            try:
                carrinho = Carrinho()
                if carrinho.enviar_carrinho(id_cliente, itens, hora_atual):  # Passa a hora atual
                    # Chama a função para remover todo o carrinho
                    carrinho.remover_todo_carrinho(id_cliente)
                    return jsonify(success=True, message="Pedido enviado com sucesso!", redirect="/exibir_pedidos")
                else:
```

```
                                return jsonify(success=False, message="Erro ao
enviar o carrinho."), 500
                    except Exception as e:
                        print(f"Erro ao processar pedido: {e}")
                          return jsonify(success=False, message="Erro interno do
servidor."), 500

                        return   jsonify(success=False,   message="Usuário   não
autenticado."), 401

            except Exception as e:
                print(f"Erro ao obter horário de Brasília: {e}")
                   return jsonify({"error": "Erro ao verificar o horário de
funcionamento."}), 500









        # ========== Histórico de Pedidos ==========
        @app.route("/historico", methods=['GET'])
        def historico():
            if 'usuario_logado' not in session or session['usuario_logado'] is
None or session['usuario_logado'].get('id_cliente') is None:
                return redirect('/logar')  # Redireciona para a página de login
            else:
                 return render_template('historico.html')  # Carrega o template
da página de histórico


        @app.route("/exibir_historico_ajax", methods=['GET'])
        def exibir_historico_ajax():
            if 'usuario_logado' not in session or session['usuario_logado'] is
None or session['usuario_logado'].get('id_cliente') is None:
                 return jsonify({'redirect': '/logar'})  # Redireciona para a
página de login via JSON
            else:
                id_cliente = session['usuario_logado']['id_cliente']
                sistema = Sistema()  # Cria uma instância da classe Sistema
                 lista_historico = sistema.exibir_historico(id_cliente)  # Obtém
a lista de pedidos
```

```
            return jsonify(lista_historico)  # Retorna os pedidos como JSON


    # ========== Carrinho ==========

    # Rota para atualizar o preço total do carrinho via AJAX
    @app.route("/atualizar_preco_total", methods=['GET'])
    def atualizar_preco_total():
        if 'usuario_logado' not in session or session['usuario_logado'] is None or session['usuario_logado'].get('id_cliente') is None:
            return jsonify({'success': False, 'message': 'Usuário não autenticado'})

        id_cliente = session.get('usuario_logado')['id_cliente']

        try:
            carrinho = Carrinho()
            lista_carrinho = carrinho.exibir_carrinho(id_cliente)
            return jsonify({'success': True, 'total_preco': lista_carrinho['total_preco']})

        except Exception as e:
            print(f"Erro ao atualizar preço total: {e}")
            return jsonify({'success': False, 'message': 'Erro ao atualizar preço total'})


    # Rota para atualizar a quantidade de um produto no carrinho via AJAX
    @app.route("/atualizar_quantidade", methods=['POST'])
    def atualizar_quantidade():
        if 'usuario_logado' not in session or session['usuario_logado'] is None or session['usuario_logado'].get('id_cliente') is None:
            return jsonify({'success': False, 'message': 'Usuário não autenticado'})

        id_cliente = session.get('usuario_logado')['id_cliente']

        try:
            data = request.get_json()
            id_carrinho = data.get('id_carrinho')
            quantidade = data.get('quantidade')

            if not id_carrinho or not quantidade:
                return jsonify({'success': False, 'message': 'Dados incompletos'})

            carrinho = Carrinho()
```

```
                    carrinho.atualizar_quantidade_produto_carrinho(id_carrinho,
quantidade)

                return jsonify({'success': True})

            except Exception as e:
                print(f"Erro ao atualizar quantidade: {e}")
                return jsonify({'success': False, 'message': 'Erro ao atualizar
quantidade'})


        # Rota para excluir um item (produto ou marmita) do carrinho
        @app.route("/excluir_produto_carrinho", methods=['POST'])
        def excluir_produto_carrinho():
            # Verifica se o usuário está autenticado
            if 'usuario_logado' not in session or session['usuario_logado'] is
None or session['usuario_logado'].get('id_cliente') is None:
                return jsonify({'success': False, 'message': 'Usuário não
autenticado'})

            try:
                # Obtém os dados JSON da requisição
                data = request.get_json()
                id_carrinho = data.get('id_carrinho')  # Obtém o ID do item no
carrinho

                # Verifica se o ID do carrinho foi fornecido
                if not id_carrinho:
                    return jsonify({'success': False, 'message': 'ID do carrinho
não encontrado'})

                carrinho = Carrinho()  # Cria uma instância da classe Carrinho
                carrinho.remover_produto_carrinho(id_carrinho)  # Remove o item
do carrinho

                return jsonify({'success': True})  # Retorna sucesso se a
exclusão for bem-sucedida

            except Exception as e:
                print(f"Erro ao excluir item do carrinho: {e}")  # Registra o
erro no console
                return jsonify({'success': False, 'message': 'Erro ao excluir
item do carrinho'})



        @app.route("/exibir_carrinho", methods=['GET', 'POST'])
        def exibir_carrinho():
```

```
          if 'usuario_logado' not in session or session['usuario_logado'] is
None or session['usuario_logado'].get('id_cliente') is None:
              return redirect('/logar')  # Redireciona para a página de login
se o usuário não estiver autenticado
          else:
              carrinho = Carrinho()  # Cria uma instância da classe Carrinho
                 id_cliente  =  session.get('usuario_logado')['id_cliente']   #
Obtém o ID do cliente da sessão

              if request.method == 'POST':
                  if 'btn-excluir' in request.form:
                      id_carrinho = request.form['btn-excluir']
                      carrinho.remover_produto_carrinho(id_carrinho)
                  else:
                      quantidades = request.form.getlist('quantidades')
                      for id_carrinho, quantidade in quantidades.items():

carrinho.atualizar_quantidade_produto_carrinho(id_carrinho, quantidade)

              lista_carrinho = carrinho.exibir_carrinho(id_cliente)  # Obtém a
lista de produtos no carrinho
                                    return      render_template("carrinho.html",
lista_carrinho=lista_carrinho)  # Renderiza a página do carrinho com a lista
de produtos


      @app.route("/inserir_carrinho", methods=['POST'])
      def carrinho():
          if 'usuario_logado' not in session or session['usuario_logado'] is
None or session['usuario_logado'].get('id_cliente') is None:
              return jsonify({"redirect_url": "/logar"}), 401  # Retorna 401
para redirecionamento

          # Lógica de inserção no carrinho
          id_cliente = session.get('usuario_logado')['id_cliente']
          cod_produto = request.form.get('cod_produto')
          id_marmita = request.form.get('id_marmita')

          if id_marmita:
              timezone_sp = pytz.timezone('America/Sao_Paulo')
              now = datetime.now(timezone_sp)
              hora_atual = now.hour
              minutos_atuais = now.minute

              if (hora_atual < 7) or (hora_atual == 15 and minutos_atuais >
30) or (hora_atual > 15):
                  return jsonify({"error": "Os pedidos de marmitas só podem
ser feitos entre 7h e 15h30."}), 403
```

```
          guarnicoes_selecionadas = request.form.getlist('guarnicao')
                                    acompanhamentos_selecionados              =
request.form.getlist('acompanhamento')

          carrinho = Carrinho()
          if cod_produto or id_marmita:
                      carrinho.inserir_item_carrinho(cod_produto,   id_marmita,
id_cliente,
                                                     guarnicoes_selecionadas,
acompanhamentos_selecionados)

                  return     jsonify({"success":     True,     "redirect_url":
"/exibir_carrinho"})









      @app.route("/editar_produto", methods=['POST', 'GET'])
      def editar_produto():
           if 'usuario_logado' not in session or session['usuario_logado'] is
None or session['usuario_logado'].get('id_cliente') is None:
               return redirect('/logar')  # Redireciona para a página de login
se o usuário não estiver autenticado

          if session['usuario_logado'].get('tipo') == 'cliente':
                return  redirect('/')  # Redireciona clientes para a página
inicial

          if request.method == 'POST':
               btn_produto = request.form.get('btn-produto')  # Obtém o ID do
produto selecionado
               session['id'] = {'id_produto': btn_produto}  # Armazena o ID na
sessão

          # Recupera o ID do produto da sessão
```

```
          id_produto = session['id'].get('id_produto')

          if id_produto is None:
              flash('ID do produto não encontrado.', 'error')
                return redirect('/')  # Redireciona se o ID não estiver na
sessão

          sistema = Sistema()  # Cria uma instância da classe Sistema
          lista_prounico = sistema.exibir_produto(id_produto)

          if lista_prounico is None:
              flash('Produto não encontrado.', 'error')
              return redirect('/')  # Ou outra página que faça sentido

          # Renderiza o template com os detalhes do produto
                          return         render_template("editarProduto.html",
lista_prounico=lista_prounico)


      @app.route("/atualizar_produto", methods=['POST'])
      def atualizar_produto():
          if 'usuario_logado' not in session:
              return redirect('/logar')

          if session['usuario_logado'].get('tipo') == 'cliente':
                return redirect('/')  # Redireciona clientes para a página
inicial

          adm = Adm()  # Cria uma instância da classe Sistema
          id_produto = request.form.get('id_produto')
          nome = request.form.get('nome')
          preco = request.form.get('preco')
          descricao = request.form.get('descricao')
          imagem = request.files.get('imagem')  # Para o upload de imagem

           # Lógica para salvar o arquivo localmente (opcional, caso esteja
enviando a URL no banco)
          imagem_url = None
          if imagem:
              caminho_imagem = os.path.join('static/uploads', imagem.filename)
              imagem.save(caminho_imagem)
              imagem_url = f"/static/uploads/{imagem.filename}"

          # Atualizar o produto no banco de dados
               adm.atualizar_produto(id_produto,   nome,   preco,   descricao,
imagem_url)

          flash('Produto atualizado com sucesso!', 'success')
          return redirect('/editar_produto')
```

```
@app.route('/imagem_produto/<int:cod_produto>')
def imagem_produto(cod_produto):
    adm = Adm()  # Cria uma instância da classe Sistema
    imagem = adm.obter_imagem_produto(cod_produto)

    if imagem:
        return Response(imagem, mimetype='image/jpeg')  # Ajuste o tipo MIME conforme o tipo de imagem armazenado
    return "Imagem não encontrada", 404  # Retorna erro 404 se não encontrar a imagem




@app.route("/editar_marmita", methods=['POST', 'GET'])
def editar_marmita():
    if 'usuario_logado' not in session or session['usuario_logado'] is None or session['usuario_logado'].get('id_cliente') is None:
        return redirect('/logar')  # Redireciona para a página de login se o usuário não estiver autenticado

    if session['usuario_logado'].get('tipo') == 'cliente':
        return redirect('/')  # Redireciona clientes para a página inicial


    if request.method == 'POST':
        btn_marmita = request.form.get('btn-marmita')  # Obtém o ID do produto selecionado
        session['id'] = {'id_marmita': btn_marmita}  # Armazena o ID na sessão

    id_marmita = session['id'].get('id_marmita')

    if id_marmita is None:
        flash('ID da marmita não encontrado.', 'error')
        return redirect('/adm')  # Redireciona se o ID não estiver na sessão

    sistema = Sistema()  # Cria uma instância da classe
    lista_marunica = sistema.exibir_marmita(id_marmita)

    if not lista_marunica:
```

```
            flash('Nenhuma marmita encontrada.', 'error')
            return redirect('/inicialadm')  # Ou outra página relevante

        # Extrai os dados retornados, incluindo as guarnições e acompanhamentos
        marmita_dados = lista_marunica[0]

        # Renderiza o template com todos os dados necessários
        return render_template(
            "editarMarmita.html",
            lista_marunica=lista_marunica,
            todos_acompanhamentos=marmita_dados['todos_acompanhamentos'],  # Passa todos os acompanhamentos
            todas_guarnicoes=marmita_dados['todas_guarnicoes']  # Passa todas as guarnições
        )







    @app.route("/atualizar_marmita", methods=['POST'])
    def atualizar_marmita():
        if 'usuario_logado' not in session:
            return jsonify({'status': 'error', 'message': 'Você precisa estar logado para realizar esta ação.'}), 401


        if session['usuario_logado'].get('tipo') == 'cliente':
            return redirect('/')  # Redireciona clientes para a página inicial

        adm = Adm()
        id_marmita = request.form['id_marmita']
        nome = request.form['nome']
        preco = request.form['preco']
        descricao = request.form['descricao']
        tamanho = request.form['tamanho']
        imagem = request.files.get('imagem')

        # Capturando acompanhamentos e guarnições selecionados
        acompanhamentos = request.form.getlist('acompanhamentos[]')
        guarnicoes = request.form.getlist('guarnicoes[]')

        # Validação dos campos obrigatórios
        if not id_marmita or not nome or not preco or not descricao or not tamanho:
```

```
                return jsonify({'status': 'error', 'message': 'Todos os campos
obrigatórios devem ser preenchidos.'}), 400

            try:
                # Atualiza a marmita com os dados fornecidos
                    adm.atualizar_marmita(id_marmita, nome, preco, descricao,
tamanho, acompanhamentos, guarnicoes, imagem)
                    return jsonify({'status': 'success', 'message': 'Marmita
atualizada com sucesso!'})
            except Exception as e:
                return jsonify({'status': 'error', 'message': f'Ocorreu um erro
ao atualizar a marmita: {str(e)}'}), 500









        @app.route('/imagem_marmita/<int:id_marmita>')
        def imagem_marmita(id_marmita):
            adm = Adm()
            imagem = adm.obter_imagem_marmita(id_marmita)

            if imagem:
                return Response(imagem, mimetype='image/jpeg')  # Ajuste o tipo
MIME conforme o tipo de imagem armazenado
            return "Imagem não encontrada", 404  # Retorna erro 404 se não
encontrar a imagem









        # Rota para solicitar troca de senha
        @app.route("/trocar_senha", methods=['GET', 'POST'])
        def trocar_senha():
            if request.method == 'GET':
                    return render_template("trocar-senha.html")  # Renderiza a
página para troca de senha
            else:
                email = request.form['email']
                telefone = request.form['telefone']

                usuario = Usuario()
```

```
            if usuario.verificar_usuario(email, telefone):  # Verifica se o
usuário existe
                verification_code = str(random.randint(1000, 9999)).zfill(4)

                # Envia o código via SMS
                message = client.messages.create(
                    to=telefone,
                    from_="+13195190041",
                              body=f'Seu  código  para  troca  de  senha  é:
{verification_code}'
                )
                print(message.sid)

                # Armazena informações na sessão
                session['telefone_verificacao'] = telefone
                session['verification_code'] = verification_code
                session['email_usuario'] = email

                 return redirect("/verificacao_troca_senha")  # Redireciona
para a verificação
            else:
                 flash("Usuário não encontrado. Verifique as informações.",
"error")
                return redirect("/trocar_senha")


    # Rota para verificação do código de troca de senha
    @app.route("/verificacao_troca_senha", methods=['GET', 'POST'])
    def verificacao_troca_senha():
        if 'verification_code' not in session:  # Verifica se o código está
na sessão
             flash("Sessão expirada. Solicite a troca de senha novamente.",
"error")
            return redirect("/trocar_senha")

        if request.method == 'GET':
            return render_template("verificacao-troca-senha.html")
        else:
            codigo1 = request.form["codigo1"]
            codigo2 = request.form["codigo2"]
            codigo3 = request.form["codigo3"]
            codigo4 = request.form["codigo4"]
            codigo_inserido = codigo1 + codigo2 + codigo3 + codigo4
            verification_code = session.get('verification_code')

            if codigo_inserido == verification_code:
                    session.pop('verification_code')  # Remove o código da
sessão após validação
```

```
                    session['verificado'] = True  # Marca que o usuário foi
verificado
                return redirect("/nova_senha")
            else:
                        return  render_template("verificacao-troca-senha.html",
erro="Código incorreto. Tente novamente.")


    # Rota para definir uma nova senha
    @app.route("/nova_senha", methods=['GET', 'POST'])
    def nova_senha():
        if 'email_usuario' not in session or not session.get('verificado'):
 # Verifica se passou pela verificação
            flash("Acesso inválido. Solicite a troca de senha novamente.",
"error")
            return redirect("/trocar_senha")

        if request.method == 'GET':
            return render_template("nova-senha.html")

        nova_senha = request.form['nova_senha']
        nova_senha_confirma = request.form['nova_senha_confirma']
        email_usuario = session.get('email_usuario')

        if nova_senha != nova_senha_confirma:
            flash("As senhas não coincidem. Tente novamente.", "error")
            return redirect("/nova_senha")

        usuario = Usuario()
        try:
            if usuario.atualizar_senha(email_usuario, nova_senha):
                flash("Senha atualizada com sucesso!", "success")
                session.clear()  # Limpa a sessão após a troca de senha
                return redirect("/logar")
            else:
                    flash("Erro ao atualizar a senha. Tente novamente.",
"error")
        except Exception as e:
            flash(f"Ocorreu um erro: {str(e)}", "error")

        return redirect("/nova_senha")



    # ================ PERFIL ================
    @app.route('/perfil', methods=['GET'])
    def perfil():
        if 'usuario_logado' not in session or session['usuario_logado'] is
None:
```

```
            return redirect('/logar')  # Redireciona para a página de login se o usuário não estiver autenticado

        # Recupera o ID do cliente da sessão
        id_cliente = session['usuario_logado'].get('id_cliente')

        perfil = Perfil()  # Cria uma instância da classe Sistema
        perfil_usuario = perfil.obter_perfil(id_cliente)  # Método que você deve criar para obter os detalhes do usuário

        if perfil_usuario is None:
            flash('Perfil não encontrado.', 'error')
            return redirect('/')  # Redireciona se o perfil não for encontrado

        # Renderiza o template com os detalhes do perfil
        return render_template("perfil.html", perfil_usuario=perfil_usuario)


    @app.route('/atualizar_perfil', methods=['POST'])
    def atualizar_perfil():
        if 'usuario_logado' not in session:
            return redirect('/logar')  # Redireciona se o usuário não estiver logado

        perfil = Perfil()  # Cria uma instância da classe Sistema

        # Obtém dados do formulário
        nome = request.form.get('nome')
        senha = request.form.get('senha')
        confirmar_senha = request.form.get('confirmar_senha')
        imagem_perfil = request.files.get('imagem_perfil')

        # Verifica se as senhas coincidem
        if senha != confirmar_senha:
            flash('As senhas não coincidem.', 'error')
            return redirect('/perfil')  # Redireciona para o perfil se houver erro

        # Obter o ID do cliente da sessão
        id_cliente = session['usuario_logado']['id_cliente']

        # Verifica a senha atual para confirmar a alteração
        if not perfil.verificar_senha(id_cliente, senha):
            flash('Senha incorreta.', 'error')
            return redirect('/perfil')

        # Verifica se uma imagem foi enviada
        caminho_imagem = None
```

```
          if imagem_perfil and imagem_perfil.filename != '':
                  caminho_imagem  =  os.path.join(app.config['UPLOAD_FOLDER'],
imagem_perfil.filename)
              try:
                  imagem_perfil.save(caminho_imagem)
                  flash('Imagem salva com sucesso!', 'success')
              except Exception as e:
                  flash(f'Erro ao salvar a imagem: {str(e)}', 'error')
                     return  redirect('/perfil')  # Redireciona se falhar ao
salvar a imagem

          # Atualiza nome e imagem (sem alterar a senha), verifica se o nome
foi preenchido
           resultado  =  perfil.atualizar_perfil(id_cliente,  nome  if  nome  else
None, caminho_imagem)

          if 'error' in resultado:
              flash(resultado['error'], 'error')
          else:
              flash('Perfil atualizado com sucesso!', 'success')

           return  redirect('/perfil')  # Redireciona para a página de perfil
após a atualização



      @app.route('/imagem_perfil/<int:id_cliente>')
      def imagem_perfil(id_cliente):
          perfil = Perfil()  # Cria uma instância da classe Sistema
          imagem = perfil.obter_imagem_perfil(id_cliente)

          if imagem:
              return Response(imagem, mimetype='image/jpeg')  # ou o tipo MIME
correto para a imagem
          else:
               # Retorna a imagem padrão caso não exista imagem personalizada
para o usuário
                    return   redirect(url_for('static',   filename='img/default-
avatar.png'))



      @app.route('/relatorio', methods=['GET', 'POST'])
      def relatorio():
          # Verifica se o usuário está logado e possui um ID de cliente válido
           if 'usuario_logado' not in session or session['usuario_logado'] is
None or session['usuario_logado'].get('id_cliente') is None:
              return redirect('/logar')  # Redireciona para a página de login
```

```
        # Verifica se o tipo do usuário é 'adm'
        if session['usuario_logado']['tipo'] == "cliente":
            return redirect("/")  # Redireciona para a rota "/"
        relatorio = Relatorio()
        relatorio_dados = []
        valor_total_geral = 0
        total_pedidos = 0
        cancelados_dados = []
        valor_total_cancelado = 0
        total_cancelados = 0

        if request.method == 'POST':
            data_inicial = request.form['data_inicial']
            data_final = request.form['data_final']
                    relatorio_dados,   valor_total_geral,   total_pedidos   =
relatorio.exibir_relatorio_entregue(data_inicial, data_final)
                cancelados_dados,  valor_total_cancelado,  total_cancelados  =
relatorio.exibir_relatorio_cancelado(data_inicial, data_final)

        return render_template("relatorio.html",
                               relatorio_dados=relatorio_dados,
                               valor_total_geral=valor_total_geral,
                               total_pedidos=total_pedidos,
                               cancelados_dados=cancelados_dados,
                               valor_total_cancelado=valor_total_cancelado,
                               total_cancelados=total_cancelados)




    @app.route('/usuario/<int:id_cliente>', methods=['GET'])
    def usuario(id_cliente):
        if 'usuario_logado' not in session:
              return  redirect('/logar')  # Redireciona se o usuário não
estiver logado

         usuario = Usuario()  # Supondo que 'Usuario' seja uma classe que
manipula os dados do cliente
          dados_cliente  =  usuario.tela_usuario(id_cliente)  # Método  que
retorna os dados do cliente
         pedidos_cliente  =  usuario.obter_pedidos(id_cliente)  # Método  que
retorna os pedidos do cliente

        # Adiciona a URL da imagem de perfil no contexto
        imagem_perfil_url = url_for('imagem_perfil', id_cliente=id_cliente)
```

```
            return   render_template('usuario.html',   cliente=dados_cliente,
pedidos=pedidos_cliente, imagem_perfil_url=imagem_perfil_url)




    if __name__ == "__main__":
        app.run(debug=True, host="0.0.0.0", port=8080)  # Define o host como
localhost e a porta como 8080
```

```
    from conexao import Conexao
    from hashlib import sha256
    import os
    from datetime import datetime

    class Adm:
        def __init__(self):
            # Inicializa a classe Sistema sem variáveis de instância
necessárias.
            self.tel = None
            self.id_produto = None


        def exibir_pedidos(self):
            mydb = Conexao.conectar()
            mycursor = mydb.cursor()

            # Consulta SQL para obter os pedidos com detalhes de produtos e
marmitas
            sql = """
                        SELECT  p.id_pedido,  cl.id_cliente,  cl.nome_comp,
cl.telefone,
                               pr.nome_produto,  pr.preco  AS  preco_produto,
pp.quantidade,
                        m.nome_marmita, m.preco AS preco_marmita, m.tamanho,
m.descricao,
                    p.data_pedido, p.status, p.hora_pedido
                FROM tb_pedidos p
                JOIN tb_cliente cl ON p.id_cliente = cl.id_cliente
                JOIN tb_produtos_pedidos pp ON p.id_pedido = pp.id_pedido
                LEFT JOIN tb_produto pr ON pp.cod_produto = pr.cod_produto
                LEFT JOIN tb_marmita m ON pp.id_marmita = m.id_marmita
                WHERE p.habilitado = 1

            """
            mycursor.execute(sql)
            resultados = mycursor.fetchall()
```

```
            pedidos = {}

            # Itera sobre os resultados e organiza as informações de pedidos
            for resultado in resultados:
                id_pedido = resultado[0]
                id_cliente = resultado[1]
                nome_cliente = resultado[2]
                telefone_cliente = resultado[3]
                nome_produto = resultado[4]
                preco_produto = resultado[5]
                quantidade_produto = resultado[6]
                nome_marmita = resultado[7]
                preco_marmita = resultado[8]
                tamanho_marmita = resultado[9]
                descricao_marmita = resultado[10]
                      data_pedido  =  resultado[11].strftime('%d/%m/%Y')  if
resultado[11] else None
                status_pedido = resultado[12]
                 hora_pedido = str(resultado[13]) if resultado[13] else None
 # Converte hora_pedido para string

                # Adiciona o cliente se não estiver no dicionário
                if id_cliente not in pedidos:
                    pedidos[id_cliente] = {
                        'nome_cliente': nome_cliente,
                        'telefone': telefone_cliente,
                        'pedidos': {}
                    }

                # Adiciona o pedido se não estiver no dicionário
                if id_pedido not in pedidos[id_cliente]['pedidos']:
                    pedidos[id_cliente]['pedidos'][id_pedido] = {
                        'data_pedido': data_pedido,
                        'status': status_pedido,
                             'hora': hora_pedido,  # Usa a hora_pedido já
formatada
                        'produtos': [],
                        'marmitas': [],
                        'guarnicoes': [],
                        'acompanhamentos': [],
                        'total_preco': 0
                    }

                # Adiciona o produto ao pedido, se houver
                if nome_produto:
                            preco_produto_float  =  float(preco_produto)  if
preco_produto is not None else 0.0
```

```
pedidos[id_cliente]['pedidos'][id_pedido]['produtos'].append({
                        'nome_produto': nome_produto,
                        'preco': preco_produto_float,
                        'quantidade': quantidade_produto
                    })
                    pedidos[id_cliente]['pedidos'][id_pedido]['total_preco']
+= preco_produto_float * quantidade_produto

                # Adiciona a marmita ao pedido, se houver
                if nome_marmita:
                            preco_marmita_float  =  float(preco_marmita)  if
preco_marmita is not None else 0.0

pedidos[id_cliente]['pedidos'][id_pedido]['marmitas'].append({
                        'nome_marmita': nome_marmita,
                        'preco': preco_marmita_float,
                        'tamanho': tamanho_marmita,
                        'descricao': descricao_marmita,
                        'quantidade': quantidade_produto
                    })
                    pedidos[id_cliente]['pedidos'][id_pedido]['total_preco']
+= preco_marmita_float * quantidade_produto

            # Buscar guarnições e acompanhamentos para cada pedido
            for id_cliente, dados in pedidos.items():
                for id_pedido in dados['pedidos']:
                    # Buscar guarnições
                    sql_guarnicoes = """
                        SELECT g.nome_guarnicao
                        FROM tb_guarnicoes_pedidos AS cg
                                JOIN  tb_guarnicao  AS  g  ON  cg.guarnicao  =
g.id_guarnicao
                        WHERE cg.id_pedido = %s
                    """
                    mycursor.execute(sql_guarnicoes, (id_pedido,))
                    guarnicoes = [row[0] for row in mycursor.fetchall()]
                     pedidos[id_cliente]['pedidos'][id_pedido]['guarnicoes']
= guarnicoes

                    # Buscar acompanhamentos
                    sql_acompanhamentos = """
                        SELECT a.nome_acompanhamento
                        FROM tb_acompanhamentos_pedidos AS ca
                         JOIN tb_acompanhamentos AS a ON ca.acompanhamento =
a.id_acompanhamento
                        WHERE ca.id_pedido = %s
                    """
                    mycursor.execute(sql_acompanhamentos, (id_pedido,))
```

```
                                    acompanhamentos = [row[0] for row in
mycursor.fetchall()]

pedidos[id_cliente]['pedidos'][id_pedido]['acompanhamentos'] = acompanhamentos

            mydb.close()

            # Ordenar pedidos por id_cliente e id_pedido
            pedidos_ordenados = {
                cliente_id: {
                    'nome_cliente': dados['nome_cliente'],
                    'telefone': dados['telefone'],
                            'pedidos':  dict(sorted(dados['pedidos'].items(),
key=lambda item: item[0]))
                }
                 for cliente_id, dados in sorted(pedidos.items(), key=lambda
item: item[0])
            }

            return pedidos_ordenados




        def atualizar_status_pedido_entregue(self, id_pedido):
            mydb = Conexao.conectar()  # Conecta ao banco de dados
            mycursor = mydb.cursor()   # Cria um cursor

            try:
                # Atualiza status e habilitado em um único comando
                mycursor.execute(
                    """
                    UPDATE tb_pedidos
                    SET status = %s, habilitado = %s
                    WHERE id_pedido = %s
                    """,
                    ('entregue', False, id_pedido)
                )
                mydb.commit()  # Confirma as alterações no banco de dados
            except Exception as e:
                 print(f"Erro ao atualizar o status do pedido: {e}")  # Log
do erro
                mydb.rollback()  # Reverte em caso de erro
            finally:
                mycursor.close()  # Fecha o cursor
                mydb.close()  # Fecha a conexão
```

```
            #  ========================  desabilitar  /  habilitar  produto
==================================
          # Modifique a função para desabilitar produto
          def desabilitar_produto_adm(self, produto_id):
              mydb = Conexao.conectar()
              mycursor = mydb.cursor()

              # Atualizar o status do produto para desabilitado (0)
               sql = "UPDATE tb_produto SET habilitado = 0 WHERE cod_produto =
%s"
              mycursor.execute(sql, (produto_id,))

              mydb.commit()
              mydb.close()

                 return {"message": "Produto desabilitado com sucesso!"}  #
Retorna um dicionário


          # Modifique a função para habilitar produto
          def habilitar_produto_adm(self, produto_id):
              mydb = Conexao.conectar()
              mycursor = mydb.cursor()

              # Atualizar o status do produto para habilitado (1)
               sql = "UPDATE tb_produto SET habilitado = 1 WHERE cod_produto =
%s"
              mycursor.execute(sql, (produto_id,))

              mydb.commit()
              mydb.close()

              return {"message": "Produto habilitado com sucesso!"}  # Retorna
um dicionário




          # Modifique a função para desabilitar marmita
          def desabilitar_marmita_adm(self, marmita_id):
```

```
            mydb = Conexao.conectar()
            mycursor = mydb.cursor()

            # Atualizar o status da marmita para desabilitado (0)
             sql = "UPDATE tb_marmita SET habilitado = 0 WHERE id_marmita =
%s"
            mycursor.execute(sql, (marmita_id,))

            mydb.commit()
            mydb.close()

               return {"message": "Marmita desabilitada com sucesso!"}  #
Retorna um dicionário


        # Modifique a função para habilitar marmita
        def habilitar_marmita_adm(self, marmita_id):
            mydb = Conexao.conectar()
            mycursor = mydb.cursor()

            # Atualizar o status da marmita para habilitado (1)
             sql = "UPDATE tb_marmita SET habilitado = 1 WHERE id_marmita =
%s"
            mycursor.execute(sql, (marmita_id,))

            mydb.commit()
            mydb.close()

            return {"message": "Marmita habilitada com sucesso!"}  # Retorna
um dicionário




        # ====================== Inserir Produtos ========================
          def inserir_produto(self, nomeP, preco, imagem_blob, descricao,
categoria, guarnicoes_novas=[]):
            mydb = Conexao.conectar()
            mycursor = mydb.cursor()

            # Query SQL para inserir o produto
             sql = "INSERT INTO tb_produto (nome_produto, preco, imagem_blob,
descricao, id_categoria) VALUES (%s, %s, %s, %s, %s)"
            valores = (nomeP, preco, imagem_blob, descricao, categoria)
            mycursor.execute(sql, valores)

               # Captura o ID do produto recém-inserido para associar
guarnições
```

```
            id_produto = mycursor.lastrowid

            # Inserir novas guarnições e associá-las ao produto
            for nova_guarnicao in guarnicoes_novas:
                self.inserir_guarnicao(nova_guarnicao)
                      sql_associacao = "INSERT INTO tb_produto_guarnicao
(id_produto, id_guarnicao) VALUES (%s, %s)"
                                mycursor.execute(sql_associacao, (id_produto,
nova_guarnicao))

            mydb.commit()
            mydb.close()
            return True



        def inserir_marmita(self, nomeP, preco, imagem, descricao, tamanho,
guarnicoes_existentes=[], guarnicoes_novas=[], acompanhamentos_existentes=[],
acompanhamentos_novos=[]):
            """
              Insere uma nova marmita no sistema, associando-a à categoria
correta. As informações da marmita
              incluem nome, preço, URL da imagem, descrição, tamanho e as
guarnições e acompanhamentos associados.

            Parâmetros:
            - nomeP: nome da marmita.
            - preco: preço da marmita.
            - imagem: URL da imagem da marmita.
            - descricao: breve descrição da marmita.
            - tamanho: tamanho da marmita (Pequena, Média, Grande).
                - guarnicoes_existentes: lista de IDs das guarnições já
existentes.
            - guarnicoes_novas: lista de novas guarnições a serem inseridas.
             - acompanhamentos_existentes: lista de IDs dos acompanhamentos
já existentes.
            - acompanhamentos_novos: lista de novos acompanhamentos a serem
inseridos.

            Retorno:
             - Retorna True se a marmita for inserida com sucesso, ou False
em caso de erro.
            """
            mydb = Conexao.conectar()  # Conecta ao banco de dados
            mycursor = mydb.cursor()

            # Inserir marmita na tabela `tb_marmita`
            sql = f"""
```

```
                INSERT INTO tb_marmita (nome_marmita, preco, imagem_binaria,
descricao, tamanho)
              VALUES (%s, %s, %s, %s, %s)
              """
                   mycursor.execute(sql, (nomeP, preco, imagem, descricao,
tamanho))

              # Capturar o ID da marmita recém-inserida
              id_marmita = mycursor.lastrowid

              # Associar guarnições existentes à marmita
              for id_guarnicao in guarnicoes_existentes:
                        sql_associacao = "INSERT INTO tb_marmita_guarnicao
(id_marmita, id_guarnicao) VALUES (%s, %s)"
                  mycursor.execute(sql_associacao, (id_marmita, id_guarnicao))

              # Inserir e associar novas guarnições
              for nova_guarnicao in guarnicoes_novas:
                  _, id_guarnicao = self.inserir_guarnicao(nova_guarnicao)
                        sql_associacao = "INSERT INTO tb_marmita_guarnicao
(id_marmita, id_guarnicao) VALUES (%s, %s)"
                  mycursor.execute(sql_associacao, (id_marmita, id_guarnicao))

              # Associar acompanhamentos existentes à marmita
              for id_acompanhamento in acompanhamentos_existentes:
                     sql_associacao = "INSERT INTO tb_marmita_acompanhamento
(id_marmita, id_acompanhamento) VALUES (%s, %s)"
                                mycursor.execute(sql_associacao,   (id_marmita,
id_acompanhamento))

              # Inserir e associar novos acompanhamentos
              for novo_acompanhamento in acompanhamentos_novos:
                                                  _,      id_acompanhamento      =
self.inserir_acompanhamento(novo_acompanhamento)
                     sql_associacao = "INSERT INTO tb_marmita_acompanhamento
(id_marmita, id_acompanhamento) VALUES (%s, %s)"
                                mycursor.execute(sql_associacao,   (id_marmita,
id_acompanhamento))

              mydb.commit()  # Confirma as alterações
              mydb.close()  # Fecha a conexão
              return True




          def exibir_guarnicao(self):
```

```
            mydb = Conexao.conectar()  # Conecta ao banco de dados
            mycursor = mydb.cursor()
            # Query SQL para inserir o produto na tabela `tb_produto`
            sql = f"SELECT * FROM tb_guarnicao"
            mycursor.execute(sql)
            resultado = mycursor.fetchall()  # Obtém todos os resultados

            lista_guarnicao = []

            # Itera sobre os resultados e adiciona cada produto à lista
            for produto in resultado:
                lista_guarnicao.append({
                    'nome_guarnicao': produto[1],
                    'id_guarnicao': produto[0]
                })

            mydb.close()  # Fecha a conexão com o banco de dados
             return lista_guarnicao if lista_guarnicao else []  # Retorna a
lista de produtos ou uma lista vazia se nenhum produto for encontrado



        def inserir_guarnicao(self, nome_guarnicao):
            mydb = Conexao.conectar()  # Conecta ao banco de dados
            mycursor = mydb.cursor()

            # Query SQL para inserir a nova guarnição
            sql = "INSERT INTO tb_guarnicao (nome_guarnicao) VALUES (%s)"
            mycursor.execute(sql, (nome_guarnicao,))

            mydb.commit()  # Confirma as alterações no banco de dados
              id_guarnicao = mycursor.lastrowid  # Captura o ID da nova
guarnição
            mydb.close()  # Fecha a conexão
            return True, id_guarnicao  # Retorna True e o ID



        def inserir_acompanhamento(self, nome_acompanhamento):
            mydb = Conexao.conectar()
            mycursor = mydb.cursor()

              sql = "INSERT INTO tb_acompanhamentos (nome_acompanhamento)
VALUES (%s)"
            mycursor.execute(sql, (nome_acompanhamento,))

            mydb.commit()
            id_acompanhamento = mycursor.lastrowid
            mydb.close()
```

```
            return True, id_acompanhamento



        def exibir_acompanhamento(self):
            mydb = Conexao.conectar()
            mycursor = mydb.cursor()
            sql = "SELECT * FROM tb_acompanhamentos"
            mycursor.execute(sql)
            resultado = mycursor.fetchall()

            lista_acompanhamento = []
            for acompanhamento in resultado:
                lista_acompanhamento.append({
                    'nome_acompanhamento': acompanhamento[1],
                    'id_acompanhamento': acompanhamento[0]
                })

            mydb.close()
            return lista_acompanhamento if lista_acompanhamento else []







        def exibir_categorias(self):
            """
                Retorna uma lista com todas as categorias de produtos
disponíveis, consultando a tabela `tb_categoria`.
            Cada categoria contém o ID da categoria e o nome da categoria.

            Retorno:
             - Uma lista de dicionários, onde cada dicionário representa uma
categoria com 'id_categoria' e 'nome'.
            """
            mydb = Conexao.conectar()  # Conecta ao banco de dados
            mycursor = mydb.cursor()

            # Query SQL para selecionar todas as categorias
            sql = "SELECT * from tb_categoria"
            mycursor.execute(sql)

            # Obtém os resultados e os organiza em uma lista
            resultado = mycursor.fetchall()
               lista_categorias = [{'id_categoria': categoria[0], 'nome':
categoria[1]} for categoria in resultado]
```

```
            mydb.commit()
            mydb.close()
            return lista_categorias


                #      ============================      Editar      Prduto
==============================
        def  atualizar_produto(self,  id_produto,  nome,  preco,  descricao,
url_img=None):
            mydb = Conexao.conectar()
            mycursor = mydb.cursor()

            # Base da consulta SQL
            sql = """
                UPDATE tb_produto
                SET nome_produto = %s, preco = %s, descricao = %s
            """
            valores = [nome, preco, descricao]

            # Adicionar a imagem caso fornecida
            if url_img:
                sql += ", url_img = %s"
                valores.append(url_img)

            sql += " WHERE cod_produto = %s"
            valores.append(id_produto)

            # Executa a consulta
            mycursor.execute(sql, valores)
            mydb.commit()
            mydb.close()



        def obter_imagem_produto(self, cod_produto):
            mydb = Conexao.conectar()
            mycursor = mydb.cursor()

            # Consulta SQL paa robter a imagem do produto
            sql = "SELECT imagem_binaria FROM tb_produto WHERE cod_produto =
%s"
            mycursor.execute(sql, (cod_produto,))
            resultado = mycursor.fetchone()

            mydb.close()

            if resultado:
                return resultado[0]  # Retorna a imagem binária
            return None  # Retorna None se não encontrar
```

```
            def atualizar_marmita(self, id_marmita, nome, preco, descricao,
tamanho, acompanhamentos, guarnicoes, file=None):
                mydb = Conexao.conectar()
                mycursor = mydb.cursor()

                # Atualiza os dados da marmita
                sql = """
                    UPDATE tb_marmita
                    SET nome_marmita = %s, preco = %s, descricao = %s, tamanho =
%s
                """
                valores = (nome, preco, descricao, tamanho)

                # Se um arquivo de imagem foi enviado
                if file and file.filename != '':
                    diretorio = os.path.join('static', 'uploads')
                    filename = f"marmita_{id_marmita}.jpg"
                    caminho_imagem = os.path.join(diretorio, filename)
                    file.save(caminho_imagem)

                    # Atualiza o campo da imagem
                    sql += ", url_img = %s"
                    valores += (f"/static/uploads/{filename}",)

                sql += " WHERE id_marmita = %s"
                valores += (id_marmita,)

                mycursor.execute(sql, valores)

                # Limpa acompanhamentos e guarnições antigos
                  mycursor.execute("DELETE FROM tb_marmita_acompanhamento WHERE
id_marmita = %s", (id_marmita,))
                    mycursor.execute("DELETE FROM tb_marmita_guarnicao WHERE
id_marmita = %s", (id_marmita,))

                # Adiciona novos acompanhamentos e guarnições
                for id_acompanhamento in acompanhamentos:
                        mycursor.execute("INSERT INTO tb_marmita_acompanhamento
(id_marmita, id_acompanhamento) VALUES (%s, %s)", (id_marmita,
id_acompanhamento))

                for id_guarnicao in guarnicoes:
                            mycursor.execute("INSERT INTO tb_marmita_guarnicao
(id_marmita, id_guarnicao) VALUES (%s, %s)", (id_marmita, id_guarnicao))

                mydb.commit()
                mydb.close()
```

```
        def obter_imagem_marmita(self, id_marmita):
            mydb = Conexao.conectar()
            mycursor = mydb.cursor()

            sql = "SELECT url_img FROM tb_marmita WHERE id_marmita = %s"
            mycursor.execute(sql, (id_marmita,))
            resultado = mycursor.fetchone()

            mydb.close()

            if resultado:
                return resultado[0]  # Retorna o caminho da imagem
            return None  # Retorna None se não encontrar
```

```
    from conexao import Conexao
    from hashlib import sha256
    import base64

    class Carrinho:
        def __init__(self):
            # Inicializa a classe Sistema sem variáveis de instância
necessárias.
            self.tel = None
            self.id_produto = None


        def inserir_item_carrinho(self, cod_produto, id_marmita, id_cliente,
guarnicoes_selecionadas=None, acompanhamentos_selecionados=None):
            mydb = Conexao.conectar()
            mycursor = mydb.cursor()

            try:
                if id_marmita:
                    # Verifica se já existe uma marmita com as mesmas
guarnições e acompanhamentos no carrinho
                    sql_verificar_marmita = """
                        SELECT c.id_carrinho
                        FROM tb_carrinho c
```

```
                            LEFT JOIN tb_carrinho_guarnicao cg ON c.id_carrinho
= cg.id_carrinho
                                    LEFT  JOIN  tb_carrinho_acompanhamento  ca  ON
c.id_carrinho = ca.id_carrinho
                            WHERE c.id_cliente = %s AND c.id_marmita = %s
                            GROUP BY c.id_carrinho
                            HAVING
                                    (COALESCE(GROUP_CONCAT(DISTINCT cg.guarnicao
ORDER BY cg.guarnicao), '') = %s) AND
                                                (COALESCE(GROUP_CONCAT(DISTINCT
ca.acompanhamento ORDER BY ca.acompanhamento), '') = %s)
                        """

                        # Concatena as guarnições e acompanhamentos selecionados
para comparação
                                                                guarnicoes_str     =
','.join(sorted(guarnicoes_selecionadas)) if guarnicoes_selecionadas else ''
                                                            acompanhamentos_str     =
','.join(sorted(acompanhamentos_selecionados)) if acompanhamentos_selecionados
else ''

                            # Executa a consulta para verificar se existe uma
marmita igual no carrinho
                        mycursor.execute(sql_verificar_marmita, (
                                        id_cliente,  id_marmita,  guarnicoes_str,
acompanhamentos_str
                        ))
                        resultado = mycursor.fetchone()

                        if resultado:
                            # Marmita com as mesmas guarnições e acompanhamentos
já existe, atualiza a quantidade
                            id_carrinho = resultado[0]
                            sql_atualizar_quantidade = """
                                UPDATE tb_carrinho
                                SET quantidade = quantidade + 1
                                WHERE id_carrinho = %s
                            """
                                        mycursor.execute(sql_atualizar_quantidade,
(id_carrinho,))
                        else:
                                    # Se não existir marmita com as mesmas
características, insere uma nova
                            sql_inserir_carrinho = """
                                        INSERT  INTO  tb_carrinho  (id_cliente,
cod_produto, id_marmita, quantidade)
                                VALUES (%s, %s, %s, 1)
                            """
```

```
                            mycursor.execute(sql_inserir_carrinho, (id_cliente,
cod_produto, id_marmita))

                        # Obtém o ID do carrinho recém-inserido
                        id_carrinho = mycursor.lastrowid

                        # Inserir guarnições na tabela tb_carrinho_guarnicao
                        if guarnicoes_selecionadas:
                            sql_inserir_guarnicao = """
                                        INSERT  INTO  tb_carrinho_guarnicao
(id_carrinho, guarnicao)
                                    VALUES (%s, %s)
                            """
                            for guarnicao in guarnicoes_selecionadas:
                                    mycursor.execute(sql_inserir_guarnicao,
(id_carrinho, guarnicao))

                                    #  Inserir  acompanhamentos  na  tabela
tb_carrinho_acompanhamento
                        if acompanhamentos_selecionados:
                            sql_inserir_acompanhamento = """
                                     INSERT INTO tb_carrinho_acompanhamento
(id_carrinho, acompanhamento)
                                    VALUES (%s, %s)
                            """
                                                for   acompanhamento   in
acompanhamentos_selecionados:
                                    mycursor.execute(sql_inserir_acompanhamento,
(id_carrinho, acompanhamento))

                else:
                      # Para produtos, verifica se já existe no carrinho e
atualiza a quantidade
                    sql_verificar_produto = """
                        SELECT quantidade FROM tb_carrinho
                        WHERE id_cliente = %s AND cod_produto = %s
                    """
                        mycursor.execute(sql_verificar_produto, (id_cliente,
cod_produto))
                    resultado = mycursor.fetchone()

                    if resultado:
                        nova_quantidade = resultado[0] + 1
                        sql_update_produto = """
                            UPDATE tb_carrinho
                            SET quantidade = %s
                            WHERE id_cliente = %s AND cod_produto = %s
                        """
```

```
                                        mycursor.execute(sql_update_produto,
(nova_quantidade, id_cliente, cod_produto))
                    else:
                          # Se o produto não existir no carrinho, insere um
novo
                        sql_inserir_produto = """
                                    INSERT  INTO  tb_carrinho  (id_cliente,
cod_produto, quantidade)
                            VALUES (%s, %s, 1)
                        """
                          mycursor.execute(sql_inserir_produto, (id_cliente,
cod_produto))

                mydb.commit()

            except Exception as e:
                mydb.rollback()
                print(f"Erro ao inserir item no carrinho: {e}")
            finally:
                mydb.close()









        def exibir_carrinho(self, id_cliente):
            mydb = Conexao.conectar()
            mycursor = mydb.cursor()

            # Consulta SQL para obter produtos no carrinho
            sql_produtos = """
                  SELECT  p.cod_produto,  p.nome_produto,  p.preco,  p.url_img,
c.id_carrinho, c.quantidade
                FROM tb_carrinho AS c
                JOIN tb_produto AS p ON c.cod_produto = p.cod_produto
                WHERE c.id_cliente = %s;
            """
            mycursor.execute(sql_produtos, (id_cliente,))
            resultado_produtos = mycursor.fetchall()

            # Consulta SQL para obter marmitas no carrinho
            sql_marmitas = """
```

```
                    SELECT m.id_marmita, m.nome_marmita, m.preco, m.url_img,
c.id_carrinho, c.quantidade, m.imagem_binaria
                    FROM tb_carrinho AS c
                    JOIN tb_marmita AS m ON c.id_marmita = m.id_marmita
                    WHERE c.id_cliente = %s;
                """
                mycursor.execute(sql_marmitas, (id_cliente,))
                resultado_marmitas = mycursor.fetchall()

                lista_carrinho = {
                    'produtos': [],
                    'marmitas': []
                }
                total_preco = 0  # Inicializa o total de preço

                # Itera sobre os resultados de produtos e adiciona ao carrinho
                for resultado in resultado_produtos:
                    preco_produto = resultado[2]
                    quantidade_produto = resultado[5]

                    lista_carrinho['produtos'].append({
                        'nome_produto': resultado[1],
                        'preco': preco_produto,
                        'imagem_produto': resultado[3],
                        'id_carrinho': resultado[4],
                        'quantidade': quantidade_produto
                    })

                        total_preco += preco_produto * quantidade_produto  #
Atualiza o total de preço

                # Itera sobre os resultados de marmitas e adiciona ao carrinho,
buscando guarnições e acompanhamentos
                for resultado in resultado_marmitas:
                    preco_marmita = resultado[2]
                    quantidade_marmita = resultado[5]

                    # Verifica se imagem_binaria existe e converte para Base64,
senão utiliza url_img
                    if resultado[6]:  # Se imagem_binaria existir
                                                            imagem_marmita     =
f"{base64.b64encode(resultado[6]).decode('utf-8')}"
                    else:  # Caso contrário, usa a URL da imagem
                        imagem_marmita = resultado[3]

                    # Consulta para obter guarnições da marmita no carrinho
                    sql_guarnicoes = """
                        SELECT g.nome_guarnicao
                        FROM tb_carrinho_guarnicao AS cg
```

```
                    JOIN tb_guarnicao AS g ON cg.guarnicao = g.id_guarnicao
                    WHERE cg.id_carrinho = %s
                """
                mycursor.execute(sql_guarnicoes, (resultado[4],))
                guarnicoes = [row[0] for row in mycursor.fetchall()]

                # Consulta para obter acompanhamentos da marmita no carrinho
                sql_acompanhamentos = """
                    SELECT a.nome_acompanhamento
                    FROM tb_carrinho_acompanhamento AS ca
                        JOIN tb_acompanhamentos AS a ON ca.acompanhamento =
a.id_acompanhamento
                    WHERE ca.id_carrinho = %s
                """
                mycursor.execute(sql_acompanhamentos, (resultado[4],))
                acompanhamentos = [row[0] for row in mycursor.fetchall()]

                 # Adiciona as guarnições e acompanhamentos ao dicionário da
marmita
                lista_carrinho['marmitas'].append({
                    'nome_marmita': resultado[1],
                    'preco': preco_marmita,
                    'imagem_produto': imagem_marmita,
                    'id_carrinho': resultado[4],
                    'quantidade': quantidade_marmita,
                    'guarnicoes': guarnicoes,
                    'acompanhamentos': acompanhamentos
                })

                    total_preco += preco_marmita * quantidade_marmita  #
Atualiza o total de preço

            mydb.close()
             total_preco_formatado = "{:.2f}".format(total_preco)  # Formata
o total de preço
            return {
                'produtos': lista_carrinho['produtos'],
                'marmitas': lista_carrinho['marmitas'],
                'total_preco': total_preco_formatado
            }
```

```
        # Método para excluir um produto do carrinho
        def remover_produto_carrinho(self, id_carrinho):
            mydb = Conexao.conectar()
            mycursor = mydb.cursor()

            try:
                # Remove guarnições relacionadas
                 sql_remover_guarnicoes = "DELETE FROM tb_carrinho_guarnicao
WHERE id_carrinho = %s"
                mycursor.execute(sql_remover_guarnicoes, (id_carrinho,))

                # Remove acompanhamentos relacionados
                            sql_remover_acompanhamentos   =   "DELETE   FROM
tb_carrinho_acompanhamento WHERE id_carrinho = %s"
                                mycursor.execute(sql_remover_acompanhamentos,
(id_carrinho,))

                # Agora remove o produto do carrinho
                    sql_remover_carrinho  =  "DELETE  FROM  tb_carrinho  WHERE
id_carrinho = %s"
                mycursor.execute(sql_remover_carrinho, (id_carrinho,))

                mydb.commit()
                print(f"Produto com ID {id_carrinho} removido com sucesso do
carrinho.")
            except Exception as e:
                mydb.rollback()
                print(f"Erro ao remover o produto do carrinho: {e}")
            finally:
                mydb.close()



         # Método para atualizar a quantidade de um produto específico no
carrinho
            def   atualizar_quantidade_produto_carrinho(self,   id_carrinho,
quantidade):
            mydb = Conexao.conectar()
            mycursor = mydb.cursor()

              # Consulta SQL para atualizar a quantidade do produto no
carrinho
             sql = "UPDATE tb_carrinho SET quantidade = %s WHERE id_carrinho
= %s"
            mycursor.execute(sql, (quantidade, id_carrinho))
```

```
                mydb.commit()
                mydb.close()

            def  enviar_carrinho(self,  id_cliente,  itens,  hora_atual):   #
Adiciona hora_atual como argumento
                try:
                    # Conexão ao banco de dados
                    mydb = Conexao.conectar()
                    mycursor = mydb.cursor()

                    # 1. Verifica se o carrinho tem itens
                     sql_carrinho = "SELECT cod_produto, id_marmita, quantidade
FROM tb_carrinho WHERE id_cliente = %s"
                    mycursor.execute(sql_carrinho, (id_cliente,))
                    itens_carrinho = mycursor.fetchall()

                    if not itens_carrinho:
                           print("Carrinho está vazio, não é possível enviar o
pedido.")
                           return False  # Retorna falso se o carrinho estiver
vazio

                    # 2. Insere um novo pedido na tabela `tb_pedidos`
                          sql_pedido  =  "INSERT  INTO  tb_pedidos  (id_cliente,
data_pedido, hora_pedido, status) VALUES (%s, CURDATE(), %s, 'Pendente')"  #
Adiciona hora_pedido
                      mycursor.execute(sql_pedido, (id_cliente, hora_atual))  #
Passa a hora_atual
                    id_pedido = mycursor.lastrowid  # Obtém o ID do novo pedido

                     # 3. Insere os produtos e marmitas do carrinho na tabela
`tb_produtos_pedidos`
                    for item in itens_carrinho:
                        cod_produto = item[0]
                        id_marmita = item[1]
                        quantidade = item[2]

                        if cod_produto:  # Se for um produto
                                           sql_produtos_pedido   =   "INSERT   INTO
tb_produtos_pedidos (id_pedido, cod_produto, quantidade) VALUES (%s, %s, %s)"
                                mycursor.execute(sql_produtos_pedido, (id_pedido,
cod_produto, quantidade))
                        elif id_marmita:  # Se for uma marmita
                                           sql_marmitas_pedido   =   "INSERT   INTO
tb_produtos_pedidos (id_pedido, id_marmita, quantidade) VALUES (%s, %s, %s)"
                                mycursor.execute(sql_marmitas_pedido, (id_pedido,
id_marmita, quantidade))

                           # 4. Insere guarnições e acompanhamentos da marmita
```

```
                                    sql_guarnicoes = "SELECT guarnicao FROM
tb_carrinho_guarnicao WHERE id_carrinho = (SELECT id_carrinho FROM tb_carrinho
WHERE id_cliente = %s AND id_marmita = %s)"
                                mycursor.execute(sql_guarnicoes, (id_cliente,
id_marmita))
                            guarnicoes = mycursor.fetchall()
                            for guarnicao in guarnicoes:
                                    sql_inserir_guarnicao = "INSERT INTO
tb_guarnicoes_pedidos (id_pedido, guarnicao) VALUES (%s, %s)"
                                    mycursor.execute(sql_inserir_guarnicao,
(id_pedido, guarnicao[0]))

                                sql_acompanhamentos = "SELECT acompanhamento FROM
tb_carrinho_acompanhamento WHERE id_carrinho = (SELECT id_carrinho FROM
tb_carrinho WHERE id_cliente = %s AND id_marmita = %s)"
                                mycursor.execute(sql_acompanhamentos, (id_cliente,
id_marmita))
                            acompanhamentos = mycursor.fetchall()
                            for acompanhamento in acompanhamentos:
                                    sql_inserir_acompanhamento = "INSERT INTO
tb_acompanhamentos_pedidos (id_pedido, acompanhamento) VALUES (%s, %s)"
                                mycursor.execute(sql_inserir_acompanhamento,
(id_pedido, acompanhamento[0]))

                    # 5. Remove os itens do carrinho após finalizar o pedido
                    self.remover_produto_carrinho(id_cliente)

                    # Confirma as operações
                    mydb.commit()
                        print(f"Pedido {id_pedido} enviado com sucesso para o
cliente {id_cliente}.")
                    return True

                except Exception as e:
                    # Se ocorrer algum erro, desfaz todas as alterações
                    mydb.rollback()
                    print(f"Erro ao enviar o pedido: {e}")
                    return False

                finally:
                    # Fecha a conexão com o banco de dados
                    mydb.close()


            def remover_todo_carrinho(self, id_cliente):
                mydb = Conexao.conectar()
                mycursor = mydb.cursor()

                try:
```

```
                # Obter todos os ids do carrinho relacionados ao cliente
                    sql_obter_ids_carrinho = "SELECT id_carrinho FROM
tb_carrinho WHERE id_cliente = %s"
                mycursor.execute(sql_obter_ids_carrinho, (id_cliente,))
                ids_carrinho = mycursor.fetchall()

                if not ids_carrinho:
                    print(f"Carrinho do cliente {id_cliente} está vazio.")
                    return

                 ids_carrinho_list = [str(id[0]) for id in ids_carrinho]  #
Converte os ids para uma lista

                # Remove guarnições relacionadas ao carrinho
                sql_remover_guarnicoes = f"DELETE FROM tb_carrinho_guarnicao
WHERE id_carrinho IN ({','.join(ids_carrinho_list)})"
                mycursor.execute(sql_remover_guarnicoes)

                # Remove acompanhamentos relacionados ao carrinho
                            sql_remover_acompanhamentos = f"DELETE FROM
tb_carrinho_acompanhamento WHERE id_carrinho IN
({','.join(ids_carrinho_list)})"
                mycursor.execute(sql_remover_acompanhamentos)

                # Remove todos os itens do carrinho
                    sql_remover_carrinho = f"DELETE FROM tb_carrinho WHERE
id_cliente = %s"
                mycursor.execute(sql_remover_carrinho, (id_cliente,))

                # Confirma a remoção
                mydb.commit()
                    print(f"Todos os itens do cliente {id_cliente} foram
removidos do carrinho com sucesso.")

            except Exception as e:
                # Em caso de erro, desfaz as alterações
                mydb.rollback()
                print(f"Erro ao remover os produtos do carrinho: {e}")

            finally:
                # Fecha a conexão com o banco de dados
                mydb.close()


    import mysql.connector

    class Conexao:

        def conectar():
```

```
            mydb = mysql.connector.connect(
                user="cantina",
                password="988430466Tel",
                host="cantina-virtual.mysql.database.azure.com",
                database="bd_cantinadalu"
            )

            return mydb
```

```
    from conexao import Conexao
    from hashlib import sha256

    class Perfil:
        def __init__(self):
            # Inicializa a classe Sistema sem variáveis de instância
necessárias.
            self.tel = None
            self.id_produto = None


        def verificar_senha(self, id_cliente, senha_fornecida):
            mydb = Conexao.conectar()  # Conecta ao banco de dados
            mycursor = mydb.cursor()

            try:
                                                    senha_fornecida       =
sha256(senha_fornecida.encode()).hexdigest()
                # Busca a senha atual do cliente no banco de dados
                     mycursor.execute("SELECT  senha  FROM  tb_cliente  WHERE
id_cliente = %s", (id_cliente,))
                senha_armazenada = mycursor.fetchone()

                if senha_armazenada is None:
                    print("Cliente não encontrado.")  # Log para depuração
                    return False  # Se o cliente não for encontrado

                # Verifica se a senha fornecida é igual à senha armazenada
                  return  senha_armazenada[0]  ==  senha_fornecida  # Retorna
True ou False

            except Exception as e:
                print(f"Erro ao verificar a senha: {str(e)}")  # Log do erro
                return {"error": f"Erro ao verificar a senha: {str(e)}"}

            finally:
                mycursor.close()  # Fecha o cursor
                mydb.close()  # Fecha a conexão com o banco de dados
```

```
                def     atualizar_perfil(self,     id_cliente,     nome=None,
caminho_imagem=None):
            mydb = Conexao.conectar()
            mycursor = mydb.cursor()

            try:
                # Inicia a query de atualização
                sql = "UPDATE tb_cliente SET"
                valores = []

                # Atualiza o nome apenas se ele for fornecido (não vazio)
                if nome:
                    sql += " nome_comp = %s"
                    valores.append(nome)

                # Se uma nova imagem foi enviada
                if caminho_imagem:
                    if nome:
                         sql += ","  # Adiciona uma vírgula se o nome já foi
incluído
                    sql += " imagem_binaria = %s"
                    with open(caminho_imagem, 'rb') as imagem:
                        dados_imagem = imagem.read()
                    valores.append(dados_imagem)

                # Se nenhuma alteração foi feita
                if not valores:
                    return {"message": "Nenhuma alteração feita no perfil."}

                # Adiciona a condição para o WHERE
                sql += " WHERE id_cliente = %s"
                valores.append(id_cliente)

                # Executa a query de atualização
                 print(f"Executando SQL: {sql}, com valores: {valores}")  #
Log para depuração
                mycursor.execute(sql, valores)
                mydb.commit()

                return {"message": "Perfil atualizado com sucesso!"}

            except Exception as e:
                mydb.rollback()
                  print(f"Erro ao atualizar o perfil: {str(e)}")  # Log do
erro
                return {"error": f"Erro ao atualizar o perfil: {str(e)}"}

            finally:
```

```
                mycursor.close()
                mydb.close()




        def obter_perfil(self, id_cliente):
            mydb = Conexao.conectar()
            mycursor = mydb.cursor()

            try:
                 print(f"Buscando perfil para id_cliente: {id_cliente}")  # Log do ID do cliente
                  sql = "SELECT nome_comp, imagem_binaria FROM tb_cliente WHERE id_cliente = %s"
                mycursor.execute(sql, (id_cliente,))
                perfil = mycursor.fetchone()

                if perfil:
                    nome, imagem = perfil
                      return {'nome': nome, 'imagem': imagem}  # Retorna a imagem binária ou None
                return None  # Retorna None se não houver perfil
            except Exception as e:
                print(f"Erro ao obter perfil: {e}")  # Log de erro
                return None  # Lida com qualquer erro
            finally:
                mycursor.close()
                mydb.close()




        def obter_imagem_perfil(self, id_cliente):
            mydb = Conexao.conectar()
            mycursor = mydb.cursor()

            try:
                  mycursor.execute("SELECT imagem_binaria FROM tb_cliente WHERE id_cliente = %s", (id_cliente,))
                imagem = mycursor.fetchone()
                return imagem[0] if imagem else None
            except Exception as e:
                return None
            finally:
                mycursor.close()
                mydb.close()
```

```
      from conexao import Conexao
      from datetime import datetime


      class Relatorio:
          def __init__(self):
                # Inicializa a classe Sistema sem variáveis de instância
necessárias.
              self.tel = None
              self.id_produto = None



          def exibir_relatorio_entregue(self, data_inicial, data_final):
              mydb = Conexao.conectar()
              mycursor = mydb.cursor()

              sql = """
                  SELECT
                      p.id_pedido,
                      p.data_pedido,
                      p.hora_pedido,
                      p.status,
                      SUM(pp.quantidade * CASE
                          WHEN pp.cod_produto IS NOT NULL THEN pr.preco
                          WHEN pp.id_marmita IS NOT NULL THEN m.preco
                          ELSE 0
                      END) AS valor_total
                  FROM tb_pedidos p
                        LEFT JOIN tb_produtos_pedidos pp ON p.id_pedido =
pp.id_pedido
                  LEFT JOIN tb_produto pr ON pp.cod_produto = pr.cod_produto
                  LEFT JOIN tb_marmita m ON pp.id_marmita = m.id_marmita
                     WHERE p.data_pedido BETWEEN %s AND %s AND p.status =
'entregue'
                  GROUP BY p.id_pedido
              """

              # Subconsultas para valores gerais e quantidade
              sql_total = """
                  SELECT
                      SUM(pp.quantidade * CASE
                          WHEN pp.cod_produto IS NOT NULL THEN pr.preco
                          WHEN pp.id_marmita IS NOT NULL THEN m.preco
                          ELSE 0
                      END) AS valor_total_geral,
                           COUNT(DISTINCT p.id_pedido) AS total_pedidos   --
Contagem distinta dos pedidos
                  FROM tb_pedidos p
```

```
                        LEFT JOIN tb_produtos_pedidos pp ON p.id_pedido =
pp.id_pedido
                LEFT JOIN tb_produto pr ON pp.cod_produto = pr.cod_produto
                LEFT JOIN tb_marmita m ON pp.id_marmita = m.id_marmita
                    WHERE p.data_pedido BETWEEN %s AND %s AND p.status =
'entregue'
            """


            # Executa a consulta principal para os pedidos
            mycursor.execute(sql, (data_inicial, data_final))
            pedidos_resultados = mycursor.fetchall()

            # Executa a subconsulta para valores gerais
            mycursor.execute(sql_total, (data_inicial, data_final))
            total_resultado = mycursor.fetchone()

            mycursor.close()
            mydb.close()

            # Formatação dos dados para o template
            pedidos_formatados = []
            for pedido in pedidos_resultados:
                    data_pedido = datetime.strptime(str(pedido[1]), "%Y-%m-
%d").strftime("%d/%m/%Y")
                        pedidos_formatados.append((pedido[0], data_pedido,
*pedido[2:]))

              valor_total_geral = total_resultado[0] if total_resultado[0]
else 0
            total_pedidos = total_resultado[1] if total_resultado[1] else 0

            return pedidos_formatados, valor_total_geral, total_pedidos




        def exibir_relatorio_cancelado(self, data_inicial, data_final):
            mydb = Conexao.conectar()
            mycursor = mydb.cursor()

            sql = """
                SELECT
                    p.id_pedido,
                    p.data_pedido,
                    p.hora_pedido,
                    p.status,
                    p.motivo_cancelamento,
```

```
                    SUM(pp.quantidade * CASE
                        WHEN pp.cod_produto IS NOT NULL THEN pr.preco
                        WHEN pp.id_marmita IS NOT NULL THEN m.preco
                        ELSE 0
                    END) AS valor_total
                FROM tb_pedidos p
                     LEFT JOIN tb_produtos_pedidos pp ON p.id_pedido =
pp.id_pedido
                LEFT JOIN tb_produto pr ON pp.cod_produto = pr.cod_produto
                LEFT JOIN tb_marmita m ON pp.id_marmita = m.id_marmita
                   WHERE p.data_pedido BETWEEN %s AND %s AND p.status =
'cancelado'
                GROUP BY p.id_pedido
            """

            # Subconsulta para obter o valor total e a quantidade de pedidos
cancelados
            sql_total_cancelado = """
                SELECT
                    SUM(pp.quantidade * CASE
                        WHEN pp.cod_produto IS NOT NULL THEN pr.preco
                        WHEN pp.id_marmita IS NOT NULL THEN m.preco
                        ELSE 0
                    END) AS valor_total_cancelado,
                       COUNT(DISTINCT p.id_pedido) AS total_cancelados  --
Contagem distinta dos pedidos
                FROM tb_pedidos p
                     LEFT JOIN tb_produtos_pedidos pp ON p.id_pedido =
pp.id_pedido
                LEFT JOIN tb_produto pr ON pp.cod_produto = pr.cod_produto
                LEFT JOIN tb_marmita m ON pp.id_marmita = m.id_marmita
                   WHERE p.data_pedido BETWEEN %s AND %s AND p.status =
'cancelado'
            """


            # Executa a consulta principal para os pedidos cancelados
            mycursor.execute(sql, (data_inicial, data_final))
            cancelados_resultados = mycursor.fetchall()

               # Executa a subconsulta para valores totais dos pedidos
cancelados
                     mycursor.execute(sql_total_cancelado,   (data_inicial,
data_final))
            total_cancelado_resultado = mycursor.fetchone()

            mycursor.close()
            mydb.close()
```

```
            # Formatação dos dados para o template
            cancelados_formatados = []
            for pedido in cancelados_resultados:
                    data_pedido = datetime.strptime(str(pedido[1]), "%Y-%m-
%d").strftime("%d/%m/%Y")
                    cancelados_formatados.append((pedido[0], data_pedido,
*pedido[2:]))

                valor_total_cancelado = total_cancelado_resultado[0] if
total_cancelado_resultado[0] else 0
                    total_cancelados = total_cancelado_resultado[1] if
total_cancelado_resultado[1] else 0

                    return cancelados_formatados, valor_total_cancelado,
total_cancelados



    from conexao import Conexao
    from hashlib import sha256
    import base64

    class Sistema:
        def __init__(self):
            # Inicializa a classe Sistema sem variáveis de instância
necessárias.
            self.tel = None
            self.id_produto = None




        import base64

        def exibir_produtos_adm(self):
            mydb = Conexao.conectar()  # Conecta ao banco de dados
            mycursor = mydb.cursor()    # Cria um cursor para executar
queries

            # Consulta SQL para selecionar todos os produtos
            sql = "SELECT * FROM tb_produto"
            mycursor.execute(sql)      # Executa a consulta
            resultado = mycursor.fetchall()  # Obtém todos os resultados

            lista_produtos = []

            # Itera sobre os resultados e adiciona cada produto à lista
            for produto in resultado:
                imagem_blob = produto[7]  # Blob da imagem (posição 7)
```

```
                imagem_url = produto[3]   # URL da imagem (posição 3)

                if imagem_blob:  # Caso o blob esteja presente
                    # Converte o blob para uma string Base64
                                                        imagem_base64      =
base64.b64encode(imagem_blob).decode('utf-8')
                    # Cria o URL de dados para a imagem
                                                      imagem_produto      =
f"{imagem_base64}"
                elif imagem_url:  # Caso o blob não exista, mas o URL esteja
presente
                    imagem_produto = imagem_url  # Usa o URL diretamente
                else:  # Caso não exista nem blob nem URL
                    imagem_produto = None

                lista_produtos.append({
                    'nome_produto': produto[1],
                    'preco': produto[2],
                    'imagem_produto': imagem_produto,  # Base64 ou URL
                    'categoria': produto[5],
                    'descricao': produto[4],
                    'id_produto': produto[0],
                    'habilitado': produto[6]
                })

            mydb.close()  # Fecha a conexão com o banco de dados
              return lista_produtos if lista_produtos else []  # Retorna a
lista de produtos ou uma lista vazia se nenhum produto for encontrado



        def exibir_produtos(self):
            mydb = Conexao.conectar()  # Conecta ao banco de dados
              mycursor = mydb.cursor()    # Cria um cursor para executar
queries

            # Consulta SQL com JOIN para obter o nome e o ID da categoria
            sql = """
                SELECT p.cod_produto, p.nome_produto, p.preco, p.url_img,
p.descricao, c.id_categoria, c.nome, p.imagem_blob
            FROM tb_produto p
            JOIN tb_categoria c ON p.id_categoria = c.id_categoria
            WHERE p.habilitado = 1
            """
            mycursor.execute(sql)      # Executa a consulta
            resultado = mycursor.fetchall()  # Obtém todos os resultados

            produtos_por_categoria = {}
```

```
            # Itera sobre os resultados e agrupa os produtos por categoria
            for produto in resultado:
                categoria_id = produto[5]
                categoria_nome = produto[6]
                url_img = produto[3]  # URL da imagem
                blob_imagem = produto[7]  # Blob da imagem

                # Lógica para determinar a imagem do produto
                if url_img:  # Se a URL estiver disponível
                    imagem_produto = url_img
                elif blob_imagem:  # Se o blob estiver disponível
                                                        imagem_base64      =
base64.b64encode(blob_imagem).decode('utf-8')
                                                       imagem_produto      =
f"{imagem_base64}"
                else:  # Se nenhuma imagem estiver disponível
                    imagem_produto = None

                # Agrupa os produtos por categoria
                if categoria_id not in produtos_por_categoria:
                    produtos_por_categoria[categoria_id] = {
                        'nome_categoria': categoria_nome,  # Armazena o nome
da categoria
                        'produtos': []
                    }

                produtos_por_categoria[categoria_id]['produtos'].append({
                    'id_produto': produto[0],
                    'nome_produto': produto[1],
                    'preco': produto[2],
                    'imagem_produto': imagem_produto,  # URL ou Base64
                    'descricao': produto[4]
                })

            mydb.close()  # Fecha a conexão com o banco de dados
             return produtos_por_categoria if produtos_por_categoria else {}
 # Retorna os produtos agrupados ou um dicionário vazio




        # Método para exibir um único produto com base no ID
        def exibir_produto(self, id):
            mydb = Conexao.conectar()
            mycursor = mydb.cursor()

            # Consulta SQL para selecionar um produto específico pelo ID
            sql = "SELECT * FROM tb_produto WHERE cod_produto = %s"
```

```
            mycursor.execute(sql, (id,))
            resultado = mycursor.fetchone()  # Obtém o resultado único

            if not resultado:
                    return None  # Retorna None caso o produto não seja
encontrado

            # Recupera as informações do produto
            imagem_url = resultado[3]  # URL da imagem (posição 3)
            imagem_blob = resultado[7]  # Blob da imagem (posição 7)

            # Lógica para definir a imagem do produto
            if imagem_url:  # Se o URL da imagem estiver disponível
                imagem_produto = imagem_url
            elif imagem_blob:  # Se o blob estiver disponível
                  imagem_base64 = base64.b64encode(imagem_blob).decode('utf-
8')
                imagem_produto = f"{imagem_base64}"
            else:  # Se nenhum estiver disponível
                imagem_produto = None

            # Cria um dicionário para o produto
            dicionario_produto = {
                'nome_produto': resultado[1],
                'preco': resultado[2],
                'imagem_produto': imagem_produto,  # Base64 ou URL
                'descricao': resultado[4],
                'cod_produto': resultado[0]
            }

            mydb.commit()
            mydb.close()
              return [dicionario_produto]  # Retorna a lista com um único
produto



        def exibir_marmita(self, id_marmita):
            # Conexão e consulta ao banco
            mydb = Conexao.conectar()
            mycursor = mydb.cursor()

                # Consulta SQL para selecionar a marmita, guarnições e
acompanhamentos associados
            sql_marmita = """
            SELECT
                        m.id_marmita, m.nome_marmita, m.preco, m.tamanho,
m.descricao, m.url_img,
```

```
                        GROUP_CONCAT(DISTINCT  CONCAT(g.id_guarnicao,  ':',
g.nome_guarnicao) SEPARATOR ', ') AS guarnicoes,
                    GROUP_CONCAT(DISTINCT  CONCAT(a.id_acompanhamento,  ':',
a.nome_acompanhamento) SEPARATOR ', ') AS acompanhamentos,
                     m.imagem_binaria
                FROM
                    tb_marmita AS m
                    LEFT  JOIN  tb_marmita_guarnicao  AS  mg  ON  m.id_marmita  =
mg.id_marmita
                LEFT JOIN tb_guarnicao AS g ON mg.id_guarnicao = g.id_guarnicao
                  LEFT  JOIN  tb_marmita_acompanhamento  AS  ma  ON  m.id_marmita  =
ma.id_marmita
                  LEFT  JOIN  tb_acompanhamentos  AS  a  ON  ma.id_acompanhamento  =
a.id_acompanhamento
                WHERE m.id_marmita = %s
                GROUP BY m.id_marmita;
                """
                mycursor.execute(sql_marmita, (id_marmita,))
                resultado = mycursor.fetchone()

                # Caso a consulta não encontre resultados
                if not resultado:
                    return None

                 # Função auxiliar para dividir os resultados de guarnições e
acompanhamentos
                def processar_itens(itens):
                    if not itens:
                        return []
                    lista_itens = []
                    for item in itens.split(', '):
                        id_item, nome_item = item.split(':')
                        lista_itens.append({'id': id_item, 'nome': nome_item})
                    return lista_itens

                # Processa guarnições e acompanhamentos associados à marmita
                 guarnicoes_associadas = processar_itens(resultado[6])  # IDs e
nomes das guarnições
                  acompanhamentos_associados = processar_itens(resultado[7])  #
IDs e nomes dos acompanhamentos

                # Consulta SQL para pegar todas as guarnições
                sql_todas_guarnicoes = "SELECT id_guarnicao, nome_guarnicao FROM
tb_guarnicao"
                mycursor.execute(sql_todas_guarnicoes)
                 todas_guarnicoes = [{'id': str(row[0]), 'nome': row[1]} for row
in mycursor.fetchall()]

                # Consulta SQL para pegar todos os acompanhamentos
```

```
                    sql_todos_acompanhamentos = "SELECT id_acompanhamento,
nome_acompanhamento FROM tb_acompanhamentos"
                mycursor.execute(sql_todos_acompanhamentos)
                todos_acompanhamentos = [{'id': str(row[0]), 'nome': row[1]} for
row in mycursor.fetchall()]

                # Função para converter imagem binária em base64
                def converter_imagem_binaria(img_binaria):
                      imagem_base64 = base64.b64encode(img_binaria).decode('utf-
8')
                    imagem_marmita = f"{imagem_base64}"
                    return imagem_marmita
                # Define a imagem da marmita (binária ou URL)
                imagem_marmita = None

                if resultado[8]:  # Verifica se há conteúdo binário
                    try:
                        # Tenta converter a imagem binária para base64
                        imagem_marmita = converter_imagem_binaria(resultado[8])
                    except Exception as e:
                        print(f"Erro ao converter imagem binária: {e}")
                         imagem_marmita = resultado[5]  # Fallback para a URL da
imagem
                else:
                      imagem_marmita = resultado[5]  # Se não houver binário,
utiliza a URL


                # Organiza os dados da marmita
                dados_marmita = {
                    'id_marmita': resultado[0],  # ID da marmita
                    'nome_marmita': resultado[1],  # Nome da marmita
                    'preco': resultado[2],  # Preço
                    'tamanho': resultado[3],  # Tamanho
                    'descricao': resultado[4],  # Descrição
                    'imagem_marmita': imagem_marmita,  # Imagem (binária ou URL)
                          'guarnicoes': guarnicoes_associadas,  # Guarnições
associadas à marmita
                           'acompanhamentos': acompanhamentos_associados,  #
Acompanhamentos associados à marmita
                    'todas_guarnicoes': todas_guarnicoes,  # Todas as guarnições
disponíveis
                     'todos_acompanhamentos': todos_acompanhamentos  # Todos os
acompanhamentos disponíveis
                }

                # Fecha a conexão
                mydb.close()
```

```
            # Retorna a lista com o dicionário da marmita
            return [dados_marmita]









        def exibir_marmitas(self):
            mydb = Conexao.conectar()  # Conecta ao banco de dados
              mycursor = mydb.cursor()    # Cria um cursor para executar
queries

            # Consulta SQL para selecionar marmitas habilitadas
            sql = """
                SELECT  m.id_marmita,  m.nome_marmita,  m.preco,  m.tamanho,
m.descricao,
                m.url_img, m.imagem_binaria, m.habilitado
            FROM tb_marmita m
            WHERE m.habilitado = 1
            """
            mycursor.execute(sql)  # Executa a consulta
            resultado = mycursor.fetchall()  # Obtém todos os resultados

            marmitas_por_tamanho = {}

            # Itera sobre os resultados e agrupa as marmitas por tamanho
            for marmita in resultado:
                tamanho = marmita[3]  # Obtém o tamanho da marmita (Pequena,
Média, Grande)
                url_img = marmita[5]  # URL da imagem
                blob_imagem = marmita[6]  # Blob da imagem

                # Lógica para determinar a imagem da marmita
                if url_img:  # Se a URL estiver disponível
                    imagem_marmita = url_img
                elif blob_imagem:  # Se o blob estiver disponível
                                                    imagem_base64      =
base64.b64encode(blob_imagem).decode('utf-8')
                                                   imagem_marmita      =
f"{imagem_base64}"
                else:  # Se nenhuma imagem estiver disponível
                    imagem_marmita = None
```

```
                # Agrupa as marmitas por tamanho
                if tamanho not in marmitas_por_tamanho:
                    marmitas_por_tamanho[tamanho] = {
                        'tamanho': tamanho,  # Armazena o tamanho da marmita
                        'marmitas': []
                    }

                marmitas_por_tamanho[tamanho]['marmitas'].append({
                    'id_marmita': marmita[0],
                    'nome_marmita': marmita[1],
                    'preco': marmita[2],
                    'imagem_marmita': imagem_marmita,  # URL ou Base64
                    'descricao': marmita[4]
                })

            mydb.close()  # Fecha a conexão com o banco de dados
             return marmitas_por_tamanho if marmitas_por_tamanho else {}  #
Retorna as marmitas agrupadas ou um dicionário vazio




        def exibir_marmitas_adm(self):
            mydb = Conexao.conectar()  # Conecta ao banco de dados
              mycursor = mydb.cursor()    # Cria um cursor para executar
queries

            # Consulta SQL para selecionar todos os produtos
                sql = "SELECT id_marmita, nome_marmita, preco, tamanho,
descricao, url_img, imagem_binaria, habilitado FROM tb_marmita"
            mycursor.execute(sql)      # Executa a consulta
            resultado = mycursor.fetchall()  # Obtém todos os resultados

            lista_marmitas = []

            # Itera sobre os resultados e adiciona cada marmita à lista
            for produto in resultado:
                id_marmita = produto[0]
                nome_marmita = produto[1]
                preco = produto[2]
                tamanho = produto[3]
                descricao = produto[4]
                url_img = produto[5]  # URL da imagem
                blob_imagem = produto[6]  # Imagem em formato binário
                habilitado = produto[7]  # Estado habilitado ou não

                # Lógica para determinar a imagem da marmita
                if url_img:  # Se a URL estiver disponível
```

```
                    imagem_marmita = url_img
                elif blob_imagem:  # Se o blob estiver disponível
                    # Converte a imagem binária para Base64
                                                        imagem_base64      =
base64.b64encode(blob_imagem).decode('utf-8')
                                                      imagem_marmita      =
f"{imagem_base64}"
                else:  # Se nenhuma imagem estiver disponível
                    imagem_marmita = None

                # Adiciona os dados da marmita à lista
                lista_marmitas.append({
                    'id_marmita': id_marmita,
                    'nome_marmita': nome_marmita,
                    'preco': preco,
                    'tamanho': tamanho,
                    'descricao': descricao,
                    'imagem_marmita': imagem_marmita,
                    'habilitado': habilitado
                })

            mydb.close()  # Fecha a conexão com o banco de dados

              return lista_marmitas if lista_marmitas else []  # Retorna a
lista de produtos ou uma lista vazia se nenhum produto for encontrado




          # Método para exibir todos os pedidos de um cliente específico com
detalhes dos produtos e marmitas
          def exibir_historico(self, id_cliente):
              mydb = Conexao.conectar()
              mycursor = mydb.cursor()

              sql = """
                          SELECT  p.id_pedido,  cl.id_cliente,  cl.nome_comp,
cl.telefone,
                                pr.nome_produto,  pr.preco  AS  preco_produto,
pp.quantidade,
                    m.nome_marmita, m.preco AS preco_marmita,
                    p.data_pedido, p.status, p.hora_pedido
                FROM tb_pedidos p
                JOIN tb_cliente cl ON p.id_cliente = cl.id_cliente
                JOIN tb_produtos_pedidos pp ON p.id_pedido = pp.id_pedido
                LEFT JOIN tb_produto pr ON pp.cod_produto = pr.cod_produto
                LEFT JOIN tb_marmita m ON pp.id_marmita = m.id_marmita
```

```
                    WHERE cl.id_cliente = %s
                    ORDER BY p.data_pedido DESC
                """
                mycursor.execute(sql, (id_cliente,))
                resultados = mycursor.fetchall()

                pedidos = {}

                for resultado in resultados:
                    id_pedido = resultado[0]
                    nome_cliente = resultado[2]
                    telefone_cliente = resultado[3]
                    nome_produto = resultado[4]
                    preco_produto = resultado[5]
                    quantidade_produto = resultado[6]
                    nome_marmita = resultado[7]
                    preco_marmita = resultado[8]
                           data_pedido  =  resultado[9].strftime('%d/%m/%Y')  if
resultado[11] else None
                    status_pedido = resultado[10]
                     hora_pedido = str(resultado[11]) if resultado[11] else None
 # Converte hora_pedido para string

                    if id_cliente not in pedidos:
                        pedidos[id_cliente] = {
                            'nome_cliente': nome_cliente,
                            'telefone': telefone_cliente,
                            'pedidos': {}
                        }

                    if id_pedido not in pedidos[id_cliente]['pedidos']:
                        pedidos[id_cliente]['pedidos'][id_pedido] = {
                            'data_pedido': data_pedido,
                            'status': status_pedido,
                                 'hora': hora_pedido,  # Usa a hora_pedido já
formatada
                            'produtos': [],
                            'marmitas': [],
                            'guarnicoes': [],
                            'acompanhamentos': [],
                            'total_preco': 0
                        }

                    if nome_produto:
                        total_produto = preco_produto * quantidade_produto

pedidos[id_cliente]['pedidos'][id_pedido]['produtos'].append({
                            'nome_produto': nome_produto,
                            'preco': preco_produto,
```

```
                        'quantidade': quantidade_produto
                    })
                    pedidos[id_cliente]['pedidos'][id_pedido]['total_preco']
+= total_produto

                if nome_marmita:

pedidos[id_cliente]['pedidos'][id_pedido]['marmitas'].append({
                        'nome_marmita': nome_marmita,
                        'preco': preco_marmita,
                        'quantidade': quantidade_produto
                    })
                    pedidos[id_cliente]['pedidos'][id_pedido]['total_preco']
+= preco_marmita * quantidade_produto

            for id_pedido in pedidos[id_cliente]['pedidos']:
                sql_guarnicoes = """
                    SELECT g.nome_guarnicao
                    FROM tb_guarnicoes_pedidos AS cg
                    JOIN tb_guarnicao AS g ON cg.guarnicao = g.id_guarnicao
                    WHERE cg.id_pedido = %s
                """
                mycursor.execute(sql_guarnicoes, (id_pedido,))
                guarnicoes = [row[0] for row in mycursor.fetchall()]
                   pedidos[id_cliente]['pedidos'][id_pedido]['guarnicoes'] =
guarnicoes

                sql_acompanhamentos = """
                    SELECT a.nome_acompanhamento
                    FROM tb_acompanhamentos_pedidos AS ca
                       JOIN tb_acompanhamentos AS a ON ca.acompanhamento =
a.id_acompanhamento
                    WHERE ca.id_pedido = %s
                """
                mycursor.execute(sql_acompanhamentos, (id_pedido,))
                acompanhamentos = [row[0] for row in mycursor.fetchall()]
                pedidos[id_cliente]['pedidos'][id_pedido]['acompanhamentos']
= acompanhamentos

            mydb.close()
            return pedidos




        def atualizar_status_pedido(self, id_pedido, novo_status):
            mydb = Conexao.conectar()  # Conecta ao banco de dados
```

```
            mycursor = mydb.cursor()

            try:
                # Atualiza o status do pedido
                 sql = "UPDATE tb_pedidos SET status = %s WHERE id_pedido =
%s"
                mycursor.execute(sql, (novo_status, id_pedido))

                mydb.commit()  # Confirma a alteração
                   return {"message": "Status atualizado com sucesso!"}  #
Retorna mensagem de sucesso
            except Exception as e:
                mydb.rollback()  # Reverte a transação em caso de erro
                return {"error": f"Erro ao atualizar o status: {str(e)}"}
            finally:
                mydb.close()  # Fecha a conexão com o banco de dados



        def cancelar_pedido(self, id_pedido, motivo_cancelamento):
            mydb = Conexao.conectar()  # Conecta ao banco de dados
            mycursor = mydb.cursor()

            try:
                # Atualiza o status do pedido e define habilitado como 0
                     sql = "UPDATE tb_pedidos SET status = 'Cancelado',
habilitado = 0, motivo_cancelamento = %s WHERE id_pedido = %s"
                mycursor.execute(sql, (motivo_cancelamento, id_pedido))

                mydb.commit()  # Confirma a alteração
                   return {"message": "Pedido cancelado com sucesso!"}  #
Retorna mensagem de sucesso
            except Exception as e:
                mydb.rollback()  # Reverte a transação em caso de erro
                return {"error": f"Erro ao cancelar o pedido: {str(e)}"}
            finally:
                mydb.close()  # Fecha a conexão com o banco de dados



        def obter_dados_cliente_por_pedido(self, id_pedido):
            """
            Obtém os dados do cliente associados a um pedido específico.

             :param id_pedido: ID do pedido para o qual os dados do cliente
serão buscados.
            :return: Um dicionário com o telefone e nome do cliente, ou None
se não for encontrado.
            """
```

```
            try:
                # Query para buscar os dados do cliente associados ao pedido
                query = """
                SELECT c.telefone, c.nome_comp
                FROM tb_cliente c
                JOIN tb_pedidos p ON c.id_cliente = p.id_cliente
                WHERE p.id_pedido = %s
                """
                # Executa a query passando o ID do pedido como parâmetro
                dados = self.executar_query(query, (id_pedido,), fetch=True)

                # Verifica se algum dado foi retornado
                if dados:
                    # Retorna o primeiro resultado encontrado
                        return {'telefone': dados[0]['telefone'], 'nome':
dados[0]['nome_comp']}

                # Caso nenhum dado seja encontrado, retorna None
                return None

            except Exception as e:
                # Loga o erro para depuração
                print(f"Erro ao obter dados do cliente: {e}")
                return None


        def executar_query(self, query, params=None, fetch=False):
            """
            Executa uma query no banco de dados.

            :param query: A string SQL a ser executada.
            :param params: Uma tupla com os parâmetros para a query.
             :param fetch: Define se a função deve retornar os resultados
(True) ou não (False).
              :return: Os resultados da query se fetch=True, ou None caso
contrário.
            """
            try:
                # Conecta ao banco de dados
                   conn = Conexao.conectar()  # Certifique-se de que esse
método está implementado
                 cursor = conn.cursor(dictionary=True)  # Retorna resultados
como dicionários

                # Executa a query com os parâmetros
                cursor.execute(query, params)

                # Se fetch=True, retorna os resultados da query
                if fetch:
```

```
                    result = cursor.fetchall()
                    return result

                    # Confirma alterações no banco (para operações como
INSERT/UPDATE/DELETE)
                conn.commit()

            except mysql.connector.Error as e:
                # Loga o erro específico do banco
                print(f"Erro ao executar a query: {e}")
                return None

            except Exception as e:
                # Loga outros erros genéricos
                print(f"Erro inesperado ao executar a query: {e}")
                return None

            finally:
                # Fecha o cursor e a conexão
                if 'cursor' in locals() and cursor:
                    cursor.close()
                if 'conn' in locals() and conn:
                    conn.close()
```

```
    from flask import Flask, request, render_template, redirect, url_for,
session
    import os
    import smtplib
    import random
    from email.mime.text import MIMEText

    app = Flask(__name__)
    app.secret_key = 'supersecretkey'

    def send_verification_email(to_email):
        code = random.randint(100000, 999999)
        subject = "Seu código de verificação"
        body = f"Seu código de verificação é: {code}"

        smtp_server = "smtp.gmail.com"
        smtp_port = 587
        smtp_user = os.getenv('EMAIL_USER')
        smtp_password = os.getenv('EMAIL_PASS')

        msg = MIMEText(body)
        msg['Subject'] = subject
        msg['From'] = smtp_user
        msg['To'] = to_email
```

```
        with smtplib.SMTP(smtp_server, smtp_port) as server:
            server.starttls()
            server.login(smtp_user, smtp_password)
            server.send_message(msg)

        return code

    @app.route('/send_code', methods=['GET', 'POST'])
    def send_code():
        if request.method == 'POST':
            email = request.form['email']
            code = send_verification_email(email)
            session['verification_code'] = code
            return redirect(url_for('verify_code'))
        return render_template('send_code.html')

    @app.route('/verify_code', methods=['GET', 'POST'])
    def verify_code():
        if request.method == 'POST':
            entered_code = request.form['code']
            if entered_code == str(session.get('verification_code')):
                return redirect(url_for('success'))
            else:
                return "Código inválido. Tente novamente."
        return render_template('verify_code.html')

    @app.route('/success')
    def success():
        return "Verificação bem-sucedida! Você pode agora acessar o site."

    if __name__ == '__main__':
        app.run(debug=True)
```

```
    from conexao import Conexao
    from hashlib import sha256

    class Usuario:

        """
        Classe responsável por gerenciar as operações de usuários e produtos
no sistema.
         Essa classe oferece funcionalidades como cadastrar usuários, logar,
exibir cursos e categorias,
         inserir produtos, entre outras operações relacionadas ao usuário e
suas interações com o sistema.
        """
```

```
          def __init__(self):
              """
                  Método inicializador que define os atributos da classe Usuario.
Esses atributos
                  armazenam informações como telefone, nome, senha, email, curso,
tipo, e o estado de login do usuário.
              """
              self.tel = None
              self.nome = None
              self.senha = None
              self.email = None
              self.imagem = None
              self.preco = None
              self.nomeP = None
              self.categoria = None
              self.descricao = None
              self.curso = None
              self.tipo = None
              self.logado = False

          # Função de cadastro do usuário
          def cadastrar(self, nome, telefone, email, senha, curso, tipo):
                senha  =  sha256(senha.encode()).hexdigest()   # Criptografa  a
senha usando o algoritmo sha256

              try:
                  mydb = Conexao.conectar()  # Conecta ao banco de dados
                  mycursor = mydb.cursor()

                  # Verifica se o email ou telefone já estão cadastrados
                  mycursor.execute(
                       "SELECT id_cliente FROM tb_cliente WHERE email = %s OR
telefone = %s",
                      (email, telefone)
                  )
                  existing_user = mycursor.fetchone()

                  if existing_user:
                      # Retorna False para indicar que o cadastro foi impedido
por duplicidade
                      return False

                  # Insere o novo usuário caso não haja duplicidade
                   sql = "INSERT INTO tb_cliente (nome_comp, telefone, email,
id_curso, senha, tipo) VALUES (%s, %s, %s, %s, %s, %s)"
                   mycursor.execute(sql, (nome, telefone, email, curso, senha,
tipo))
```

```
                # Atualiza os atributos do objeto
                self.tel = telefone
                self.nome = nome
                self.senha = senha
                self.curso = curso
                self.email = email
                self.tipo = tipo
                self.logado = True  # Marca o usuário como logado

                mydb.commit()  # Confirma as alterações no banco de dados
                mydb.close()  # Fecha a conexão
                return True
            except Exception as e:
                print(f"Ocorreu um erro: {e}")  # Exibe uma mensagem de erro
em caso de falha
                return False

        def verificar_duplicidade(self, email, telefone):
            mydb = Conexao.conectar()
            mycursor = mydb.cursor()
            mycursor.execute(
                  "SELECT id_cliente FROM tb_cliente WHERE email = %s OR
telefone = %s",
                (email, telefone)
            )
            return bool(mycursor.fetchone())


        def exibir_cursos(self):
            """
             Retorna uma lista com todos os cursos disponíveis no sistema,
consultando a tabela `tb_curso`.
            Cada curso contém o ID do curso e o nome do curso.

            Retorno:
             - Uma lista de dicionários, onde cada dicionário representa um
curso com 'id_curso' e 'curso'.
            """
            mydb = Conexao.conectar()  # Conecta ao banco de dados
            mycursor = mydb.cursor()

            # Query SQL para selecionar todos os cursos
            sql = "SELECT * from tb_curso"
            mycursor.execute(sql)

            # Obtém os resultados e os organiza em uma lista
            resultado = mycursor.fetchall()
             lista_cursos = [{'id_curso': curso[0], 'curso': curso[1]} for
curso in resultado]
```

```
            mydb.commit()
            mydb.close()
            return lista_cursos



        def logar(self, email, senha):
            """
             Realiza o login de um usuário verificando o email e a senha criptografada no banco de dados.
             Se a combinação for encontrada, o estado do usuário é marcado como logado e os dados do usuário
            são carregados para os atributos da classe.

            Parâmetros:
            - email: endereço de email do usuário.
              - senha: senha do usuário, que será criptografada antes da verificação.

            Retorno:
             - None. O estado de login e os dados do usuário são atualizados nos atributos da classe.
            """
              senha = sha256(senha.encode()).hexdigest()  # Criptografa a senha usando o algoritmo sha256
            mydb = Conexao.conectar()  # Conecta ao banco de dados
            mycursor = mydb.cursor()

            try:
                # Query SQL para verificar se o email e a senha correspondem a um registro no banco de dados
                sql = "SELECT * FROM tb_cliente WHERE email = %s AND senha = %s"
                mycursor.execute(sql, (email, senha))

                # Busca um único registro (se houver)
                resultado = mycursor.fetchone()

                 # Se um registro for encontrado, atualiza os atributos do usuário
                if resultado:
                    self.logado = True
                    self.id_cliente = resultado[0]
                    self.nome = resultado[1]
                    self.tel = resultado[2]
                    self.email = resultado[3]
                    self.senha = resultado[5]
                    self.tipo = resultado[6]
```

```
                        self.primeiro_login = bool(resultado[8])  # Converte
para bool
                else:
                    self.logado = False
            finally:
                # Fecha o cursor e a conexão para liberar recursos
                mycursor.close()
                mydb.close()



        def atualizar_dados(self, id_cliente, telefone, email, senha):
            """
             Atualiza o telefone, email e senha do administrador e define
'primeiro_login' como False.

            Parâmetros:
            - id_cliente: ID do cliente (administrador) a ser atualizado
            - telefone: Novo número de telefone
            - email: Novo email
             - senha: Nova senha em texto puro que será criptografada para o
banco de dados

            Retorno:
            - True se a atualização for bem-sucedida, False caso contrário
            """
            # Hash da senha antes de armazenar no banco
            hashed_password = sha256(senha.encode()).hexdigest()

            # Conecta ao banco de dados
            mydb = Conexao.conectar()
            mycursor = mydb.cursor()

            try:
                 # Query para atualizar os dados do administrador e marcar
`primeiro_login` como `False`
                sql = """
                    UPDATE tb_cliente
                           SET telefone = %s, email = %s, senha = %s,
primeiro_login = %s
                    WHERE id_cliente = %s
                """
                     valores = (telefone, email, hashed_password, False,
id_cliente)

                # Executa a query
                mycursor.execute(sql, valores)
                mydb.commit()
```

```
                # Atualiza o atributo `primeiro_login` localmente
                self.primeiro_login = False
                return True
            except Exception as e:
                print("Erro ao atualizar dados do administrador:", e)
                mydb.rollback()
                return False
            finally:
                # Fecha o cursor e a conexão
                mycursor.close()
                mydb.close()


        def atualizar_telefone(self, id_cliente, telefone):
            """
            Atualiza o telefone do cliente no banco de dados.

            Parâmetros:
            - id_cliente: ID do cliente que terá o telefone atualizado
            - telefone: Novo número de telefone a ser salvo

            Retorno:
            - True se a atualização for bem-sucedida, False caso contrário
            """
            # Conectar ao banco de dados
            mydb = Conexao.conectar()
            mycursor = mydb.cursor()

            try:
                # Comando SQL para atualizar o telefone
                 sql = "UPDATE tb_cliente SET telefone = %s WHERE id_cliente
= %s"
                valores = (telefone, id_cliente)

                # Executa a atualização
                mycursor.execute(sql, valores)
                mydb.commit()
                print("Telefone atualizado com sucesso!")
                return True
            except Exception as e:
                print("Erro ao atualizar o telefone:", e)
                mydb.rollback()
                return False
            finally:
                # Fecha o cursor e a conexão com o banco
                mycursor.close()
                mydb.close()
```

```
          def logout(self, id_cliente):
              """
                Realiza o logout do usuário, removendo os itens do carrinho de
compras associado ao cliente.
                  Ao fazer logout, o carrinho e seus itens relacionados
(acompanhamentos e guarnições) são esvaziados.

              Parâmetros:
              - id_cliente: o ID do cliente que está realizando o logout.

              Retorno:
              - None.
              """
              try:
                  mydb = Conexao.conectar()  # Conecta ao banco de dados
                  mycursor = mydb.cursor()

                    # Remover os acompanhamentos relacionados ao carrinho do
cliente
                  sql_remover_acompanhamentos = """
                  DELETE FROM tb_carrinho_acompanhamento
                  WHERE id_carrinho IN (
                       SELECT id_carrinho FROM tb_carrinho WHERE id_cliente =
%s
                  )
                  """
                  mycursor.execute(sql_remover_acompanhamentos, (id_cliente,))

                  # Remover as guarnições relacionadas ao carrinho do cliente
                  sql_remover_guarnicoes = """
                  DELETE FROM tb_carrinho_guarnicao
                  WHERE id_carrinho IN (
                       SELECT id_carrinho FROM tb_carrinho WHERE id_cliente =
%s
                  )
                  """
                  mycursor.execute(sql_remover_guarnicoes, (id_cliente,))

                  # Remover os itens do carrinho do cliente
                      sql_remover_carrinho = "DELETE FROM tb_carrinho WHERE
id_cliente = %s"
                  mycursor.execute(sql_remover_carrinho, (id_cliente,))

                  # Confirmar as alterações no banco de dados
                  mydb.commit()

              except Exception as e:
                  print(f"Erro ao fazer logout: {e}")
                  mydb.rollback()  # Reverter alterações em caso de erro
```

```
            finally:
                mydb.close()  # Garantir que a conexão seja fechada



        def verificar_usuario(self, email, telefone):
            """
            Verifica se um usuário com o email e telefone fornecidos existe no sistema.

            Parâmetros:
            - email: endereço de email do usuário.
            - telefone: número de telefone do usuário.

            Retorno:
              - Retorna True se o usuário existir; caso contrário, retorna False.
            """
            mydb = Conexao.conectar()  # Conecta ao banco de dados
            mycursor = mydb.cursor()

             # Query SQL para verificar se o email e telefone existem na tabela tb_cliente
            sql = "SELECT * FROM tb_cliente WHERE email = %s AND telefone = %s"
            mycursor.execute(sql, (email, telefone))

            resultado = mycursor.fetchone()  # Busca um único registro

            mydb.close()  # Fecha a conexão

              return resultado is not None  # Retorna True se o usuário existir, False caso contrário

        def atualizar_senha(self, email, nova_senha):
            """
              Atualiza a senha de um usuário no sistema. A nova senha é criptografada antes de ser armazenada.

            Parâmetros:
            - email: endereço de email do usuário.
             - nova_senha: nova senha que será criptografada e atualizada no banco de dados.

            Retorno:
              - Retorna True se a atualização da senha for realizada com sucesso; caso contrário, retorna False.
            """
```

```
                    nova_senha   =   sha256(nova_senha.encode()).hexdigest()    #
Criptografa a nova senha usando o algoritmo sha256

                mydb = Conexao.conectar()  # Conecta ao banco de dados
                mycursor = mydb.cursor()

                # Query SQL para atualizar a senha na tabela tb_cliente
                sql = "UPDATE tb_cliente SET senha = %s WHERE email = %s"
                mycursor.execute(sql, (nova_senha, email))

                mydb.commit()  # Confirma as alterações
                mydb.close()  # Fecha a conexão

                  return mycursor.rowcount > 0  # Retorna True se a senha foi
atualizada, False caso contrário


            def tela_usuario(self, id_cliente):
                mydb = Conexao.conectar()
                mycursor = mydb.cursor()

                # Modifica a consulta SQL para incluir o INNER JOIN com tb_curso
                sql = f"""
                        SELECT  c.id_cliente,  c.nome_comp,  c.telefone,  c.email,
c.tipo, cu.curso
                    FROM tb_cliente c
                    INNER JOIN tb_curso cu ON c.id_curso = cu.id_curso
                    WHERE c.id_cliente = {id_cliente}
                """

                mycursor.execute(sql)

                 resultado = mycursor.fetchone()  # Use fetchone para obter um
único registro

                # Cria um dicionário com os dados do cliente e o nome do curso
                if resultado:
                    cliente_dict = {
                        "id_cliente": resultado[0],
                        "nome_comp": resultado[1],
                        "telefone": resultado[2],
                        "email": resultado[3],
                         "nome_curso": resultado[5]  # Adiciona o nome do curso
ao dicionário
                    }
                else:
                    cliente_dict = None

                mydb.close()  # Fecha a conexão
```

```
            return cliente_dict  # Retorna o dicionário com os dados do
cliente e o nome do curso
```

APÊNDICE K – Front-End

```
      <!DOCTYPE html>
      <html lang="pt-BR">

      <head>
          <meta charset="UTF-8">
              <meta   name="viewport"   content="width=device-width,   initial-
scale=1.0">
          <title>Atualizar Dados</title>
                                          <link               rel="stylesheet"
href="../static/styles/atualizarDadosIniciais.css">
      </head>

      <body>
          <div class="btn-voltar">
                              <button     onclick="window.history.back()"><img
src="https://www.svgrepo.com/show/324205/back-arrow-navigation.svg"
                      alt=""></button>
          </div>
          <main>

              <div class="titulo">
                  <h2>Atualizar Dados Iniciais</h2>
              </div>

              <div class="container-principal">
                  <div class="formulario">
                      <form action="{{ url_for('atualizar_dados_iniciais') }}"
method="POST">
                               <input type="tel" id="telefone" name="telefone"
required placeholder="*Telefone">
                                  <input  type="email"  id="email"  name="email"
required placeholder="*E-mail">
                               <input type="password" id="senha" name="senha"
required placeholder="*Senha">
                              <button type="submit">Atualizar Dados</button>
                      </form>
                  </div>
```

```
            </div>

        </main>
    </body>

    </html>
```

```
    {% extends "modelo-adm.html" %}
    {% block conteudo %}
    <style>
                                                                @import
url('https://fonts.googleapis.com/css2?family=Inter:ital,opsz,wght@0,14..32,10
0..900;1,14..32,100..900&display=swap');
    </style>
    <link rel="stylesheet" href="../static/styles/cad-produto.css">

    <body>
        <main>
            <div class="btn-voltar">
                <button onclick="window.history.back()"><img
                        src="https://www.svgrepo.com/show/324205/back-arrow-
navigation.svg" alt=""></button>
            </div>

                        <form    action="/inserir_produtos"    method="post"
enctype="multipart/form-data">
                <div class="espacamento1"></div>
                <h2>Adicione um produto!</h2>
                  <input name="nome" type="text" id="nome" placeholder="Nome
do produto" required>
                    <input  name="preco"  type="number"  id="preco"  min="0.00"
max="1000.00" step="0.01" placeholder="Preço"
                     required>
                    <input  name="img"  type="file"  id="img"  accept="image/*"
class="input-image" required>
                                <input    name="descricao"    id="descricao"
placeholder="Descrição do produto" class="input-descricao" required>
                        <select  id="categoria"  name="categoria"  required
onchange="exibirGuarnicoes()">
                    {% for registro in categorias %}
                            <option  value="{{  registro.id_categoria  }}">{{
registro.nome }}</option>
                    {% endfor %}
                </select>

                <!-- Campo de tamanho da marmita (inicialmente oculto) -->
                <div id="tamanhoDiv" style="display: none;">
```

```
                    <select name="tamanho" id="tamanho">
                            <option value="" disabled selected>Selecione um
tamanho</option> <!-- Placeholder -->
                        <option value="Pequena">Pequena</option>
                        <option value="Média">Média</option>
                        <option value="Grande">Grande</option>
                    </select>
                </div>

                <!-- Guarnições e Acompanhamentos (inicialmente ocultos) -->
                  <div id="guarnicoes_acompanhamentos" style="display: none;
margin-top: 15px;">


                    <!-- Guarnições -->
                    <h5>Guarnições:</h5>
                    {% for guarnicao in lista_guarnicao %}
                    <div class="acompanhamento">
                                                                        <img
src="https://pedido.anota.ai/assets/item_no_image-a8c57261.png" alt="">
                                <label for="guarnicao_{{ guarnicao.id_guarnicao
}}">{{ guarnicao.nome_guarnicao }}</label>
                                        <input type="checkbox" id="guarnicao_{{
guarnicao.id_guarnicao }}" name="guarnicoes"
                                value="{{ guarnicao.id_guarnicao }}">
                    </div>
                    {% endfor %}
                    <div id="lista_novas_guarnicoes"></div>

                    <!-- Acompanhamentos -->
                    <h5>Acompanhamentos:</h5>
                    {% for acompanhamento in lista_acompanhamento %}
                    <div class="acompanhamento">
                                                                        <img
src="https://pedido.anota.ai/assets/item_no_image-a8c57261.png" alt="">
                                                <label for="acompanhamento_{{
acompanhamento.id_acompanhamento }}">{{
                                acompanhamento.nome_acompanhamento }}</label>
                                    <input type="checkbox" id="acompanhamento_{{
acompanhamento.id_acompanhamento }}"
                                                name="acompanhamentos" value="{{
acompanhamento.id_acompanhamento }}">
                    </div>
                    {% endfor %}
                    <div id="lista_novos_acompanhamentos"></div>
                </div>
```

```
                    <div  id="nova_guarnicao_acompanhamento"  style="display:
none; margin-top: 15px;">
                    <h4>Adicionar nova guarnição ou acompanhamento:</h4>

                    <!-- Guarnição -->
                    <div class="input-container">
                                    <input  type="text"  id="nome_guarnicao"
placeholder="Nome da nova guarnição">
                                                        <button    type="button"
onclick="adicionarGuarnicao()">+</button>
                    </div>

                    <!-- Acompanhamento -->
                    <div class="input-container">
                                <input  type="text"  id="nome_acompanhamento"
placeholder="Nome do novo acompanhamento">
                                                        <button    type="button"
onclick="adicionarAcompanhamento()">+</button>
                    </div>
                </div>


                                    <button    type="submit"    class="btn-
Cadastrar">Adicionar</button>
            </form>
        </main>

        <script>
            function exibirGuarnicoes() {
                var categoriaSelect = document.getElementById("categoria");
                                                var      guarnicoesDiv      =
document.getElementById("guarnicoes_acompanhamentos");
                                            var      novaGuarnicaoDiv      =
document.getElementById("nova_guarnicao_acompanhamento");
                var tamanhoDiv = document.getElementById("tamanhoDiv");
                var tamanhoSelect = document.getElementById("tamanho");

                    // Se a categoria selecionada for "Marmitex", exibe as
guarnições, acompanhamentos e o tamanho
                                                                        if
(categoriaSelect.options[categoriaSelect.selectedIndex].text === "Marmitex") {
                    guarnicoesDiv.style.display = "block";
                    novaGuarnicaoDiv.style.display = "block";
                    tamanhoDiv.style.display = "block";
                    tamanhoSelect.setAttribute('required', 'required');
                } else {
                    guarnicoesDiv.style.display = "none";
                    novaGuarnicaoDiv.style.display = "none";
                    tamanhoDiv.style.display = "none";
```

```
                    tamanhoSelect.removeAttribute('required');
                    tamanhoSelect.value = "";
                }
            }

            function adicionarGuarnicao() {
                                                var      nomeGuarnicao      =
document.getElementById("nome_guarnicao").value.trim();
                if (nomeGuarnicao) {
                    fetch('/adicionar_guarnicao', {
                        method: 'POST',
                        headers: {
                                    'Content-Type':  'application/x-www-form-
urlencoded',
                        },
                                                body:    'nome_guarnicao='    +
encodeURIComponent(nomeGuarnicao)
                    })
                        .then(response => response.json())
                        .then(data => {
                            if (data.success) {
                                    // Adiciona a nova guarnição na lista de
guarnições
                                                var  listaNovasGuarnicoesDiv  =
document.getElementById("lista_novas_guarnicoes");

                                                    var   novaGuarnicaoDiv   =
document.createElement("div");

novaGuarnicaoDiv.classList.add("acompanhamento");

                                    // Cria o elemento de imagem
                                    var imagem = document.createElement("img");
                                                            imagem.src    =
"https://pedido.anota.ai/assets/item_no_image-a8c57261.png";
                                    imagem.alt = "Imagem de guarnição";

                                    // Cria o input de checkbox
                                                        var    checkbox    =
document.createElement("input");
                                    checkbox.type = "checkbox";
                                    checkbox.name = "guarnicoes";
                                    checkbox.value = data.id_guarnicao;
                                    checkbox.checked = true;

                                    // Cria o label para a guarnição
                                    var label = document.createElement("label");
                                    label.textContent = nomeGuarnicao;
```

```
                                // Anexa a imagem, checkbox e label à div de
guarnição
                                novaGuarnicaoDiv.appendChild(imagem);
                                novaGuarnicaoDiv.appendChild(checkbox);
                                novaGuarnicaoDiv.appendChild(label);

                                // Adiciona a nova div à lista de guarnições

listaNovasGuarnicoesDiv.appendChild(novaGuarnicaoDiv);


document.getElementById("nome_guarnicao").value = ""; // Limpa o campo de
texto
                            } else {
                                alert("Erro ao adicionar a guarnição.");
                            }
                        });
                }
            }


            function adicionarAcompanhamento() {
                                            var     nomeAcompanhamento     =
document.getElementById("nome_acompanhamento").value.trim();
                if (nomeAcompanhamento) {
                    fetch('/adicionar_acompanhamento', {
                        method: 'POST',
                        headers: {
                                    'Content-Type': 'application/x-www-form-
urlencoded',
                        },
                                                body:   'nome_acompanhamento='   +
encodeURIComponent(nomeAcompanhamento)
                    })
                        .then(response => response.json())
                        .then(data => {
                            if (data.success) {
                                  // Adiciona o novo acompanhamento na lista
de acompanhamentos
                                            var listaNovosAcompanhamentosDiv =
document.getElementById("lista_novos_acompanhamentos");
                                                var novoAcompanhamentoDiv =
document.createElement("div");

novoAcompanhamentoDiv.classList.add("acompanhamento");

                                // Cria o elemento de imagem
                                var imagem = document.createElement("img");
```

```
                                                                imagem.src    =
"https://pedido.anota.ai/assets/item_no_image-a8c57261.png";
                                imagem.alt = "Imagem de acompanhamento";

                                // Cria o input de checkbox
                                                        var   checkbox   =
document.createElement("input");
                                checkbox.type = "checkbox";
                                checkbox.name = "acompanhamentos";
                                checkbox.value = data.id_acompanhamento;
                                checkbox.checked = true;

                                // Cria o label para o acompanhamento
                                var label = document.createElement("label");
                                label.textContent = nomeAcompanhamento;

                                // Anexa a imagem, checkbox e label à div de
acompanhamento
                                novoAcompanhamentoDiv.appendChild(imagem);
                                novoAcompanhamentoDiv.appendChild(checkbox);
                                novoAcompanhamentoDiv.appendChild(label);

                                        // Adiciona a nova div à lista de
acompanhamentos

listaNovosAcompanhamentosDiv.appendChild(novoAcompanhamentoDiv);


document.getElementById("nome_acompanhamento").value = ""; // Limpa o campo de
texto
                            } else {
                                                alert("Erro  ao  adicionar  o
acompanhamento.");
                            }
                        });
                }
            }


        </script>
    </body>
    {% endblock %}
```

```
    {% extends "modelo.html" %}
    {% block conteudo %}

    <link rel="stylesheet" href="../static/styles/cadastro.css">
```

```
      <link        href="https://cdnjs.cloudflare.com/ajax/libs/select2/4.1.0-
rc.0/css/select2.min.css" rel="stylesheet" />
      <link rel="stylesheet" href="../static/styles/cadastro.css">

      <body>
          <main>
              <!-- Botão Voltar -->
              <div class="btn-voltar">
                  <button onclick="window.history.back()" aria-label="Voltar">
                          <img  src="https://www.svgrepo.com/show/324205/back-
arrow-navigation.svg" alt="Voltar">
                  </button>
              </div>

              <!-- Título -->
              <div class="titulo">
                  <h2>Vamos Começar!</h2>
              </div>

              <!-- Formulário -->
              <section class="container-principal">
                  <div class="formulario">
                      <form id="form-cadastro" action="" method="post">
                              <input  id="nome"  name="nome"  placeholder="Nome
Completo" type="text" required>

                          <div class="form-group">
                              <div class="input-group">
                                  <span class="prefix">+55</span>
                                                  <input   id="tel"   name="tel"
placeholder="Número de Telefone" type="text"
                                      oninput="formatPhone()" required>
                              </div>
                          </div>

                           <input id="email" name="email" placeholder="E-mail"
type="email" required>
                           <input id="senha" name="senha" placeholder="Senha"
type="password" required>

                          <div class="select">
                               <select id="curso" name="curso" class="select2"
required>
                                          <option  value="">Selecione  o  seu
curso</option>
                                  {% for registro in cursos %}
                                    <option value="{{ registro.id_curso }}">{{
registro.curso }}</option>
                                  {% endfor %}
```

```
                            </select>
                        </div>

                        <button type="submit">Registrar</button>
                                <p>Já  está  registrado?  <a  href="/logar"
class="login-btn">Faça o login</a></p>
                    </form>
                </div>
            </section>

            <!-- Popup de erro -->
            {% if get_flashed_messages() %}
            <div class="popup-container">
                <div class="popup error-popup">
                    <div class="popup-icon error-icon">
                                                    <svg    class="error-svg"
xmlns="http://www.w3.org/2000/svg" viewBox="0 0 20 20" aria-hidden="true">
                            <path fill-rule="evenodd"
                                  d="M10 18a8 8 0 100-16 8 8 0 000 16zM8.707
7.293a1  1  0  00-1.414  1.414L8.586  10l-1.293  1.293a1  1  0  101.414  1.414L10
11.414l1.293 1.293a1 1 0 001.414-1.414L11.414 10l1.293-1.293a1 1 0 00-1.414-
1.414L10 8.586 8.707 7.293z"
                                clip-rule="evenodd"></path>
                            </svg>
                    </div>
                     <div class="error-message">{{ get_flashed_messages()[0]
}}</div>
                                        <div    class="popup-icon    close-icon"
onclick="closePopup()">
                         <svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0
20 20" class="close-svg">
                            <path
                                        d="m15.8333  5.34166-1.175-1.175-4.6583
4.65834-4.65833-4.65834-1.175 1.175 4.65833 4.65834-4.65833 4.6583 1.175 1.175
4.65833-4.6583 4.6583 4.6583 1.175-1.175-4.6583-4.6583z"
                                class="close-path"></path>
                            </svg>
                    </div>
                </div>
            </div>
            {% endif %}
        </main>

        <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
          <script  src="https://cdnjs.cloudflare.com/ajax/libs/select2/4.1.0-
rc.0/js/select2.min.js"></script>

        <script>
```

```
                                                document.getElementById('form-
cadastro').addEventListener('submit', function (e) {
                    e.preventDefault();
                    var telInput = document.getElementById('tel');
                    var telValue = telInput.value.trim();

                    if (telValue) {
                            telValue = '+55' + telValue.replace(/\D/g, ''); //
Remove qualquer caractere não numérico
                        telInput.value = telValue;
                    }

                    this.submit();
                });

                function formatPhone() {
                    var telInput = document.getElementById('tel');
                    var value = telInput.value.replace(/\D/g, ''); // Remove não
numéricos
                    if (value.length > 10) {
                        value = value.replace(/(\d{2})(\d{5})(\d{4})/, '($1) $2-
$3'); // Formato para celular
                    } else if (value.length > 6) {
                        value = value.replace(/(\d{2})(\d{4})(\d{4})/, '($1) $2-
$3'); // Formato para telefone fixo
                    } else if (value.length > 2) {
                        value = value.replace(/(\d{2})(\d{1,5})/, '($1) $2'); //
Formato para DDD
                    }
                    telInput.value = value;
                }

                $(document).ready(function () {
                    $('.select2').select2({
                        placeholder: 'Selecione o seu curso',
                        allowClear: true,
                        width: 'resolve'
                    });

                    var popup = $('.popup-container');
                    if (popup.length && popup.is(':visible')) {
                        setTimeout(function () {
                            popup.fadeOut();
                        }, 3000);
                    }
                });
            </script>
            {% endblock %}
```

```
      {% extends "modelo.html" %}
      {% block conteudo %}
      <link rel="stylesheet" href="../static/styles/carrinho.css">

      <div class="espacamento1"></div>

      <body>
          <main>

              {% for registro in lista_carrinho.produtos %}
              <div class="form-exclui-carrinho">
                  <section class="container-principal">
                      <div class="produto-card">
                          <div class="imgdiv">
                                              <img  class="img-produto"  src="{{
registro.imagem_produto }}" alt="Imagem Produto">
                                  <button class="button btn-excluir" data-id="{{
registro.id_carrinho }}">
                                               <img  src="../static/img/lixo.png"
alt="Excluir Produto">
                              </button>
                              <div class="direita-card">
                                  <div class="cima">
                                      <span>{{ registro.nome_produto }}</span>
                                      <div class="qnt-btn">
                                                    <button  class="menos-btn"
type="button"
                                                 data-id="{{ registro.id_carrinho
}}">-</button>
                                             <input type="number" id="quantidade-
{{ registro.id_carrinho }}"
                                                    value="{{ registro.quantidade
}}" data-id="{{ registro.id_carrinho }}" min="1">
                                                      <button  class="mais-btn"
type="button"
                                                 data-id="{{ registro.id_carrinho
}}">+</button>
                                      </div>
                                  </div>
                                  <div class="baixo">
                                        <span class="desc-baixo">{{ 'Descrição
do produto' }}</span>
                                         <span  class="preco-item">R$  {{
registro.preco }}</span>
                                         <span  class="obs-item">OBS:  {{
'Observação do produto' }}</span>
```

```
                                </div>
                            </div>
                        </div>
                    </div>
                </section>
            </div>
            {% endfor %}

            {% for registro in lista_carrinho.marmitas %}
            <div class="form-exclui-carrinho">
                <section class="container-principal">
                    <div class="produto-card">
                        <div class="imgdiv">
                                            <img  class="img-produto"  src="{{
registro.imagem_produto }}" alt="Imagem Marmita">
                                <button class="button btn-excluir" data-id="{{
registro.id_carrinho }}">
                                            <img  src="../static/img/lixo.png"
alt="Excluir Marmita">
                            </button>
                            <div class="direita-card">
                                <div class="cima">
                                    <span>{{ registro.nome_marmita }}</span>
                                    <div class="qnt-btn">
                                                <button  class="menos-btn"
type="button"
                                            data-id="{{ registro.id_carrinho
}}">-</button>
                                        <input type="number" id="quantidade-
{{ registro.id_carrinho }}"
                                                value="{{ registro.quantidade
}}" data-id="{{ registro.id_carrinho }}" min="1">
                                                <button  class="mais-btn"
type="button"
                                            data-id="{{ registro.id_carrinho
}}">+</button>
                                    </div>
                                </div>
                                <div class="baixo">
                                        <span  class="desc-baixo">{{
registro.descricao }}</span>
                                    <span  class="preco-item">R$  {{
registro.preco }}</span>

                                    <div class="guarnicoes">
                                        <strong>Guarnições:</strong>
                                        <ul>
                                                {% for guarnicao in
registro.guarnicoes %}
```

```
                                        <li>{{ guarnicao }}</li>
                                        {% endfor %}
                                    </ul>
                                </div>

                                <div class="acompanhamentos">
                                    <strong>Acompanhamentos:</strong>
                                    <ul>
                                        {% for acompanhamento in
registro.acompanhamentos %}
                                        <li>{{ acompanhamento }}</li>
                                        {% endfor %}
                                    </ul>
                                </div>

                                    <span class="obs-item">OBS: {{
'Observação da marmita' }}</span>
                            </div>
                        </div>
                    </div>
                </div>
            </section>
        </div>
        {% endfor %}

        <div class="finalizacao">
            <div class="preco">
                <span class="span1">Total:</span>
                    <span id="total-preco" class="span2">R$ {{
lista_carrinho.total_preco }}</span>
            </div>

                    <button type="button" onclick="enviarCarrinho()"
class="button1">Enviar</button>
            <a href="/">Continuar Comprando</a>
        </div>
    </main>


    <div id="modal-confirmacao" class="modal" style="display: none;">
        <div class="modal-conteudo">
                <p>Tem certeza desse pedido? Não será possível o
cancelamento.</p>
            <div class="botoes-modal">
                    <button id="confirmar-envio" class="button-
confirmar">Confirmar</button>
                    <button id="cancelar-envio" class="button-
cancelar">Cancelar</button>
            </div>
```

```
            </div>
        </div>


        <script>
            document.addEventListener('DOMContentLoaded', function () {
                // Função para atualizar a quantidade de produtos
                document.querySelectorAll('.mais-btn').forEach(button => {
                    button.addEventListener('click', function () {
                        const idCarrinho = this.getAttribute('data-id');
                                                const    quantidadeInput    =
document.getElementById(`quantidade-${idCarrinho}`);
                            let quantidade = parseInt(quantidadeInput.value,
10);
                        quantidade++;
                        quantidadeInput.value = quantidade;
                        atualizarQuantidade(idCarrinho, quantidade);
                    });
                });

                document.querySelectorAll('.menos-btn').forEach(button => {
                    button.addEventListener('click', function () {
                        const idCarrinho = this.getAttribute('data-id');
                                                const    quantidadeInput    =
document.getElementById(`quantidade-${idCarrinho}`);
                            let quantidade = parseInt(quantidadeInput.value,
10);
                        if (quantidade > 1) {
                            quantidade--;
                            quantidadeInput.value = quantidade;
                            atualizarQuantidade(idCarrinho, quantidade);
                        } else {
                            alert("Número mínimo de pedido é 1");
                        }
                    });
                });

                 // Função AJAX para atualizar a quantidade de produtos no
backend
                function atualizarQuantidade(idCarrinho, quantidade) {
                    fetch('/atualizar_quantidade', {
                        method: 'POST',
                        headers: {
                            'Content-Type': 'application/json'
                        },
                            body: JSON.stringify({ id_carrinho: idCarrinho,
quantidade: quantidade })
                    })
                        .then(response => response.json())
```

```
                        .then(data => {
                            if (data.success) {
                                atualizarPrecoTotal(); // Atualiza o preço
total
                            } else {
                                alert('Erro ao atualizar quantidade');
                            }
                        })
                        .catch(error => {
                            console.error('Erro:', error);
                        });
                }

                // Função para excluir produtos do carrinho
                 document.querySelectorAll('.btn-excluir').forEach(button =>
{
                    button.addEventListener('click', function () {
                        const idCarrinho = this.getAttribute('data-id');
                        excluirProduto(idCarrinho);
                    });
                });

                // Função AJAX para excluir produto do carrinho
                function excluirProduto(idCarrinho) {
                    fetch('/excluir_produto_carrinho', {
                        method: 'POST',
                        headers: {
                            'Content-Type': 'application/json'
                        },
                        body: JSON.stringify({ id_carrinho: idCarrinho })
                    })
                        .then(response => {
                            if (!response.ok) {
                                throw new Error('Erro na requisição: ' +
response.status);
                            }
                            return response.json();
                        })
                        .then(data => {
                            if (data.success) {
                                location.reload(); // Atualiza a página após
exclusão bem-sucedida
                            } else {
                                alert('Erro ao excluir produto');
                            }
                        })
                        .catch(error => {
                            console.error('Erro:', error);
                        });
```

```
                }

                // Função AJAX para atualizar o preço total do carrinho
                function atualizarPrecoTotal() {
                    fetch('/atualizar_preco_total', {
                        method: 'GET'
                    })
                        .then(response => response.json())
                        .then(data => {
                            if (data.success) {
                                document.getElementById('total-
preco').innerText = `R$ ${data.total_preco}`;
                            } else {
                                alert('Erro ao atualizar o preço total');
                            }
                        })
                        .catch(error => {
                            console.error('Erro:', error);
                        });
                }
            });

            // Função para enviar o carrinho via AJAX
            function enviarCarrinho() {
                const modal = document.getElementById('modal-confirmacao');
                modal.style.display = 'flex'; // Exibe o modal

                // Botão para confirmar o envio
                    document.getElementById('confirmar-envio').onclick =
function () {
                    modal.style.display = 'none'; // Oculta o modal após a
confirmação

                    // Coletando os itens do carrinho
                    const itensCarrinho = [];
                                document.querySelectorAll('.produto-
card').forEach(produto => {
                                    const idCarrinho =
produto.querySelector('input[type="number"]').getAttribute('data-id');
                                    const quantidade =
produto.querySelector('input[type="number"]').value;
                                itensCarrinho.push({ id_carrinho: idCarrinho,
quantidade: quantidade });
                    });

                    fetch('/enviar_carrinho', {
                        method: 'POST',
                        headers: {
                            'Content-Type': 'application/json'
```

```
                        },
                        body: JSON.stringify({ itens: itensCarrinho })
                    })
                        .then(response => {
                            if (response.status === 403) {
                                alert("Site offline. Não é possível enviar
pedidos neste horário.");
                                return;
                            }
                            if (!response.ok) {
                                throw new Error('Erro na requisição: ' +
response.status);
                            }
                            return response.json();
                        })
                        .then(data => {
                            if (data.success) {
                                alert(data.message);
                                window.location.href = "/"; // Redireciona
para a página inicial
                            } else {
                                alert(data.message || 'Erro ao enviar o
carrinho');
                            }
                        })
                        .catch(error => {
                            console.error('Erro:', error);
                        });
                };

                // Botão para cancelar o envio
                document.getElementById('cancelar-envio').onclick = function
() {
                    modal.style.display = 'none'; // Oculta o modal
                };
            }
        </script>
    </body>
    {% endblock %}
```

```
    {% extends "modelo-adm.html" %}
    {% block conteudo %}
    <link rel="stylesheet" href="../static/styles/editarMarmita.css">

    <body>
        <main>
            <table>
```

```
                <tbody>
                    {% for marmita in lista_marunica %}
                    <tr>
                        <td>
                            <form id="form-marmita-{{ marmita.id_marmita }}"
enctype="multipart/form-data"       onsubmit="atualizarMarmita(event,       {{
marmita.id_marmita }})">
                                <input type="hidden" name="id_marmita"
value="{{ marmita.id_marmita }}">

                                <div style="display: flex; flex-direction:
column; align-items: center; position: relative; border: 3px solid rgb(165,
165, 165); border-radius: 12px; width: 253px;">
                                    <!-- Imagem atual da marmita -->
                                    <img src="{{ marmita.imagem_marmita }}"
                                         alt="Imagem da Marmita {{
marmita.nome_marmita }}"
                                        style="width: 250px; height: 250px;
cursor: pointer; border-radius: 8px;"

 onclick="document.getElementById('inputImagem{{            marmita.id_marmita
}}').click();"
                                                              id="previewImagem{{
marmita.id_marmita }}">

                                    <!-- Input para selecionar nova imagem -
->
                                        <input type="file" name="imagem"
accept="image/png, image/jpeg"
                                                               id="inputImagem{{
marmita.id_marmita }}" style="display: none;"
                                                  onchange="previewImagem(event,
'{{ marmita.id_marmita }}')">

                                    <!-- Ícone de edição -->
                                    <img src="../static/img/btn-editar.png"
                                        alt="Editar imagem"
                                          style="position: absolute; top: -
13px; left: -15px; width: 30px; height: 30px; border-radius: 50%; background-
color: white; padding: 5px; box-shadow: 0px 0px 5px rgba(0, 0, 0, 0.2);
cursor: pointer;"

 onclick="document.getElementById('inputImagem{{            marmita.id_marmita
}}').click();">
                                </div>
                            </td>
                            <td>
                                <input type="text" name="nome" value="{{
marmita.nome_marmita }}" required>
```

```
                        </td>
                        <td>
                                <input type="text" name="preco" value="{{
marmita.preco }}" required>
                        </td>
                        <td>
                                    <textarea name="descricao" required>{{
marmita.descricao }}</textarea>
                        </td>
                        <td>
                            <select name="tamanho" required>
                                        <option value="Pequena" {% if
marmita.tamanho == 'Pequena' %}selected{% endif %}>Pequena</option>
                                    <option value="Média" {% if marmita.tamanho
== 'Média' %}selected{% endif %}>Média</option>
                                <option value="Grande" {% if marmita.tamanho
== 'Grande' %}selected{% endif %}>Grande</option>
                            </select>
                        </td>
                        <td class="acompanhamento-container">
                                                                        <span
class="nomeguaracomp">Acompanhamentos</span>
                              {% for acompanhamento in todos_acompanhamentos
%}

                            <label>
                                                    <input type="checkbox"
name="acompanhamentos[]" value="{{ acompanhamento.id }}" {% if
acompanhamento.id in marmita.acompanhamentos %}checked{% endif %}>
                                <span>{{ acompanhamento.nome }}</span>
                            </label>
                            {% endfor %}
                        </td>

                        <td class="guarnicao-container">
                            <span class="nomeguaracomp">Guarnições</span>
                            {% for guarnicao in todas_guarnicoes %}

                            <label>
                                  <input type="checkbox" name="guarnicoes[]"
value="{{ guarnicao.id }}" {% if guarnicao.id in marmita.guarnicoes
%}checked{% endif %}>
                                <span>{{ guarnicao.nome }}</span>
                            </label>
                            {% endfor %}
                        </td>
                        <td>
                            <button type="submit">Atualizar Marmita</button>
                        </td>
```

```
                        </form>
                    </tr>
                    {% endfor %}
                </tbody>
            </table>
        </main>
    </body>

    <div class="espacamento1"></div>

    <script>
    function atualizarMarmita(event, idMarmita) {
        event.preventDefault();

        let form = document.getElementById('form-marmita-' + idMarmita);
        let formData = new FormData(form);

        fetch("/atualizar_marmita", {
            method: "POST",
            body: formData,
        })
        .then(response => response.json())
        .then(data => {
            if (data.status === 'success') {
                alert(data.message);
                if (data.url_imagem) {
                    document.getElementById('imagem-atual-' + idMarmita).src
= data.url_imagem;
                }
            } else {
                alert(data.message);
            }
        })
        .catch(error => {
            console.error('Erro:', error);
            alert('Erro ao tentar atualizar a marmita.');
        });
    }

    function previewImagem(event, codProduto) {
                    var preview = document.getElementById('previewImagem' +
codProduto);
                preview.src = URL.createObjectURL(event.target.files[0]);
            }
    </script>

    {% endblock %}
```

```
      {% extends "modelo-adm.html" %}
      {% block conteudo %}
      <link rel="stylesheet" href="../static/styles/editarProdutos.css">

      <body>
          <main>
              <table>
                  <tbody>
                      {% for registro in lista_prounico %}
                      <tr>
                          <td>
                               <form method="POST" action="/atualizar_produto"
enctype="multipart/form-data">
                                    <input type="hidden" name="id_produto"
value="{{ registro.cod_produto }}">

                                  <div style="display: flex; flex-direction:
column; align-items: center; position: relative; border: 3px solid rgb(165,
165, 165); border-radius: 12px; width: 253px;">
                                    <!-- Imagem atual do produto -->
                                    <img src="{{ registro.imagem_produto }}"
                                                alt="Imagem do produto {{
registro.nome_produto }}"
                                         style="width: 250px; height: 250px;
cursor: pointer; border-radius: 8px;"

 onclick="document.getElementById('inputImagem{{          registro.cod_produto
}}').click();"
                                                          id="previewImagem{{
registro.cod_produto }}">

                                    <!-- Input para selecionar nova imagem -
->
                                           <input type="file" name="imagem"
accept="image/png, image/jpeg"
                                                                id="inputImagem{{
registro.cod_produto }}" style="display: none;"
                                                   onchange="previewImagem(event,
'{{ registro.cod_produto }}')">

                                    <!-- Ícone de edição -->
                                    <img src="../static/img/btn-editar.png"
                                         alt="Editar imagem"
                                           style="position: absolute; top: -
13px; left: -15px; width: 30px; height: 30px; border-radius: 50%; background-
```

```
color: white; padding: 5px; box-shadow: 0px 0px 5px rgba(0, 0, 0, 0.2);
cursor: pointer;"

 onclick="document.getElementById('inputImagem{{          registro.cod_produto
}}').click();">
                                    </div>

                            </td>
                            <td>
                                        <input type="text" name="nome" value="{{
registro.nome_produto }}" required>
                            </td>
                            <td>
                                       <input type="text" name="preco" value="{{
registro.preco }}" required>
                            </td>
                            <td>
                                              <textarea name="descricao" required
style="height: 80px;">{{ registro.descricao }}</textarea>
                            </td>
                            <td>
                                <button type="submit">Atualizar</button>
                            </td>
                            </form>
                        </tr>
                        {% endfor %}
                    </tbody>
                </table>
            </main>

            <script>
                function previewImagem(event, codProduto) {
                       var preview = document.getElementById('previewImagem' +
codProduto);
                    preview.src = URL.createObjectURL(event.target.files[0]);
                }
            </script>

        </body>
        {% endblock %}
```

```
        {% extends "modelo.html" %}
        {% block conteudo %}
        <link rel="stylesheet" href="../static/styles/historico.css">

        <section class="container-cards">
            <h2>Histórico de Pedidos</h2>
```

```
            <div id="order-list">
                <!-- O conteúdo dos pedidos será carregado aqui via AJAX -->
            </div>
        </section>

        <script>
            function fetchOrders() {
                fetch('/exibir_historico_ajax')
                    .then(response => response.json())
                    .then(data => {
                        if (data.redirect) {
                                    window.location.href  =  data.redirect;   //
Redireciona para login, se necessário
                        } else {
                            renderOrders(data);
                        }
                    })
                     .catch(error => console.error('Erro ao carregar pedidos:',
error));
            }

            function renderOrders(lista_historico) {
                const orderList = document.getElementById('order-list');
                if (!orderList) {
                    console.error("Element with ID 'order-list' not found.");
                    return;
                }

                orderList.innerHTML = '';

                            for    (const    [id_cliente,    dados_cliente]    of
Object.entries(lista_historico)) {
                    const clientDiv = document.createElement('div');
                    clientDiv.classList.add('card-pedidos');

                                for    (const    [id_pedido,    dados_pedido]    of
Object.entries(dados_cliente.pedidos)) {
                        const orderCard = document.createElement('div');
                        orderCard.classList.add('order-card');

                        const orderHeader = document.createElement('div');
                        orderHeader.classList.add('order-header');
                        orderHeader.innerHTML = `
                        <span class="order-number">#${id_pedido}</span>
                        <span class="order-time">${dados_pedido.hora}</span>
                                                    <span     class="order-
date">${dados_pedido.data_pedido}</span>
                    `;
                        orderCard.appendChild(orderHeader);
```

```
                    const orderDetails = document.createElement('div');
                    orderDetails.classList.add('order-details');
                    const productList = document.createElement('ul');

                    dados_pedido.produtos.forEach(produto => {
                        const productItem = document.createElement('li');
                            const precoProduto = typeof produto.preco ===
'number' ? produto.preco : parseFloat(produto.preco) || 0;
                            productItem.textContent = `${produto.quantidade}x
${produto.nome_produto} - R$ ${precoProduto.toFixed(2)}`;
                        productList.appendChild(productItem);
                    });

                    dados_pedido.marmitas.forEach(marmita => {
                        const marmitaItem = document.createElement('li');
                            const precoMarmita = typeof marmita.preco ===
'number' ? marmita.preco : parseFloat(marmita.preco) || 0;
                            marmitaItem.textContent = `${marmita.quantidade}x
Marmita: ${marmita.nome_marmita} - R$ ${(precoMarmita *
marmita.quantidade).toFixed(2)}`;
                        productList.appendChild(marmitaItem);

                        if (dados_pedido.guarnicoes.length > 0) {
                                                const guarnicaoItem =
document.createElement('li');
                            guarnicaoItem.classList.add('order-guarnicoes');
                                guarnicaoItem.textContent = `Guarnições:
${dados_pedido.guarnicoes.join(', ')}`;
                            productList.appendChild(guarnicaoItem);
                        }

                        if (dados_pedido.acompanhamentos.length > 0) {
                                                const acompanhamentoItem =
document.createElement('li');
                                acompanhamentoItem.classList.add('order-
acompanhamentos');
                                                acompanhamentoItem.textContent =
`Acompanhamentos: ${dados_pedido.acompanhamentos.join(', ')}`;
                            productList.appendChild(acompanhamentoItem);
                        }
                    });

                    orderDetails.appendChild(productList);
                    orderCard.appendChild(orderDetails);

                    const orderStatus = document.createElement('div');
                    orderStatus.classList.add('order-status');
```

```
                    const totalPreco = typeof dados_pedido.total_preco ===
'number' ? dados_pedido.total_preco : parseFloat(dados_pedido.total_preco) ||
0;

                    // Adiciona uma classe de status com base no status
atual
                                        const statusClass = `status-
${dados_pedido.status.toLowerCase()}`;
                    orderStatus.classList.add(statusClass);

                    orderStatus.innerHTML = `
                                            <span class="order-price">R$
${totalPreco.toFixed(2)}</span>
                                              <span class="order-status-
text">${dados_pedido.status}</span>
                `;
                    orderCard.appendChild(orderStatus);

                    clientDiv.appendChild(orderCard);
                }

                orderList.appendChild(clientDiv);
            }
        }


        // Chama a função para carregar os pedidos quando a página carrega
        document.addEventListener('DOMContentLoaded', fetchOrders);

            // Atualiza a página a cada 2 segundos (2000 milissegundos)
            setInterval(fetchOrders, 2000);
    </script>
    {% endblock %}
```

```
    {% extends "modelo-adm.html" %}
    {% block conteudo %}
    <link rel="stylesheet" href="../static/styles/inicial.css">
    <link                                                rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0-
beta3/css/all.min.css">

    <body>

      <main>
        <section class="container-produtos">

          <h2>Nossos Produtos</h2>
```

```
            <div class="grid-produto">

              <div class="produtos" id="produtos-lista">
                {% for registro in lista_produtos %}
                <div class="produto-item">
                  <form action="/editar_produto" method="post">
                        <button value="{{ registro.id_produto }}" name="btn-
produto" type="submit">
                      <div class="card-produtos" style="justify-content: flex-
start;">
                        <img src="{{ registro.imagem_produto }}" alt="">
                        <span>{{ registro.nome_produto }}</span>
                        <div class="icone-container">
                      </div>
                      </div>
                      {% if registro.habilitado %}
                            <i class="fa fa-check btn-desabilitar" data-id="{{
registro.id_produto }}" title="Desabilitar" style="font-size: 40px; color:
rgb(0, 109, 0);"></i>
                             <i class="fa fa-times btn-habilitar" data-id="{{
registro.id_produto }}" style="display:none; font-size: 40px; color: rgb(187,
0, 0);" title="Habilitar"></i>
                          {% else %}
                             <i class="fa fa-times btn-habilitar" data-id="{{
registro.id_produto }}" title="Habilitar" style="font-size: 40px; color:
rgb(187, 0, 0);"></i>
                           <i class="fa fa-check btn-desabilitar" data-id="{{
registro.id_produto }}" style="display:none; font-size: 40px; color: rgb(0,
109, 0);" title="Desabilitar"></i>
                          {% endif %}
                    </button>
                  </form>
                </div>
                {% endfor %}
              </div>


              <div class="produtos" id="marmitas-lista">
                {% for registro in lista_marmitas %}
                <div class="produto-item">
                  <form action="/editar_marmita" method="post">
                        <button value="{{ registro.id_marmita }}" name="btn-
marmita" type="submit">
                      <div class="card-produtos" style="justify-content: flex-
start;">
                        <img src="{{ registro.imagem_marmita }}" alt="">
                        <span>{{ registro.nome_marmita }}</span>
                      </div>
```

```
                    {% if registro.habilitado %}
                  <i class="fa fa-check btn-desabilitar-marmita" data-id="{{
registro.id_marmita  }}"  title="Desabilitar"  style="font-size:  40px;  color:
rgb(0, 109, 0);"></i>
                   <i class="fa fa-times  btn-habilitar-marmita" data-id="{{
registro.id_marmita }}" style="display:none; font-size: 40px; color: rgb(187,
0, 0);" title="Habilitar"></i>
                {% else %}
                   <i class="fa fa-times btn-habilitar-marmita" data-id="{{
registro.id_marmita  }}"  title="Habilitar"  style="font-size:  40px;  color:
rgb(187, 0, 0);"></i>
                  <i class="fa fa-check btn-desabilitar-marmita" data-id="{{
registro.id_marmita  }}"  style="display:none;  font-size:  40px;  color:  rgb(0,
109, 0);" title="Desabilitar"></i>
                {% endif %}
              {% endfor %}
                  </button>

                </form>


            </div>

          </div>

        </section>

      </main>

      <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
      <script>
    $(document).ready(function() {
      // Função para desabilitar produtos
      $(document).on('click', '.btn-desabilitar', function(e) {
        e.preventDefault();
        const idProduto = $(this).data('id');

        $.ajax({
          url: '/desabilitar_produto_adm',
          method: 'POST',
          data: { btn_desabilitar: idProduto },
          success: function(response) {
            if (response.status === 'success') {
              // Esconde o botão de desabilitar e mostra o de habilitar
              $(`.btn-desabilitar[data-id="${idProduto}"]`).hide();
              $(`.btn-habilitar[data-id="${idProduto}"]`).show();
            }
          },
          error: function(xhr) {
```

```
            console.error('Erro ao desabilitar produto:', xhr.responseText);
          }
        });
      });

      // Função para habilitar produtos
      $(document).on('click', '.btn-habilitar', function(e) {
        e.preventDefault();
        const idProduto = $(this).data('id');

        $.ajax({
          url: '/habilitar_produto_adm',
          method: 'POST',
          data: { btn_habilitar: idProduto },
          success: function(response) {
            if (response.status === 'success') {
              // Esconde o botão de habilitar e mostra o de desabilitar
              $(`.btn-habilitar[data-id="${idProduto}"]`).hide();
              $(`.btn-desabilitar[data-id="${idProduto}"]`).show();
            }
          },
          error: function(xhr) {
            console.error('Erro ao habilitar produto:', xhr.responseText);
          }
        });
      });

      // Função para desabilitar marmitas
      $(document).on('click', '.btn-desabilitar-marmita', function(e) {
        e.preventDefault();
        const idMarmita = $(this).data('id');

        $.ajax({
           url: '/desabilitar_marmita_adm',  // Verifique se essa URL está
correta
          method: 'POST',
          data: { btn_desabilitar: idMarmita },
          success: function(response) {
            if (response.status === 'success') {
              // Esconde o botão de desabilitar e mostra o de habilitar
              $(`.btn-desabilitar-marmita[data-id="${idMarmita}"]`).hide();
              $(`.btn-habilitar-marmita[data-id="${idMarmita}"]`).show();
            }
          },
          error: function(xhr) {
            console.error('Erro ao desabilitar marmita:', xhr.responseText);
          }
        });
      });
```

```
        // Função para habilitar marmitas
        $(document).on('click', '.btn-habilitar-marmita', function(e) {
          e.preventDefault();
          const idMarmita = $(this).data('id');

          $.ajax({
              url:  '/habilitar_marmita_adm',  // Verifique se essa URL está
correta
            method: 'POST',
            data: { btn_habilitar: idMarmita },
            success: function(response) {
              if (response.status === 'success') {
                // Esconde o botão de habilitar e mostra o de desabilitar
                $(`.btn-habilitar-marmita[data-id="${idMarmita}"]`).hide();
                $(`.btn-desabilitar-marmita[data-id="${idMarmita}"]`).show();
              }
            },
            error: function(xhr) {
              console.error('Erro ao habilitar marmita:', xhr.responseText);
            }
          });
        });
      });
        </script>

      {% endblock %}
```

```
      {% extends "modelo.html" %}
      {% block conteudo %}
      <div class="central" style="display: {% if not site_aberto %} flex {%
else %} none {% endif %};">
          <div  id="horario"  class="horario"  style="display:  {%  if  not
site_aberto %} block {% else %} none {% endif %};">
          <h2>Site offline.</h2>
        </div>
      </div>

      <div class="popup-container" style="display: {% if success %} block {%
else %} none {% endif %};">
        <div class="popup success-popup">
          <div class="popup-icon success-icon">
              <svg  xmlns="http://www.w3.org/2000/svg"  viewBox="0  0  24  24"
class="success-svg">
              <path fill-rule="evenodd"
```

```
                d="m12 1c-6.075 0-11 4.925-11 11s4.925 11 11 11 11-4.925 11-
11-4.925-11-11-11zm4.768        9.14c.0878-.1004.1546-.21726.1966-.34383.0419-
.12657.0581-.26026.0477-.39319-.0105-.13293-.0475-.26242-.1087-.38085-.0613-
.11844-.1456-.22342-.2481-.30879-.1024-.08536-.2209-.14938-.3484-.18828s-
.2616-.0519-.3942-.03823c-.1327.01366-.2612.05372-.3782.1178-.1169.06409-
.2198.15091-.3027.25537l-4.3   5.159-2.225-2.226c-.1886-.1822-.4412-.283-.7034-
.2807s-.51301.1075-.69842.2929-.29058.4362-.29285.6984c-
.00228.2622.09851.5148.28067.703413
3c.0983.0982.2159.1748.3454.2251.1295.0502.2681.0729.4069.0665.1387-
.0063.2747-.0414.3991-.1032.1244-.0617.2347-.1487.3236-.2554z"
               clip-rule="evenodd"></path>
            </svg>
          </div>
          <div class="success-message">Login realizado com sucesso!</div>
        </div>
      </div>

      <main>
        <section class="carrossel">
            <div  id="carouselExampleControls"  class="carousel  slide"  data-
ride="carousel">
            <div class="carousel-inner">
              <div class="carousel-item active">
                     <img  class="d-block  w-100"  src="/static/img/test.png"
alt="Primeiro Slide">
              </div>
              <div class="carousel-item">
                    <img  class="d-block  w-100"  src="/static/img/test2.png"
alt="Segundo Slide">
              </div>
              <div class="carousel-item">
                    <img  class="d-block  w-100"  src="/static/img/test3.png"
alt="Terceiro Slide">
              </div>
            </div>
             <a  class="carousel-control-prev"  href="#carouselExampleControls"
role="button" data-slide="prev">
                          <span    class="carousel-control-prev-icon"    aria-
hidden="true"></span>
              <span class="sr-only">Anterior</span>
            </a>
             <a  class="carousel-control-next"  href="#carouselExampleControls"
role="button" data-slide="next">
                          <span    class="carousel-control-next-icon"    aria-
hidden="true"></span>
              <span class="sr-only">Próximo</span>
            </a>
          </div>
        </section>
```

```
        <!-- Seção de Produtos -->
        <section class="container-produtos">
          <!-- Seção de Produtos -->
          <h2>Nossos Produtos</h2>
          <section class="container-produtos">

            <div class="grid-produto">
              <div class="produtos" id="produtos-list">
                <!-- Produtos renderizados dinamicamente aqui -->
              </div>
            </div>
          </section>


          <!-- Seção de Marmitas -->
          <section class="container-produtos">
              <h3 style="margin-top: 1rem !important; margin-bottom: 1rem
!important; font-size: 1.75rem;">Marmitex</h3>

            <div class="grid-produto">
              <div class="produtos" id="marmitas-list">
                {% for marmita in lista_marmitas %}
                <form action="/marmita_unica" method="post">
                   <button value="{{ marmita.id_marmita }}" name="btn-produto"
type="submit">
                    <div class="card-produtos">
                      <div class="left-lado">
                            <span class="produto-nome">{{ marmita.nome_marmita
}}</span>
                              <span class="produto-desc">{{ marmita.descricao
}}</span>
                              <span class="produto-preco">R$ {{ marmita.preco
}}</span>
                      </div>
                      <div class="right-lado">
                        {% if marmita.img_marmita is string %}
                        <!-- Se for base64, insere diretamente -->
                                        <img src="{{
marmita.img_marmita }}" alt="{{ marmita.nome_marmita }}">
                        {% else %}
                        <!-- Se for URL, insere a URL da imagem -->
                                <img src="{{ marmita.img_marmita }}" alt="{{
marmita.nome_marmita }}">
                        {% endif %}
                      </div>
                    </div>
                  </button>
```

```
                </form>
                {% endfor %}
              </div>
            </div>
          </section>

      </main>

      <!-- <footer>
        <a href="/exibir_carrinho" class="car-icon">
          <img src="https://cdn-icons-png.flaticon.com/512/4/4295.png" alt="">
        </a>
      </footer> -->


      <!-- Scripts do jQuery -->
      <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
      <script>
        $(document).ready(function () {
          // Se o popup for exibido, defina um temporizador para escondê-lo
após 3 segundos
          if ($('.popup-container').is(':visible')) {
            setTimeout(function () {
              $('.popup-container').fadeOut();
            }, 3000); // Esconde após 3000ms (3 segundos)
          }
        });

        function atualizarProdutos() {
          $.ajax({
              url: '/produtos_json',  // URL da rota que retorna os produtos
habilitados em JSON
            method: 'GET',
            success: function (data) {
              const produtosContainer = $('#produtos-list');
                produtosContainer.empty();  // Limpa a lista existente para
evitar duplicação

              // Cria um objeto para agrupar produtos por categoria
              const produtosPorCategoria = {};

              // Organiza os produtos por categoria
              data.forEach(function (produto) {
                const categoria = produto.nome_categoria;

                if (!produtosPorCategoria[categoria]) {
                  produtosPorCategoria[categoria] = [];
                }
                produtosPorCategoria[categoria].push(produto);
```

```
            });

            // Itera sobre as categorias e insere os produtos agrupados
            Object.keys(produtosPorCategoria).forEach(function (categoria) {
              produtosContainer.append(`<h3>${categoria}</h3>`);

              produtosPorCategoria[categoria].forEach(function (produto) {
                produtosContainer.append(`
                    <form action="/produto_unico" method="post">
                           <button  value="${produto.id_produto}"  name="btn-
produto" type="submit">
                        <div class="card-produtos">
                          <div class="left-lado">
                                                        <span   class="produto-
nome">${produto.nome_produto}</span>
                                                        <span   class="produto-
desc">${produto.descricao}</span>
                                                <span   class="produto-preco">R$
${produto.preco}</span>
                          </div>
                          <div class="right-lado">
                                          <img  src="${produto.imagem_produto}"
alt="${produto.nome_produto}">
                          </div>
                        </div>
                      </button>
                    </form>
                  `);
              });
            });
          },
          error: function (error) {
            console.error("Erro ao carregar produtos:", error);
          }
        });
      }

      function atualizarMarmitas() {
        $.ajax({
           url: '/marmitas_json',  // URL da rota que retorna as marmitas
habilitadas em JSON
          method: 'GET',
          success: function (data) {
            const marmitasContainer = $('#marmitas-list');
              marmitasContainer.empty();  // Limpa a lista existente para
evitar duplicação

            if (data && typeof data === 'object') {
              // Itera sobre os tamanhos e adiciona marmitas de cada tamanho
```

```
                Object.keys(data).forEach(function (tamanho) {
                  const marmitas = data[tamanho].marmitas;
                  if (Array.isArray(marmitas)) {
                    marmitas.forEach(function (marmita) {
                      marmitasContainer.append(`
                        <form action="/marmita_unica" method="post">
                              <button  value="${marmita.id_marmita}"  name="btn-
produto" type="submit">
                            <div class="card-produtos">
                              <div class="left-lado">
                                                          <span   class="produto-
nome">${marmita.nome_marmita}</span>
                                                          <span   class="produto-
desc">${marmita.descricao}</span>
                                                  <span  class="produto-preco">R$
${marmita.preco}</span>
                              </div>
                              <div class="right-lado">
                                        <img  src="${marmita.imagem_marmita}"
alt="${marmita.nome_marmita}">
                              </div>
                            </div>
                          </button>
                        </form>
                      `);
                    });
                  }
                });
              } else {
                  console.error("Erro:  Dados  não  são  um  objeto  esperado.",
data);
              }
            },
            error: function (error) {
              console.error("Erro ao carregar marmitas:", error);
            }
          });
        }



        // Chama as funções na primeira carga da página
        $(document).ready(function () {
          atualizarProdutos();
          atualizarMarmitas();
```

```
        // Atualiza os produtos e marmitas a cada 30 segundos (30000 ms)
        setInterval(function () {
          atualizarProdutos();
          atualizarMarmitas();
        }, 2000);  // Ajuste o tempo conforme necessário
      });
    </script>
    {% endblock %}
```

```
    {% extends "modelo-adm.html" %}
    {% block conteudo %}

    <head>
        <style>
            /* Estilo padrão (mobile-first) */
            .container {
                margin-top: 35px;
                display: flex;
                flex-direction: column; /* Empilha os cards verticalmente */
                align-items: center;
                justify-content: center;
                gap: 15px; /* Espaço entre os cards */
                width: 100%; /* Largura total */
                margin-bottom: 20px;
            }

            .card {
                background-color: white;
                border-radius: 15px;
                box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);
                overflow: hidden; /* Evita que a imagem transborde */
                text-align: center;
                width: 100%; /* Ajusta para largura total do container */
                max-width: 350px; /* Largura máxima */
                text-decoration: none; /* Remove sublinhado do link */
                color: inherit; /* Usa a cor do texto pai */
                display: flex; /* Usar flexbox para o alinhamento interno */
                flex-direction: column; /* Empilha a imagem e o texto */
                justify-content: center; /* Centraliza verticalmente */
            }

            .card img {
                width: 100%; /* A imagem ocupa toda a largura do card */
                height: 120px; /* Altura da imagem */
                object-fit: cover; /* Cobre o espaço da imagem sem distorcer
*/
            }
```

```
            .card p {
                font-weight: 650;
                padding: 10px; /* Espaçamento interno do texto */
                margin: 0; /* Remove a margem padrão */
                font-size: 14px; /* Tamanho da fonte */
                color: #333; /* Cor do texto */
            }

            .card:hover {
                 background-color: #f9f9f9; /* Muda a cor ao passar o mouse
*/
                  box-shadow: 0 4px 15px rgba(0, 0, 0, 0.2); /* Aumenta a
sombra ao passar o mouse */
                text-decoration: none;
            }

        </style>
        <link rel="stylesheet" href="/Projeto TCC/static/styles/media/media-
inicialAdm.css">
    </head>

    <body>
        <div class="container">
            <a class="card" href="/exibir_pedidos">
                <img src="../static/img/delivery.jpg" alt="Pedidos">
                <p>Pedidos</p>
            </a>
            <a class="card" href="/exibir_guarnicao">
                <img src="../static/img/coxinhadefanho.png" alt="Adicionar">
                <p>Adicionar Produto</p>
            </a>
            <a class="card" href="/adm">
                <img src="../static/img/hamburguin.jpg" alt="Editar">
                <p>Editar Produtos</p>
            </a>
            <a class="card" href="/relatorio">
                <img src="../static/img/cantinaComanda.jpg" alt="Editar">
                <p>Relatório</p>
            </a>
        </div>

        {% endblock %}
    </body>

    </html>
```

```
      {% extends "modelo.html" %}
      {% block conteudo %}
      <link rel="stylesheet" href="../static/styles/login.css">

      {% if erro %}
      <div class="popup-container" style="display: flex;">
          <div class="popup alert-popup">
              <div class="popup-icon alert-icon">
                    <svg  class="alert-svg"  xmlns="http://www.w3.org/2000/svg"
viewBox="0 0 20 20" aria-hidden="true">
                     <path fill-rule="evenodd"
                           d="M8.257  3.099c.765-1.36  2.722-1.36  3.486  0l5.58
9.92c.75  1.334-.213  2.98-1.742  2.98H4.42c-1.53  0-2.493-1.646-1.743-2.98l5.58-
9.92zM11  13a1  1  0  11-2  0  1  1  0  012  0zm-1-8a1  1  0  00-1  1v3a1  1  0  002  0V6a1  1  0
00-1-1z"
                          clip-rule="evenodd"></path>
                </svg>
              </div>
              <div class="alert-message">Login ou senha inválidos!</div>
          </div>
      </div>
      {% endif %}

      <body>

          <main>
              <div class="btn-voltar">
                  <button onclick="window.history.back()"><img
                          src="https://www.svgrepo.com/show/324205/back-arrow-
navigation.svg" alt=""></button>
              </div>

              <div class="titulo">
                  <h2>Entrar</h2>
              </div>
              <section class="container-principal">

                  <div class="formulario">

                      <form action="" method="post">
                          <input name="email" id="email" placeholder="*E-mail"
type="text" required>
                           <input name="senha" id="senha" placeholder="*Senha"
type="password" required>
                                  <a  href="/trocar_senha"><span>Esqueceu  sua
senha</span></a>
                          <button type="submit">Entrar</button>
                      </form>
```

```
                </div>

                        <span  class="semConta">Ainda  não  tem  conta?<a
href="/cadastro"> Crie Aqui!</a></span>

            </section>

            {% endblock %}

        </main>

    </body>

    <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>

    <script>
        $(document).ready(function () {
            // Se o popup for exibido, defina um temporizador para escondê-
lo após 3 segundos
            if ($('.popup-container').is(':visible')) {
                setTimeout(function () {
                    $('.popup-container').fadeOut(); // Usamos fadeOut para
uma transição suave
                }, 3000); // Esconde após 3000ms (3 segundos)
            }
        });
    </script>

    </html>
```

```
    {% extends "modelo.html" %}
    {% block conteudo %}
    <link rel="stylesheet" href="../static/styles/marmita.css">
    <link rel="stylesheet" href="../static/styles/media/media-marmita.css">

    <body>
        <main>
            <div id="horario-popup" class="popup-container" style="display:
none;">
                <div class="popup alert-popup">
                    <div class="popup-icon alert-icon">
                                                <svg    class="alert-svg"
xmlns="http://www.w3.org/2000/svg" viewBox="0 0 20 20" aria-hidden="true">
                            <path fill-rule="evenodd"
                                d="M8.257 3.099c.765-1.36 2.722-1.36 3.486
0l5.58  9.92c.75  1.334-.213  2.98-1.742  2.98H4.42c-1.53  0-2.493-1.646-1.743-
```

```
2.98l5.58-9.92zM11 13a1 1 0 11-2 0 1 1 0 012 0zm-1-8a1 1 0 00-1 1v3a1 1 0 002
0V6a1 1 0 00-1-1z"
                                clip-rule="evenodd"></path>
                        </svg>
                    </div>
                       <div class="alert-message">Os pedidos de marmitas só
podem ser feitos entre 7h e 15h30!</div>
                </div>
            </div>


            <section class="posicao-marmita">

                <div class="produto">


                    <div class="foto-produto">
                         <img class="ft-item" src="{{ marmita.imagem_marmita
}}" alt="Imagem da Marmita">
                    </div>


                    <section class="descricao">
                          <span class="nome-produto">{{ marmita.nome_marmita
}}</span>
                        <span class="descricao-produto">{{ marmita.descricao
}}</span>
                            <span class="preco-produto">R$ {{ marmita.preco
}}</span>

                        <div class="opc">
                                 <form id="form-marmita" class="formulario-
marmita">
                                     <input type="hidden" name="id_marmita"
value="{{ marmita.id_marmita }}" />
                                <!-- ID da marmita -->

                                <!-- Campos para Acompanhamentos -->
                                     {% if marmita.acompanhamentos and
marmita.acompanhamentos|length > 0 %}
                                <div class="acompanhamentos">
                                    <div class="header">
                                        <div class="titulosubtitulo">
                                            <h3>Acompanhamentos</h3>
                                                    <p id="acompanhamento-
info">Escolha até <span id="max-acompanhamentos"></span>
                                                    acompanhamentos</p>
                                        </div>
```

```
                                                    <span  class="seta"
onclick="toggleAcompanhamentos(this)">&#9660;</span>
                                        </div>
                                              <div class="opcoes-acompanhamentos
hidden">
                                                        {% for acompanhamento in
marmita.acompanhamentos %}
                                            <div class="acompanhamento">
                                                                        <img
src="https://pedido.anota.ai/assets/item_no_image-a8c57261.png"
                                                    alt="{{ acompanhamento.nome
}}" />
                                                <span>{{ acompanhamento.nome
}}</span>
                                            <label>
                                                        <input type="checkbox"
name="acompanhamento"
                                                                        value="{{
acompanhamento.id }}" />
                                                <span></span>
                                            </label>
                                        </div>
                                        {% endfor %}
                                    </div>
                                </div>
                                {% endif %}

                                <!-- Campos para Guarnições -->
                                            {% if marmita.guarnicoes and
marmita.guarnicoes|length > 0 %}
                                <div class="acompanhamentos">
                                    <div class="header">
                                        <div class="titulosubtitulo">
                                            <h3>Guarnições</h3>
                                              <p id="guarnicao-info">Escolha
até <span id="max-guarnicoes"></span> guarnições
                                            </p>
                                        </div>
                                                    <span  class="seta"
onclick="toggleGuarnicoes(this)">&#9660;</span>
                                    </div>
                                    <div class="opcoes-guarnicoes hidden">
                                                        {% for guarnicao in
marmita.guarnicoes %}
                                        <div class="acompanhamento">
                                                                        <img
src="https://pedido.anota.ai/assets/item_no_image-a8c57261.png"
                                                    alt="{{ guarnicao.nome }}"
/>
```

```
                                                    <span>{{ guarnicao.nome
}}</span>
                                                <label>
                                                    <input type="checkbox"
name="guarnicao" value="{{ guarnicao.id }}" />
                                                    <span></span>
                                                </label>
                                            </div>
                                            {% endfor %}
                                        </div>
                                    </div>
                                    {% endif %}

                                    <!-- Botões no final -->

                                </form>

                            </div>

                        </section>
                        <div class="sair">
                            <a href="/" class="btn-voltar">Voltar</a>
                                                    <button    type="button"
id="adicionarCarrinhoBtn">Adicionar ào
                                Carrinho</button>
                        </div>
                    </div>
                </section>
            </main>
        </body>

        <div class="espacamento1"></div>

        <script>
            // Função para definir os limites com base no tamanho da marmita
            function setLimits() {
                 const tamanhoMarmita = "{{ marmita.tamanho }}"; // Obtenha o
tamanho da marmita
                let maxAcompanhamentos, maxGuarnicoes;

                switch (tamanhoMarmita) {
                    case 'Pequena':
                        maxAcompanhamentos = 2;
                        maxGuarnicoes = 1;
                        break;
                    case 'Média':
                        maxAcompanhamentos = 3;
                        maxGuarnicoes = 1;
                        break;
```

```
                case 'Grande':
                    maxAcompanhamentos = 3;
                    maxGuarnicoes = 2;
                    break;
                default:
                    maxAcompanhamentos = 0;
                    maxGuarnicoes = 0;
            }

                document.getElementById('max-acompanhamentos').textContent  =
maxAcompanhamentos;
                    document.getElementById('max-guarnicoes').textContent   =
maxGuarnicoes;

            return { maxAcompanhamentos, maxGuarnicoes };
        }

        // Função para limitar o número de acompanhamentos e guarnições
        function limitSelection(checkboxes, max, alertMessage) {
            checkboxes.forEach(checkbox => {
                checkbox.addEventListener('change', () => {
                     const checkedCount = Array.from(checkboxes).filter(i =>
i.checked).length;
                    if (checkedCount > max) {
                        checkbox.checked = false;
                        alert(alertMessage);
                    }
                });
            });
        }

        document.addEventListener('DOMContentLoaded', function () {
            const { maxAcompanhamentos, maxGuarnicoes } = setLimits();

                                    const      checkboxesAcompanhamento      =
document.querySelectorAll('input[name="acompanhamento"]');
                                        const       checkboxesGuarnicao       =
document.querySelectorAll('input[name="guarnicao"]');

                limitSelection(checkboxesAcompanhamento,  maxAcompanhamentos,
`Você só pode selecionar até ${maxAcompanhamentos} acompanhamentos.`);
            limitSelection(checkboxesGuarnicao, maxGuarnicoes, `Você só pode
selecionar até ${maxGuarnicoes} guarnições.`);

            // Evento para adicionar ao carrinho com fetch()

document.getElementById('adicionarCarrinhoBtn').addEventListener('click',
function (event) {
                event.preventDefault();
```

```
                const form = document.getElementById('form-marmita');
                const formData = new FormData(form);

                fetch('/inserir_carrinho', {
                    method: 'POST',
                    body: formData
                })
                    .then(response => response.json())
                    .then(data => {
                        if (!data.success) {
                                if (data.error && data.error.includes("Os
pedidos de marmitas só podem ser feitos")) {
                                // Mostra o popup de erro de horário
                                                const popup =
document.getElementById('horario-popup');
                                popup.style.display = 'block';

                                // Esconde o popup após 5 segundos
                                setTimeout(() => {
                                    popup.style.display = 'none';
                                }, 5000);
                            } else if (!data.is_logged_in) {
                                window.location.href = "/logar";
                            } else if (data.error) {
                                alert(data.error);
                            }
                        } else {
                            window.location.href = data.redirect_url;
                        }
                    })
                    .catch(error => {
                             console.error('Erro ao inserir no carrinho:',
error);
                    });
            });
        });


        function toggleAcompanhamentos(element) {
                                            const opcoes =
element.closest('.acompanhamentos').querySelector('.opcoes-acompanhamentos');
            opcoes.classList.toggle('hidden');
        }

        function toggleGuarnicoes(element) {
                                            const opcoes =
element.closest('.acompanhamentos').querySelector('.opcoes-guarnicoes');
            opcoes.classList.toggle('hidden');
        }
```

```
    </script>
    {% endblock %}
```

```
    <!DOCTYPE html>
    <html lang="pt-br">
    <head>
        <meta charset="UTF-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
            <meta   name="viewport"   content="width=device-width,   initial-
scale=1.0">
        <title>Cantina Digital</title>

        <link rel="stylesheet" href="../static/styles/inicial.css">

            <script   src="https://code.jquery.com/jquery-3.3.1.slim.min.js"
integrity="sha384-
q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo"
crossorigin="anonymous"></script>
                                                                    <script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.3/umd/popper.min.js
"                                                                integrity="sha384-
ZMP7rVo3mIykV+2+9J3UJ46jBk0WLaUAdn689aCwoqbBJiSnjAK/l8WvCWPIPm49"
crossorigin="anonymous"></script>
                                                                    <script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/js/bootstrap.min.js"
integrity="sha384-
ChfqqxuZUCnJSK3+MXmPNIyE6ZbWh2IMqE241rYiqJxyMiZ6OW/JmZQ5stwEULTy"
crossorigin="anonymous"></script>
                                        <link                rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css
"                                                                integrity="sha384-
MCw98/SFnGE8fJT3GXwEOngsV7Zt27NXFoaoApmYm81iuXoPkFOJwJ8ERdknLPMO"
crossorigin="anonymous">
                    <script     src="https://kit.fontawesome.com/ded267d5dc.js"
crossorigin="anonymous"></script>
    </head>
    <body>
        <header class="container-header">

            <nav class="cbc-cima">


                    <div class="perfil-icon">
                        <a href="/perfil">
                            {% if session.get('usuario_logado') %}
                                <img class="perfil"
```

```
                                        src="{{ url_for('imagem_perfil',
id_cliente=session['usuario_logado']['id_cliente']) }}"
                                    alt="Imagem do Perfil"
                                    width="150"
                                    height="150"

onerror="this.onerror=null;this.src='../static/img/default-avatar.png';">
                                {% else %}
                                    <!-- Se o usuário não estiver logado, exibe
uma imagem padrão -->
                                <img class="perfil"
                                    src="../static/img/default-avatar.png"
                                    alt="Imagem do Perfil"
                                    width="150"
                                    height="150">
                                {% endif %}
                            </a>
                        </div>



                        <a class="logo-lu" href="/inicialadm"><figure>
                                 <img  class="logo"  src="../static/img/logo.png"
alt="">
                        </figure></a>

                    <div class="menu-container" onclick="toggleMenu(this)">
                        <div class="bar1"></div>
                        <div class="bar2"></div>
                        <div class="bar3"></div>
                    </div>

                    <div id="mySidenav" class="sidenav">
                          <a href="/" class="nav-button"><i class="fa-solid fa-
house" style="color: #FFD43B;"></i>Início</a>
                        <hr>
                            <a href="/exibir_guarnicao"><i class="fa-solid fa-
circle-plus" style="color: #FFD43B;"></i>Adicionar Produto</a>
                        <hr>
                          <a href="/exibir_pedidos"><i class="fa-solid fa-list"
style="color: #FFD43B;"></i>Pedidos</a>
                        <hr>
                            <a href="/adm"><i class="fa-solid fa-pen-to-square"
style="color: #FFD43B;"></i>Editar Produto</a>
                        <hr>
                            <a href="/relatorio"><i class="fa-regular fa-paste"
style="color: #FFD43B;"></i>Relatorio</a>
                        <hr>
```

```
                        <a href="/logout"><i class="fa-solid fa-arrow-right-
from-bracket" style="color: #FFD43B;"></i>Sair</a>

                    </div>
                </nav>
                <div class="espacamento">
                    <span>espaço</span>
                </div>
            </header>

            <!-- o conteúdo do site será inserido abaixo -->
            {% block conteudo %}
            {% endblock %}

            <script>
                function toggleMenu(x) {
                    // Alternar animação das barras
                    x.classList.toggle("change");

                    // Abrir ou fechar o menu lateral
                    const menu = document.getElementById("mySidenav");
                    const test = document.getElementById("profile-container")
                    if (menu.style.width === "100%") {
                        menu.style.width = "0";
                    } else {
                        menu.style.width = "100%";
                    }
                }

                const chk = document.getElementById('chk')
                const inputs = document.querySelectorAll('input')
                chk.addEventListener('change', () => {
                    document.body.classList.toggle('dark')
                    document.getElementById('form').classList.toggle('dark1')
                    inputs.forEach(input => {
                         input.classList.toggle('dark2') // Adiciona ou remove a
classe 'dark1' em cada input
                    })
                })
            </script>


        </main>

        </body>
        </html>
        <!DOCTYPE html>
        <html lang="pt-br">
```

```
      <head>
          <meta charset="UTF-8">
          <meta http-equiv="X-UA-Compatible" content="IE=edge">
              <meta   name="viewport"   content="width=device-width,   initial-
scale=1.0">
          <title>Cantina Digital</title>
          <link rel="stylesheet" href="../static/styles/inicial.css">
               <link    rel="stylesheet"    href="../static/styles/media/media-
inicial.css">

          <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
          <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"
                                                                integrity="sha384-
q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo"
              crossorigin="anonymous"></script>
                                                                      <script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.3/umd/popper.min.js
"
                                                                integrity="sha384-
ZMP7rVo3mIykV+2+9J3UJ46jBk0WLaUAdn689aCwoqbBJiSnjAK/l8WvCWPIPm49"
              crossorigin="anonymous"></script>
                                                                      <script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/js/bootstrap.min.js"
                                                                integrity="sha384-
ChfqqxuZUCnJSK3+MXmPNIyE6ZbWh2IMqE241rYiqJxyMiZ6OW/JmZQ5stwEULTy"
              crossorigin="anonymous"></script>
                                          <link                rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css
"
                                                                integrity="sha384-
MCw98/SFnGE8fJT3GXwEOngsV7Zt27NXFoaoApmYm81iuXoPkFOJwJ8ERdknLPMO"
crossorigin="anonymous">

                   <script      src="https://kit.fontawesome.com/ded267d5dc.js"
crossorigin="anonymous"></script>

      </head>

      <body>
          <header class="container-header">
              <nav class="cbc-cima">
                  <div class="perfil-icon">
                      <a href="/perfil">
                          {% if session.get('usuario_logado') %}
                          <img class="perfil"
                                                  src="{{  url_for('imagem_perfil',
id_cliente=session['usuario_logado']['id_cliente']) }}"
                               alt="Imagem do Perfil" width="150" height="150"
```

```
onerror="this.onerror=null;this.src='../static/img/default-avatar.png';">
                        {% else %}
                            <img class="perfil" src="../static/img/default-
avatar.png" alt="Imagem do Perfil" width="150"
                            height="150">
                        {% endif %}
                    </a>
                </div>
                <a class="logo-lu" href="/">
                    <figure>
                            <img class="logo" src="../static/img/logo.png"
alt="Logo da Cantina Digital">
                    </figure>
                </a>
                <div class="menu-container" onclick="toggleMenu(this)">
                    <div class="bar1"></div>
                    <div class="bar2"></div>
                    <div class="bar3"></div>
                </div>
                <div id="mySidenav" class="sidenav">
                    {% if not session.get('usuario_logado') %}
                      <a href="/logar" class="nav-button"><i class="fa-solid
fa-right-to-bracket"></i>Entrar</a>
                    <hr>
                    {% endif %}
                      <a href="/" class="nav-button"><i class="fa-solid fa-
house"></i>Início</a>
                    <hr>
                     <a href="/perfil" class="nav-button"><i class="fa-solid
fa-user"></i>Perfil</a>
                    <hr>
                          <a href="/exibir_carrinho" class="nav-button"><i
class="fa-solid fa-cart-shopping"></i>Carrinho</a>
                    <hr>
                    {% if session.get('usuario_logado') %}
                       <a href="/historico" class="nav-button"><i class="fa-
solid fa-clock-rotate-left"></i>Pedidos</a>
                    <hr>
                     <a href="/logout" class="nav-button"><i class="fa-solid
fa-arrow-right-from-bracket"></i>Sair</a>
                    {% endif %}
                </div>

            </nav>
            <div class="espacamento">

            </div>
        </header>
```

```
        <!-- O conteúdo do site será inserido abaixo -->
        {% block conteudo %}
        {% endblock %}

        <script>
            function toggleMenu(x) {
                // Alternar animação das barras
                x.classList.toggle("change");

                // Abrir ou fechar o menu lateral
                const menu = document.getElementById("mySidenav");
                if (menu.style.width === "400px") {
                    menu.style.width = "0";
                } else {
                    menu.style.width = "400px";
                }
            }

            const chk = document.getElementById('chk');
            const inputs = document.querySelectorAll('input');
            chk.addEventListener('change', () => {
                document.body.classList.toggle('dark');
                inputs.forEach(input => {
                    input.classList.toggle('dark2');
                });
            });

        </script>
    </body>

    </html>
```

```
    <!DOCTYPE html>
    <html lang="pt-br">

    <head>
        <meta charset="UTF-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
            <meta   name="viewport"   content="width=device-width,   initial-
scale=1.0">
        <title>Trocar Senha</title>
                <link     rel="stylesheet"     href="{{     url_for('static',
filename='styles/trocar-senha.css') }}">
    </head>
```

```
      <body>
          <main>
              <div class="btn-voltar">
                  <button onclick="window.history.back()">
                          <img  src="https://www.svgrepo.com/show/324205/back-
arrow-navigation.svg" alt="Voltar">
                  </button>
              </div>

              <div class="titulo">
                  <h2>Defina sua nova senha</h2>
              </div>

              <section class="container-principal">
                  <div class="formulario">
                      <form method="POST" action="/nova_senha">
                            <input name="nova_senha" placeholder="*Nova Senha"
type="password" required>
                                                <input   name="nova_senha_confirma"
placeholder="*Confirme a senha" type="password" required>
                           <button type="submit">Trocar Senha</button>
                      </form>
                  </div>
              </section>
          </main>
      </body>

      </html>
```

```
      {% extends "modelo.html" %}
      {% block conteudo %}
      <link rel="stylesheet" href="../static/styles/perfil.css">
      <body>
          <main>
              <section>
                              <form   action="/atualizar_perfil"   method="post"
enctype="multipart/form-data" class="container-principal">
                      <h2>Perfil</h2>

                      <div class="profile-container" id="profile-container">
                          <div class="profile-image">
                              <!-- Pré-visualização da imagem do perfil -->
                                                  <img   id="profile-img"   src="{{
url_for('imagem_perfil',    id_cliente=session['usuario_logado']['id_cliente'])
}}"       alt="Imagem       do       Perfil"       width="150"       height="150"
onerror="this.src='../static/img/default-avatar.png';">
```

```
                        </div>

                        <div class="overlay">
                            <!-- Input para seleção da nova imagem -->
                                        <input  type="file"  id="file-upload"
name="imagem_perfil"               accept="image/png,              image/jpeg"
onchange="previewImage(event)">
                                    <img  src="../static/img/btn-editar.png"
alt="Imagem Sobreposta" class="overlay-image">
                        </div>
                    </div>

                    <div class="formulario">
                        <div class="form">
                              <input placeholder="{{ perfil_usuario.nome }}"
type="text" name="nome" value="{{ session['usuario_logado']['nome_comp'] }}">

                                        <input  type="password"  id="password"
name="senha" placeholder="*Senha" required>
                                    <button type="button" class="toggle-button"
onclick="togglePassword1(event)">
                                                          <img   id="toggle-icon"
src="../static/img/fecheido-removebg.png"   alt="Mostrar   Senha"   width="20"
height="20">
                                </button>

                                       <input  placeholder="*Confirmar  Senha"
id="passwordConfirmar" name="confirmar_senha" type="password" required>
                                  <button type="button" class="toggle-button2"
onclick="togglePassword2(event)">
                                                         <img   id="toggle-icon2"
src="../static/img/fecheido-removebg.png"   alt="Mostrar   Senha"   width="20"
height="20">
                                </button>

                                              <button   class="btn-confirmar"
type="submit">Confirmar Mudanças</button>
                        </div>
                    </div>
                </form>
            </section>
        </main>

        <!-- Script para pré-visualizar a imagem selecionada -->
        <script>
            function previewImage(event) {
                var input = event.target;  // O input file
                var reader = new FileReader();
```

```
                reader.onload = function() {
                    var dataURL = reader.result;  // O resultado será o URL da imagem
                    var imgElement = document.getElementById('profile-img');  // Seleciona a tag img
                    imgElement.src = dataURL;  // Define o src da imagem como o novo URL gerado
                };

                if (input.files && input.files[0]) {
                    reader.readAsDataURL(input.files[0]);  // Lê a imagem selecionada e chama o onload
                }
            }

            // Funções para exibir e ocultar senha
            function togglePassword1(event) {
                event.preventDefault();
                const passwordField = document.getElementById('password');
                const toggleIcon = document.getElementById('toggle-icon');

                passwordField.type = passwordField.type === 'password' ? 'text' : 'password';
                toggleIcon.src = passwordField.type === 'text' ? './static/img/abrido-removebg.png' : './static/img/fecheido-removebg.png';
                toggleIcon.alt = passwordField.type === 'text' ? 'Ocultar Senha' : 'Mostrar Senha';
            }

            function togglePassword2(event) {
                event.preventDefault();
                const passwordField = document.getElementById('passwordConfirmar');
                const toggleIcon = document.getElementById('toggle-icon2');

                passwordField.type = passwordField.type === 'password' ? 'text' : 'password';
                toggleIcon.src = passwordField.type === 'text' ? './static/img/abrido-removebg.png' : './static/img/fecheido-removebg.png';
                toggleIcon.alt = passwordField.type === 'text' ? 'Ocultar Senha' : 'Mostrar Senha';
            }
        </script>
    </body>
    {% endblock %}
```

```
    {% extends "modelo.html" %}
```

```
      {% block conteudo %}
      <link rel="stylesheet" href="../static/styles/produto.css">
      <link rel="stylesheet" href="../static/styles/media/media-produtos.css">

      <body>
          <main>
              {% for registro in lista_prounico %}
              <section class="posicao-produto">
                  <div class="produto">
                      <div class="foto-produto">
                          <img class="ft-item" src="{{ registro.imagem_produto
}}" alt="Imagem do produto">
                      </div>
                      <section class="descricao">
                           <span class="nome-produto">{{ registro.nome_produto
}}</span>
                                                <span   class="descricao-produto">{{
registro.descricao }}</span>
                                <span class="preco-produto">R$ {{ registro.preco
}}</span>


                          <div class="sair">
                              <a href="/" class="btn-voltar">Voltar</a>

                                                        <form   id="carrinhoForm"
action="/inserir_carrinho" method="post">
                                              <input type="hidden" name="cod_produto"
value="{{ registro.cod_produto }}">
                                                 <button type="submit" class="btn-
adicionar">Adicionar ao Carrinho</button>
                                </form>
                          </div>
                      </section>
                  </div>
              </section>



              <div class="espacamento"></div>

              <script>

document.getElementById('carrinhoForm').addEventListener('submit',       async
function (event) {
                          event.preventDefault(); // Impede o envio padrão do
formulário

                      const formData = new FormData(this);
```

```
                    try {
                        const response = await fetch(this.action, {
                            method: 'POST',
                            body: formData
                        });

                        if (response.status === 401) {
                            // Redireciona para a página de login em caso de
status 401
                            const result = await response.json();
                            window.location.href = result.redirect_url;
                            return;
                        }

                        const result = await response.json();

                        if (result.success) {
                            // Redireciona para a página do carrinho
                            window.location.href = result.redirect_url;
                        }
                    } catch (error) {
                            console.error('Erro ao enviar o formulário:',
error);
                    }
                });


                const input = document.getElementById('meuInput');
                const contador = document.getElementById('contador');

                input.addEventListener('input', () => {
                    contador.textContent = input.value.length; // Atualiza o
número de caracteres digitados
                });
            </script>
            {% endfor %}
        </main>
        {% endblock %}
```

```
    {% extends "modelo-adm.html" %}
    {% block conteudo %}
    <link rel="stylesheet" href="../static/styles/recebePedido.css">
    <link       rel="stylesheet"       href="../static/styles/media/media-
recebePedido.css">

    <body>
```

```
        <main>
            <section class="container-cards" id="order-container">
                <!-- Os pedidos serão carregados dinamicamente aqui -->
            </section>
        </main>

        <!-- Modal para selecionar motivo de cancelamento -->
        <div id="cancelModal" class="modal">
            <div class="modal-content">
                <span class="close">&times;</span>
                <h2>Razão do Cancelamento</h2>
                <select id="cancelReason">
                          <option value="" disabled selected>Selecione um
motivo</option>
                             <option value="Pedido em atraso">Pedido em
atraso</option>
                                <option value="Cliente desistiu">Cliente
desistiu</option>
                    <option value="Erro no pedido">Erro no pedido</option>
                </select>

                <!-- Motivo de cancelamento (texto adicional) -->
                   <textarea id="cancelDescription" placeholder="Descreva o
motivo do cancelamento (opcional)"></textarea>

                                    <button id="confirmCancelBtn">Confirmar
Cancelamento</button>
            </div>
        </div>

        <script>
            // Função para exibir os pedidos via AJAX
            function carregarPedidos() {
                fetch('/obter_pedidos')
                    .then(response => response.json())
                    .then(data => {
                                                const orderContainer =
document.getElementById('order-container');
                            orderContainer.innerHTML = ''; // Limpa o conteúdo
antes de carregar os pedidos

                                    for (const [id_cliente, cliente] of
Object.entries(data)) {
                                let cardPedidos = `<div class="card-pedidos">`;

                                        for (const [id_pedido, pedido] of
Object.entries(cliente.pedidos)) {
                                    cardPedidos += `
                                <div class="order-card">
```

```
                                                    <h2>Cliente:   <a
href="/usuario/{{id_cliente}}">${cliente.nome_cliente}</a></h2>
                                <div class="order-header">
                                                    <span  class="order-
number">#${id_pedido}</span>
                                                    <span  class="order-
time">${pedido.hora}</span>
                                                    <span  class="order-
date">${pedido.data_pedido}</span>
                                </div>
                                <div class="order-details">
                                    <h3>Pedidos:</h3>
                                    <ul>`;

                                // Exibe os produtos com quantidade
                                                if  (pedido.produtos  &&
pedido.produtos.length > 0) {
                                    pedido.produtos.forEach(produto => {
                                        if (produto.nome_produto) {
                                                cardPedidos += `<li> Produto:
${produto.nome_produto} <br> - Quantidade: ${produto.quantidade} <br> - Preço:
R$ ${produto.preco.toFixed(2)}</li>`;
                                        }
                                    });
                                }

                                 // Exibe as marmitas com quantidade e
associa guarnições e acompanhamentos
                                                if  (pedido.marmitas  &&
pedido.marmitas.length > 0) {
                                    pedido.marmitas.forEach(marmita => {
                                        if (marmita.nome_marmita) {
                                                cardPedidos += `<li> Marmita:
${marmita.nome_marmita} <br> - Tamanho: ${marmita.tamanho} <br> - Quantidade:
${marmita.quantidade} <br> - Preço: R$ ${marmita.preco.toFixed(2)}</li>`;

                                                // Guarnições associadas à
marmita
                                                if (pedido.guarnicoes &&
pedido.guarnicoes.length > 0) {
                                                    cardPedidos  +=
`<ul><li><strong>Guarnições:</strong></li>`;

pedido.guarnicoes.forEach(guarnicao => {
                                                    cardPedidos  +=
`<li>${guarnicao}</li>`;
                                                });
                                                cardPedidos += `</ul>`;
                                            }
```

```
                                        // Acompanhamentos associados à
marmita
                                          if (pedido.acompanhamentos &&
pedido.acompanhamentos.length > 0) {
                                                                cardPedidos  +=
`<ul><li><strong>Acompanhamentos:</strong></li>`;

pedido.acompanhamentos.forEach(acompanhamento => {
                                                                cardPedidos  +=
`<li>${acompanhamento}</li>`;
                                           });
                                           cardPedidos += `</ul>`;
                                       }
                                   }
                               });
                           }

                                   // Mensagem caso não haja produtos,
marmitas, guarnições ou acompanhamentos
                                   if (pedido.produtos.length  ===  0  &&
pedido.marmitas.length   ===   0   &&   pedido.guarnicoes.length   ===   0   &&
pedido.acompanhamentos.length === 0) {
                                     cardPedidos += `<li>Nenhum produto,
marmita, guarnição ou acompanhamento</li>`;
                                   }

                                   cardPedidos += `</ul>
                                   </div>
                                   <div class="order-status">
                                         <span class="order-price">Total: R$
${pedido.total_preco.toFixed(2)}</span>
                                   </div>
                                   <div class="btn-cancelar">
                                        <button class="status-btn preparando"
data-pedido="${id_pedido}" value="preparando">Preparando</button>
                                    <button class="status-btn pronto" data-
pedido="${id_pedido}" value="feito">Pronto</button>
                                         <button class="status-btn entregue"
data-pedido="${id_pedido}" value="entregue">Entregue</button>
                                         <button class="cancel-btn cancelar"
data-pedido="${id_pedido}">Cancelar</button>
                                   </div>
                               </div>`;
                               }
                               cardPedidos += `</div>`;
                                   orderContainer.innerHTML  +=  cardPedidos;  //
Adiciona os pedidos na página
                           }
```

```
                    })
                        .catch(error => console.error('Erro ao carregar os
pedidos:', error));
            }

               // Atualiza a lista de pedidos a cada 2 segundos (tempo
ajustável)
            setInterval(carregarPedidos, 2000);

            // Carrega os pedidos quando a página é carregada
            window.onload = carregarPedidos;

            // Função para lidar com cliques nos botões de status e cancelar
            document.addEventListener('click', function (e) {
                // Para os botões de status
                if (e.target.classList.contains('status-btn')) {
                    const idPedido = e.target.getAttribute('data-pedido');
                    const novoStatus = e.target.value;

                     // Se o status for "entregue", chama a rota específica
para marcar como entregue
                    if (novoStatus === 'entregue') {
                        fetch(`/marcar_entregue/${idPedido}`, {
                            method: 'POST',
                            headers: {
                                    'Content-Type': 'application/x-www-form-
urlencoded'
                            }
                        })
                            .then(response => response.json())
                            .then(data => {
                                if (data.status === 'sucesso') {
                                    alert('Pedido marcado como entregue!');
                                     carregarPedidos();  // Atualiza a lista
de pedidos na página
                                } else {
                                    alert('Erro ao marcar como entregue: ' +
data.mensagem);
                                }
                            })
                             .catch(error => console.error('Erro ao enviar a
solicitação:', error));
                    } else {
                             // Envia uma solicitação POST via AJAX para
atualizar o status do pedido
                        fetch('/atualizar_status_pedido', {
                            method: 'POST',
                            headers: {
```

```
                                    'Content-Type': 'application/x-www-form-
urlencoded'
                            },
                                                                    body:
`id_pedido=${idPedido}&status=${novoStatus}`
                        })
                            .then(response => response.json())
                            .then(data => {
                                if (data.status === 'sucesso') {
                                    alert('Status atualizado com sucesso!');
                                     carregarPedidos();  // Atualiza a lista
de pedidos na página
                                } else {
                                      alert('Erro ao atualizar o status: ' +
data.mensagem);
                                }
                            })
                             .catch(error => console.error('Erro ao enviar a
solicitação:', error));
                    }
                }

                // Para o botão de cancelar pedido
                if (e.target.classList.contains('cancel-btn')) {
                    const idPedido = e.target.getAttribute('data-pedido');

                    // Abre o modal de cancelamento
                      document.getElementById('cancelModal').style.display =
'flex';

document.getElementById('confirmCancelBtn').setAttribute('data-pedido',
idPedido);
                }
            });

            // Função para fechar o modal
            document.querySelector('.close').onclick = function () {
                     document.getElementById('cancelModal').style.display  =
'none';
            };

            // Bloqueio de campos entre o select e a textarea

document.getElementById('cancelReason').addEventListener('change', function ()
{
                                                const      descricao      =
document.getElementById('cancelDescription');
                if (this.value) {
                    descricao.disabled = true; // Bloqueia o textarea
```

```
                    descricao.value = ''; // Reseta o valor do textarea
                } else {
                    descricao.disabled = false; // Libera o textarea
                }
            });


document.getElementById('cancelDescription').addEventListener('input',
function () {
                const motivo = document.getElementById('cancelReason');
                if (this.value) {
                    motivo.disabled = true; // Bloqueia o select
                    motivo.value = ''; // Reseta o valor do select
                } else {
                    motivo.disabled = false; // Libera o select
                }
            });

            // Função para confirmar o cancelamento
              document.getElementById('confirmCancelBtn').onclick = function
() {
                const idPedido = this.getAttribute('data-pedido');
                                            const     motivoCancelamento     =
document.getElementById('cancelReason').value;
                                           const     descricaoCancelamento     =
document.getElementById('cancelDescription').value;

                // Verifica se pelo menos uma das opções foi preenchida
                if (!motivoCancelamento && !descricaoCancelamento) {
                     alert('Por favor, escolha uma das opções: selecione um
motivo ou preencha a descrição.');
                    return;
                }

                  // Envia uma solicitação POST via AJAX para cancelar o
pedido
                fetch('/cancelar_pedido', {
                    method: 'POST',
                    headers: {
                        'Content-Type': 'application/x-www-form-urlencoded'
                    },
                                                                            body:
`id_pedido=${idPedido}&motivo_cancelamento=${encodeURIComponent(motivoCancelam
ento)}&descricao_cancelamento=${encodeURIComponent(descricaoCancelamento)}`
                })
                    .then(response => response.json())
                    .then(data => {
                        if (data.status === 'sucesso') {
                            alert('Pedido cancelado com sucesso!');
```

```
                                    carregarPedidos(); // Atualiza a lista de
pedidos na página
                            } else {
                                        alert('Erro ao cancelar o pedido: ' +
data.mensagem);
                            }
                            document.getElementById('cancelModal').style.display
= 'none'; // Fecha o modal
                        })
                            .catch(error => console.error('Erro ao enviar a
solicitação:', error));
                };

                // Função para resetar os campos ao fechar o modal
                document.querySelector('.close').onclick = function () {
                    const motivo = document.getElementById('cancelReason');
                                                    const descricao =
document.getElementById('cancelDescription');
                    motivo.value = '';
                    motivo.disabled = false; // Libera o select
                    descricao.value = '';
                    descricao.disabled = false; // Libera o textarea
                         document.getElementById('cancelModal').style.display =
'none';
                };

            </script>
            {% endblock %}
```

```
        {% extends "modelo-adm.html" %}
        {% block conteudo %}
        <link rel="stylesheet" href="../static/styles/relatorio.css">
        <section class="filtro-relatorio">
            <h2>Relatório de Vendas</h2>
            <h3>Filtrar por Período</h3>
            <form method="POST" action="{{ url_for('relatorio') }}">
                    <label class="label-data-inicial" for="data-inicial">Data
Inicial:</label>
                   <input type="date" class="data-inicial" name="data_inicial"
required>

                       <label class="label-data-final" for="data-final">Data
Final:</label>
                     <input type="date" class="data-final" name="data_final"
required>
```

```
                    <button  type="submit"  class="botao-gerar-relatorio">Gerar
Relatório</button>
            </form>
        </section>

        <div class="relatorio-vendas">

            <section class="resumo-relatorio">
                <h3>Pedidos Entregues</h3>

                {% if relatorio_dados %}
                <table>
                    <thead>
                        <tr>
                            <th>ID</th>
                            <th>Data</th>
                            <th>Hora</th>
                            <th>Valor</th>
                        </tr>
                    </thead>
                    <tbody>
                        {% for pedido in relatorio_dados %}
                        <tr>
                            <td>{{ pedido[0] }}</td>
                            <td>{{ pedido[1] }}</td>
                            <td>{{ pedido[2] }}</td>
                            <td>{{ pedido[4] }}</td>
                        </tr>
                        {% endfor %}
                    </tbody>
                </table>
                <div class="spanpontilhado">
                    <span>Valor Total: R$ {{ valor_total_geral }}</span> <br>
                    <span>Pedidos Entregues: {{ total_pedidos }}</span>
                </div>

                {% else %}
                     <p>Nenhum  relatório  disponível.  Por  favor,  insira  um
período.</p>
                {% endif %}
            </section>

            <section class="resumo-cancelados">
                <h3>Pedidos Cancelados</h3>

                {% if cancelados_dados %}
                <table>
                    <thead>
                        <tr>
```

```
                        <th>ID</th>
                        <th>Data</th>
                        <th>Hora</th>
                        <th>Valor</th>
                    </tr>
                </thead>
                <tbody>
                    {% for pedido in cancelados_dados %}
                    <tr>
                        <td>{{ pedido[0] }}</td>
                        <td>{{ pedido[1] }}</td>
                        <td>{{ pedido[2] }}</td>
                        <td>{{ pedido[5] }}</td>
                    </tr>
                    <tr>
                            <td colspan="4" style="text-align: center; font-
weight: bolder;" >{{ pedido[4] }}</td>
                    </tr>


                    {% endfor %}
                </tbody>
            </table>
            <div class="spanpontilhado">
                <span>Valor Cancelado: R$ {{ valor_total_cancelado }}</span>
<br>
                <span>Pedidos Cancelados: {{ total_cancelados }}</span>
            </div>

            {% else %}
            <p>Nenhum pedido cancelado no período selecionado.</p>
            {% endif %}
        </section>
    </div>

    {% endblock %}
```

```
    {% extends "modelo.html" %}
    {% block conteudo %}
    <link rel="stylesheet" href="../static/styles/trocar-senha.css">

    <body>
        <main>
            <div class="btn-voltar">
                            <button     onclick="window.history.back()"><img
src="https://www.svgrepo.com/show/324205/back-arrow-navigation.svg"
alt=""></button>
```

```
        </div>

        <div class="titulo">
            <h2>Esqueceu a sua Senha?</h2>
            <p>Coloque suas informações cadastradas e receba o código</p>
        </div>

        <section class="container-principal">

            <div class="formulario">

                <form method="POST">
                       <input name="email" id="email" placeholder="*E-mail"
type="email" required>
                            <input  name="telefone"  placeholder="*Telefone"
type="tel" required>
                    <button type="submit">Enviar Código</button>
                </form>

            </div>

        </section>




        {% endblock %}


        </main>


    </body>

    </html>
```

```
    {% extends "modelo-adm.html" %}
    {% block conteudo %}
    <link rel="stylesheet" href="../static/styles/perfilVisaoAdm.css">
    <!-- Imagem de fundo e imagem de perfil -->
    <div class="header">
        <img src="{{ imagem_perfil_url }}" alt="Foto de {{ cliente.nome_comp
}}" class="img-perfil">
    </div>

    <!-- Informações do cliente -->
```

```
    <div class="perfil-info">
        <span class="nome">{{ cliente.nome_comp }}</span><br>
        <span class="curso">{{ cliente.nome_curso }}</span><br>
        <a href="https://wa.me/{{ cliente.telefone }}">
            <span class="dados-pessoais">{{ cliente.telefone }}</span>
        </a>
    </div>

    <div class="relatorio-vendas">
        <section class="resumo-relatorio">
            <h3>Últimos Pedidos</h3>

            {% if pedidos %}
            <table>
                <thead>
                    <tr>
                        <th>Data</th>
                        <th>Valor</th>
                        <th>Item</th>
                    </tr>
                </thead>
                <tbody>
                    {% for pedido in pedidos %}
                    <tr>
                        <td>{{ pedido.data }}</td>
                        <td>{{ pedido.valor }}</td>
                        <td>
                            <ul>
                                {% for item in pedido.itens %}
                                <li>{{ item }}</li>
                                {% endfor %}
                            </ul>
                        </td>
                    </tr>
                    {% endfor %}
                </tbody>
            </table>
            <div class="spanpontilhado">
                <span>Valor Total: R$: {{ total_valor }}</span> <br>
                <span>Pedidos Entregues: {{ pedidos_entregues }}</span>
            </div>
            {% else %}
            <p>Este cliente ainda não fez pedidos.</p>
            {% endif %}
        </section>
    </div>
    {% endblock %}
```

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <meta   name="viewport"   content="width=device-width,   initial-
scale=1.0">
    <title>Cantina Digital</title>
    <link rel="stylesheet" href="../static/styles/verificacao.css">
</head>
<body>

    <div class="btn-voltar">
                        <button     onclick="window.history.back()"><img
src="https://www.svgrepo.com/show/324205/back-arrow-navigation.svg"
                alt=""></button>
    </div>

    <section class="container">

        <span class="titulo">Codigo de Acesso</span>
            <span  class="subtitulo">Insira  o  código  para  trocar  a
senha</span>
              <span   class="telefone">Enviamos   o   código   para:   <a
href="/cadastro">+5516998999889</a></span>

        <div class="formulario">
            <form method="POST">
                <label for="codigo"></label>
                <div class="codigo-verificacao">
                        <input type="text" maxlength="1" placeholder="0"
name="codigo1" class="digito" required>
                        <input type="text" maxlength="1" placeholder="0"
name="codigo2" class="digito" required>
                        <input type="text" maxlength="1" placeholder="0"
name="codigo3" class="digito" required>
                        <input type="text" maxlength="1" placeholder="0"
name="codigo4" class="digito" required>
                </div>
                <button type="submit">Verificar</button>
            </form>
        </div>
        {% if erro %}
        <p style="color: red;">{{ erro }}</p>
        {% endif %}
```

```
        </section>

        <script>
            const digitos = document.querySelectorAll(".digito");

            digitos.forEach((digito, index) => {
                digito.addEventListener("input", () => {
                    if (digito.value.length === 1 && index < digitos.length
- 1) {
                        digitos[index + 1].focus();
                    }
                });

                digito.addEventListener("keydown", (e) => {
                        if (e.key === "Backspace" && index > 0 &&
digito.value.length === 0) {
                        digitos[index - 1].focus();
                    }
                });
            });
        </script>

        {% if erro %}
        <p>{{ erro }}</p>
        {% endif %}

    </body>
    </html>
```

```
    *{
        box-sizing: border-box;
        margin: 0;
        padding: 0;
    }


    .titulo{
        margin-top: 2.5rem;
        display: flex;
        margin-left: 26px;
        margin-right: 20px;
        margin-bottom: 20px;
    }

    h2{
        font-weight: bolder !important;
        text-align: left;
```

```
            font-size: 2.5rem !important;
             font-family: system-ui, -apple-system, BlinkMacSystemFont, 'Segoe
UI', Roboto, Oxygen, Ubuntu, Cantarell, 'Open Sans', 'Helvetica Neue', sans-
serif;
        }

        .container-principal{
            display: flex;
            align-items: center;
            flex-direction: column;
        }

        form{
            display: flex;
            flex-direction: column;
        }

        form input{
            width: 23.7rem;
            height: 30px;
            padding: 30px;
            border-radius: 5px;
            border: 2px solid rgb(238, 238, 238);
            background-color: rgb(245, 245, 245);
            margin-top: 15px;
        }

        input::placeholder {
            color: #7e7d7d; /* Mude para a cor desejada */
        }

        .checkbox{
            display: flex;
            flex-direction: row;
            margin-top: 30px;
            margin-right: 204px;
            margin-bottom: 40px;
        }

        .container-principal button{
            padding: 15px;
            background-color: rgb(161, 23, 23);
            width: 23.7rem;
            border-radius: 6px;
            border: none;
            margin-top: 20px;
            color: white;
            font-size: 18px;
        }
```

```
form input:focus{
    outline: none;
}
.formulario form{
    width: 30rem;
    height: 30rem;
    display: flex;
    align-items: center;
    flex-direction: column;
}

.select{
    margin-top: 20px;
    display: flex;
    flex-direction: column;
    align-items: center;
    justify-content: center;
    gap: 1rem;
}

.select select{
    width: 23.7rem;

    padding: 19px;
    border-radius: 5px;
    border: 2px solid rgb(238, 238, 238);
    background-color: rgb(245, 245, 245);

}

.login-btn{
    text-decoration: none;
    color: rgb(156, 0, 0);
}

.login-btn{
    text-decoration: none;
    color: rgb(156, 0, 0);
    text-decoration: none;
}

.login-btn:hover{
    text-decoration: none;
    color: rgb(156, 0, 0);
    text-decoration: none;
}

.cbc-cima{
```

```
        display: none    ;
    }

    .gambiarra{
        display: none;
    }

    .btn-voltar img{
        height: 20px;

    }

    .btn-voltar button{
        display: flex;
        align-items: center;
        justify-content: center;
        height: 40px;
        background-color: transparent;
        border: 1px solid rgb(236, 236, 236);
        width: 40px;
        border-radius: 6px;
        margin-top: 10px;
        margin-left: 26px;
    }

    option{
        background-color: rgb(245, 245, 245);
        color: #000;
        font-size: 8px; /* Tamanho da fonte das opções */

    }

    form a{
        margin-top: 10px;
        margin-bottom: 15px;
        margin-left: 240px;
        color: black;
    }

    form a:hover{
        color: black;
        text-decoration: none;
    }

    .semConta{
        margin-top: 15px;
    }
```

```
    .popup-container {
        position: fixed;        /* Fixa o elemento na tela */
        top: 30px !important;             /* Distância do topo da tela */
        left: 50%;             /* Centraliza horizontalmente */
        transform: translateX(-50%); /* Ajusta a posição para o centro */
         z-index:  1000;            /*  Garante  que  fique  sobre  os  outros
elementos */
        display: flex;
        justify-content: center;
        opacity: 0;            /* Inicialmente invisível */
         animation: slideIn 0.5s forwards; /* Adiciona animação ao aparecer
*/
    }

    .espacamento{
        display: none;
    }
```

```
    *{
      font-family: Inter,sans-serif;
      box-sizing: border-box;
      margin: 0;
      padding: 0;


    }

    main{
      width: 100vw;
      display: flex;
      justify-content: center;
    }

    form {
      border-radius: 12px;
      max-width: 400px;
      width: 100%;
      display: flex;
      flex-direction: column;
    }

    h2{
      margin-top: 1rem;
      display: flex;
```

```
    margin-right: 20px;
    margin-bottom: 10px;
    font-weight: bolder;
    text-align: left;
  }



   input{
    height: 25px;
    padding: 26px;
    border-radius: 5px;
    border: 2px solid rgb(238, 238, 238);
    background-color: rgb(245, 245, 245);
    margin-top: 15px;
    width: 351px;

  }

  input::placeholder {
    color: #7e7d7d; /* Mude para a cor desejada */
  }

  .input-descricao::placeholder {
    color: #7e7d7d; /* Mude para a cor desejada */
  }
  .input-descricao{
    height: 50px;
    padding-right: 30px;
    border-radius: 5px;
    border: 2px solid rgb(238, 238, 238);
    background-color: rgb(245, 245, 245);
    margin-top: 15px;
  }

  .btn-Cadastrar {
    margin-top: 15px;
    padding: 15px;
    background-color: rgb(161, 23, 23);
    border-radius: 6px;
    border: none;
    color: white;
    margin-bottom: 5px;
  }


  .btn-Cadastrar:active {
      background-color: #004a92;
  }
```

```css
/* Responsividade */
@media (max-width: 768px) {
    form {
        width: 90%;
    }
}

::placeholder {
  color: black;
  opacity: 1;
}

form{

  display: flex;
  flex-direction: column;
  color: black;
}


form input :active{
color: rgb(0, 217, 255);
}




select::-ms-expand {
display: none;
}

select:after {
content: '<>';
font: 17px "Consolas", monospace;
color: #333;
-webkit-transform: rotate(90deg);
-moz-transform: rotate(90deg);
-ms-transform: rotate(90deg);
transform: rotate(90deg);
right: 11px;
top: 18px;
padding: 0 0 2px;
border-bottom: 1px solid #999;
position: absolute;
pointer-events: none;
}
```

```
select{
  height: 60px;
  width: 351px;
  border: 2px solid rgb(238, 238, 238);
  background-color: rgb(245, 245, 245);
  margin-top: 15px;
  border-radius: 5px;
  padding-left: 23px;
  color: #6e6e6e;
}

.btn-voltar img{
  height: 20px;

}

.btn-voltar button{
  position: absolute;
  display: flex;
  align-items: center;
  justify-content: center;
  height: 40px;
  background-color: transparent;
  border: 1px solid rgb(236, 236, 236);
  width: 40px;
  border-radius: 6px;
  margin-top: 15px;

}


#guarnicoes {
  margin-top: 15px;
  padding: 15px;
  background-color: #f9f9f9;
  border-radius: 8px;
  border: 1px solid #ddd;
}

#guarnicoes h4 {
  margin-bottom: 10px;
  font-size: 18px;
  font-weight: 600;
  color: #333;
}

#guarnicoes div {
  display: flex;
  align-items: center;
```

```
    margin-bottom: 10px;
  }

  #guarnicoes input[type="checkbox"] {
    margin-right: 10px;
  }

  #guarnicoes label {
    font-size: 16px;
    color: #333;
  }




  .acompanhamento {
    display: flex;
    align-items: center;
    padding: 10px;
    border-top: 1px solid rgb(233, 231, 231);
  }

  .acompanhamento img {
    width: 40px;
    /* Tamanho da imagem */
    height: 40px;
    margin-left: 11px;
    margin-right: 10px;
    /* Espaço entre imagem e texto */
    border-radius: 5px;
    border: 0.5px solid red;
  }

  .acompanhamento label {
    position: relative;
    /* Para alinhar a bolinha à direita */
    cursor: pointer;
    /* Espaço entre o texto e a bolinha */
    z-index: 1;
    margin-top: 10px;
  }

  .acompanhamento label span {
    position: absolute;
    left: 0;
    top: 50%;
    transform: translateY(-50%);
    width: 20px;
    height: 20px;
```

```
    border: 2px solid #ff0000;
    /* Cor da bolinha */
    border-radius: 50%;
    background-color: white;
    /* Fundo da bolinha */
    transition: background-color 0.3s, border-color 0.3s;
}

.acompanhamento span {
    flex-grow: 1;
    font-size: 18px;
    /* Tamanho do texto do acompanhamento */
    margin-right: 5px;
    /* Espaço entre o texto e a bolinha */
}

#acompanhamentos input[type="checkbox"],
#guarnicoes_acompanhamentos input[type="checkbox"] {
    margin-top: 5px;
    position: absolute;
    left: 325px;
    appearance: none;
    -webkit-appearance: none;
    width: 20px; /* Defina o tamanho da bolinha */
    height: 20px;
    border-radius: 50%;
    border: 2px solid #ff0000;;
    transition: background-color 0.3s, border-color 0.3s;
    background-color: white;
}

/* Estilo para o checkbox marcado */
#acompanhamentos input[type="checkbox"]:checked,
#guarnicoes_acompanhamentos input[type="checkbox"]:checked {
    background-color: #ff0000;
    /* Cor quando selecionado */
    border-color: #ff0000;
}


/* Define um container para cada campo de input e botão */
.input-container {
    position: relative;
    width: fit-content; /* Ajusta a largura do container ao conteúdo */
    margin-top: 10px; /* Espaço entre os containers */
}

/* Estilo do input com espaço para o botão */
.input-container input[type="text"] {
```

```
    padding-right: 30px; /* Espaço para o botão dentro do input */
}

/* Estilo do botão de confirmação */
.input-container button[type="button"] {
    position: absolute;
    right: 0px;
    top: 60%;
    transform: translateY(-50%);
    background: transparent;
    border: none;
    color: rgb(214, 214, 214); /* Cor do ícone de confirmação */
    font-size: 30px;
    cursor: pointer;
    padding-top: 5px;
    padding-bottom: 5px;
    padding-right: 15px;
    padding-left: 15px;
    display: flex;
}

.input-container button[type="button"]:hover {
    color: rgb(177, 177, 177);
}

.espacamento1{
    height: 3rem;
}
```

```
*{
    box-sizing: border-box;
    margin: 0;
    padding: 0;
}


.titulo{
    margin-top: 2.5rem;
    display: flex;
    margin-left: 26px;
    margin-right: 20px;
    margin-bottom: 20px;
}

h2{
    font-weight: bolder;
    text-align: left;
```

```
}

.container-principal{
    display: flex;
    align-items: center;
    flex-direction: column;
}

form{
    display: flex;
    flex-direction: column;
}

form input{
    width: 23.7rem;
    height: 30px;
    padding: 30px;
    border-radius: 5px;
    border: 2px solid rgb(238, 238, 238);
    background-color: rgb(245, 245, 245);
    margin-top: 15px;
}

input::placeholder {
    color: #7e7d7d; /* Mude para a cor desejada */
}

.checkbox{
    display: flex;
    flex-direction: row;
    margin-top: 30px;
    margin-right: 204px;
    margin-bottom: 40px;
}

.container-principal button{
    padding: 15px;
    background-color: rgb(161, 23, 23);
    width: 23.7rem;
    border-radius: 6px;
    border: none;
    margin-top: 10px;
    color: white;
    display: flex;
    justify-content: center;
}

form input:focus{
    outline: none;
```

```
}
.formulario form{
    width: 30rem;
    height: 30rem;
    display: flex;
    align-items: center;
    flex-direction: column;
}

/* Ajuste geral do container */
.select {
  position: relative; /* Garante o alinhamento com o dropdown */
  display: flex;
  flex-direction: column;
  gap: 1rem; /* Espaço entre os elementos */
}


.select select{
    width: 23.7rem;

    padding: 19px;
    border-radius: 5px;
    border: 2px solid rgb(238, 238, 238);
    background-color: rgb(245, 245, 245);

}

.login-btn{
    text-decoration: none;
    color: rgb(156, 0, 0);
}

.login-btn{
    text-decoration: none;
    color: rgb(156, 0, 0);
    text-decoration: none;
}

.login-btn:hover{
    text-decoration: none;
    color: rgb(156, 0, 0);
    text-decoration: none;
}

.cbc-cima{
    display: none    ;
}
```

```
      .gambiarra{
          display: none;
      }

      .btn-voltar img{
          height: 20px;

      }

      .btn-voltar button{
          display: flex;
          align-items: center;
          justify-content: center;
          height: 40px;
          background-color: transparent;
          border: 1px solid rgb(236, 236, 236);
          width: 40px;
          border-radius: 6px;
          margin-top: 10px;
          margin-left: 26px;
      }

      option{
          background-color: rgb(245, 245, 245);
          color: #000;
          font-size: 8px; /* Tamanho da fonte das opções */

      }

      .form-group {
           width: 23.7rem; /* Certifica-se de que o grupo de entrada tenha a
mesma largura que os outros campos */
          position: relative; /* Necessário para posicionamento absoluto */
      }

      .input-group {
          display: flex;
           align-items: center; /* Garante que o input e o prefixo tenham a
mesma altura */
      }

      .prefix {
          margin-top: 15px;
           background-color: rgb(245, 245, 245); /* A mesma cor de fundo do
input */
          padding: 10px; /* Espaço ao redor do prefixo */
          border: 2px solid rgb(238, 238, 238); /* A mesma borda dos inputs */
          border-radius: 5px 0 0 5px; /* Bordas arredondadas */
```

```
            margin-right: -2px; /* Ajusta o espaçamento entre o prefixo e o
campo de entrada */
             display: flex; /* Garante que o conteúdo do prefixo fique
centralizado */
          align-items: center; /* Alinha verticalmente */
          justify-content: center; /* Centraliza horizontalmente */
          height: 100%; /* Garante que o prefixo tenha a mesma altura do input
*/
      }

      input[type="text"] {
           padding: 10px; /* Ajusta o padding do input para alinhar com o
prefixo */
          border-radius: 0 5px 5px 0; /* Bordas arredondadas */
          flex: 1; /* O campo de entrada ocupa o espaço restante */
          height: 40px; /* Mantém uma altura consistente */
      }


      /* Se necessário, você pode ajustar a altura do campo de entrada */
      input[type="text"], .prefix {
           height: 56px; /* Define uma altura padrão para o prefixo e o input
*/
      }

      input::placeholder {
           color: #7e7d7d; /* Cor do placeholder, mantenha o mesmo estilo que
já está usando */
      }









      /* From Uiverse.io by revanth-004 */
      /* COMMON STYLES*/
      .popup-container {
          position: fixed;        /* Fixa o elemento na tela */
          top: 50px;             /* Distância do topo da tela */
          left: 50%;             /* Centraliza horizontalmente */
          transform: translateX(-50%); /* Ajusta a posição para o centro */
```

```
            z-index: 1000;            /* Garante que fique sobre os outros
elementos */
            display: flex;
            justify-content: center;
            opacity: 0;            /* Inicialmente invisível */
             animation: slideIn 0.5s forwards; /* Adiciona animação ao aparecer
*/
        }

        /* Animação para fazer o alerta aparecer deslizando de cima */
        @keyframes slideIn {
            0% {
                 transform: translate(-50%, -20px); /* Começa um pouco acima do
topo */
                opacity: 0;                        /* Inicialmente invisível */
            }
            100% {
                transform: translate(-50%, 0);     /* Termina na posição central
*/
                opacity: 1;                        /* Fica totalmente visível */
            }
        }

        .popup {
            margin: 0;           /* Remove margem para evitar deslocamento */
            box-shadow: 4px 4px 10px -10px rgba(0, 0, 0, 1);
            width: 300px;        /* Define a largura do popup */
            justify-content: space-around;
            align-items: center;
            display: flex;
            border-radius: 4px;
            padding: 5px 0;
            font-weight: 300;
        }
          .popup svg {
            width: 1.25rem;
            height: 1.25rem;
          }
          .popup-icon svg {
            margin: 5px;
            display: flex;
            align-items: center;
          }
          .close-icon {
            margin-left: auto;
          }
          .close-svg {
            cursor: pointer;
          }
```

```
.close-path {
  fill: grey;
}

/*SEPERATE STYLES*/

/* SUCCESS */
.success-popup {
  background-color: #edfbd8;
  border: solid 1px #84d65a;
}
.success-icon path {
  fill: #84d65a;
}
.success-message {
  color: #2b641e;
}

/* ALERT */
.alert-popup {
  background-color: #fefce8;
  border: solid 1px #facc15;
}
.alert-icon path {
  fill: #facc15;
}
.alert-message {
  color: #ca8a04;
}

/* ERROR */

.error-popup {
  background-color: #fef2f2;
  border: solid 1px #f87171;
}
.error-icon path {
  fill: #f87171;
}
.error-message {
  color: #991b1b;
}

/* INFO */

.info-popup {
  background-color: #eff6ff;
  border: solid 1px #1d4ed8;
}
```

```
    .info-icon path {
      fill: #1d4ed8;
    }
    .info-message {
      color: #1d4ed8;
    }




#senha {
  margin-bottom: 10px;
}



/* Estilo do Select2 */
.select2-container {
  width: 23.7rem; /* Ajuste a largura para combinar com o seu design */
}

/* Personalização da barra de pesquisa do Select2 */
.select2-search__field {
  padding: 19px;
  border-radius: 5px;
  border: 2px solid rgb(238, 238, 238);
  background-color: rgb(245, 245, 245);
  font-size: 16px;
  color: #333;
}

/* Estilo para as opções do dropdown */
.select2-results__option {
  padding: 10px;
  background-color: rgb(245, 245, 245);
  color: #333;
  font-size: 16px;
  border: 1px solid rgb(238, 238, 238);
  cursor: pointer;
}

/* Hover nas opções do dropdown */
.select2-results__option:hover {
  background-color: rgb(255, 102, 178);
  color: #fff;
}

/* Seleção da opção */
.select2-results__option[aria-selected="true"] {
```

```
    background-color: #ff66b2;
    color: white;
  }

  .select2-container--default .select2-selection--single {
    background-color: #F5F5F5;
    border: 2px solid rgb(232 232 232);
    border-radius: 4px;
  }
  .select2-container .select2-selection--single {
    box-sizing: border-box;
    cursor: pointer;
    display: block;
    height: 64px;
    user-select: none;
    padding-left: 20px;
    -webkit-user-select: none;
    display: flex;
    flex-direction: row;
    align-items: center;
  }

  /* Esconde o 'X' de limpar seleção */
  .select2-selection__clear {
    display: none !important;
  }
```

```
  * {
      box-sizing: border-box;
      margin: 0;
      padding: 0;
  }

  .container-principal {
      display: flex;
      flex-direction: column;
      align-items: center;
      /* Centraliza o conteúdo */
  }

  .subtitle {
      margin-left: 20px;
  }

  .imgdiv {
      display: flex;
      align-items: center;
```

```
        /* Alinha as imagens verticalmente */
    }

    .produto-card {
        display: flex;
        width: 100%;
        /* O card ocupa 100% da largura da div pai */
        max-width: 600px;
        /* Ajuste a largura máxima conforme necessário */
        height: auto;
        /* Permite que a altura se ajuste automaticamente ao conteúdo */
        align-items: center;
        border-bottom: 1px solid rgb(180, 180, 180);
        padding: 10px;
        /* Adiciona um espaçamento interno */
    }

    .img-produto {
        margin-left: 20px;
        max-width: 130px;
        /* Reduzindo a largura máxima da imagem */
        height: auto;
        /* Mantém a proporção da imagem */
    }

    .button {
        border: none;
        background: none;
        height: 30px;
        width: 30px;
        position: absolute;
        margin-left: 10px;
        margin-top: -10px;
        border-radius: 15px;
        background-color: white;
        display: flex;
        justify-content: center;
        align-items: center;
        box-shadow: 5px 5px 10px rgba(0, 0, 0, 0.5);
    }

    .button img {
        height: 20px;
        width: 20px;
        display: flex;
    }

    .cima {
        display: flex;
```

```
        justify-content: space-between;
        align-items: center;
        margin-bottom: 5px;
        /* Espaço entre o título e a quantidade */
    }

    .btn-lixo {
        border: none;
        background: none;
        height: 10px;
    }

    .cima input {
        width: 35px;
        border: none;
        text-align: center;
        pointer-events: none;
        background-color: #f8f8f8;
    }

    input[type="number"]::-webkit-outer-spin-button,
    input[type="number"]::-webkit-inner-spin-button {
        -webkit-appearance: none;
        margin: 0;
    }

    .cima input:focus {
        outline: none;
    }

    .cima button {
        border: none;
        background: none;
    }

    .baixo {
        display: flex;
        flex-direction: column;
        margin-top: 5px;
        /* Espaço entre a parte de cima e a parte de baixo */
    }

    .baixo span {
        margin-left: 8px;
    }



    .desc-baixo {
```

```
        font-size: 10px;
        margin-top: 5px;
        /* Espaço acima da descrição do produto */
    }

    .qnt-btn {
        display: flex;
        align-items: center;
        justify-content: center;
        margin-right: 0px;
        background-color: #f8f8f8;
        border-radius: 5px;
        padding: 5px 10px;
        /* Padding uniforme */
    }

    .qnt-btn button {
        color: red;
    }

    .qnt-btn button:focus {
        outline: none;
    }

    .obs-item {
        font-size: 10px;
        margin-top: 5px;
        /* Espaço acima da observação */
    }

    .guarnicoes,
    .acompanhamentos {
        margin-top: 5px;
        /* Espaço acima da lista de guarnições e acompanhamentos */
    }

    .guarnicoes strong,
    .acompanhamentos strong {
        display: block;
        /* Faz o título ocupar uma linha inteira */
        margin-bottom: 3px;
        /* Espaço abaixo do título */
    }

    .guarnicoes ul,
    .acompanhamentos ul {
        margin-left: 20px;
        /* Espaço à esquerda para as listas */
        list-style-type: disc;
```

```
        /* Adiciona marcadores */
    }

    /* Estilo da finalização */
    .finalizacao {
        background: white;
        padding: 15px;
        position: fixed;
        bottom: 0;
        left: 0;
        right: 0;
        box-shadow: 0 -2px 10px rgba(0, 0, 0, 0.1);
        display: flex;
        flex-direction: column;
        /* Coloca os botões um em cima do outro */
        align-items: flex-start;
        /* Alinha à esquerda */
    }

    .preco {
        font-size: 18px;
        font-weight: bold;
        margin-bottom: 10px;
        /* Espaço entre o preço e os botões */
    }

    .button1 {
        background-color: rgb(161, 23, 23);
        color: white;
        border: none;
        padding: 10px 20px;
        border-radius: 5px;
        cursor: pointer;
        width: 100%;
        /* Faz o botão ocupar toda a largura */
        margin-bottom: 10px;
        /* Espaço entre os botões */
    }

    /* Links */
    .finalizacao a {

        padding: 5px 100px;
        border: 1px solid rgb(0, 0, 0);
        text-decoration: none;
        color: black;
    }

    a:hover {
```

```
        text-decoration: none;
        color: black;

    }

    .modal {
        position: fixed;
        top: 0;
        left: 0;
        width: 100%;
        height: 100%;
        background-color: rgba(0, 0, 0, 0.5);
        display: flex;
        justify-content: center;
        align-items: center;
        z-index: 1000;
    }

    .modal-conteudo {
        background-color: #fff;
        padding: 20px;
        border-radius: 8px;
        text-align: center;
        width: 300px;
    }

    .botoes-modal {
        margin-top: 20px;
    }

    .botoes-modal button {
        margin: 0 10px;
        padding: 10px 20px;
        border: none;
        cursor: pointer;
        border-radius: 5px;
    }

    .button-confirmar {
        background-color: #4caf50;
        color: white;
    }

    .button-cancelar {
        background-color: #f44336;
        color: white;
    }
```

```css
/* Estilos gerais */
body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
    background-color: #ffffff;
    color: #333;
    height: 100vh;
}

main {
    max-width: 600px;
    margin: auto;
}

h1 {
    text-align: center;
    color: #333;
}

/* Estilo para cards */
table, thead, tbody, th, tr {
    display: block;
    width: 100%;
}

tbody {
    padding: 0;
}

tbody tr {
    background-color: #ffffff;
    border-radius: 5px;
    padding: 10px 15px;
    margin-bottom: 15px;
}

tbody tr td {
    display: flex;
    justify-content: center;
    align-items: center;
    padding: 8px 5px;
    border-bottom: 1px solid #ddd;
}

tbody tr td:last-child {
    border-bottom: none;
}
```

```
td label {
    font-weight: bold;
    margin-right: 10px;
}

input[type="text"],
textarea,
input[type="file"] {
    width: 100%;
    padding: 8px;
    border: 1px solid #ddd;
    border-radius: 4px;
    margin-top: 5px;
}

button[type="submit"] {
    width: 100%;
    padding: 10px;
    border: none;
    background-color: rgb(161, 23, 23);
    color: white;
    border-radius: 4px;
    cursor: pointer;
    transition: background-color 0.3s;
    margin-top: 10px;
}

/* Imagem do produto */
td img {
    width: 250px;
    height: auto;
    border-radius: 4px;
}

select{
    display: flex;
    flex-direction: column;
    align-items: center;
    justify-content: center;
    gap: 1rem;
    width: 100%;

    padding: 19px 5px;
    border-radius: 5px;
    border: 2px solid rgb(238, 238, 238);
}

/* Estilos para o container das opções */
.acompanhamento-container, .guarnicao-container {
```

```
            display: flex;
            align-items: flex-start;
            flex-direction: column;
            gap: 10px;
            padding: 10px;
            border: 1px solid #ddd;
            border-radius: 8px;
            background-color: #f9f9f9;
            max-width: 97%;
            margin-left: 5.5px;
        }

        /* Estilo individual para os labels dos checkboxes */
        .acompanhamento-container label, .guarnicao-container label {
            display: flex;
            align-items: center;
            padding: 5px;
            border-radius: 5px;
            transition: background-color 0.3s;
            cursor: pointer;
        }

        /* Estilo para o checkbox */
        .acompanhamento-container  input[type="checkbox"],  .guarnicao-container
input[type="checkbox"] {
            margin-right: 10px;
            accent-color: #ff0000; /* Cor personalizada do checkbox */
            width: 18px;
            height: 18px;
            cursor: pointer;
        }

        /* Efeito de hover para realçar o item */
        .acompanhamento-container label:hover, .guarnicao-container label:hover
{
            background-color: #f7e6e6;
        }

        /* Estilo do texto do label */
        .acompanhamento-container label span, .guarnicao-container label span {
            font-size: 14px;
            color: #333;
        }

        /* Estilo para quando o checkbox está marcado */
        .acompanhamento-container input[type="checkbox"]:checked + span,
        .guarnicao-container input[type="checkbox"]:checked + span {
            font-weight: bold;
        }
```

```
.nomeguaracomp{
    font-weight: bolder;
    font-size: 20px;
}
/* Estilos gerais */
body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
    background-color: #ffffff;
    color: #333;
    height: 100vh;
}

main {
    max-width: 600px;
    margin: auto;
}

h1 {
    text-align: center;
    color: #333;
}

/* Estilo para cards */
table, thead, tbody, th, tr {
    display: block;
    width: 100%;
}

tbody {
    padding: 0;
}

tbody tr {
    background-color: #ffffff;
    border-radius: 5px;
    padding: 10px 15px;
    margin-bottom: 15px;
}

tbody tr td {
    display: flex;
    justify-content: center;
    align-items: center;
    padding: 8px 5px;
    border-bottom: 1px solid #ddd;
}
```

```
tbody tr td:last-child {
    border-bottom: none;
}

td label {
    font-weight: bold;
    margin-right: 10px;
}

input[type="text"],
textarea,
input[type="file"] {
    width: 100%;
    padding: 8px;
    border: 1px solid #ddd;
    border-radius: 4px;
    margin-top: 5px;
}

button[type="submit"] {
    width: 100%;
    padding: 10px;
    border: none;
    background-color: rgb(161, 23, 23);
    color: white;
    border-radius: 4px;
    cursor: pointer;
    transition: background-color 0.3s;
    margin-top: 10px;
}

/* Imagem do produto */
td img {
    width: 250px;
    height: auto;
    border-radius: 4px;
}
```

```
/* INICIAL */
.status{
    color: rgb(255, 255, 255);
}

.status-cor{
    color: rgb(21, 255, 0);
}
```

```
.footer button{
    color: white;
}


/* CARRINHO */
.subtitle {
     font-family: system-ui, -apple-system, BlinkMacSystemFont, 'Segoe UI', Roboto, Oxygen, Ubuntu, Cantarell, 'Open Sans', 'Helvetica Neue', sans-serif;
    font-size: 20px;
    font-weight: bolder;
}

.cima{
     font-family: system-ui, -apple-system, BlinkMacSystemFont, 'Segoe UI', Roboto, Oxygen, Ubuntu, Cantarell, 'Open Sans', 'Helvetica Neue', sans-serif;
}

.preco{
    font-weight: bolder;
    font-size: 25px;
}
```

```
* {
    box-sizing: border-box;
    margin: 0;
    padding: 0;
}



.cbc-cima .perfil {
    height: 50px;
    width: 50px;
    border-radius: 100px;
}

.cbc-cima .perfil:active {
    filter: grayscale(70%);

}

.logo {
    height: 70px;
```

```
}

.menu {
    height: 40px;
}

.cbc-baixo {
    background-color: rgb(156, 0, 0);
    display: none;
    justify-content: space-around;

}

body {
    font-family: Arial, sans-serif;
    background-color: #f9f9f9;
    color: #333;
    margin: 0;
    padding: 0;
}

.container-cards {
    max-width: 800px;
    margin: 20px auto;
    padding: 20px;

}

.container-cards h2 {
    font-size: 28px;
    color: #333;
    text-align: center;

}

#order-list {
    display: flex;
    flex-direction: column;
    gap: 15px;

}

.card-pedidos {
    border-radius: 8px;
    padding: 15px;

}

.card-pedidos h3 {
```

```
        font-size: 22px;
        color: #555;
        margin-bottom: 10px;
    }

    .order-card {
        background-color: #fff;
        border-radius: 10px;
        box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
        padding: 20px;
        max-width: 450px;
        min-width: 300px;
        margin: 5px 0px 35px 0px;
        transition: transform 0.3s;
    }

    .order-header {
        display: flex;
        justify-content: space-between;
        font-size: 18px;
        color: #777;
        margin-bottom: 10px;
    }

    .order-header span {
        display: inline-block;
    }

    .order-details ul {
        margin: 0;
        padding-left: 20px;
        list-style-type: disc;
        font-size: 16px;
        color: #333;
    }

    .order-status {
        display: flex;
        justify-content: space-between;
        align-items: center;
        margin-top: 10px;
        font-weight: bold;
    }

    .order-price {
        font-size: 18px;
        color: #27ae60;
    }
```

```
.order-status-text {
    font-size: 16px;
    padding: 5px 10px;
    border-radius: 4px;
}

.status-pendente .order-status-text {
    color: #e67e22; /* Laranja */
    background-color: #f9e1c1;
}

.status-preparando .order-status-text {
    color: #525305; /* Amarelo */
    background-color: #f6ffa6;
}

.status-feito .order-status-text {
    color: #20587e; /* Azul */
    background-color: #d9edf7;
}

.status-entregue .order-status-text {
    color: #103b23; /* Verde bem escuro */
    background-color: #c1d6ae; /* Verde muito suave quase branco */
}

.status-cancelado .order-status-text {
    color: #e74c3c; /* Vermelho */
    background-color: #f5c6cb;
}
```

```
@font-face {
    font-family: "Speedee";
    src: url("path-para-font/speedee.woff") format("woff");
}

* {
    box-sizing: border-box;
    margin: 0;
    padding: 0;
    font-family: "Speedee", arial, helvetica, sans-serif;
}

figure {
```

```
        margin: 0 !important;
        display: flex;
        align-items: center;
        justify-content: center;
    }

    html {

        overflow-y: scroll;
    }

    .perfil-icon {
        height: 70px;
        width: 70px;
        display: flex;
        align-items: center;
        justify-content: center;
    }


    .perfil-icon img {

        object-fit: cover;
        margin: 0 auto;
        width: 100%;
        height: 100%;
    }

    .btn-lixo {
        border: none;
        background: none;
        height: 10px;
    }

    .cbc-cima {
        justify-items: center;
        position: fixed;
        /* Fixa a posição no topo da página */
        z-index: 1000;
        /* Garante que fique acima de outros elementos */
        width: 100%;
        /* Largura total da tela */
        /* Espaçamento nas laterais */
        display: grid;
        /* Usando grid para o layout */
        grid-template-columns: repeat(4, 1fr);
        /* Três colunas de largura igual */
        grid-template-rows: repeat(1, 1fr);
        align-items: center;
```

```
        /* Alinha os itens verticalmente ao centro */
        background-color: rgba(207, 34, 38, 1);
        /* Cor de fundo */

    }


    .espacamento1 {
        height: 70px;
    }

    .espacamento {
        height: 70px;
    }
    .cbc-cima .perfil {

        border-radius: 100px;
    }

    .cbc-cima .perfil:active {
        filter: grayscale(70%);

    }

    .logo {
        height: 70px;
    }

    .excluir img {
        height: 15px;
        width: 15px;
        display: flex;
    }

    .cbc-baixo {
        background-color: rgb(156, 0, 0);
        display: none;
        justify-content: space-around;

    }


    h2 {
        color: #333;
        font-size: 2rem !important;
        /* Tamanho ajustado para mobile */
        text-align: center;
        margin-bottom: 30px ;
        margin-top: 20px !important;
```

```
        font-weight: 600 !important;
    }

    .sub h3{
        width: 100%;
        margin-top: 10px;
        font-weight: 660;
        text-align: center;
    }


    .container-produtos {
        display: flex;
        flex-direction: column;
        align-items: center;
        margin-top: 1rem;
        padding: 0 20px; /* Espaço lateral para mobile */
    }

    .container-produtos h3{
        font-weight: 660;
    }

    h2 {
        color: #333;
        font-size: 1.8rem;
        text-align: center;
        margin-bottom: 30px;
        margin-top: 10px;
        font-weight: 600;
    }

    .grid-produto {
        width: 100%;
    }

    .produtos {
        display: flex;
        flex-direction: column; /* Exibição em coluna */
        margin-top: 20px;
        width: 100%;
    }

    .produtos h3{
        margin: 30px 0px;
        font-weight: 660;
    }

    .produtos button {
```

```
        background-color: transparent;
        border: none;
        border-top: 1px solid rgb(210, 210, 210);
        border-bottom: 1px solid rgb(210, 210, 210);
        width: 100%; /* O botão ocupa toda a largura da tela */
        display: flex;
        align-items: center; /* Centraliza verticalmente */
        padding: 10px;
        transition: transform 0.4s ease, box-shadow 0.4s ease;
    }

    .produtos button:focus {
        outline: none;
    }

    .produtos button:hover {
        transform: scale(1.02);
    }

    .card-produtos {
        display: flex;
        align-items: center; /* Alinha imagem e texto verticalmente */
        width: 100%;
    }

    .left-lado {
        display: flex;
        flex-direction: column;
        align-items: baseline;
        font-size: 60px;
        flex: 1; /* Permite que o texto ocupe espaço disponível */
    }

    .card-produtos img {
        height: 90px; /* Tamanho da imagem */
        width: 90px; /* Tamanho da imagem */
        border-radius: 12px; /* Bordas arredondadas */
        object-fit: cover;
    }

    .produto-nome {
        font-size: 1.4rem; /* Ajustado para mobile */
        font-weight: 500;
        color: #333;
        text-align: left;
    }

    .produto-desc {
        text-align: left;
```

```
        font-size: 0.9rem;
    }

    .produto-preco {
        text-align: left;
        font-size: 1.2rem;
        font-weight: 700;
        color: #727272 ;
    }

    /* ========================================================= */
    /* ========================================================= */

    .footer {
        background-color: rgba(207, 34, 38, 255);
        width: 100%;
        height: 80px;
        display: flex;
        justify-content: space-around;
        align-items: center;
        position: fixed;
        left: 0;
        bottom: 0;
    }

    .status {
        color: rgb(255, 255, 255);
    }

    .status-cor {
        color: rgb(21, 255, 0);
    }

    .footer button {

        background-color: rgb(17, 177, 2);
        border-radius: 10px;
        border: none;
        padding: 10px;
    }

    .footer button:active {
        filter: grayscale(20%);
    }

    .perfil-icon a {

        display: flex;
        justify-content: center;
```

```
        align-items: center;
        width: 70%;
        height: 70%;
    }

    .perfil {

        width: 100%;
        height: 100%;
    }

    /*
================================================================================
======================================== */

    .menu-container {
        grid-column-start: 4;
        grid-row-start: 1;
        display: inline-block;
        cursor: pointer;
        z-index: 10;
    }

    .bar1,
    .bar2,
    .bar3 {
        width: 40px;
        height: 5px;
        background-color: rgba(255, 215, 7, 1);
        margin: 6px 0;
        transition: 0.6s;
    }

    .logo-lu {
        grid-column-start: 2;
        grid-column-end: 4;
        grid-row-start: 1;
    }

    .change .bar1 {
        transform: rotate(-45deg) translate(-7px, 7px);
    }

    .change .bar2 {
        opacity: 0;
    }

    .change .bar3 {
        transform: rotate(45deg) translate(-7px, -7px);
```

```
}

.sidenav {
    height: 100%;
    width: 0;
    position: fixed;
    z-index: 2;
    top: 0;
    right: 0;
    background-color: rgb(209, 31, 31);
    overflow-x: hidden;
    transition: 0.5s;
    padding-top: 60px;
}

.sidenav a {
    padding: 8px 8px 8px 32px;
    text-decoration: none;
    font-size: 25px;
    color: rgb(255, 255, 255);
    display: block;
    transition: 0.3s;
}

.sidenav a:hover {
    color: #bbbbbb;
    text-decoration: none;
}

.sidenav .closebtn {
    position: absolute;
    top: 0;
    right: 25px;
    font-size: 36px;
    margin-left: 50px;
}

hr {

    background-color: rgb(255, 255, 255);
}

.car-icon {
    box-shadow: 4px 4px 8px rgba(0, 0, 0, 0.2);
    border-radius: 40px;
    transition: transform 0.4s ease, box-shadow 0.4s ease;
    background-color: white;
    height: 70px;
    width: 70px;
```

```
        display: flex;
        align-items: center;
        justify-content: center;
    }

    .car-icon:hover {
        transform: scale(1.1);
        box-shadow: 4px 4px 9px rgba(0, 0, 0, 0.2);
    }

    .car-icon:active {
        background-color: rgb(250, 250, 250);
    }

    .car-icon img {
        height: 50px;
    }

    footer {
        position: fixed;
        bottom: 0;
        margin-left: 370px;
        margin-bottom: 10px;
        transition: transform 0.4s ease, box-shadow 0.4s ease;
    }

    footer:hover {
        transform: translate(-50px, 0px);
        /* Move a div 20px para a direita e para baixo */
    }

    footer section {
        display: flex;
        justify-content: flex-end;
        margin-right: 15px;
        margin-bottom: 10px;
    }

    /* Modo Escuro */

    /* The switch - the box around the slider */
    .switch {
        display: block;
        --width-of-switch: 3.5em;
        --height-of-switch: 2em;
        /* size of sliding icon -- sun and moon */
        --size-of-icon: 1.4em;
        /* it is like a inline-padding of switch */
        --slider-offset: 0.3em;
```

```
        position: relative;
        width: var(--width-of-switch);
        height: var(--height-of-switch);
    }

    /* Hide default HTML checkbox */
    .switch input {
        opacity: 0;
        width: 0;
        height: 0;
    }

    /* The slider */
    .slider {
        position: absolute;
        cursor: pointer;
        top: 0;
        left: 0;
        right: 0;
        bottom: 0;
        background-color: #f4f4f5;
        transition: .4s;
        border-radius: 30px;
    }

    .slider:before {
        position: absolute;
        content: "";
        height: var(--size-of-icon, 1.4em);
        width: var(--size-of-icon, 1.4em);
        border-radius: 20px;
        left: var(--slider-offset, 0.3em);
        top: 50%;
        transform: translateY(-50%);
        background: linear-gradient(40deg, #ff0080, #ff8c00 70%);
        ;
        transition: .4s;
    }

    input:checked+.slider {
        background-color: #303136;
    }

    input:checked+.slider:before {
        left: calc(100% - (var(--size-of-icon, 1.4em) + var(--slider-offset,
0.3em)));
        background: #303136;
         /* change the value of second inset in box-shadow to change the
angle and direction of the moon  */
```

```
            box-shadow: inset -3px -2px 5px -2px #8983f7, inset -10px -4px 0 0
#a3dafb;
        }

        .carousel-inner {
            background-color: #ffbc0d;
        }

        .carousel-item img {
            height: 600px;
            margin-bottom: 20px;
        }

        .dark-mode {
            height: 290px;
            width: 380px;
            display: flex;
            justify-content: flex-end;
            align-items: flex-end;
        }

        .horario {
            width: 95%;
            height: 1.5rem;
            background-color: rgb(255, 154, 154);
            color: rgb(255, 0, 0);
            display: flex;
            justify-content: center;
            align-items: center;
            text-align: center;
            border-radius: 0.5rem;
        }

        .horario h2 {
            font-size: 0.6rem;
            display: flex;
            justify-content: center;
            text-align: center;
            align-items: center;
            width: 100%;
            margin: 0 !important;
            height: 100%;
        }

        .central {
            background-color: #ffbf00;
            padding: 0.5rem 0 0.5rem 0;
            width: 100%;
            height: 2.5rem;
```

```
        display: flex;
        justify-content: center;
        align-items: center;
    }


    /* From Uiverse.io by revanth-004 */
    /* COMMON STYLES*/
    .popup-container {
        position: fixed;        /* Fixa o elemento na tela */
        top: 80px;              /* Distância do topo da tela */
        left: 50%;              /* Centraliza horizontalmente */
        transform: translateX(-50%); /* Ajusta a posição para o centro */
         z-index: 1000;             /* Garante que fique sobre os outros
elementos */
        display: flex;
        justify-content: center;
        opacity: 0;             /* Inicialmente invisível */
         animation: slideIn 0.5s forwards; /* Adiciona animação ao aparecer
*/
    }

    /* Animação para fazer o alerta aparecer deslizando de cima */
    @keyframes slideIn {
        0% {
             transform: translate(-50%, -20px); /* Começa um pouco acima do
topo */
            opacity: 0;                         /* Inicialmente invisível */
        }
        100% {
            transform: translate(-50%, 0);      /* Termina na posição central
*/
            opacity: 1;                         /* Fica totalmente visível */
        }
    }

    .popup {
        margin: 0;           /* Remove margem para evitar deslocamento */
        box-shadow: 4px 4px 10px -10px rgba(0, 0, 0, 1);
        width: 300px;        /* Define a largura do popup */
        justify-content: space-around;
        align-items: center;
        display: flex;
        border-radius: 4px;
        padding: 5px 0;
        font-weight: 300;
    }
```

```
.popup svg {
    width: 1.25rem;
    height: 1.25rem;
}

.popup-icon svg {
    margin: 5px;
    display: flex;
    align-items: center;
}

.close-icon {
    margin-left: auto;
}

.close-svg {
    cursor: pointer;
}

.close-path {
    fill: grey;
}

/*SEPERATE STYLES*/

/* SUCCESS */
.success-popup {
    background-color: #edfbd8;
    border: solid 1px #84d65a;
}

.success-icon path {
    fill: #84d65a;
}

.success-message {
    color: #2b641e;
}

/* ALERT */
.alert-popup {
    background-color: #fefce8;
    border: solid 1px #facc15;
}

.alert-icon path {
    fill: #facc15;
}
```

```
      .alert-message {
          color: #ca8a04;
      }

      /* ERROR */

      .error-popup {
          background-color: #fef2f2;
          border: solid 1px #f87171;
      }

      .error-icon path {
          fill: #f87171;
      }

      .error-message {
          color: #991b1b;
      }

      /* INFO */

      .info-popup {
          background-color: #eff6ff;
          border: solid 1px #1d4ed8;
      }

      .info-icon path {
          fill: #1d4ed8;
      }

      .info-message {
          color: #1d4ed8;
      }


      /* Estiliza a barra de rolagem para navegadores baseados no WebKit
(Chrome, Safari) */
      ::-webkit-scrollbar {
          width: 12px; /* Define a largura da barra de rolagem */
            height: 12px; /* Define a altura para rolagem horizontal, se
aplicável */
        }

        /* Estiliza o trilho da barra de rolagem */
        ::-webkit-scrollbar-track {
          background: rgba(0, 0, 0, 0); /* Trilho transparente */
        }

        /* Estiliza o "thumb" (parte rolável da barra de rolagem) */
```

```
        ::-webkit-scrollbar-thumb {
          background-color: #cf2226; /* Cor de fundo do thumb */
          border: 1px solid yellow; /* Borda amarela ao redor do thumb */
          border-radius: 10px; /* Borda arredondada para o thumb */
        }

        /* Estiliza o "thumb" quando o usuário passa o mouse sobre ele */
        ::-webkit-scrollbar-thumb:hover {
          background-color: #cf2226; /* Cor de fundo alterada para um vermelho
mais escuro no hover */
        }

      /*
================================================================================
======================================== */

      .menu-container {
          grid-column-start: 4;
          grid-row-start: 1;
          display: inline-block;
          cursor: pointer;
          z-index: 10;
      }

      .bar1,
      .bar2,
      .bar3 {
          width: 40px;
          height: 5px;
          background-color: rgba(255, 215, 7, 1);
          margin: 6px 0;
          transition: 0.6s;
      }

      .logo-lu {
          grid-column-start: 2;
          grid-column-end: 4;
          grid-row-start: 1;
      }

      .change .bar1 {
          transform: rotate(-45deg) translate(-7px, 7px);
      }

      .change .bar2 {
          opacity: 0;
      }

      .change .bar3 {
```

```
            transform: rotate(45deg) translate(-7px, -7px);
        }

        .sidenav {
            height: 100%;
            width: 0; /* O menu inicia fechado */
            position: fixed;
            z-index: 2;
            top: 0;
            right: 0;
            background-color: #D11F1F;
            box-shadow: -2px 0 10px rgba(0, 0, 0, 0.2);
            transition: 0.5s ease;
            padding-top: 60px;
        }


        .sidenav a {
            padding: 15px 32px; /* Aumenta o padding interno para mais espaço */
            font-size: 20px;
            color: #FFFFFF;
            display: flex;
            align-items: baseline;
            justify-content: flex-start;
            gap: 2rem; /* Espaçamento entre o ícone e o texto */
            border-radius: 10px; /* Arredonda os cantos dos links */
             transition: background-color 0.3s, color 0.3s, text-shadow 0.3s
ease; /* Transição suave */
        }

        .sidenav a i {
            font-size: 22px; /* Ajusta o tamanho dos ícones */
            color: #FFD43B;
            transition: text-shadow 0.3s ease, transform 0.3s ease; /* Suaviza a
transição */
        }

        .sidenav a:hover {
            background-color: #dd3939; /* Cor de fundo no hover */
            color: #FFD43B; /* Muda a cor do texto no hover */
            text-decoration: none;

            /* Efeitos no texto e no ícone */
            text-shadow:  0 0 2.5px #FFD43B, 0 0 5px #FFD43B, 0 0 10px #FFD43B;
        }

        .sidenav a:hover i {
            text-shadow: 0 0 2.5px #FFD43B, 0 0 5px #FFD43B, 0 0 10px #FFD43B;
/* Brilho no ícone */
```

```
}


.sidenav hr {
    border: 0.5px solid #ffffff66; /* Linhas divisórias mais suaves */
    width: 80%;
    margin: 10px auto;
}

.sidenav .closebtn {
    position: absolute;
    top: 0;
}

.nav-button{
    display: flex;
    justify-content: space-between;
}

hr {
    background-color: rgb(255, 255, 255);
}

.car-icon {
    box-shadow: 4px 4px 8px rgba(0, 0, 0, 0.2);
    border-radius: 40px;
    transition: transform 0.4s ease, box-shadow 0.4s ease;
    background-color: white;
    height: 70px;
    width: 70px;
    display: flex;
    align-items: center;
    justify-content: center;
}

.car-icon:hover {
    transform: scale(1.1);
    box-shadow: 4px 4px 9px rgba(0, 0, 0, 0.2);
}

.car-icon:active {
    background-color: rgb(250, 250, 250);
}

.car-icon img {
    height: 50px;
}
```

```
      footer {
          position: fixed;
          bottom: 0;
          margin-left: 370px;
          margin-bottom: 10px;
          transition: transform 0.4s ease, box-shadow 0.4s ease;
      }

      footer:hover {
          transform: translate(-50px, 0px);
          /* Move a div 20px para a direita e para baixo */
      }

      footer section {
          display: flex;
          justify-content: flex-end;
          margin-right: 15px;
          margin-bottom: 10px;
      }

      input:checked+.slider:before {
          left: calc(100% - (var(--size-of-icon, 1.4em) + var(--slider-offset,
0.3em)));
          background: #303136;
           /* change the value of second inset in box-shadow to change the
angle and direction of the moon  */
           box-shadow: inset -3px -2px 5px -2px #8983f7, inset -10px -4px 0 0
#a3dafb;
      }

      .carousel-inner {
          background-color: #ffbc0d;
      }

      .carousel-item img {
          height: 600px;
          margin-bottom: 20px;
      }

      .dark-mode {
          height: 290px;
          width: 380px;
          display: flex;
          justify-content: flex-end;
          align-items: flex-end;
      }

      .horario {
          width: 95%;
```

```
        height: 1.5rem;
        background-color: rgb(255, 154, 154);
        color: rgb(255, 0, 0);
        display: flex;
        justify-content: center;
        align-items: center;
        text-align: center;
        border-radius: 0.5rem;
    }

    .horario h2 {
        font-size: 0.6rem !important;
        display: flex;
        justify-content: center;
        text-align: center;
        align-items: center;
        width: 100%;
        margin: 0 !important;
        height: 100%;
    }

    .central {
        background-color: #ffbf00;
        padding: 0.5rem 0 0.5rem 0;
        width: 100%;
        height: 2.5rem;
        display: flex;
        justify-content: center;
        align-items: center;
    }

    .card-produtos span{
        margin-left: 10px;
    }




    /* From Uiverse.io by revanth-004 */
    /* COMMON STYLES*/
    .popup-container {
        position: fixed;        /* Fixa o elemento na tela */
        top: 80px;              /* Distância do topo da tela */
        left: 50%;              /* Centraliza horizontalmente */
        transform: translateX(-50%); /* Ajusta a posição para o centro */
         z-index: 1000;             /* Garante que fique sobre os outros
elementos */
        display: flex;
```

```
        justify-content: center;
        opacity: 0;            /* Inicialmente invisível */
         animation: slideIn 0.5s forwards; /* Adiciona animação ao aparecer
*/
    }

    /* Animação para fazer o alerta aparecer deslizando de cima */
    @keyframes slideIn {
        0% {
             transform: translate(-50%, -20px); /* Começa um pouco acima do
topo */
            opacity: 0;                         /* Inicialmente invisível */
        }
        100% {
            transform: translate(-50%, 0);     /* Termina na posição central
*/
            opacity: 1;                         /* Fica totalmente visível */
        }
    }

    .popup {
        margin: 0;           /* Remove margem para evitar deslocamento */
        box-shadow: 4px 4px 10px -10px rgba(0, 0, 0, 1);
        width: 300px;        /* Define a largura do popup */
        justify-content: space-around;
        align-items: center;
        display: flex;
        border-radius: 4px;
        padding: 5px 0;
        font-weight: 300;
    }


    .popup svg {
        width: 1.25rem;
        height: 1.25rem;
    }

    .popup-icon svg {
        margin: 5px;
        display: flex;
        align-items: center;
    }

    .close-icon {
        margin-left: auto;
    }

    .close-svg {
```

```
        cursor: pointer;
    }

    .close-path {
        fill: grey;
    }

    /*SEPERATE STYLES*/

    /* SUCCESS */
    .success-popup {
        background-color: #edfbd8;
        border: solid 1px #84d65a;
    }

    .success-icon path {
        fill: #84d65a;
    }

    .success-message {
        color: #2b641e;
    }

    /* ALERT */
    .alert-popup {
        background-color: #fefce8;
        border: solid 1px #facc15;
    }

    .alert-icon path {
        fill: #facc15;
    }

    .alert-message {
        color: #ca8a04;
    }

    /* ERROR */

    .error-popup {
        background-color: #fef2f2;
        border: solid 1px #f87171;
    }

    .error-icon path {
        fill: #f87171;
    }

    .error-message {
```

```
        color: #991b1b;
    }

    /* INFO */

    .info-popup {
        background-color: #eff6ff;
        border: solid 1px #1d4ed8;
    }

    .info-icon path {
        fill: #1d4ed8;
    }

    .info-message {
        color: #1d4ed8;
    }
```

```
    *{
        box-sizing: border-box;
        margin: 0;
        padding: 0;
    }


    .espacamento{

        display: none !important;
    }

    .titulo{
        margin-top: 2.5rem;
        display: flex;
        margin-left: 26px;
        margin-right: 20px;
        margin-bottom: 20px;
    }

    h2{
        font-weight: bolder !important;
        text-align: left;
    }

    .container-principal{
        display: flex;
        align-items: center;
        flex-direction: column;
```

```
}

form{
    display: flex;
    flex-direction: column;
}

form input{
    width: 23.7rem;
    height: 30px;
    padding: 30px;
    border-radius: 5px;
    border: 2px solid rgb(238, 238, 238);
    background-color: rgb(245, 245, 245);
    margin-top: 15px;
}

input::placeholder {
    color: #7e7d7d; /* Mude para a cor desejada */
}

.checkbox{
    display: flex;
    flex-direction: row;
    margin-top: 30px;
    margin-right: 204px;
    margin-bottom: 40px;
}

.container-principal button{
    padding: 15px;
    background-color: rgb(161, 23, 23);
    width: 23.7rem;
    border-radius: 6px;
    border: none;
    margin-top: 10px;
    color: white;
    display: flex;
    justify-content: center;
}

form input:focus{
    outline: none;
}
.formulario form{
    width: 30rem;
    height: 20rem;
    display: flex;
    align-items: center;
```

```
        flex-direction: column;
    }

    .select{
        margin-top: 20px;
        display: flex;
        flex-direction: column;
        align-items: center;
        justify-content: center;
        gap: 1rem;
    }

    .select select{
        width: 23.7rem;

        padding: 19px;
        border-radius: 5px;
        border: 2px solid rgb(238, 238, 238);
        background-color: rgb(245, 245, 245);

    }

    .login-btn{
        text-decoration: none;
        color: rgb(156, 0, 0);
    }

    .login-btn{
        text-decoration: none;
        color: rgb(156, 0, 0);
        text-decoration: none;
    }

    .login-btn:hover{
        text-decoration: none;
        color: rgb(156, 0, 0);
        text-decoration: none;
    }

    .cbc-cima{
        display: none   ;
    }

    .gambiarra{
        display: none;
    }

    .btn-voltar img{
        height: 20px;
```

```
    }

    .btn-voltar button{
        display: flex;
        align-items: center;
        justify-content: center;
        height: 40px;
        background-color: transparent;
        border: 1px solid rgb(236, 236, 236);
        width: 40px;
        border-radius: 6px;
        margin-top: 10px;
        margin-left: 26px;
    }

    option{
        background-color: rgb(245, 245, 245);
        color: #000;
        font-size: 8px; /* Tamanho da fonte das opções */

    }

    form a{
        margin-top: 10px;
        margin-bottom: 15px;
        margin-left: 240px;
        color: black;
    }

    form a:hover{
        color: black;
        text-decoration: none;
    }
    /* GUILO FEIO */


    .popup-container {
        position: fixed;        /* Fixa o elemento na tela */
        top: 30px !important;             /* Distância do topo da tela */
        left: 50%;             /* Centraliza horizontalmente */
        transform: translateX(-50%); /* Ajusta a posição para o centro */
         z-index: 1000;            /* Garante que fique sobre os outros
elementos */
        display: flex;
        justify-content: center;
        opacity: 0;            /* Inicialmente invisível */
         animation: slideIn 0.5s forwards; /* Adiciona animação ao aparecer
*/
```

```
}

.espacamento{
    display: none;
}
```

```
body {
    font-family: Arial, sans-serif;
    background-color: #ffffff;
    margin: 0;
    padding: 0;
    box-sizing: border-box;

}

main {
    max-width: 100vw;
    background-color: #ffffff;
    box-shadow: 0 2px 8px rgba(0, 0, 0, 0.1);
    min-height: 89vh;
}

.posicao-marmita {
    display: flex;
    width: 100%;
    height: 100%;
    flex-direction: column;
    flex-wrap: wrap;
    align-content: center;
    padding-top: 100px;

}

.produto {
    display: flex;
    flex-direction: column; /* Coloca os filhos em coluna */
    align-items: center; /* Centraliza horizontalmente */
    justify-content: space-between; /* Garante espaçamento adequado */
    background-color: #fff;
    border-radius: 12px;
    box-shadow: 0 4px 10px rgba(0, 0, 0, 0.1);
    padding: 20px;
    max-width: 600px;
    width: 80%; /* Garante que o conteúdo ocupe a largura disponível */
    overflow: hidden; /* Evita que elementos escapem */
    position: relative; /* Permite controle melhor dos filhos */
    box-sizing: border-box;
```

```
}

.foto-produto {
    width: 100%;
    height: auto;
    margin-bottom: 20px; /* Espaço abaixo da imagem */
    display: flex;
    flex-direction: column;

}

.ft-item {
    max-width: 100%; /* A imagem ocupa 100% da largura do container */
    height: auto; /* Mantém a proporção da imagem */
    border-radius: 12px; /* Bordas arredondadas para a imagem */
}

.descricao {
    height: 60%;
    display: flex;
    flex-direction: column;
    text-align: center; /* Centraliza o texto da descrição */
    align-items: center;
    width: 85%;

}

.nome-produto {
    font-size: 2rem; /* Tamanho grande para o nome do produto */
    font-weight: bold;
    color: #333;

}

.descricao-produto {
    font-size: 1rem;
    color: #666;

}

.preco-produto {
    font-size: 1.5rem;
    font-weight: bold;
    color: #28a745; /* Verde para o preço */
    margin-bottom: 40px;

}
```

```
a {
    text-decoration: none !important;
}

.btn-adicionar {
    display: flex;
    background-color: red; /* Fundo vermelho */
    color: white; /* Texto branco */
    border: none; /* Sem borda */
    border-radius: 5px; /* Bordas arredondadas */
    padding: 10px 20px; /* Espaçamento interno */
    cursor: pointer;
    transition: background-color 0.3s; /* Transição suave */
    align-content: center;
    justify-content: center;
    flex-wrap: wrap;
    height: 60px;
}

.btn-adicionar:hover {
    background-color: #c82333; /* Cor mais escura ao passar o mouse */
}

.opc{
    width: 100%;
}

/* Seção de avaliação por estrelas */
.rating {
    display: inline-block;
}

.rating input {
    display: none;
}

.rating label {
    float: right;
    cursor: pointer;
    color: #ccc;
    transition: color 0.3s;
}

.rating label:before {
    content: '\2605';
    font-size: 30px;
}

.rating input:checked~label,
```

```
.rating label:hover,
.rating label:hover~label {
    color: #ffbb00;
    transition: color 0.3s;
}

/* Botão de adicionar ao carrinho */
form button {
    display: inline-block;
    padding: 12px 25px;
    background-color: #ff0000;
    color: #fff;
    font-size: 17px;
    font-weight: bold;
    border: none;
    border-radius: 5px;
    cursor: pointer;
    transition: background-color 0.3s ease;
}

form button:active {
    background-color: #bb0606;
    border: none;
}

.sair {
    display: flex;
    align-items: center;
    justify-content: space-between;
    width: 85%;
    padding-top: 10px;
    background-color: white;
    border: 1px solid rgb(255, 255, 255);

}



a {
    text-decoration: none !important;
}

.btn-voltar {
    display: flex;
    align-content: center;
    justify-content: center;
    color: red; /* Cor do texto */
    background-color: transparent; /* Fundo transparente */
    border: 2px solid red; /* Borda vermelha */
```

```
        border-radius: 5px; /* Bordas arredondadas */
        padding: 10px 20px; /* Espaçamento interno */
        cursor: pointer;
        flex-wrap: wrap;
        height: 60px;
        transition: background-color 0.3s, color 0.3s; /* Transição suave */
        font-weight: bolder;
        font-size: 18px;
    }

    .btn-voltar:hover {
        color: red;
    }


    #adicionarCarrinhoBtn{
        display: flex;
        background-color: red;
        color: white;
        border: none;
        border-radius: 5px;
        padding: 10px 20px;
        cursor: pointer;
        transition: background-color 0.3s;
        align-content: center;
        justify-content: center;
        flex-wrap: wrap;
        height: 60px;
        font-weight: bolder;
        font-size: 18px;
    }

    #adicionarCarrinhoBtn:hover {
        background-color: #c82333; /* Cor mais escura ao passar o mouse */
    }

    .espacamento1 {
        width: 100%;
        height: 80px;
        background-color: white;
    }

    .cima {
        display: flex;
        justify-content: space-between;
    }

    .observacao img {
        height: 25px;
```

```
}

.observacao input {
    width: 90vw;
    border: 1px solid rgb(212, 212, 212);
    padding: 10px;
}


.acompanhamentos {
    width: 100%;
    border-radius: 5px;
}

.header {
    background-color: #e0e0e0;
    /* Fundo cinza claro */
    padding: 10px;
    border-radius: 5px;
    display: flex;
    justify-content: space-between;
    align-items: center;
}

.header h3 {
    font-size: 18px;
    margin: 0;
    color: #333;
    font-weight: 900;
}

.header p {
    font-size: 12px;
    margin: 5px 0 0 0;
    color: #666;
}

.titulosubtitulo{
    margin-left: 11px;
}

.seta {
    margin-right: 10px;
    cursor: pointer;
    font-size: 20px;
    transition: transform 0.3s;
}

.opcoes-acompanhamentos {
```

```
        display: flex;
        flex-direction: column;
    }

    .opcoes-guarnicoes {
        display: flex;
        flex-direction: column;
    }

    .acompanhamento {
        display: flex;
        align-items: center;
        padding: 10px;
        border-top: 1px solid rgb(233, 231, 231);
    }

    .acompanhamento2 {
        display: flex;
        align-items: center;
        padding: 10px;
        border-top: 1px solid rgb(233, 231, 231);
    }

    .acompanhamento img {
        width: 40px;
        /* Tamanho da imagem */
        height: 40px;
        margin-left: 11px;
        margin-right: 10px;
        /* Espaço entre imagem e texto */
        border-radius: 5px;
        border: 0.5px solid red;
    }

    .acompanhamento2 img {
        width: 40px;
        /* Tamanho da imagem */
        height: 40px;
        margin-left: 11px;
        margin-right: 10px;
        /* Espaço entre imagem e texto */
        border-radius: 5px;
        border: 0.5px solid red;
    }

    .acompanhamento label {
        position: relative;
        margin-left: auto;
        /* Para alinhar a bolinha à direita */
```

```
        cursor: pointer;
        padding-left: 30px;
        /* Espaço entre o texto e a bolinha */
        z-index: 1;
        margin-top: 10px;
    }

    .acompanhamento2 label {
        position: relative;
        margin-left: auto;
        /* Para alinhar a bolinha à direita */
        cursor: pointer;
        padding-left: 30px;
        /* Espaço entre o texto e a bolinha */
        z-index: 1;
        margin-top: 10px;
    }

    .acompanhamento label span {
        position: absolute;
        left: 0;
        top: 50%;
        transform: translateY(-50%);
        width: 20px;
        height: 20px;
        border: 2px solid #ff0000;
        /* Cor da bolinha */
        border-radius: 50%;
        background-color: white;
        /* Fundo da bolinha */
        transition: background-color 0.3s, border-color 0.3s;
    }

    .acompanhamento2 label span {
        position: absolute;
        left: 0;
        top: 50%;
        transform: translateY(-50%);
        width: 20px;
        height: 20px;
        border: 2px solid #ff0000;
        /* Cor da bolinha */
        border-radius: 50%;
        background-color: white;
        /* Fundo da bolinha */
        transition: background-color 0.3s, border-color 0.3s;
    }
    /* Para ajustar o alinhamento da bolinha ao texto */
    .acompanhamento span {
```

```
        flex-grow: 1;
        font-size: 18px;
        /* Tamanho do texto do acompanhamento */
        margin-right: 5px;
        /* Espaço entre o texto e a bolinha */
    }

    .acompanhamento2 span {
        flex-grow: 1;
        font-size: 18px;
        /* Tamanho do texto do acompanhamento */
        margin-right: 5px;
        /* Espaço entre o texto e a bolinha */
    }
    input[type="checkbox"]{
        display: none;
    }


    input[type="checkbox"]+span {

        transform: translateY(-50%);
        border: 2px solid #ff0000;
        /* Cor da bolinha */
        border-radius: 50%;
        background-color: white;
        /* Fundo da bolinha */
        transition: background-color 0.3s, border-color 0.3s;
    }

    input[type="checkbox"]:checked+span {
        background-color: #ff0000;
        /* Cor quando selecionado */
        border-color: #ff0000;
        /* Cor da borda quando selecionado */
    }









    .acompanhamento2 label {
        position: relative;
        padding-left: 30px; /* Espaçamento para a bolinha */
        cursor: pointer;
    }
```

```
    .acompanhamento2 label span {
        position: absolute;
        left: 0;
        top: 50%;
        transform: translateY(-50%);
        width: 20px; /* Tamanho da bolinha */
        height: 20px; /* Tamanho da bolinha */
        border-radius: 50%; /* Torna a bolinha redonda */
        background-color: white; /* Cor de fundo da bolinha */
        border: 2px solid #ff0000; /* Borda vermelha */
        transition: background-color 0.3s, border-color 0.3s;
    }

    input[type="radio"] {
        display: none; /* Oculta o input padrão */
    }

    input[type="radio"]:checked + span {
        background-color: #ff0000; /* Cor quando selecionado */
        border-color: #ff0000; /* Cor da borda quando selecionado */
    }


    .opcoes-acompanhamentos {
        display: flex; /* Mantenha como flex */
    }







    .opcoes-acompanhamentos, .opcoes-guarnicoes {
        max-height: 500px; /* Define uma altura máxima para a animação */
        overflow: hidden; /* Esconde o conteúdo que transborda */
        transition: max-height 0.5s ease-out; /* Transição suave */
    }

    .opcoes-acompanhamentos.hidden, .opcoes-guarnicoes.hidden {
        max-height: 0; /* Altura 0 para esconder */
    }

    .seta {
        display: inline-block;
          transition: transform 0.3s ease; /* Transição suave para a
transformação */
    }
```

```
      .seta.rotated {
           transform: rotate(90deg); /* Rotaciona a seta 90 graus para a
direita */
      }
```

```
      * {
          box-sizing: border-box;
          margin: 0;
          padding: 0;
      }

      body {
          background-color: white;
          color: black;
      }

      body.dark {
          background-color: rgb(0, 0, 0);
          color: rgb(255, 255, 255);
      }

      h2 {
          margin-top: 30px;
          margin-bottom: 20px;
          font-size: 40px;
          font-weight: bold;
          font-style: italic;
           font-family: system-ui, -apple-system, BlinkMacSystemFont, 'Segoe
UI', Roboto, Oxygen, Ubuntu, Cantarell, 'Open Sans', 'Helvetica Neue', sans-
serif;
      }

      .container-principal {
          display: flex;
          align-items: center;
          flex-direction: column;
      }

      .form {
          margin-top: 30px;
          display: flex;
          flex-direction: column;
          position: relative;
          /* superior | direita | inferior | esquerda */
          padding: 150px 20px 20px 20px;
          border: 1px solid rgb(0 0 0);
```

```
        /* border-radius: 15px; */
        background-color: white;
        width: 300px;
        margin-top: -30px;
    }

    .form input {
        width: 100%;
        border: none;
        border-bottom: 1px solid black;
        margin-bottom: 40px;
        padding-right: 40px;
        background-color: transparent;
    }

    input.dark2 {
        border-bottom: 1px solid rgb(255, 255, 255);
        color: white;
    }

    .form.dark1 {
        border: 1px solid rgb(255, 255, 255);
        background-color: rgb(0, 0, 0);
    }

    .form input:focus {
        outline: none;
    }

    .btn-confirmar {
        padding: 10px 2px;
        background-color: rgba(207, 34, 38, 255);
        width: 150px;
        border-radius: 6px;
        border: none;
        margin-bottom: 10px;
        margin-top: -10px;
        color: white;
        font-weight: 700;
        font-style: italic;
        align-self: center;
    }

    /* Estilo da imagem de perfil e sobreposição */
    .profile-container {
        z-index: 1;
        position: relative;
        height: 150px;
        width: 150px;
```

```
            border-radius: 100px;
            overflow: hidden;
              margin-bottom:  -75px;  /*  Ajuste  para  que  o  formulário  fique
sobreposto */
        }

        .profile-image {
            width: 100%;
            height: 100%;
            background-image: url('../img/default-avatar.png');
            background-size: cover;
            background-position: center;
            overflow: hidden;
            transition: filter 0.5s ease;
        }

        .profile-image img{

            object-fit: cover;
            margin: 0 auto;
            width: 100%;
            height: 100%;
        }

        .overlay {
            position: absolute;
            top: 0;
            left: 0;
            width: 100%;
            height: 100%;
            display: flex;
            align-items: center;
            justify-content: center;
            opacity: 0;
            transition: opacity 0.5s ease;
        }

        .overlay-image {
            width: 80%;
            height: auto;
        }

        .profile-container:hover .profile-image {
            filter: blur(3px) grayscale(30%);
        }

        .profile-container:hover .overlay {
            opacity: 1;
        }
```

```
    .overlay input[type="file"] {
        position: absolute;
        top: 0;
        left: 0;
        z-index: -1;
        width: 100%;
        height: 100%;
        opacity: 0;
        cursor: pointer;
    }

    .toggle-button, .toggle-button2 {
        z-index: 0;
        position: absolute;
        right: 22px;
        top: 50%;
        transform: translateY(-50%);
        border: none;
        background-color: transparent;
        cursor: pointer;
        padding: 0;
    }

    .toggle-button2 {
        top: calc(100% - 151px);
    }

    .overlay input{

        z-index: 1000 !important;
    }
```

```
        /* Definições globais */
    /* Definições globais */
    body {
         font-family: system-ui, -apple-system, BlinkMacSystemFont, 'Segoe UI', Roboto, Oxygen, Ubuntu, Cantarell, 'Open Sans', 'Helvetica Neue', sans-serif;
        margin: 0;
        padding: 0;
        box-sizing: border-box;
    }

    /* Estilos da área de cabeçalho (imagem de fundo) */
    .header {
        position: relative;
```

```
            width: 100%;
            height: 150px; /* Ajuste a altura conforme necessário */
                                                        background-image:
url('https://br.web.img2.acsta.net/r_654_368/img/12/ce/12ceb64b011a2a629879b4d
3b0a256af.png');  /* Coloque o caminho da sua imagem de fundo */
            background-size: cover;
            background-position: center;
        }

        /* Imagem de perfil */
        .img-perfil {
            position: absolute;
            top: 100%;
            left: 50%;
            transform: translate(-50%, -50%);
            width: 150px;  /* Tamanho da imagem de perfil */
            height: 150px;
            border-radius: 50%;
            border: 4px solid white;  /* Borda branca */
             z-index: 1;  /* Garante que a imagem de perfil fique sobreposta à
imagem de fundo */
              box-shadow: 0 0 10px rgba(0, 0, 0, 0.3);  /* Sombra leve para
destacar a imagem */
        }

        /* Conteúdo abaixo da imagem de perfil */
        .perfil-info {
            text-align: center;
              margin-top: 80px;  /* Ajuste o valor para colocar o conteúdo
corretamente abaixo da imagem de perfil */
        }

        /* Nome e curso do cliente */
        .perfil-info .nome {
            font-size: 24px;
            font-weight: bold;

        }

        .perfil-info .curso {
            font-size: 18px;
            color: #666;
        }

        /* Subtítulo "Informações Pessoais" */
        .informacoes-pessoais {
            font-weight: bold;
            font-size: 20px;
            margin-top: 40px;
```

```
            margin-bottom: 15px;
        }

        /* Estilo dos dados (telefone e email) */
        .dados-pessoais {
            font-size: 16px;
            color: #444;
        }

        .relatorio-vendas {
            margin-top: 30px;
            width: auto;
            max-width: 340px; /* Largura do recibo */
            margin-right: auto;
            margin-left: auto;
            padding: 10px;
             background: url('../img/fundoNotafiscal.jpg'); /* Fundo branco com
transparência */
            border: 1px solid #ccc;
            border-radius: 5px;
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.5);
            margin-bottom: 20px;
        }

        .resumo-relatorio{
            display: flex;
            justify-content: center;
            font-weight: bolder;
            flex-direction: column;
        }


        h2, h3, h4 {
            text-align: center;
            margin: 5px 0;
             font-family: 'receipt', 'Anonymous Pro', 'Courier New', Courier,
monospace;
        }

        .filtro-relatorio {
            margin-bottom: 20px;
            padding: 5px;
        }

        .label-data-inicial,
        .label-data-final {
            display: block;
            margin: 5px 0;
        }
```

```
.data-inicial,
.data-final {
    width: 100%;
    padding: 5px;
    margin-bottom: 10px;
}

.botao-gerar-relatorio {
    padding: 15px;
    background-color: rgb(161, 23, 23);
    width: 23.7rem;
    border-radius: 6px;
    border: none;
    cursor: pointer;
    margin-top: 10px;
    color: white;
}

table {
    width: 100%;
    border-collapse: collapse;
}

th, td {
    padding: 5px;
    text-align: left;
}

th {
    font-weight: normal;
    border-bottom: 1px dashed black;
    border-top: 1px dashed black;

}
tr{
    margin-bottom: 10px;
    border-bottom: 1px dashed black;
}
h4 {
    margin: 10px 0;
    text-align: center;
}

h2{
    text-decoration: underline;
}

h3{
```

```
        font-weight: bolder;
        font-family: 'Helvetica Neue', Helvetica, sans-serif;
    }

    .spanpontilhado{
        border-bottom: 1px dashed black;
        border-top: 1px dashed black;
    }

    ul {
        list-style-type: none; /* Remove as bolinhas dos itens da lista */
        margin-top: 15px;
      }
```

```
    /* Estilos gerais */
    body {
        font-family: Arial, sans-serif;
        background-color: #ffffff;
        margin: 0;
        padding: 0;
        box-sizing: border-box;

    }

    main {
        max-width: 100vw;
        background-color: #ffffff;
        box-shadow: 0 2px 8px rgba(0, 0, 0, 0.1);
        height: 89vh;
    }

    .posicao-produto {
        display: flex;
        width: 100%;
        height: 100%;
        flex-direction: column;
        flex-wrap: wrap;
        align-content: center;
        padding-top: 100px;
    }

    .produto {
        display: flex;
        flex-direction: column; /* Coloca a imagem acima da descrição */
        align-items: center; /* Centraliza horizontalmente */
        background-color: #fff;
        border-radius: 12px;
```

```
        box-shadow: 0 4px 10px rgba(0, 0, 0, 0.1);
        padding: 20px;
        height: auto;
        max-width: 600px; /* Largura máxima do container */

    }

    .foto-produto {
        width: 100%;
        height: auto;
        margin-bottom: 20px; /* Espaço abaixo da imagem */
        display: flex;
        flex-direction: column;

    }

    .ft-item {
        max-width: 100%; /* A imagem ocupa 100% da largura do container */
        height: auto; /* Mantém a proporção da imagem */
        border-radius: 12px; /* Bordas arredondadas para a imagem */
    }

    .descricao {
        height: 60%;
        display: flex;
        flex-direction: column;
        text-align: center; /* Centraliza o texto da descrição */
        align-items: center;
    }

    .nome-produto {
        font-size: 2rem; /* Tamanho grande para o nome do produto */
        font-weight: bold;
        color: #333;

    }

    .descricao-produto {
        font-size: 1rem;
        color: #666;

    }

    .preco-produto {
        font-size: 1.5rem;
        font-weight: bold;
        color: #28a745; /* Verde para o preço */
        margin-bottom: 40px;
```

```
    }

    .sair {
        display: flex;
        justify-content: center; /* Alinha os botões horizontalmente */
         align-items: center; /* Centraliza os elementos dentro da div sair
*/
        gap: 20px; /* Espaçamento entre os botões */
    }

    a {
        text-decoration: none !important;
    }

    .btn-voltar {
        display: flex;
        align-content: center;
        justify-content: center;
        color: red; /* Cor do texto */
        background-color: transparent; /* Fundo transparente */
        border: 2px solid red; /* Borda vermelha */
        border-radius: 5px; /* Bordas arredondadas */
        padding: 10px 20px; /* Espaçamento interno */
        cursor: pointer;
        flex-wrap: wrap;
        height: 60px;
        transition: background-color 0.3s, color 0.3s; /* Transição suave */
        font-weight: bolder;
        font-size: 18px;
    }

    .btn-voltar:hover {
        color: red;
    }

    .btn-adicionar {
        display: flex;
        background-color: red; /* Fundo vermelho */
        color: white; /* Texto branco */
        border: none; /* Sem borda */
        border-radius: 5px; /* Bordas arredondadas */
        padding: 10px 20px; /* Espaçamento interno */
        cursor: pointer;
        transition: background-color 0.3s; /* Transição suave */
        align-content: center;
        justify-content: center;
        flex-wrap: wrap;
        height: 60px;
    }
```

```
.btn-adicionar:hover {
    background-color: #c82333; /* Cor mais escura ao passar o mouse */
}


/* Seção de avaliação por estrelas */
.rating {
  display: inline-block;
}

.rating input {
  display: none;
}

.rating label {
  float: right;
  cursor: pointer;
  color: #ccc;
  transition: color 0.3s;
}

.rating label:before {
  content: '\2605';
  font-size: 30px;
}

.rating input:checked ~ label,
.rating label:hover,
.rating label:hover ~ label {
  color: #ffbb00;
  transition: color 0.3s;
}

/* Botão de adicionar ao carrinho */
form button {
    display: inline-block;
    padding: 10px 30px;
    background-color: #ff0000;
    color: #fff;
    font-size: 18px;
    font-weight: bold;
    border: none;
    border-radius: 5px;
    cursor: pointer;
    transition: background-color 0.3s ease;
}

form button:active {
```

```
        background-color: #bb0606;
        border: none;
    }



    .espacamento{
        width: 100%;
        height: 50px;
        background-color: white;
    }

    .cima{
        display: flex;
        justify-content: space-between;
    }

    .observacao img{
        height: 25px;
    }

    .observacao input{
        width: 90vw;
        border: 1px solid rgb(212, 212, 212);
        padding: 10px;
    }
```

```
    * {
        box-sizing: border-box;
        margin: 0;
        padding: 0;
    }

    body {
        font-family: Arial, sans-serif;
        background-color: #f9f9f9;
        color: #333;
        margin: 0;
        padding: 0;
    }

    .order-filter {
        display: flex;
        justify-content: center;
        margin-bottom: 20px;
        flex-direction: column;
```

```
        align-items: center;
    }

    .order-filter label {
        margin-right: 10px;
        font-size: 1.1em;
    }

    .order-filter select {
        padding: 5px;
        font-size: 1.1em;
        border-radius: 5px;
        border: 1px solid #ccc;
        transition: border-color 0.3s ease;
    }

    .order-filter select:hover {
        border-color: #888;
    }

    .card-pedidos {

        margin: 20px 0; /* Adiciona margem superior e inferior */
        padding: 16px;

    }


    .order-card {
        border: 1px solid #e0e0e0; /* Borda para o card do pedido */
        border-radius: 8px; /* Bordas arredondadas para o card do pedido */
        padding: 20px; /* Espaçamento interno do card */
        margin-bottom: 15px; /* Margem inferior para separar os pedidos */
        background-color: #f1f1f1; /* Fundo claro para o card do pedido */
    }

    .order-card:hover {
        transition: transform 0.3s;
        transform: translateY(-5px);
    }

    .order-card h2 {
        margin: 0;
        font-size: 24px;
        color: #333;
    }

    .order-card a {
        color: #3498db;
```

```
            text-decoration: none;
            font-weight: bold;
        }

        .order-header {
            display: flex;
            justify-content: space-between;
            margin-top: 10px;
            font-size: 18px;
            color: #555;
        }

        .order-details {
            margin-top: 10px;
            /* padding: 10px; */
            background-color: #f1f1f1; /* Fundo claro para os detalhes do pedido
*/
            border-radius: 8px;
        }

        .order-details h3 {
            margin: 0 0 8px;
            font-size: 18px;
            color: #444;
        }

        .order-details ul {
            margin: 0;
            padding-left: 20px;
            list-style-type: none;
            color: #797979;
            padding: 0;
            font-size: 18px;
            font-weight: bold;
        }

        .order-details li {
            padding-bottom: 10px;
        }

        .order-status {
            margin: 12px 0 0 0;
            font-size: 16px;
            font-weight: bold;
            color: #333;
        }

        .btn-cancelar {
            margin-top: 15px;
```

```
        display: flex;
        gap: 10px;
        justify-content: space-between;
        flex-direction: row;
        align-items: center;
        flex-wrap: wrap;

    }

    .status-btn, .cancel-btn {
        padding: 12px 23px;
        border-radius: 6px;
        border: none;
        font-weight: bold;
        font-size: 18px;
        cursor: pointer;
        transition: background-color 0.3s, transform 0.2s;
    }

    .status-btn {
        background-color: #3498db;
        color: #fff;
    }

    .status-btn:hover {
        background-color: #2980b9;
    }

    .cancel-btn {
        background-color: #e74c3c;
        color: #fff;
    }

    .cancel-btn:hover {
        background-color: #c0392b;
    }

    .status-btn:active, .cancel-btn:active {
        transform: scale(0.95);
    }

    .preparando {
        color: #525305; /* Amarelo */
        background-color: #f6ffa6;
    }

    .pronto {
        color: #20587e; /* Azul */
        background-color: #d9edf7;
```

```
}

.entregue {
    color: #103b23; /* Verde bem escuro */
    background-color: #c1d6ae; /* Verde muito suave quase branco */
}

.cancelado {

    color: #e74c3c; /* Vermelho */
    background-color: #f5c6cb;
}


.preparando:hover {
    color: #ffffff; /* Amarelo */
    background-color: #c1c40c;

}

.pronto:hover {
    color: #ffffff; /* Azul */
    background-color: #3498db;
}

.entregue:hover {
    color: #ffffff; /* Verde bem escuro */
    background-color: #248b51; /* Verde muito suave quase branco */
}

.cancelado:hover {
    color: #e74c3c; /* Vermelho */
    background-color: #f5c6cb;
}

/* Media Queries para Responsividade */
@media (max-width: 768px) {
    .order-filter {
        flex-direction: column;
        align-items: flex-start;
    }

    .order-filter label {
        margin-bottom: 10px; /* Espaçamento inferior nos rótulos */
    }

    .order-filter select {
        width: 100%; /* Largura total em telas pequenas */
        margin-bottom: 10px; /* Espaço inferior para o seletor */
```

```
        }

        .card-pedidos {
            padding: 12px; /* Menos padding nos cards */
        }

        .order-card h2 {
            font-size: 20px; /* Reduzir tamanho do título */
        }

        .order-header {
            flex-direction: column; /* Colocar os elementos em coluna */
            align-items: flex-start; /* Alinhar à esquerda */
            margin-top: 8px; /* Menos margem superior */
        }

        .order-details h3 {
            font-size: 16px; /* Tamanho menor para detalhes */
        }

        .order-status {
            font-size: 14px; /* Tamanho menor para status */
        }

        .status-btn, .cancel-btn {
            padding: 12px 24px;
            font-size: 12px; /* Tamanho menor para os botões */
        }
    }

    @media (max-width: 480px) {
        .order-filter label {
            font-size: 1em; /* Menor tamanho em telas muito pequenas */
        }

        .order-filter select {
            font-size: 1em; /* Menor tamanho para o seletor */
        }

        .order-card h2 {
            font-size: 23px; /* Tamanho menor do título */
        }

        .order-header {
            display: flex;
            flex-direction: row;
            font-size: 18px; /* Reduzir ainda mais o tamanho */
            font-weight: bold;
        }
```

```
        .order-details h3 {
            font-size: 23px; /* Tamanho menor para detalhes */
            font-weight: bold;
        }

        .order-status {
            font-size: 20px; /* Tamanho menor para status */
        }
    }

    .modal {
        display: none; /* Inicialmente escondida */
        position: fixed; /* Fixa na tela */
        top: 0;
        left: 0;
        z-index: 1000; /* Acima de outros elementos */
        width: 100%; /* Largura total */
        height: 100%; /* Altura total */
        overflow: auto; /* Adiciona rolagem se necessário */
         background-color: rgba(0, 0, 0, 0.7); /* Fundo mais escuro e semi-
transparente */
        justify-content: center; /* Centraliza horizontalmente */
        align-items: center; /* Centraliza verticalmente */
    }

    .modal-content {
        background-color: #fefefe; /* Fundo mais claro */
        padding: 20px; /* Espaçamento interno */
        border: 1px solid #ccc; /* Borda mais suave */
        border-radius: 15px; /* Bordas arredondadas */
        width: 90%; /* Ajustável conforme necessário */
        max-width: 400px; /* Largura máxima para telas grandes */
          box-shadow: 0 8px 16px rgba(0, 0, 0, 0.3); /* Sombra mais
pronunciada */
        animation: fadeIn 0.3s ease; /* Animação suave ao abrir */
    }

    /* Animação de suavidade para o modal */
    @keyframes fadeIn {
        from {
            opacity: 0;
            transform: scale(0.9);
        }
        to {
            opacity: 1;
            transform: scale(1);
        }
    }
```

```css
/* Estilo do botão de fechar */
.close {
    color: #888; /* Cor da cruz */
    float: right; /* Alinha à direita */
    font-size: 28px; /* Tamanho da fonte */
    font-weight: bold; /* Negrito */
    cursor: pointer; /* Cursor de ponteiro */
}

.close:hover,
.close:focus {
    color: #333; /* Cor ao passar o mouse */
}

/* Estilo do título */
h2 {
    margin: 0 0 15px; /* Margem abaixo do título */
    font-family: 'Arial', sans-serif; /* Fonte do título */
    color: #333; /* Cor do título */
}

/* Estilo do select */
select {
    width: 100%; /* Largura total */
    padding: 10px; /* Espaçamento interno */
    margin-bottom: 15px; /* Margem abaixo do select */
    border: 1px solid #ccc; /* Borda suave */
    border-radius: 5px; /* Bordas arredondadas */
    background-color: #f9f9f9; /* Fundo do select */
    font-size: 16px; /* Tamanho da fonte */
    color: #333; /* Cor do texto */
}

/* Estilo do botão de confirmação */
#confirmCancelBtn {
    margin-top: 0.6rem;
    background-color: rgb(161, 23, 23); /* Cor verde */
    color: white; /* Cor do texto */
    padding: 10px; /* Espaçamento interno */
    border: none; /* Sem borda */
    border-radius: 5px; /* Bordas arredondadas */
    cursor: pointer; /* Cursor de ponteiro */
    font-size: 16px; /* Tamanho da fonte */
    transition: background-color 0.3s; /* Transição suave */
    width: 100%; /* Largura total */
}
```

```
      #confirmCancelBtn:hover {
          background-color: rgb(146, 22, 22); /* Cor ao passar o mouse */
      }

      input{
          padding: 10px;
          border-radius: 5px;
          border: 2px solid rgb(238, 238, 238);
          background-color: rgb(245, 245, 245);
          margin-bottom: 10px;
      }
```

```
      .relatorio-vendas {
          width: 300px; /* Largura do recibo */
          margin: auto;
          padding: 10px;
           background: url('../img/fundoNotafiscal.jpg'); /* Fundo branco com
transparência */
          border: 1px solid #ccc;
          border-radius: 5px;
          box-shadow: 0 0 10px rgba(0, 0, 0, 0.5);
          margin-bottom: 20px;
      }

      h2, h3, h4 {
          text-align: center;
          margin: 5px 0;
      }

      h2, h3 {
          font-family: 'Helvetica Neue', Helvetica, sans-serif;
          font-weight: bolder;
      }

      h4 {
          margin: 10px 0;
          text-align: center;
      }

      .filtro-relatorio {
          margin-bottom: 20px;
          padding: 5px;
      }

      .label-data-inicial,
      .label-data-final {
```

```
        display: block;
        margin: 5px 0;
    }

    .data-inicial,
    .data-final {
        width: 100%;
        padding: 5px;
        margin-bottom: 10px;
    }

    .botao-gerar-relatorio {
        padding: 15px;
        background-color: rgb(161, 23, 23);
        width: 23.7rem;
        border-radius: 6px;
        border: none;
        cursor: pointer;
        margin-top: 10px;
        color: white;
    }

    table {
        width: 100%;
        border-collapse: collapse;
    }

    th, td {
        padding: 5px;
        text-align: left;

    }

    th {
        font-weight: normal;
        border-bottom: 1px dashed black;
        border-top: 1px dashed black;

    }
    tr{
        margin-bottom: 10px;
    }


    .spanpontilhado{
        border-bottom: 1px dashed black;
        border-top: 1px dashed black;
    }
```

```
*{
    box-sizing: border-box;
    margin: 0;
    padding: 0;
     font-family: system-ui, -apple-system, BlinkMacSystemFont, 'Segoe UI', Roboto, Oxygen, Ubuntu, Cantarell, 'Open Sans', 'Helvetica Neue', sans-serif;
}


.titulo{
    margin-top: 2.5rem;
    display: flex;
    margin-left: 26px;
    margin-right: 80px;
    margin-bottom: 20px;
    flex-direction: column;
}

h2{
    font-weight: bolder !important;
    text-align: left;
    font-size: 30px !important;

}

.container-principal{
    display: flex;
    align-items: center;
    flex-direction: column;
}

form{
    display: flex;
    flex-direction: column;
}

form input{
    width: 23.7rem;
    height: 30px;
    padding: 30px;
    border-radius: 5px;
    border: 2px solid rgb(238, 238, 238);
    background-color: rgb(245, 245, 245);
    margin-top: 15px;
}
```

```
input::placeholder {
    color: #7e7d7d; /* Mude para a cor desejada */
}

.checkbox{
    display: flex;
    flex-direction: row;
    margin-top: 30px;
    margin-right: 204px;
    margin-bottom: 40px;
}

.container-principal button{
    padding: 15px;
    background-color: rgb(161, 23, 23);
    width: 23.7rem;
    border-radius: 6px;
    border: none;
    font-size: 15px;
    margin-top: 50px;
    color: white;
}

form input:focus{
    outline: none;
}
.formulario form{
    width: 30rem;
    height: 30rem;
    display: flex;
    align-items: center;
    flex-direction: column;
}

.select{
    margin-top: 20px;
    display: flex;
    flex-direction: column;
    align-items: center;
    justify-content: center;
    gap: 1rem;
}

.select select{
    width: 23.7rem;

    padding: 19px;
    border-radius: 5px;
```

```
        border: 2px solid rgb(238, 238, 238);
        background-color: rgb(245, 245, 245);

    }

    .login-btn{
        text-decoration: none;
        color: rgb(156, 0, 0);
    }

    .login-btn{
        text-decoration: none;
        color: rgb(156, 0, 0);
        text-decoration: none;
    }

    .login-btn:hover{
        text-decoration: none;
        color: rgb(156, 0, 0);
        text-decoration: none;
    }

    .cbc-cima{
        display: none   ;
    }

    .gambiarra{
        display: none;
    }

    .btn-voltar img{
        height: 20px;

    }

    .btn-voltar button{
        display: flex;
        align-items: center;
        justify-content: center;
        height: 40px;
        background-color: transparent;
        border: 1px solid rgb(236, 236, 236);
        width: 40px;
        border-radius: 6px;
        margin-top: 10px;
        margin-left: 26px;
    }

    option{
```

```
        background-color: rgb(245, 245, 245);
        color: #000;
        font-size: 8px; /* Tamanho da fonte das opções */

    }

    form a{
        margin-top: 10px;
        margin-bottom: 15px;
        margin-left: 240px;
        color: black;
    }

    form a:hover{
        color: black;
        text-decoration: none;
    }

    .semConta{
        margin-top: 15px;
    }

    .espacamento{
        display: none;
    }
```

```
    body {
        font-family: Arial, sans-serif;
        margin: 0;
        padding: 0;
    }

    .btn-voltar img {
        height: 20px;

    }

    .btn-voltar button {
        display: flex;
        align-items: center;
        justify-content: center;
        height: 40px;
        background-color: transparent;
        border: 1px solid rgb(236, 236, 236);
        width: 40px;
        border-radius: 6px;
        margin-top: 10px;
```

```
        margin-left: 26px;
    }

    .container {
        display: flex;
        flex-direction: column;
        max-width: 400px;
        margin: 40px auto;
        padding: 20px;
        border-radius: 10px;
    }

    .formulario {
        margin-bottom: 20px;
    }

    .formulario label {
        display: block;
        margin-bottom: 10px;
    }

    .formulario input[type="text"] {
        width: 100%;
        height: 40px;
        padding: 10px;
        border: 1px solid #ccc;
        border-radius: 5px;
    }

    .formulario button[type="submit"] {
        width: 100%;
        padding: 15px;
        background-color: rgb(161, 23, 23);
        border-radius: 6px;
        border: none;
        margin-top: 20px;
        color: white;
    }

    .codigo-verificacao {
        display: flex;
        gap: 10px;
        /* Espaçamento entre os cubos */
    }

    .digito {
        width: 40px;

        height: 50px;
```

```
        border: 2px solid #ccc;

        border-radius: 5px;

        text-align: center;

        font-size: 24px;

        background-color: rgb(245, 245, 245);
    }

    .titulo{
        font-size: 30px;
        font-weight: 700;
        margin-bottom: 5px;
    }

    .subtitulo{
        margin-bottom: 70px;

    }

    .telefone{
        margin-bottom: 10px;
    }
```

```
    @media (min-width: 768px) {
        .left-lado {
            display: flex;
            flex-direction: column;
            align-items: flex-start;
            font-size: 60px;
            flex: 1;
            flex-wrap: wrap;}

        .card-produtos img {
            height: 120px; /* Tamanho da imagem */
            width: 120px; /* Tamanho da imagem */
            border-radius: 12px; /* Bordas arredondadas */
            object-fit: cover;
        }

        .produto-nome {
            font-size: 1.6rem; /* Ajustado para mobile */
            font-weight: 600;
            color: #333;
```

```
            text-align: left;
        }

        .produto-desc {
            text-align: left;
            font-size: 1.2rem;
        }

        .produto-preco {
            text-align: left;
            font-size: 1.4rem;
            font-weight: 700;
            color: #727272 ;
        }

        .produtos button:hover {
            transform: scale(1.01);
        }

        h3{
            font-size: 1.8rem;
        }
    }
```

```
            /* Estilo para dispositivos maiores */
            @media (min-width: 768px) {
                .container {
                    flex-direction: row; /* Exibe os cards na horizontal */
                    flex-wrap: wrap; /* Permite quebra para a próxima linha,
se necessário */
                        justify-content: center; /* Centraliza os cards
horizontalmente */
                    gap: 20px; /* Espaçamento maior entre os cards */
                }

                .card {
                        width: calc(25% - 20px); /* Cada card ocupa 25% da
largura menos o gap */
                    max-width: 250px; /* Define uma largura máxima */
                }
            }
```

```
    @media (max-width: 768px) {
        .produto {
```

```
            width: 90%;
            max-width: 800px;
            /* Largura máxima para o container */
        }

        .foto-produto {
            width: 95%;
            height: auto;
            margin-bottom: 20px;
            /* Espaço abaixo da imagem */
            display: flex;
            flex-direction: column;
        }

        .nome-produto {
            font-size: 2.4rem;
            /* Tamanho maior para nome do produto */
        }

        .descricao-produto {
            font-size: 1.6rem;
            /* Tamanho maior para descrição */
        }

        .preco-produto {
            font-size: 1.8rem;
            /* Tamanho maior para o preço */
        }

        .observacao input {
            width: 100%;
            /* Aumenta a largura do campo de input */
        }

        .descricao {
            width: 100%;
            align-items: center;
        }
        .sair{
            width: 100%;
        }
    }
```

```
    /* Estilos gerais para tablets e desktops */
    @media (min-width: 768px) {
        .produto {
            width: 600px;
```

```
            max-width: 800px;
            /* Largura máxima para o container */
        }

        .foto-produto {
            width: 95%;
            height: auto;
            margin-bottom: 20px;
            /* Espaço abaixo da imagem */
            display: flex;
            flex-direction: column;
        }

        .nome-produto {
            font-size: 2.4rem;
            /* Tamanho maior para nome do produto */
        }

        .descricao-produto {
            font-size: 1.6rem;
            /* Tamanho maior para descrição */
        }

        .preco-produto {
            font-size: 1.8rem;
            /* Tamanho maior para o preço */
        }

        .observacao input {
            width: 100%;
            /* Aumenta a largura do campo de input */
        }

        .descricao {
            width: 100%;
            align-items: center;
        }
    }
```

```
    @media(min-width: 1000px){
        .card-pedidos{


            justify-content: center;
            align-items: center;
            display: grid;
```

```
            grid-template-columns: repeat(2, 1fr);

            gap: 150px;

    }}

    @media(min-width: 1300px){

        .card-pedidos{


            justify-content: center;
            align-items: center;
            display: grid;
            grid-template-columns: repeat(3, 1fr);

            gap: 75px;
```

APÊNDICE L – Requisitos funcionais

A.1 RF001 - Cadastrar Usuário

Quadro 1 - A.1 RF001 - Cadastrar Usuário

Descrição	O sistema deve permitir que um novo usuário se registre, fornecendo as informações necessárias para sua identificação e acesso ao sistema.
Pré-condições	• O usuário deve acessar a página ou tela de cadastro. • O sistema deve estar disponível e funcionando corretamente.
Entradas	• Nome completo • E-mail • Número de telefone • Senha • Curso
Saídas	• Mensagem de sucesso confirmando o cadastro • Criação de um novo registro no banco de dados com as

	informações do usuário
Pós-Condições	• O usuário cadastrado deve ser capaz de fazer login no sistema utilizando as credenciais fornecidas. • O sistema deve enviar um código de confirmação para o telefone cadastrado.
Prioridade	Essencial
Dependência	RNF001 - Criptografia de Senha

A.2 RF002 - Logar Usuário

Quadro 2 - A.2 RF002 - Logar Usuário

Descrição	O sistema deve permitir que um usuário previamente cadastrado acesse sua conta, fornecendo suas credenciais de login.
Pré-condições	• O usuário deve estar cadastrado no sistema. • O sistema deve estar disponível e funcionando corretamente. • O usuário deve acessar a página ou tela de login.
Entradas	• E-mail cadastrado • Senha cadastrada
Saídas	• Mensagem de sucesso confirmando o login • Acesso as funções com as funcionalidades disponíveis para o usuário logado.
Pós-Condições	• O sistema deve registrar o login do usuário. • As informações do usuário logado devem estar disponíveis para o sistema, permitindo a personalização da experiência do usuário.
Prioridade	Essencial
Dependência	RF001 (Cadastro de Usuário)

A.3 RF003 - Adicionar ao Carrinho

Quadro 3 - A.3 RF003 - Adicionar ao Carrinho

Descrição	O sistema deve permitir que o usuário adicione um ou mais produtos ao seu carrinho de compras, com a finalidade de realizar uma compra posteriormente.
Pré-condições	• O usuário deve estar navegando na página de um produto específico ou em uma lista de produtos. • O usuário deve estar logado no sistema. • O produto escolhido deve estar disponível em estoque.
Entradas	• ID do produto • Quantidade desejada do produto
Saídas	• Mensagem de sucesso confirmando a adição do produto ao carrinho • Atualização do carrinho de compras do usuário, exibindo os produtos adicionados, suas quantidades e valores.
Pós-Condições	• O produto adicionado deve ser armazenado no carrinho de compras do usuário, junto com as informações de quantidade e preço. • O sistema deve calcular o valor total do carrinho de compras e exibi-lo para o usuário.
Prioridade	Essencial
Dependência	RF001 (Cadastro de Usuário), RF002 (Logar Usuário)

A.4 RF004 - Enviar Pedido

Quadro 4 - A.4 RF004 - Enviar Pedido

Descrição	O sistema deve permitir que o usuário finalize a compra, enviando os produtos do carrinho para processamento.
Pré-condições	• O usuário deve ter pelo menos um produto adicionado ao carrinho. • O usuário deve estar logado no sistema.
Entradas	• Confirmação do pedido.
Saídas	• Mensagem de confirmação do pedido • Número do pedido
Pós-Condições	• O sistema deve criar um novo registro no banco de dados com as informações do pedido, incluindo produtos, quantidades, valores e hora do pedido. • O pedido vai para a lista de pedidos, com todas as informações.
Prioridade	Essencial
Dependência	RF001 (Cadastro de Usuário), RF002 (Logar Usuário), RF003 (Adicionar ao Carrinho)

A.5 RF005 - Adicionar Produtos ao Banco de Dados

Quadro 5 - A.5 RF005 - Adicionar Produtos ao Banco de Dados

Descrição	O sistema deve permitir que um administrador adicione novos produtos ao catálogo, incluindo suas características e informações relevantes para a venda.
Pré-condições	• O usuário deve estar logado no sistema com perfil de administrador. • O sistema deve ter um formulário de cadastro de produtos disponível.
Entradas	• Nome do produto • Descrição do produto

	• Preço do produto • Categoria do produto • Imagem do produto
Saídas	• Mensagem de sucesso confirmando a adição do produto ao banco de dados • O novo produto deve aparecer listado no catálogo do sistema
Pós-Condições	• As informações do produto devem ser armazenadas em um banco de dados, em uma tabela específica para produtos. • O sistema deve gerar um ID único para cada produto, para fins de identificação. • A imagem do produto deve ser salva em um local específico e seu caminho armazenado no banco de dados.
Prioridade	Essencial
Dependência	RF002 - Logar Usuário

A.6 RF006 - Remover Produtos do Banco de Dados

Quadro 6 - A.6 RF006 - Remover Produtos do Banco de Dados

Descrição	O sistema deve permitir que um administrador remova produtos do catálogo, quando necessário.
Pré-condições	• O usuário deve estar logado no sistema com perfil de administrador. • O produto a ser removido deve existir no banco de dados. • O produto não deve estar associado a nenhum pedido em andamento.
Entradas	• Clicar no botão para remover
Saídas	• O produto deve ser removido do catálogo do sistema
Pós-	• O registro do produto deve ocultado do cardápio

Condições	
Prioridade	Essencial
Dependência	RF005 (Adicionar Produtos ao Banco de Dados)

A.7 RF007 - Status do Pedido

Quadro 7 - A.7 RF007 - Status do Pedido

Descrição	O sistema deve permitir que o usuário acompanhe o status do seu pedido, desde a realização até a entrega.
Pré-condições	• O usuário deve estar logado no sistema. • O usuário deve ter realizado pelo menos um pedido.
Entradas	• Número do pedido
Saídas	• Exibição do status atual do pedido (por exemplo: em processamento, enviado, entregue) • Detalhes do pedido, como data do pedido, produtos, quantidade e valor total.
Pós-Condições	• O sistema deve buscar as informações do pedido no banco de dados e apresentar ao usuário de forma clara e concisa. • O status do pedido deve ser atualizado no sistema pela equipe da cantina
Prioridade	Essencial
Dependência	RF004 (Enviar Pedido) RF002 - Logar Usuário

A.8 RF008 - Cancelamento de pedidos

Quadro 8 - A.8 RF008 - Cancelamento de pedidos

Descrição	O sistema deverá permitir que o administrador cancele pedidos quando for necessário, a atualização ocorrerá assim que possível para a equipe.
Pré-condições	• O usuário deve ter realizado pelo menos um pedido.
Entradas	• Número do pedido
Saídas	• O pedido é cancelado. • Retorna a confirmação do cancelamento na interface do administrador.
Pós-Condições	• O banco de dados atualizará as informações do pedido.
Prioridade	Importante
Dependência	RF004 (Enviar Pedido)

A.9 RF009 - Esqueceu sua Senha

Quadro 9 - A.9 RF009 - Esqueceu sua Senha

Descrição	O sistema deverá permitir que o usuário troque a sua senha.
Pré-condições	• O usuário deve estar cadastrado no sistema.
Entradas	• Nova senha
Saídas	• A senha esquecida é trocada.
Pós-Condições	• O banco de dados atualizará as informações do usuário, e sua senha será alterada
Prioridade	Importante
Depend	RF001 (Cadastro)

ência	RF002 - Logar Usuário

A.10 RF010 - Exibir Carrinho

Quadro 10 - A.10 RF010 - Exibir Carrinho

Descrição	O sistema deve permitir que o usuário acesse e visualize o conteúdo do seu carrinho de compras.
Pré-condições	• O usuário deve estar logado no sistema. • O usuário deve ter adicionado pelo menos um item ao carrinho de compras.
Entradas	• Acessar o carrinho pelo produto ou pelo menu.

Saídas	• Visualização detalhada do carrinho de compras, incluindo:
	  ○ Lista de itens no carrinho.
	  ○ Quantidade de cada item.
	  ○ Preço unitário de cada item.
	  ○ Preço total dos itens no carrinho.
	  ○ Opções para alterar a quantidade de itens ou remover itens do carrinho.

Pós-Condições	• O usuário consegue ver uma visão clara e precisa dos itens no seu carrinho de compras.
Prioridade	Essencial
Dependência	RF002 (Logar usuário) RF003 - Adicionar ao Carrinho

A.11 RF011 - Excluir item do carrinho

Quadro 11 - A.11 RF011 - Excluir item do carrinho

Descrição	O sistema deve permitir que o usuário remova um item específico do seu carrinho de compras.
Pré-	• O usuário deve estar logado no sistema.

condições	• O usuário deve ter um carrinho de compras com pelo menos um item. • O item a ser excluído deve existir no carrinho de compras.
Entradas	• Botão de excluir disponível ao lado do produto.
Saídas	• Atualização da visualização do carrinho, refletindo a remoção do item. • Atualização do preço total do carrinho após a remoção do item.
Pós-Condições	• O item especificado é removido do carrinho de compras. • O carrinho de compras é atualizado para refletir a remoção do item e o preço total é recalculado.
Prioridade	Essencial
Dependência	RF002 Logar usuário RF003 - Adicionar ao Carrinho RF010 - Exibir Carrinho

A.12 RF012 - Exibir produtos

Quadro 12 - A.12 RF 012 - Exibir produtos

Descrição	O sistema deve permitir que o usuário visualize uma lista de produtos disponíveis, com informações básicas sobre cada produto.
Pré-condições	• O sistema deve ter um conjunto de produtos cadastrados e disponíveis para exibição.
Entradas	• Produtos devem estar adicionados no banco de dados.
Saídas	• Lista de produtos disponíveis, incluindo informações básicas

	sobre cada produto, tais como: o Nome do produto. o Imagem do produto. o Preço do produto.
Pós-Condições	• O usuário consegue visualizar uma lista de produtos com informações básicas. • O sistema permite que o usuário navegue pelos produtos e selecione um produto para obter detalhes adicionais, se desejar.
Prioridade	Essencial
Dependência	RF005 - Adicionar Produtos ao Banco de Dados

A.13 RF013 - Exibir produto único

Quadro 13 - A.13 RF013 - Exibir produto único

Descrição	O sistema deve permitir que o usuário visualize os detalhes completos de um produto específico ao selecionar um item da lista de produtos ou buscar diretamente pelo produto.
Pré-condições	• O produto deve estar cadastrado e disponível no sistema.
Entradas	• Clicar no produto para ter o acesso individual dele.
Saídas	• Detalhes completos do produto, incluindo: o Nome do produto. o Descrição completa do produto. o Imagens do produto

	○ Preço do produto. ○ Opções para adicionar o produto ao carrinho de compras ou realizar outras ações relacionadas.
Pós-Condições	• O usuário tem acesso a todas as informações detalhadas sobre o produto selecionado. • O sistema garante que as informações exibidas são precisas e atualizadas.
Prioridade	Essencial
Dependência	RF005 - Adicionar Produtos ao Banco de Dados

A.14 RF014 - Modificação de nome do perfil

Quadro 14 - A.14 RF014 - Modificação de nome do perfil

Descrição	Permitir que o usuário altere o nome associado ao seu perfil.
Pré-condições	• O usuário deve estar autenticado no sistema. • O nome atual deve estar corretamente carregado no perfil do usuário.
Entradas	• Acessar a página de perfil do usuário. • Novo nome desejado pelo usuário.
Saídas	• Atualização de todas as instâncias do sistema onde o nome do usuário é exibido.
Pós-Condições	• O nome do usuário é atualizado em todo o sistema. • Todas as funcionalidades dependentes do nome do usuário refletem essa mudança.
Prioridade	Importante
Dependência	RF002 - Logar usuário RF001 - Cadastrar Usuário RF015 - Confirmação de senha do perfil

A.15 RF015 - Confirmação de senha do perfil

Quadro 15 - A.15 RF015 – Confirmação de senha do perfil

Descrição	O sistema deve permitir que o usuário confirme sua senha antes de mudar suas informações pessoais
Pré-condições	• O usuário deve estar logado no sistema. • O usuário deve conhecer a senha atual.
Entradas	• Senha atual do usuário. • Confirmação da senha.
Saídas	• Altera o nome ou a foto do perfil.
Pós-Condições	• Informações do perfil são alteradas de acordo com o necessário.
Prioridade	Importante
Dependência	RF002 - Logar usuário RF001 - Cadastrar Usuário

A.16 RF016 - Modificação de foto do perfil

Quadro 16 - A.16 RF016 - Modificação de foto do perfil

Descrição	O sistema deve permitir que o usuário altere a foto associada ao seu perfil.
Pré-condições	• O usuário deve estar logado no sistema.
Entradas	• Nova foto de perfil (upload de arquivo).
Saídas	• Confirmação de que a foto foi alterada com sucesso. • Atualização de todas as instâncias do sistema onde a foto do perfil é exibida.
Pós-Condições	• A foto de perfil do usuário é atualizada em todo o sistema. • Todas as funcionalidades dependentes da foto do usuário

	refletem essa mudança.
Prioridade	Importante
Dependência	RF002 - Logar usuário RF001 - Cadastrar Usuário RF015 - Confirmação de senha do perfil

A.17 RF017 - Histórico de pedidos

Quadro 17 - A.17 RF017 - Histórico de pedidos

Descrição	O sistema deve permitir que o usuário visualize o histórico de todos os pedidos realizados, exibindo informações detalhadas de cada pedido, incluindo status, data e produtos adquiridos.
Pré-condições	• O usuário deve estar logado no sistema.
Entradas	• Acessar a página de histórico do pedido pelo menu.

Saídas	• Se houverem pedidos exibir uma lista com as seguintes informações:
	  ◦ ID do pedido
	  ◦ Data do pedido
	  ◦ Status do pedido (ex: entregue, pendente, cancelado)
	  ◦ Lista de produtos adquiridos (nome, quantidade, preço)
	  ◦ Total pago no pedido

Pós-Condições	• O usuário terá acesso ao histórico completo de seus pedidos. • O sistema permanece consistente, e as informações exibidas refletem com precisão o estado atual dos pedidos.
Prioridade	Essencial
Dependência	RF002 - Logar usuário

A.18 RF018 - Exibir pedidos

Quadro 18 - A.18 RF018 - Exibir pedidos

Descrição	O sistema deve permitir que o administrador visualize uma lista de todos os pedidos realizados por qualquer usuário no sistema, com informações resumidas de cada pedido, incluindo data, status, valor total e informações do cliente.
Pré-condições	• O administrador deve estar autenticado no sistema com permissões adequadas. • Devem existir pedidos realizados no sistema para exibição.
Entradas	• Página de pedidos feitos
Saídas	• Exibição de uma lista de pedidos com informações resumidas, incluindo: o ID do pedido o Nome do cliente o Data do pedido o Status do pedido o Valor total do pedido o Quantidade de itens
Pós-Condições	• O administrador pode visualizar todos os pedidos do sistema com uma visão clara e organizada. • O administrador pode alterar o status do pedido conforme ele é preparado.
Prioridade	Essencial
Dependência	RF002 - Logar Usuário RF004 - Enviar Pedido

A.19 RF 019 - Apagar carrinho após o envio do pedido

Quadro 19 - A.19 RF019 - Apagar carrinho após o envio do pedido

Descrição	Após o cliente finalizar a compra e o pedido ser enviado, todos os itens presentes no carrinho devem ser removidos do carrinho.
Pré-condições	• O cliente deve estar logado no sistema. • O carrinho deve conter produtos adicionados pelo cliente. • Um pedido deve ser enviado com sucesso para que os itens do carrinho possam ser removidos.
Entradas	• O pedido ser enviado com sucesso.
Saídas	• O carrinho do cliente será esvaziado. • Confirmação de que o pedido foi enviado com sucesso e o carrinho foi apagado.
Pós-Condições	• O carrinho do cliente estará vazio, pronto para novos itens serem adicionados. • O pedido será enviado para a equipe da cantina.
Prioridade	Essencial
Dependência	RF002 - Logar Usuário RF004 - Enviar Pedido RF003 - Adicionar ao Carrinho

A.20 RF020 – Logout

Quadro 20 - A.20 RF020 - Logout

Descrição	O sistema deve permitir que o usuário saia da sua sessão, encerrando seu acesso ao sistema e garantindo que as informações de sessão sejam descartadas de forma segura.
Pré-condições	O usuário deve estar logado no sistema.
Entradas	Clicar no botão “Sair” presente no menu.
Saídas	O usuário tem sua conta desconectada.
Pós-	O usuário sai da conta, e precisa entrar novamente.

Condições	
Prioridade	Essencial
Dependência	RF002 - Logar Usuário

A.21 RF021 - Apagar carrinho após o logout

Quadro 21 - A.21 RF021 - Apagar carrinho após o logout

Descrição	O sistema deve apagar o conteúdo do carrinho de compras do usuário quando ele realizar o logout, garantindo que os itens selecionados não sejam mantidos após o término da sessão.
Pré-condições	• O usuário deve estar logado no sistema. • O carrinho de compras deve conter produtos adicionados pelo usuário.
Entradas	Clicar no botão “Sair” presente no menu.
Saídas	• O carrinho de compras é apagado após o logout.
Pós-Condições	• O carrinho de compras do usuário é completamente limpo após o logout, removendo todos os itens selecionados. • Ao realizar um novo login, o carrinho do usuário estará vazio.
Prioridade	Importante
Dependência	RF002 - Logar Usuário RF003 - Adicionar ao Carrinho

A.22 RF022 - Editar produto

Quadro 22 - A.22 RF022 - Editar produto

Descrição	O sistema deve permitir a edição das informações de produtos já cadastrados no banco de dados, incluindo nome, preço,

	descrição e imagem.
Pré-condições	• O administrador deve estar autenticado no sistema. • O produto a ser editado deve estar previamente cadastrado no banco de dados.
Entradas	O administrador insere as novas informações do produto, como nome, preço, descrição e imagem.
Saídas	As informações do produto são atualizadas no banco de dados.
Pós-Condições	O produto atualizado estará disponível no catálogo com as novas informações.
Prioridade	Importante
Dependência	RF002 – Logar Usuário RF005 – Adicionar produtos ao banco de dados

A.23 RF023 - Habilitar/Desabilitar guarnição

Quadro 23 - A.23 RF023 - Habilitar/Desabilitar guarnição

Descrição	O sistema deve permitir que o administrador habilite ou desabilite a disponibilidade de guarnições (acompanhamentos) para as marmitas no cardápio. A funcionalidade deve ser acessível por meio de um painel administrativo, onde o administrador pode escolher quais guarnições estarão disponíveis para seleção.
Pré-condições	O administrador deve estar autenticado no sistema.
Entradas	• Identificador da guarnição que será habilitada ou desabilitada.

Saídas	• A guarnição será exibida na interface de seleção dos clientes ao realizar pedidos de marmitas. • Confirmação de que a alteração foi realizada com sucesso.
Pós-Condições	• Guarnições habilitadas estarão disponíveis para seleção no momento de um pedido. • Guarnições desabilitadas não poderão ser selecionadas até que sejam habilitadas novamente.
Prioridade	Essencial
Dependência	RF002 (Logar Usuário) RF005 (Adicionar Produtos ao Banco de Dados) RF012 (Exibir produtos)

A.24 RF024 - Exibir Marmita

Quadro 24 - A.24 RF024 - Exibir Marmita

Descrição	O sistema deve permitir a visualização das marmitas disponíveis no cardápio, mostrando os detalhes de cada marmita, como, acompanhamentos e preço. O cliente deve conseguir visualizar essas informações tanto na tela inicial quanto na página de detalhes do produto.
Pré-condições	Marmitas devem estar cadastradas e disponíveis no sistema.
Entradas	• Acessar a sessão de marmitas do cardápio.
Saídas	• A Exibição das informações completas da marmita: • Nome da marmita • Guarnições (arroz, feijão, batata, salada, etc.) • Preço • Imagem (se aplicável)
Pós-	• As informações sobre as marmitas serão exibidas na tela

Condições	para o cliente selecionar.
Prioridade	Essencial
Dependência	RF012 (Exibir produtos) RF005 (Adicionar Produtos ao Banco de Dados)

A.25 RF025 - Exibir Guarnição

Quadro 25 - A.25 RF025 - Exibir Guarnição

Descrição	O sistema deve exibir as guarnições disponíveis para serem adicionadas às marmitas no momento da seleção dos produtos.
Pré-condições	Guarnições devem estar cadastradas no banco de dados.
Entradas	• Exibir os detalhes das marmitas.
Saídas	• Uma lista de guarnições é apresentada ao usuário.
Pós-Condições	• O usuário consegue visualizar as guarnições disponíveis e escolher uma ou mais opções para adicionar à marmita.
Prioridade	Essencial
Dependência	RF023 (Habilitar/Desabilitar Guarnição) RF024 (Exibir marmita) RF005 (Adicionar Produtos ao Banco de Dados)

A.26 RF026 - Exibir Relatório

Quadro 26 - A.26 RF026 - Exibir Relatório

Descrição	O sistema deve permitir que o administrador visualize um relatório de vendas de um determinado período de tempo, com dados detalhados sobre as vendas realizadas.
Pré-condições	O sistema deve ter registros de vendas no banco de dados e o administrador deve estar autenticado no sistema.
Entradas	• O administrador escolhe o intervalo de datas desejado para gerar o relatório.
Saídas	• Um relatório de vendas com informações detalhadas sobre o total de vendas é exibido.
Pós-Condições	• O administrador visualiza o relatório com os dados de vendas do período solicitado e pode usar essas informações para análise.
Prioridade	Essencial
Dependência	RF002 (Logar usuário)

	RF005 (Adicionar Produtos ao Banco de Dados) RF018 (Exibir Pedidos)

APÊNDICE M - Requisitos não funcionais

A.1 RNF 001 - Criptografia de senha

Quadro 27 - A.1 RNF 001 - Criptografia de senha

Descrição	Todas as senhas dos usuários devem ser armazenadas de forma criptografada no banco de dados.
Critérios de Aceitação	• Utilizar um algoritmo de criptografia forte e amplamente utilizado, como SHA-256 ou bcrypt. • As senhas criptografadas devem ser armazenadas separadamente dos demais dados do usuário, em uma tabela específica do banco de dados.
Impacto e observações	• **Desempenho:** A criptografia pode adicionar um pequeno overhead de processamento durante o cadastro e login de usuários. • **Complexidade:** Requer conhecimento técnico para implementar corretamente a criptografia.
Prioridade	importante
Dependência	

A.2 RNF 002 - Facilidade de uso

Quadro 28 - A.2 RNF 002 ¬- Facilidade de uso

Descrição	O sistema deve ser projetado e desenvolvido de forma a ser intuitivo e fácil de usar por todos os usuários,

	independentemente do seu nível de conhecimento técnico.
Critérios de Aceitação	• **Interface intuitiva:** A interface gráfica do usuário (GUI) deve ser clara, organizada e seguir padrões de design conhecidos (como material design ou human interface guidelines). • **Feedback visual**: O sistema deve fornecer feedback visual claro para as ações do usuário, como mensagens de confirmação, alertas e indicadores de progresso. • **Atalhos de teclado**: Implementar atalhos de teclado para as funções mais utilizadas, agilizando o trabalho dos usuários. • **Personalização**: Permitir que os usuários personalizem a interface de acordo com suas preferências. • **Treinamento**: Deve ser possível realizar o treinamento do sistema em menos de 48h.
Impacto e observações	• **Complexidade:** Requer um técnico especializado em frontend.
Prioridade	Importante
Dependência	

A.3 RNF 003 - Integridade de dados de cadastro

Quadro 29 - A.3 RNF 003 - Integridade de dados de cadastro

Descrição	O sistema deve garantir a integridade dos dados de cadastro, assegurando que cada e-mail utilizado para criar uma conta seja único no banco de dados.
Critérios de Aceitação	• Utilizar um algoritmo de criptografia forte e amplamente utilizado, como SHA-256 ou bcrypt. •
Impacto e	• **Desempenho:** A criptografia pode adicionar um pequeno

observações	overhead de processamento durante o cadastro e login de usuários. • **Complexidade:** Requer conhecimento técnico para implementar corretamente a criptografia.
Prioridade	essêncial
Dependência	

A.4 RNF 004 - Compatibilidade

Quadro 30 - A.4 RNF 004 - Compatibilidade

Descrição	O sistema deve ser projetado de forma a facilitar a manutenção e suporte técnico, com documentação clara e modularidade no código.
Critérios de Aceitação	• Documentação técnica detalhada disponível para desenvolvedores e equipe de suporte. • O Código deve ser modular e seguir boas práticas de programação para facilitar futuras atualizações e correções. • Deve haver um plano de manutenção regular para atualizar o sistema e corrigir bugs.
Impacto e observações	• **Desempenho:** A modularidade do código e a documentação clara podem reduzir o tempo de resposta para manutenção, mas podem adicionar um overhead inicial durante o desenvolvimento para garantir que essas práticas sejam seguidas. • **Complexidade:** A necessidade de manter a documentação atualizada e garantir a modularidade do código requer um nível elevado de disciplina e conhecimento técnico, o que pode aumentar a curva de aprendizado para novos membros da equipe.
Prioridade	Importante
Depend	

ência	

A.5 RNF 005 - Suporte e manutenção

Quadro 31 - A.5 RNF 005 - Suporte e manutenção

Descrição	O sistema deve ser projetado de forma a facilitar a manutenção e suporte técnico, com documentação clara e modularidade no código.
Critérios de Aceitação	• Documentação técnica detalhada disponível para desenvolvedores e equipe de suporte. • O Código deve ser modular e seguir boas práticas de programação para facilitar futuras atualizações e correções. • Deve haver um plano de manutenção regular para atualizar o sistema e corrigir bugs.
Impacto e observações	• **Desempenho:** A modularidade do código e a documentação clara podem reduzir o tempo de resposta para manutenção, mas podem adicionar um overhead inicial durante o desenvolvimento para garantir que essas práticas sejam seguidas. • **Complexidade:** A necessidade de manter a documentação atualizada e garantir a modularidade do código requer um nível elevado de disciplina e conhecimento técnico, o que pode aumentar a curva de aprendizado para novos membros da equipe.
Prioridade	Importante
Dependência	

A.6 RNF 006 - Segurança na manutenção de perfil

Quadro 32 - A.6 RNF 006 - Segurança na manutenção de perfil

Descrição	O sistema deve garantir que a modificação de nome,

	senha e foto de perfil seja realizada de forma segura, protegendo os dados do usuário.
Critérios de Aceitação	• Implementação de autenticação dupla (como senha e código de verificação) para modificar o perfil. • Logs de auditoria gerados para cada modificação de perfil. • Verificação de senha atual antes de permitir a alteração da senha. • Notificação ao usuário via e-mail ou SMS após qualquer modificação de perfil.
Impacto e observações	• **Desempenho:** A adição de medidas de segurança pode aumentar o tempo de resposta durante a modificação do perfil. • **Complexidade:** Requer conhecimento avançado de segurança para implementar medidas como autenticação dupla e monitoramento de logs.
Prioridade	Essencial
Dependência	RF001 (Autenticação de Usuário) RF002 (Logar Usuário) RF003 (Modificação de Nome, Senha, e Foto do Perfil)

APÊNDICE N - Regras de negócio

A.1 RG 001 - Sistema aberto até 21:45

Quadro 33 - A.1 RG 001 - Sistema aberto até 21:45

Descrição	Compras só poderão ser efetuadas antes das 21:45
Observações	A regra visa garantir que os funcionários não trabalhem fora de seu tempo contratual.
Ações	• Se o pedido for feito após o horário final, aparecerá uma frase

	em vermelho:A Loja está fechada.

A.2 RG 002 - Pedido de marmitas até 09:45

Quadro 34 - A.2 RG 002 - Pedido de marmitas até 09:45

Descrição	Os pedidos de marmitas só poderão ser efetuados antes das 09:45.
Observações	A regra visa garantir que haja tempo na preparação dos pedidos e menos desperdício.
Ações	• Se o pedido for feito após o horário final, aparecerá uma frase em vermelho: Não foi possível realizar seu pedido.

A.3 RG 003 - Limite de adicionais em lanches

Quadro 35 - A.3 RG 003 - Limite de adicionais em lanches

Descrição	Cada lanche pode ter, no máximo, 3 adicionais (como cheddar, batata frita, etc.).
Observações	Essa regra visa garantir que o processo de preparação dos lanches seja eficiente e dentro dos padrões de qualidade estabelecidos pela cantina.
Ações	• Se o cliente tentar adicionar mais de 3 itens extras ao lanche, aparecerá uma mensagem em vermelho: "Limite de 3 adicionais por lanche excedido."

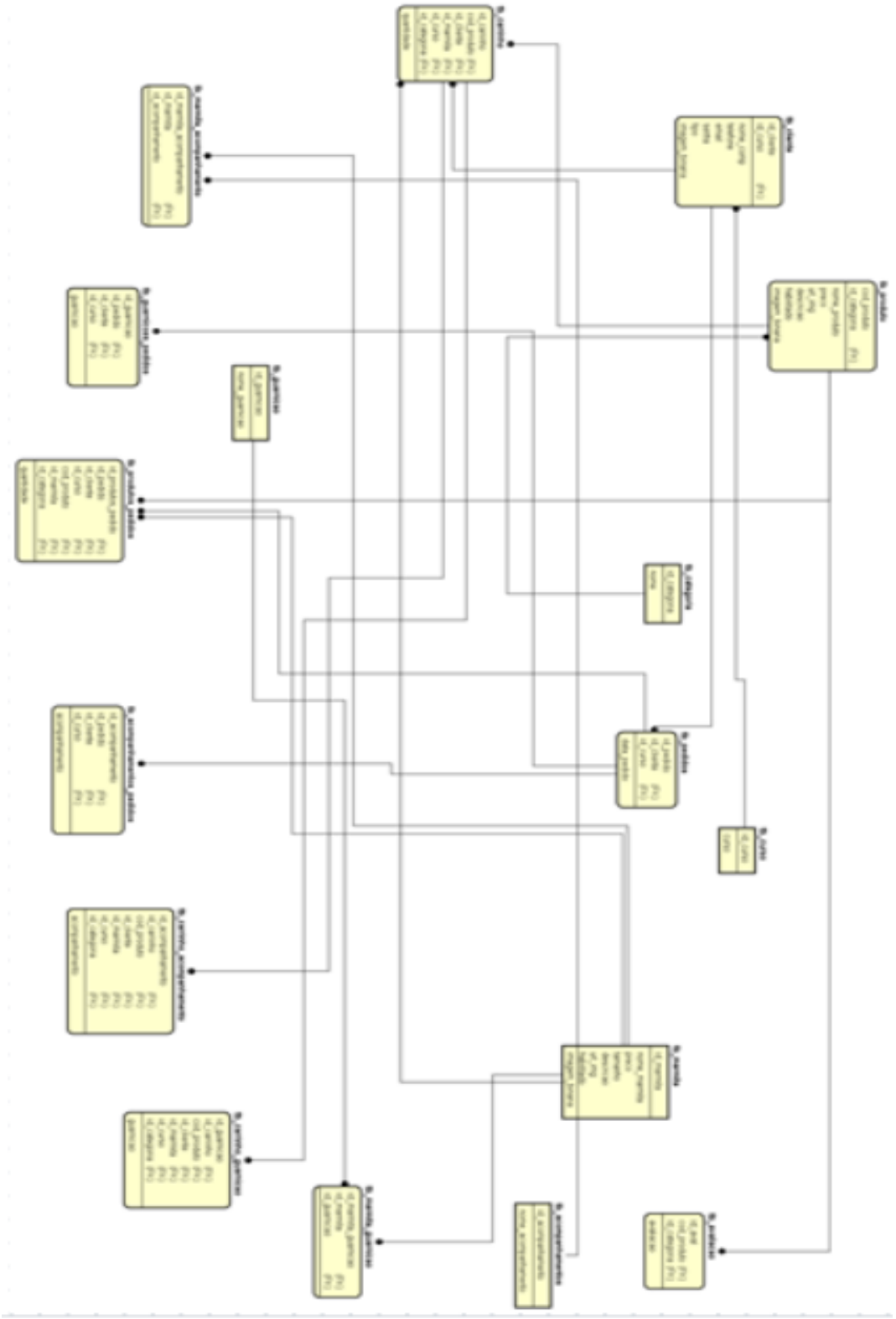
A.4 RG 004 - Gerenciamento de estoque em tempo real

Quadro 36 - A.4 RG 004 - Gerenciamento de estoque em tempo real

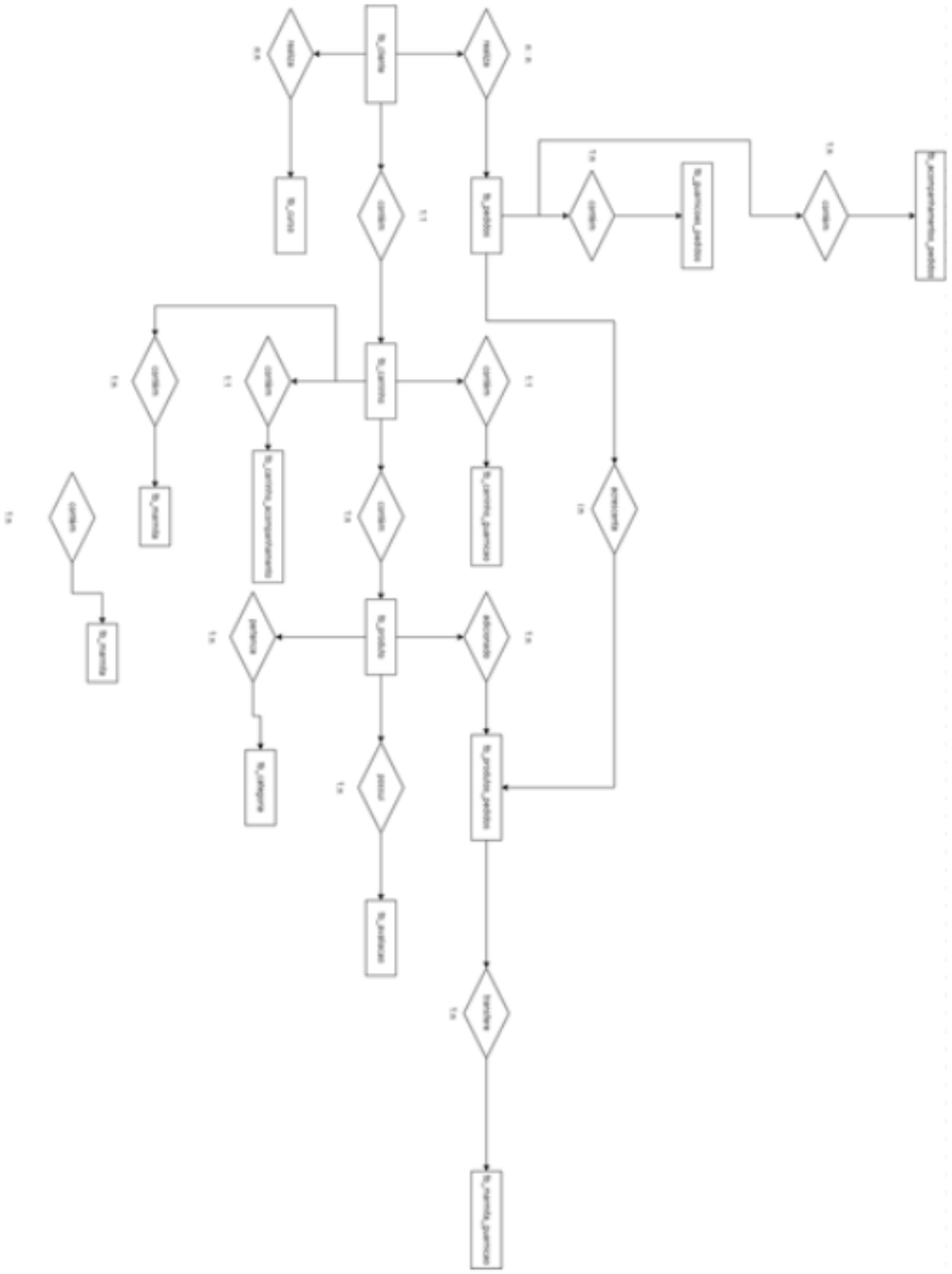
Descrição	O sistema deve garantir que o estoque seja atualizado em tempo real à medida que os pedidos são realizados.
Observações	Deve ser implementada uma política para lidar com situações em que o estoque não esteja sincronizado (por exemplo, vendas simultâneas do último item em estoque).

Ações	Desabilitar a seleção dos itens que estejam indisponíveis.

APÊNDICE O – Diagrama DER



APÊNDICE P – Diagrama MER



ANEXO A - Título do anexo