CENTRO PAULA SOUZA
FACULDADE DE TECNOLOGIA DE FRANCA
“Dr. THOMAZ NOVELINO”

TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

BRUNO CASON DA SILVA

SUSHITECH:

sistema web para gerenciamento de pedidos

Trabalho de Graduação apresentado à Faculdade de Tecnologia de Franca - “Dr. Thomaz Novelino”, como parte dos requisitos obrigatórios para obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Orientador: Prof. Me. Carlos Alberto Lucas

FRANCA/SP
2024

SUSHITECH

Sistema web para gerenciamento de pedidos

Bruno Cason da Silva[1]

Resumo

A decisão de criar o projeto foi tomada após uma conversa informal com um amigo e, em seguida, após a análise dos desafios enfrentados por alguns restaurantes japoneses no processo de gestão de pedidos e atendimento ao cliente. Com essa análise, identificou-se que a lentidão e os erros frequentes no registro manual dos pedidos eram algumas das principais dificuldades enfrentadas nesse segmento. Em resposta a essa problemática, foi apresentado o SushiTech, um sistema *web* para gerenciamento de pedidos desenvolvido especificamente para atender às demandas desses restaurantes, oferecendo uma solução eficiente e simplificada para a gestão de mesas e pedidos. O sistema possibilitou que os estabelecimentos gerenciassem seus produtos, mesas e pedidos de forma centralizada, enquanto os clientes tiveram a conveniência de acessar o cardápio digitalmente, personalizar seus pedidos e acompanhar o *status* em tempo real. Essa abordagem inovadora buscou resolver os desafios enfrentados pelos restaurantes japoneses, promovendo uma modernização significativa em suas operações, agilizando o atendimento e melhorando a experiência dos clientes. Assim, a solução procurou proporcionar uma experiência satisfatória tanto para os restaurantes quanto para os consumidores.

Palavras-chave: Cardápio Digital. Gestão de Pedidos. Restaurantes Japoneses. Sistema Web.

Abstract

The decision to create the project was made after an informal conversation with a friend and subsequently after analyzing the challenges faced by some Japanese restaurants in managing orders and customer service. This analysis identified that delays and frequent errors in manual order recording were among the main difficulties faced in this sector. In response to this issue, SushiTech was introduced—a web-based order management system specifically developed to meet the needs of these restaurants, providing an efficient and simplified solution for table and order management. The system enabled establishments to manage their products, tables, and orders in a centralized manner, while customers enjoyed the convenience of accessing the menu digitally, customizing their orders, and tracking their status in real-time. This innovative approach aimed to address the challenges faced by Japanese restaurants, promoting significant modernization in their operations, streamlining service, and enhancing the customer experience. Thus, the solution sought to provide a satisfying experience for both restaurants and consumers.

Keywords: *Digital Menu. Japanese Restaurants. Order Management. Web System.*

---

[1] Graduando em Análise e Desenvolvimento de Sistemas pela Fatec Dr. Thomaz Novelino – Franca/SP. Endereço eletrônico: brunocasons@gmail.com

1 Introdução

Atualmente, o uso de *softwares* é fundamental para otimizar processos e melhorar a eficiência em diversos setores. Esse fenômeno é particularmente evidente no ramo de serviços, onde a agilidade e a precisão são essenciais para a satisfação do cliente e a competitividade no mercado.

No contexto da gastronomia, especificamente em restaurantes japoneses, observa-se a necessidade de modernizar e aprimorar o processo de gestão de pedidos e atendimento ao cliente. A lentidão e os erros frequentes no registro manual dos pedidos se configuram como desafios significativos enfrentados pelos estabelecimentos.

Diante dessa problemática, a questão central que norteia este projeto é: como um sistema de comanda online pode transformar e otimizar o atendimento em restaurantes japoneses, melhorando a experiência do cliente e a eficiência operacional?

Para responder a essa questão, formulamos a hipótese de que a implementação de um sistema *web* para gerenciamento de pedidos, permitirá uma gestão mais eficiente, possibilitando um aumento na agilidade do atendimento e redução de erros.

Os objetivos deste trabalho incluem desenvolver uma aplicação que ofereça um cardápio digital e possibilite o acompanhamento em tempo real do *status* dos pedidos.

A justificativa para a realização deste projeto reside na relevância de proporcionar aos restaurantes, ferramentas que não apenas modernizem suas operações, mas também elevem a qualidade do serviço prestado ao cliente, contribuindo para a fidelização e satisfação do público.

A metodologia adotada para o desenvolvimento do sistema adota uma abordagem ágil, utilizando tecnologias reconhecidas como TypeScript e Firebase para o gerenciamento de dados. Essa escolha visa garantir uma plataforma funcional, de fácil uso e adaptável às necessidades dos usuários.

1.1 Termo de Abertura do Projeto (TAP)

O início de um projeto é oficialmente reconhecido pelo Termo de Abertura do Projeto (TAP), um documento importante que descreve vários componentes principais do projeto. Isto abrange a finalidade, descrição, objetivo, entrega, premissas e restrições do projeto.

A importância do TAP reside na sua capacidade de estabelecer bases sólidas

para o projeto, garantindo que todas as partes envolvidas compartilhem uma compreensão uniforme dos objetivos e expectativas com as outras (PROJECT BUILDER, 2021).

O TAP estabelece uma estrutura concisa tanto para o planejamento como para a implementação, reduzindo assim os riscos que poderiam potencialmente ser enfrentados durante um projeto, ao mesmo tempo que promete os resultados desejados.

1.1.1 Finalidade do Projeto

A finalidade principal do projeto foi desenvolver uma solução tecnológica que otimize o processo de pedidos, permitindo aos clientes realizarem suas solicitações diretamente por dispositivo móvel. O objetivo visa modernizar e agilizar a experiência do usuário, diminuindo o tempo de espera e o uso de papel, além de integrar o gerenciamento de mesas, pedidos e pagamentos de maneira eficiente. Isso também possibilita uma melhor gestão para o restaurante, facilitando o controle de fluxo e a análise de dados operacionais.

1.1.2 Descrição do Produto

O produto consiste em uma plataforma *web* de gerenciamento de pedidos voltada para restaurantes japoneses, oferecendo uma interface intuitiva para que os clientes possam realizar seus pedidos diretamente dos dispositivos móveis fornecidos pelo restaurante. A plataforma possibilita a visualização atualizada do cardápio, a seleção dos itens desejados, o acompanhamento do progresso dos pedidos e a conclusão do pagamento, tudo de forma integrada.

Para o restaurante, o sistema oferece ferramentas de gestão, como o controle de mesas, a visualização de pedidos e produção, atualização de cardápio. O design garante um atendimento personalizado e eficiente, beneficiando tanto o cliente quanto o estabelecimento. Através de um fluxo otimizado, a experiência se torna mais agradável e a operação, mais rápida.

1.1.3 Objetivo do Projeto

O objetivo do projeto consiste em criar uma solução *web* que melhore a eficiência operacional de restaurantes japoneses, automatizando o processo de pedidos e pagamento através de uma plataforma de comanda online. A intenção é proporcionar uma experiência aprimorada para os clientes, oferecendo um sistema

ágil e intuitivo, ao mesmo tempo em que o restaurante ganha maior controle e visibilidade sobre seus processos, resultando em uma operação mais organizada e produtiva.

1.1.4 Entrega do Projeto

A entrega do projeto consiste no desenvolvimento e entrega, navegável e totalmente funcional, pronto para o uso. Isso inclui o desenvolvimento completo do sistema de comanda online, com todas as funcionalidades integradas.

1.1.5 Premissas

As premissas críticas para o sucesso deste projeto estão baseadas nas necessidades e comportamentos dos consumidores e dos estabelecimentos no Brasil. Primeiramente, a crescente demanda por experiências digitais no setor de alimentação, especialmente após a pandemia, destaca a urgência de soluções tecnológicas que agilizem o atendimento e melhorem a experiência do cliente. Isso ressalta a relevância da aplicação *web*, que pode oferecer uma abordagem prática e acessível para que os clientes realizem pedidos de maneira rápida e eficiente.

Além disso, a ampla adoção de dispositivos móveis e a familiaridade dos usuários com a tecnologia criam um ambiente favorável para a implementação de uma plataforma online de comanda. A penetração dos *smartphones* na sociedade brasileira sugere que a maioria dos clientes estará confortável em utilizar uma aplicação *web* para interagir com o restaurante.

Essas premissas fundamentais não apenas indicam uma demanda potencialmente alta pela comanda online, mas também estabelecem uma base sólida para sua implementação bem-sucedida e aceitação no mercado, garantindo que tanto clientes quanto restaurantes possam se beneficiar dessa inovação.

1.1.6 Restrição

Assegurar a compatibilidade da aplicação *web* com diferentes navegadores e dispositivos. A necessidade de integração entre diversos sistemas operacionais, pode representar um desafio significativo, pois cada plataforma pode ter suas particularidades que afetam o desempenho e a usabilidade da aplicação.

A figura 1 apresenta o modelo SMART aplicado neste projeto.

Figura 1 – Modelo SMART

Fonte: Elaborado pelo autor.

O modelo SMART é uma ferramenta amplamente utilizada para definir metas e objetivos de forma clara e eficaz, garantindo que eles sejam bem estruturados e realizáveis.

> *Ele se baseia em cinco critérios principais: Específico (Specific) significa que o objetivo deve ser claro e direto, Mensurável (Measurable) se refere à capacidade de acompanhar e verificar o progresso do objetivo, Atingível (Attainable) garante que o objetivo possa ser realisticamente alcançado dentro das limitações de tempo, orçamento e recursos disponíveis, Relevante (Relevant) se concentra na importância do objetivo em relação ao projeto ou à organização como um todo e Temporal (Timely) está relacionado à definição de um prazo para o cumprimento do objetivo (Miro, 2024).*

2 Viabilidade do Projeto

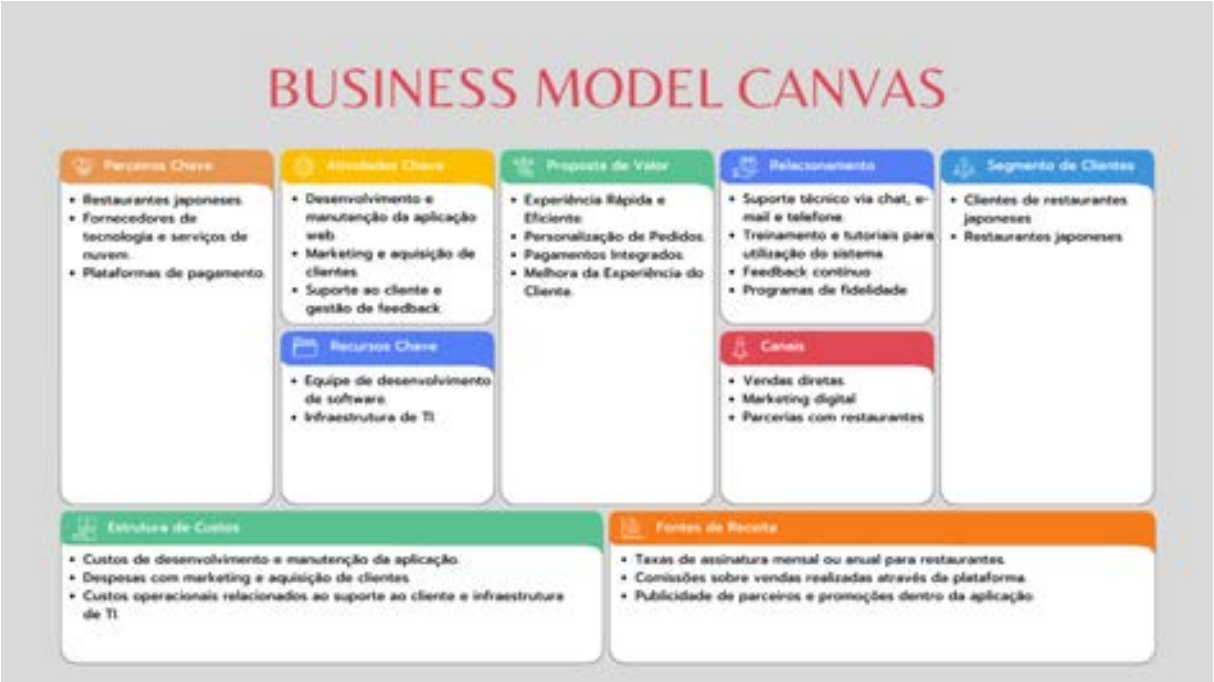
O *Business Model Canvas* (BMC) é uma ferramenta de gestão estratégica que permite desenvolver e esboçar modelos de negócio novos ou existentes em uma única página (O Analista de Modelos de Negócios, 2021). Ele é um mapa visual que contém nove blocos do modelo de negócio.

A importância *do Business Model Canvas* é inegável, pois, como afirma Atitude e Negócios (2021), ele permite que a empresa tenha um maior controle sobre sua cadeia de suprimentos, identificando possíveis problemas e tomando medidas corretivas de forma mais rápida e eficiente.

Na Figura 2 podemos identificar o modelo de BMC, pois além de cumprir uma

missão visual ele é importante para alinhar estratégias de inovação às necessidades do mercado e objetivos de negócio (Playstudio, 2021).

Figura 2 – BMC (Business Model Canvas)



Fonte: Elaborado pelo autor.

3 Levantamento de Requisitos

3.1 Elicitação e especificação dos Requisitos

A elicitação de requisitos é uma das etapas mais críticas em qualquer empreendimento de desenvolvimento de *software* (DevMedia, 2021). Um projeto será perdido se for mal realizado, é nele que estabelecemos como identificar o que o cliente precisa.

A necessidade de elicitação de requisitos não pode ser subestimada, definir o que o sistema deve fazer corretamente constitui a base para o sucesso do projeto (Curto Conselho, 2021). Com isso, se economiza tempo e dinheiro que seriam necessários para o esforço de desenvolvimento.

Neste momento, não é atribuído a um analista um problema para resolver, mas sim a tarefa de descobrir o que um usuário deseja ou precisa.

3.2 BPMN

BPMN significa Notação de Modelagem de Processos de Negócios.

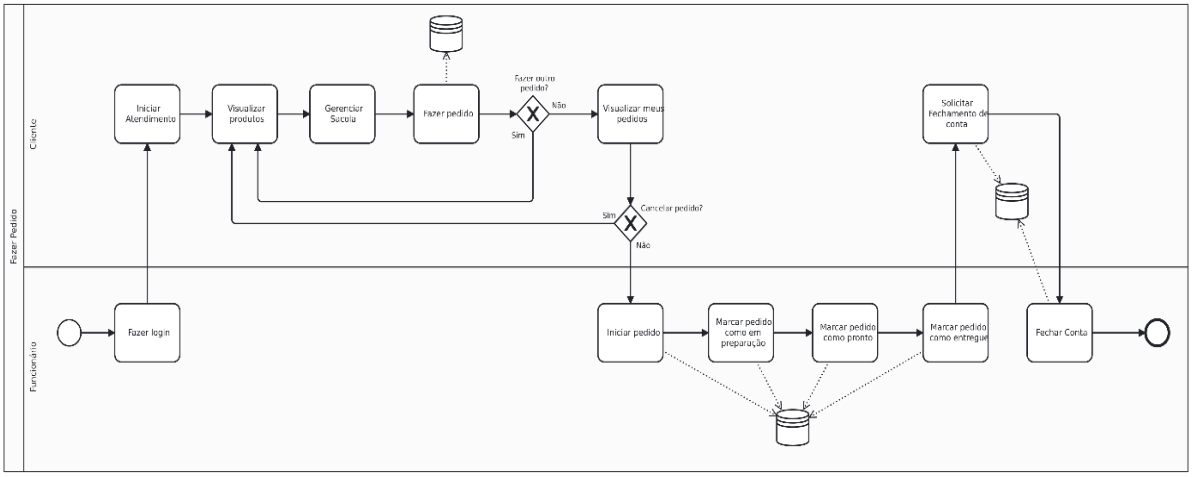
É uma linguagem visual criada com o objetivo de representar os processos de negócios de forma facilmente compreensível, porém rica em detalhes, utilizando notações padronizadas, como mostrado nas figuras abaixo.

O público principal do BPMN inclui analistas de negócios (que criam e refinam processos), desenvolvedores técnicos (responsáveis pela implementação) e gerentes de negócios (que monitoram esses processos).

Isto garante uma comunicação eficaz entre as partes interessadas durante o ciclo de vida do processo (Chinosi & Trombetta, 2012).

A figura 3, representa a ação de fazer um pedido.

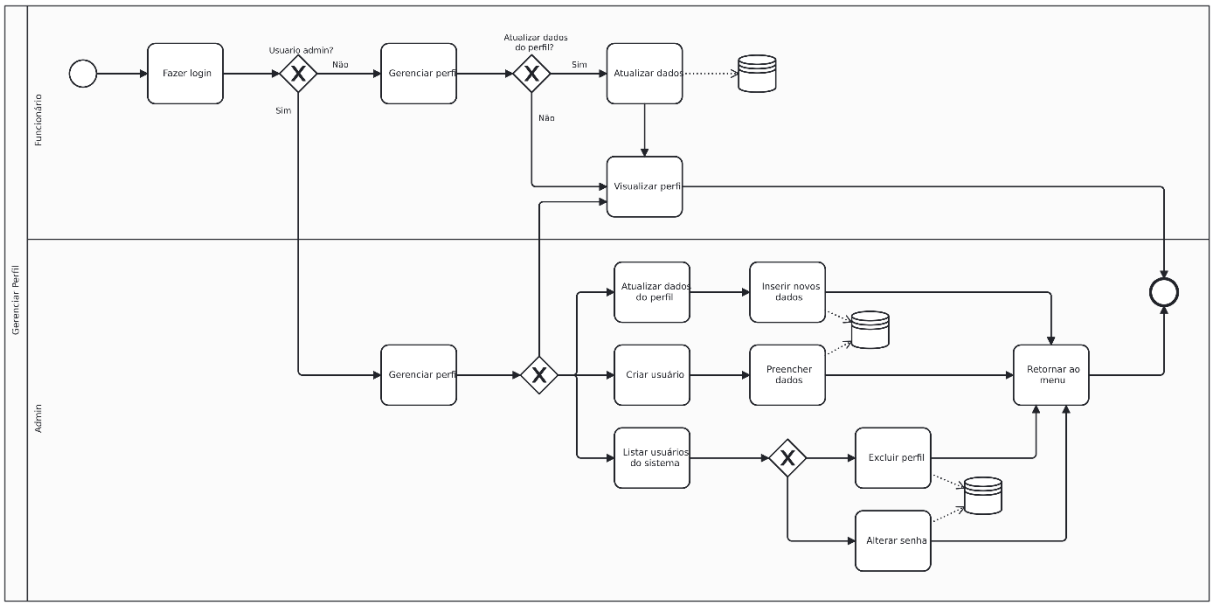
Figura 3 – BPMN - Fazer um pedido



Fonte: Elaborado pelo autor.

A figura 4 representa a ação de gerenciar o perfil do usuário.

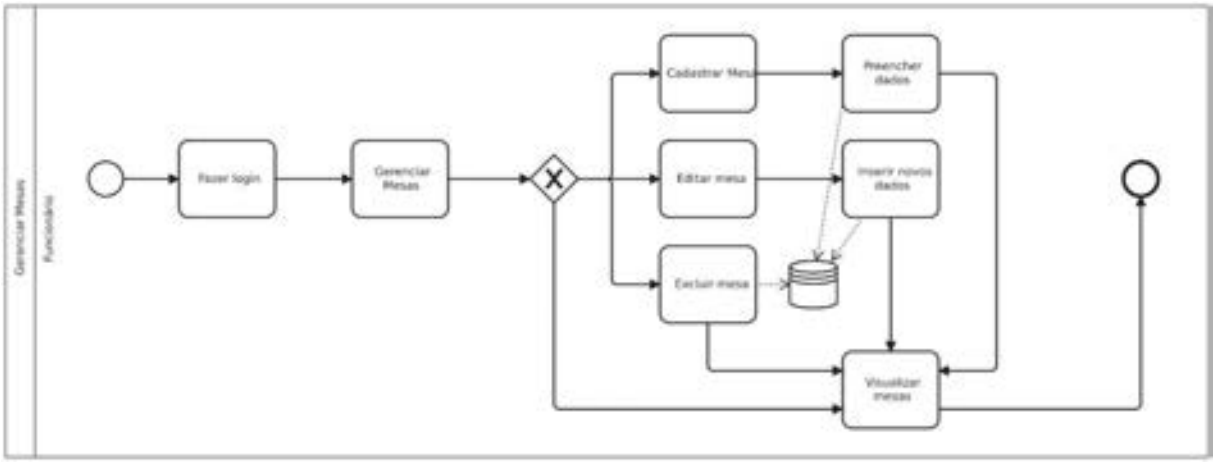
Figura 4 – BPMN Gerenciar perfil



Fonte: Elaborado pelo autor.

A figura 5 representa a ação de gerenciar as mesas.

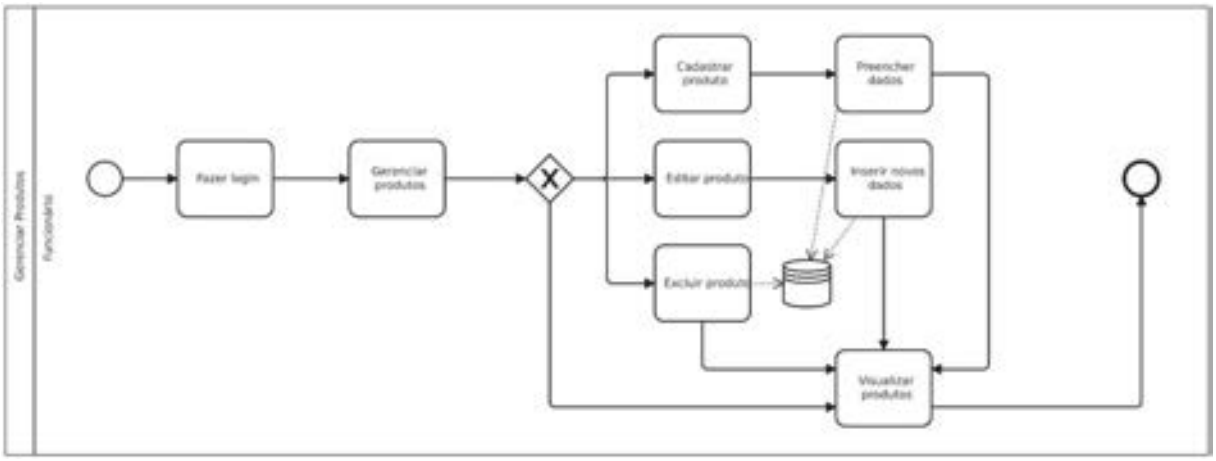
Figura 5 – BPMN - Gerenciar as mesas



Fonte: Elaborado pelo autor.

A figura 6 representa a ação de adicionar produtos.

Figura 6 – BPMN - Adicionar produtos



Fonte: Elaborado pelo autor.

3.3 Requisitos Funcionais

Os requisitos funcionais detalham as funções precisas que um sistema ou *software* deve ter capacidade de completar, caracterizando o que os usuários esperam e desejam ter em seu sistema, eles são compostos por uma parte evidente e uma parte oculta.

Eles dizem o que o sistema deve fazer e são necessários para produzir um sistema bem-sucedido com um bom nível de eficiência, pois nos dão uma visão clara do que o sistema precisa para funcionar (Leffingwell, 2011). Além disso, contribuem para confirmar que o sistema desenvolvido atende tanto às expectativas dos usuários quanto às necessidades do negócio, consulte o Quadro 1 para observar o seu uso nesse projeto.

Quadro 1 – Requisitos Funcionais do sistema

ID: RF001	**Nome:** Realizar Login
Categoria: Evidente	**Prioridade:** Essencial
Descrição	O sistema deverá permitir que os usuários façam login utilizando suas credenciais, que incluem e-mail e senha.
Informações	Campos necessários no formulário de login: e-mail e senha.
Regra do Negócio	Não há regra.
ID: RF002	**Nome:** Gerenciar Perfil
Categoria: Evidente	**Prioridade:** Essencial
Descrição	O sistema deverá gerenciar (CRUD) os dados do perfil.

Informações	Campos necessários para o gerenciamento de perfil: nome, e-mail, telefone, senha e papel do usuário.
Regra do Negócio	O sistema não poderá permitir a inclusão de usuários com o mesmo e-mail.
ID: RF003	**Nome:** Gerenciar Mesas
Categoria: Evidente	**Prioridade:** Essencial
Descrição	O sistema deverá gerenciar (CRUD) os dados das mesas.
Informações	Campos necessários gerenciamento de mesas: número da mesa.
Regra do Negócio	O sistema não poderá permitir a inclusão de mesas com o mesmo número.
ID: RF004	**Nome:** Gerenciar Produtos
Categoria: Evidente	**Prioridade:** Essencial
Descrição	O sistema deverá gerenciar (CRUD) os dados dos produtos.
Informações	Campos necessários no gerenciamento de produtos: imagem, nome, descrição, preço e categoria.
Regra do Negócio	Não há regra.
ID: RF005	**Nome:** Iniciar Atendimento
Categoria: Evidente	**Prioridade:** Essencial
Descrição	O sistema deverá permitir que o cliente inicie o atendimento, marcando o início do processo da realização de pedidos.
Informações	Campos necessários para iniciar o atendimento: número da mesa.
Regra do Negócio	Não há regra.
ID: RF006	**Nome:** Gerenciar Sacola
Categoria: Evidente	**Prioridade:** Essencial
Descrição	O sistema deverá permitir que o cliente gerencie a sacola (CRUD).
Informações	Campos necessários para gerenciar a sacola: ID do produto, nome do produto, quantidade, preço e total por item.
Regra do Negócio	Não há regra.
ID: RF007	**Nome:** Fazer Pedido
Categoria: Evidente	**Prioridade:** Essencial
Descrição	O sistema deverá permitir que o cliente finalize e envie o pedido após revisar o conteúdo da sacola.
Informações	Campos necessários para o processamento do pedido: número do pedido, número da mesa, detalhes da sacola (produtos, quantidades, preços), e status do pedido.
Regra do Negócio	Os clientes só podem fazer o pedido se produtos forem adicionados à sacola.
ID: RF008	**Nome:** Cancelar Pedido
Categoria: Evidente	**Prioridade:** Essencial
Descrição	O sistema deverá permitir que o usuário (cliente ou funcionário) cancele um pedido dentro de um período específico após a confirmação.

Informações	Campos necessários para o cancelamento: número do pedido, número da mesa e status do pedido.
Regra do Negócio	Os clientes só podem cancelar um pedido até o momento em que o funcionário inicie a sua preparação.
ID: RF009	**Nome:** Iniciar Pedido
Categoria: Evidente	**Prioridade:** Essencial
Descrição	O sistema deverá permitir que o funcionário inicie a preparação de um pedido assim que ele for recebido do cliente.
Informações	Campos necessários para iniciar o pedido: número do pedido, número da mesa, quantidade e status.
Regra do Negócio	Não há regra.
ID: RF010	**Nome:** Concluir Pedido
Categoria: Evidente	**Prioridade:** Essencial
Descrição	O sistema deverá permitir que o funcionário marque um pedido como concluído quando a preparação do pedido estiver pronta.
Informações	Campos necessários para concluir o pedido: número do pedido, número da mesa, quantidade e status.
Regra do Negócio	Não há regra.
ID: RF011	**Nome:** Entregar Pedido
Categoria: Evidente	**Prioridade:** Essencial
Descrição	O sistema deverá permitir que o funcionário marque um pedido como entregue assim que o cliente o receber.
Informações	Campos necessários para entregar o pedido: número do pedido, número da mesa, quantidade e status.
Regra do Negócio	Não há regra.
ID: RF012	**Nome:** Solicitar Fechamento de Conta
Categoria: Evidente	**Prioridade:** Essencial
Descrição	O sistema deverá permitir que o cliente solicite o fechamento de conta ao concluir seus pedidos.
Informações	Campos necessários para solicitar o fechamento da conta: número da mesa e status.
Regra do Negócio	O sistema não poderá permitir o fechamento da conta se houver pedidos ativos do cliente
ID: RF013	**Nome:** Fechar Conta
Categoria: Evidente	**Prioridade:** Essencial
Descrição	O sistema deverá permitir que o cliente finalize a conta ao concluir seus pedidos.
Informações	Campos necessários para fechar a conta: número da mesa, total a pagar.
Regra do Negócio	O sistema não poderá permitir o fechamento da conta se houver pedidos ativos do cliente

Fonte: Elaborado pelo autor.

3.4 Requisitos Não Funcionais

No quadro 2 podemos ver os requisitos não funcionais (RNFs), que se referem-se às restrições ou requisitos do sistema que delineiam seus atributos de qualidade (Visure Solutions, 2021). Eles abordam questões como escalabilidade, capacidade de manutenção, desempenho, portabilidade, segurança, confiabilidade e outras.

Estas demandas não interferem diretamente na construção do sistema em si; não são requisitos ou regras de negócios que determinariam o que precisa ser feito no desenvolvimento de *software* (Blog Trybe, 2021). Em vez disso, RNFs são requisitos que especificam o comportamento do sistema sob condições específicas.

Quadro 2 – Requisitos Não Funcionais do sistema

RNF001- Autenticação	O acesso ao sistema deve ser restrito a usuários autenticados, com diferentes níveis de acesso para clientes e administradores.	Tipo: Segurança	( ) Desejável (X) Obrigatório	(X) Permanente ( ) Transitório
RNF002- Usabilidade	A interface deve ser simples e intuitiva, facilitando o uso por clientes e administradores sem a necessidade de treinamento extenso.	Tipo: Interface	( ) Desejável (X) Obrigatório	(X) Permanente ( ) Transitório
RNF003- Compatibilidade	O sistema deve ser compatível com diferentes dispositivos e sistemas operacionais.	Tipo: Performance	( ) Desejável (X) Obrigatório	(X) Permanente ( ) Transitório
RNF004- Desempenho	O sistema deve responder às solicitações dos usuários de forma rápida e eficiente.	Tipo: Eficiência	( ) Desejável (X) Obrigatório	(X) Permanente ( ) Transitório
RNF005- Confiabilidade	O sistema deve ter uma alta disponibilidade, garantindo um tempo de inatividade mínimo.	Tipo: Eficiência	( ) Desejável (X) Obrigatório	(X) Permanente ( ) Transitório

Fonte: Elaborado pelo autor.

3.5 Regras de Negócio

As Regras do Negócio são instruções precisas que definem, controlam ou influenciam o comportamento operacional de um sistema (Alura, 2021). Elas refletem as políticas, procedimentos e condições essenciais para que a organização atinja seus objetivos, garantindo consistência, eficiência e conformidade nas operações.

Sua importância reside nos fatos de serem responsáveis por padronizar inúmeros processos internos e externos de uma empresa, alinhando as atividades com os seus valores organizacionais e otimizando resultados a partir da economia de recursos e da correção de erros comuns em projetos de produtos e serviços (Rocketseat,

2021). Conforme mostrado nas Regras de Negócio do sistema presentes no Quadro 3.

Quadro 3 – Regras de Negócio do sistema.

RN001 - Limitação de acesso
Descrição: Os clientes só podem acessar o sistema por meio de dispositivos fornecido pelo restaurante.
RN002 - Validação de usuário
Descrição: O sistema não poderá permitir a inclusão de usuários com o mesmo e-mail.
RN003 - Validação de mesa
Descrição: O sistema não poderá permitir a inclusão de mesas com o mesmo número.
RN004 – Validação de sacola para pedido
Descrição: Os clientes só podem fazer o pedido se produtos forem adicionados à sacola.
RN005 - Cancelamento de pedido
Descrição: Os clientes só podem cancelar um pedido até o momento em que o funcionário inicie a sua preparação.
RN006 – Fechamento de conta
Descrição: O sistema não poderá permitir o fechamento da conta se houver pedidos ativos do cliente.

Fonte: Elaborado pelo autor.

3.6 Casos de Uso

Um Caso de Uso é uma interação típica entre um usuário e um sistema. Ele captura alguma função visível ao usuário e, em especial, busca atingir uma meta do usuário.

Assim, um caso de uso pode ser definido como uma sequência de ações que o sistema executa e produz um resultado de valor para o ator (Universidade Federal de Minas Gerais, 2021), sendo essencial para compreender como o sistema interage com os usuários.

Conforme veremos na figura 7, ele é uma ferramenta importante para identificar e documentar as funcionalidades específicas que o sistema precisa oferecer para seus usuários.

Sua importância ainda é crucial para a análise de sistemas. Nesse contexto, o caso de uso possibilita uma análise detalhada sobre as interações que sucedem entre os usuários e os sistemas, detectando as ações necessárias para a execução de tarefas específicas (Conceito de, 2021), o que permite uma melhor compreensão dos processos envolvidos.

Figura 7 – Caso de Uso

Fonte: Elaborado pelo autor.

A documentação do caso de uso estabelece de forma clara e precisa as interações entre os usuários e o sistema, garantindo uma compreensão unificada das funcionalidades esperadas. Ao fornecer uma visão minuciosa das funcionalidades necessárias, as regras de caso de uso também desempenham um papel crucial no planejamento do projeto.

Em resumo, elas são essenciais em todas as fases do desenvolvimento de *software*, assegurando o sucesso do projeto ao garantir uma compreensão efetiva e o atendimento às necessidades dos usuários (MEDIUM, 2022), o que contribui para a criação de soluções mais alinhadas com as expectativas dos clientes.

O quadro 4 apresenta a especificação dos casos de uso registrados.

Quadro 4 – Casos de Uso

Caso de Uso – Realizar Login	
ID	UC 001
Descrição	Este caso de uso tem por objetivo permitir que o usuário se autentique no sistema utilizando suas credenciais (e-mail e senha).

Ator Primário	Funcionário
Pré-condição	O funcionário deve possuir uma conta cadastrada no sistema.
Cenário Principal	1. O use case inicia quando o funcionário preenche os campos e-mail e senha e seleciona o botão “login”. 2. O sistema valida as credenciais informadas. 3. Se as credenciais forem válidas, o sistema autentica o usuário e exibe a tela principal do sistema. 4. O sistema encerra o caso de uso.
Pós-condição	Nenhuma
Cenário Alternativo	2a – O sistema identifica que as credenciais informadas são inválidas. 2a.1 O sistema exibe uma mensagem de erro informando que o e-mail ou a senha estão incorretos. 2a.2 O usuário pode optar por tentar novamente inserindo as credenciais corretas.
Include	-0-
Extend	-0-
Caso de Uso – Gerenciar Produtos	
ID	UC 002
Descrição	Este caso de uso tem por objetivo permitir que o usuário do sistema adicione, edite, remova ou visualize produtos no cardápio.
Ator Primário	Funcionário
Pré-condição	O funcionário deve estar autenticado no sistema.
Cenário Principal	1. O caso de uso inicia quando o funcionário seleciona a opção " Produtos" no menu principal. 2. O sistema exibe a lista de produtos cadastradas. 3. O funcionário pode optar por adicionar, editar ou remover produtos. 4. O sistema exibe a lista de produtos atualizada. 5. O sistema encerra o caso de uso.
Pós-condição	A lista de produtos é atualizada no sistema com as alterações feitas
Cenário Alternativo	2a – Falha ao carregar a lista de produtos. 2a.1 O sistema apresenta uma mensagem de erro indicando que houve falha ao carregar a lista de produtos. 3a – Falha na validação das informações ao adicionar ou editar um produto. 3a.1 O sistema exibe uma mensagem de erro informando que as informações inseridas estão inválidas ou incompletas.
Include	-0-
Extend	-0-
Caso de Uso – Gerenciar Mesas	
ID	UC 003
Descrição	Este caso de uso tem por objetivo permitir que o usuário do sistema adicione, edite, remova ou visualize mesas no sistema.
Ator Primário	Funcionário
Pré-condição	O usuário deve estar autenticado no sistema.
Cenário Principal	1. O caso de uso inicia quando o funcionário seleciona a opção " Mesas" no menu principal. 2. O sistema exibe a lista de mesas cadastradas. 3. O funcionário pode optar por adicionar, editar ou remover mesas. 4. O sistema exibe a lista de mesas atualizada. 5. O sistema encerra o caso de uso.
Pós-condição	A lista de mesas é atualizada no sistema com as alterações feitas.
Cenário Alternativo	2a – Falha ao carregar a lista de mesas. 2a.1 O sistema apresenta uma mensagem de erro indicando que houve falha ao carregar a lista de mesas. 3a – Falha na validação das informações ao adicionar ou editar uma mesa. 3a.1 O sistema exibe uma mensagem de erro informando que as informações inseridas estão inválidas ou incompletas.

Include	-0-
Extend	-0-
Caso de Uso – Gerenciar Perfil	
ID	UC 004
Descrição	Este caso de uso tem por objetivo permitir que o usuário do sistema adicione, edite, remova ou visualize perfis do sistema.
Ator Primário	Funcionário
Pré-condição	O usuário deve estar autenticado no sistema.
Cenário Principal	1. O caso de uso inicia quando o usuário seleciona a opção "Configurações" no menu principal. 2. O sistema exibe as opções de perfil. 3. O usuário pode optar por adicionar, editar ou remover usuários. 4. O sistema exibe a lista de perfis atualizada. 5. O sistema encerra o caso de uso.
Pós-condição	A lista de mesas é atualizada no sistema com as alterações feitas.
Cenário Alternativo	2a – Falha ao carregar a lista de usuários. 2a.1 O sistema apresenta uma mensagem de erro indicando que houve falha ao carregar a lista de usuários. 3a – Falha na validação das informações ao adicionar ou editar um perfil. 3a.1 O sistema exibe uma mensagem de erro informando que as informações inseridas estão inválidas ou incompletas.
Include	-0-
Extend	-0-
Caso de Uso – Iniciar Atendimento	
ID	UC 005
Descrição	Este caso de uso tem por objetivo permitir que o cliente inicie o atendimento, dando início a realização dos pedidos.
Ator Primário	Cliente
Pré-condição	O cliente deve estar autenticado no sistema.
Cenário Principal	1. O caso de uso inicia quando o cliente seleciona a opção "Iniciar Atendimento". 2. O sistema atualiza o status da mesa para "Ocupado". 3. O sistema encerra o caso de uso.
Pós-condição	A mesa passa para o status de "Ocupado".
Cenário Alternativo	Não possui.
Include	-0-
Extend	-0-
Caso de Uso – Gerenciar Sacola	
ID	UC 006
Descrição	Este caso de uso permite que o cliente adicione, edite, remova ou visualize produtos em sua sacola.
Ator Primário	Cliente
Pré-condição	O cliente deve ter iniciado o atendimento.
Cenário Principal	1. O caso de uso inicia quando o cliente visualiza os produtos cadastrados. 2. O cliente clica na opção "Adicionar" ao lado de cada produto que deseja incluir na sacola. 3. O sistema adiciona o item selecionado a sacola. 4. O cliente pode visualizar sua sacola e optar por adicionar mais itens, editar a quantidade de produtos ou remover produtos da sacola 5. O sistema encerra o caso de uso.
Pós-condição	A sacola é atualizada com os itens adicionados, editados ou removidos, e

	fica disponível para visualização e alteração pelo cliente.
Cenário Alternativo	4a - O sistema não consegue salvar as alterações feitas na sacola e exibe uma mensagem de erro informando o cliente sobre a falha.
Include	-0-
Extend	-0-
Caso de Uso – Fazer Pedido	
ID	UC 007
Descrição	Este caso de uso tem como objetivo permitir que o cliente finalize o pedido, enviando os itens adicionados na sacola para o restaurante.
Ator Primário	Cliente
Pré-condição	O cliente deve ter adicionado pelo menos um item a sacola.
Cenário Principal	1. O caso de uso inicia quando o cliente seleciona a opção "Fazer Pedido" na sacola. 2. O sistema registra o pedido e notifica o restaurante. 3. O sistema exibe uma mensagem de confirmação ao cliente, indicando que o pedido foi enviado com sucesso. 4. O sistema encerra o caso de uso.
Pós-condição	O pedido é enviado ao restaurante, que passa a prepará-lo.
Cenário Alternativo	3a – O cliente cancela o pedido, se estiver no prazo antes do início da preparação do pedido.
Include	-0-
Extend	UC007 – Cancelar Pedido
Caso de Uso – Cancelar Pedido	
ID	UC 008
Descrição	Este caso de uso tem como objetivo permitir que o cliente ou o funcionário cancele o pedido antes do início da sua preparação.
Ator Primário	Cliente
Pré-condição	O pedido deve ter sido realizado e ainda estar dentro do prazo permitido para cancelamento.
Cenário Principal	1. O caso de uso inicia quando o cliente seleciona a opção “Meus Pedidos”. 2. O sistema exibe a lista de pedidos feitos. 3. O sistema verifica quais os pedidos que estão dentro do prazo permitido para cancelamento. 4. O cliente seleciona o pedido que deseja cancelar. 5. O sistema exibe uma mensagem de confirmação solicitando que o cliente confirme o cancelamento do pedido. 6. O sistema encerra o caso de uso.
Pós-condição	O pedido será removido da fila de iniciar preparação.
Cenário Alternativo	3a – O pedido está fora do prazo permitido para cancelamento.
Include	-0-
Extend	-0-
Caso de Uso – Iniciar Pedido	
ID	UC 009
Descrição	Este caso de uso tem por objetivo permitir que o funcionário inicie a preparação de um pedido realizado por um cliente.
Ator Primário	Funcionário
Pré-condição	Um pedido deve ter sido realizado por um cliente e estar disponível para visualização no sistema.
Cenário Principal	1. O caso de uso inicia quando o funcionário seleciona a opção "Pedidos" no menu principal. 2. O sistema exibe os pedidos feitos pelos clientes.

	3. O funcionário seleciona a opção “Iniciar Pedido. 4. O sistema atualiza o status do pedido 5. O pedido é movido para a lista de pedidos em preparação. 6. O sistema encerra o caso de uso.
Pós-condição	O pedido passa para o status de "Em preparação".
Cenário Alternativo	2a – O sistema não carrega a lista de pedidos pendentes 2a.1 O sistema exibe uma mensagem de erro indicando a falha no carregamento dos pedidos. 4a – O sistema não consegue atualizar o status do pedido 4a.1 O sistema exibe uma mensagem de erro informando a falha na atualização do status do pedido.
Include	UC010 – Concluir Pedido
Extend	-0-
Caso de Uso – Concluir Pedido	
ID	UC 0010
Descrição	Este caso de uso tem como objetivo permitir que o funcionário marque um pedido como "Pronto" após a conclusão da preparação.
Ator Primário	Funcionário
Pré-condição	O pedido deve estar em preparação.
Cenário Principal	1. O caso de uso inicia quando o funcionário seleciona a opção "Pedidos" no menu principal e, em seguida, escolhe "Pedidos Realizados". 2. O sistema exibe os pedidos que estão em preparação. 3. O funcionário seleciona “Concluir” para o pedido que foi finalizado. 4. O sistema atualiza o status do pedido para "Pronto". 5. O sistema encerra o caso de uso.
Pós-condição	O pedido passa para o status de "Pronto".
Cenário Alternativo	2a – Não há pedidos em produção 2a.1 O sistema informa que não há pedidos atualmente em preparação.
Include	UC011 – Entregar Pedido
Extend	-0-
Caso de Uso – Entregar Pedido	
ID	UC 011
Descrição	Este caso de uso tem como objetivo permitir que o funcionário marque um pedido como "Entregue" após a entrega do pedido ao cliente.
Ator Primário	Funcionário
Pré-condição	O pedido deve estar pronto.
Cenário Principal	1. O caso de uso inicia quando o funcionário seleciona a opção "Pedidos" no menu principal e, em seguida, escolhe "Pedidos Prontos". 2. O sistema exibe os pedidos que estão concluídos. 3. O funcionário seleciona “Entregar” para o pedido que foi finalizado. 4. O sistema atualiza o status do pedido para "Entregue". 5. O sistema encerra o caso de uso.
Pós-condição	O pedido passa para o status de "Entregue".
Cenário Alternativo	2a – Não há pedidos em prontos 2a.1 O sistema informa que não há pedidos atualmente prontos.
Include	-0-
Extend	-0-
Caso de Uso – Solicitar Fechamento de Conta	
ID	UC 012
Descrição	Este caso de uso tem como objetivo permitir que o cliente solicite o fechamento de sua conta após consumir os itens no restaurante.
Ator Primário	Cliente
Pré-condição	O cliente deve ter concluído seu pedido e deseja finalizar a conta.

Cenário Principal	1. O caso de uso inicia quando o cliente seleciona a opção "Meus Pedidos" no sistema. 2. O sistema exibe o resumo da conta, incluindo todos os pedidos feitos e o total a pagar. 3. O cliente seleciona a opção “Solicitar Fechamento de Conta” 4. O sistema exibe uma mensagem de confirmação informando que um funcionário será notificado para realizar o fechamento da conta. 5. O sistema encerra o caso de uso.
Pós-condição	A solicitação de fechamento de conta é registrada, e um funcionário será notificado para prosseguir com o fechamento e o pagamento.
Cenário Alternativo	3a – Cliente cancela o fechamento da conta 3a.1 O cliente decide não fechar a conta e cancela o processo antes da confirmação do pagamento.
Include	-0-
Extend	-0-
Caso de Uso – Fechar Conta	
ID	UC 013
Descrição	Este caso de uso tem como objetivo permitir que o funcionário finalize o pagamento de sua conta após consumir os itens no restaurante.
Ator Primário	Funcionário
Pré-condição	O cliente deve ter solicitado o fechamento da conta.
Cenário Principal	1. O caso de uso inicia quando o funcionário recebe uma notificação da solicitação de fechamento de conta. 2. O funcionário acessa os pedidos da mesa e seleciona a opção "Fechar Conta". 3. O sistema exibe o total a pagar. 4. O funcionário confirma o fechamento da conta após o pagamento ser realizado. 5. O sistema encerra o caso de uso.
Pós-condição	O status da mesa é alterado para "Livre", indicando que está disponível para novos clientes, e todos os produtos e pedidos associados à mesa são removidos do sistema.
Cenário Alternativo	4a – Cliente cancela o fechamento da conta 4a.1 O cliente decide não fechar a conta e cancela o processo antes da confirmação do pagamento.
Include	-0-
Extend	-0-

Fonte: Elaborado pelo autor.

4 Ferramentas e Métodos

Neste capítulo citaremos as ferramentas e seus respectivos recursos, bem como o método utilizado neste projeto.

4.1 Ferramentas

As ferramentas selecionadas para o desenvolvimento deste projeto desempenham um papel crucial na criação de uma aplicação eficiente e de alta qualidade. A escolha dessas tecnologias foi baseada na sua capacidade de atender às necessidades específicas do projeto, aliada à familiaridade com as ferramentas e aos recursos que elas oferecem para garantir a agilidade no desenvolvimento, a integração fluída e

a otimização das funcionalidades.

a) **Figma (Figma, 2024).** Aplicação responsável pela prototipação das telas, foi utilizada devido à previa familiaridade e disponibilidade de conteúdo na Internet para estudo de novas funcionalidades.

b) **Visual Studio (Microsoft, 2022).** Ambiente de desenvolvimento integrado (IDE) completo e poderoso para desenvolvedores de *software*. A ferramenta oferece suporte a uma ampla variedade de linguagens de programação. Além disso, possui recursos avançados de edição, depuração e gerenciamento de código, permitindo que os desenvolvedores criem, testem e implementem aplicativos com eficiência.

c) **Firebase (Firebase, 2022).** O Firebase Authentication é um serviço utilizado para gerenciar a autenticação do usuário na aplicação, permitindo o cadastro e login de maneira centralizada. O Realtime Database e o Cloud Firestore são bancos de dados NoSQL que gerenciam e armazenam dados em tempo real, sincronizando informações de forma eficiente.

O Firebase Storage é utilizado para armazenar arquivos, como imagens e documentos, de forma segura e escalável. Todos esses serviços são incluídos na versão 10.12.4 do Firebase, que possui licença GPL (*General Public License*). A escolha dessas ferramentas se deve à sua facilidade de uso, pois já incluem SDKs (*Software Development Kit*) e bibliotecas que simplificam a implementação e a integração, centralizando os processos em uma única plataforma.

d) **HTML (HTML, 2024).** Linguagem de marcação utilizada para estruturar e apresentar conteúdo na *web*, atualmente na versão HTML5, com licença livre sob os padrões do W3C. Foi escolhido por ser a base de qualquer página ou aplicação *web*, permitindo a criação de interfaces amigáveis e acessíveis. Sua compatibilidade universal com navegadores e dispositivos faz do HTML uma tecnologia indispensável para o desenvolvimento *web* moderno.

e) **React (React, 2024).** Biblioteca baseada em JavaScript para a criação de

interfaces de usuário, especialmente para aplicações *web*, na versão 18.3.1 com licença MIT. Foi escolhido por ser uma ferramenta facilitadora de desenvolvimento, que permite a construção de componentes reutilizáveis e uma excelente performance na atualização de interfaces, utilizando o conceito de *Virtual DOM* para otimizar as operações de renderização e proporcionando uma experiência interativa e responsiva ao usuário.

f) **TypeScript (TypeScript, 2024).** Superset de JavaScript que adiciona tipagem estática à linguagem, na versão 5.2.2 com licença MIT. Foi escolhido por proporcionar uma experiência de desenvolvimento mais robusta, permitindo a detecção precoce de erros durante a compilação e facilitando a manutenção do código. A tipagem estática ajuda os desenvolvedores a escreverem código mais claro e compreensível, melhorando a colaboração em equipe e a escalabilidade das aplicações.

g) **Express (Express, 2024).** *Framework* para Node.js que simplifica o desenvolvimento de aplicações *web* e *APIs*. Na versão 4.19.2, com licença MIT, o Express foi escolhido por sua leveza e flexibilidade, permitindo a criação de rotas de forma intuitiva e rápida. Ele oferece um conjunto robusto de recursos para lidar com solicitações HTTP, gerenciamento de *middleware* e integração com bancos de dados, facilitando a construção de aplicações escaláveis e de alto desempenho. A ampla documentação e a comunidade ativa também contribuem para a adoção do Express, tornando-o uma escolha popular entre desenvolvedores que buscam eficiência no desenvolvimento de servidores.

h) **Redux (Redux, 2024).** Biblioteca para gerenciamento de estado em aplicações JavaScript, na versão 2.3.0 com licença MIT. O Redux foi escolhido por sua capacidade de centralizar o estado da aplicação, facilitando a gestão e a previsibilidade das mudanças de estado em um único local. Essa abordagem é especialmente útil em aplicações complexas, onde múltiplos componentes precisam acessar e modificar o mesmo estado. O Redux utiliza o padrão Flux, permitindo que os desenvolvedores criem ações e redutores que descrevem como o estado deve mudar em resposta a ações específicas, promovendo a manutenção e a escalabilidade do código.

i) **Tailwind CSS (Tailwind CSS, 2022).** *Framework* CSS utilitário na versão 3.4.5 com licença MIT. Foi escolhido por sua abordagem de design que permite a criação de interfaces responsivas de forma rápida e eficiente, utilizando classes utilitárias pré-definidas. Essa abordagem promove a consistência e a manutenção do estilo da aplicação, permitindo que os desenvolvedores personalizem rapidamente o design sem a necessidade de escrever CSS complexo.

j) **Canva (Canva, 2024).** Plataforma de design gráfico online, com versão gratuita e planos pagos, licenciada sob o modelo *SaaS*. Foi escolhida por sua interface amigável e intuitiva, que permite a criação de gráficos, apresentações e conteúdos visuais de maneira rápida e acessível, sem a necessidade de conhecimentos avançados em design.

k) **BPMN.io (BPMN.io, 2024).** Ferramenta online para a modelagem de processos de negócios, licenciada sob a licença bpmn.io. Foi escolhida por sua capacidade de permitir a criação de diagramas de processos de negócios com o padrão BPMN (*Business Process Model and Notation*) de maneira intuitiva e prática. A plataforma oferece uma interface de fácil uso, ideal para profissionais que desejam documentar e visualizar fluxos de trabalho e processos empresariais.

l) **Microsoft Word (Microsoft Word, 2022).** *Software* de processamento de texto na versão 2022 com licença proprietária. Foi escolhido por ser uma ferramenta amplamente utilizada para a criação, edição e formatação de documentos de texto de forma profissional. Suas funcionalidades incluem verificação ortográfica e gramatical, suporte a colaboração em tempo real e integração com outros produtos da Microsoft, como OneDrive e SharePoint. O Microsoft Word oferece uma interface amigável e recursos avançados para a criação de documentos, tornando-o uma escolha popular para escritórios e ambientes acadêmicos.

m) **Miro (Miro, 2024).** Plataforma colaborativa de visualização e *brainstorming*, licenciada sob licença comercial. Foi escolhida por sua versatilidade na criação

de quadros virtuais que facilitam a colaboração em equipe, permitindo o desenvolvimento de fluxogramas, mapas mentais e diagramas em tempo real. A ferramenta é amplamente utilizada em projetos ágeis e *workshops*, promovendo uma abordagem visual para o planejamento e a resolução de problemas.

n) **Lucidchart (Lucidchart, 2024)**. Ferramenta online para criação de diagramas, fluxogramas e modelagem visual, utilizada principalmente para o design de diagramas UML, diagramas de caso de uso e fluxos de processo. Com licença *SaaS* (*Software as a Service*), o Lucidchart foi escolhido por sua interface intuitiva e facilidade de colaboração em tempo real, permitindo que equipes trabalhem simultaneamente na criação de representações visuais de sistemas e processos. A ferramenta também se destaca pela compatibilidade com múltiplas plataformas e pela ampla biblioteca de *templates*, tornando o processo de documentação e planejamento visual mais eficiente e acessível.

4.2 Métodos ou Desenvolvimento

No desenvolvimento deste projeto, foram selecionadas ferramentas que proporcionam consistência e flexibilidade, fundamentais para garantir uma base sólida e adaptável para a aplicação. A escolha dessas ferramentas visou atender às necessidades específicas do cliente, ao mesmo tempo em que permitisse futuras atualizações e melhorias sem grandes dificuldades.

Para o desenvolvimento do *front-end*, o React foi a biblioteca escolhida, dada sua popularidade e capacidade de criar interfaces dinâmicas e interativas de forma eficiente.

O React, com sua abordagem baseada em componentes reutilizáveis, facilita o gerenciamento do código e a criação de interfaces complexas sem comprometer o desempenho da aplicação. Isso permite que mantido uma estrutura clara e organizada ao longo do desenvolvimento, tornando o processo mais ágil e intuitivo.

Complementando o *front-end*, o Tailwind CSS foi escolhido como *framework* de design. Ele se destaca por seu uso de classes utilitárias que facilitam a personalização e à criação de interfaces modernas sem a necessidade de escrever CSS extenso e complexo. Com o Tailwind, a aplicação pode ser desenvolvida de maneira responsiva, mantendo a consistência visual em diferentes dispositivos e tamanhos de tela.

No aspecto da lógica de negócios e segurança de tipagem, foi utilizado o

Express juntamente com TypeScript. O Express, um *framework* minimalista para Node.js, facilita a criação de APIs e a gestão das rotas da aplicação.

Com o uso do TypeScript, a tipagem estática foi integrada, permitindo a detecção precoce de erros durante o desenvolvimento e aumentando a segurança e a confiabilidade do código.

Essa combinação proporciona uma maior robustez na implementação da lógica de negócios e facilita a manutenção e escalabilidade da aplicação, garantindo que o sistema se mantenha estável conforme cresce.

Para o armazenamento de dados e autenticação, o Firebase desempenhou um papel central. Os serviços do Cloud Firestore, um banco de dados NoSQL, foram utilizados para a gestão em tempo real de pedidos e mesas, enquanto o Firebase Realtime Database garantiu que as informações fossem sincronizadas instantaneamente entre diferentes usuários.

O Firebase Authentication simplificou o gerenciamento de usuários, oferecendo uma solução segura para login e cadastro. Enquanto o Firebase Storage foi utilizado para armazenar arquivos relacionados ao sistema, como imagens de produtos e comprovantes de pedidos.

4.2.1 Estrutura das pastas

A organização da estrutura de pastas no projeto foi planejada de forma a garantir clareza, modularidade e facilidade de manutenção.

A divisão das funcionalidades do sistema em componentes menores facilita o desenvolvimento colaborativo e a escalabilidade da aplicação, além de permitir a rápida localização de arquivos. A seguir, detalha-se a estrutura de pastas implementada no *front-end* do projeto:

Src: A pasta principal que contém todo o código-fonte da aplicação.

Components: Esta pasta abriga todos os componentes reutilizáveis da aplicação. Componentes como botões, formulários, modais e tabelas foram separados aqui, promovendo a reutilização e a padronização.

Pages: Cada página da aplicação (por exemplo, tela de *login*, listagem de pedidos, gerenciamento de mesas) está organizada nesta pasta. Isso permite uma navegação clara e uma divisão bem definida das seções principais da aplicação.

Redux: Esta pasta contém toda a lógica relacionada ao gerenciamento de estado da aplicação utilizando Redux. Nela, são encontrados os arquivos que definem

as ações, redutores e store da aplicação. As ações representam os eventos que podem alterar o estado, enquanto os redutores são funções puras que determinam como o estado deve mudar em resposta a essas ações.

Routes: Esta pasta armazena a configuração das rotas da aplicação, que utilizam o React Router. As rotas são configuradas de forma a garantir o acesso controlado a diferentes páginas, de acordo com o nível de permissão do usuário.

Services: Serviços relacionados à comunicação com o Firebase. Aqui estão localizadas as funções responsáveis por realizar operações de CRUD (*Create, Read, Update, Delete*) no Firestore, autenticação e o gerenciamento de pedidos.

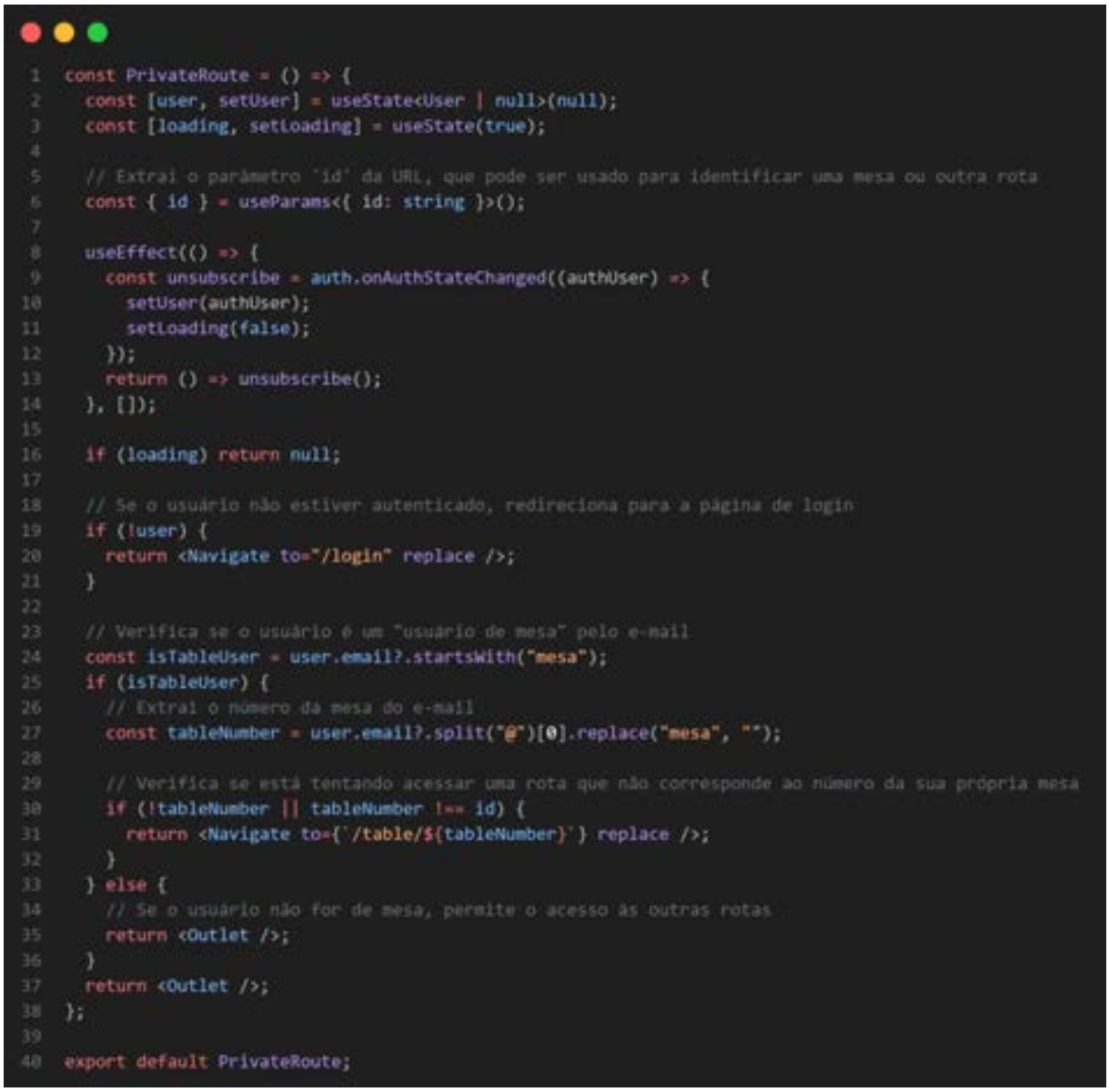
Style: Onde se encontram os arquivos de estilo global e configurações específicas do Tailwind CSS. Esta pasta permite a customização de temas e ajustes de design de maneira centralizada.

Types: Definições de tipos TypeScript utilizados por toda a aplicação. Aqui ficam centralizadas as interfaces e tipos personalizados que ajudam a tipar componentes, contextos e funções. Manter os tipos em uma pasta dedicada melhora a organização do código, facilita a manutenção e garante que as definições de tipos estejam acessíveis em qualquer parte do projeto.

Utils: Funções utilitárias, que são comumente usadas em diversos pontos do sistema, como formatação de datas, manipulação de *strings* e cálculos específicos. Essas funções estão centralizadas nesta pasta para facilitar o reuso e a manutenção.

Na figura 8, foi implementada uma rota privada que restringe o acesso a certas áreas da aplicação com base na autenticação do usuário e no seu tipo de conta.

Figura 8 – Rotas Privadas



```tsx
const PrivateRoute = () => {
  const [user, setUser] = useState<User | null>(null);
  const [loading, setLoading] = useState(true);

  // Extrai o parâmetro 'id' da URL, que pode ser usado para identificar uma mesa ou outra rota
  const { id } = useParams<{ id: string }>();

  useEffect(() => {
    const unsubscribe = auth.onAuthStateChanged((authUser) => {
      setUser(authUser);
      setLoading(false);
    });
    return () => unsubscribe();
  }, []);

  if (loading) return null;

  // Se o usuário não estiver autenticado, redireciona para a página de login
  if (!user) {
    return <Navigate to="/login" replace />;
  }

  // Verifica se o usuário é um "usuário de mesa" pelo e-mail
  const isTableUser = user.email?.startsWith("mesa");
  if (isTableUser) {
    // Extrai o número da mesa do e-mail
    const tableNumber = user.email?.split("@")[0].replace("mesa", "");

    // Verifica se está tentando acessar uma rota que não corresponde ao número da sua própria mesa
    if (!tableNumber || tableNumber !== id) {
      return <Navigate to={`/table/${tableNumber}`} replace />;
    }
  } else {
    // Se o usuário não for de mesa, permite o acesso às outras rotas
    return <Outlet />;
  }
  return <Outlet />;
};

export default PrivateRoute;
```

Fonte: Elaborado pelo autor.

Essa implementação garante a segurança da navegação ao restringir o acesso com base no tipo de usuário (usuário geral ou usuário de mesa) e no estado de autenticação. Ela é essencial para assegurar que apenas usuários autenticados tenham acesso às rotas protegidas e que os usuários de mesa possam acessar apenas a rota correspondente à sua mesa específica.

A figura 9, representa a Exclusão de Usuário. Função lida com erros que garante que o estado de carregamento seja atualizado corretamente ao final da operação.

Figura 9 – Exclusão de Usuário

```
const handlePlaceOrder = async () => {
    if (!tableDocId || cartItems.length === 0) return;

    setLoading(true);
    setShowModal(false);

    try {
      const tableRef = doc(db, "tables", tableDocId);

      const tableDoc = await getDoc(tableRef);

      // Recupera a lista atual de produtos da mesa, ou um array vazio se não houver produtos
      const currentProducts = tableDoc.data()?.products || [];

      // Cria um novo array de produtos a partir dos itens no carrinho
      const newProducts = cartItems.map((item) => ({
        ...item,
        status: "pendente", // Define o status inicial de cada produto como "pendente"
        image: products.find((product) => product.id === item.id)?.image || "",
        orderNumber: generateOrderNumber(), // Gera um número único de pedido para cada produto
      }));

      // Combina os produtos existentes na mesa com os novos produtos do carrinho
      const mergedProducts = [...currentProducts, ...newProducts];

      await updateDoc(tableRef, {
        products: mergedProducts,
      });

      dispatch(clearCart());

      setModalMessage("Pedido realizado com sucesso!");
      setShowModal(true);

      setTimeout(() => {
        setShowModal(false);
      }, 3000);
    } catch (error) {
      console.error(error);
    } finally {
      setLoading(false);
    }
  };
```

Fonte: Elaborado pelo autor.

A função assíncrona handlePlaceOrder lida com a ação de finalizar o pedido a partir dos itens presentes no carrinho. Inicialmente, ele verifica se há uma identificação válida da mesa (tableDocId) e se o carrinho possui itens.

Em seguida, a função busca os dados da mesa no banco de dados Firestore, recuperando a lista atual de produtos relacionados à mesa. Ela então cria um *array* de produtos a partir dos itens no carrinho, atribuindo um *status* de "pendente" a cada um, uma imagem associada e um número único de pedido.

Após isso, os produtos existentes são combinados com os novos produtos e

atualizados no documento correspondente à mesa no Firestore. Em seguida, o carrinho é limpo, a função exibe uma mensagem de sucesso através de um *modal* e, após um intervalo de 3 segundos, o *modal* é fechado.

O código (figura 10) cria uma *slice* de Redux chamada cartSlice, ele define funções (*reducers*) que permitem adicionar itens a sacola, removê-los, incrementar ou decrementar suas quantidades e limpar o carrinho.

Figura 10 – Exclusão de Usuário

```
const cartSlice = createSlice({
  name: 'cart',
  initialState,
  reducers: {
    // Função para adicionar um item ao carrinho
    addToCart: (
      state: CartState,
      action: PayloadAction<{ id: string; name: string; price: number; image: string; quantity: number }>
    ) => {
      // Verifica se o item já existe no carrinho
      const existingItem = state.items.find(item => item.id === action.payload.id);

      if (existingItem) {
        // Se o item já existir, incrementa a quantidade dele
        existingItem.quantity += action.payload.quantity;
      } else {
        state.items.push({ ...action.payload, quantity: action.payload.quantity });
      }
      localStorage.setItem('cartItems', JSON.stringify(state.items));
    },

    // Função para remover um item do carrinho
    removeFromCart: (state: CartState, action: PayloadAction<{ id: string }>) => {
      state.items = state.items.filter(item => item.id !== action.payload.id);

      localStorage.setItem('cartItems', JSON.stringify(state.items));
    },

    // Função para incrementar a quantidade de um item no carrinho
    incrementQuantity: (state: CartState, action: PayloadAction<{ id: string }>) => {
      const existingItem = state.items.find(item => item.id === action.payload.id);

      if (existingItem) {
        existingItem.quantity += 1;
      }
      localStorage.setItem('cartItems', JSON.stringify(state.items));
    },

    // Função para decrementar a quantidade de um item no carrinho
    decrementQuantity: (state: CartState, action: PayloadAction<{ id: string }>) => {
      const existingItem = state.items.find(item => item.id === action.payload.id);

      if (existingItem && existingItem.quantity > 1) {
        existingItem.quantity -= 1;
      } else {
        state.items = state.items.filter(item => item.id !== action.payload.id);
      }
      localStorage.setItem('cartItems', JSON.stringify(state.items));
    },

    // Função para limpar o carrinho
    clearCart: (state: CartState) => {
      state.items = [];
      localStorage.removeItem('cartItems');
    },
  },
});
```

Fonte: Elaborado pelo autor.

Cada vez que uma dessas operações é realizada, o estado atualizado é salvo no localStorage, garantindo que os itens do carrinho sejam persistidos entre sessões

do usuário.

Especificamente, o *reducer* addToCart verifica se um item já existe no carrinho e, se sim, incrementa sua quantidade; caso contrário, o item é adicionado. O removeFromCart exclui um item pelo seu ID, enquanto os *reducers* incrementQuantity e decrementQuantity ajustam a quantidade de um item ou o removem se a quantidade for 1. O clearCart limpa todos os itens da sacola, removendo-os também do localStorage.

Para realizar a exclusão de um usuário, no código apresentado (figura 11) implementa uma função assíncrona chamada deleteUser para excluir um usuário de um sistema, usando *Firebase Authentication* e *Firebase Firestore*.

Figura 11 – Exclusão de Usuário

```
// Função para excluir um usuário
export const deleteUser = async (req: Request, res: Response): Promise<void> => {
  const { email } = req.body;

  try {
    const userRecord = await admin.auth().getUserByEmail(email);
    const uid = userRecord.uid;

    await admin.auth().deleteUser(uid);
    await admin.firestore().collection('users').doc(uid).delete();

    res.status(200).json({ message: 'Usuário excluído com sucesso.', uid });
  } catch (error) {
    console.error('Erro ao excluir usuário:', error);
    res.status(500).json({ error: 'Erro ao excluir usuário.' });
  }
};
```

Fonte: Elaborado pelo autor.

A função manipula uma solicitação HTTP (do tipo *Request*), que contém o e-mail do usuário a ser excluído, e uma resposta HTTP (do tipo *Response*), para enviar uma resposta de sucesso ou erro.

6 Resultados e Discussão

No início do projeto SUSHITECH, o objetivo principal era criar uma solução para otimizar o processo de pedidos em restaurantes, especificamente no contexto de restaurantes japoneses, onde a interação rápida e eficiente entre clientes e cozinha é essencial.

Com a documentação dos requisitos funcionais e não funcionais foi possível entender quais eram as necessidades e o que deveríamos atender.

Com a criação de alguns artefatos de engenharia de *software* conseguimos mapear os processos e minimizar possíveis riscos no desenvolvimento do protótipo, possibilitando identificar as funcionalidades fundamentais para o desenvolvimento do sistema.

O problema central enfrentado pelos restaurantes era o tempo gasto com o atendimento manual para anotar os pedidos e a falta de uma integração eficiente entre cliente e cozinha.

O SUSHITECH solucionou esse problema ao permitir que os clientes realizem seus pedidos diretamente através de suas mesas utilizando dispositivos disponibilizados pelo restaurante, eliminando a necessidade de interações manuais e possibilitando que a cozinha receba os pedidos de forma imediata e organizada.

Isso reduz o tempo necessário para processar os pedidos e aumenta a precisão na preparação dos pratos.

A aplicação se inicia na tela de *login* (figura 12), onde diferentes tipos de acesso são disponibilizados. O sistema conta com duas categorias de usuários: os funcionários do restaurante e os clientes das mesas.

Ao fazer *login*, o funcionário tem acesso a todas as funcionalidades do sistema, como gerenciamento de pedidos, controle de mesas, e administração de produtos e contas.

Esse acesso é totalmente controlado pelo próprio usuário, oferecendo autonomia para a gestão do restaurante.

Já no caso dos clientes, o *login* das mesas direciona o usuário diretamente para a tela de realização de pedidos, onde podem visualizar o cardápio e fazer seus pedidos de forma autônoma.

Essa segmentação de acessos facilita o uso do sistema tanto para os funcionários quanto para os clientes, tornando o processo de atendimento mais ágil

.

Figura 12 – Tela de Login

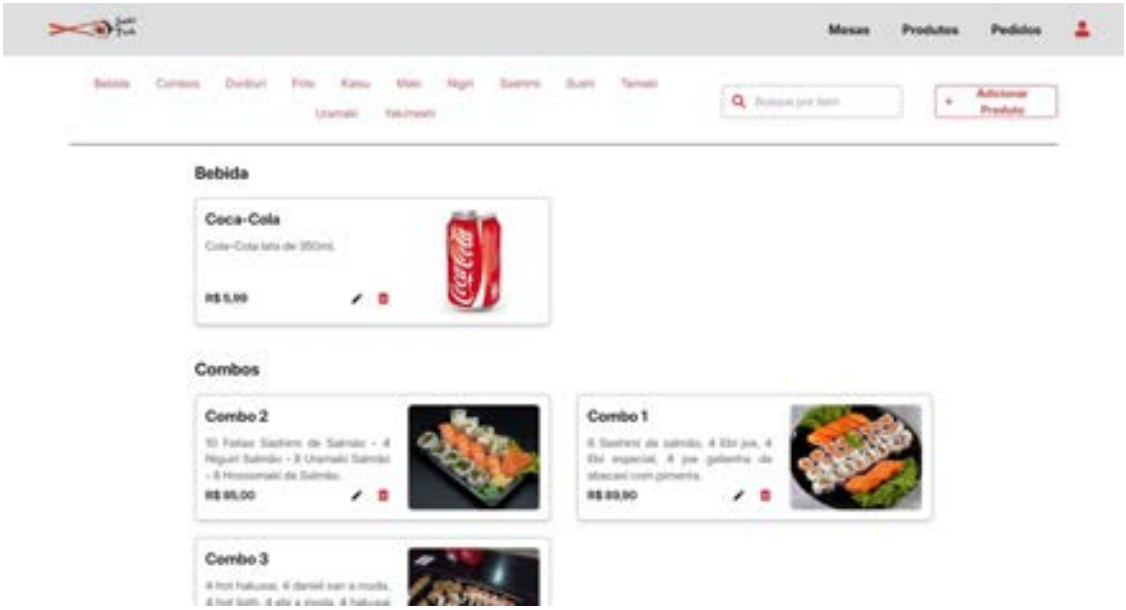
Fonte: Elaborado pelo autor.

Logo após realizar o *login*, o usuário funcionário é redirecionado para a página principal da aplicação, que se trata da página de gerenciamento de produtos (figura 13).

Nesta página, o funcionário poderá visualizar, adicionar, editar ou remover itens do cardápio, facilitando o controle sobre os produtos oferecidos pelo restaurante.

A interface é intuitiva, permitindo que o gerenciamento seja feito de forma prática e eficiente, ajudando a manter o cardápio atualizado de acordo com a necessidade do restaurante.

Figura 13 – Tela de Gerenciamento de Produtos

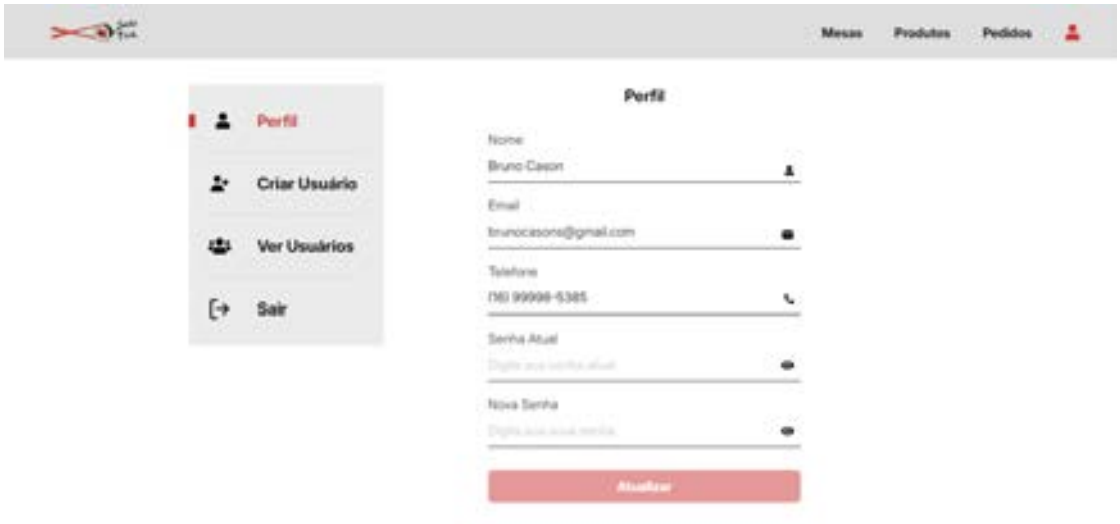


Fonte: Elaborado pelo autor.

Além da página de gerenciamento de produtos, o funcionário poderá acessar a área de gerenciamento do perfil mostrado através da figura 14, onde consegue visualizar seus próprios dados. Nesse mesmo espaço, é possível adicionar novos usuários, como outros colaboradores, ao sistema de forma simples.

Todos os perfis cadastrados são listados de maneira organizada, permitindo que o funcionário edite as informações ou exclua perfis quando necessário.

Figura 14 – Tela de Gerenciamento do Perfil

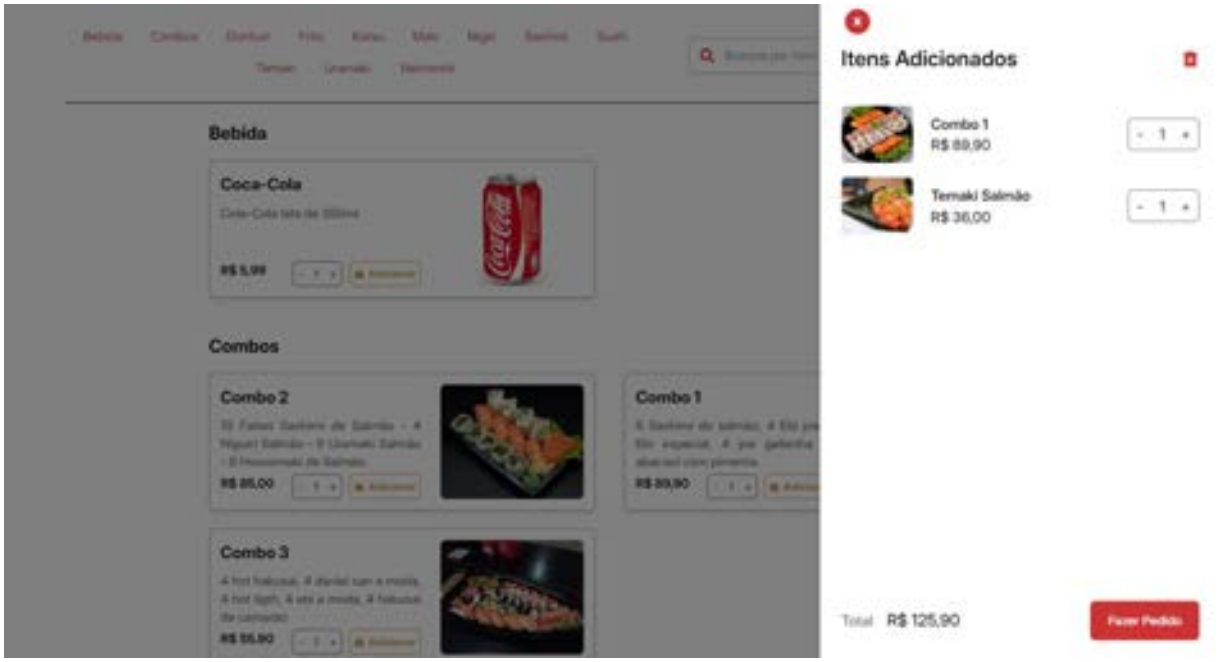


Fonte: Elaborado pelo autor.

Na área destinada aos usuários das mesas, a experiência é voltada para a realização dos pedidos. Nessa tela, os clientes conseguem visualizar todos os produtos cadastrados no sistema, como pratos, bebidas e outros itens disponíveis no cardápio, conforme a figura 15.

Com um *layout* simples e intuitivo, o cliente pode adicionar os produtos desejados à sacola de compras. Depois de escolher todos os itens, ele tem a opção de fazer o pedido diretamente pela interface da sacola, agilizando o processo e tornando a experiência de fazer pedidos mais prática e eficiente.

Figura 15 – Tela de Gerenciamento de Produtos



Fonte: Elaborado pelo autor.

O sistema foi desenvolvido utilizando a biblioteca React, combinada com a tipagem do TypeScript, o que garantiu uma estrutura sólida e flexível para o desenvolvimento da interface do usuário. A escolha dessas tecnologias proporcionou uma maior segurança no código e facilitou a criação de componentes reutilizáveis, permitindo rápidas adaptações e futuras atualizações solicitadas pelos restaurantes.

Junto da integração com os serviços do Firebase, foi essencial para a autenticação de usuários e o armazenamento de dados, como pedidos e perfis, por meio do Firestore, um banco de dados NoSQL altamente escalável e adequado às demandas de sistemas de pedidos online.

Durante o desenvolvimento, desafios como a adaptação das interfaces para diferentes perfis de usuário (funcionários e clientes) e a implementação de funcionalidades que otimizem o processo de pedido foram superados.

O próximo passo é o aprimoramento do gerenciamento de produtos, permitindo que o restaurante crie promoções e ofertas temporárias diretamente pelo sistema, e a adição de um módulo específico para rodízios, onde os clientes poderiam selecionar opções ilimitadas de pratos dentro do sistema, seguindo o modelo de rodízio japonês.

Além disso, a implementação de um painel de relatórios gerenciais seria uma excelente ferramenta para os administradores do restaurante, fornecendo dados detalhados sobre vendas, performance de produtos, promoções e fluxo de pedidos.

Considerações finais

Ao observar o dia a dia dos restaurantes japoneses, ficou claro que muitos enfrentam problemas como a demora no atendimento e a falta de informações sobre o *status* dos pedidos, algo que os clientes costumam reclamar. Isso acontece porque muitos restaurantes ainda dependem de métodos manuais para gerenciar pedidos, o que pode causar erros e atrasos.

Para resolver esses problemas, foi criado o SushiTech, um sistema *web* de gerenciamento de pedidos que visa tornar o processo de atendimento mais rápido e eficiente. Com ele, os clientes podem fazer seus pedidos diretamente da mesa e acompanhar o *status* em tempo real, enquanto os funcionários conseguem gerenciar tudo de forma mais organizada e centralizada.

O desenvolvimento do sistema focou em tornar o sistema fácil de usar, além de permitir que os restaurantes façam mudanças e atualizações no sistema de maneira rápida, sempre que necessário. Ele também garante que os dados dos clientes e dos pedidos sejam armazenados de forma segura, facilitando a administração do restaurante.

Durante o desenvolvimento, foi aprendido muito sobre como criar soluções simples e eficientes para os usuários, sem perder de vista as necessidades tanto dos clientes quanto dos funcionários do restaurante, superando desafios de como garantir que o sistema fosse fácil de usar para diferentes tipos de pessoas, com funções específicas para clientes e funcionários, e também a implementação de funcionalidades

que tornaram o processo de pedido mais rápido e sem erros.

Com o sistema, os restaurantes agora conseguem reduzir o tempo de espera e melhorar a experiência dos clientes, tornando o atendimento mais ágil e transparente. Este trabalho contribui para a área de tecnologia aplicada à gastronomia, mostrando como a automação e a inovação podem melhorar a gestão dos restaurantes e a satisfação dos clientes.

O aprendizado obtido durante o desenvolvimento do sistema foi importante para entender como as soluções tecnológicas podem ser aplicadas de forma prática para resolver problemas reais. Além disso, o projeto mostrou como a integração de diferentes funções e processos pode otimizar o trabalho no dia a dia de um restaurante.

Referências

ALURA. **O que são regras de negócio?** 2021. Disponível em: https://www.alura.com.br/artigos/o-que-sao-regras-de-negocio.

BLOG DA TRYBE. **Requisitos não funcionais**: o guia completo! 2021. Disponível em: https://blog.betrybe.com/tecnologia/requisitos-nao-funcionais/.
BPMN.IO. Modelos e Negócios. 2024 Disponível em: <https://bpmn.io>.

CANVA. Gestão. 2024. Disponível em: <https://www.canva.com/pt_br/>.

CONCEITO DE. **Caso de uso** - O que é, propósito, no desenvolvimento e requisitos. 2021. Disponível em: https://conceito.de/caso-de-uso.
UNIVERSIDADE FEDERAL DE MINAS GERAIS. **Caso de Uso**. 2021. Disponível em:https://homepages.dcc.ufmg.br/~amendes/GlossarioUML/glossario/conteudo/caso_de_uso/caso_de_uso.htm.

CURTO CONSELHO. **Qual o objetivo da elicitação de requisitos?** 2021. Disponível em: https://conceito.de/caso-de-uso.

DEVMEDIA. **Elicitação de Requisitos:** Levantamento de requisitos e técnicas de Elicitação. 2021. Disponível em: https://homepages.dcc.ufmg.br/~amendes/GlossarioUML/glossario/conteudo/caso_de_uso/caso_de_uso.htm.

EXPRESS. Framework web rápido, flexível e minimalista para Node.js. 2024. Disponível em: < https://expressjs.com/pt-br/>.

FIGMA. Prototipação de telas. 2024. Disponível em: <https://www.figma.com/>.

FIREBASE. Modelos e Negócios. 2022. Disponível em: <https://firebase.google.com/>.

HTML. Linguagem de Marcação de Hipertexto. 2024. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/HTML/>.

LEFFINGWELL, Dean. **Agile Software Requirements**: Lean Requirements Practices for Teams, Programs, and the Enterprise. Addison-Wesley Professional, 2011.

LUCIDCHART. Modelagem de sistemas. 2024. Disponível em: < https://www.lucidchart.com/>.

MICROSOFT. Bem-vindo ao IDE do Visual Studio. 2022. Disponível em: <https://docs.microsoft.com/pt-br/visualstudio/get-started/visual-studioide?view=vs2022>.

MIRO. Modelo de Metas SMART. 2024. Disponível em <https://miro.com/pt/modelos/metas-smart/>.

O ANALISTA DE MODELOS DE NEGÓCIOS. **O que é o Business Model Canvas**. 2021. Disponível em: https://analistamodelosdenegocios.com.br/o-que-e-o-business-model-canvas/.

PLAYSTUDIO. **Business Model Canvas**: aprenda o que é e como aplicar. 2021. Dis-ponível em: https://www.playstudio.io/blog/business-model-canvas.

PROJECT BUILDER. **Sistema de Gestão de Projetos**. 2021. Disponível em: https://www.projectbuilder.com.br/?utm_source=google&utm_medium=cpc&utm_campaign=brand&gad_source=1&gclid=Cj0KCQjwjLGyBhCYARIsAPqTz1-yaJTr1nRhGLEB6QoIx5cUIIaFNiLmkbo-AwA-Kpic3O0HDA_5lpYaAtF-EALw_wcB.

REACT. Linguagens e soluções. 2024. Disponível em: <https://react.dev/>.

REDUX. Aplicações Java. 2024. Disponível em: <https://redux.js.org>.

ROCKETSEAT. **Regras de Negócio**: O que você precisa saber. 2021. Disponível em: https://blog.rocketseat.com.br/regras-de-negocios-o-que-voce-precisa-saber/.

TAILWIND CSS. FrameWork CSS. 2022. Disponível em: <https://tailwindcss.com>.

TYPESCRIPT. Linguagens de sucesso. 2024. Disponível em: <https://www.typescriptlang.org/>.

UNIVERSIDADE FEDERAL DE MINAS GERAIS. **Caso de Uso**. 2021. Disponível em:https://homepages.dcc.ufmg.br/~amendes/GlossarioUML/glossario/conteudo/caso_de_uso/caso_de_uso.htm.

VISURE SOLUTIONS. **O que são Requisitos Não Funcionais**: Exemplos, Definição, Guia Completo. 2021. Disponível em: https://visuresolutions.com/pt/blog/requisitos-n%C3%A3o-Funcionais/.