Universidade de Brasília – UnB

Faculdade de Ciências e Tecnologias em Engenharia – FCTE

Engenharia de Software

Desenvolvimento de um Sistema para Gerenciamento Interno de Delivery em Restaurantes: Uma Proposta de Aplicação Tecnológica Responsiva

Autor: Guilherme Keyti Cabral Kishimoto e Iago de Paula Cabral

Orientador: Dr. Ricardo Matos Chaim

Brasília, DF

2025

Guilherme Keyti Cabral Kishimoto e Iago de Paula Cabral

Desenvolvimento de um Sistema para Gerenciamento Interno de Delivery em Restaurantes: Uma Proposta de Aplicação Tecnológica Responsiva

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Universidade de Brasília – UnB

Faculdade de Ciências e Tecnologias em Engenharia – FCTE

Orientador: Dr. Ricardo Matos Chaim

Brasília, DF

2025

Guilherme Keyti Cabral Kishimoto e Iago de Paula Cabral

Desenvolvimento de um Sistema para Gerenciamento Interno de Delivery em Restaurantes: Uma Proposta de Aplicação Tecnológica Responsiva/ Guilherme Keyti Cabral Kishimoto e Iago de Paula Cabral. – Brasília, DF, 2025-

143 p. : il. (algumas color.) ; 30 cm.

Orientador: Dr. Ricardo Matos Chaim

Trabalho de Conclusão de Curso – Universidade de Brasília – UnB
Faculdade de Ciências e Tecnologias em Engenharia – FCTE , 2025.

1. Gerenciamento de Delivery. 2. Engenharia de Software. I. Dr. Ricardo Matos Chaim. II. Universidade de Brasília. III. Faculdade de Ciências e Tecnologias em Engenharia. IV. Desenvolvimento de um Sistema para Gerenciamento Interno de Delivery em Restaurantes: Uma Proposta de Aplicação Tecnológica Responsiva

CDU 02:141:005.6

Agradecimentos

Guilherme Keyti Cabral Kishimoto

Quero agradecer, primeiramente, a Deus. Porque, se não fossem os desafios que Ele colocou em meu caminho, eu não teria desenvolvido a força necessária para continuar. Se não fossem as dificuldades que Ele permitiu que eu enfrentasse, eu não teria adquirido a sabedoria que carrego hoje. Cada momento difícil, cada obstáculo superado, fez parte do processo que Ele preparou para mim, não só durante o curso, mas ao longo de toda a minha vida. E é por isso que eu chego até aqui grato, fortalecido e satisfeito com a pessoa que me tornei.

Agradeço também à minha família, que sempre esteve ao meu lado. Por acreditarem em mim, por me apoiarem mesmo sem entenderem tudo o que eu estava passando e por me darem motivos reais para continuar. Sem o amor, o cuidado e o suporte de vocês, eu não teria conseguido. Amo vocês.

Um agradecimento muito especial ao Iago, que esteve comigo em todos os momentos dessa jornada. Sem a sua parceria, esse trabalho não teria sido realizado da forma como foi. A caminhada teria sido mais pesada, mais difícil e, com certeza, muito mais solitária. Ter alguém ao lado nos momentos mais desafiadores, nas noites sem dormir, dividindo ideias, tarefas e inseguranças, fez toda a diferença. Obrigado por estar junto do começo ao fim.

Sou também muito grato aos meus amigos, que foram apoio, escape e incentivo ao longo dessa caminhada. Cada momento de descontração, cada gesto de carinho teve um papel importante, seja para o meu crescimento, seja para me alegrar nos dias difíceis. Obrigado por estarem presentes, mesmo nos detalhes, e por me lembrarem do valor da leveza no meio da correria.

Agradeço também ao professor Ricardo Chaim, por ter nos aceitado como orientandos e acompanhado formalmente esse trabalho. E aos professores da banca, por dedicarem seu tempo e por cada sugestão e contribuição para o amadurecimento do nosso projeto.

Por fim, agradeço a todos que, de alguma forma, contribuíram para que esse TCC se tornasse realidade. Cada conversa, conselho, crítica construtiva ou palavra de incentivo fez parte dessa conquista.

Iago de Paula Cabral

Ao encerrar este ciclo de grande aprendizado e dedicação, meu primeiro pensamento se volta à gratidão. Este trabalho não é fruto de um esforço solitário, mas sim o resultado de um processo construído com o apoio de muitas pessoas essenciais. A Deus, pela força e serenidade concedidas nos momentos de maior desafio.

À minha mãe, Maristela, minha base e meu porto seguro. Dedico a ela cada vitória, pois foi seu trabalho sob o sol que me trouxe a calmaria da sombra e a força para continuar. Meu sucesso é seu.

De forma muito especial, agradeço a você, Guilherme. Esta jornada não teria sido a mesma sem a sua parceria. Dividimos não apenas as tarefas, mas também as incertezas, as longas noites de estudo e, finalmente, a alegria de ver nosso esforço concluído. Sua dedicação, seu ponto de vista e nossa complementaridade foram a força motriz deste trabalho. Esta conquista é tão sua quanto minha.

Agradeço ao meu orientador, Professor Ricardo Chaim, pela orientação fundamental, e aos membros da banca, pelo tempo e pelas valiosas contribuições.x Aos meus amigos, obrigado pelo apoio e parceria. A todos que, de alguma forma, fizeram parte desta conquista, meu muito obrigado.

Resumo

A gestão de pedidos e entregas em restaurantes pode ser um processo desafiador, impactando diretamente a eficiência operacional e a experiência do cliente. Muitos estabelecimentos enfrentam dificuldades na organização de pedidos, comunicação interna e otimização do fluxo de entregas, o que pode resultar em atrasos e insatisfação do consumidor. Com base nesse cenário, este trabalho propõe o desenvolvimento de uma aplicação responsiva voltada para o gerenciamento interno de delivery em restaurantes. O objetivo é proporcionar maior controle sobre os pedidos, melhorar a organização operacional e facilitar a tomada de decisões estratégicas. Para alcançar esse objetivo, foram realizadas pesquisa bibliográfica, reengenharia de processos e entrevistas com gestores de restaurantes, a fim de compreender o mercado e identificar as principais dificuldades enfrentadas. A metodologia adotada baseou-se em técnicas ágeis de desenvolvimento de software, permitindo a criação de um sistema com intuito de apoiar as operações administrativas, aumentar a eficiência operacional do estabelecimento e fornecer dados analíticos sobre o desempenho do restaurante.

Palavras-chave: Aplicação Responsiva. Automação de Processos. Eficiência Organizacional. Gestão Interna de Restaurantes. Gerenciamento de Delivery. Metodologias Ágeis.

Abstract

The management of orders and deliveries in restaurants can be a challenging process, directly impacting operational efficiency and customer experience. Many establishments struggle with order organization, internal communication, and delivery flow optimization, which can lead to delays and customer dissatisfaction. Based on this scenario, this work proposes the development of a responsive application for internal delivery management in restaurants. The objective is to provide greater control over orders, improve operational organization, and facilitate strategic decision-making. To achieve this, bibliographic research, process reengineering, and interviews with restaurant managers were conducted to understand the market and identify the main difficulties faced. The adopted methodology was based on agile software development techniques, enabling the creation of a system designed to support administrative operations, increasing the establishment's operational efficiency, and providing analytical data on the restaurant's performance.

Key-words: Responsive Application. Process Automation. Organizational Efficiency. Internal Restaurant Management. Delivery Management. Agile Methodologies.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

ABDI	Agência Brasileira de Desenvolvimento Industrial
ABRASEL	Associação Brasileira de Bares e Restaurantes
ACID	Atomicidade, Consistência, Isolamento e Durabilidade
API	*Application Programming Interface*
CI	*Continuous Integration*
CAGR	*Compound Annual Growth Rate*
DER	Diagrama de Entidade-Relacionamento
DevOps	*Development and Operations*
DLD	Diagrama de Lógica de Dados
FGV	Fundação Getulio Vargas
GUI	*Graphical User Interface*
IHC	Interação Humano-Computador
ISO	*International Organization for Standardization*
JSON	*JavaScript Object Notation*
LGPD	*Lei Geral de Proteção de Dados*
RESTFul	*Representational State Transfer*
RF	Requisitos Funcionais
RNF	Requisitos Não Funcionais
SEBRAE	Serviço Brasileiro de Apoio às Micro e Pequenas Empresas
SQL	*Structured Query Language*
TCC	Trabalho de Conclusão de Curso
TI	Tecnologia da Informação
TQM	*Total Quality Management*

UAT	*User Acceptance Test*
UUID	*Universally Unique Identifier*
UX	*User Experience*
XP	Extreme Programming

Sumário

1 Introdução

O objetivo deste capítulo é apresentar uma visão geral do trabalho, abordando a contextualização do tema e explorando a relevância do setor de delivery e gestão de restaurantes, além dos desafios enfrentados pelos estabelecimentos. Também são definidos a questão de pesquisa, que orienta o desenvolvimento do projeto, e os objetivos geral e específicos, que estabelecem as metas a serem alcançadas com a aplicação proposta.

A justificativa destaca a importância do estudo e os benefícios esperados com a implementação da solução tecnológica. Por fim, são descritas a metodologia adotada e a organização do trabalho, detalhando a estrutura dos capítulos.

1.1 Contextualização

O setor de delivery e gestão de restaurantes desempenha um papel estratégico na economia brasileira, evidenciando um crescimento acelerado e a constante digitalização de suas operações. A pandemia de COVID-19 intensificou a adoção de serviços de entrega, consolidando-os como parte essencial da cadeia de alimentação, tanto para consumidores quanto para estabelecimentos. Dados da Associação Brasileira de Bares e Restaurantes (Abrasel) aponta que o mercado de delivery de comida registrou um crescimento entre 7,5% e 8% em 2023 em relação a 2022 (ABRASEL, 2023b). Além disso, a Fundação Getulio Vargas (FGV) indica que o Brasil responde por cerca de 50% da movimentação de delivery alimentício na América Latina (MASSA, 2022), reforçando a importância desse mercado. Nesse contexto, torna-se evidente a necessidade de ferramentas tecnológicas específicas que auxiliem a gestão interna dos restaurantes, aprimorando a eficiência operacional e a experiência dos gestores.

Historicamente, o gerenciamento de restaurantes no Brasil era realizado de forma manual e descentralizada, com pouca integração de tecnologias. A última década trouxe mudanças significativas, marcadas pela popularização de plataformas de delivery e o aumento da digitalização no setor alimentício. Pesquisa aponta que cerca de 83% dos estabelecimentos já utilizam algum tipo de software para controle de vendas, 75% empregam ferramentas para gestão financeira, e 67% adotam soluções para delivery, evidenciando a crescente adoção de tecnologias no setor (ABRASEL, 2023a). Esse movimento vem sendo acelerado pela crescente demanda por serviços ágeis e convenientes, evidenciando as necessidades de soluções tecnológicas acessíveis para atender às necessidades internas dos restaurantes.

No cenário global, o setor de delivery de alimentos segue em forte expansão. De

acordo com projeções da Statista, a receita global desse mercado deverá alcançar aproximadamente US$ 1,40 trilhão em 2025, com um crescimento anual composto (CAGR) de 7,79% entre 2025 e 2029, resultando em um volume de mercado estimado de US$ 1,89 trilhão até 2029 (STATISTA, 2024). Os principais mercados desse setor continuam sendo China e Estados Unidos, que lideram em volume de pedidos e receita. Em 2025, a China deverá gerar a maior receita global no setor de delivery de alimentos, com um valor estimado de US$ 500,50 bilhões (STATISTA, 2024). No segmento de *Meal Delivery* (entrega de refeições preparadas), a expectativa é que o número de usuários alcance 2,5 bilhões até 2029, com uma penetração de mercado de 28,2% da população global em 2025, evidenciando a crescente adesão dos consumidores a esses serviços (STATISTA, 2024). Esses números demonstram a consolidação do setor de delivery de alimentos em escala global, impulsionada pelo aumento da digitalização do consumo, a ampliação das plataformas de entrega e o fortalecimento da infraestrutura logística nos principais mercados.

Uma tendência global consolidada é a adoção de *dark kitchens* (ou *ghost kitchens*), que são cozinhas profissionais projetadas exclusivamente para delivery, sem atendimento presencial. Esse modelo reduz custos operacionais e aumenta a eficiência logística. Nos Estados Unidos, empresas como CloudKitchens e Reef Technology expandiram rapidamente esse conceito, transformando estacionamentos e espaços ociosos em hubs de preparação de pedidos (CONNECTION, 2024). No Brasil, as *dark kitchens* também ganharam força. Grandes redes de restaurantes e empreendedores independentes têm adotado esse modelo para atender a demanda crescente por entregas, minimizando custos com espaço físico e estrutura tradicional de atendimento.

Nesse contexto, o desenvolvimento de ferramentas tecnológicas robustas surge como uma solução estratégica para enfrentar os desafios do setor de alimentação. Aplicações responsivas, que integrem funcionalidades como controle de pedidos, análise de métricas operacionais e geração de relatórios gerenciais, podem auxiliar nas operações internas dos restaurantes, reduzir custos operacionais e aprimorar a tomada de decisões estratégicas. Além disso, ao priorizar a usabilidade e a interação eficiente entre humanos e computadores, essas ferramentas têm o potencial de agilizar processos organizacionais e aumentar a eficiência das equipes (BARBOSA; ANDRADE, 2021). Tais soluções democratizam o acesso a tecnologias avançadas, atendendo especialmente às necessidades de pequenos e médios estabelecimentos que buscam maior autonomia e competitividade.

A relevância econômica dessas soluções é evidente, pois elas contribuem diretamente para a sustentabilidade financeira dos restaurantes, fortalecendo sua competitividade em um mercado em constante transformação. No âmbito social, essas tecnologias são essenciais para impulsionar pequenos negócios, promovendo maior autonomia e eficiência na gestão das operações. Além disso, ao possibilitar o crescimento dos restaurantes, essas ferramentas geram novas oportunidades de emprego, impactando positivamente a

economia local e o desenvolvimento do setor.

O desenvolvimento dessa aplicação responsiva voltada para o gerenciamento interno de pedidos de delivery em restaurantes busca integrar funcionalidades que não apenas auxiliam as operações administrativas, mas também forneçam dados estratégicos para apoiar gestores em suas decisões, contribuindo para a sustentabilidade e competitividade dos estabelecimentos no setor de alimentação.

1.2 Questão de Pesquisa

Em relação a contextualização apresentada, a questão de pesquisa que orienta esse trabalho é:

Como desenvolver um sistema responsivo que auxilie na gestão interna de pedidos de delivery, apoiando as operações administrativas e a tomada de decisões em restaurantes?

1.3 Objetivos

1.3.1 Objetivo Geral

O objetivo geral deste trabalho é desenvolver uma aplicação responsiva voltada para o gerenciamento interno de delivery em restaurantes, adaptada para *mobile* e *desktop*, com o propósito de apoiar as operações administrativas, aprimorar a experiência dos gestores e fornecer dados analíticos sobre o desempenho da equipe, contribuindo para a organização do fluxo de trabalho e a tomada de decisões estratégicas.

1.3.2 Objetivos Específicos

Para alcançar o objetivo geral, os seguintes objetivos específicos foram definidos:

- Desenvolver uma aplicação responsiva para o gerenciamento interno de pedidos de delivery, adaptada a diferentes dispositivos (*desktop* e *mobile*) e interfaces de uso, garantindo usabilidade e acessibilidade;
- Criar funcionalidades que auxiliam no controle de pedidos e na visualização do desempenho da equipe, com base em dados operacionais;
- Implementar recursos que possibilitem a geração de relatórios gerenciais, fornecendo *insights* estratégicos para apoiar a gestão do restaurante;
- Garantir a segurança das informações sensíveis de usuários, adotando práticas robustas de proteção e conformidade com padrões de segurança de dados;

- Desenvolver mecanismos que assegurem a escalabilidade do sistema, permitindo sua aplicação em diferentes cenários, tamanhos de equipe e volumes de pedidos.

1.4 Justificativa

O presente trabalho justifica-se pela necessidade de desenvolver uma solução tecnológica que atenda diretamente aos gestores de restaurantes, fornecendo ferramentas que aprimorem a eficiência operacional e auxiliem na tomada de decisões estratégicas. Embora o mercado de delivery tenha crescido substancialmente, as soluções atuais são predominantemente voltadas ao consumidor final, deixando lacunas no suporte às operações internas dos estabelecimentos.

A criação de uma aplicação responsiva, que integre funcionalidades como gestão de pedidos, análise de métricas de desempenho e geração de relatórios gerenciais, representa uma oportunidade de inovação. Esse sistema não apenas facilita a administração diária, mas também contribui para a otimização de recursos, a melhoria da experiência dos gestores e a redução de custos operacionais. Para pequenos restaurantes, que frequentemente enfrentam limitações financeiras, a automação de processos pode ser um fator decisivo para alcançar competitividade e sustentabilidade no mercado. Além de reduzir erros operacionais, sistemas automatizados ajudam a aprimorar o atendimento ao cliente, aumentando a fidelidade e a retenção de consumidores.

Além dos benefícios operacionais, a aplicação proposta também tem impacto social e econômico. Ao otimizar a gestão interna, pequenos restaurantes podem se concentrar em entregar um serviço de maior qualidade, gerando melhores experiências para seus clientes e fortalecendo sua posição no mercado local. Isso contribui para a geração de empregos, o fortalecimento de pequenos negócios e o aumento da competitividade no setor de delivery. Por exemplo, sistemas de controle baseados em indicadores, como o *Balanced Scorecard*, permitem monitorar tanto a eficiência operacional quanto a satisfação do cliente, identificando falhas e aprimorando processos de forma estratégica. Além disso, métricas demonstram a relevância de monitorar a qualidade do atendimento, evidenciando como a automação eleva os padrões de serviço e a eficiência geral (FISCHMANN; ZILBER, 2000).

A tecnologia também influencia diretamente os trabalhadores do setor, incluindo entregadores, cozinheiros e atendentes. A automação de processos pode reduzir erros operacionais e melhorar a eficiência da equipe, proporcionando um ambiente de trabalho mais organizado e produtivo. Além disso, ferramentas de planejamento de rotas e notificações automatizadas podem ajudar entregadores a otimizar seu tempo e reduzir atrasos.

Por outro lado, a implementação de tecnologias pode exigir adaptação por parte dos trabalhadores, demandando capacitação para o uso de novos sistemas. É essencial ga-

rantir que a automação não comprometa a valorização do trabalho humano, mas sim atue como um suporte para melhorar as condições de trabalho e garantir maior previsibilidade na execução das tarefas.

Este trabalho fundamenta-se nos princípios de Interação Humano-Computador (IHC) e Experiência do Usuário (UX), investigando a aplicação de tecnologias responsivas em contextos empresariais, como o setor de delivery e gestão de restaurantes. Conforme demonstrado nos estudos de caso apresentados por Albert e Tullis no livro *Measuring the User Experience: Collecting, Analyzing, and Presenting Usability Metrics*, capítulo 10, a implementação de práticas de usabilidade resulta em melhorias significativas na produtividade e na satisfação dos usuários, fatores essenciais para otimizar operações e atender às necessidades dos gestores (ALBERT; TULLIS, 2013).

Por fim, este trabalho aborda uma lacuna no mercado e oferece uma solução prática para um problema real, alinhando-se às demandas contemporâneas por eficiência, inovação tecnológica e experiência de uso aprimorada. Essa abordagem garante relevância tanto acadêmica quanto prática, trazendo benefícios significativos para pequenos restaurantes, promovendo sua modernização e eficiência, e fortalecendo o setor de delivery no Brasil.

1.5 Metodologia

1.5.1 Metodologia de Pesquisa

As pesquisas podem ser classificadas com base em critérios como natureza, forma de abordagem, objetivos gerais e procedimentos técnicos, que orientam a estrutura do estudo e a escolha das técnicas mais adequadas (MARTINS, 2017).

A tabela 1 apresenta as classificações e modalidades de pesquisa que serão utilizadas neste trabalho. No Capítulo 3, cada uma delas será explorada e explicada detalhadamente, destacando sua aplicação e adequação ao objetivo e ao contexto do estudo.

Tabela 1 – Classificação da pesquisa e suas modalidades

Classificação	**Modalidade**
Quanto à natureza	Aplicada
Quanto à forma de abordagem	Qualitativa
Quanto aos objetivos gerais	Exploratória
Quanto aos procedimentos técnicos	Bibliográfica

Fonte: Autores (2024).

1.6 Organização do Trabalho

A estruta do trabalho está organizada da seguinte forma:

- **Capítulo 1 – Introdução**: Apresenta uma visão geral do projeto, incluindo o contexto em que está inserido, a questão de pesquisa, os objetivos, a justificativa e as metodologias adotadas.
- **Capítulo 2 – Referencial Teórico**: Reúne as bases teóricas necessárias para compreender o tema tratado, abrangendo conceitos relevantes e a análise de recursos tecnológicos que possam contribuir para a proposta
- **Capítulo 3 – Metodologia**: Expõe os métodos utilizados na pesquisa e no desenvolvimento do trabalho, detalhando as atividades planejadas, os requisitos estabelecidos, as tecnologias escolhidas, a arquitetura do sistema e os recursos empregados.
- **Capítulo 4 – Desenvolvimento**: Descreve o processo de desenvolvimento do sistema, incluindo a execução das estratégias apresentadas no Capítulo 3, abordando as etapas de planejamento, implementação, testes, integração tecnológica, validação com base nos requisitos estabelecidos anteriormente, as telas do sistema, os ajustes feitos com base em *feedbacks* de usuários e as medidas de segurança implementadas para proteger os dados.
- **Capítulo 5 – Considerações Finais**: Conclui o Trabalho de Conclusão de Curso, apresentando as considerações finais, as dificuldades encontradas, as limitações do estudo e sugestões para trabalhos futuros.

2 Referencial Teórico

Neste capítulo, serão apresentados os princípios fundamentais que embasam este trabalho acadêmico, incluindo conceitos, teorias e estudos relevantes relacionados ao tema em questão. Esses fundamentos forneceram os alicerces necessários para as discussões e análises que ocorreram ao longo do estudo, garantindo uma base sólida para as conclusões propostas. As seções abordam tópicos essenciais, como design responsivo, conceitos de interação humano-computador (IHC) e experiência do usuário (UX), além de explorar o cenário do mercado de delivery no Brasil.

2.1 Design Responsivo

O desenvolvimento responsivo busca garantir que uma aplicação se adapte automaticamente a diferentes tamanhos de tela e dispositivos, proporcionando uma experiência fluida e intuitiva. Com a diversidade de plataformas, desde *smartphones* até monitores de alta definição, é essencial que a interface consiga se ajustar dinamicamente ao ambiente de uso (FLUTTER, 2025). Em ambientes como restaurantes, essa adaptabilidade se torna crucial, uma vez que a utilização de dispositivos móveis, como tablets, oferecem um custo-benefício atrativo, pois podem substituir a infraestrutura de computadores fixo e permite que sejam utilizados de forma prática em ambientes dinâmicos e com espaços reduzidos, otimizando o fluxo operacional e a mobilidade dos colaboradores. Além disso, a familiaridade dos funcionários com tecnologias *touchscreen*, adquirida pelo uso cotidiano de *smartphones*, contribui para uma adoção mais rápida e eficiente.

2.1.1 Abordagens do Design Responsivo

Dentro desse contexto, existem duas abordagens principais: design responsivo e design adaptativo.

2.1.1.1 Design Responsivo

O design responsivo reorganiza os elementos da interface para ocupar o espaço disponível de maneira proporcional, garantindo que a aplicação funcione bem independentemente da tela (FLUTTER, 2025).

2.1.1.2 Design Adaptativo

O design adaptativo vai além da simples reorganização, ele ajusta a estrutura da interface conforme o dispositivo, otimizando a navegação e os métodos de entrada

(FLUTTER, 2025).

2.1.2 Vantagens do Design Responsivo

A adoção de um design responsivo traz diversas vantagens, como:

- **Maior acessibilidade**: Usuários podem acessar a aplicação em diferentes dispositivos sem perda de funcionalidade.
- **Melhor experiência do usuário**: Interfaces intuitivas e bem adaptadas melhoram a usabilidade e a satisfação do usuário.
- **Redução da necessidade de múltiplas versões**: Evita a criação de versões separadas para cada plataforma, economizando tempo e recursos.

2.1.3 Ferramentas para Implementação

Ferramentas modernas permitem a implementação eficiente do design responsivo, garantindo que as aplicações ofereçam uma experiência consistente em qualquer dispositivo. Este projeto utilizará o Flutter 3.5.4.2, o qual permite a implementação de um design responsivo, garantindo compatibilidade entre diferentes dispositivos e oferecendo uma experiência fluida para os usuários.

2.2 Conceitos Fundamentais

2.2.1 Interação Humano-Computador (IHC)

A Interação Humano-Computador (IHC) é uma área focada no design, implementação e avaliação de sistemas computacionais interativos para uso humano, considerando também os fenômenos associados a essa interação. Seus principais objetos de estudo incluem a natureza da interação, características humanas, contexto de uso, sistemas computacionais e os processos de desenvolvimento de interfaces (HEWETT et al., 1992).

Os sistemas interativos devem equilibrar interesses distintos de desenvolvedores, que priorizam robustez e funcionalidade, e usuários, que valorizam eficiência e facilidade de uso. Uma abordagem de "fora para dentro", que considera o contexto e as necessidades dos usuários, é essencial para desenvolver soluções mais adequadas (BARBOSA et al., 2021).

2.2.2 Experiência do Usuário (UX)

A Experiência do Usuário (UX) abrange percepções e respostas dos usuários antes, durante e após o uso de um sistema, produto ou serviço. A ISO 9241-210-2019 define UX

como um conjunto de fatores que inclui emoções, crenças, conforto e realizações dos usuários (STANDARDIZATION, 2019). A UX vai além de aspectos funcionais, englobando elementos hedônicos, como prazer e estética, para proporcionar experiências positivas (BLYTHE et al., 2004).

Embora relacionadas, IHC e UX possuem diferenças sutis. Enquanto a IHC se concentra na realização de metas em contextos de trabalho, a UX expande esse escopo, considerando também aspectos emocionais e contextuais. Ambas, no entanto, se complementam na busca por qualidade de uso e impacto positivo na vida dos usuários (BARBOSA et al., 2021).

2.2.3 Benefícios da IHC e da UX

Investir na qualidade de uso em sistemas interativos gera benefícios como aumento de produtividade, redução de erros, menor custo de treinamento e maior fidelidade do cliente. Sistemas bem projetados promovem uma interação intuitiva, reduzem barreiras e elevam a percepção de valor do produto e da empresa (BARBOSA et al., 2021).

2.2.4 Qualidade em IHC

A qualidade em sistemas interativos depende de critérios como usabilidade, acessibilidade e comunicabilidade (BARBOSA et al., 2021).

2.2.4.1 Usabilidade

A usabilidade é o grau em que um sistema permite que usuários atinjam objetivos com eficácia, eficiência e satisfação. Segundo (NIELSEN, 1994), fatores fundamentais incluem:

- **Facilidade de aprendizado:** rapidez para aprender a usar o sistema.
- **Facilidade de recordação:** facilidade para lembrar como utilizá-lo.
- **Eficiência:** rapidez na realização de tarefas.
- **Segurança no uso:** prevenção e recuperação de erros.
- **Satisfação:** avaliação subjetiva do uso.

2.3 Cenário do Delivery no Brasil

O mercado de delivery no Brasil tem crescido significativamente nos últimos anos, impulsionado por mudanças nos hábitos de consumo e pelo aumento da digitalização nos

serviços de alimentação. De acordo com dados da Associação Brasileira de Bares e Restaurantes (Abrasel), o mercado de delivery de comida registrou um crescimento entre 7,5% e 8% em 2023 em relação a 2022 (ABRASEL, 2023b). Além disso, o número de pedidos aumentou 10%, com uma média de dois pedidos por mês por pessoa, evidenciando uma mudança significativa nos padrões de consumo (ABRASEL, 2023b). Apesar desse avanço, muitos pequenos e médios estabelecimentos enfrentam dificuldades em adotar tecnologias que otimizem suas operações. Um estudo realizado pela Agência Brasileira de Desenvolvimento Industrial (ABDI) em conjunto com a Fundação Getúlio Vargas (FGV) revelou que 66% das micro e pequenas empresas brasileiras ainda estão na fase inicial do processo de transformação digital, indicando desafios na adoção de tecnologias otimizadoras de operações (SEBRAE, 2023). A eficiência operacional depende de processos bem estruturados e automatizados, uma lacuna crítica para negócios que ainda dependem de sistemas manuais (SLACK; BRANDON-JONES; JOHNSTON, 2010).

Embora o setor de alimentação tenha se beneficiado amplamente de plataformas digitais que conectam consumidores a restaurantes, muitos estabelecimentos ainda enfrentam a necessidade de ferramentas tecnológicas específicas para atender às suas demandas operacionais internas. Segundo o Serviço Brasileiro de Apoio às Micro e Pequenas Empresas (SEBRAE), os empresários reconhecem a importância de agir, estando abertos a transformar seus negócios e expandir mercados por meio de novos formatos. Essa transformação envolve o uso de ferramentas e tecnologias digitais para converter operações tradicionais em empresas digitais, com uma gestão orientada a dados, maior rapidez na inovação, colaboração eficiente e geração de mais valor para os clientes (SEBRAE, 2023). Nesse contexto, o desenvolvimento de soluções tecnológicas acessíveis, que considerem as particularidades e o cenário específico de cada negócio, é essencial para fortalecer a competitividade e garantir a sustentabilidade desses estabelecimentos no mercado.

2.4 Gestão da Qualidade Total no Setor de Delivery

De acordo com W. Edwards Deming e Joseph Juran, o conceito de Gestão da Qualidade Total (TQM) destaca a importância de envolver todos os níveis da organização no compromisso com a qualidade (LAUDON; LAUDON, 2007). No contexto de delivery, a TQM pode ser aplicada para monitorar tanto a produtividade quanto a qualidade das equipes.

A ausência de mecanismos para integrar essas informações limita a capacidade de análise do desempenho individual, dificultando intervenções como treinamentos específicos ou mudanças estratégicas na equipe. Essa lacuna operacional representa uma oportunidade para o desenvolvimento de ferramentas mais completas e focadas em qualidade.

2.5 Reengenharia de Processos para Delivery

A reengenharia de processos busca repensar fluxos de trabalho para eliminar redundâncias e otimizar resultados. Uma abordagem de reengenharia bem-sucedida requer a análise detalhada dos processos existentes e a implementação de mudanças tecnológicas que tragam eficiência e flexibilidade (LAUDON; LAUDON, 2007). No setor de delivery, isso pode significar automatizar processos como:

- Atribuição de pedidos a funcionários;
- Coleta e análise de desempenho do restaurante;
- Geração de relatórios integrados para identificar gargalos e oportunidades de melhoria.

Ao simplificar as operações e centralizar informações críticas, é possível reduzir custos operacionais e aumentar a satisfação do cliente.

2.6 Análise de Dados e Tomada de Decisão Estratégica

No cenário atual, a análise de dados é um recurso indispensável para decisões estratégicas. As organizações que utilizam dados para embasar suas ações têm maior capacidade de se adaptar às demandas do mercado (WISER, 2024).

Para restaurantes, integrar a análise de dados às operações de delivery pode gerar *insights* valiosos, como:

- Identificação de padrões de clientes;
- Monitoramento de desempenho da equipe;
- Ajustes em processos para melhorar eficiência e reduzir erros.

Esses dados também podem ser utilizados para criar modelos preditivos, ajudando a antecipar variações na demanda e planejar recursos com maior precisão.

2.7 Usabilidade e Design Responsivo em Sistemas Gerenciais

Sistemas gerenciais eficientes devem ser acessíveis e intuitivos, especialmente para pequenos negócios com equipes heterogêneas em termos de familiaridade com tecnologia. Nielsen destaca que interfaces bem projetadas promovem uma interação mais fluida, reduzindo a curva de aprendizado (NIELSEN, 1994).

No contexto de delivery, o uso de design responsivo é essencial para permitir que gestores e equipes acessem ferramentas de gerenciamento em qualquer dispositivo, seja *mobile* ou *desktop*. Isso facilita o acompanhamento em tempo real das operações e a tomada de decisões rápidas, mesmo fora do local de trabalho.

2.8 Benefícios de Soluções Integradas no Delivery

A adoção de soluções integradas pode resolver muitos dos desafios enfrentados por pequenos e médios restaurantes. Entre os principais benefícios estão:

- **Automatização**: Redução de processos manuais e erros operacionais;
- **Centralização de Dados**: Coleta e integração de informações sobre pedidos e desempenho da equipe;
- **Flexibilidade**: Adaptação a diferentes cenários e volumes de pedidos;
- **Melhoria da Qualidade**: Monitoramento do desempenho dos colaboradores.

Essas soluções permitem maior controle gerencial e criam um diferencial competitivo para negócios que enfrentam concorrência de grandes plataformas.

2.9 Conclusão e Conexão com o Problema de Pesquisa

Com base nos conceitos apresentados, observa-se que a falta de integração entre ferramentas de análise de desempenho e operações internas limita o crescimento de pequenos e médios restaurantes. Soluções que combinam automação, análise de dados e usabilidade são essenciais para superar essas barreiras e promover maior eficiência operacional.

Além disso, sistemas que incorporam princípios de TQM e reengenharia de processos oferecem uma oportunidade única para melhorar a experiência do cliente e a gestão interna, criando uma base sólida para decisões estratégicas. Essas abordagens formam o alicerce teórico para o desenvolvimento de uma aplicação voltada para o gerenciamento interno de delivery.

2.10 Benchmarking de Funcionalidades

Para embasar o desenvolvimento do sistema proposto, foi realizada uma análise comparativa com duas das principais soluções utilizadas no setor: o **Consumer**, sistema de gestão e ponto de venda, e o **iFood**, plataforma de marketplace amplamente consolidada no mercado brasileiro. Destaca-se que o sistema proposto apresenta diferenciais

relevantes em comparação às demais soluções analisadas, especialmente pelos indicadores fornecidos por meio dos relatórios gerenciais voltados aos estabelecimentos, além do seu design responsivo, que garante acesso eficiente tanto em *desktop* quanto em dispositivos móveis.

A tabela 2 abaixo resume o resultado do benchmarking, com base em funcionalidades relevantes ao contexto do projeto. A análise foi construída a partir de documentação técnica, materiais institucionais e experimentação direta dos sistemas, sempre que possível. Os links que comprovam ou ilustram essas funcionalidades estão organizados no apêndice A.

Tabela 2 – Benchmarking de funcionalidades: Sistema Proposto, Consumer e iFood

Funcionalidade	**Sistema Proposto**	**Consumer**	**iFood**
Agrupamento automático inteligente de pedidos por proximidade	✓	×	×
Notificações em tempo real com status detalhado do pedido	✓	✓	✓
Registro completo de clientes (dados detalhados e históricos)	✓	✓	×
Preenchimento automático de taxas de entrega por zonas	✓	✓	✓
Geração de relatórios de toda equipe envolvida no delivery	✓	×	×
Geração de relatórios de pedidos	✓	✓	✓
Geração de relatórios de vendas	✓	✓	✓
Assistente IA para responder perguntas usando dados do restaurante	✓	×	×
Design responsivo (uso eficiente tanto em *desktop* quanto em celulares)	✓	×	×

Fonte: Autores (2025).

Legenda:
✓– Funcionalidade disponível de forma completa
×– Funcionalidade não disponível

3 Metodologia

Nesta seção, são descritas as metodologias adotadas para o desenvolvimento deste trabalho, divididas em metodologia de pesquisa e metodologia de desenvolvimento de software. Essas abordagens foram escolhidas para atender tanto à necessidade de embasamento teórico quanto à criação e validação do aplicativo proposto. A seguir, detalham-se as etapas e técnicas empregadas.

3.1 Metodologia de Pesquisa

Conforme destacado na seção 1.5.1 apresentado na tabela 1, a metodologia deste trabalho segue uma classificação sistemática voltada para a aplicação prática da pesquisa, utilizando uma abordagem qualitativa para compreender as necessidades dos usuários. Com objetivos exploratórios, busca-se identificar lacunas no setor de delivery e fundamentar teoricamente o desenvolvimento de um sistema responsivo. Essa metodologia conecta a fundamentação teórica à criação prática, garantindo que o projeto seja conduzido de forma estruturada e alinhada às demandas reais do mercado.

3.1.1 Quanto a Natureza

A pesquisa realizada é classificada como aplicada, pois busca criar uma solução prática para o problema real da gestão interna de processos de delivery em restaurantes. Diferentemente da pesquisa pura, o foco aqui é a geração de conhecimento diretamente aplicável (MARTINS, 2017), auxiliando nos processos operacionais e promover maior eficiência e autonomia para pequenos e médios estabelecimentos. Ao integrar teorias e práticas específicas do setor, a abordagem aplicada auxilia no alinhamento do sistema desenvolvido às demandas reais do mercado e dos usuários.

3.1.2 Quanto à forma de abordagem

A pesquisa deste trabalho adota uma abordagem qualitativa, caracterizada pela exploração de particularidades e traços subjetivos relacionados às percepções e opiniões dos participantes sobre a realidade estudada. Esse tipo de abordagem considera os significados, motivos, aspirações e valores expressos pelos entrevistados em relação ao problema investigado (Minayo, 2001, apud Martins, 2017, (MARTINS, 2017), p. 22). No contexto deste estudo, o foco qualitativo permite compreender de forma mais profunda as necessidades dos gestores de restaurantes no gerenciamento interno de pedidos de delivery, proporcionando uma análise detalhada de suas perspectivas e vivências.

A pesquisa qualitativa também assume um caráter descritivo e exploratório, permitindo que os entrevistados expressem suas opiniões livremente, o que possibilita identificar demandas específicas e alinhar o desenvolvimento do sistema às necessidades reais do setor (MARTINS, 2017). Além disso, o pesquisador assume um papel ativo no processo, analisando os dados coletados e relacionando-os com o objetivo de criar uma solução prática e efetiva para os desafios enfrentados pelos restaurantes.

3.1.3 Quanto aos objetivos gerais

A pesquisa exploratória é utilizada quando o problema é pouco conhecido e as hipóteses ainda não estão bem definidas, exigindo um maior envolvimento do pesquisador para delineá-lo e compreendê-lo melhor (MARTINS, 2017). No contexto deste trabalho, que visa desenvolver uma aplicação para gestão de processos de delivery em restaurantes, essa abordagem permitirá identificar áreas de oportunidade e aspectos específicos a serem trabalhados. Utilizando métodos como pesquisa bibliográfica e entrevistas com gestores, o estudo orienta o desenvolvimento da solução proposta de forma mais eficaz e alinhada às reais necessidades do setor.

3.1.4 Quanto aos procedimentos técnicos

A pesquisa bibliográfica permite acessar contribuições científicas que contextualizam e fundamentam o tema abordado. Segundo (Martins, 2000, apud Martins, 2017, (MARTINS, 2017), p. 25), esse tipo de pesquisa envolve a leitura, análise e interpretação de materiais publicados, como livros, artigos acadêmicos, periódicos e fontes eletrônicas, fornecendo uma base teórica sólida para o estudo. Nesse contexto, a pesquisa bibliográfica será utilizada para embasar o desenvolvimento da aplicação proposta, oferecendo suporte conceitual e contribuindo para a compreensão do cenário em que o problema está inserido, além de orientar a elaboração de estratégias para as soluções desenvolvidas.

3.2 Metodologia de Desenvolvimento

O desenvolvimento do sistema proposto segue uma abordagem híbrida, combinando técnicas de diferentes metodologias ágeis para aproveitar seus pontos fortes e atender às demandas específicas do projeto. Essa combinação inclui práticas do Scrum, eXtreme Programming (XP), Kanban e princípios do DevOps, proporcionando maior flexibilidade, organização e qualidade ao longo do processo.

A integração dessas metodologias foi projetada para garantir um fluxo de trabalho eficiente, com foco na colaboração da equipe, entrega contínua de valor e melhoria constante do sistem. Na seção 3.2.6 são detalhadas as práticas adotadas de cada metodologia ágil e como elas contribuem para o desenvolvimento do projeto.

3.2.1 Metodologias Ágeis

As metodologias ágeis foram desenvolvidas na década de 1990 como uma alternativa aos métodos tradicionais de desenvolvimento de software, que frequentemente apresentavam elevados custos e burocracia associados a processos altamente planejados, especialmente em projetos de grande escala. Consideradas métodos leves, as abordagens ágeis se destacam por seus ciclos de desenvolvimento curtos e iterativos, equipes auto-organizadas, designs simplificados, refatoração contínua do código, desenvolvimento orientado a testes e envolvimento frequente dos clientes. Essas características permitem que cada ciclo de desenvolvimento resulte em um produto funcional e demonstrável, alinhado às expectativas do cliente (IEEE, 2024). A seguir, são apresentados os principais métodos e práticas que embasam o desenvolvimento, destacando como cada um contribui para atender às necessidades do projeto e garantir sua execução com sucesso.

3.2.2 Scrum

O Scrum é uma abordagem ágil para o gerenciamento de projetos que organiza o trabalho em ciclos chamados *sprints*, com duração máxima de 30 dias, embora frequentemente durem duas semanas. Durante os *sprints*, a equipe trabalha em tarefas priorizadas a partir do *backlog* do produto, criado e mantido pelo proprietário do produto. Cada *sprint* resulta em um incremento funcional do software, garantindo adaptações rápidas às mudanças (IEEE, 2024).

O processo Scrum é estruturado em eventos, conhecidos como cerimônias Scrum, alguns eventos incluem:

- **Criação do Product Backlog**: lista de todas as funcionalidades, requisitos, melhorias e correções que o produto necessita, priorizada pelo proprietário do produto de acordo com o valor de negócio e os riscos envolvidos.
- **Planejamento de Sprint**: A equipe define metas claras e reúne os itens selecionados do *product backlog* que serão desenvolvidos durante a *sprint*, incluindo as tarefas necessárias para sua realização.
- **Scrum Diário**: Reuniões curtas realizadas diariamente, onde os membros alinham as atividades, relatam avanços e identificam desafios, mantendo o foco nas metas do *sprint*.
- **Revisão de Sprint**: Ao final do *sprint*, a equipe apresenta o trabalho concluído às partes interessadas e ajusta o *backlog* do produto conforme necessário.
- **Retrospectiva do Sprint**: A equipe reflete sobre o que funcionou bem e o que pode ser melhorado, gerando *insights* para aprimorar os próximos *sprints*.

Com essa estrutura clara e iterativa, o Scrum permite o desenvolvimento eficiente e colaborativo, entregando valor contínuo ao projeto (AWS, 2025).

3.2.3 Extreme Programming

A abordagem Extreme Programming (XP) é centrada em histórias ou cenários que detalham os requisitos do cliente. Antes do desenvolvimento, testes são criados para verificar os requisitos (IEEE, 2024). Algumas das práticas utlizadas no XP são:

- **Programação em Par**: Dois desenvolvedores trabalham juntos em uma mesma tarefa, alternando entre o papel de escritor e revisor, o que melhora a qualidade do código e facilita o compartilhamento de conhecimento.
- **Refatoração Contínua**: O código é constantemente revisado e melhorado para reduzir a complexidade, manter a clareza e evitar dívidas técnicas.
- **Integração Contínua**: Mudanças no código são integradas e testadas frequentemente, minimizando riscos de integração e garantindo a estabilidade do sistema.
- **Pequenos Ciclos de *Feedback***: Iterações rápidas permitem ajustes frequentes e entregas incrementais, garantindo que o cliente possa avaliar e sugerir melhorias continuamente.
- **Simplicidade no Design**: O XP prioriza a implementação de soluções que atendam aos requisitos atuais, evitando a adição de funcionalidades desnecessárias.

A abordagem XP permite que equipes se adaptem rapidamente e façam ajustes conforme necessário, assim garante que o produto final atenda às necessidades do projeto (CTC, 2024).

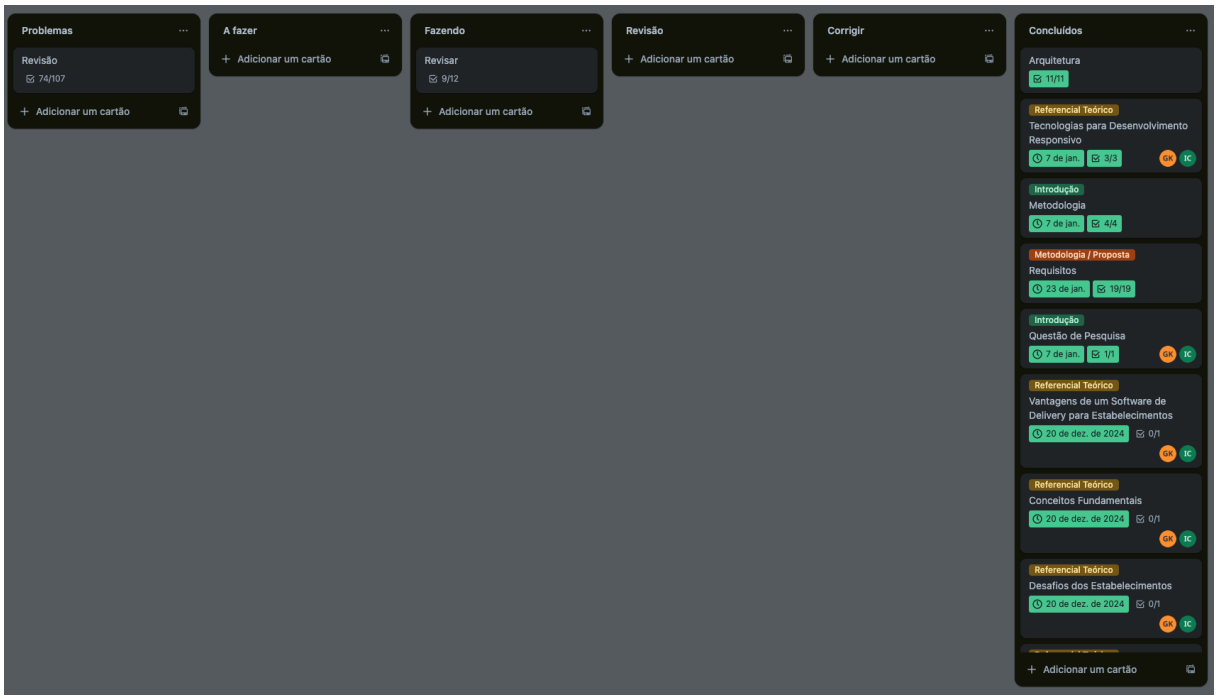
3.2.4 Kanban

O Kanban é um método de gestão de trabalho que visa aprimorar a eficiência e a produtividade das equipes por meio da visualização do fluxo de tarefas e da limitação do trabalho em andamento. Originado no Sistema Toyota de Produção na década de 1950, o termo japonês “kanban” significa “sinalização” ou “cartão”, referindo-se aos cartões utilizados para indicar o andamento das atividades.

Sua implementação geralmente envolve o uso de um quadro dividido em colunas que representam as etapas do processo de trabalho, como “A Fazer”, “Em Progresso” e “Concluído”. As tarefas são representadas por cartões que se movem entre as colunas conforme avançam, proporcionando uma visualização clara do status de cada atividade e do fluxo de trabalho como um todo (SABINO, 2023).

Na figura 1 é apresentado o quadro Kanban utilizado para o desenvolvimento do TCC, onde as tarefas são organizadas em colunas de acordo com seu status.

Figura 1 – Quadro Kanban.



Fonte: Autores (2025).

3.2.5 DevOps

O DevOps é uma abordagem moderna que busca integrar as equipes de desenvolvimento de software (Dev) e operações de TI (Ops), promovendo a automação, a colaboração contínua e a entrega ágil de software. As operações de TI envolvem a gestão da infraestrutura tecnológica, incluindo servidores, redes, segurança, monitoramento e automação de processos, garantindo a estabilidade e escalabilidade dos sistemas. Segundo *The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations*, a filosofia DevOps fundamenta-se nos Três Caminhos:

- **Primeiro Caminho**: Foca no fluxo contínuo de trabalho do desenvolvimento até a produção, reduzindo atrasos e aumentando a eficiência.
- **Segundo Caminho**: Priorização do feedback rápido, permitindo identificar e corrigir problemas antes que afetem os usuários finais.
- **Terceiro Caminho**: Incentiva o aprendizado contínuo e a experimentação para aprimoramento constante dos processos.

A aplicação desses princípios no desenvolvimento deste projeto será essencial para garantir agilidade, confiabilidade e eficiência, utilizando práticas como Integração Contínua (CI) para automatizar o fluxo de trabalho (KIM et al., 2016).

3.2.5.1 Integração Contínua

A Integração Contínua (*Continuous Integration* - CI) é uma prática essencial dentro do desenvolvimento moderno de software, permitindo maior eficiência, qualidade e confiabilidade no ciclo de desenvolvimento. A CI refere-se ao processo de automação na integração de código de múltiplos desenvolvedores em um repositório compartilhado, onde ele é automaticamente testado para garantir sua funcionalidade. (REDHAT, 2024).

A implementação de CI traz benefícios como a redução de erros humanos, a melhoria da qualidade do código e a aceleração do ciclo de desenvolvimento. Isso é possível por meio da automação de testes, garantindo que cada alteração de código seja testada e entregue com rapidez e segurança (UNITY, 2024).

3.2.5.2 GitHub Actions

O GitHub Actions é uma ferramenta oferecida pelo GitHub 3.8 que permite automatizar tarefas dentro do desenvolvimento de software, como testar código automaticamente e gerar versões do software (GITHUB, 2024). Com isso, desenvolvedores podem configurar fluxos de trabalho (*workflows*) que executam ações de forma automática sempre que algo acontece no repositório, como um novo código sendo enviado (*push*) ou um pedido de revisão (*pull request*).

No contexto deste projeto, o GitHub Actions será utilizado para implementar um processo de CI. Isso significa que, sempre que um desenvolvedor fizer uma alteração no código e enviá-lo para o repositório, o GitHub Actions irá automaticamente verificar se tudo está funcionando corretamente.

3.2.5.3 Docker

O Docker é uma plataforma aberta que facilita o desenvolvimento, envio e execução de aplicações de forma ágil e consistente. Ele utiliza containers para isolar aplicações e suas dependências, garantindo que funcionem de maneira previsível em qualquer ambiente. Ao compartilhar o kernel do sistema operacional do host, os containers são leves e eficientes, reduzindo significativamente o tempo entre a escrita do código e sua execução em produção. Com ferramentas como o Docker Compose, é possível orquestrar aplicações multicontainers, simplificando a gestão de serviços complexos. Essa abordagem permite gerenciar a infraestrutura como as aplicações, promovendo rapidez, consistência e eficiência no ciclo de vida do software (DOCKER, 2025).

3.2.6 Práticas Utilizadas

A implementação prática do projeto baseou-se em um conjunto de práticas selecionadas e adaptadas das metodologias ágeis para atender às necessidades específicas do projeto. Essa adaptação será fundamental para aumentar a eficiência do trabalho em equipe, melhorar a qualidade do código e entregar valor de forma contínua e incremental. As práticas adotadas incluem:

- **Criação do backlog** (Scrum, XP): Listar os requisitos e organizar por funcionalidades e épicos a serem desenvolvidos.
- **Priorização do backlog** (Scrum): Organização das tarefas de acordo com a relevância e impacto para o sistema.
- **Kanban** (Kanban): Um quadro visual será utilizado para gerenciar o fluxo de trabalho, permitindo que a equipe acompanhe o andamento das tarefas em tempo real e identifique rapidamente possíveis gargalos.
- **Planejamento de Sprint** (Scrum): Reunião da equipe para definir metas claras para a *sprint* e selecionar os itens do *backlog* a serem abordados, considerando a capacidade e o comprometimento da equipe.
- **Scrum Diário** (Scrum): Reuniões diárias para alinhamento de atividades e comunicação constante entre os membros.
- **Simplicidade no Design** (XP): Desenvolver soluções com foco na simplicidade, buscando atender aos requisitos de forma direta e eficaz, evitando complexidades desnecessárias que podem dificultar futuras manutenções.
- **Programação em Par** (XP): Sessões de programação em duplas para revisão de código e compartilhamento de conhecimento imediato.
- **Pequenos Ciclos de *Feedback*** (XP, Scrum): Entregas incrementais e frequentes para avaliação contínua.
- **Refatoração Contínua** (XP): Melhorias constantes do código para garantir sua qualidade e eficiência.
- **Testes Contínuos** (XP, DevOps): A cada nova funcionalidade desenvolvida, serão realizados testes para verificar sua funcionalidade e integração.
- **Integração Contínua** (DevOps, XP): Integração frequente do código para garantir sua estabilidade.

- **Revisão de Sprint** (Scrum): Apresentação do trabalho concluído ao final de cada *sprint* para coletar *feedbacks* que ajudarão a ajustar o planejamento das próximas iterações.

3.3 Requisitos

A definição de requisitos é fundamental para o sucesso de qualquer projeto de software. Contudo, o termo "requisito" pode ser interpretado de diferentes maneiras por desenvolvedores e especialistas em tecnologia. Uma interpretação é que requisitos são "uma especificação do que deve ser implementado, descrições de como o sistema deve se comportar ou de uma propriedade ou atributo do sistema" (Ian Sommerville e Pete Sawyer, 1997, apud K. Wiegers and J. Beatty, 2013, (WIEGERS; BEATTY, 2013), p. 39). Além disso, requisitos podem incluir restrições no processo de desenvolvimento, destacando tanto o comportamento externo esperado pelo usuário quanto características internas necessárias para a implementação.

Requisitos abrangem diferentes dimensões, como necessidades atuais, prioridades futuras e até mesmo aspectos descartados ao longo do projeto. Essa diversidade reflete o papel central dos requisitos como direcionadores das escolhas de design, garantindo que as soluções desenvolvidas atendam aos objetivos e às condições do ambiente de uso (WIEGERS; BEATTY, 2013).

3.3.1 Levantamento de Requisitos

O levantamento de requisitos é uma etapa crucial no desenvolvimento de software, pois é nesse momento que são coletadas as informações necessárias para definir o que o sistema deve realizar. Essa atividade busca compreender as necessidades e expectativas dos clientes, transformando-as em especificações claras e viáveis para orientar o desenvolvimento do projeto (WIEGERS; BEATTY, 2013).

Para garantir a abrangência e precisão, serão utilizadas diferentes técnicas que possibilitaram explorar as necessidades de forma ampla e identificar os requisitos do sistema. Essas técnicas combinadas possibilitaram uma coleta de informações abrangente e alinhada às necessidades dos usuários, estabelecendo uma base sólida para o desenvolvimento do sistema e garantindo que o projeto atenda às expectativas e objetivos definidos.

3.3.1.1 Observação

A técnica de observação é uma ferramenta poderosa para compreender como os usuários executam suas tarefas no ambiente de trabalho, capturando detalhes muitas vezes implícitos ou não verbalizados devido à complexidade ou ao caráter rotineiro das atividades (WIEGERS; BEATTY, 2013). No contexto deste projeto, a observação foi conduzida

diretamente em um restaurante self-service com serviço de delivery, acompanhando as operações diárias para identificar gargalos operacionais, práticas ineficientes e possíveis melhorias nos processos.

A observação também serviu de base para validar os dados obtidos nas entrevistas e fundamentar a reengenharia de processos, descrita na Seção 2.5, destacando oportunidades para eliminar redundâncias e otimizar fluxos de trabalho. A seguir, detalham-se os objetivos, abordagem e análise da observação realizada.

3.3.1.1.1 Objetivos da Observação

O principal objetivo da observação foi entender os desafios práticos e operacionais enfrentados na gestão de um restaurante com foco em delivery. Especificamente, buscou-se:

- Identificar problemas nos fluxos de trabalho relacionados ao recebimento, preparo e entrega de pedidos.
- Reconhecer tarefas que poderiam ser otimizadas ou automatizadas.
- Analisar como a equipe interage com os sistemas existentes e identificar limitações.

3.3.1.1.2 Metodologia da Observação

A observação foi realizada no ambiente do restaurante, com o propósito de compreender os fluxos de trabalho e identificar desafios operacionais de maneira detalhada. Adotou-se uma abordagem estruturada, direcionada aos objetivos estabelecidos, priorizando a análise de práticas rotineiras e a identificação de etapas críticas, suscetíveis a erros ou com potencial para automação. As informações coletadas desempenharam um papel fundamental no levantamento de requisitos, oferecendo uma base sólida para o desenvolvimento de melhorias alinhadas às necessidades reais do sistema.

3.3.1.1.3 Resultados da Observação

A análise das observações permitiu identificar fluxos críticos e levantar requisitos funcionais e não funcionais para o sistema proposto. Esses requisitos foram organizados no apêndice B utilizando uma estrutura hierárquica composta por épicos, funcionalidades e requisitos. Essa abordagem é amplamente utilizada em metodologias ágeis, como Scrum, para proporcionar clareza e rastreabilidade no desenvolvimento. Os épicos representam grandes objetivos do sistema, as funcionalidades detalham os componentes principais que atendem a esses objetivos, e os requisitos especificam as implementações detalhadas necessárias para cada funcionalidade (COHN, 2004).

3.3.1.2 Entrevistas

Essa técnica, destacada no livro *Software Requirements (Third Edition)*, consiste em coletar informações diretamente de partes interessadas relevantes. Por meio de perguntas direcionadas, busca-se identificar necessidades específicas, expectativas e possíveis problemas enfrentados (WIEGERS; BEATTY, 2013).

Para garantir a eficácia do processo, as entrevistas foram planejadas, com base nos livros *Software Requirements (Third Edition)* e Engenharia de Requisitos: Software Orientado ao Negócio. O roteiro foi estruturado em seções principais, incluindo apresentação, objetivos da entrevista, e um caráter informal, buscando criar um ambiente confortável para o entrevistado. Além disso, perguntas direcionadoras foram elaboradas para manter o foco nos temas desejados, mas com flexibilidade suficiente para que os entrevistados compartilhassem espontaneamente suas experiências e opiniões.

3.3.1.2.1 Objetivos da entrevista

O objetivo desta entrevista é identificar os principais problemas e desafios nos processos de gerenciamento interno de pedidos de delivery em restaurantes, explorando dificuldades operacionais, lacunas nas ferramentas existentes e oportunidades de melhoria. Busca-se compreender as necessidades e expectativas de gestores e equipes para gerar *insights* que orientem o levantamento de requisitos, visando à otimização dos fluxos de trabalho e ao aumento da eficiência operacional.

3.3.1.2.2 Técnicas para Elaboração das Perguntas

A elaboração das perguntas para as entrevistas utilizará técnicas reconhecidas para garantir que as informações relevantes sejam coletadas de forma clara e objetiva. Uma das principais abordagens aplicadas será a técnica 5W + 2H, que organizará as questões em torno de sete dimensões fundamentais: *What* (o quê), *Who* (quem), *When* (quando), *Where* (onde), *Why* (por quê), *How* (como) e *How Much* (quanto). Essa técnica permitirá uma visão abrangente e estruturada do problema, abordando aspectos como o que será realizado, quem estará envolvido nos processos, quando e onde as atividades ocorrerão, por que determinadas práticas serão adotadas, como elas serão executadas e quanto esforço ou recursos serão consumidos (VAZQUEZ; SIMõES, 2016).

Além disso, as perguntas serão classificadas em abertas e fechadas para atender a diferentes objetivos durante a entrevista. As perguntas abertas serão utilizadas para capturar opiniões, percepções e experiências detalhadas, proporcionando uma compreensão mais rica e exploratória dos problemas e desafios enfrentados. Por outro lado, as perguntas fechadas serão desenvolvidas para obter respostas objetivas e padronizadas, facilitando a análise comparativa e a extração de dados específicos. Essa combinação de abordagens

garantirá que o levantamento de informações seja completo, coerente e alinhado aos objetivos do projeto.

3.3.1.2.3 Roteiro de Perguntas

O roteiro de perguntas foi elaborado e estruturado para explorar de maneira abrangente os diversos aspectos do gerenciamento de delivery. Ele serve como uma base orientadora para conduzir a entrevista, assegurando que todos os tópicos relevantes sejam abordados de forma eficiente, otimizando o tempo durante sua realização. A seguir, apresenta-se a lista consolidada de perguntas que irá conduzir a entrevista:

- **What (O quê?):**
  - Que tarefas você realiza diariamente no sistema atual? Há algo que gostaria de melhorar?
  - Quais funcionalidades ou ferramentas você sente falta para otimizar e automatizar seu trabalho?
  - Quais informações são essenciais para você no momento de tomar decisões?
- **Why (Por quê?):**
  - Por que os gestores têm dificuldade em acompanhar métricas ou indicadores de desempenho?
  - Por que as ferramentas externas que você usa (se houver) são necessárias para complementar o sistema?
  - Por que você acha que alguns processos são mais demorados ou propensos a erros?
- **Who (Quem?):**
  - Quem são as pessoas envolvidas no processo de pedidos?
  - Quem é o seu melhor funcionário? E por quê?
- **Where (Onde?):**
  - Onde os problemas mais comuns ocorrem no fluxo de pedidos?
  - Onde você costuma acessar o sistema de gerenciamento? (Ex.: *mobile, desktop*, ambos)
  - Onde você acredita que melhorias no sistema poderiam ter maior impacto?
- **When (Quando?):**

  - Quando seria mais útil receber relatórios sobre o desempenho da equipe ou operações?

- **How (Como?):**
  - Como você gerencia os pedidos atualmente? Há passos manuais no processo?
  - Como você verifica o desempenho da equipe ou acompanha os pedidos?
  - Como as tarefas poderiam ser simplificadas ou otimizadas no sistema atual?
  - Como você organiza os dados ou relatórios para tomada de decisão?

- **How Much (Quanto?):**
  - Quantos erros ou problemas relacionados a pedidos acontecem em uma semana, aproximadamente?

3.3.1.2.4 Entrevistados

Duas entrevistas foram realizadas, ambas com profissionais experientes na gestão e operação de restaurantes com foco em delivery. A seguir, apresentamos o perfil detalhado dos entrevistados:

Rafael Almeida:

Rafael Almeida tem 24 anos e é o proprietário de um restaurante self-service familiar. Crescendo ao lado dos pais e acompanhando de perto o dia a dia do negócio, Rafael aprendeu desde cedo a importância de oferecer um serviço de qualidade. Com a chegada da pandemia, ele percebeu que, para manter o restaurante relevante e competitivo principalmente para expandir e otimizar o serviço de delivery era fundamental utilizar tecnologia. Ao se aprofundar na área de delivery, Rafael identificou a carência de sistemas que permitissem a obtenção e análise de indicadores precisos do desempenho do estabelecimento.

Marcos Oliveira:

Marcos Oliveira, 35 anos, é gestor de um restaurante popular em sua cidade. Com anos de experiência prática, Marcos consegue equilibrar as operações de atendimento presencial com a crescente demanda pelo delivery.

Sua rotina inclui atividades essenciais, como planejamento de rotas para entregadores, supervisão do preparo e envio de pedidos, resolução de problemas operacionais, como atrasos e erros, e monitoramento contínuo do desempenho da equipe. Marcos utiliza relatórios analíticos para embasar suas decisões e prioriza a melhoria da comunicação e colaboração entre atendentes, cozinheiros e entregadores. Ele busca soluções tecnológicas que centralizem as operações de delivery, reduzam o tempo dedicado a tarefas administrativas e aumentem a eficiência operacional do restaurante.

3.3.1.2.5 Execução da Entrevista

As entrevistas foram conduzidas em formato semiestruturado, permitindo explorar as perguntas previamente definidas, mas com flexibilidade para aprofundar questões relevantes que surgissem durante o diálogo ou omitir perguntas que se mostrassem repetitivas. O fluxo de cada entrevista foi organizado da seguinte forma:

1. **Apresentação:** Introdução breve ao projeto, com explicação dos objetivos da entrevista, visando criar um ambiente de confiança e alinhamento com o entrevistado.

2. **Exploração:** Aplicação do roteiro de perguntas, com foco em compreender as necessidades, desafios e expectativas dos entrevistados.

3. **Finalização:** Ao término da entrevista, foi elaborado em conjunto com o entrevistado o texto de apresentação de seu perfil, disponível na seção 3.3.1.2.4. A finalização incluiu um agradecimento pelo tempo e contribuições do entrevistado.

Durante as entrevistas, foram feitas anotações para assegurar que todas as informações relevantes fossem capturadas e analisadas posteriormente. As respostas foram registradas de maneira objetiva e clara, visando facilitar a organização e a interpretação dos dados coletados. Para consulta completa, as anotações das entrevistas estão disponíveis nos apêndices C e D.

3.3.1.2.6 Análise da Entrevista

A análise das entrevistas revelou *insights* valiosos sobre os desafios e oportunidades no gerenciamento de delivery. As anotações realizadas durante as entrevistas permitiram organizar os dados coletados em categorias-chave, facilitando a interpretação e o entendimento das informações. Para detalhes mais específicos, os *insights* individuais de cada entrevista estão apresentados no apêndice C.1 e D.1. A seguir, são destacados os principais resultados consolidados das entrevistas:

- **Acesso ao Sistema:** Melhorar a experiência em dispositivos móveis e *desktop* para possibilitar acesso eficiente, independentemente do local.

- **Análise de Desempenho:** Relatórios automáticos com métricas e monitoramento em tempo real, viabilizando decisões rápidas.

- **Automação:** Automatizar relatórios, atualizações de status, monitoramento e planejamento de rotas para reduzir o trabalho manual e aumentar a eficiência.

- **Capacitação da Equipe:** Implementar procedimentos padronizados, como etiquetas e *checklists*, combinados com treinamento contínuo, uso de simulações práticas e programas de reconhecimento para elevar a produtividade.
- **Centralização:** Integrar sistemas para registro, rastreamento e análise de dados, com monitoramento em tempo real.
- **Centralização e Automação:** Sistemas integrados e automáticos para reduzir erros, retrabalhos e melhorar a eficiência operacional.
- **Comunicação:** Melhorar a interação entre cozinha, entregadores e gerentes por meio de ferramentas integradas, minimizando falhas na transição de pedidos e aumentando a rastreabilidade.
- **Dificuldades com Métricas:** Solucionar a falta de integração e automação, que tornam a análise de métricas lenta e sujeita a erros.
- **Foco no Cliente:** Desenvolver ferramentas que permitam rastreamento em tempo real, coleta de *feedbacks* e notificações automáticas para atualizar o status dos pedidos, promovendo maior satisfação do cliente.
- **Gestão de Pedidos:** Automatizar o registro, atualização e acompanhamento de pedidos para evitar gargalos e economizar tempo.
- **Impacto Geral:** Promover operações mais rápidas, confiáveis e escaláveis, mesmo em períodos de alta demanda.
- **Melhoria Desejada:** Criar um sistema responsivo e integrado, compatível com dispositivos móveis e *desktop*.
- **Melhoria na Experiência do Cliente:** Incorporar funcionalidades como rastreamento em tempo real, notificações automáticas e ferramentas de *feedback* para otimizar a comunicação e a satisfação do cliente.
- **Operações Mais Ágeis:** Automatizar rotas para evitar atrasos.
- **Organização de Dados:** Substituir planilhas manuais por relatórios automáticos e dashboards gráficos que consolidem informações críticas.
- **Problemas no Fluxo de Pedidos:** Identificar e corrigir problemas recorrentes na comunicação entre setores e no registro de informações.
- **Relatórios e Frequência:** Implementar relatórios diários para ajustes imediatos e relatórios semanais para planejamento estratégico, com envio automatizado por e-mail ou notificações.

- **Tarefas Simplificadas:** Automatizar e integrar processos para evitar retrabalho e aumentar a produtividade geral.

3.3.1.3 Brainstorming

Conforme abordado no livro Interação Humano-Computador e Experiência do Usuário, o brainstorming é uma técnica colaborativa utilizada para explorar ideias de forma ampla e criativa. No contexto deste projeto, será conduzido para identificar funcionalidades, características e prioridades do sistema a ser desenvolvido. Essa abordagem é especialmente útil no estágio inicial do desenvolvimento, quando o objetivo é capturar ideias de maneira abrangente e sem limitações (BARBOSA; ANDRADE, 2021).

3.3.1.3.1 Objetivos do Brainstorming

O principal objetivo do brainstorming foi consolidar e expandir os requisitos do sistema, aproveitando os dados coletados durante as entrevistas e os requisitos levantados durante a observações.

3.3.1.3.2 Metodologia do Brainstorming

A metodologia adotada seguiu os passos descritos abaixo, garantindo um processo estruturado e eficiente:

1. **Preparação:** Análise e organização dos dados coletados durante as entrevistas e observações, com foco na identificação de pontos-chave e lacunas nos processos existentes.
2. **Sessão de Brainstorming:** Encontro colaborativo entre os membros do projeto para discutir ideias livremente, incentivando a criatividade e a participação ativa de todos os envolvidos.
3. **Documentação:** Registro de todas as ideias apresentadas, categorizadas em épicos para facilitar a organização e o planejamento subsequente.

3.3.1.3.3 Execução do Brainstorming

Durante a execução, os participantes proporão soluções para atender às demandas práticas observadas no ambiente do restaurante. A dinâmica incluirá momentos de discussão para refinar as ideias apresentadas, garantindo que os requisitos sejam viáveis e adequados aos objetivos do projeto. Para organizar e visualizar as contribuições, será utilizada a ferramenta Obsidian 3.8, que possibilita a criação de tabelas detalhadas e estruturadas.

3.3.1.3.4 Resultados do Brainstorming

O brainstorming resultou em uma lista consolidada de funcionalidades e requisitos para o sistema, situados no anexo A, abrangendo desde melhorias operacionais até inovações tecnológicas. Essas ideias serão integradas ao planejamento do projeto, constituindo a base para o desenvolvimento de uma aplicação funcional, eficiente e alinhada às necessidades práticas do restaurante no contexto do delivery.

3.3.1.4 Considerações Finais

A técnica de observação, aliada às entrevistas, foi essencial para compreender as nuances do gerenciamento de um restaurante self-service com foco em delivery. Os *insights* obtidos não apenas reforçam a importância de soluções tecnológicas integradas, mas também destacam a necessidade de ferramentas que sejam intuitivas, eficientes e alinhadas às demandas reais do mercado.

3.3.2 Priorização de Requisitos

A priorização de requisitos é uma prática essencial no desenvolvimento de produtos, especialmente em contextos onde os recursos são limitados e as expectativas dos clientes são altas. Essa abordagem permite que equipes priorizem funcionalidades e capacidades com base em seu valor relativo, garantindo que os elementos mais críticos sejam entregues primeiro. Além de maximizar o valor do negócio dentro das restrições do projeto, a priorização também ajuda a ajustar o escopo de forma dinâmica, permitindo que o trabalho esteja sempre alinhado aos objetivos estratégicos (WIEGERS; BEATTY, 2013).

3.3.2.1 MoSCoW

Dentre as várias técnicas de priorização disponíveis, destaca-se o método MoSCoW, que classifica os requisitos em quatro categorias principais (WIEGERS; BEATTY, 2013):

- **Must (Deve)**: Requisitos indispensáveis para o sucesso da solução.
- **Should (Deveria)**: Requisitos importantes que devem ser incluídos, se possível.
- **Could (Poderia)**: Requisitos desejáveis, mas que podem ser adiados ou eliminados dependendo da disponibilidade de recursos.
- **Won't (Não será agora)**: Requisitos que não serão implementados no momento, mas que podem ser considerados em versões futuras.

Escolher o método MoSCoW em detrimento de outras técnicas, como a priorização tradicional por escalas de alta, média e baixa prioridade, deve-se à sua simplicidade e

clareza. O MoSCoW facilita a comunicação ao fornecer categorias que enfatizam a importância e a urgência dos requisitos, em vez de classificações genéricas. Isso ajuda a alinhar expectativas e evitar mal-entendidos quanto à priorização dos itens. Além disso, o uso do MoSCoW incentiva uma abordagem iterativa e flexível, permitindo ajustes conforme as condições do projeto evoluem (WIEGERS; BEATTY, 2013).

3.3.2.2 Resultado da Priorização de Requisitos

Abaixo encontra-se a priorização detalhada dos requisitos funcionais (RF), organizada com base na metodologia MoSCoW. Para este projeto, optou-se por implementar inicialmente os requisitos classificados como Must e Should, garantindo o foco nas funcionalidades essenciais para o alcance dos objetivos propostos. Ressalta-se que o MoSCoW é uma abordagem iterativa e flexível, permitindo que as prioridades sejam ajustadas conforme as condições e necessidades do projeto evoluem. Dessa forma, requisitos classificados como Could ou até mesmo Won't podem ser reavaliados e implementados em etapas futuras, de acordo com o andamento do desenvolvimento e a disponibilidade de recursos. A justificativa detalhada para a priorização de cada requisito encontra-se disponível no apêndice F.

Must

- RF01 - Registrar Nome (E01 - F01)
- RF02 - Registrar Telefone (E01 - F01)
- RF03 - Registrar Endereço (E01 - F01)
- RF04 - Registrar Pedido (E02 - F02)
- RF05 - Resumo do Pedido (E02 - F02)
- RF06 - Forma de Pagamento (E02 - F02)
- RF07 - Notificar Cozinha (E02 - F03)
- RF08 - Ler QR Code do Cozinheiro (E02 - F03)
- RF09 - Vincular Cozinheiro ao Pedido (E02 - F03)
- RF10 - Atualizar Status do Pedido para Estabelecimento (E02 - F03)
- RF11 - Organizar Pedidos por Localidade (E03 - F04)
- RF12 - Ler QR Code do Motoboy (E03 - F05)
- RF13 - Informar Pedidos ao Motoboy (E03 - F05)

- RF14 - Vincular Pedido ao Motoboy (E03 - F05)
- RF15 - Registrar Motoboy (E04 - F06)
- RF16 - Gerar QR Code do Motoboy (E04 - F06)
- RF17 - Registrar Cozinheiro (E04 - F06)
- RF18 - Gerar QR Code do Cozinheiro (E04 - F06)
- RF19 - Definir Cargo e Permissões do Funcionário (E04 - F06)
- RF20 - Registrar Marmitas Feitas (E06 - F10)
- RF21 - Registrar Entregas Feitas (E06 - F10)
- RF22 - Autenticação e Controle de Acesso por Cargo (E04 - F06)
- RF23 - Gerenciar Áreas de Entrega (E03 - F04)
- RF24 - Registrar Produtos (E02 - F02)
- RF25 - Registrar Pratos (Pratos do Cardápio) (E02 - F02)
- RF26 - Atualizar Status do Pedido via Leitura de QR Code (E02 - F03)

Should

- RF27 - Editar Registro (E01 - F01)
- RF28 - Registrar Observações do Pedido (E02 - F02)
- RF29 - Gerar Balanço (E06 - F08)
- RF30 - Identificar Recorrência de Clientes (E06 - F09)
- RF31 - Registrar Erros por Funcionário (E06 - F10)
- RF32 - Notificar em Tempo Real com WebSockets (E02 - F-02)
- RF33 - Calcular Tempo de Trânsito da Entrega via API (E03 - F05)
- RF34 - Gerar QR Code Único Baseado em UUID (E04 - F06)
- RF35 - Realizar Controle Básico de Estoque (E06 - F08)
- RF36 - Permitir Atualização Manual de Status pelo Gestor (E02 - F03)
- RF37 - Exibir Dashboard com Indicadores Gerais (E06 - F08)
- RF38 - Calcular Frete Automaticamente com Base na Região (E03 - F04)

- RF39 - Calcular Ticket Médio (E06 - F08)
- RF40 - Categorizar Pratos Vendidos (E06 - F08)
- RF41 - Contabilizar Novos Clientes (E06 - F09)
- RF42 - Quantidade de Pedidos por Cliente (E06 - F09)
- RF43 - Gerar Lista de Desempenho de Funcionário (E06 - F10)
- RF44 - Gerar Relatório de Produção Detalhado (E06 - F08)
- RF45 - Analisar Tempo de Preparo por Prato (E06 - F08)
- RF46 - Gerar Relatório de Entregas com Métricas Avançadas (E06 - F10)
- RF47 - Realizar Análise de Desempenho por Período (E06 - F08)
- RF48 - Gerar Insights de Negócio com Análise de IA (E06 - F08)

Could

- RF49 - Atualizar Status do Pedido para Cliente (E02 - F03)
- RF50 - Imprimir Pedido (E02 - F03)
- RF51 - Destacar Observações na Nota (E02 - F02)
- RF52 - Notificar Cliente pelo WhatsApp (E02 - F02)
- RF53 - Notificar Cliente sobre Entrega (E03 - F05)
- RF54 - Definir Limite de Itens por Região de Entrega (E03 - F04)
- RF55 - Itens mais Excluídos pelo Cliente (E06 - F09)

Won't

- RF56 - Observações mais Feitas por Cliente (E06 - F09)
- RF57 - Avisar Motoboy (E02 - F03)
- RF58 - Feedback pelo Cliente (E06 - F09)
- RF59 - Enviar Pesquisa de Satisfação (E05 - F07)
- RF60 - Coletar Dados de Feedback (E05 - F07)
- RF61 - Analisar Feedback (E05 - F07)

Legenda:

- **E01**: Épico relacionado aos Clientes.
- **E02**: Épico relacionado aos Pedidos.
- **E03**: Épico relacionado à Entrega.
- **E04**: Épico relacionado aos Funcionários.
- **E05**: Épico relacionado ao Feedback.
- **E06**: Épico relacionado ao Dashboard.
- **F01**: Funcionalidade de Registro de Clientes.
- **F02**: Funcionalidade de Registro de Pedidos.
- **F03**: Funcionalidade de Preparação de Pedidos.
- **F04**: Funcionalidade de Organização de Entregas.
- **F05**: Funcionalidade do Motoboy.
- **F06**: Funcionalidade de Registros de Funcionários.
- **F07**: Funcionalidade de Feedback do Cliente.
- **F08**: Funcionalidade de Relatórios Financeiros.
- **F09**: Funcionalidade de Relatórios de Clientes.
- **F10**: Funcionalidade de Relatórios de Funcionários.

3.3.3 Requisitos Não Funcionais

Os requisitos não funcionais (RNF) complementam os requisitos funcionais ao estabelecerem padrões e níveis de qualidade para o funcionamento do software. Eles garantem que o sistema não apenas cumpra suas funções, mas o faça de forma eficiente, segura e confiável. Enquanto os requisitos funcionais tratam de tarefas específicas do usuário, os não funcionais se aplicam de forma geral ao sistema, abrangendo aspectos como desempenho, segurança, usabilidade e escalabilidade. Esses requisitos são essenciais para assegurar que a aplicação atenda às expectativas técnicas e de qualidade ao longo de seu uso (VAZQUEZ; SIMõES, 2016).

Abaixo, encontram-se os requisitos não funcionais definidos para este projeto, com base nos objetivos gerais e específicos estabelecidos:

- **RNF01 - Desempenho:** O sistema deve responder às solicitações de funcionalidades críticas, como registro de pedidos e atualizações de status, em até 2 segundos sob condições normais de uso.
- **RNF02 - Segurança:** O software deve implementar autenticação de usuários, criptografia de dados sensíveis, garantindo sigilo e privacidade das informações.
- **RNF03 - Conformidade com Padrões:** O sistema deve estar em conformidade com normas e guias técnicos, como práticas de segurança e usabilidade, assegurando a aderência a padrões da organização e às regulamentações legais.
- **RNF04 - Usabilidade:** A interface deve ser intuitiva e acessível, permitindo que usuários realizem operações administrativas com facilidade e reduzindo a necessidade de treinamento adicional.
- **RNF05 - Portabilidade:** A aplicação deve ser compatível com diferentes sistemas operacionais modernos e dispositivos móveis (Android e iOS).
- **RNF06 - Escalabilidade:** O sistema deve suportar o aumento gradual do número de usuários e do volume de pedidos sem comprometer o desempenho ou a estabilidade.
- **RNF07 - Manutenibilidade:** A aplicação deve ser desenvolvida com uma arquitetura modular, facilitando futuras atualizações, correções de bugs e adição de novas funcionalidades.
- **RNF08 - Qualidade de Serviço:** O software deve garantir confiabilidade, facilidade de uso e eficiência, atendendo às expectativas de experiência do usuário e suporte operacional.

3.4 Casos de Uso

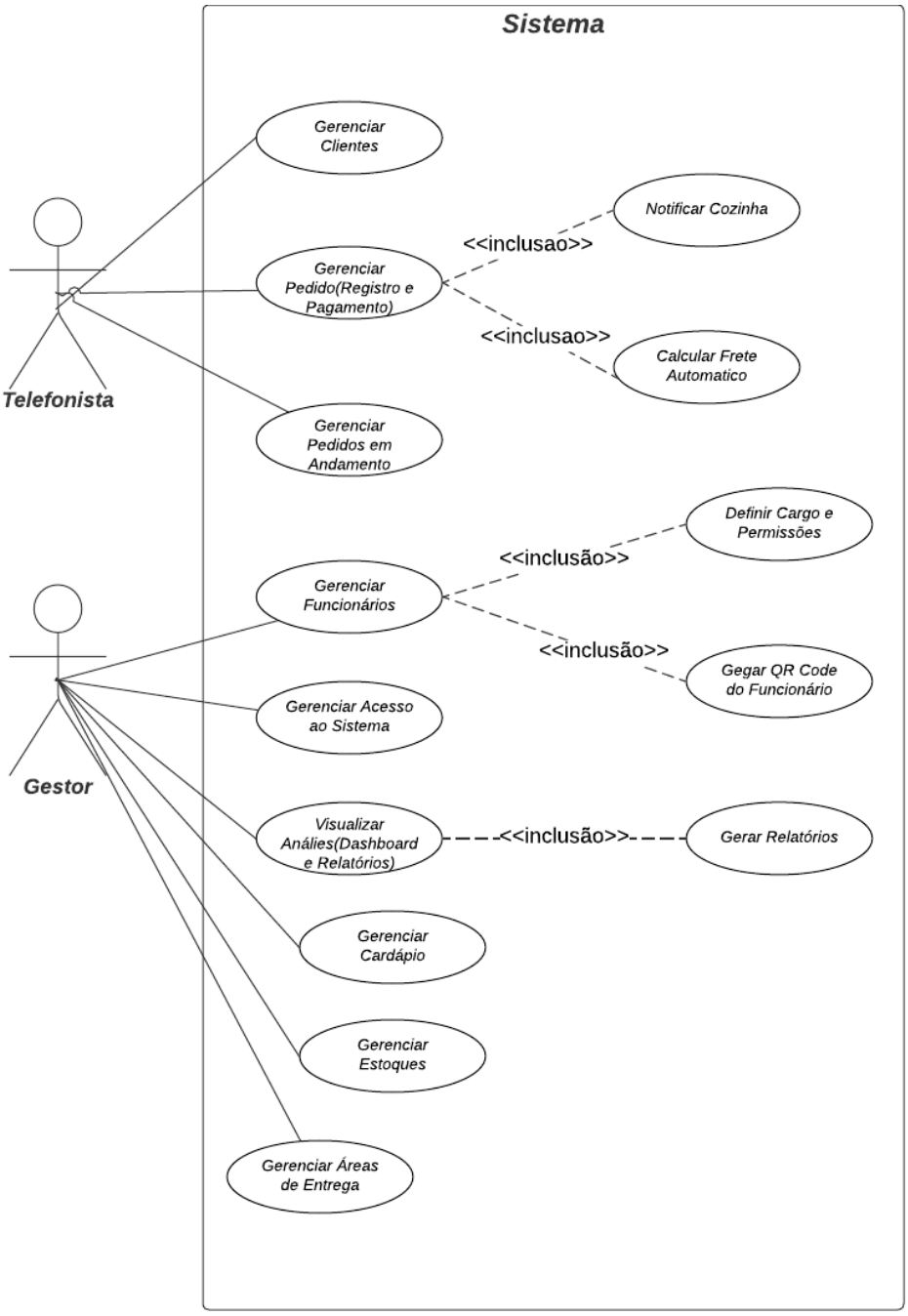
Casos de uso são representações dinâmicas do comportamento de um sistema, mostrando as interações entre os atores (usuários ou sistemas externos) e os elementos do sistema. Eles são centrais para modelar o comportamento e o contexto do sistema, ajudando a visualizar, especificar e documentar os requisitos de maneira clara e compreensível.

Os casos de uso apresentam um conjunto de interações, os atores envolvidos e seus relacionamentos, proporcionando uma visão externa de como o sistema será usado. Eles são fundamentais tanto para o desenvolvimento (engenharia direta) quanto para a análise de sistemas existentes (engenharia reversa), auxiliando desenvolvedores a entender e implementar o comportamento esperado (BOOCH; RUMBAUGH; JACOBSON, 2005).

Diagrama de Casos de Uso - Gestor e Telefonista

Abaixo se encontra o diagrama dos casos de uso para o fluxo do gestor e telefonista, figura 2. É importante ressaltar que este diagrama apresenta especificamente os casos de uso relacionados aos atores Gestor e Telefonista, não abrangendo todos os casos de uso do sistema. Os casos de uso referentes aos demais atores (Cozinheiro e Motoboy) são descritos textualmente nas seções subsequentes.

Figura 2 – Casos de uso, Fluxo do Gestor e Telefonista.



Fonte: Autores (2025).

3.4.0.1 Descrição dos casos de uso

A Descrição de casos de uso descreve os cenários de interação entre os atores e o sistema em termos detalhados, incluindo descrição, ator primarário, fluxos de eventos principais e alternativos, pré-condições, pós-condições e requisitos associados. Cada caso de uso deve ser estruturado para capturar o comportamento esperado do sistema, garantindo que as necessidades dos usuários sejam atendidas.

Descrição de Casos de Uso - Gestor e Telefonista

Caso de Uso: Gerenciar Acesso ao Sistema

Descrição: Permite que um usuário realize login para acessar as funcionalidades correspondentes às suas permissões e realize logout de forma segura.

Ator Primário: Gestor, Telefonista, Cozinheiro, Motoboy

Pré-condições:

- O usuário deve possuir um registro ativo no sistema.

Fluxo Principal:

1. O usuário acessa a tela de login e insere suas credenciais (ex: CPF e senha).
2. O sistema valida as credenciais e concede acesso ao painel correspondente ao seu cargo e permissões.
3. Ao final do uso, o usuário seleciona a opção "Logout"e o sistema encerra a sessão de forma segura.

Fluxos Alternativos:

- **F1:** Se as credenciais de login estiverem incorretas, o sistema exibe uma mensagem de erro e nega o acesso.

Pós-condições:

- O usuário está autenticado e pode utilizar o sistema (após login).
- A sessão do usuário é encerrada com segurança (após logout).

Requisitos Funcionais Relacionados:

- RF22 - Autenticação e Controle de Acesso por Cargo

Caso de Uso: Gerenciar Clientes

Descrição: Permite que o gestor ou telefonista registre novas informações de clientes e edite os dados de clientes já existentes no sistema.

Ator Primário: Gestor ou Telefonista

Pré-condições:

- O ator deve estar autenticado no sistema.

Fluxo Principal (Registrar):

1. O ator seleciona a opção "Registrar Cliente".
2. O sistema exibe um formulário para preenchimento.
3. O ator insere as informações do cliente (nome, telefone, endereço) e confirma.
4. O sistema salva os dados do novo cliente no banco de dados.

Fluxo Principal (Editar):

1. O ator busca e seleciona um cliente já existente.
2. O sistema exibe as informações atuais do cliente.
3. O ator altera os campos desejados e confirma as alterações.
4. O sistema salva as informações atualizadas.

Pós-condições:

- As informações do cliente estão registradas ou atualizadas no sistema e disponíveis para uso.

Requisitos Funcionais Relacionados:

- RF01 - Registrar Nome
- RF02 - Registrar Telefone
- RF03 - Registrar Endereço
- RF27 - Editar Registro

Caso de Uso: Gerenciar Cardápio

Descrição: Permite ao gestor adicionar, editar ou remover pratos e produtos (bebidas, sobremesas) que são oferecidos no cardápio do restaurante.

Ator Primário: Gestor

Pré-condições:

- O gestor deve estar autenticado no sistema.

Fluxo Principal:

1. O gestor acessa a área de "Gerenciamento de Cardápio".
2. O gestor pode (a) Adicionar um novo item, preenchendo seus detalhes como nome e preço; (b) Selecionar um item existente para editar suas informações; ou (c) Remover um item do cardápio.
3. O sistema valida e salva as alterações no banco de dados.

Pós-condições:

- O cardápio do restaurante está atualizado e pronto para ser utilizado em novos pedidos.

Requisitos Funcionais Relacionados:

- RF24 - Registrar Produtos
- RF25 - Registrar Pratos (Pratos do Cardápio)

Caso de Uso: Gerenciar Pedido (Registro e Pagamento)

Descrição: Permite que o gestor ou telefonista registre um pedido completo para um cliente, incluindo seleção de itens, observações, cálculo de frete e definição do pagamento, notificando a cozinha ao final.

Ator Primário: Gestor ou Telefonista

Pré-condições:

- O cliente e os itens do cardápio devem estar registrados no sistema.
- O ator deve estar autenticado.

Fluxo Principal:

1. O ator seleciona o cliente para o qual o pedido será feito.
2. O ator adiciona os itens do cardápio (pratos e produtos) desejados ao pedido.
3. O sistema, com base no endereço do cliente, calcula e exibe automaticamente o valor do frete.
4. O ator insere observações específicas do cliente para o pedido (ex: "sem cebola").
5. O sistema apresenta um resumo do pedido com todos os itens, subtotal, frete e valor total.
6. O ator seleciona a forma de pagamento informada pelo cliente.
7. O ator confirma o pedido.
8. O sistema registra o pedido com um identificador único, define seu status inicial e notifica a cozinha automaticamente.

Pós-condições:

- O pedido está registrado no sistema com o status "Aguardando Preparo".
- A cozinha é notificada sobre o novo pedido.

Requisitos Funcionais Relacionados:

- RF04 - Registrar Pedido
- RF05 - Resumo do Pedido
- RF06 - Forma de Pagamento
- RF07 - Notificar Cozinha
- RF28 - Registrar Observações do Pedido
- RF38 - Calcular Frete Automaticamente com Base na Região
- RF51 - Destacar Observações na Nota

Caso de Uso: Gerenciar Pedidos em Andamento

Descrição: Permite ao gestor ou telefonista supervisionar os pedidosvquando necessário e atualizando seus status manualmente para corrigir eventuais falhas no fluxo.

Ator Primário: Gestor ou Telefonista

Pré-condições:

- O ator deve estar autenticado no sistema.
- Devem existir pedidos em andamento.

Fluxo Principal:

1. O ator visualiza a lista de todos os pedidos em andamento.
2. Para acelerar a produção de um pedido, o ator o seleciona e aplica o marcador "Prioritário".
3. Para corrigir um status incorreto, o ator seleciona o pedido, escolhe a opção "Atualizar Status Manualmente"e seleciona o status correto na lista.

Pós-condições:

- O status do pedido é corrigido e o fluxo do processo continua corretamente.

Requisitos Funcionais Relacionados:

- RF36 - Permitir Atualização Manual de Status pelo Gestor

Caso de Uso: Gerenciar Funcionários

Descrição: Permite ao gestor registrar novos funcionários (cozinheiros, motoboys), definir seus cargos, permissões de acesso e gerar seus respectivos QR Codes de identificação.

Ator Primário: Gestor

Pré-condições:

- O gestor deve estar autenticado no sistema.

Fluxo Principal:

1. O gestor acessa a funcionalidade de "Gerenciar Funcionários".
2. O gestor preenche os dados do novo funcionário (nome, CPF, etc.).
3. O gestor define o cargo (Cozinheiro, Motoboy), e o sistema atribui as permissões padrão para aquele cargo.
4. Ao salvar, o sistema cria o registro do funcionário e gera um QR Code único para sua identificação.

Pós-condições:

- O funcionário está registrado no sistema, com permissões definidas e com um QR Code pronto para uso.

Requisitos Funcionais Relacionados:

- RF15 - Registrar Motoboy
- RF16 - Gerar QR Code do Motoboy
- RF17 - Registrar Cozinheiro
- RF18 - Gerar QR Code do Cozinheiro
- RF19 - Definir Cargo e Permissões do Funcionário
- RF34 - Gerar QR Code Único Baseado em UUID

Caso de Uso: Gerenciar Áreas de Entrega

Descrição: Permite ao gestor criar, visualizar e editar as áreas geográficas de entrega, associando a cada uma delas um valor de frete e regras específicas.

Ator Primário: Gestor

Pré-condições:

- O gestor deve estar autenticado no sistema.

Fluxo Principal:

1. O gestor seleciona a opção "Gerenciar Áreas de Entrega".
2. O sistema exibe um mapa interativo.
3. O gestor cria uma nova área (desenhando no mapa) ou seleciona uma existente para editar.
4. O gestor define para a área o valor do frete e, opcionalmente, um limite de itens por entrega.
5. O gestor salva a configuração.

Pós-condições:

- As áreas de entrega, com suas taxas e regras, estão configuradas para uso no cálculo automático de frete.

Requisitos Funcionais Relacionados:

- RF11 - Organizar Pedidos por Localidade
- RF23 - Gerenciar Áreas de Entrega
- RF38 - Calcular Frete Automaticamente com Base na Região
- RF54 - Definir Limite de Itens por Região de Entrega

Caso de Uso: Visualizar Análises (Dashboard e Relatórios)

Descrição: Permite ao gestor acessar um dashboard com indicadores em tempo real e gerar diversos relatórios detalhados para uma análise aprofundada do desempenho do negócio.

Ator Primário: Gestor

Pré-condições:

- O gestor deve estar autenticado no sistema.

Fluxo Principal (Dashboard):

1. O gestor acessa a tela principal do sistema (Dashboard).
2. O sistema exibe gráficos e números atualizados sobre pedidos do dia, faturamento, etc.

Fluxo Principal (Relatórios):

1. O gestor acessa a seção de "Relatórios".
2. O gestor seleciona o tipo de relatório desejado (Balanço financeiro, Produção, Desempenho de funcionários, Recorrência de clientes, etc.).
3. O gestor pode aplicar filtros, como período de datas.
4. O sistema processa os dados e gera o relatório, que pode ser visualizado ou exportado.

Pós-condições:

- O gestor obtém insights valiosos para a tomada de decisões estratégicas.

Requisitos Funcionais Relacionados:

- RF29 - Gerar Balanço
- RF30 - Identificar Recorrência de Clientes
- RF31 - Registrar Erros por Funcionário
- RF37 - Exibir Dashboard com Indicadores Gerais
- RF39 a RF48 (Todos os demais requisitos de análise, relatórios e IA)

Caso de Uso: Gerenciar Estoque

Descrição: Permite ao gestor registrar a entrada de produtos no estoque, enquanto o sistema realiza a baixa automática na conclusão de um pedido.

Ator Primário: Gestor, Sistema

Pré-condições:

- O gestor deve estar autenticado.
- Os produtos devem estar cadastrados e marcados para controle de estoque.

Fluxo Principal:

1. O gestor acessa a função "Controle de Estoque", seleciona um produto e informa a quantidade de entrada (compra).
2. O sistema incrementa a quantidade do produto em estoque.
3. Quando um pedido que contém esse produto é pago/finalizado, o sistema automaticamente decrementa a quantidade correspondente do estoque.

Pós-condições:

- A quantidade de produtos em estoque está sempre atualizada.

Requisitos Funcionais Relacionados:

- RF36 - Realizar Controle Básico de Estoque

Caso de Uso: Gerenciar Pesquisa de Satisfação

Descrição: Descreve como o sistema envia pesquisas de satisfação aos clientes após a entrega e como o gestor analisa os dados coletados.

Ator Primário: Sistema, Gestor

Pré-condições:

- O pedido deve ter sido concluído e entregue.
- O cliente deve ter um meio de contato válido.

Fluxo Principal:

1. Após a conclusão da entrega, o sistema envia automaticamente um link para a pesquisa de satisfação ao cliente.
2. O sistema coleta e armazena as respostas do cliente.
3. O gestor acessa a área de "Análise de Feedback"para visualizar os resultados consolidados.

Pós-condições:

- O feedback do cliente é coletado e fica disponível para análise gerencial.

Requisitos Funcionais Relacionados:

- RF58 - Enviar Pesquisa de Satisfação
- RF59 - Coletar Dados de Feedback
- RF60 - Analisar Feedback

Descrição de Casos de Uso - Funcionário (Cozinheiro e Motoboy)

Caso de Uso: Executar Etapa do Pedido

Descrição: Descreve o fluxo de trabalho completo de um funcionário (cozinheiro ou motoboy) ao interagir com o sistema para avançar um pedido em sua respectiva etapa (preparo ou entrega).

Ator Primário: Cozinheiro, Motoboy, Sistema

Pré-condições:

- O funcionário deve estar registrado e autenticado no sistema (ou em um dispositivo do restaurante).
- Devem existir pedidos aguardando preparo ou aguardando entrega.

Fluxo Principal:

1. O funcionário (ou um gestor) realiza a leitura do QR Code do funcionário para associá-lo a um pedido.
2. O sistema vincula o cozinheiro ao preparo do pedido ou o motoboy à sua entrega.
3. O sistema atualiza o status do pedido internamente ("Em Preparo", "Em Rota de Entrega"). Esta atualização é visível em tempo real nos painéis de gestão.
4. O sistema pode opcionalmente imprimir os detalhes do pedido para o cozinheiro ou um resumo para o motoboy.
5. O sistema pode enviar uma notificação automática ao cliente informando sobre a nova etapa ("Seu pedido está sendo preparado!", "Seu pedido saiu para entrega!").
6. Ao finalizar sua tarefa (marmita pronta ou entrega concluída), o funcionário sinaliza a conclusão no sistema (por exemplo, através de uma nova leitura de QR Code ou um botão).
7. O sistema registra a conclusão da etapa (marmita feita / entrega feita).

Pós-condições:

- A etapa do pedido (preparo ou entrega) é concluída e seu status é atualizado no sistema.
- Os registros de produção e entrega são contabilizados.

Requisitos Funcionais Relacionados:

- RF08 - Ler QR Code do Cozinheiro
- RF09 - Vincular Cozinheiro ao Pedido
- RF10 - Atualizar Status do Pedido para Estabelecimento
- RF12 - Ler QR Code do Motoboy
- RF13 - Informar Pedidos ao Motoboy
- RF14 - Vincular Pedido ao Motoboy

- RF20 - Registrar Marmitas Feitas
- RF21 - Registrar Entregas Feitas
- RF26 - Atualizar Status do Pedido via Leitura de QR Code
- RF32 - Implementar Notificações em Tempo Real com WebSockets
- RF33 - Calcular Tempo de Trânsito da Entrega via API
- RF49 - Atualizar Status do Pedido para Cliente
- RF50 - Imprimir Pedido
- RF52 - Notificar Cliente pelo WhatsApp
- RF53 - Notificar Cliente sobre Entrega

Matriz de Priorização dos Casos de Uso

A matriz de priorização dos casos de uso é uma ferramenta que ajuda a identificar e classificar os requisitos de acordo com sua importância e impacto no sistema. Ela é útil para orientar o desenvolvimento. A priorização foi feita com base na priorização dos requisitos 3.3.2.2.

Tabela 3 – Matriz de Priorização dos Casos de Uso

Caso de Uso	**Must**	**Should**	**Could**	**Won't**
Gerenciar Acesso ao Sistema	X			
Gerenciar Clientes	X			
Gerenciar Cardápio	X			
Gerenciar Pedido (Registro e Pagamento)	X			
Gerenciar Pedidos em Andamento		X		
Gerenciar Funcionários	X			
Gerenciar Áreas de Entrega			X	
Visualizar Análises (Dashboard e Relatórios)		X		
Gerenciar Estoque			X	
Gerenciar Pesquisa de Satisfação			X	
Executar Etapa do Pedido	X			

Fonte: Autores (2025).

3.5 Arquitetura

A arquitetura de software é essencial para garantir que um sistema seja sustentável, flexível e de fácil manutenção ao longo do tempo. Neste projeto, adotamos os princípios da *Clean Architecture*, para organizar e estruturar o sistema (MARTIN, 2018).

3.5.1 Clean Architecture

A *Clean Architecture* promove a independência de componentes, garantindo que as mudanças em uma parte do sistema não afetem outras de maneira desnecessária. Esse conceito é alcançado através da aplicação de princípios fundamentais (MARTIN, 2018):

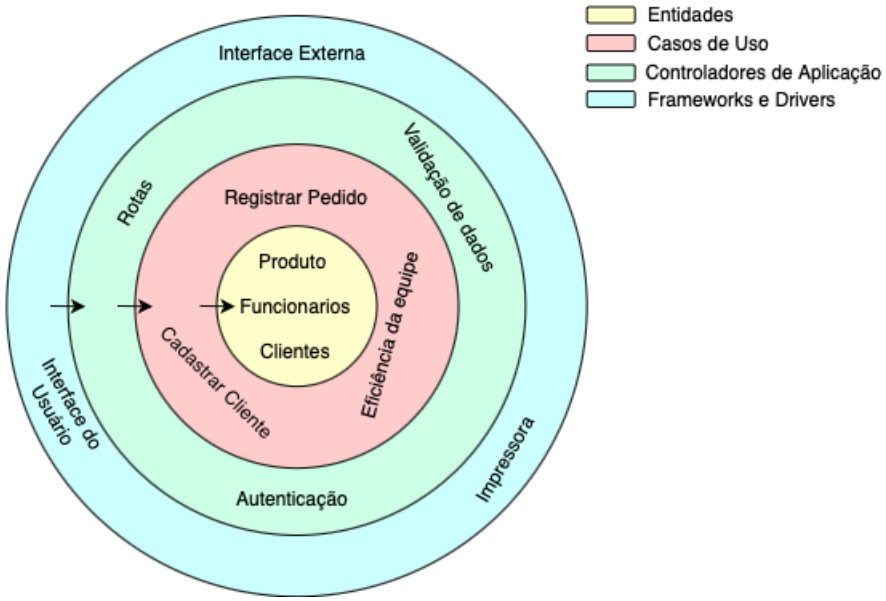
- **Regra da Dependência:** As dependências devem apontar para dentro, isto é, os componentes de níveis mais baixos (detalhes) devem depender dos componentes de níveis mais altos (regras de negócio e casos de uso), e nunca o contrário.
- **Separação de Responsabilidades:** O sistema é dividido em camadas bem definidas, como Entidades, Casos de Uso, Controle de Aplicação e Frameworks e Drivers. Cada camada tem uma responsabilidade clara e independente.
- **Testabilidade:** Ao isolar a lógica de negócios das interfaces externas, como banco de dados ou frameworks, os testes se tornam mais simples e eficazes.
- **Flexibilidade para Mudanças:** A arquitetura é desenhada para ser robusta em mudanças de requisitos, permitindo que novos recursos sejam integrados sem a necessidade de reformulações extensas.

Essa abordagem permite criar sistemas que são fáceis de manter, já que alterações podem ser realizadas com impacto mínimo em outras partes do sistema. Além disso, os sistemas se tornam escaláveis, possibilitando a adição de novos módulos e funcionalidades sem comprometer a estrutura existente. Por fim, o esforço de desenvolvimento e manutenção é reduzido, tornando os sistemas sustentáveis, mesmo à medida que crescem em complexidade e tamanho (MARTIN, 2018).

No contexto deste projeto, a *Clean Architecture* garante a modularidade necessária para integrar novas funcionalidades, suportar múltiplas plataformas — *desktop* e *tablet* — e permitir a comunicação eficiente entre esses dispositivos. Além disso, sua estrutura desacoplada facilita a integração com periféricos, como impressoras, e possibilita expansões futuras para outros equipamentos, mantendo a flexibilidade e a escalabilidade do sistema.

Abaixo está a representação da *Clean Architecture* no nosso projeto, figura 3.

Figura 3 – Visualização da Clean Architecture.

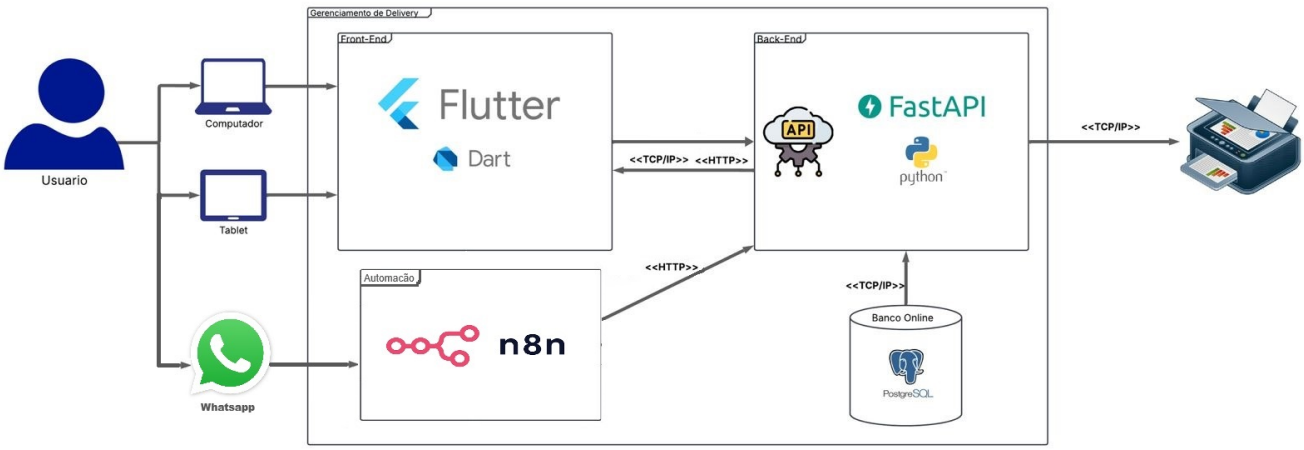


Fonte: Autores (2025).

3.5.2 Arquitetura de Tecnologias

Abaixo está o diagrama de arquitetura listando as tecnologias utilizadas no sistema, figura 4.

Figura 4 – Visualização da Arquitetura de Tecnologias.



Fonte: Autores (2025).

3.5.3 Backend

O *backend* é a camada responsável pelo processamento lógico, comunicação com o armazenamento de dados e *frontend*. Ele atua gerenciando as operações de negócio, garantindo a segurança e respondendo às solicitações dos usuários (AMAZON, 2025).

3.5.3.1 Python

Python é uma linguagem de programação amplamente usada devido à sua simplicidade e poder. No contexto do *backend*, ela é ideal para criar sistemas robustos e escaláveis, graças ao seu extenso ecossistema de bibliotecas e frameworks. A escolha de Python reflete sua capacidade de atender a requisitos complexos com uma curva de aprendizado acessível, permitindo desenvolver soluções que se integram facilmente a outras tecnologias (PYTHON, 2025).

3.5.3.2 FastAPI

FastAPI é um framework moderno e de alto desempenho para a construção de APIs RESTful (Application Programming Interfaces Representational State Transfer). Ele se destaca pelo suporte nativo à validação de dados, documentação automática baseada em OpenAPI, uma especificação padronizada para descrever APIs REST, e uso eficiente de recursos assíncronos. Dessa forma o FastAPI é uma solução para criar serviços rápidos, seguros e escaláveis, essenciais para atender às demandas dos usuários em tempo real (FASTAPI, 2025).

3.5.3.3 Pytest

Pytest é uma ferramenta para a criação e execução de testes automatizados no backend. Ela simplifica a escrita de testes unitários e de integração, assegurando que os componentes do sistema funcionem corretamente e que alterações no código não introduzam novos erros. Assim mantendo a confiabilidade e a qualidade do sistema durante o desenvolvimento contínuo (PYTEST, 2025).

3.5.3.4 n8n

O n8n é uma ferramenta de automação de workflows licenciada sob o modelo fair-code, que combina funcionalidades de inteligência artificial com automação de processos de negócio. Seu principal objetivo é permitir a integração entre diferentes aplicações que possuam API, possibilitando a manipulação de dados com pouco ou nenhum código (N8N, 2025).

3.5.4 Frontend

O termo front-end se refere à interface gráfica do usuário (GUI) com a qual as pessoas interagem. É nela que a interface visual é projetada e implementada (AMAZON, 2025).

3.5.4.1 Dart

Dart é uma linguagem de programação otimizada para o desenvolvimento de aplicações cliente. Desenvolvida pelo Google, ela é o motor por trás do framework Flutter. Suas características, como forte tipagem, suporte a programação assíncrona e desempenho nativo, a tornam ideal para criar aplicativos rápidos e responsivos. No contexto deste projeto, Dart permite implementar uma interface moderna e eficiente, garantindo que o sistema seja amigável e acessível (GOOGLE, 2025a).

3.5.4.2 Flutter

Flutter é um framework multiplataforma que possibilita o desenvolvimento de interfaces de usuário consistentes e de alta qualidade para diferentes dispositivos, como *desktops, smartphones* e navegadores. Ele oferece um conjunto de widgets personalizáveis e ferramentas integradas que aceleram o processo de desenvolvimento, além de garantir que a experiência do usuário seja uniforme independentemente da plataforma utilizada. A escolha do Flutter se da a necessidade de criar um aplicativo visualmente atrativo e funcional com um único código-fonte (GOOGLE, 2025b).

3.5.5 Banco de Dados

O banco de dados é o componente central para o armazenamento e gerenciamento das informações do sistema. Ele permite que os dados sejam estruturados, consultados e manipulados de forma eficiente.

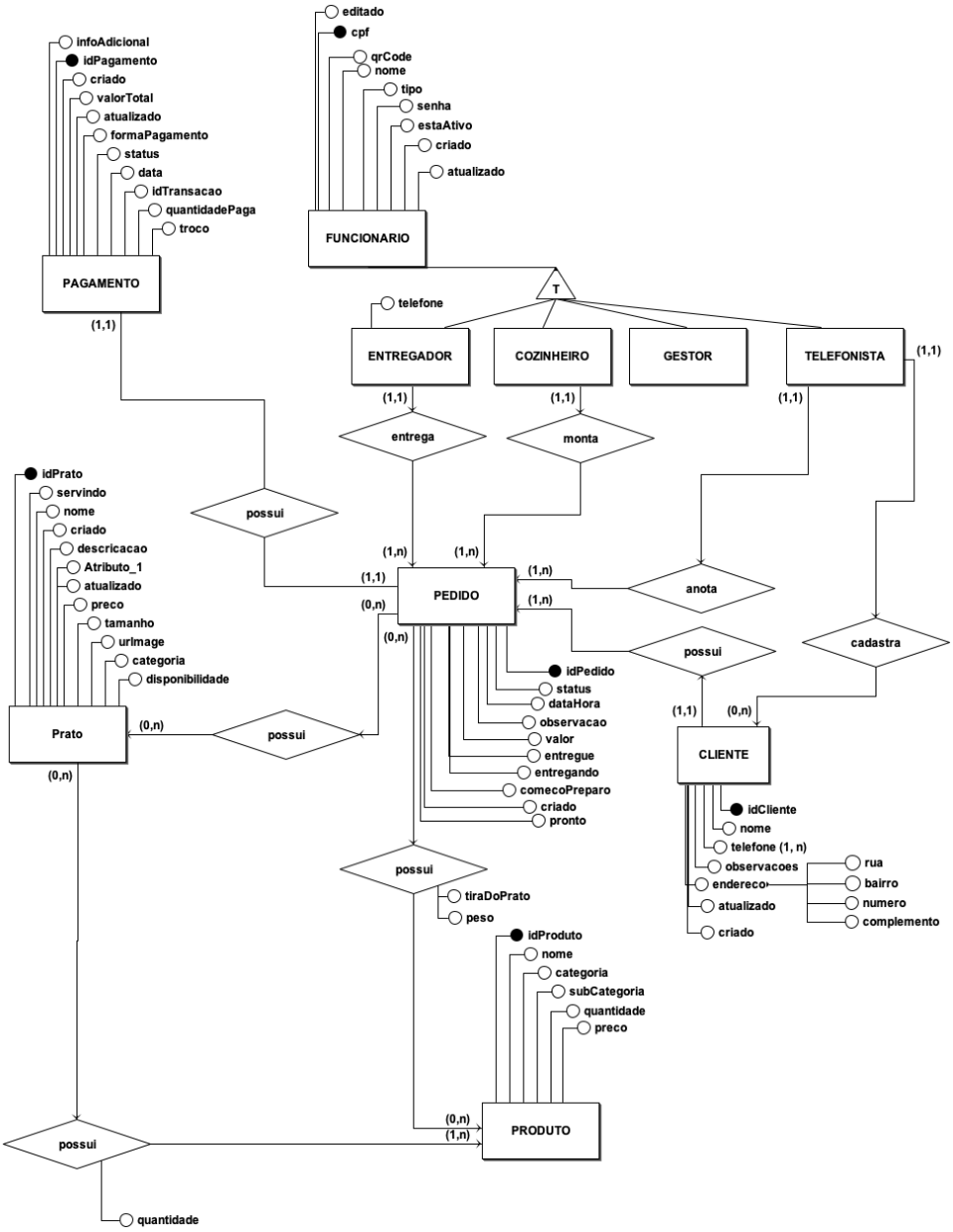
3.5.5.1 PostgreSQL

O PostgreSQL foi escolhido como o banco de dados devido à sua confiabilidade e capacidade de lidar com grandes volumes de dados. Ele oferece suporte a transações ACID (Atomicidade, Consistência, Isolamento e Durabilidade), garantindo a integridade das operações realizadas. Além disso, o PostgreSQL inclui recursos avançados, como suporte nativo a tipos de dados JSON, consultas otimizadas e criação de índices personalizados, tornando-o uma escolha ideal para sistemas que exigem alta performance e escalabilidade (POSTGRESQL, 2025).

3.5.5.2 Diagrama Entidade-Relacionamento

O Diagrama Entidade-Relacionamento (DER) é uma ferramenta amplamente utilizada na modelagem de dados para representar a estrutura lógica de um banco de dados. Ele descreve as entidades do sistema, os atributos dessas entidades e os relacionamentos entre elas. Um DER é fundamental durante a fase inicial do desenvolvimento, pois ajuda a compreender as necessidades de dados e a definir as principais relações no sistema. A representação gráfica das entidades e suas associações permite uma comunicação clara entre desenvolvedores, analistas e partes interessadas, garantindo uma base sólida para o design do banco de dados (KORTH; SILBERSCHATZ; SUDARSHAN, 2019).

Figura 5 – Diagrama Entidade-Relacionamento



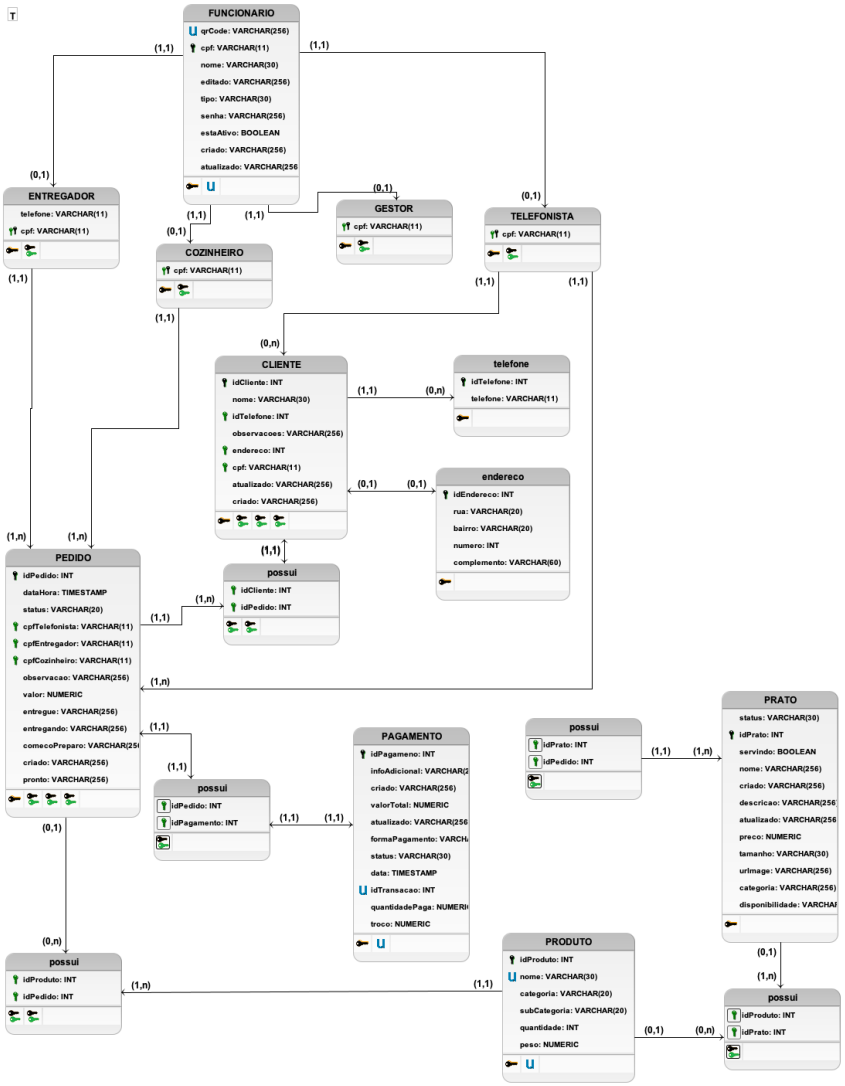
Fonte: Autores (2025).

3.5.5.3 Diagrama Lógico de Dados

O diagrama lógico de dados é elaborado para organizar e detalhar as estruturas que serão armazenadas no banco de dados, apresentando de forma gráfica os componentes e os relacionamentos entre eles. Essa etapa se concentra na definição das entidades, seus atributos e as conexões lógicas que refletem as interações existentes no sistema.

Nessa representação, os elementos principais do sistema são nomeados e suas relações são estabelecidas, permitindo uma visão clara e estruturada da lógica do banco de dados (SAE-UNB, 2025). Um dicionário de dados detalhando os atributos, propriedades, tipos, tamanhos e descrições referentes as entidades, atributos compostos e multivalorados do DLD pode ser encontrado no apendice G.

Figura 6 – Diagrama Logico



Fonte: Autores (2025).

3.6 Validação e Testes

A validação e os testes desempenham um papel essencial no desenvolvimento de software, garantindo a qualidade, segurança e confiabilidade do sistema proposto. O processo de testes é responsável por identificar falhas e garantir que as funcionalidades implementadas atendam aos requisitos especificados. Um sistema confiável deve passar por diferentes níveis de validação para mitigar riscos e assegurar a conformidade com as especificações de projeto (SOMMERVILLE, 2007).

Dessa forma, esta seção detalha a estratégia de testes adotada, abrangendo testes unitários, testes de integração e testes de aceitação do usuário. Cada categoria de teste desempenha um papel específico na garantia da robustez do sistema, contribuindo para sua confiabilidade e desempenho final.

3.6.1 Testes Unitários

Os testes unitários são a primeira camada de validação do software e têm o objetivo de verificar o comportamento correto de componentes individuais do sistema. Esses testes avaliam funções, métodos e classes isoladamente, garantindo que cada unidade de código produza os resultados esperados (MYERS; BADGETT; THOMAS, 2011).

- **Backend (FastAPI)**: Serã utilizado pytest para validar funções e serviços isoladamente.

Os testes serão integrados ao GitHub Actions para execução automática no fluxo de CI.

3.6.2 Testes de Integração

Os testes de integração são fundamentais para verificar a interação entre os diferentes módulos do sistema, assegurando que os componentes comunicam-se corretamente entre si. Esses testes são essenciais para validar a integridade dos fluxos de dados e a consistência das transações entre as camadas da aplicação (BINDER, 2000).

Os testes de integração incluirão:

- Comunicação entre frontend e backend via API REST.
- Interação do backend com o banco de dados PostgreSQL, validando inserção, leitura e remoção de registros.
- Validação da persistência de dados no banco PostgreSQL;
- Validação de endpoints com Pytest.

3.6.3 Testes de Aceitação do Usuário (UAT)

Os testes de aceitação do usuário (*User Acceptance Testing* - UAT) visam validar se o sistema atende às expectativas dos usuários finais e aos requisitos de negócio estabelecidos (STANFORD, 2025). Para isso, serão conduzidos testes com gestores e funcionários de um restaurante real, observando a usabilidade da aplicação e sua adequação ao fluxo operacional do estabelecimento.

A execução dos testes de aceitação ocorrerá por meio da realização dos casos de uso 3.4 previamente definidos no contexto deste trabalho. Durante essa etapa, serão observados possíveis erros cometidos, além de serem consideradas sugestões fornecidas pelos próprios usuários que participarem da validação.

3.7 Segurança e Privacidade

A proteção das informações sensíveis de clientes e restaurantes é fundamental neste sistema. Para garantir a segurança e a privacidade dos dados, serão implementadas medidas como:

- **Autenticação e Autorização**: Uso de mecanismos robustos para assegurar que apenas usuários autorizados tenham acesso às funcionalidades apropriadas.
- **Criptografia**: Aplicação de técnicas de criptografia para proteger dados sensíveis tanto em trânsito quanto em repouso.
- **Conformidade Legal**: Adoção de práticas alinhadas à Lei nº 13.709, de 14 de agosto de 2018, conhecida como Lei Geral de Proteção de Dados (LGPD), que regula o tratamento de dados pessoais, inclusive nos meios digitais, por pessoas físicas ou jurídicas, visando proteger os direitos fundamentais de liberdade, privacidade e o livre desenvolvimento da personalidade do indivíduo (Presidência da República, 2018).

Essas medidas visam assegurar a integridade, confidencialidade e disponibilidade dos dados, alinhando o sistema às melhores práticas de segurança da informação.

3.8 Ferramentas de Apoio Utilizadas

No desenvolvimento do projeto, foram utilizadas diversas ferramentas que auxiliaram na organização, comunicação, modelagem e implementação do sistema.

- **brModelo**: Ferramenta utilizada para modelagem de banco de dados, ajudando na criação do esquema relacional do sistema.

- **Coverage**: Ferramenta utilizada para medir a cobertura dos testes automatizados em projetos Python, indicando quais partes do código foram executadas durante os testes. Auxilia na identificação de trechos não testados, contribuindo para a melhoria da qualidade do software.
- **Discord e WhatsApp**: Aplicativos utilizados para comunicação e alinhamento entre os membros da equipe, facilitando discussões e resolução de problemas.
- **GitHub**: Plataforma utilizada para versionamento e hospedagem de código e repositórios, garantindo colaboração eficiente.
- **Google Scholar e Publish or Perish**: Utilizados para pesquisa acadêmica, ajudando na obtenção de referências bibliográficas para fundamentação teórica do projeto.
- **Lucidchart e Draw.io**: Ferramentas de modelagem visual utilizadas para criar diagramas do sistema, fluxogramas e representações da arquitetura do projeto.
- **Obsidian**: Aplicação para organização de anotações e documentação do projeto, permitindo um fluxo estruturado de informações.
- **Trello**: Ferramenta de gerenciamento de projetos utilizada para organizar tarefas, definir prazos e acompanhar o progresso do desenvolvimento.
- **Visual Studio Code**: Editor de código utilizado para escrever, testar e depurar o código do projeto.

4 Desenvolvimento

Este capítulo descreve o processo mais relevantes de desenvolvimento do sistema, detalhando a execução prática das estratégias apresentadas no Capítulo 3, que trata da metodologia adotada para o projeto. Aqui, serão abordadas as etapas de planejamento, implementação, testes, integração tecnológica e validação com base nos requisitos estabelecidos anteriormente.

Todo o código-fonte do sistema desenvolvido foi hospedado no GitHub, organizado por pastas e tecnologias utilizadas, e pode ser acessado pelos seguintes links:

- **Frontend**: <https://github.com/TCC-Guilherme-Iago/app/tree/main/front>
- **Backend**: <https://github.com/TCC-Guilherme-Iago/app/tree/main/back>
- **Automação (n8n)**: <https://github.com/TCC-Guilherme-Iago/app/tree/main/n8n>

4.1 Etapas e Planejamento das Sprints

Como citado na subseção de práticas utilizadas 3.2.6, foram adotadas abordagens ágeis para organizar e conduzir o desenvolvimento do sistema.

O processo iniciava com reuniões de planejamento, nas quais a equipe discutia os objetivos da sprint e definia as funcionalidades prioritárias a serem desenvolvidas. A partir dessas decisões, era criado e atualizado um quadro Kanban, que tornava visível o fluxo de trabalho, promovendo clareza sobre o andamento de cada tarefa e facilitando a identificação de gargalos e prioridades.

Durante a execução das tarefas, realizavam-se encontros diários breves, os chamados Scrum diários, nos quais os membros da equipe compartilhavam o que estavam fazendo, os obstáculos encontrados e as próximas ações. Essa comunicação era complementada por conversas contínuas por meio do WhatsApp, o que permitia rápida resolução de dúvidas e atualizações ágeis.

Além disso, eram realizadas reuniões semanais para discussões mais aprofundadas e atualizações gerais do projeto. Nessas ocasiões, também se revisava o quadro Kanban conforme as necessidades e o progresso das atividades.

A implementação prática das funcionalidades adotava princípios do XP, como simplicidade no design e entregas frequentes. A equipe seguia uma abordagem de programação colaborativa: enquanto um membro desenvolvia uma funcionalidade, o outro revisava o

código, fornecendo *feedbacks* rápidos e sugerindo melhorias. Caso fosse identificado algo que pudesse ser melhorado, o próprio revisor, ao invés de apenas comentar, podia aplicar diretamente a correção e avisar sobre a alteração — promovendo agilidade e autonomia.

Refatorações foram realizadas de forma constante para manter a qualidade do código. Durante o desenvolvimento, os testes foram realizados principalmente por meio de chamadas diretas à API, permitindo a validação prática das funcionalidades à medida que eram implementadas. Posteriormente, com a base funcional mais consolidada, foram desenvolvidos testes unitários automatizados, acompanhados de relatórios de cobertura, com o objetivo de reforçar a confiabilidade e a estabilidade do sistema.

4.1.1 Implementação das Funcionalidades

O desenvolvimento das funcionalidades do sistema foi estruturado com base nos épicos 3.3.2.2, agrupando os requisitos em conjuntos funcionais maiores que representavam partes essenciais do sistema. Dessa forma, as tarefas foram organizadas de forma a entregar incrementos de valor coesos e alinhados com os objetivos do projeto.

O processo de desenvolvimento teve início pelo backend e foi conduzido com base nos épicos definidos previamente. Cada épico era atribuído a um dos membros da equipe, e sua implementação começava pelas funcionalidades relacionadas ao banco de dados, como criação de tabelas, modelos e integração com o repositório. Após a conclusão dessa etapa, o mesmo responsável seguia com a implementação das demais funcionalidades do backend relacionadas ao mesmo épico, como regras de negócio e serviços específicos.

Esse ciclo se repetia a cada novo épico atribuído, mantendo uma ordem de desenvolvimento consistente e modular. Ao final da implementação de todos os épicos individualmente, foram desenvolvidas as funcionalidades do backend que exigiam a integração entre múltiplos módulos do sistema, garantindo a coesão e o funcionamento completo da aplicação.

Concluído o backend, a equipe deu início ao desenvolvimento do frontend. Cada funcionalidade foi implementada separadamente, mantendo correspondência direta com os épicos desenvolvidos no backend. O frontend foi elaborado com atenção à responsividade e à experiência do usuário.

A tabela 4 abaixo resume a divisão das tarefas por épico, organizadas cronologicamente por sprint, tanto para o backend quanto para o frontend.

Tabela 4 – Cronograma de Implementação por Sprint

Configuração Inicial		
1	10/03 - 16/03	Estruturação do projeto; Setup do ambiente.
Desenvolvimento do Backend		
2	17/03 - 30/03	E01: Épico relacionado aos Clientes; E04: Épico relacionado aos Funcionários.
3	31/03 - 13/04	E02: Épico relacionado aos Pedidos.
4	14/04 - 27/04	E03: Épico relacionado à Entrega.
5	28/04 - 04/05	E06: Épico relacionado ao Dashboard.
6	05/05 - 18/05	Refinamentos no sistema.
Desenvolvimento do Frontend		
7	19/05 - 01/06	E01: Épico relacionado aos Clientes; E04: Épico relacionado aos Funcionários.
8	02/06 - 08/06	E02: Épico relacionado aos Pedidos.
9	09/06 - 22/06	E03: Épico relacionado à Entrega.
10	23/06 - 29/06	E06: Épico relacionado ao Dashboard.
Atividades Gerais do Sistema		
11	30/06 - 13/07	Refinamentos no sistema; Testes Finais.
12	14/07 - 20/07	Preparação para a Defesa.

4.2 Integrações e Arquitetura Tecnológica

Conforme definido na seção 3.5, a arquitetura do projeto foi baseada nos princípios da *Clean Architecture*, com o objetivo de facilitar a escalabilidade, manutenção e testabilidade da aplicação. Durante o desenvolvimento, essa arquitetura se mostrou vantajosa por permitir a separação de responsabilidades em camadas bem definidas, o que organizou melhor o código e possibilitou que diferentes membros da equipe atuassem de forma paralela em módulos distintos, sem causar conflitos.

No backend, a divisão entre as pastas `application`, `domain`, `infrastructure` e `interface` permitiu organizar o código de forma modular, separando claramente os casos de uso, modelos de dados, serviços de infraestrutura e rotas HTTP. Isso tornou o desenvolvimento mais fluido, já que cada membro da equipe pôde atuar em uma camada específica sem comprometer o restante do sistema.

Já no frontend, a estruturação do projeto com pastas como `core` e `features` seguiu uma organização semelhante. A pasta `core` centraliza elementos reutilizáveis e configurações globais, enquanto `features` abriga os módulos funcionais da aplicação. Essa organização também contribuiu para a clareza do projeto e facilitou a implementação de novas funcionalidades de forma incremental.

A figura 7 apresenta a estrutura dos diretórios do frontend (Flutter) e backend

(FastAPI), evidenciando como os princípios da *Clean Architecture* foram aplicados na prática.

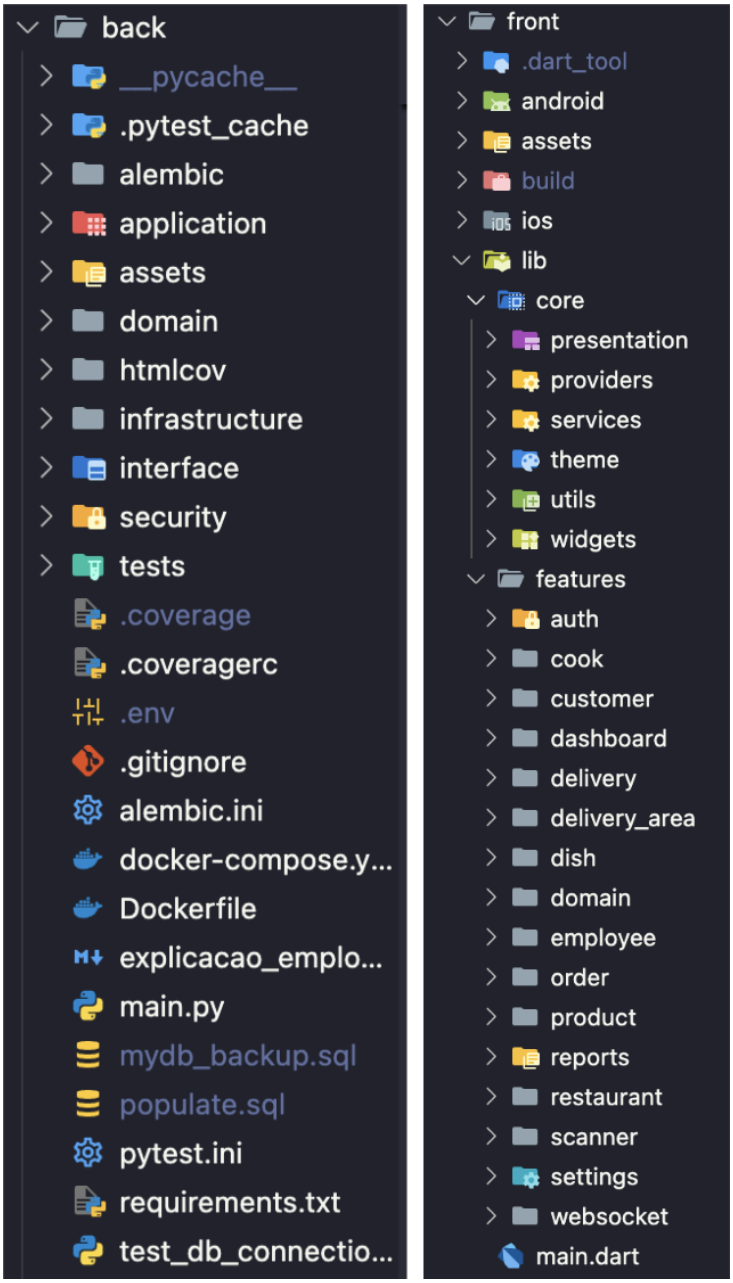


Figura 7 – Estrutura de diretórios do projeto: frontend e backend.

4.3 Telas do Sistema

Conforme descrito na seção 1.3.1, o sistema foi desenvolvido com foco em responsividade. Para ilustrar essa característica, as imagens a seguir mostram o sistema em execução em dispositivos distintos: tablets (com imagens maiores) e *desktops* (com imagens menores), exibidas lado a lado.

As telas apresentadas abaixo representam as principais interfaces da aplicação, fornecendo uma visão geral do padrão de design adotado. Embora não estejam listadas todas as telas implementadas, estas já demonstram a estrutura, organização e consistência visual da aplicação.

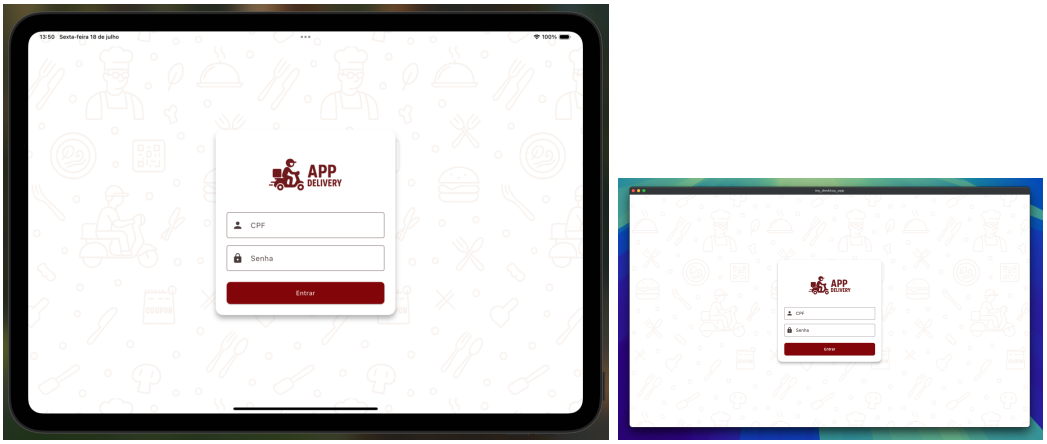


Figura 8 – Tela de login do sistema.

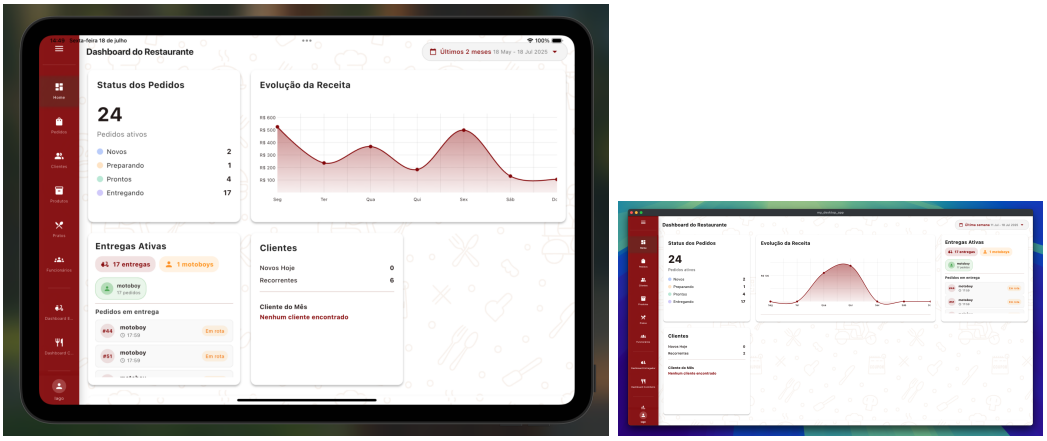


Figura 9 – Dashboard geral da aplicação.

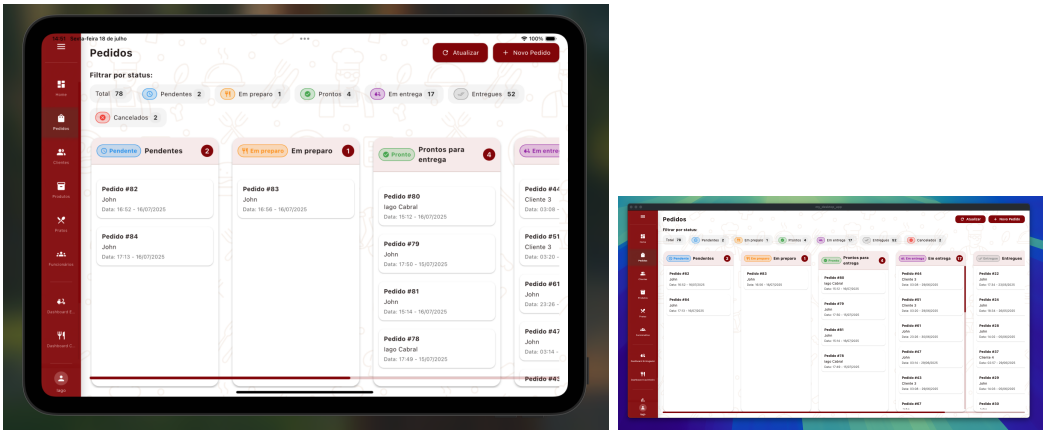


Figura 10 – Lista de pedidos realizados.

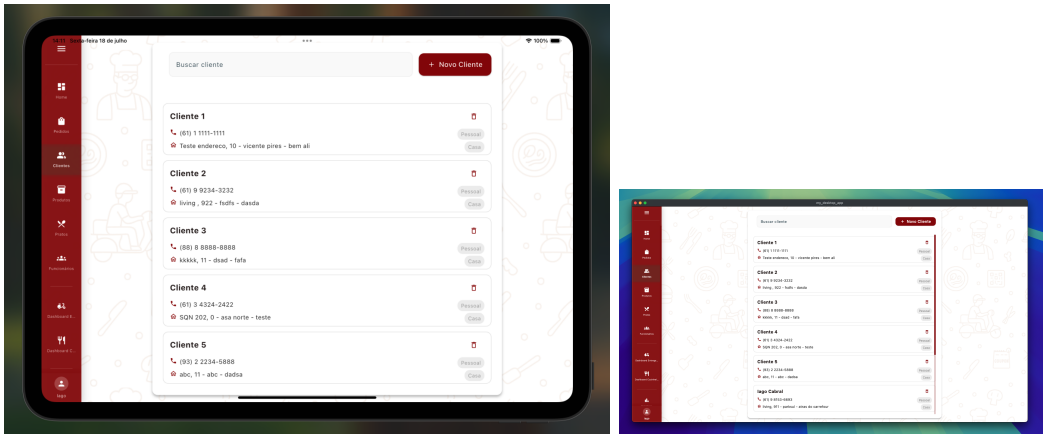


Figura 11 – Listagem de clientes cadastrados.

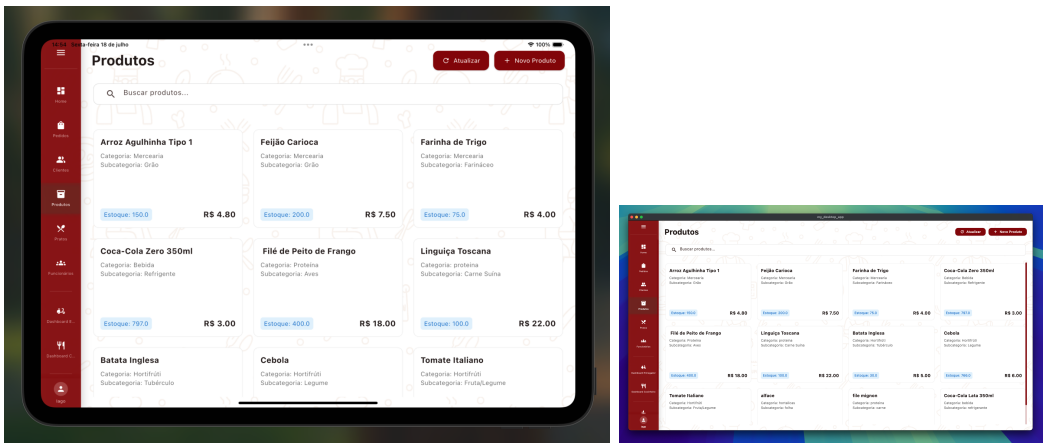


Figura 12 – Cadastro e edição de produtos.

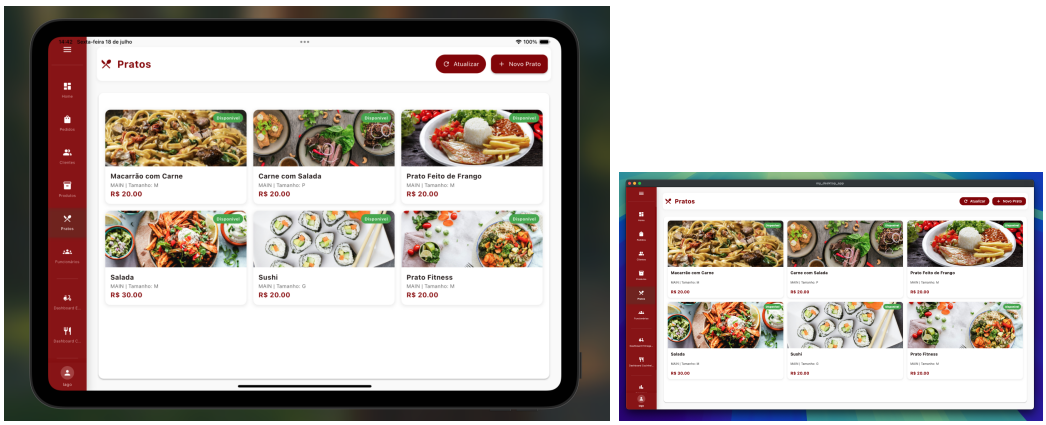


Figura 13 – Cadastro de pratos no cardápio.

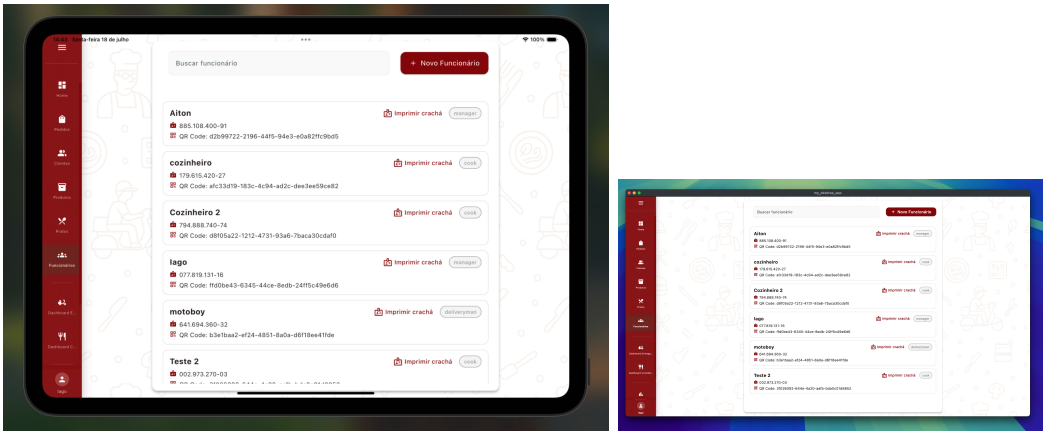


Figura 14 – Listagem de funcionários.

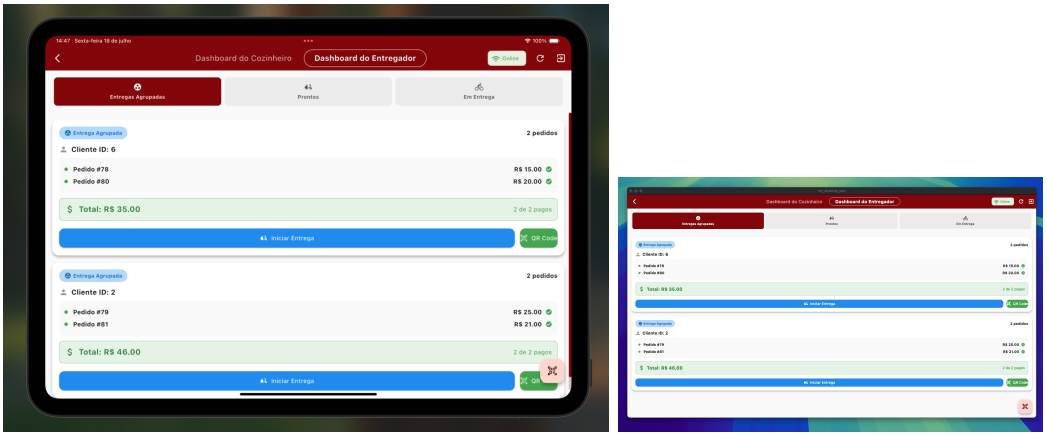


Figura 15 – Dashboard dos entregadores.

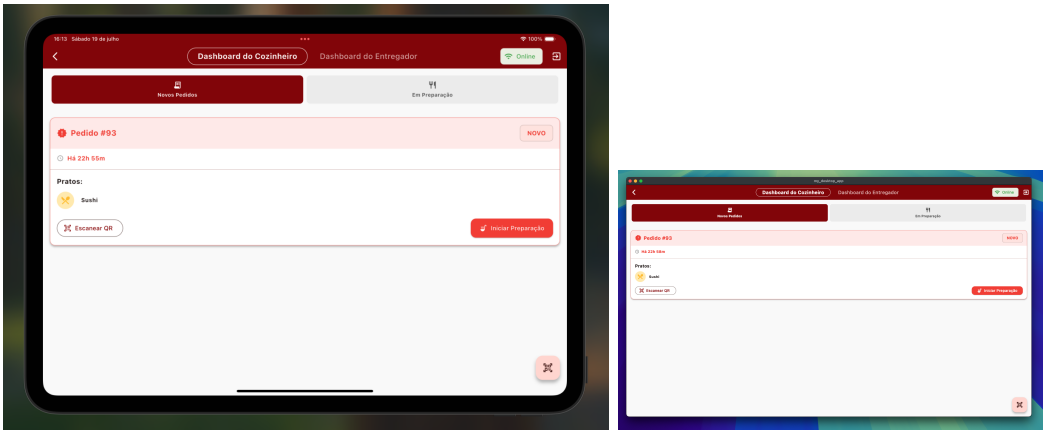


Figura 16 – Dashboard dos cozinheiros.

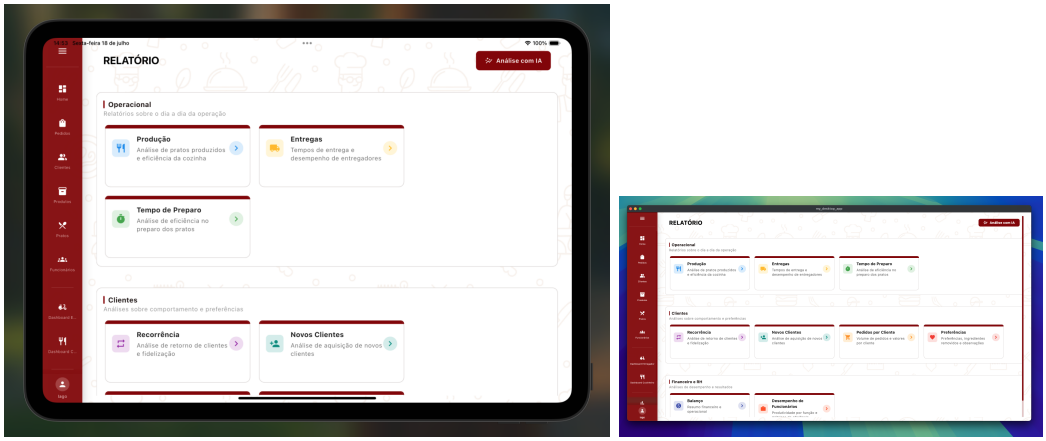


Figura 17 – Relatórios e análises.

4.4 Processos de Testes

Os processos de testes da aplicação foram divididos em três categorias principais: testes unitários, testes de integração e testes de aceitação com usuários (UAT). Conforme descrito na seção 3.6, cada uma dessas categorias tem um papel fundamental na garantia da qualidade, robustez e usabilidade do sistema desenvolvido.

4.4.0.1 Testes Unitários

Conforme detalhando na subseção 3.6.1, os testes unitários foram executados com apoio da ferramenta coverage 3.8, utilizada para mensurar a porcentagem de código coberto pelos testes. O resultado obtido foi de 86% de cobertura total.

Considerando que o projeto segue os princípios da *Clean Architecture*, os testes focaram nas camadas responsáveis pelas lógicas centrais do sistema, como casos de uso, entidades, repositórios, modelos, schemas, rotas e módulos de segurança.

A tabela 5 abaixo apresenta a cobertura dos testes unitários nas principais camadas da aplicação, com uma média de 89%. Esse valor é superior ao percentual geral de cobertura reportado pela ferramenta coverage, que foi de 86%, pois o cálculo total inclui arquivos auxiliares, como os de tratamento de exceções, que não estão diretamente ligados às regras de negócio do sistema. Os registros visuais do relatório de cobertura estão disponíveis no anexo B.

Tabela 5 – Cobertura de Testes Unitários por Camada da Aplicação

Camada / Arquivo	**Cobertura (%)**
Casos de Uso (Use Cases)	89%
Entidades	92%
Repositórios (Repos + Implementações)	86%
Models	99%
Schemas	96%
Rotas (Routers)	88%
Módulo de Segurança	83%
Cobertura Total Média (Arquivos Principais)	**89%**

Fonte: Autores (2025).

4.4.0.2 Testes de Integração

Conforme descrito na subseção 3.6.2, os testes de integração foram realizados ao longo do desenvolvimento para verificar o funcionamento conjunto dos módulos da aplicação. As validações incluíram a comunicação entre frontend e backend via API REST, bem como a interação entre o backend e o banco de dados PostgreSQL, com foco na inserção, leitura, remoção e persistência de dados.

Esses testes foram conduzidos com o apoio de arquivos específicos e também de forma manual, durante o uso e verificação prática do sistema. A seguir, os pontos validados durante o processo:

- Comunicação entre frontend e backend via API REST;
- Integração entre backend e banco de dados PostgreSQL;
- Validação de inserção, leitura e remoção de registros;
- Confirmação da persistência correta dos dados no banco;
- Funcionamento de rotas e fluxos de autenticação.

4.4.0.3 Testes de Aceitação do Usuário (UAT)

Conforme apresentado na subseção 3.6.3, os testes de aceitação foram realizados seguindo os casos de uso pré-estabelecidos do nosso projeto. A seguir, são apresentados

os resultados observados durante a execução dessas atividades.

O primeiro teste foi realizado com um usuário do sexo masculino, com aproximadamente 30 anos, que já exerceu a função de telefonista em um restaurante e tem familiaridade com computador. Este perfil foi escolhido por sua familiaridade com os fluxos típicos de atendimento ao cliente, como o cadastro de clientes e pedidos via telefone.

Os fluxos testados foram definidos com base nos casos de uso previamente mapeados no projeto, abrangendo as funcionalidades de:

- **Gerenciar clientes**: incluindo o registro e a edição de dados cadastrais;
- **Gerenciar pedidos**: contemplando o registro do pedido, adição de observações e definição da forma de pagamento;
- **Gerenciar pedidos em andamento**: visualização e acompanhamento do status dos pedidos em preparação.

Durante a execução dessas tarefas, o usuário não apresentou dificuldades nem cometeu erros, demonstrando compreensão total dos fluxos propostos. Executou todas as ações com agilidade e de forma autônoma, o que evidenciou a boa usabilidade da aplicação. Além disso, destacou de forma espontânea que gostou da interface, considerando-a clara e fácil de usar.

O segundo teste foi realizado com uma usuária do sexo feminino, com aproximadamente 40 anos, gerente de restaurante. Ela possui certa familiaridade com computadores, embora tenha relatado algumas dificuldades em tarefas específicas. Os fluxos testados também seguiram os casos de uso previamente definidos, abrangendo um escopo mais amplo de funcionalidades, incluindo:

- **Gerenciar clientes**;
- **Gerenciar cardápio**;
- **Gerenciar pedidos**;
- **Gerenciar pedidos em andamento**;
- **Gerenciar funcionários**;
- **Visualizar análises (dashboard e relatórios)**;
- **Gerenciar estoque**.

Apesar das dificuldades pontuais, a usuária conseguiu executar todas as tarefas propostas. Durante a navegação pelos relatórios, comentou que teve dificuldade em localizar certas informações, considerando essa parte da interface do relatório um pouco confusa. Também observou que, em algumas telas, não estava claro que havia mais informações disponíveis ao rolar para baixo, o que gerou incertezas durante o uso.

Ainda assim, elogiou a aplicação, destacando que está fácil de utilizar, especialmente nos fluxos em que interage diretamente com o sistema para realizar as tarefas (gerenciar clientes, gerenciar cardápio, gerenciar pedidos). Ressaltou que a experiência geral de uso foi positiva.

Esses apontamentos se mostraram valiosos para a aplicação de melhorias de usabilidade em pontos específicos do sistema, principalmente na visibilidade de informações.

4.5 Validação e Ajustes

Durante o desenvolvimento, foram realizados ajustes contínuos com base no *feedback* de usuários reais e na validação sistemática dos requisitos definidos. Essa etapa foi essencial para garantir que o sistema atenda, de fato, às necessidades do restaurante, tanto em suas funcionalidades quanto nos aspectos de qualidade.

A validação foi conduzida de duas formas complementares:

- Requisitos funcionais: validados com base em critérios de aceitação definidos previamente e testes com usuários reais.
- Requisitos não funcionais: verificados por meio de um checklist estruturado, conforme recomendações de Wiegers e Beatty (2013) (WIEGERS; BEATTY, 2013).

4.5.1 Validação dos Requisitos Funcionais

A validação dos requisitos funcionais foi orientada por critérios de aceitação, alinhando-se à prática recomendada de envolver os usuários na definição de condições claras para considerar o sistema pronto para uso. Segundo Wiegers e Beatty, critérios de aceitação ajudam a mudar a perspectiva da pergunta “O que o sistema deve fazer?” para “Como saberemos que ele atende às nossas necessidades?” (WIEGERS; BEATTY, 2013).

Na prática, esses critérios foram representados por condições específicas de sucesso associadas a cada requisito priorizado. Entre os principais critérios considerados estavam:

- Completude funcional: o sistema realiza todas as ações esperadas conforme o requisito.

- Clareza e usabilidade: o usuário consegue executar as ações com facilidade, sem necessidade de treinamento.
- Navegação sem erros: ausência de falhas ou comportamentos inesperados.
- Alinhamento com os fluxos reais de operação do restaurante.

A validação desses critérios foi feita por meio dos Testes de Aceitação do Usuário (UAT), descritos na seção 4.4.0.3, e também com base na experiência prática de um dos integrantes do grupo, que atua diretamente em um restaurante. Essa abordagem confirmou que o sistema atende às necessidades do público-alvo e permitiu ajustes relevantes na interface e na usabilidade.

Vale ressaltar que os requisitos validados foram aqueles priorizados na seção 3.3.2.2, os quais foram implementados até a classificação *could*.

4.5.2 Validação dos Requisitos Não Funcionais

Para apoiar a validação dos requisitos não funcionais, foi adotada a técnica de checklist, conforme recomendada por Wiegers e Beatty (2013) (WIEGERS; BEATTY, 2013). Essa abordagem auxilia os revisores na identificação de erros recorrentes por meio de uma lista estruturada de verificação, adaptada às necessidades do projeto.

Segundo os autores, o uso de checklists é eficaz porque destaca problemas historicamente frequentes e ajuda os avaliadores a manter um processo de validação mais consistente. Estudos indicam que a atribuição de responsabilidades específicas na detecção de falhas também aumenta a eficácia da revisão.

Com base nessa técnica, foram definidos critérios claros para avaliar requisitos de desempenho, segurança, usabilidade e outros aspectos de qualidade, conforme descrito na seção 3.3.3. A seguir, apresenta-se o resultado da verificação:

- **Desempenho**
  RNF01 – O sistema responde a funcionalidades críticas, como o registro de pedidos e atualização de status, em até 2 segundos sob uso normal?
  Sim. Durante os testes manuais, o tempo médio de resposta nessas operações permaneceu abaixo do limite definido.
- **Segurança**
  RNF02 – O sistema implementa autenticação e criptografia de dados sensíveis?
  Sim. A autenticação é feita via JWT, e senhas de usuários são armazenadas com criptografia segura no banco de dados. Além disso, alguns dados sensíveis dos clientes também estão protegidos de forma segura na base de dados.

- **Conformidade com Padrões**
  RNF03 – O sistema segue práticas e guias técnicos reconhecidos?
  Sim. Foram adotadas boas práticas de segurança, padrões de validação em formulários e mensagens amigáveis ao usuário.

- **Usabilidade**
  RNF04 – A interface é acessível, intuitiva e fácil de usar?
  Sim. Usuários conseguiram operar o sistema sem necessidade de orientação adicional, como evidenciado nos testes de aceitação.

- **Portabilidade**
  RNF05 – A aplicação funciona em diferentes sistemas operacionais modernos e dispositivos móveis?
  Sim. O sistema foi validado em navegadores *desktop*, Android, tablets e dispositivos iOS, com comportamento consistente.

- **Escalabilidade**
  RNF06 – O sistema suporta aumento de carga sem comprometer a estabilidade?
  Sim. O sistema é baseado em *Clean Architecture*, de forma que permite a escalabilidade independete de modulos e uso de Docker permite escalar o backend conforme a demanda.

- **Manutenibilidade**
  RNF07 – O código foi desenvolvido com arquitetura modular que facilita manutenção?
  Sim. A divisão em camadas (conforme a *Clean Architecture*) permite atualizações localizadas, com baixo acoplamento entre os módulos.

- **Qualidade de Serviço**
  RNF08 – O sistema garante confiabilidade, eficiência e boa experiência do usuário?
  Sim. A aplicação manteve comportamento estável durante testes e obteve *feedback* positivo dos usuários quanto à usabilidade e performance.

4.6 Segurança e Proteção de Dados

Com base nas práticas descritas na seção 3.7, durante o desenvolvimento do sistema, foram incorporadas práticas de segurança voltadas à proteção dos dados sensíveis de clientes. Entre os principais cuidados adotados, destacam-se:

- **Autenticação via JWT**: Após o login, o sistema utiliza JSON Web Token (JWT) para garantir que apenas usuários devidamente autenticados possam acessar áreas protegidas. O token contém informações codificadas sobre o usuário e uma assinatura

que impede alterações maliciosas. Ele é enviado pelo cliente em cada requisição, eliminando a necessidade de sessões persistentes (JWT.io, 2025).

- **Armazenamento de senhas com bcrypt**: As senhas dos usuários não são armazenadas diretamente. Em vez disso, são transformadas por meio do algoritmo bcrypt, que gera uma sequência embaralhada (hash) da senha. Esse processo inclui um salt, que garante que senhas iguais gerem hashes diferentes. O bcrypt também aplica o processo várias vezes para tornar ataques ainda mais difíceis, mesmo com computadores potentes (AUTH0, 2025).
- **Hash de dados sensíveis com SHA-256**: Telefones e endereços dos clientes são protegidos usando a função de hash criptográfica SHA-256, sendo armazenados de forma criptografada no banco de dados. Essa função gera uma "impressão digital" única para cada dado, de forma que qualquer alteração na informação original resulta em um hash completamente diferente. Isso permite verificar a integridade dos dados e detectar modificações indevidas. (MARTINO; CILARDO, 2020).

Essas medidas contribuem para a confiabilidade, integridade e conformidade legal da aplicação. As figuras 18, 19 e 20 ilustram exemplos reais de como os dados são armazenados de forma protegida no banco de dados.

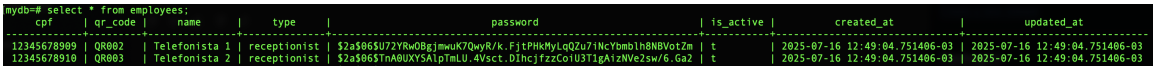


Figura 18 – Senhas criptografadas com bcrypt na tabela `employees`.

Figura 19 – Telefones de clientes criptografados e com hash.

Figura 20 – Endereços de clientes criptografados e com hash.

5 Considerações Finais

O objetivo deste trabalho, abordado na seção 1, foi desenvolver uma aplicação responsiva voltada para o gerenciamento interno de pedidos de delivery em restaurantes. A proposta buscou apoiar a organização das operações administrativas, auxiliar a tomada de decisões gerenciais e fornecer dados analíticos sobre o desempenho da equipe, contribuindo para a eficiência e a sustentabilidade dos estabelecimentos.

A pesquisa bibliográfica possibilitou a compreensão do panorama atual do delivery e das principais dificuldades operacionais enfrentadas pelos gestores. A partir dessa análise, foram identificadas lacunas e oportunidades para o desenvolvimento de uma solução voltada à automação de processos, análise de dados e integração de informações.

Para garantir que a proposta atendesse às reais necessidades do setor, foram utilizados métodos exploratórios, incluindo levantamento de requisitos por meio de observações e entrevistas com profissionais da área. A definição desses requisitos seguiu uma abordagem estruturada, priorizando funcionalidades essenciais por meio da técnica MoSCoW. Além disso, metodologias ágeis, como Scrum e Extreme Programming, foram adotadas para garantir maior flexibilidade e eficiência no desenvolvimento da solução.

O desenvolvimento teve início pela camada de *backend*, priorizando funcionalidades dependentes de banco de dados, estruturado com PostgreSQL, conforme apresentado na seção 3.5.5.1. Essa decisão teve como objetivo reduzir riscos de retrabalho decorrentes de eventuais mudanças na estrutura do banco da aplicação. O *backend* foi implementado com o framework FastAPI, também citado na seção 3.5.3.2, permitindo o desenvolvimento modular baseado em APIs específicas para cada requisito. Essa abordagem favoreceu o controle individual de funcionalidades, facilitando a verificação contínua dos requisitos e permitindo que cada módulo fosse desenvolvido de forma independente, sem interferir no funcionamento dos demais. Já o *frontend* foi desenvolvido com Flutter, conforme descrito na seção 3.5.4.2, o que viabilizou a criação de uma interface única e responsiva para múltiplas plataformas (*mobile* e *desktop*), utilizando uma única base de código.

Com base nas etapas desenvolvidas e nos recursos implementados, é possível concluir que os objetivos do trabalho foram alcançados. A aplicação proposta atende aos requisitos previamente levantados, integrando funcionalidades voltadas à organização das operações internas, análise de desempenho da equipe e apoio à tomada de decisões. Além disso, o uso de tecnologias responsivas, metodologias ágeis e a escolha do framework FastAPI, que se mostrou leve, eficiente e adequado ao desenvolvimento de APIs modulares, contribuiu para a eficiência do processo de desenvolvimento e para a adaptabilidade do sistema em diferentes contextos operacionais.

5.1 Dificuldades Encontradas

Ao longo da concepção e desenvolvimento do projeto, a equipe enfrentou uma série de desafios que foram cruciais para o amadurecimento técnico e acadêmico do trabalho. Um dos primeiros desafios manifestou-se na delimitação do escopo do projeto. Durante a fase de levantamento de requisitos, a combinação de técnicas como observação, entrevistas e brainstorming resultou em um vasto backlog de funcionalidades desejadas. O processo de filtrar essas demandas e categorizá-las através do método MoSCOW foi uma tarefa complexa, exigindo um balanço cuidadoso entre a ambição de criar uma ferramenta completa e a necessidade de definir um escopo exequível dentro do cronograma do trabalho.

Adicionalmente, na fase escrita do TCC, a busca por referências bibliográficas que embasassem a proposta de forma integrada mostrou-se desafiadora. O caráter multidisciplinar do projeto exigiu a conexão entre conceitos de Engenharia de Software, como metodologias ágeis e arquitetura de software , com teorias de gestão e dados específicos sobre o setor de delivery. Consolidar essa base teórica diversa em uma argumentação coesa demandou um esforço de pesquisa e síntese considerável.

Já no âmbito do desenvolvimento prático, a implementação da *Clean Architecture*, embora teoricamente vantajosa por promover a separação de responsabilidades, apresentou uma curva de aprendizado íngreme. A tradução do modelo conceitual para uma estrutura de projeto resultou em uma proliferação de diretórios, o que, inicialmente, dificultou a navegação e a localização de componentes específicos, tornando o fluxo de desenvolvimento menos intuitivo até a completa familiarização da equipe com a estrutura.

Por fim, outro obstáculo significativo durante o desenvolvimento foi a necessidade de realizar alterações no banco de dados relacional, foi preciso adicionar ou modificar colunas nas entidades previamente modeladas no Diagrama Entidade-Relacionamento, consumindo um tempo de desenvolvimento não previsto e reforçando a importância de uma modelagem de dados robusta desde o início.

5.2 Limitações do Estudo

Apesar dos avanços alcançados no desenvolvimento da aplicação, algumas limitações importantes foram identificadas e deverão ser abordadas em trabalhos futuros. A principal limitação refere-se à ausência de testes em um ambiente real de operação. Embora tenham sido realizados testes com pessoas envolvidas no cenário de delivery, como um telefonista e um gestor, a aplicação ainda não foi submetida a um ciclo completo de validação em um restaurante em funcionamento.

Outra limitação relevante está relacionada à integração com dispositivos de impressão térmica para a geração automática de pedidos. A funcionalidade de impressão

foi implementada utilizando uma impressora comum, devido à indisponibilidade de uma impressora térmica durante o desenvolvimento. No entanto, a impressão direta em impressoras térmicas é um recurso amplamente utilizado em operações de delivery para agilizar o repasse de pedidos à cozinha e ao setor de expedição. A ausência dessa integração pode limitar a adoção da solução em estabelecimentos que dependem desse fluxo operacional, tornando essencial o desenvolvimento e a validação dessa funcionalidade em versões futuras do sistema.

Adicionalmente, uma terceira limitação foi encontrada na compatibilidade de funcionalidades entre as diferentes plataformas visadas pelo projeto. A funcionalidade central de criação e gerenciamento de áreas de entrega, implementada com o plugin **google maps flutter web**, não pôde ser portada para a versão *desktop*. Durante o ciclo de desenvolvimento deste trabalho, o suporte oferecido pelo plugin para sistemas operacionais mostrou-se insuficiente ou instável, impossibilitando a renderização e interação com os mapas.

Adicionalmente, mesmo o plugin **google maps flutter** oferecendo suporte às SDKs de sistemas operacionais móveis, não foi possível implementar e manter seu funcionamento corretamente durante o desenvolvimento, por motivos técnicos e limitações encontradas.

Essa restrição impacta diretamente a experiência de usuários que poderiam utilizar a plataforma de gestão como aplicativo *desktop*, tornando uma funcionalidade estratégica do sistema indisponível. Para trabalhos futuros, recomenda-se o acompanhamento da evolução de novas versões do plugin ou a exploração de APIs de mapas alternativas que garantam paridade de recursos e estabilidade no ambiente de sistema operacionais.

Essas limitações evidenciam a necessidade de novas etapas de validação e aprimoramento, especialmente no que diz respeito à aplicação em cenários reais e à integração com equipamentos essenciais ao cotidiano dos restaurantes.

5.3 Trabalhos Futuros

A conclusão deste projeto abre um leque de oportunidades para futuras expansões e aprofundamentos, tanto do ponto de vista tecnológico quanto acadêmico.

Evolução da Aplicação: O próximo passo natural é enriquecer o sistema com funcionalidades do backlog que não foram priorizadas inicialmente. Isso inclui a implementação de recursos classificados como Won't, como um sistema de coleta de *feedback* de clientes.

Integrações com o Ecossistema Externo: Para aumentar o valor da ferramenta, futuras versões poderiam focar em integrações com: Plataformas de Delivery: APIs para consolidar pedidos de diferentes aplicativos (iFood, Rappi) em uma única interface.

Sistemas de Pagamento: Integração com gateways para processar pagamentos online e em máquinas de cartão.

Software de Gestão Financeira: Exportação de dados de vendas para sistemas contábeis, automatizando o fluxo financeiro.

Aprofundamento da Pesquisa: Agora que existe uma ferramenta funcional, novas pesquisas podem ser conduzidas: Estudo de Caso Longitudinal: Acompanhar a utilização do sistema em um restaurante por um período de 6 a 12 meses para medir de forma robusta os ganhos de produtividade, a redução de custos operacionais e o impacto na satisfação do cliente. Pesquisa Quantitativa: Aplicar um questionário em larga escala para validar as necessidades do setor e medir o interesse nas funcionalidades propostas e futuras, conferindo significância estatística aos achados.

Referências

ABRASEL, A. B. D. B. E. R. *Bares e restaurantes estão mais digitais na gestão e mais presentes nas redes sociais, diz pesquisa exclusiva da Abrasel*. [S.l.], 2023. Disponível em: <https://pr.abrasel.com.br/noticias/noticias/bares-e-restaurantes-estao-mais-digitais-na-gestao-e-mais-presentes-nas-redes-sociais-diz-pesquisa-e>. Citado na página 19.

ABRASEL, A. B. D. B. E. R. *O que mais vendeu no delivery em 2023*. [S.l.], 2023. Disponível em: <https://abrasel.com.br/revista/mercado-e-tendencias/vendas-delivery-2023/>. Citado 2 vezes nas páginas 19 e 28.

ALBERT, W.; TULLIS, T. *Measuring the user experience: collecting, analyzing, and presenting usability metrics*. [S.l.]: Newnes, 2013. Citado na página 23.

AMAZON. *The Difference Between Frontend and Backend*. [S.l.], 2025. Accessed: 2025-01-28. Disponível em: <https://aws.amazon.com/pt/compare/the-difference-between-frontend-and-backend/>. Citado 2 vezes nas páginas 68 e 69.

AUTH0. *Hashing in Action: Understanding bcrypt*. [S.l.], 2025. Disponível em: <https://auth0.com/blog/hashing-in-action-understanding-bcrypt/>. Citado na página 88.

AWS, A. W. S. *O que é Scrum?* [S.l.], 2025. Disponível em: <https://aws.amazon.com/pt/what-is/scrum/>. Citado na página 36.

BARBOSA, L.; ANDRADE, M. Desenvolvimento web responsivo: Benefícios e desafios na implementação. *Revista Brasileira de Tecnologia da Informação*, v. 12, n. 3, p. 45–58, 2021. Disponível em: <https://revistasbrasilti.com.br/rbti/article/view/123>. Citado 2 vezes nas páginas 20 e 47.

BARBOSA, S. D. J. et al. *Interação Humano-Computador e Experiência do Usuário*. [S.l.]: Autopublicação, 2021. Citado 2 vezes nas páginas 26 e 27.

BINDER, R. *Testing Object-Oriented Systems: Models, Patterns, and Tools*. [S.l.]: Addison-Wesley, 2000. Citado na página 72.

BLYTHE, M. et al. *Funology: From Usability to Enjoyment*. 1. ed. Dordrecht: Springer Netherlands, 2004. v. 3. (Human-Computer Interaction Series, v. 3). Citado na página 27.

BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. *The Unified Modeling Language User Guide*. [S.l.]: Pearson Education, Inc., 2005. Citado na página 53.

COHN, M. *User Stories Applied: For Agile Software Development*. [S.l.]: Addison-Wesley, 2004. Citado na página 41.

CONNECTION, F. *Food Service 2024: Fique por dentro das principais tendências para o setor*. [S.l.], 2024. Disponível em: <https://www.foodconnection.com.br/artigos/

food-service-2024-fique-por-dentro-das-principais-tendencias-para-o-setor>. Citado na página 20.

CTC, T. *O que é Extreme Programming (XP) e como ela ajuda no desenvolvimento de softwares?* [S.l.], 2024. Disponível em: <https://ctctech.com.br/blog/extreme-programming/>. Citado na página 36.

DOCKER. *What is Docker?* [S.l.], 2025. Disponível em: <https://docs.docker.com/get-started/docker-overview/>. Citado na página 38.

FASTAPI. *FastAPI Documentation*. [S.l.], 2025. Accessed: 2025-01-28. Disponível em: <https://fastapi.tiangolo.com/>. Citado na página 68.

FISCHMANN, A.; ZILBER, M. Balanced scorecard e teoria dos stakeholders: Um estudo exploratório. *Revista de Administração*, SciELO Brasil, v. 35, n. 3, p. 57–67, 2000. Citado na página 22.

FLUTTER. *Adaptive and responsive design in Flutter*. 2025. Disponível em: <https://docs.flutter.dev/ui/adaptive-responsive/>. Citado 2 vezes nas páginas 25 e 26.

GITHUB. *Entendendo o GitHub Actions*. [S.l.], 2024. Accessed: 2025-01-29. Disponível em: <https://docs.github.com/pt/actions/about-github-actions/understanding-github-actions>. Citado na página 38.

GOOGLE. *Dart Programming Language*. [S.l.], 2025. Accessed: 2025-01-28. Disponível em: <https://dart.dev/>. Citado na página 69.

GOOGLE. *Flutter Documentation*. [S.l.], 2025. Accessed: 2025-01-28. Disponível em: <https://flutter.dev/multi-platform>. Citado na página 69.

HEWETT, T. T. et al. *ACM SIGCHI Curricula for Human-Computer Interaction*. New York, NY, USA: Association for Computing Machinery, 1992. Citado na página 26.

IEEE, C. S. *Guide to the Software Engineering Body of Knowledge SWEBOK V4*. 4th edition. ed. Piscataway, NJ, USA: IEEE, 2024. Disponível em: <https://www.computer.org/education/bodies-of-knowledge/software-engineering>. Citado 2 vezes nas páginas 35 e 36.

JWT.io. *Introduction to JSON Web Tokens (JWT)*. [S.l.], 2025. Disponível em: <https://jwt.io/introduction>. Citado na página 88.

KIM, G. et al. *The DevOps Handbook: How to Create World-Class Agility, Reliability, Security in Technology Organizations*. Portland, OR: IT Revolution Press, 2016. ISBN 9781942788003. Disponível em: <https://itrevolution.com/product/the-devops-handbook/>. Citado na página 38.

KORTH, H. F.; SILBERSCHATZ, A.; SUDARSHAN, S. *Database System Concepts*. [S.l.]: McGraw-Hill Education, 2019. Citado na página 70.

LAUDON, K. C.; LAUDON, J. P. *Sistemas de informação gerenciais*. São Paulo: Pearson, 2007. Citado 2 vezes nas páginas 28 e 29.

MARTIN, R. C. *Clean Architecture: A Craftsman's Guide to Software Structure and Design*. [S.l.]: Pearson Education, 2018. Citado na página 66.

MARTINO, R.; CILARDO, A. Sha-2 acceleration meeting the needs of emerging applications: A comparative survey. *IEEE Access*, v. 8, p. 28415–28436, 2020. Disponível em: <https://ieeexplore.ieee.org/document/8986620>. Citado na página 88.

MARTINS, J. *Metodologia da Pesquisa Científica*. [S.l.]: Editora Dowbis, 2017. Citado 3 vezes nas páginas 23, 33 e 34.

MASSA, R. M. O "boom"das plataformas de delivery no brasil e suas consequências peculiares. *Portal FGV*, 2022. Disponível em: <https://portal.fgv.br/artigos/boom-plataformas-delivery-brasil-e-suas-consequencias-peculiares>. Citado na página 19.

MYERS, G. J.; BADGETT, T.; THOMAS, T. M. *The Art of Software Testing*. [S.l.]: John Wiley & Sons, 2011. Citado na página 72.

N8N, D. T. *n8n Documentation*. [S.l.], 2025. Accessed: 2025-07-07. Disponível em: <https://docs.n8n.io/#where-to-start>. Citado na página 68.

NIELSEN, J. *Usability Engineering*. San Francisco: Morgan Kaufmann, 1994. Citado 2 vezes nas páginas 27 e 29.

POSTGRESQL, G. D. G. *PostgreSQL Official Documentation*. [S.l.], 2025. Accessed: 2025-01-28. Disponível em: <https://www.postgresql.org/about/>. Citado na página 69.

Presidência da República. *Lei nº 13.709, de 14 de agosto de 2018. Dispõe sobre a proteção de dados pessoais e altera a Lei nº 12.965, de 23 de abril de 2014 (Marco Civil da Internet)*. [S.l.], 2018. Acesso em: 9 fev. 2025. Disponível em: <http://www.planalto.gov.br/ccivil_03/_ato2015-2018/2018/lei/L13709.htm>. Citado na página 73.

PYTEST, D. T. *pytest Documentation*. [S.l.], 2025. Accessed: 2025-01-28. Disponível em: <https://docs.pytest.org/en/stable/#>. Citado na página 68.

PYTHON, S. F. *About Python*. [S.l.], 2025. Accessed: 2025-01-28. Disponível em: <https://www.python.org/about/>. Citado na página 68.

REDHAT. *O que é CI/CD?* [S.l.], 2024. Accessed: 2025-01-29. Disponível em: <https://www.redhat.com/pt-br/topics/devops/what-is-ci-cd>. Citado na página 38.

SABINO, R. *Kanban: o que é, o Método Kanban, principais conceitos e como funciona no dia a dia*. [S.l.], 2023. Disponível em: <https://www.alura.com.br/artigos/metodo-kanban?srsltid=AfmBOoqrEpNiHLjAfOaN9y-2eEp59G7nw1VGEyaszmyiPHe5mhIOp58x>. Citado na página 36.

SAE-UNB, S. de A. E. *Modelagem de Banco de Dados*. [S.l.], 2025. Disponível em: <https://sae.unb.br/cae/conteudo/unbfga/sbd/new_bancomodelagem.html>. Citado na página 71.

SEBRAE, S. B. de Apoio às Micro e P. E. *O Estágio da Transformação Digital nas Pequenas e Médias Empresas*. [S.l.], 2023. Disponível em: <https://sebrae.com.br/sites/PortalSebrae/artigos/o-estagio-da-transformacao-digital-nas-pequenas-e-medias-empresas%2C4fc28c180dfc5810VgnVCM1000001b00320aRCRD>. Citado na página 28.

SLACK, N.; BRANDON-JONES, A.; JOHNSTON, R. *Operations Management*. London: Pearson, 2010. Citado na página 28.

SOMMERVILLE, I. *Software Engineering*. [S.l.]: Pearson Education, 2007. Citado na página 72.

STANDARDIZATION, I. O. for. *ISO 9241-210:2019 Ergonomics of human-system interaction — Part 210: Human-centred design for interactive systems*. [S.l.], 2019. Disponível em: <https://www.iso.org/standard/77520.html>. Citado na página 27.

STANFORD, U. *User Acceptance Testing (UAT)*. [S.l.], 2025. Disponível em: <https://uit.stanford.edu/pmo/UAT>. Citado na página 73.

STATISTA. *Online Food Delivery - Worldwide: Market Forecast and Trends*. [S.l.], 2024. Disponível em: <https://www.statista.com/outlook/emo/online-food-delivery/worldwide>. Citado na página 20.

UNITY. *O que é CI/CD?* [S.l.], 2024. Accessed: 2025-01-29. Disponível em: <https://unity.com/pt/topics/what-is-ci-cd>. Citado na página 38.

VAZQUEZ, C. E.; SIMõES, G. S. *Engenharia de Requisitos: Software Orientado ao Negócio*. [S.l.]: Brasport Livros e Multimídia Ltda., 2016. Citado 2 vezes nas páginas 42 e 52.

WIEGERS, K.; BEATTY, J. *Software Requirements*. Redmond, WA: Microsoft Press, 2013. Citado 6 vezes nas páginas 40, 42, 48, 49, 85 e 86.

WISER. *Análise de dados: transforme dados em decisões estratégicas*. [S.l.], 2024. Disponível em: <https://wisertecnologia.com.br/blog/analise-de-dados-transforme-dados-em-decisoes-estrategicas/>. Citado na página 29.

Apêndices

APÊNDICE A – Links de Referência do Benchmarking

Este apêndice apresenta os links e observações utilizadas como base para a análise comparativa realizada na tabela 2. As informações estão organizadas por funcionalidade avaliada.

1. Agrupamento automático inteligente de pedidos por proximidade

- **Consumer**: Agrupamento feito manualmente pelo operador. <https://ajuda.programaconsumer.com.br/como-funciona-o-smart-delivery-do-programa-cons>

- **iFood**: Não possui link oficial. Observado por testes próprios.

2. Notificações em tempo real com status detalhado do pedido

- **Consumer**: Envio de notificações automatizadas por canal digital (ex: WhatsApp). <https://atualizacoes.consumer.com.br/pdv/anteriores/15.0.0/15.0.0.3-beta/nova-funcionalid

- **iFood**: Notificações automáticas ativadas via plataforma. <https://comunidade.ifood.com.br/t/gerencie-as-notificacoes-do-ifood/119>

3. Registro completo de clientes (dados detalhados e históricos)

- **Consumer**: Cadastro permite histórico de pedidos, múltiplos contatos e endereços. <https://ajuda.programaconsumer.com.br/como-cadastrar-clientes-no-programa-consumer/>

- **iFood**: O iFood mantém os dados em sua própria plataforma e não fornece acesso completo aos estabelecimentos. Não possui link oficial. Observado por testes próprios.

4. Preenchimento automático de taxas de entrega por zonas

- **Consumer**: Preenche o campo de frete, em pedido de forma automática. <https://ajuda.programaconsumer.com.br/como-cadastrar-a-area-de-entrega-por-bairros-faixa-de-
>
- **iFood**: Não possui link oficial. Observado por testes próprios.

5. Geração de relatórios de toda equipe envolvida no delivery

- **Consumer**: Não há detalhamento por função na equipe. <https://ajuda.programaconsumer.com.br/quais-relatorios-estao-presentes-no-consumer-connect/
>
- **iFood**: Não possui link oficial. Observado por testes próprios.

6. Geração de relatórios de pedidos

- **Consumer**: Relatório com dados de pedidos. <https://ajuda.programaconsumer.com.br/quais-relatorios-estao-presentes-no-consumer-connect/
>
- **iFood**: Relatório com dados de pedidos. <https://www.youtube.com/watch?v=wTKvfpVv8Pk&t=25s>

7. Geração de relatórios de vendas

- **Consumer**: Relatórios detalhados por período, formas de pagamento e produtos vendidos. <https://ajuda.programaconsumer.com.br/quais-relatorios-estao-presentes-no-consumer-connect/
>
- **iFood**: Relatórios de repasses e vendas disponíveis no painel do restaurante. <https://www.youtube.com/watch?v=wTKvfpVv8Pk&t=25s>

8. Assistente baseado em IA para responder perguntas usando dados do restaurante

- **Consumer**: Não possui link oficial. Observado por testes próprios.

- **iFood**: Não possui link oficial. Observado por testes próprios.

9. Design responsivo (uso eficiente tanto em *desktop* quanto em celulares)

- **Consumer**: Não possui link oficial. Observado por testes próprios.
- **iFood**: Não possui link oficial. Observado por testes próprios.

APÊNDICE B – Requisitos do Sistema - Observação

B.1 backlog dos Requsitos por Observação

Épico - Clientes (E01)

Descrição: O sistema deve ser capaz de gerenciar os registros de clientes.

F01 - Registro de Clientes

- **R01 - Registrar Nome**
  Descrição: O sistema deve ser capaz de registrar o nome dos clientes.
- **R02 - Registrar Telefone**
  Descrição: O sistema deve ser capaz de registrar o telefone dos clientes.
- **R03 - Registrar Endereço**
  Descrição: O sistema deve ser capaz de registrar o endereço dos clientes.
- **R04 - Editar Registro**
  Descrição: O sistema deve ser capaz de editar os registros de clientes.

Épico - Pedidos (E02)

Descrição: O sistema deve ser capaz de gerenciar os pedidos e o fluxo de preparação.

F02 - Registro de Pedidos

- **R01 - Registrar Pedido**
  Descrição: O sistema deve ser capaz de registrar os pedidos dos clientes.
- **R02 - Registrar Observações do Pedido**
  Descrição: O sistema deve ser capaz de registrar observações adicionais para os pedidos.
- **R03 - Destacar Observações na Nota**
  Descrição: O sistema deve destacar as observações do pedido na nota fiscal.

- **R04 - Resumo do Pedido**
  Descrição: O sistema deve gerar um resumo do pedido.
- **R05 - Forma de Pagamento**
  Descrição: O sistema deve registrar a forma de pagamento do cliente.
- **R06 - Notificar Cliente pelo WhatsApp**
  Descrição: O sistema deve informar o cliente sobre o pedido via WhatsApp.
- **R07 - Editar Pedido**
  Descrição: O sistema deve permitir a edição de pedidos registrados.
- **R08 - Registrar Produtos**
  Descrição: O sistema deve ser capaz de registrar os produtos incluídos no pedido.
- **R09 - Registrar Pratos (Pratos do Cardápio)**
  Descrição: O sistema deve ser capaz de registrar os pratos do cardápio.
- **R10 - Notificar Cliente pelo WhatsApp**
  Descrição: O sistema deve notificar o cliente sobre o status do pedido via WhatsApp.

F03 - Preparação de Pedidos

- **R01 - Notificar Cozinha**
  Descrição: O sistema deve notificar a cozinha sobre novos pedidos.
- **R02 - Ler QR Code do Cozinheiro**
  Descrição: O sistema deve permitir a leitura do QR Code do cozinheiro.
- **R03 - Vincular Cozinheiro ao Pedido**
  Descrição: O sistema deve vincular um cozinheiro a um pedido.
- **R04 - Atualizar Status do Pedido via Leitura de QR Code**
  Descrição: O sistema deve atualizar o status do pedido durante a preparação utilizando o QR Code.
- **R05 - Atualizar Status do Pedido para Estabelecimento**
  Descrição: O sistema deve atualizar o status do pedido durante a preparação.
- **R06 - Avisar Motoboy**
  Descrição: O sistema deve avisar o motoboy quando o pedido estiver pronto.
- **R07 - Permitir Atualização Manual de Status pelo Gestor**
  Descrição: O sistema deve permitir que o gestor atualize manualmente o status do pedido.

- **R08 - Atualizar Status do Pedido para Cliente**
  Descrição: O sistema deve atualizar o status do pedido para o cliente

- **R09 - Imprimir Pedido**
  Descrição: O sistema deve permitir a impressão do pedido para conferência.

- **R10 - Avisar Motoboy**
  Descrição: O sistema deve avisar o motoboy quando o pedido estiver pronto.

Épico - Entrega (E03)

Descrição: O sistema deve ser capaz de gerenciar as entregas e motoboys.

F04 - Organização de Entregas

- **R01 - Definir Limite de Itens por Região de Entrega**
  Descrição: O sistema deve permitir a definição de um limite de itens por região de entrega.

- **R02 - Organizar Pedidos por Localidade**
  Descrição: O sistema deve organizar os pedidos por localidade para otimizar as entregas.

- **R03 - Gerenciar Áreas de Entrega**
  Descrição: O sistema deve permitir o gerenciamento das áreas de entrega.

- **R04 - Calcular Frete Automaticamente com Base na Região**
  Descrição: O sistema deve calcular o frete automaticamente com base na região de entrega.

- **R05 - Definir Limite de Itens por Região de Entrega**
  Descrição: O sistema deve permitir a definição de um limite de itens por região de entrega.

F05 - Motoboy

- **R01 - Ler QR Code do Motoboy**
  Descrição: O sistema deve permitir a leitura do QR Code do motoboy.

- **R02 - Informar Pedidos ao Motoboy**
  Descrição: O sistema deve informar ao motoboy os pedidos que ele deve entregar.

- **R03 - Vincular Pedido ao Motoboy**
  Descrição: O sistema deve vincular um pedido a um motoboy.

- **R04 - Calcular Tempo de Trânsito da Entrega via API**
  Descrição: O sistema deve calcular o tempo de trânsito da entrega utilizando uma API de mapas.

- **R05 - Notificar Cliente sobre Entrega**
  Descrição: O sistema deve notificar o cliente sobre o status da entrega.

- **R06 - Atualizar Status da Entrega**
  Descrição: O sistema deve permitir a atualização do status da entrega.

Épico - Funcionários (E04)

Descrição: O sistema deve ser capaz de gerenciar os registros de funcionários.

F06 - Registros de Funcionários

- **R01 - Registrar Motoboy**
  Descrição: O sistema deve registrar informações dos motoboys.

- **R02 - Gerar QR Code do Motoboy**
  Descrição: O sistema deve gerar QR Codes para os motoboys.

- **R03 - Registrar Cozinheiro**
  Descrição: O sistema deve registrar informações dos cozinheiros.

- **R04 - Gerar QR Code do Cozinheiro**
  Descrição: O sistema deve gerar QR Codes para os cozinheiros.

APÊNDICE C – Requisitos do Sistema - Entrevista 1

Anotações da Entrevista – Pontos Principais

1. **Que tarefas você realiza diariamente no sistema atual? Há algo que gostaria de melhorar?**

   - **Tarefas Diárias:**
     - Planejamento de rotas: trânsito, proximidade, otimização.
     - Supervisão de pedidos: preparo, verificação, saída.
     - Gerenciamento de imprevistos: atrasos, erros, alterações.
     - Monitoramento: relatórios de produtividade.
     - Comunicação: reuniões rápidas com a equipe.

2. **(Pergunta improvisada) Como você verifica se os pedidos estão corretos e completos? E a saída dos entregadores?**

   - **Verificação dos Pedidos:**
     - Conferência manual: comparação com comandas.
     - Etiquetas: identificação do cliente e dos itens.
     - Checklist: conferência de itens adicionais.
   - **Supervisão da Saída:**
     - Organização: rotas agrupadas por ordem de entrega.
     - Confirmação: entregadores verificam os pedidos.
     - Orientação final: reforço de prazos e solicitações.

3. **(Pergunta improvisada) Como você resolve problemas como atrasos, pedidos incorretos ou mudanças de última hora nos endereços?**

   - **Problemas e Soluções Atuais:**
     - Atrasos: monitoramento, redistribuição de entregas.
     - Pedidos incorretos: correção rápida e treinamento preventivo.
     - Mudanças de endereço: replanejamento e rastreamento.

4. **(Pergunta improvisada) Como você analisa o que causou o erro e reforça o treinamento da equipe?**

   - **Processo de Análise:**
     - Revisão do processo: coleta de informações e falhas.
     - Registro de padrões: histórico de erros recorrentes.
   - **Treinamento:**
     - Feedback imediato e simulações práticas.
     - Procedimentos atualizados: etiquetas e checklists.

5. **(Pergunta improvisada) Como funciona sua avaliação de produtividade, identificação de gargalos e medição da satisfação dos clientes?**

   - **Produtividade:**
     - Monitoramento manual: tempo e número de tarefas.
     - Relatórios para identificar suporte necessário.
   - **Gargalos:**
     - Análise do fluxo: redistribuição de tarefas.
     - Indicadores: atrasos e picos de pedidos.
   - **Satisfação:**
     - Feedback direto: comentários e reclamações.
     - Análise de padrões: satisfação e taxa de recorrência.

6. **Quais funcionalidades ou ferramentas você sente falta para otimizar e automatizar seu trabalho?**

   - **Funcionalidades em Falta:**
     - Sistema Integrado: Registro, acompanhamento e atualização de pedidos em tempo real.
     - Rotas Automáticas: Planejamento otimizado considerando trânsito e prioridades.
     - Rastreamento em Tempo Real: Monitoramento de entregadores e status para clientes.
     - Relatórios Automáticos: Dados consolidados sobre produtividade e tempos médios.
     - Registro de Funcionários: Identificação de responsáveis por cada etapa do pedido.
     - Gestão de Feedback: Ferramenta para coletar e organizar avaliações.

  - Notificações Automatizadas: Atualizações automáticas para clientes.
  - Pagamentos Integrados: Solução que associe pagamentos diretamente ao sistema.

7. **Quais informações são essenciais para você no momento de tomar decisões?**

   - **Informações Necessárias:**
     - Status dos pedidos: andamento, atrasos, erros.
     - Desempenho da equipe: tempos, erros e feedbacks.
     - Satisfação do cliente: avaliações e recorrência.
     - Logística: rotas eficientes e custos.
     - Indicadores financeiros: receitas, custos e margem.

8. **Por que os gestores têm dificuldade em acompanhar métricas ou indicadores?**

   - **Dificuldades Identificadas:**
     - Falta de ferramentas adequadas: ausência de automação.
     - Sobrecarga de tarefas: foco em atividades operacionais.
     - Falhas de padronização: dados dispersos ou incompletos.
     - Resistência da equipe: desconforto com monitoramento.

9. **Por que as ferramentas externas são necessárias para complementar o sistema?**

   - **Razões Identificadas:**
     - Funcionalidades ausentes: rotas, rastreamento e produtividade.
     - Integração de dados: feedbacks e monitoramento externo.
     - Automação: relatórios e notificações automáticas.
     - Economia de custos: soluções específicas mais acessíveis.

10. **Por que você acha que alguns processos são mais demorados ou propensos a erros?**

    - **Causas Principais:**
      - Falta de automação: processos manuais demorados.
      - Comunicação falha: instruções incompletas ou confusas.
      - Sobrecarga em horários de pico: maior propensão a erros.

  - Treinamento insuficiente: dificuldades com tarefas complexas.

11. **Quem são as pessoas envolvidas no processo de pedidos?**

    - **Pessoas Envolvidas:**
      - **Atendentes:** Registram pedidos, garantem informações corretas e comunicam alterações. **Ponto Crítico:** Erros no registro.
      - **Cozinheiros:** Preparam pratos e informam atrasos ou faltas. **Ponto Crítico:** Retrabalho por erros.
      - **Embaladores:** Conferem e embalam pedidos. **Ponto Crítico:** Faltas ou embalagens inadequadas.
      - **Entregadores:** Conferem pedidos e seguem rotas planejadas. **Ponto Crítico:** Atrasos ou descuidos.
      - **Gerente:** Supervisiona, resolve problemas e garante a experiência do cliente. **Ponto Crítico:** Gargalos por falhas gerenciais.
      - **Clientes:** Fornecem informações e reportam problemas. **Ponto Crítico:** Informações incorretas.
      - **Suporte Técnico:** Mantém sistemas e ferramentas. **Ponto Crítico:** Falhas técnicas.
    - **Interações:** Colaboração entre equipes é essencial para evitar erros e atrasos.

12. **Quem é o seu melhor funcionário? E por quê?**

    - **Características do Melhor Funcionário:**
      - Proatividade: Antecipar problemas e buscar soluções rapidamente.
      - Consistência: Trabalho de qualidade constante.
      - Colaboração: Apoia colegas em momentos de necessidade.
      - Boa comunicação: Clareza e eficiência no diálogo com a equipe.
      - Resolução de problemas: Corrige erros com calma e rapidez.
    - **Importância:**
      - Reduz problemas operacionais e melhora o ambiente de trabalho.
      - Inspira a equipe e promove uma cultura de excelência.

13. **Onde você costuma acessar o sistema de gerenciamento? (Ex.: *mobile*, *desktop*, ambos)**

    - **Acesso ao Sistema:**
      - Desktop: Preferido para tarefas detalhadas, como análise de relatórios e planejamento.

  - Mobile: Usado para ações rápidas, como verificar pedidos e acompanhar entregas.

14. **Quando seria mais útil receber relatórios sobre o desempenho da equipe ou operações?**

    - **Momentos Ideais:**
      - **Início do Dia:** Planejar o turno com base no dia anterior.
      - **Fim do Dia:** Consolidar operações e analisar resultados.
      - **Semanalmente:** Identificar tendências e ajustar estratégias.
      - **Após Horários de Pico:** Avaliar gargalos e corrigir problemas.
      - **Mensalmente:** Revisar desempenho financeiro e definir metas.
    - **Características Desejadas:** Atualização em tempo real e consolidação diária/mensal.

15. **Como você gerencia os pedidos atualmente? Há passos manuais no processo?**

    - **Processo Atual:**
      - **Recebimento:** Registro manual, sujeito a erros.
      - **Preparo:** Comunicação manual com a cozinha.
      - **Conferência:** Verificação e embalagem feitas manualmente.
      - **Entrega:** Rotas não otimizadas e sem rastreamento.
      - **Feedback:** Solicitação manual, com baixa resposta.
    - **Desafios:** Processos manuais aumentam tempo e erros; falta de integração e automação.

16. **Como você verifica o desempenho da equipe ou acompanha os pedidos?**

    - **Desempenho da Equipe:**
      - Observação diária, relatórios simples e feedback de clientes.
      - Supervisão presencial e reuniões semanais.
    - **Acompanhamento dos Pedidos:**
      - Verificação manual do status e comunicação direta com a equipe.
      - Planejamento de rotas e confirmação com clientes após entrega.
    - **Desafios:**
      - Processo manual e ausência de sistema centralizado.
      - Monitoramento em tempo real limitado.

17. **Como as tarefas poderiam ser simplificadas ou otimizadas no sistema atual?**

    - **Centralizar Pedidos:** Sistema único para integrar todos os canais.
    - **Rotas Automáticas:** Planejamento com base em trânsito e urgência.
    - **Automação de Conferência:** Checklists digitais e QR codes.
    - **Rastreamento em Tempo Real:** Monitoramento de entregas automatizado.
    - **Feedback Integrado:** Solicitação automática de avaliações.
    - **Relatórios Automáticos:** Consolidação de métricas rapidamente.
    - **Comunicação Simplificada:** Notificações automáticas entre equipes.
    - **Treinamento Padronizado:** Checklists digitais para consistência.
    - **Priorização em Picos:** Automação para demandas urgentes.

18. **Como você organiza os dados ou relatórios para tomada de decisão?**

    - **Categorias:** Pedidos, entregas, equipe, financeiro e clientes.
    - **Ferramentas:** Google Sheets para relatórios; WhatsApp/e-mail para compartilhamento.
    - **Relatórios:**
        - Diários: Resolução de problemas pontuais.
        - Semanais: Análise de tendências e desempenho.
    - **Melhorias Desejadas:**
        - Sistema automatizado e dashboards interativos.
        - Alertas automáticos para atrasos ou falhas.

19. **Quantos erros ou problemas relacionados a pedidos acontecem em uma semana, aproximadamente?**

    - **Média semanal:** 5 a 10 problemas.
    - **Tipos e Frequência:**
        - **Registro do Pedido:** 2-3 (ex.: itens errados, falta de especificações).
        - **Preparo:** 1-2 (ex.: itens esquecidos ou errados).
        - **Entrega:** 2-3 (ex.: atrasos, trocas, problemas com endereço).
        - **Mudanças de Última Hora:** 1-2 (ex.: alteração de endereço ou itens).
        - **Pagamentos:** 1 ocasional (ex.: falhas ou despreparo do cliente).
    - **Picos de Problemas:** Durante horários de pico (almoço/jantar) e finais de semana.

- **Impacto:**
  - Insatisfação do cliente.
  - Retrabalho e atrasos.

C.1 Insights e Melhorias Sugeridas

- **Análise de Desempenho:** Relatórios automáticos com métricas consolidadas e monitoramento em tempo real para decisões rápidas.
- **Automação:** Relatórios, monitoramento automáticos e planejamento de rotas.
- **Capacitação da Equipe:** Procedimentos padronizados, como etiquetas e checklists, além de treinamento contínuo e uso de simulações práticas.
- **Centralização:** Implementação de sistemas integrados para registro, rastreamento e análise de dados com status em tempo real.
- **Centralização e Automação:** Sistemas integrados e automáticos reduziriam erros e retrabalhos, aumentando a eficiência.
- **Comunicação:** Ferramentas integradas podem melhorar colaboração e rastreabilidade.
- **Foco no Cliente:** Ferramentas para rastreamento em tempo real e coleta de feedbacks, além de notificações automáticas para atualização de status.
- **Impacto Geral:** Processos mais rápidos, confiáveis e escaláveis, mesmo com alta demanda.
- **Melhoria Desejada:** Sistema responsivo e integrado para ambos os dispositivos.
- **Melhoria na Experiência do Cliente:** Rastreamento em tempo real, notificações automáticas e gestão de feedback otimizariam a comunicação e a satisfação.
- **Operações Mais Ágeis:** Rotas automáticas evitariam atrasos.

APÊNDICE D – Requisitos do Sistema - Entrevista 2

Anotações da Entrevista – Pontos Principais

1. **Que tarefas você realiza diariamente no sistema atual? Há algo que gostaria de melhorar?**
    - Registro de pedidos.
    - Acompanhamento do fluxo e desempenho da equipe.
    - Necessidade de maior integração, automação de atualizações e relatórios automáticos.
    - O aplicativo pode oferecer um painel centralizado e intuitivo, com automação para atualização de status e registro eficiente.
2. **Quais funcionalidades ou ferramentas você sente falta para otimizar e automatizar seu trabalho?**
    - Relatórios gerenciais automatizados e acessíveis.
    - Notificações de atrasos.
    - Painel de controle em tempo real.
    - Coleta e organização de feedback dos clientes.
    - Acesso eficiente via dispositivos móveis.
3. **Quais informações são essenciais para você no momento de tomar decisões?**
    - Indicadores de desempenho da equipe.
    - Satisfação dos clientes.
    - Relatórios de vendas e previsões de picos de demanda.
    - Dados financeiros básicos, apresentados em dashboards gráficos.
4. **Por que os gestores têm dificuldade em acompanhar métricas ou indicadores de desempenho?**
    - Falta de integração e automação.

- Sistemas pouco intuitivos.
- Processos manuais demorados e dados não consolidados em tempo real.

5. **Por que as ferramentas externas que você usa (se houver) são necessárias para complementar o sistema?**
   - Complementam lacunas do sistema atual, como relatórios e feedbacks.
   - Tornam o gerenciamento mais fragmentado e trabalhoso.
6. **Por que você acha que alguns processos são mais demorados ou propensos a erros?**
   - Falta de integração e monitoramento em tempo real.
   - Dependência de tarefas manuais e sistemas pouco amigáveis.
7. **Quem são as pessoas envolvidas no processo de pedidos?**
   - Atendentes, equipe de cozinha, entregadores, clientes e o gerente.
8. **Quem é o seu melhor funcionário? E por quê?**
   - João, atendente, destaca-se por precisão, agilidade, comunicação clara e flexibilidade.
   - O aplicativo pode incluir métricas individuais de desempenho e reconhecimento para motivação.
9. **Onde os problemas mais comuns ocorrem no fluxo de pedidos?**
   - Registro incorreto de informações.
   - Comunicação com a cozinha.
   - Organização de entregas.
10. **Onde você costuma acessar o sistema de gerenciamento? (Ex.: *mobile*, *desktop*, ambos)**
    - Principalmente via *desktop*.
    - Acesso móvel ineficiente é uma limitação importante.
11. **Quando seria mais útil receber relatórios sobre o desempenho da equipe ou operações?**
    - Relatórios diários (final do dia) e semanais (início da semana).
    - O aplicativo pode programar relatórios automáticos com envio por e-mail ou notificação.

12. **Como você gerencia os pedidos atualmente? Há passos manuais no processo?**
    - Registro manual de pedidos, atualização de status e repasse de informações.
    - Gargalos e alta demanda de tempo.
    - O aplicativo deve automatizar essas etapas para reduzir o retrabalho.
13. **Como você verifica o desempenho da equipe ou acompanha os pedidos?**
    - Acompanhamento por observação direta e consolidação manual de dados.
    - O aplicativo pode oferecer relatórios em tempo real e dashboards de desempenho.
14. **Como as tarefas poderiam ser simplificadas ou otimizadas no sistema atual?**
    - Automação, integração e relatórios simplificados são as maiores necessidades.
    - Relatórios automáticos e melhor usabilidade para dispositivos móveis.
    - O aplicativo poderia ter funcionalidades intuitivas.
15. **Como você organiza os dados ou relatórios para tomada de decisão?**
    - Consolidação manual em planilhas, o que é demorado e sujeito a erros.
16. **Quantos erros ou problemas relacionados a pedidos acontecem em uma semana, aproximadamente?**
    - Entre 5 e 10 erros, envolvendo atrasos, registros incorretos, trocas de pedidos e comunicação falha.

D.1 Insights e Melhorias Sugeridas

- **Acesso ao Sistema:** Melhorar a experiência em dispositivos móveis e *desktop* para permitir acesso eficiente, independentemente do local.
- **Análise de Desempenho:** Implementação de relatórios automáticos com métricas consolidadas e monitoramento em tempo real, facilitando decisões rápidas.
- **Automação:** Automatizar relatórios, atualizações de status para reduzir trabalho manual e aumentar a eficiência.
- **Capacitação da Equipe:** Introduzir métricas individuais de desempenho, reconhecimento e treinamento contínuo para elevar a produtividade.

- **Centralização:** Integrar o controle de pedidos e relatórios em um sistema centralizado com status em tempo real.
- **Comunicação:** Melhorar a interação entre cozinha, entregadores e gerentes com ferramentas integradas, reduzindo falhas na transição de pedidos.
- **Dificuldades com Métricas:** Resolver a falta de integração e automação que torna a análise de métricas lenta e sujeita a erros.
- **Foco no Cliente:** Fornecer rastreamento em tempo real e coleta de feedback para melhorar a satisfação do cliente.
- **Gestão de Pedidos:** Automatizar o registro, atualização e acompanhamento dos pedidos para evitar gargalos e economizar tempo.
- **Impacto Geral:** Promover operações mais rápidas, confiáveis e escaláveis, mesmo em períodos de alta demanda.
- **Melhoria Desejada:** Desenvolver um sistema responsivo e integrado para dispositivos móveis e *desktop*.
- **Melhoria na Experiência do Cliente:** Incorporar rastreamento em tempo real, notificações automáticas e ferramentas de feedback para otimizar a comunicação e satisfação.
- **Organização de Dados:** Substituir planilhas manuais por relatórios automáticos e dashboards gráficos que consolidem informações críticas.
- **Problemas no Fluxo de Pedidos:** Identificar e corrigir problemas frequentes na comunicação entre setores e no registro de informações.
- **Relatórios e Frequência:** Relatórios diários para correções imediatas e semanais para planejamento estratégico, com envio automatizado por e-mail ou notificações.
- **Tarefas Simplificadas:** Automatizar e integrar processos para evitar retrabalho e aumentar a produtividade geral.

APÊNDICE E – Requisitos do Sistema - Brainstorm

E.1 Funcionalidades e Requisitos

Épico - Clientes (E01)

Descrição: O sistema deve ser capaz de gerenciar os registros de clientes.

F01 - Registro de Clientes

- **R01 - Registrar Nome**
  Descrição: O sistema deve ser capaz de registrar o nome dos clientes.
- **R02 - Registrar Telefone**
  Descrição: O sistema deve ser capaz de registrar o telefone dos clientes.
- **R03 - Registrar Endereço**
  Descrição: O sistema deve ser capaz de registrar o endereço dos clientes.
- **R04 - Editar Registro**
  Descrição: O sistema deve ser capaz de editar os registros de clientes.

Épico - Pedidos (E02)

Descrição: O sistema deve ser capaz de gerenciar os pedidos e o fluxo de preparação.

F02 - Registro de Pedidos

- **R01 - Registrar Pedido**
  Descrição: O sistema deve ser capaz de registrar os pedidos dos clientes.
- **R02 - Registrar Observações do Pedido**
  Descrição: O sistema deve ser capaz de registrar observações adicionais para os pedidos.
- **R03 - Destacar Observações na Nota**
  Descrição: O sistema deve destacar as observações do pedido na nota fiscal.

- **R04 - Resumo do Pedido**
  Descrição: O sistema deve gerar um resumo do pedido.
- **R05 - Forma de Pagamento**
  Descrição: O sistema deve registrar a forma de pagamento do cliente.
- **R06 - Notificar Cliente pelo WhatsApp**
  Descrição: O sistema deve informar o cliente sobre o pedido via WhatsApp.
- **R07 - Editar Pedido**
  Descrição: O sistema deve permitir a edição de pedidos registrados.
- **R08 - Registrar Produtos**
  Descrição: O sistema deve ser capaz de registrar os produtos incluídos no pedido.
- **R09 - Registrar Pratos (Pratos do Cardápio)**
  Descrição: O sistema deve ser capaz de registrar os pratos do cardápio.
- **R10 - Notificar Cliente pelo WhatsApp**
  Descrição: O sistema deve notificar o cliente sobre o status do pedido via WhatsApp.

F03 - Preparação de Pedidos

- **R01 - Notificar Cozinha**
  Descrição: O sistema deve notificar a cozinha sobre novos pedidos.
- **R02 - Ler QR Code do Cozinheiro**
  Descrição: O sistema deve permitir a leitura do QR Code do cozinheiro.
- **R03 - Vincular Cozinheiro ao Pedido**
  Descrição: O sistema deve vincular um cozinheiro a um pedido.
- **R04 - Atualizar Status do Pedido via Leitura de QR Code**
  Descrição: O sistema deve atualizar o status do pedido durante a preparação utilizando o QR Code.
- **R05 - Atualizar Status do Pedido para Estabelecimento**
  Descrição: O sistema deve atualizar o status do pedido durante a preparação.
- **R06 - Avisar Motoboy**
  Descrição: O sistema deve avisar o motoboy quando o pedido estiver pronto.
- **R07 - Permitir Atualização Manual de Status pelo Gestor**
  Descrição: O sistema deve permitir que o gestor atualize manualmente o status do pedido.

- **R08 - Atualizar Status do Pedido para Cliente**
  Descrição: O sistema deve atualizar o status do pedido para o cliente

- **R09 - Imprimir Pedido**
  Descrição: O sistema deve permitir a impressão do pedido para conferência.

- **R10 - Avisar Motoboy**
  Descrição: O sistema deve avisar o motoboy quando o pedido estiver pronto.

Épico - Entrega (E03)

Descrição: O sistema deve ser capaz de gerenciar as entregas e motoboys.

F04 - Organização de Entregas

- **R01 - Definir Limite de Itens por Região de Entrega**
  Descrição: O sistema deve permitir a definição de um limite de itens por região de entrega.

- **R02 - Organizar Pedidos por Localidade**
  Descrição: O sistema deve organizar os pedidos por localidade para otimizar as entregas.

- **R03 - Gerenciar Áreas de Entrega**
  Descrição: O sistema deve permitir o gerenciamento das áreas de entrega.

- **R04 - Calcular Frete Automaticamente com Base na Região**
  Descrição: O sistema deve calcular o frete automaticamente com base na região de entrega.

- **R05 - Definir Limite de Itens por Região de Entrega**
  Descrição: O sistema deve permitir a definição de um limite de itens por região de entrega.

F05 - Motoboy

- **R01 - Ler QR Code do Motoboy**
  Descrição: O sistema deve permitir a leitura do QR Code do motoboy.

- **R02 - Informar Pedidos ao Motoboy**
  Descrição: O sistema deve informar ao motoboy os pedidos que ele deve entregar.

- **R03 - Vincular Pedido ao Motoboy**
  Descrição: O sistema deve vincular um pedido a um motoboy.

- **R04 - Calcular Tempo de Trânsito da Entrega via API**
  Descrição: O sistema deve calcular o tempo de trânsito da entrega utilizando uma API de mapas.

- **R05 - Notificar Cliente sobre Entrega**
  Descrição: O sistema deve notificar o cliente sobre o status da entrega.

- **R06 - Atualizar Status da Entrega**

Épico - Funcionários (E04)

Descrição: O sistema deve ser capaz de gerenciar os registros de funcionários.

F06 - Registros de Funcionários

- **R01 - Registrar Motoboy**
  Descrição: O sistema deve registrar informações dos motoboys.

- **R02 - Gerar QR Code do Motoboy**
  Descrição: O sistema deve gerar QR Codes para os motoboys.

- **R03 - Registrar Cozinheiro**
  Descrição: O sistema deve registrar informações dos cozinheiros.

- **R04 - Gerar QR Code do Cozinheiro**
  Descrição: O sistema deve gerar QR Codes para os cozinheiros.

- **R05 - Definir Cargo e Permissões do Funcionário**
  Descrição: O sistema deve definir cargo e atribuir suas retrospectivas permissões.

- **R06 - Autenticação e Controle de Acesso por Cargo**
  Descrição: O sistema deve autenticar e controlar o acesso dos funcionários com base em seus cargos.

- **R07 - Gerar QR Code Único Baseado em UUID**
  Descrição: O sistema deve gerar um QR Code único para cada funcionário, baseado em UUID.

Épico - Feedback (E05)

Descrição: O sistema deve ser capaz de coletar e analisar feedbacks de clientes.

F07 - Feedback do Cliente

- **R01 - Enviar Pesquisa de Satisfação**
  Descrição: O sistema deve enviar pesquisas de satisfação aos clientes.
- **R02 - Coletar Dados de Feedback**
  Descrição: O sistema deve coletar os dados de feedback dos clientes.
- **R03 - Analisar Feedback**
  Descrição: O sistema deve analisar os feedbacks coletados.

Épico - Dashboard (E06)

Descrição: O sistema deve ser capaz de fornecer relatórios e análises para o restaurante.

F08 - Relatórios Financeiros

- **R01 - Gerar Balanço**
  Descrição: O sistema deve gerar balanços diários, semanais e mensais.
- **R02 - Calcular Ticket Médio**
  Descrição: O sistema deve calcular o ticket médio dos pedidos.
- **R03 - Categorizar Pratos Vendidos**
  Descrição: O sistema deve categorizar os pratos vendidos por categorias, identificar as bebidas mais vendidas e os itens mais frequentes em um pedido.
- **R04 - Realizar Controle Básico de Estoque**
  Descrição: O sistema deve realizar um controle básico de estoque, permitindo o registro de entradas e saídas de produtos.
- **R05 -Exibir Dashboard com Indicadores Gerais**
  Descrição: O sistema deve exibir um dashboard com indicadores gerais, como vendas totais, número de pedidos e satisfação do cliente.
- **R06 - Gerar Relatório de Produção Detalhado**
  Descrição: O sistema deve gerar um relatório detalhado de produção, incluindo a quantidade de pratos preparados, tempo médio de preparo e eficiência da equipe.
- **R07 - Analisar Tempo de Preparo por Prato**
  Descrição: O sistema deve analisar o tempo de preparo por prato.
- **R08 - Realizar Análise de Desempenho por Período**
  Descrição: O sistema deve realizar uma análise de desempenho por período.

- **R09 - Gerar Insights de Negócio com Análise de IA**
  Descrição: O sistema deve gerar insights de negócio utilizando análise de IA.

F09 - Relatórios de Clientes

- **R01 - Identificar Recorrência de Clientes**
  Descrição: O sistema deve identificar clientes recorrentes.
- **R02 - Contabilizar Novos Clientes**
  Descrição: O sistema deve contabilizar os novos clientes.
- **R03 - Quantidade de Pedidos por Cliente**
  Descrição: O sistema deve contabilizar a quantidade de pedidos por cliente.
- **R04 - Observações mais Feitas por Cliente**
  Descrição: O sistema deve contabilizar as observações mais feitas por cliente.
- **R05 - Itens mais Excluidos pelo Cliente**
  Descrição: O sistema deve contabilizar itens mais excluidos pelo cliente.
- **R06 - Feedback pelo Cliente**
  Descrição: O sistema deve gerar relatórios de feeback do cliente.

F11 - Relatórios de Funcionários

- **R01 - Registrar Marmitas Feitas**
  Descrição: O sistema deve registrar a quantidade de marmitas feitas por cada cozinheiro.
- **R02 - Registrar Entregas Feitas**
  Descrição: O sistema deve registrar a quantidade de entregas feitas por cada motoboy.
- **R03 - Registrar Erros por Funcionário**
  Descrição: O sistema deve registrar a quantidade de erros cometidos por funcionário.
- **R04 - Gerar Lista de Desempenho de Funcionário**
  Descrição: O sistema deve gerar uma lista de desempenho do funcionário.
- **R05 - Gerar Relatório de Entregas com Métricas Avançadas**
  Descrição: O sistema deve gerar um relatório de entregas com métricas avançadas, incluindo tempo médio de entrega e eficiência do motoboy.

APÊNDICE F – Resultado da Priorização dos Requisitos

Must

- **RF01 - Registrar Nome**: Essencial para a identificação pessoal de cada cliente.
- **RF02 - Registrar Telefone**: Fundamental para contato em caso de problemas com o pedido ou entrega.
- **RF03 - Registrar Endereço**: Crítico para a funcionalidade de entrega, base de toda a logística.
- **RF04 - Registrar Pedido**: O ato central do sistema; sem ele, a operação não existe.
- **RF05 - Resumo do Pedido**: Necessário para que o cliente e o atendente confirmem os itens e valores antes de finalizar.
- **RF06 - Forma de Pagamento**: Essencial para o fechamento financeiro do pedido.
- **RF07 - Notificar Cozinha**: Gatilho que inicia o processo de produção, crucial para a agilidade da operação.
- **RF08 - Ler QR Code do Cozinheiro**: Mecanismo chave para a automação e rastreabilidade do início do preparo.
- **RF09 - Vincular Cozinheiro ao Pedido**: Fundamental para a rastreabilidade e atribuição de responsabilidade na produção.
- **RF10 - Atualizar Status do Pedido para Estabelecimento**: Essencial para que a equipe de gestão e atendimento saiba o andamento de cada pedido.
- **RF11 - Organizar Pedidos por Localidade**: Necessário para otimizar a logística e agrupar entregas de forma eficiente.
- **RF12 - Ler QR Code do Motoboy**: Mecanismo chave para automatizar o início da etapa de entrega.
- **RF13 - Informar Pedidos ao Motoboy**: Necessário para que o entregador saiba quais pacotes coletar.

- **RF14 - Vincular Pedido ao Motoboy**: Fundamental para a rastreabilidade da entrega e atribuição de responsabilidade.
- **RF15 - Registrar Motoboy**: Essencial para criar um cadastro de entregadores e gerenciar a equipe.
- **RF16 - Gerar QR Code do Motoboy**: Necessário para a identificação única e digital de cada entregador.
- **RF17 - Registrar Cozinheiro**: Essencial para o cadastro da equipe de cozinha e gestão de pessoal.
- **RF18 - Gerar QR Code do Cozinheiro**: Necessário para a identificação inequívoca de cada cozinheiro no sistema.
- **RF19 - Definir Cargo e Permissões do Funcionário**: Crítico para a segurança do sistema, garantindo que cada usuário acesse apenas o que é permitido.
- **RF20 - Registrar Marmitas Feitas**: Fundamental para a coleta de métricas de produtividade da cozinha.
- **RF21 - Registrar Entregas Feitas**: Fundamental para a coleta de métricas de desempenho dos motoboys.
- **RF22 - Autenticação e Controle de Acesso por Cargo**: Requisito de segurança não funcional que é mandatório para qualquer sistema com múltiplos usuários.
- **RF23 - Gerenciar Áreas de Entrega**: Essencial para a automação do cálculo de frete e para definir os limites operacionais do delivery.
- **RF24 - Registrar Produtos**: Base para a criação do cardápio; sem produtos, não há o que vender.
- **RF25 - Registrar Pratos (Pratos do Cardápio)**: Base para a criação do cardápio; sem pratos, não há o que vender.
- **RF26 - Atualizar Status do Pedido via Leitura de QR Code**: É a implementação central que garante a automação e a atualização de status em tempo real.

Should

- **RF27 - Editar Registro**: Importante para corrigir erros e manter a base de dados de clientes atualizada.

- **RF28 - Registrar Observações do Pedido**: Aumenta a satisfação do cliente e reduz erros no preparo, sendo uma funcionalidade de alto valor.
- **RF29 - Gerar Balanço**: Fornece uma visão financeira essencial para a gestão do negócio.
- **RF30 - Identificar Recorrência de Clientes**: Importante para estratégias de marketing e fidelização.
- **RF31 - Registrar Erros por Funcionário**: Fundamental para a gestão de qualidade e identificação de necessidades de treinamento.
- **RF32 - Notificações em Tempo Real com WebSockets**: Melhora drasticamente a usabilidade e a eficiência do sistema, evitando atualizações manuais.
- **RF33 - Calcular Tempo de Trânsito da Entrega via API**: Agrega grande valor à experiência do cliente e ajuda na gestão de expectativas.
- **RF34 - Gerar QR Code Único Baseado em UUID**: Uma melhoria técnica importante que garante a robustez e evita a duplicidade de códigos.
- **RF35 - Realizar Controle Básico de Estoque**: Importante para a gestão de insumos e para evitar a venda de produtos indisponíveis.
- **RF36 - Permitir Atualização Manual de Status pelo Gestor**: Essencial como um mecanismo de contingência para corrigir falhas no fluxo automatizado.
- **RF37 - Exibir Dashboard com Indicadores Gerais**: Oferece uma visão gerencial rápida e consolidada, de alto valor para a tomada de decisões diárias.
- **RF38 - Calcular Frete Automaticamente com Base na Região**: Automatiza uma tarefa crítica, economizando tempo e evitando erros de cobrança.
- **RF39 - Calcular Ticket Médio**: Métrica importante para a análise da saúde financeira e comportamento de compra.
- **RF40 - Categorizar Pratos Vendidos**: Ajuda a entender quais itens do cardápio são mais e menos populares, guiando decisões de negócio.
- **RF41 - Contabilizar Novos Clientes**: Métrica importante para avaliar o crescimento do negócio.
- **RF42 - Quantidade de Pedidos por Cliente**: Ajuda a identificar os clientes mais valiosos (VIPs).
- **RF43 - Gerar Lista de Desempenho de Funcionário**: Ferramenta de gestão de pessoas importante para feedbacks e avaliações.

- **RF44 - Gerar Relatório de Produção Detalhado**: Fornece insights aprofundados sobre a eficiência da cozinha.
- **RF45 - Analisar Tempo de Preparo por Prato**: Permite otimizar o cardápio e o processo de preparo.
- **RF46 - Gerar Relatório de Entregas com Métricas Avançadas**: Oferece uma visão detalhada da performance da logística.
- **RF47 - Realizar Análise de Desempenho por Período**: Permite comparar resultados e identificar tendências sazonais.
- **RF48 - Gerar Insights de Negócio com Análise de IA**: Funcionalidade de alto valor agregado para otimização e predição, um diferencial competitivo.

Could

- **RF49 - Atualizar Status do Pedido para Cliente**: Melhora a experiência do cliente com mais transparência, mas não é vital para a operação interna.
- **RF50 - Imprimir Pedido**: Funciona como um backup físico útil, mas não é essencial para um fluxo primariamente digital.
- **RF51 - Destacar Observações na Nota**: Uma melhoria de usabilidade (UX) que ajuda a evitar erros, mas pode ser implementada posteriormente.
- **RF52 - Notificar Cliente pelo WhatsApp**: Um canal de comunicação específico que agrega valor, mas o sistema pode funcionar com notificações mais simples.
- **RF53 - Notificar Cliente sobre Entrega**: Refinamento da experiência do cliente, mas não uma funcionalidade central.
- **RF54 - Definir Limite de Itens por Região de Entrega**: Uma regra de negócio avançada para otimizar a logística em casos específicos, mas não essencial para o início.
- **RF55 - Itens mais Excluídos pelo Cliente**: Requisito de análise de dados definido como fora do escopo para esta versão.

Won't

- **RF56 - Observações mais Feitas por Cliente**: Requisito de análise de dados definido como fora do escopo para esta versão.

- **RF57 - Avisar Motoboy**: Definido como fora do escopo, pois a comunicação com o motoboy sobre a prontidão do pedido pode ser realizada por outros meios no fluxo atual.
- **RF58 - Feedback pelo Cliente**: A coleta de feedback será tratada pelo requisito de pesquisa de satisfação (Could), mas um relatório específico de "Feedback pelo Cliente"foi despriorizado.
- **RF59 - Enviar Pesquisa de Satisfação**: Importante para a melhoria contínua a longo prazo, mas não impacta a operação diária.
- **RF60 - Coletar Dados de Feedback**: É uma consequência da RF52.
- **RF61 - Analisar Feedback**: É a análise posterior dos dados coletados, uma atividade gerencial de menor prioridade inicial.

Legenda:

- **E01**: Épico relacionado aos Clientes.
- **E02**: Épico relacionado aos Pedidos.
- **E03**: Épico relacionado à Entrega.
- **E04**: Épico relacionado aos Funcionários.
- **E05**: Épico relacionado ao Feedback.
- **E06**: Épico relacionado ao Dashboard.
- **F01**: Funcionalidade de Registro de Clientes.
- **F02**: Funcionalidade de Registro de Pedidos.
- **F03**: Funcionalidade de Preparação de Pedidos.
- **F04**: Funcionalidade de Organização de Entregas.
- **F05**: Funcionalidade do Motoboy.
- **F06**: Funcionalidade de Registros de Funcionários.
- **F07**: Funcionalidade de Feedback do Cliente.
- **F08**: Funcionalidade de Relatórios Financeiros.
- **F09**: Funcionalidade de Relatórios de Clientes.
- **F10**: Funcionalidade de Relatórios de Funcionários.

APÊNDICE G – Dicionário de Dados

Tabela 6 – Estrutura da entidade FUNCIONARIO

Entidade: FUNCIONARIO				
Descrição: Armazena informações sobre os funcionários do restaurante.				
Atributo	**Propriedades dos Atributos**	**Tipo de dado**	**Tamanho**	**Descrição**
cpf	Chave primária	VARCHAR	11	CPF do funcionário
nome	Obrigatória	VARCHAR	100	Nome completo do funcionário
senha	Obrigatória	VARCHAR	256	Hash da senha de acesso
tipo	Obrigatória	VARCHAR	20	Tipo de funcionário (e.g., 'GESTOR')
qrCode	Chave única, Opcional	VARCHAR	256	Código QR para identificação
dataAtivo	Obrigatória	DATE	-	Data de ativação do cadastro
editado	Opcional	BOOLEAN	-	Indica se o registro foi editado
criado	Obrigatório	TIMESTAMP	-	Data e hora de criação do registro
atualizado	Opcional	TIMESTAMP	-	Data da última atualização

Fonte: Autores (2025).

Tabela 7 – Estrutura da entidade CLIENTE

Entidade: CLIENTE				
Descrição: Armazena informações sobre os clientes do restaurante.				
Atributo	**Propriedades dos Atributos**	**Tipo de dado**	**Tamanho**	**Descrição**
idCliente	Chave primária	INT	-	Identificador único do cliente
nome	Obrigatória	VARCHAR	100	Nome completo do cliente
observacoes	Opcional	VARCHAR	256	Observações gerais do cliente
criado	Obrigatório	TIMESTAMP	-	Data e hora de criação do registro
atualizado	Opcional	TIMESTAMP	-	Data da última atualização

Fonte: Autores (2025).

Tabela 8 – Estrutura da entidade PEDIDO

Entidade: PEDIDO				
Descrição: Armazena informações sobre os pedidos realizados.				
Atributo	**Propriedades dos Atributos**	**Tipo de dado**	**Tamanho**	**Descrição**
idPedido	Chave primária	INT	-	Identificador único do pedido
status	Obrigatória	VARCHAR	30	Status atual do pedido
dataHora	Obrigatória	TIMESTAMP	-	Data e hora do pedido
valor	Obrigatório	DECIMAL	10,2	Valor total do pedido
observacao	Opcional	VARCHAR	256	Observações para o pedido
entregue	Opcional	VARCHAR	256	Horario em que foi entregue
entregando	Opcional	VARCHAR	256	Horario em que começou a entrega
comecoPreparo	Opcional	VARCHAR	256	Horario em que iniciou preparo
criado	Obrigatório	VARCHAR	256	Horário em que foi criado
pronto	Opcional	VARCHAR	256	Horário em que ficou pronto

Fonte: Autores (2025).

Tabela 9 – Estrutura da entidade PAGAMENTO

Entidade: PAGAMENTO				
Descrição: Detalhes de pagamento associados a um pedido.				
Atributo	**Propriedades dos Atributos**	**Tipo de dado**	**Tamanho**	**Descrição**
idPagamento	Chave primária	INT	-	Identificador único
infoAdicional	Opcional	VARCHAR	256	Observações
criado	Obrigatório	VARCHAR	256	Data da criação
valorTotal	Obrigatório	DECIMAL	10,2	Valor total a ser pago
atualização	Opcional	VARCHAR	256	Data da atualização
formaPagamento	Obrigatória	VARCHAR	30	Forma de pagamento
status	Obrigatória	VARCHAR	20	Status (e.g., 'Aprovado')
data	Obrigatória	TIMESTAMP	-	Data e hora da transação
idTransacao	Opcional	VARCHAR	100	ID da transação (gateway)
quantidadePaga	Opcional	DECIMAL	10,2	Valor pago
troco	Opcional	DECIMAL	10,2	Valor do troco, se aplicável

Fonte: Autores (2025).

Tabela 10 – Estrutura da entidade PRATO

Entidade: PRATO				
Descrição: Catálogo de pratos oferecidos pelo restaurante.				
Atributo	**Propriedades dos Atributos**	**Tipo de dado**	**Tamanho**	**Descrição**
idPrato	Chave primária	INT	-	Identificador único do prato
status	Obrigatória	VARCHAR	20	Status do prato (e.g., 'Ativo')
servindo	Obrigatória	VARCHAR	20	Tipo de prato (e.g., 'Almoço')
nome	Obrigatória	VARCHAR	100	Nome do prato
criado	Obrigatório	VARCHAR	256	Data de criação
descricao	Opcional	VARCHAR	256	Descrição e ingredientes
atualizado	Opcional	VARCHAR	256	Data da última atualização
preço	Obrigatória	DECIMAL	10,2	Preço unitário
tamanho	Opcional	VARCHAR	20	Tamanho do prato
urlimagem	Opcional	VARCHAR	256	URL da imagem do prato
categoria	Opcional	VARCHAR	50	Categoria do prato
disponibilidade	Obrigatória	BOOLEAN	-	Disponível no cardápio

Fonte: Autores (2025).

Tabela 11 – Estrutura da entidade PRODUTO

Entidade: PRODUTO				
Descrição: Produtos avulsos vendidos (e.g., bebidas, sobremesas).				
Atributo	**Propriedades dos Atributos**	**Tipo de dado**	**Tamanho**	**Descrição**
idProduto	Chave primária	INT	-	Identificador único
nome	Obrigatória	VARCHAR	100	Nome do produto
preco	Obrigatória	DECIMAL	10,2	Preço unitário
categoria	Opcional	VARCHAR	50	Categoria do produto
subCategoria	Opcional	VARCHAR	50	Subcategoria do produto
quantidade	Obrigatória	INT	-	Quantidade em estoque
peso	Opcional	DECIMAL	10,2	Peso do produto

Fonte: Autores (2025).

Tabela 12 – Estrutura do atributo composto e multivalorado ENDERECO

Atributo Composto: ENDERECO				
Descrição: Armazena múltiplos endereços cadastrados.				
Atributo	**Propriedades dos Atributos**	**Tipo de dado**	**Tamanho**	**Descrição**
idCliente	FK	INT	-	Referência ao cliente
rua	Obrigatória	VARCHAR	20	Nome da rua
bairro	Obrigatória	VARCHAR	20	Bairro do endereço
numero	Obrigatória	INT	-	Número do endereço
complemento	Opcional	VARCHAR	60	Complemento do endereço

Fonte: Autores (2025).

Tabela 13 – Estrutura do atributo multivalorado TELEFONE

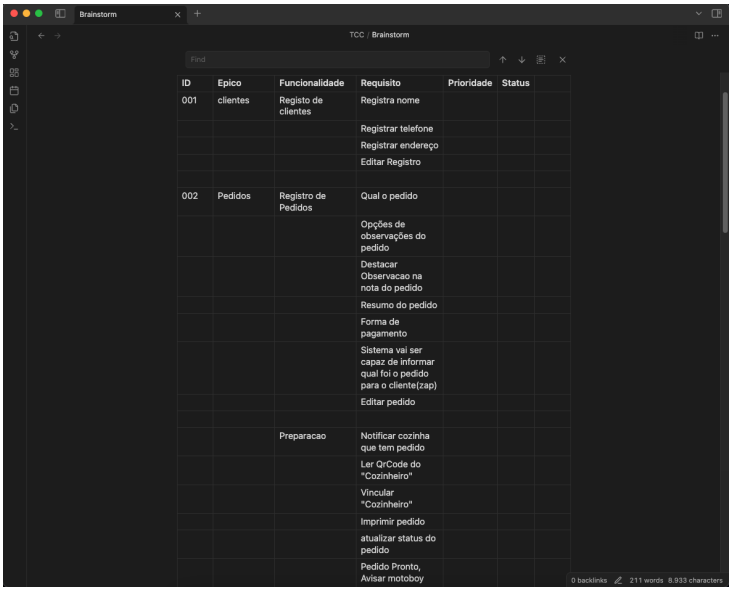
Atributo Multivalorado: TELEFONE				
Descrição: Armazena múltiplos telefones para clientes.				
Atributo	**Propriedades dos Atributos**	**Tipo de dado**	**Tamanho**	**Descrição**
idCliente	FK	INT	-	Referência ao cliente
telefone	Obrigatória	VARCHAR	11	Número de telefone

Fonte: Autores (2025).

Anexos

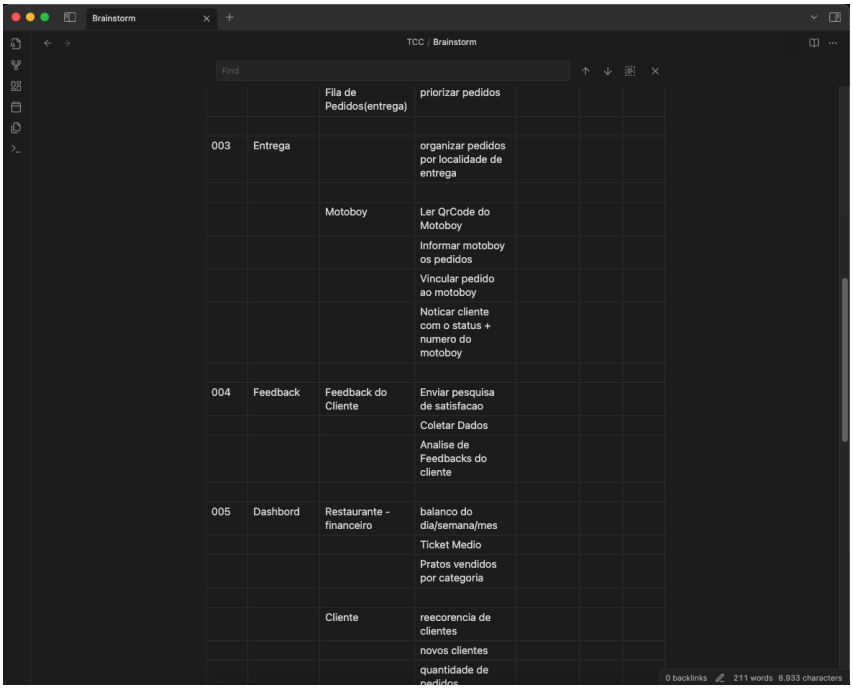
ANEXO A – Brainstorm

Figura 21 – Brainstorm parte 1



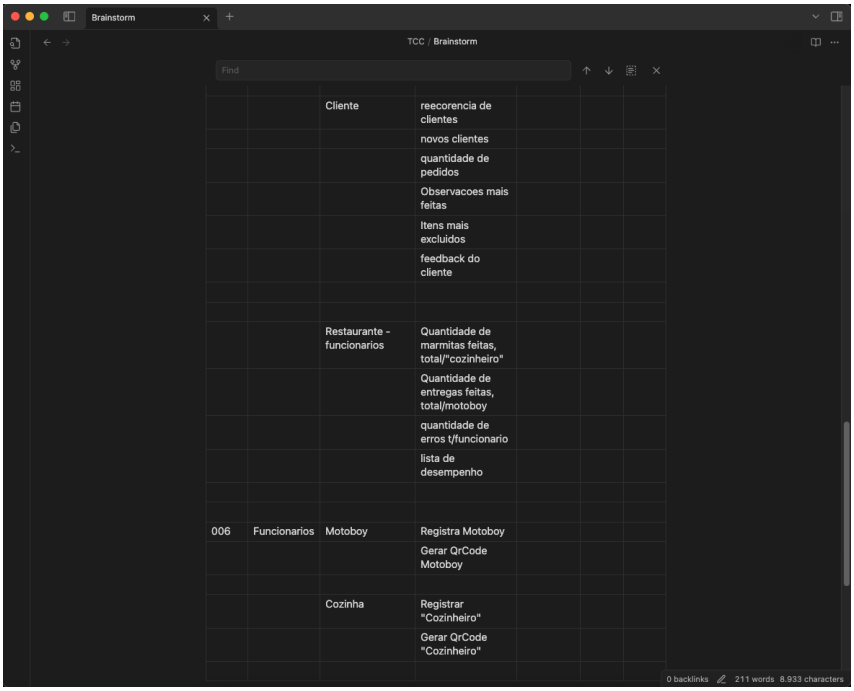
Fonte: Autores (2025).

Figura 22 – Brainstorm parte 2



Fonte: Autores (2025).

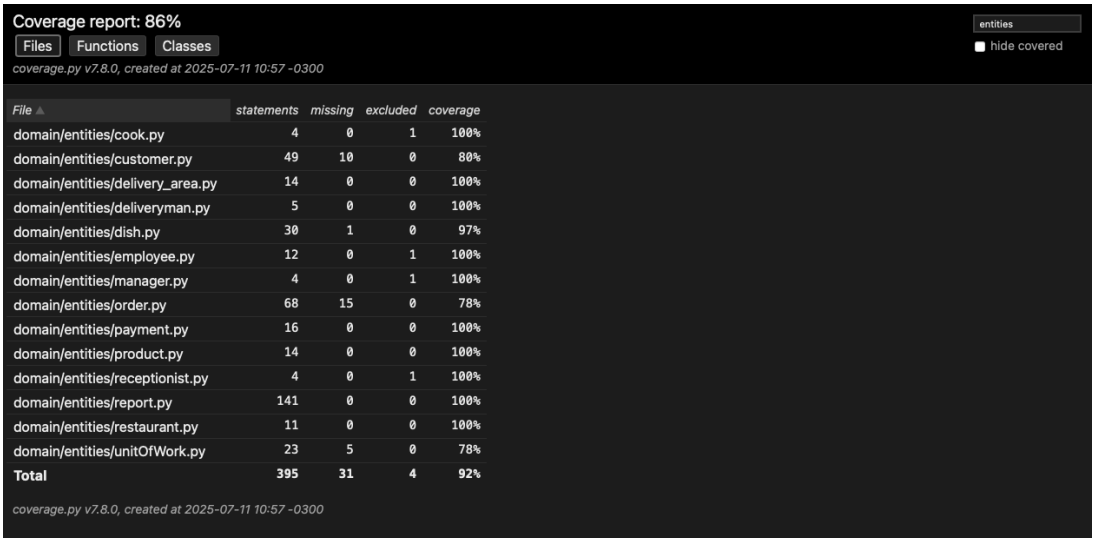
Figura 23 – Brainstorm parte 3



Fonte: Autores (2025).

ANEXO B – Relatórios de Cobertura de Testes Unitários

Figura 24 – Cobertura de Testes - Entidades



Coverage report: 86%

Files | Functions | Classes

entities

hide covered

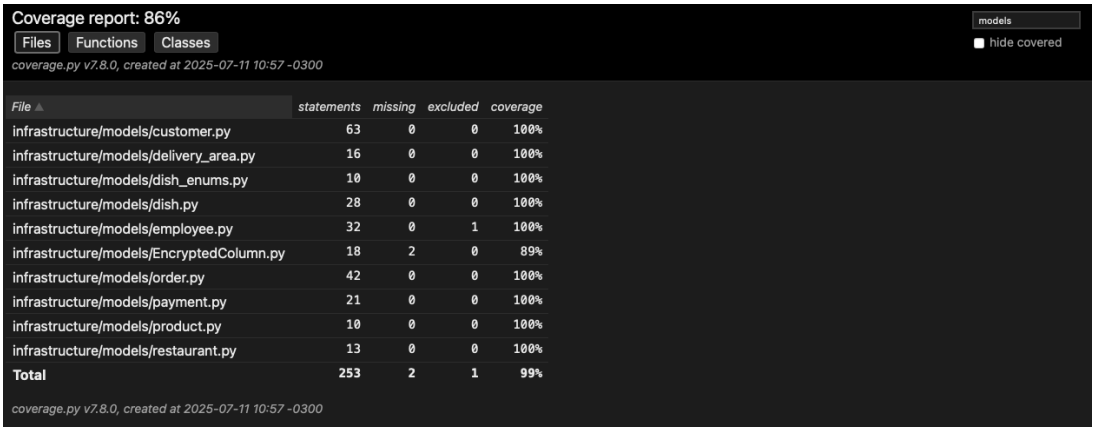
coverage.py v7.8.0, created at 2025-07-11 10:57 -0300

File	statements	missing	excluded	coverage
domain/entities/cook.py	4	0	1	100%
domain/entities/customer.py	49	10	0	80%
domain/entities/delivery_area.py	14	0	0	100%
domain/entities/deliveryman.py	5	0	0	100%
domain/entities/dish.py	30	1	0	97%
domain/entities/employee.py	12	0	1	100%
domain/entities/manager.py	4	0	1	100%
domain/entities/order.py	68	15	0	78%
domain/entities/payment.py	16	0	0	100%
domain/entities/product.py	14	0	0	100%
domain/entities/receptionist.py	4	0	1	100%
domain/entities/report.py	141	0	0	100%
domain/entities/restaurant.py	11	0	0	100%
domain/entities/unitOfWork.py	23	5	0	78%
Total	**395**	**31**	**4**	**92%**

coverage.py v7.8.0, created at 2025-07-11 10:57 -0300

Fonte: Autores (2025).

Figura 25 – Cobertura de Testes - Models



Coverage report: 86%

Files | Functions | Classes

models

hide covered

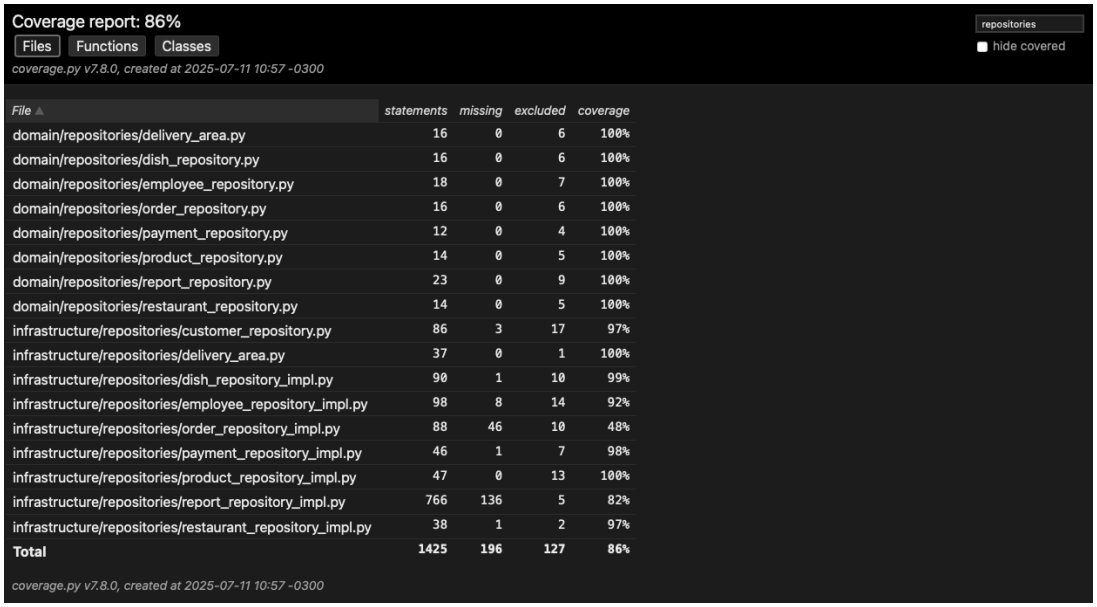
coverage.py v7.8.0, created at 2025-07-11 10:57 -0300

File	statements	missing	excluded	coverage
infrastructure/models/customer.py	63	0	0	100%
infrastructure/models/delivery_area.py	16	0	0	100%
infrastructure/models/dish_enums.py	10	0	0	100%
infrastructure/models/dish.py	28	0	0	100%
infrastructure/models/employee.py	32	0	1	100%
infrastructure/models/EncryptedColumn.py	18	2	0	89%
infrastructure/models/order.py	42	0	0	100%
infrastructure/models/payment.py	21	0	0	100%
infrastructure/models/product.py	10	0	0	100%
infrastructure/models/restaurant.py	13	0	0	100%
Total	**253**	**2**	**1**	**99%**

coverage.py v7.8.0, created at 2025-07-11 10:57 -0300

Fonte: Autores (2025).

Figura 26 – Cobertura de Testes - Repositórios



Coverage report: 86%

Files | Functions | Classes

coverage.py v7.8.0, created at 2025-07-11 10:57 -0300

repositories

hide covered

File	statements	missing	excluded	coverage
domain/repositories/delivery_area.py	16	0	6	100%
domain/repositories/dish_repository.py	16	0	6	100%
domain/repositories/employee_repository.py	18	0	7	100%
domain/repositories/order_repository.py	16	0	6	100%
domain/repositories/payment_repository.py	12	0	4	100%
domain/repositories/product_repository.py	14	0	5	100%
domain/repositories/report_repository.py	23	0	9	100%
domain/repositories/restaurant_repository.py	14	0	5	100%
infrastructure/repositories/customer_repository.py	86	3	17	97%
infrastructure/repositories/delivery_area.py	37	0	1	100%
infrastructure/repositories/dish_repository_impl.py	90	1	10	99%
infrastructure/repositories/employee_repository_impl.py	98	8	14	92%
infrastructure/repositories/order_repository_impl.py	88	46	10	48%
infrastructure/repositories/payment_repository_impl.py	46	1	7	98%
infrastructure/repositories/product_repository_impl.py	47	0	13	100%
infrastructure/repositories/report_repository_impl.py	766	136	5	82%
infrastructure/repositories/restaurant_repository_impl.py	38	1	2	97%
Total	**1425**	**196**	**127**	**86%**

coverage.py v7.8.0, created at 2025-07-11 10:57 -0300

Fonte: Autores (2025).

Figura 27 – Cobertura de Testes - Routers

Coverage report: 86%

Files | Functions | Classes

coverage.py v7.8.0, created at 2025-07-11 10:57 -0300

routers

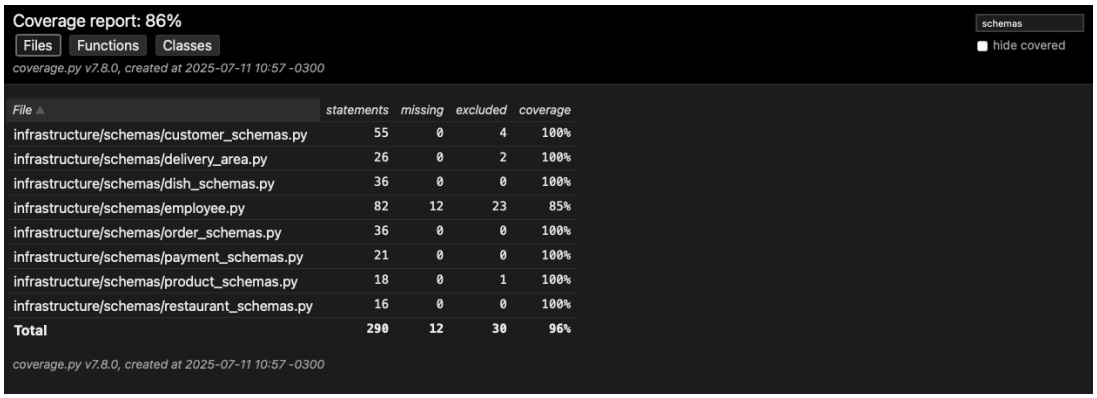
hide covered

File	statements	missing	excluded	coverage
interface/http/routers/auth_routes.py	20	0	6	100%
interface/http/routers/customer_routes.py	77	9	20	88%
interface/http/routers/delivery_area_routes.py	60	0	3	100%
interface/http/routers/dish_routes.py	43	2	4	95%
interface/http/routers/employee_routes.py	56	1	6	98%
interface/http/routers/maps_routes.py	18	0	2	100%
interface/http/routers/order_routes.py	197	39	38	80%
interface/http/routers/product_routes.py	37	4	3	89%
interface/http/routers/report_routes.py	203	33	5	84%
interface/http/routers/restaurant_routes.py	26	1	2	96%
interface/http/routers/websocket_routes.py	69	9	17	87%
Total	**806**	**98**	**106**	**88%**

coverage.py v7.8.0, created at 2025-07-11 10:57 -0300

Fonte: Autores (2025).

Figura 28 – Cobertura de Testes - Schemas



Coverage report: 86%

Files | Functions | Classes

coverage.py v7.8.0, created at 2025-07-11 10:57 -0300

schemas

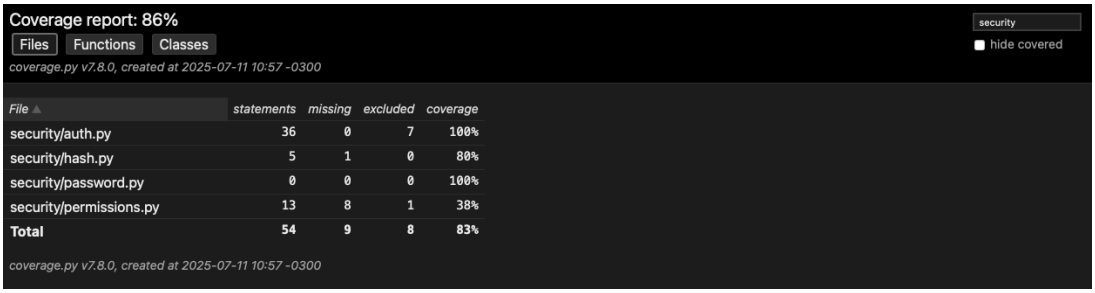
hide covered

File	statements	missing	excluded	coverage
infrastructure/schemas/customer_schemas.py	55	0	4	100%
infrastructure/schemas/delivery_area.py	26	0	2	100%
infrastructure/schemas/dish_schemas.py	36	0	0	100%
infrastructure/schemas/employee.py	82	12	23	85%
infrastructure/schemas/order_schemas.py	36	0	0	100%
infrastructure/schemas/payment_schemas.py	21	0	0	100%
infrastructure/schemas/product_schemas.py	18	0	1	100%
infrastructure/schemas/restaurant_schemas.py	16	0	0	100%
Total	**290**	**12**	**30**	**96%**

coverage.py v7.8.0, created at 2025-07-11 10:57 -0300

Fonte: Autores (2025).

Figura 29 – Cobertura de Testes - Módulo de Segurança



Fonte: Autores (2025).

Figura 30 – Cobertura de Testes - Casos de Uso

Fonte: Autores (2025).