CENTRO ESTADUAL DE EDUCAÇÃO TECNOLÓGICA PAULA SOUZA
FACULDADE DE TECNOLOGIA DE PRAIA GRANDE
CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

RELATÓRIO TÉCNICO DE CONCLUSÃO DE CURSO

CHEFPRICE: SISTEMA DE GERENCIAMENTO DE GASTOS E CUSTOS NA PRODUÇÃO DE ALIMENTOS

LIGIA PIVA DE SANTANA

GEOVANA VIEIRA DE SOUSA

THIAGO SANSALONE D'ALESSANDRO

Orientador: Alessandro Ferreira Paz Lima

PRAIA GRANDE

2025

LIGIA PIVA DE SANTANA

GEOVANA VIEIRA DE SOUSA

THIAGO SANSALONE D'ALESSANDRO

CHEFPRICE: SISTEMA DE GERENCIAMENTO DE GASTOS E CUSTOS NA PRODUÇÃO DE ALIMENTOS

Relatório técnico de conclusão de curso, apresentado ao curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, como requisito parcial para obtenção do título de tecnólogo.

Orientador: Alessandro Ferreira Paz Lima

PRAIA GRANDE

2025

AGRADECIMENTOS

A realização deste trabalho só foi possível graças ao apoio, à paciência e ao amor de pessoas especiais, que caminharam ao nosso lado mesmo nos momentos mais desafiadores. A cada uma delas, dedicamos esta conquista com todo o nosso carinho e gratidão.

À Fatec, por nos proporcionar uma base sólida de conhecimento e por nos desafiar a transformar ideias em soluções reais.

Ao nosso orientador, professor Alessandro, que com sabedoria, dedicação e conselhos valiosos, guiou nossos passos e nos ajudou a moldar nossa ideia em um projeto concreto. Seu incentivo fez toda a diferença.

Aos nossos familiares, que mesmo diante da nossa ausência em momentos importantes, nunca deixaram de nos apoiar. Obrigada por compreenderem nossas renúncias e acreditarem no nosso sonho tanto quanto nós. Sem o suporte emocional de vocês, nada disso teria sido possível.

A todos que, de alguma forma, contribuíram com palavras de incentivo, com escuta atenta ou simplesmente acreditando no nosso potencial: nosso mais sincero muito obrigada.

Essa entrega é fruto de esforço, resiliência e, acima de tudo, de muito amor. E a cada um que fez parte dessa trajetória, deixamos nossa eterna gratidão.

RESUMO

Este trabalho apresenta o desenvolvimento do ChefPrice, um sistema web inovador para o gerenciamento de custos e despesas na produção de produtos alimentícios. A motivação para a criação do sistema surgiu da experiência da autora como confeiteira, que enfrentava desafios na precificação e no cálculo preciso dos custos de produção. O ChefPrice foi concebido para auxiliar pequenos empreendedores do setor alimentício, oferecendo uma ferramenta prática, intuitiva e eficaz para o controle financeiro. Suas principais funcionalidades incluem o cadastro de materiais com informações detalhadas de quantidade e valor por embalagem, a montagem de receitas com cálculo automático de custos diretos e indiretos, e a sugestão de preço de venda baseada na margem de lucro desejada. Destaca-se também a criação de listas de receitas para referência de mercado e a leitura automatizada de notas fiscais com inteligência artificial para atualização de preços. O sistema foi desenvolvido utilizando **angular** no **Front-end** com estilização em **Bootstrap**, **Laravel** na API, e **Microsoft SQL Server** como banco de dados. O ChefPrice visa proporcionar maior praticidade, controle e organização para empreendedores do setor de alimentos, promovendo uma gestão financeira mais eficiente e sustentável.

Palavras-chave: Gestão de custos, sistema web, precificação, produtos alimentícios, empreendedorismo.

ABSTRACT

This work presents the development of ChefPrice, an innovative web system for managing costs and expenses in the production of food products. The motivation for creating the system arose from the author's experience as a confectioner, who faced challenges in pricing and accurately calculating production costs. ChefPrice was designed to assist small entrepreneurs in the food sector, providing a practical, intuitive, and effective tool for financial control. Its main features include the registration of materials with detailed information on quantity and value per package, the assembly of recipes with automatic calculation of direct and indirect costs, and the suggestion of selling prices based on the desired profit margin. It also features the creation of recipe lists for market reference and automated reading of invoices with artificial intelligence for price updates. The system was developed using Angular In the front-end with styling in Bootstrap, Laravel in the API, and Microsoft SQL Server as the database. ChefPrice aims to provide greater practicality, control, and organization for entrepreneurs in the food sector, promoting a more efficient and sustainable financial management.

Keywords: Cost management, web system, pricing, food products, entrepreneurship.

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

PHP	*Hypertext Preprocessor*
UML	*Unified Modeling Language*
IA	*Inteligência Artificial*
API	*Application Programming Interface*
JSON	*JavaScript Object Notation*
RNF	*Requisito Não Funcional*
RF	*Requisito Funcional*
OCR	*Optical Character Recognition*
AWS	*Amazon Web Service*
RDS	*Relational Database Service*
UX	*User Experience*
HTTP	*Hypertext Transfer Protocol*
BPMN	*Business Process Model and Notation*
TS	*TypeScript*

SUMÁRIO

1.INTRODUÇÃO

Na atualidade, uma boa gestão das produções alimentícias é essencial para o crescimento e a sustentabilidade de qualquer empreendimento no setor. Pequenos empreendedores, como confeiteiros, donos de lanchonetes, *food trucks* e produtores artesanais, enfrentam uma série de desafios operacionais e financeiros que, quando mal administrados, podem comprometer significativamente o sucesso do negócio. A complexidade desse cenário evidencia a urgência por soluções tecnológicas que auxiliem na organização dos custos de produção, no controle de estoque e na precificação adequada dos produtos.

Esses empreendedores precisam conhecer detalhadamente os custos dos materiais e insumos utilizados, diferenciando entre gastos diretos — como ingredientes e embalagens — e gastos indiretos, que englobam energia elétrica, água, internet, mão de obra, entre outros. A ausência de um controle estruturado dessas informações é uma das principais causas de falência de micro e pequenas empresas no Brasil, como apontam Azevedo e Leone (2011), que identificam a gestão financeira empírica e a falta de planejamento como os maiores obstáculos enfrentados por negócios desse porte.

Além disso, no contexto da chamada Era da Informação, o uso de sistemas de informação gerencial passou a ser indispensável para a eficiência empresarial, pois fornece aos gestores dados organizados e relevantes para a tomada de decisões rápidas e acertadas (Bazzotti; Garcia, 2003). Os sistemas de informação, quando bem estruturados, têm o poder de reduzir incertezas, aumentar a produtividade e fornecer vantagens competitivas para pequenos negócios.

A isso se soma a evolução recente no uso da inteligência artificial como ferramenta para extração de dados em documentos não estruturados, como notas fiscais, contribuindo significativamente para a automatização de tarefas repetitivas e a redução de erros manuais. Conforme Silva (2016), o uso de algoritmos de machine learning e técnicas como *Named Entity Recognition* (NER) é estratégico para processos de classificação, leitura e interpretação de textos complexos, promovendo ganho de tempo e confiabilidade nas informações utilizadas.

Diante dessa realidade, surge o ChefPrice, um sistema web desenvolvido para facilitar a gestão de custos e a precificação de produtos alimentícios, com foco em pequenos produtores. Construído com angular no Front-end e Laravel na

API, utilizando Microsoft SQL Server para armazenamento de dados, o ChefPrice oferece uma plataforma integrada que centraliza o cadastro de materiais, receitas, cálculo automático de custos e margens de lucro, além de funcionalidades como lista de compras inteligentes e leitura de notas fiscais via IA.

Ao unir fundamentos de gestão financeira, tecnologia da informação e inteligência artificial, o ChefPrice visa democratizar o acesso a ferramentas de gestão profissional, permitindo que pequenos empreendedores tomem decisões com base em dados reais, reduzam desperdícios e aumentem sua lucratividade.

1.1. OBJETIVO GERAL

Desenvolver um sistema web de gestão de custos e precificação de produtos alimentícios, denominado ChefPrice, voltado para pequenos empreendedores do setor, com o objetivo de proporcionar uma ferramenta acessível, prática e eficiente para o controle financeiro da produção, integrando recursos de tecnologia da informação, inteligência artificial e sistemas de apoio à decisão, de forma a otimizar o gerenciamento de materiais, automatizar cálculos de custos diretos e indiretos, sugerir preços de venda com base em margem de lucro e gerar relatórios analíticos que auxiliem na tomada de decisões estratégicas.

.

1.1.1. Objetivos específicos

São objetivos específicos os seguintes tópicos:

- Desenvolver uma interface web responsiva e intuitiva para o cadastro de materiais utilizados na produção culinária, contendo campos para nome, quantidade, valor por embalagem e unidade de medida.
- Implementar o cadastro de receitas com cálculo automático de custos diretos e indiretos, baseado na proporção de uso dos materiais e na inclusão de despesas operacionais como energia elétrica, água e mão de obra.

- Criar uma funcionalidade para definição de margem de lucro desejada, com geração automática de sugestão de preço de venda por receita.
- Integrar um módulo de leitura automatizada de notas fiscais, utilizando inteligência artificial, para extração dos valores pagos por insumos e atualização automática dos preços dos produtos cadastrados.
- Desenvolver uma funcionalidade para lista de compras vinculadas a receitas, permitindo o planejamento de aquisições com base em demandas de produção.
- Disponibilizar dashboards analíticos que apresentem a variação dos preços dos insumos ao longo do tempo, permitindo identificar tendências, fornecedores mais vantajosos.

2. JUSTIFICATIVA

A economia brasileira atual, é composta também por pequenos empreendedores do setor alimentício que representam uma parcela significativa da produção e geração de renda, mas enfrentam grandes dificuldades na gestão de seus negócios. Confeiteiros, donos de lanchonetes e outros produtores artesanais lidam diariamente com a complexidade de controlar gastos, precificar produtos de forma estratégica e tomar decisões financeiras com base em informações muitas vezes desorganizadas ou inexistentes. A ausência de ferramentas acessíveis e eficazes contribui para o alto índice de falência de micro e pequenas empresas, conforme apontado por Azevedo e Leone (2011), que identificam a má gestão financeira como uma das principais causas desse cenário.

Apesar da crescente importância da transformação digital, muitos desses empreendedores ainda utilizam métodos manuais, como planilhas ou cadernos, para registrar seus custos. Isso compromete a precisão dos cálculos, dificulta a visualização de lucros e inviabiliza uma gestão eficiente dos recursos. Neste sentido, a tecnologia da informação e os sistemas de informação gerencial se apresentam como aliados essenciais para transformar dados em informações

úteis e estratégicas, capazes de sustentar decisões mais seguras e rentáveis (Bazzotti; Garcia, 2003).

Além disso, os avanços recentes em inteligência artificial permitiram soluções inovadoras para a automação de tarefas repetitivas e sensíveis ao erro humano, como a leitura de notas fiscais digitalizadas. Segundo Silva (2016), a aplicação de técnicas como *Machine Learning* e *Named Entity Recognition* pode otimizar o processo de extração de informações relevantes em documentos não estruturados, como é o caso das notas fiscais de compras. Com isso, é possível garantir maior agilidade, redução de erros e atualização automática de preços em sistemas de gestão.

Diante desse cenário, o sistema ChefPrice justifica-se como uma solução prática, acessível e moderna, capaz de atender de forma personalizada as necessidades dos pequenos empreendedores alimentícios. Sua proposta de integrar cadastro de materiais, cálculo automático de receitas, análise de custos diretos e indiretos, definição de margem de lucro, geração de relatórios financeiros e leitura de notas fiscais com IA, oferece um pacote completo de funcionalidades voltado à sustentabilidade dos negócios e à autonomia financeira dos seus usuários.

Ao proporcionar mais controle, organização e eficiência na rotina de produção e vendas, o ChefPrice contribui para a profissionalização da gestão financeira de pequenos negócios, oferecendo uma base tecnológica que facilita o crescimento sustentável, a tomada de decisão estratégica e a competitividade no mercado.

Garantir a organização e armazenamento das informações em banco de dados estruturado, assegurando a integridade e a disponibilidade dos dados para consulta e geração de relatórios financeiros.

3. PLANEJAMENTO DO PROJETO

Em grande parte do Brasil, a gestão de custos ainda representa um desafio significativo para pequenos empreendedores, especialmente aqueles que atuam no ramo da alimentação. Muitos desses empreendedores ingressam nesse setor por ser uma área que permite iniciar com baixo investimento inicial e,

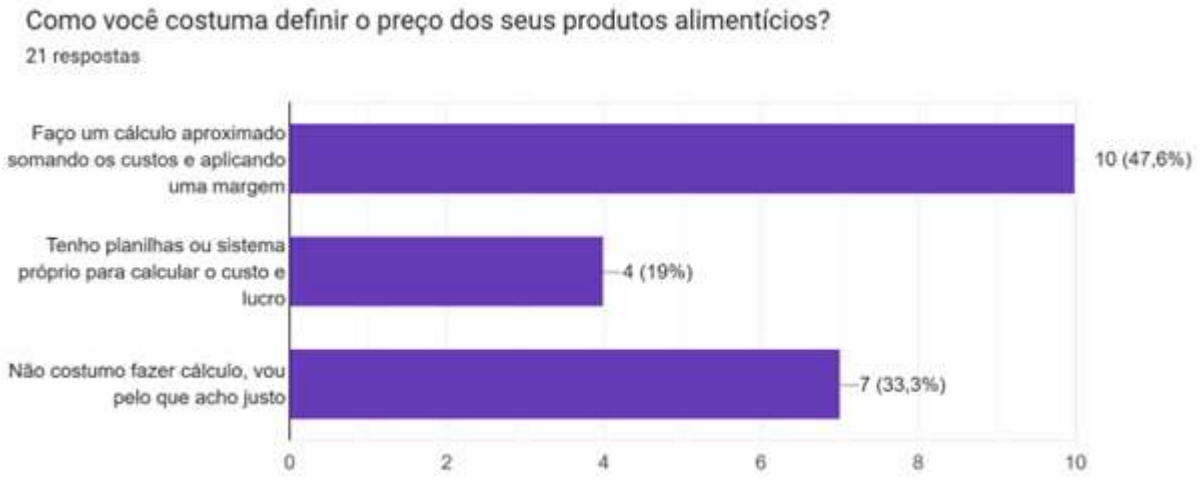
frequentemente, por surgir como alternativa diante de dificuldades financeiras ou da necessidade de uma renda extra. No entanto, pela falta de planejamento e conhecimento sobre práticas de gestão, acabam enfrentando dificuldades para diferenciar o que é o lucro e o que são despesas reais, comprometendo a saúde financeira do negócio.

Essa informalidade na condução financeira é recorrente em micro e pequenas empresas e foi destacada por Azevedo e Leone (2011), ao apontarem que muitos gestores atuam com base em práticas empíricas, sem planejamento estruturado, o que aumenta significativamente os riscos de falência. No contexto alimentar, essa desorganização acaba gerando baixo lucro, alto volume de desperdícios e uma percepção distorcida da real lucratividade do empreendimento.

A pesquisa de campo realizada pelo período de 1 semana, utilizando a metodologia quantitativa, obteve a participação de 21 empreendedores do setor alimentício e teve como objetivo compreender os métodos utilizados atualmente para a formação de preço de venda dos produtos.

A Figura 1 apresenta os resultados da pergunta:

Figura 1. Resultado da primeira pergunta realizada a profissionais da área



Fonte: Imagem do autor.

Dos 21 participantes, 47,6% relataram utilizar um cálculo aproximado, somando os custos e aplicando uma margem de lucro subjetiva. Outros 33,3% afirmaram não realizar nenhum tipo de cálculo, baseando o preço apenas em percepções pessoais ou comparações informais com o mercado. Apenas 19%

disseram usar planilhas ou sistemas próprios para calcular custo e lucro de forma estruturada.

Esse resultado reforça a relevância da proposta do sistema ChefPrice, que tem como um de seus principais objetivos oferecer uma ferramenta simples, precisa e automatizada para a precificação de receitas. Conforme apontado por Azevedo e Leone (2011), a informalidade na gestão financeira é um fator crítico na sobrevivência das microempresas, e a ausência de critérios objetivos para definição de preços pode levar ao subdimensionamento de gastos, margem de lucro insuficiente e, consequentemente, à inviabilidade do negócio.

Diante disso, a funcionalidade de cálculo automático de custo direto e indireto por receita, integrada à sugestão de preço de venda com base na margem desejada, se mostra promissora para apoiar decisões mais conscientes e sustentáveis por parte dos empreendedores do ramo alimentício.

Um dos principais desafios enfrentados por pequenos empreendedores desse setor é a definição correta do preço de venda de seus produtos. Essa etapa envolve variáveis como custo dos insumos, despesas indiretas e margem de lucro desejada. No entanto, muitos desses profissionais iniciam suas atividades sem conhecimento técnico sobre precificação, o que compromete diretamente a rentabilidade do negócio.

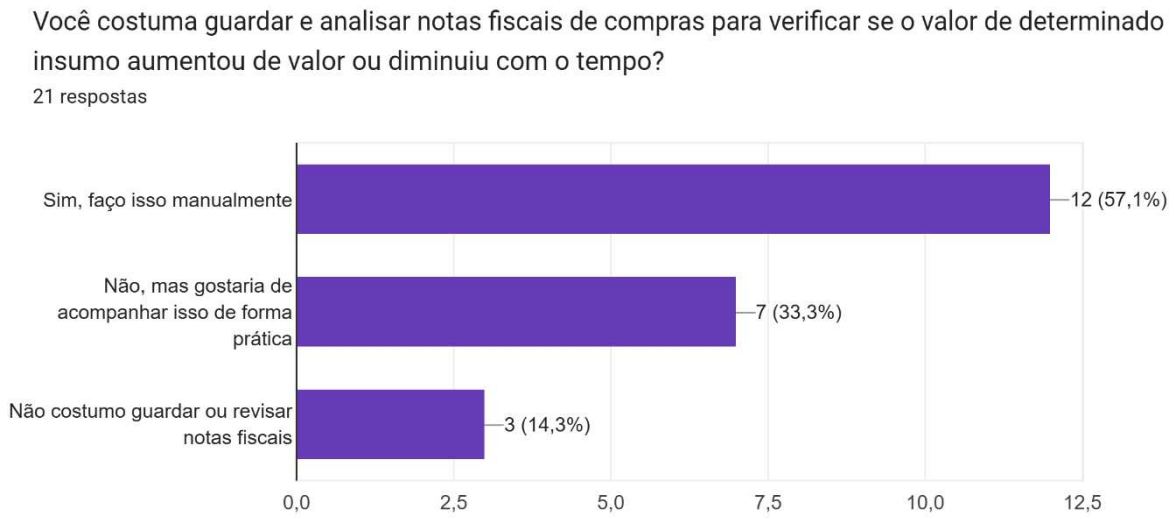
Com base nisso, esta pergunta foi elaborada com o objetivo de investigar como os participantes do setor alimentício realizam a precificação dos seus produtos. O intuito é compreender se utilizam algum tipo de método estruturado ou se ainda dependem de estratégias informais, como a percepção subjetiva de valor. Os resultados dessa pergunta são fundamentais para validar a proposta do sistema ChefPrice, que visa automatizar e profissionalizar o processo de formação de preço de venda.

Além da precificação correta, um dos fatores que mais influenciam diretamente no lucro de pequenos empreendimentos alimentícios é a escolha dos fornecedores e o acompanhamento do valor pago pelos insumos ao longo do tempo. A busca constante por locais com melhores preços é fundamental para garantir uma margem de lucro saudável, especialmente em um mercado competitivo e de baixa escala de produção.

Dessa forma, a segunda pergunta foi formulada com o objetivo de compreender como os empreendedores analisam a variação de preços dos

insumos utilizados, se costumam guardar e revisar suas notas fiscais, e se possuem o hábito de comparar valores praticados por diferentes fornecedores. A resposta a essa pergunta é essencial para validar uma das principais funcionalidades do sistema ChefPrice: a leitura automática de notas fiscais.

Figura 2. Resultado da segunda pergunta realizada a profissionais da área



Fonte: Imagem do autor

Na figura 2, os dados revelam que a maioria dos empreendedores já reconhece a importância de observar o histórico de preços como parte do processo de controle de gastos, mas ainda dependem de métodos manuais, que demandam tempo, organização e podem conter falhas humanas.

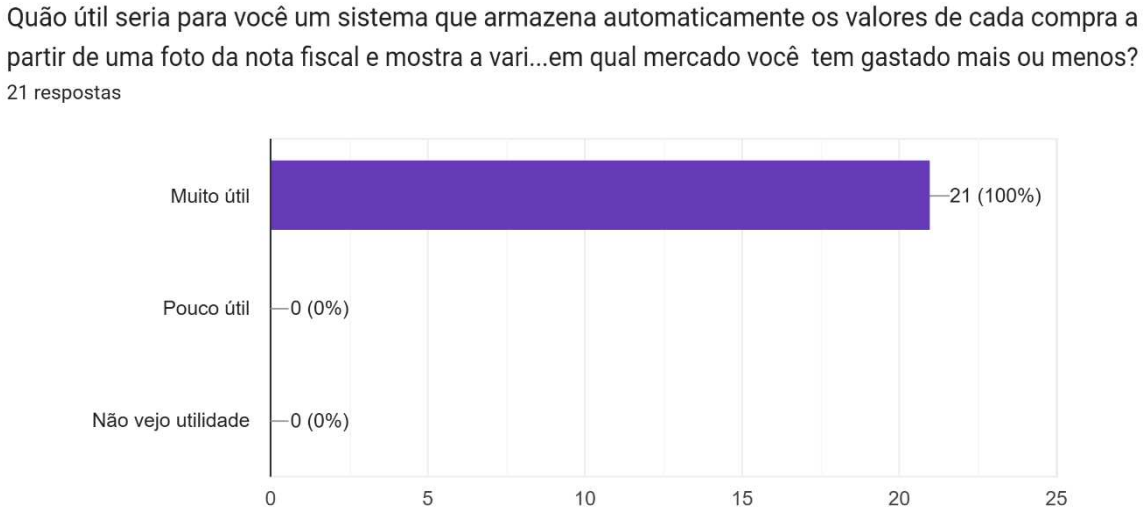
A automatização desse processo representa um grande diferencial competitivo, especialmente para negócios com pouca estrutura administrativa. Com base nisso, o ChefPrice propõe a leitura automática de notas fiscais por imagem, identificando insumos comprados, valores pagos e datas de aquisição. Com esses dados organizados, o sistema irá gerar gráficos comparativos por produto, por fornecedor e por período, ajudando o empreendedor a decidir onde comprar e quando comprar para maximizar sua margem de lucro.

De acordo com Silva (2016), o uso de inteligência artificial em documentos não estruturados, como notas fiscais, permite automatizar processos burocráticos e aumentar a confiabilidade das informações, aplicando técnicas como o *Named Entity Recognition* para extrair nomes de produtos, quantidades e valores de forma inteligente. Além disso, Bazzotti e Garcia (2003) destacam que sistemas de

informação gerencial fortalecem a atuação do empreendedor ao fornecerem informações úteis e rápidas para decisões estratégicas.

Portanto, o resultado dessa pergunta valida diretamente a necessidade e a aceitação de uma solução como o ChefPrice, que substitui o esforço manual pela tecnologia, gerando ganhos de tempo, organização e controle financeiro real sobre as compras realizadas.

Figura 3. Resultado da terceira pergunta realizada a profissionais da área



Fonte: Imagem do autor.

Na figura 3, é possível observar o resultado, que foi categórico: 100% dos participantes consideraram essa funcionalidade muito útil. Essa unanimidade indica não apenas a aceitação da proposta tecnológica do sistema ChefPrice, como também evidencia uma necessidade latente e não atendida no dia a dia dos pequenos empreendedores alimentícios.

Esse dado reforça que o público-alvo reconhece a importância de automatizar tarefas rotineiras que exigem tempo e organização, como o controle dos preços pagos por insumos em diferentes momentos e locais. A leitura automática da nota fiscal se mostra como uma solução prática e desejada, que elimina o esforço manual, reduz o risco de erros e oferece uma base real e objetiva para tomadas de decisão mais estratégicas.

Além disso, Bazzotti e Garcia (2003) defendem que os sistemas de informação gerencial são ferramentas essenciais para transformar dados brutos

em informações relevantes para o planejamento e a tomada de decisão. No caso do ChefPrice, essa conversão se materializa nos gráficos gerados a partir das notas lidas, que ajudam o usuário a visualizar como os preços de insumos estão variando ao longo dos meses, o que pode influenciar diretamente sua lucratividade.

Portanto, o resultado dessa pergunta não apenas válida a implementação da funcionalidade de leitura de nota fiscal por imagem com inteligência artificial, como também demonstra o alinhamento entre a proposta do sistema e as dores reais dos seus futuros usuários. Essa resposta unânime é um forte indicativo de que o ChefPrice pode representar um diferencial relevante no cotidiano de pequenos empreendedores do setor alimentício, oferecendo controle, clareza e praticidade em processos até então manuais e negligenciados.

3.1. Planejamento do Projeto

Para o desenvolvimento do ChefPrice, além da escolha cuidadosa das tecnologias utilizadas, uma das principais preocupações da equipe foi garantir uma arquitetura bem estruturada das funcionalidades, especialmente aquelas que apresentam dependência entre si.

Nesse sentido, o sistema foi pensado para que o usuário tenha acesso completo a todas as funcionalidades de forma fluida, organizada e intuitiva. A seguir, as telas principais, que representam os pontos centrais da navegação, os diagramas de banco de dados, essenciais para a visualização da estrutura das entidades, atributos e relacionamentos, trechos do código onde estão as regras de negócio e a metodologia ágil adotada durante o processo de desenvolvimento.

4. DESENVOLVIMENTO DO PROJETO

A escolha das ferramentas considerou fatores como escalabilidade, desempenho, facilidade de manutenção, integração com recursos de inteligência artificial e acessibilidade para o usuário final.

Front-end

- **Angular 16**: Utilizado para a construção da interface do usuário de forma responsiva, interativa e modular. Por ser um framework robusto, permite um desenvolvimento escalável e organizado com componentização.
- **Bootstrap**: Utilizado para estilização dos componentes com foco em responsividade e experiência visual amigável, otimizando o tempo de desenvolvimento.
- **Whimsical**: Utilizado para prototipação da interface e validação da experiência do usuário antes da implementação.
- **Amazon S3 Bucket:** Cria um *bucket* S3 para armazenar e servir arquivos estáticos do Front-end (Angular). O *bucket* é configurado para permitir destruição forçada e recebe tags de identificação.
- **Amazon S3 Website Hosting:** Configura o *bucket* S3 para funcionar como um site estático, definindo o documento de índice e de erro como index.html.
- **S3 Public Access Block** :Define regras de acesso público ao *bucket*, permitindo acesso público para que o site estático possa ser acessado por qualquer usuário.
- **S3 Bucket Policy**: Aplica uma política ao *bucket* S3 para liberar leitura pública dos arquivos, permitindo que o Front-end seja acessado via HTTP.

API

- **Laravel (PHP)**: Framework escolhido por sua robustez, flexibilidade, segurança e recursos integrados para autenticação, rotas e integração com banco de dados. Por oferecer algumas funcionalidades já prontas, facilita o desenvolvimento e diminui a quantidade de linhas do código, tornando-o mais limpo, e facilitando futuras manutenções.
- **API RESTful**: Estrutura usada para comunicação entre Front-end e API. Por seguir padrões de métodos HTTP, a API RESTful facilita essa comunicação e a integração da interface e da parte lógica, que carrega a regra de negócios da aplicação, permitindo que os dados trafeguem de forma segura e organizada. Além disso garante que o sistema esteja preparado para futuras integrações.

- **EC2 Instance:** Cria uma instância EC2 (Amazon Linux 2) para hospedar a API feita em PHP (Laravel). O *script* de inicialização instala dependências, configura o Apache e prepara o ambiente Laravel.

Banco de Dados

- **Microsoft SQL Server**: Banco relacional utilizado para armazenar todas as informações da aplicação, como cadastros de insumos, receitas e notas fiscais. A escolha foi baseada em confiabilidade, performance e compatibilidade com sistemas corporativos e Amazon RDS.
- **Amazon RDS**: O serviço de banco de dados relacional da Amazon foi utilizado para escalabilidade e disponibilidade do SQL Server, de forma que os mesmos recursos serão acessados independente de quem e de onde forem acessados. Isso garante a integridade dos dados gravados e confiabilidade do sistema.

Inteligência Artificial

- **API Gemini:** Um serviço de inteligência artificial da Google baseado em modelos de linguagem generativa (GenAI). Essa IA é especializada no processamento de linguagem natural e possui capacidade multimodal, permitindo interpretar imagens e textos simultaneamente.

Versionamento e Colaboração

- **GitHub**: Repositório utilizado para versionamento do código e trabalho colaborativo entre os desenvolvedores do sistema, garantindo controle de alterações e integração contínua.

Metodologia ágil

- **Scrum**: Metodologia escolhida pela equipe visando o controle e gerenciamento do projeto. Organizado em reuniões semanais e *sprints* com duração de duas semanas para garantir que as funcionalidades fossem entregues a tempo do prazo limite.

5. ETAPAS DE DESENVOLVIMENTO

5.1. Planejamento

Desde o início do desenvolvimento do ChefPrice, o público-alvo foi claramente definido: pequenos empreendedores do setor alimentício, como confeiteiros, doceiros, salgadeiros e produtores caseiros. Essa definição partiu da própria vivência dos idealizadores do projeto, que identificaram uma lacuna no mercado quanto à precificação eficiente de produtos alimentícios.

Com base em entrevistas exploratórias e questionários aplicados em feiras, grupos de confeitaria e redes sociais, foi possível validar que os principais desafios enfrentados por esse público giram em torno do cálculo adequado de custos, da definição de preço de venda justo e da atualização constante dos valores de insumos.

Essas interações resultaram na construção de uma persona representativa: mulheres entre 25 e 45 anos, empreendedoras individuais, com pouca familiaridade com ferramentas digitais complexas, mas alto engajamento em aplicativos móveis e redes sociais. A confirmação do perfil final do cliente foi realizada por meio de uma pesquisa de campo abrangente, que contou com a participação de aproximadamente 100 pessoas e já foi citada anteriormente. Essas pessoas foram entrevistadas e responderam a um questionário cuidadosamente elaborado, permitindo a coleta de informações relevantes para entender melhor o perfil e as necessidades do público-alvo.

A partir da coleta de dados através da pesquisa foi possível chegar a confirmação do perfil do cliente, que é microempreendedor do ramo alimentício.

5.2. Identificação de requisitos

A partir da vivência prática dos idealizadores do projeto com a produção e venda de alimentos, foram identificadas as principais necessidades enfrentadas por pequenos empreendedores na gestão de seus produtos e precificação. Essas experiências revelaram desafios recorrentes no controle de custos, organização de insumos e definição de preços justos de venda.

Com base nesse contexto real, foram elencadas as funcionalidades essenciais para o sistema:

1. Cadastro de receitas com cálculo automático de custo unitário;
2. Inclusão de gastos diretos e indiretos no cálculo final;
3. Sugestão de preço de venda com base em margem de lucro;
4. Cadastro e atualização de materiais com histórico de preços;
5. Lista de compras vinculada às receitas;
6. Importação automática de valores via leitura de nota fiscal (com auxílio de IA);
7. Dashboard com análise de variação de preços e gráficos de desempenho financeiro.

Esses requisitos foram fundamentais para orientar o desenvolvimento das telas, fluxos e regras de negócio do ChefPrice, com o objetivo de entregar uma ferramenta prática, funcional e de alto valor para o público empreendedor.

5.3. Levantamento de recursos e custos

5.3.1 Equipe

O desenvolvimento do sistema ChefPrice contou com uma equipe multidisciplinar, comprometida em entregar uma solução prática, funcional e tecnicamente robusta, voltada à realidade de pequenos empreendedores do setor alimentício. Cada integrante contribuiu com competências específicas e complementares, garantindo que o sistema fosse desenvolvido com qualidade, usabilidade e alinhamento às necessidades reais dos usuários finais.

Abaixo, apresenta-se a composição da equipe e suas respectivas atribuições:

Geovana Vieira de Sousa - Idealizadora do projeto. Foi responsável pelo levantamento de requisitos, condução de testes com usuários reais, análise dos dados da pesquisa de campo e apoio no desenvolvimento do Front-end, além da elaboração e estruturação da documentação técnica. A concepção do ChefPrice surgiu a partir de sua vivência pessoal como confeiteira, o que permitiu transformar

um desafio cotidiano em uma solução inovadora, com potencial de impacto direto sobre a gestão de custos e precificação de produtos caseiros.

Ligia Piva de Santana - Responsável pelo desenvolvimento da API utilizando o framework Laravel. Atuou na modelagem do banco de dados com SQL Server, na definição da arquitetura geral do sistema e na integração entre as camadas Front-end e API. Sua atuação assegurou a robustez da estrutura lógica e a escalabilidade da aplicação para futuras evoluções.

Thiago Sansalone D'Alessandro - Responsável pela implementação da API RESTful em Laravel e pela integração da funcionalidade de leitura automática de notas fiscais com apoio de inteligência artificial (IA), permitindo a extração de dados de forma automatizada e precisa. Participou ativamente da modelagem de dados, contribuindo para a organização lógica das entidades e relacionamentos do sistema. Também desempenhou um papel essencial na definição da arquitetura da solução e no processo de *deploy* do sistema em ambiente de testes, garantindo estabilidade e segurança na publicação da aplicação.

Professor Alessandro - Atuou como orientador técnico do projeto, oferecendo suporte em momentos estratégicos de tomada de decisão. Contribuiu com a validação das funcionalidades desenvolvidas, revisão técnica da aplicação e sugestões voltadas à melhoria da experiência do usuário (UX). Também auxiliou na organização do cronograma de desenvolvimento, orientando a equipe quanto ao uso de boas práticas de codificação e à estruturação adequada do sistema dentro dos padrões acadêmicos e de mercado.

5.3.2. Tecnologias

Para o desenvolvimento do ChefPrice, foram selecionados recursos tecnológicos modernos e acessíveis, com o objetivo de garantir uma solução escalável, segura e de alto desempenho, sem comprometer o orçamento do projeto.

A equipe utilizou os seguintes recursos:

5.3.2.1. Ferramenta de desenvolvimento

Visual Studio Code, escolhida por ser leve, altamente personalizável e compatível com as tecnologias adotadas;

5.3.2.2. Linguagens

- PHP para a API, responsável pelas regras de negócio e integração com o banco de dados;
- Angular para o Front-end, garantindo uma interface dinâmica, responsiva e moderna;
- TypeScript, HTML5 e CSS3 complementando a construção e o comportamento visual da interface;

5.3.2.3. Banco de Dados

MySQL, hospedado via AWS RDS, para garantir disponibilidade, segurança e escalabilidade da base de dados;

5.3.2.4. Ferramentas de Design

Whimsical, utilizado para criação de *wireframes* e protótipos de média fidelidade, facilitando a definição das interfaces antes do desenvolvimento;

5.3.2.5. Versionamento

Git, com repositórios hospedados no GitHub, para controle de versões, rastreabilidade de alterações e colaboração eficiente entre os membros da equipe;

5.3.2.6. Custos

O desenvolvimento do projeto foi orientado pela premissa de manter baixos os custos operacionais, explorando ao máximo recursos gratuitos disponíveis no mercado. Foram utilizados planos educacionais e serviços com políticas de uso gratuito, em especial os ofertados pela plataforma AWS (*Amazon Web Services*), que hospedou toda a arquitetura do sistema.

Durante a fase de desenvolvimento, os únicos custos efetivamente incorridos referem-se à utilização de serviços da AWS que excederam as cotas gratuitas. Esses custos totalizaram US$ 10, o que equivale a R$ 55,10 na cotação do dólar vigente à época. O valor foi dividido igualmente entre os integrantes do grupo, não representando impacto financeiro significativo para o projeto.

6. ANÁLISE DE RNF E RF

6.1. Requisitos Funcionais

O levantamento de requisitos foi realizado com base nas necessidades do público-alvo, identificado como microempreendedores do ramo alimentício que buscam melhorar o controle de lucros de suas vendas. A partir de pesquisas e entrevistas, foram definidos os principais requisitos que o sistema deveria atender para oferecer uma solução completa e eficiente.

Esses requisitos foram definidos para garantir que o sistema atendesse às necessidades reais dos usuários e oferecesse funcionalidades modernas e acessíveis.

6.1.1. Usuário x Sistema

Os requisitos funcionais representam o conjunto de funcionalidades que o sistema ChefPrice deve oferecer para atender aos objetivos do projeto. Com base na experiência prática dos idealizadores e na análise das dores do público-alvo, foram definidos os seguintes requisitos principais:

- **Cadastro de receitas** culinárias, com cálculo automático do custo unitário com base nos insumos utilizados;

- **Integração com leitura de nota fiscal via IA**, para extração automática dos preços dos produtos com base em códigos de barras;
- **Criação de listas de compras vinculadas às receitas**, com cálculo automático das quantidades totais necessárias de cada insumo;
- **Cálculo automático do preço de venda sugerido**, com base no rendimento da receita e nas porcentagens de gastos diretos e indiretos informadas pelo usuário;
- **Histórico de receitas cadastradas**, com opção para editar, excluir e visualizar os detalhes de cada item;
- **Autenticação de usuários** e gerenciamento de sessões, garantindo que cada empreendedor tenha acesso apenas aos seus próprios dados.

6.1.2. Administrador x Sistema

O usuário administrador terá as mesmas funcionalidades que os usuários que não são administradores com algumas funcionalidades de gerenciamento vinculada.

- **Cadastro de receitas** culinárias, com cálculo automático do custo unitário com base nos insumos utilizados.
- **Integração com leitura de nota fiscal via IA**, para extração automática dos preços dos produtos com base em códigos de barras.
- **Criação de listas de compras** vinculadas às receitas, com cálculo automático das quantidades totais necessárias de cada insumo.
- **Cálculo automático do preço de venda** sugerido, com base no rendimento da receita e nas porcentagens de gastos diretos e indiretos informadas pelo usuário.
- **Histórico de receitas cadastradas**, com opção para editar, excluir e visualizar os detalhes de cada item.
- **Autenticação de usuários** e gerenciamento de sessões, garantindo que cada empreendedor tenha acesso apenas aos seus próprios dados.

- **Cadastro, edição e exclusão de materiais**/insumos utilizados nas receitas.
- **Cadastro, edição e exclusão de categorias** para organização dos materiais.
- **Visualização da lista de todos os usuários** cadastrados no sistema.
- **Atribuição ou remoção de permissões** de administrador a outros usuários.

6.2. Requisitos Não Funcionais

Os requisitos não funcionais descrevem as qualidades esperadas do sistema ChefPrice, assegurando que sua operação seja segura, eficiente e compatível com diferentes ambientes de uso. Os principais requisitos não funcionais são:

- **Usabilidade:** A interface deve ser clara, intuitiva e acessível, permitindo que mesmo usuários com baixa familiaridade tecnológica consiga utilizar o sistema com facilidade;
- **Desempenho**: O sistema deve operar com fluidez e agilidade, inclusive em dispositivos com menor capacidade de processamento;
- **Compatibilidade:** A aplicação deve funcionar corretamente nos principais navegadores e dispositivos (computadores, tablets e smartphones);
- **Escalabilidade:** A arquitetura deve permitir o crescimento contínuo do número de usuários e dados sem comprometer o desempenho;
- **Manutenibilidade:** O código-fonte deve ser modularizado, limpo e documentado, facilitando a correção de erros, adição de novas funcionalidades e atualizações futuras.

7. AVALIAÇÃO DE VIABILIDADES

Os recursos tecnológicos disponíveis são adequados para o desenvolvimento do software. A equipe possui experiência suficiente para operar

essas ferramentas, garantindo o desenvolvimento eficiente e a implementação das funcionalidades planejadas.

Os custos envolvidos são minimizados, pois as tecnologias empregadas são amplamente acessíveis, muitas delas gratuitas ou de baixo custo. A equipe de trabalho já está alocada, reduzindo a necessidade de investimentos adicionais em mão de obra.

Com uma equipe pequena, porém bem-organizada, o projeto tem condições de ser desenvolvido dentro de um cronograma adequado.

8. CONSTRUÇÃO DA APLICAÇÃO

A partir da vivência prática de um dos idealizadores do projeto com a produção e venda de alimentos, foram identificadas as principais necessidades a serem sanadas pelo sistema. Essa experiência revelou desafios recorrentes no controle de custos, organização de insumos e definição de preços justos de venda.

Com base nesse contexto real, foram elencadas as funcionalidades essenciais para o sistema:

1. Cadastro de receitas com cálculo automático de custo unitário;
2. Inclusão de gastos diretos e indiretos no cálculo final;
3. Sugestão de preço de venda com base em margem de lucro;
4. Cadastro e atualização de materiais com histórico de preços;
5. Lista de compras vinculada às receitas;
6. Importação automática de valores via leitura de nota fiscal (com auxílio de IA);

Esses requisitos foram fundamentais para orientar o desenvolvimento das telas, fluxos e regras de negócio do ChefPrice, com o objetivo de entregar uma ferramenta prática, funcional e de alto valor para o público empreendedor.

9. PROCESSOS

O Diagrama de **Casos de Uso UML** foi essencial para a modelagem dos processos que compõem o sistema **ChefPrice**. Ele tem como objetivo representar, de forma clara e objetiva, as funcionalidades principais do sistema,

os atores envolvidos e as interações entre cada ator e o sistema, facilitando o entendimento do escopo funcional da aplicação.

Diferente de diagramas que detalham a lógica interna, o diagrama de casos de uso foca nas ações percebidas externamente, isto é, como os usuários interagem com o sistema. Ele serve como uma ferramenta fundamental para garantir a coerência entre os requisitos levantados e o que será implementado.

Este diagrama foi modelado com base nos requisitos funcionais definidos previamente, destacando as ações mais relevantes para o funcionamento da aplicação.

Atores envolvidos:

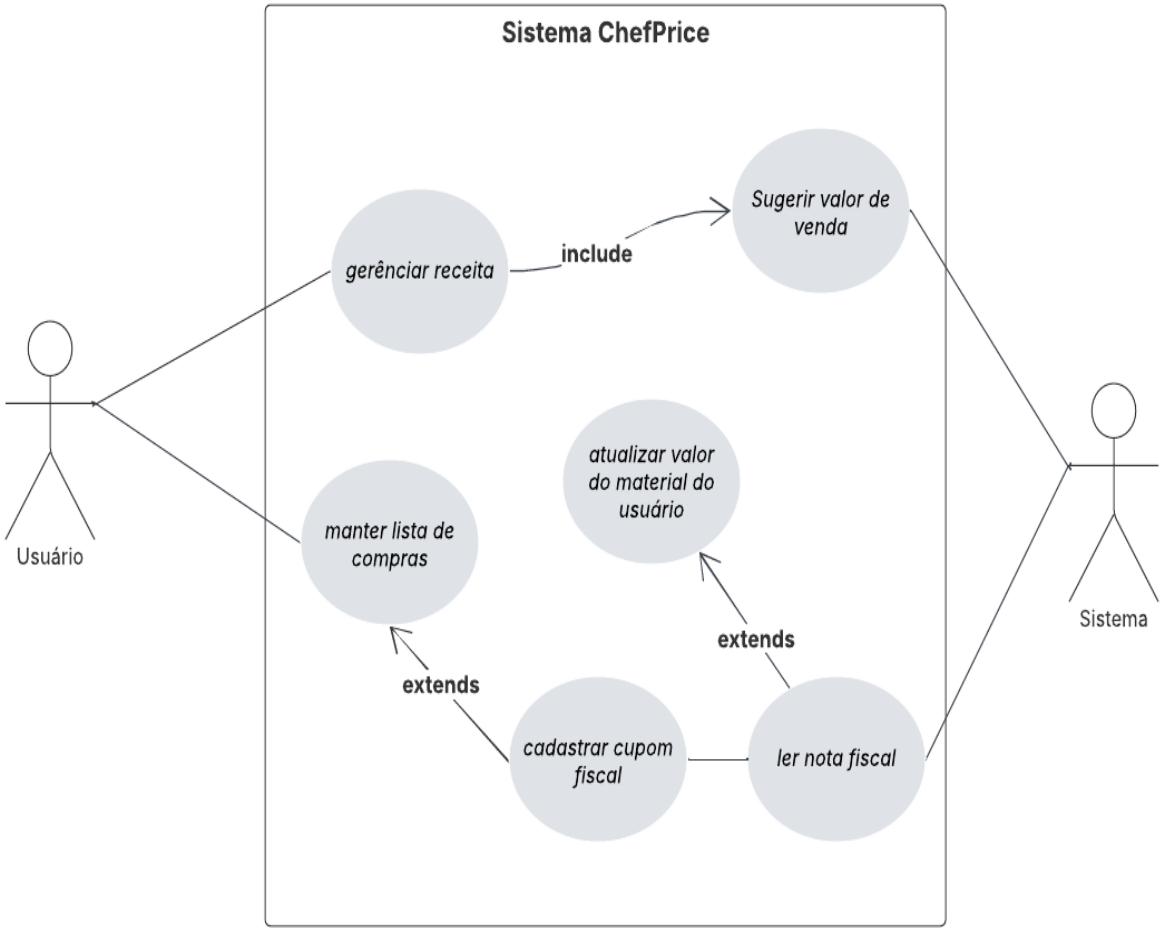
- **Usuário** - Representa o pequeno empreendedor ou confeiteiro que utiliza o sistema para cadastrar e gerenciar suas receitas, insumos e custos.
- **Administrador** - Responsável pela gestão de dados gerais do sistema, controle de contas de usuários e suporte administrativo.
- **Sistema** (ator secundário ou entidade interna) - Representa a execução dos processos automatizados e das regras de negócio definidas na API (opcionalmente pode ser omitido no diagrama se considerado interno).

Casos de Uso:

1. Fazer login;
2. Fazer cadastro (novo usuário);
3. Cadastrar receita;
4. Editar receita;
5. Deletar receita;
6. Calcular valor de venda da receita;
7. Ver lista de compras da receita;
8. Cadastrar material (insumo);
9. Editar material;
10. Ver histórico de edições de material;
11. Cadastrar lista de compras;
12. Visualizar lista de compras;
13. Cadastrar cupom fiscal (nota fiscal com leitura via IA);

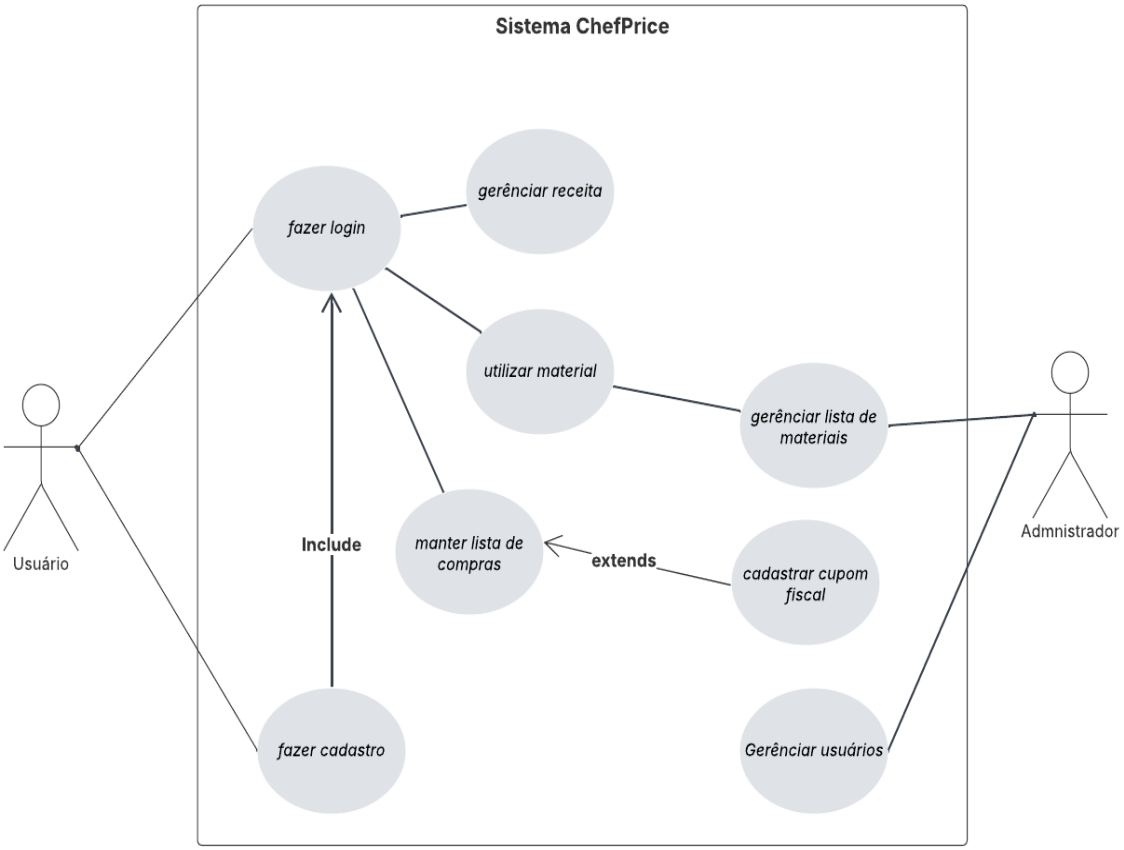
14. Visualizar histórico de receitas.

Figura 4. Diagrama de Casos de Uso: Interação do Usuário com o Sistema



Fonte: Imagem do autor.

Figura 5. Diagrama de Casos de Uso: interação do Administrador com o Sistema



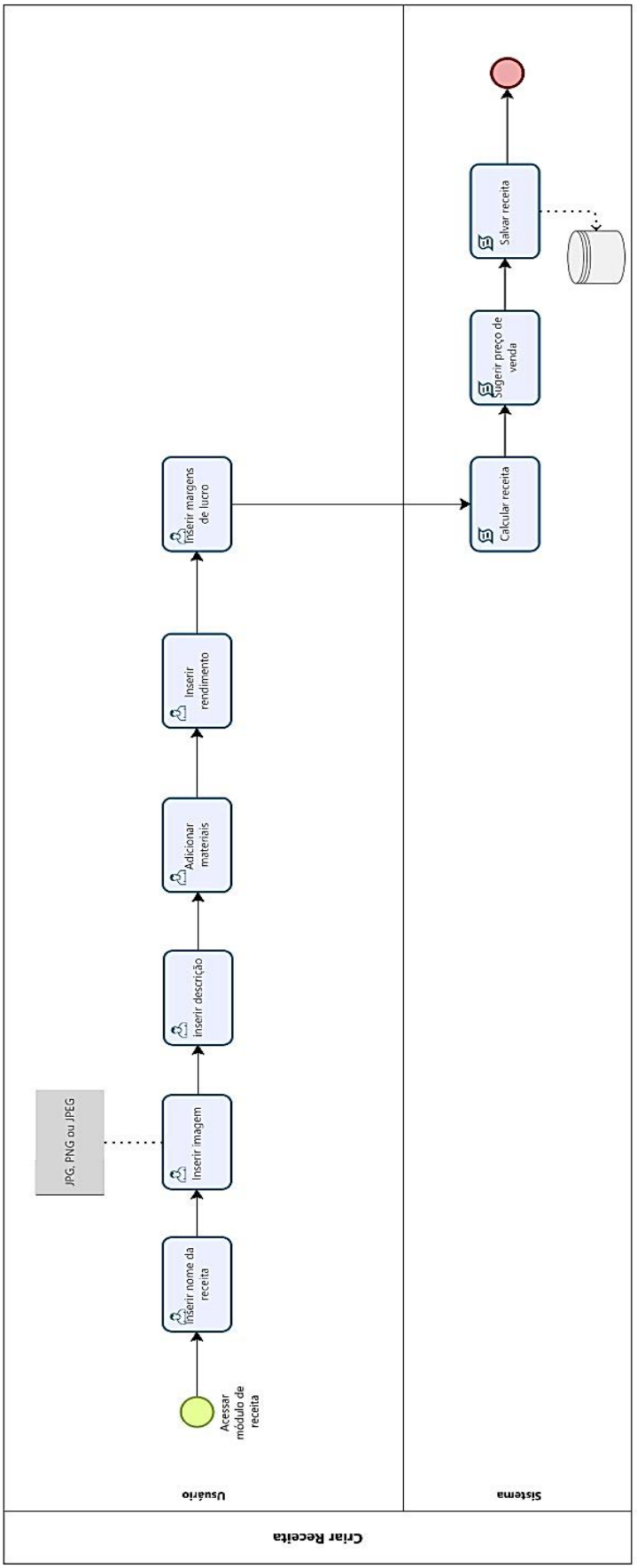
Fonte: Imagem do autor.

9.1. Modelagem BPMN

A modelagem BPMN se mostra extremamente útil quando se trata de otimizar a comunicação, clarear responsabilidades e promover melhorias contínuas nos processos organizacionais. Por meio de diagramas padronizados e visualmente intuitivos, as equipes conseguem identificar gargalos, eliminar redundâncias e alinhar todos os atores envolvidos em uma linguagem comum.

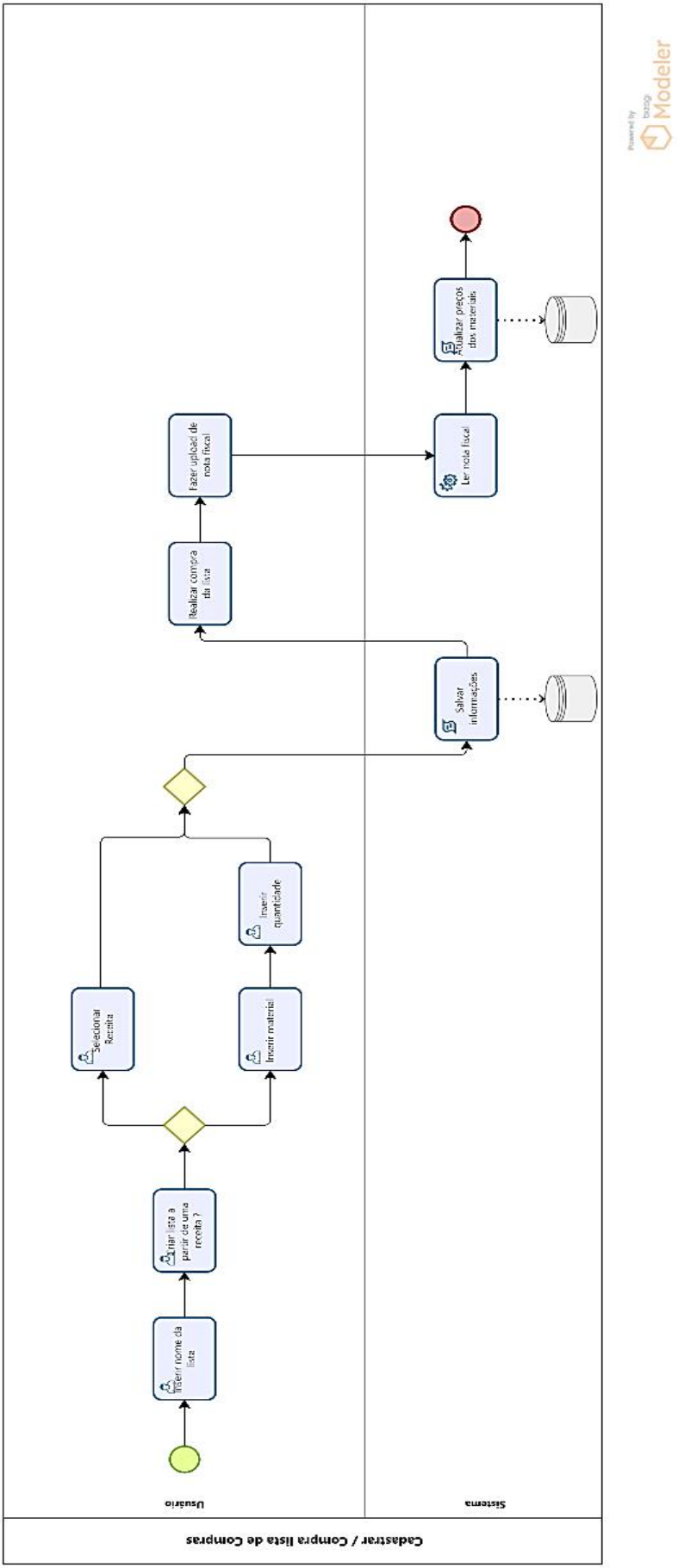
As figuras 6, 7 e 8 são apresentados sobre os BPMN dos processos de criação de receita, cadastro e compra de lista de compras, e de criação de lista de compras, respectivamente.

Figura 6. BPMN do processo criar receita.



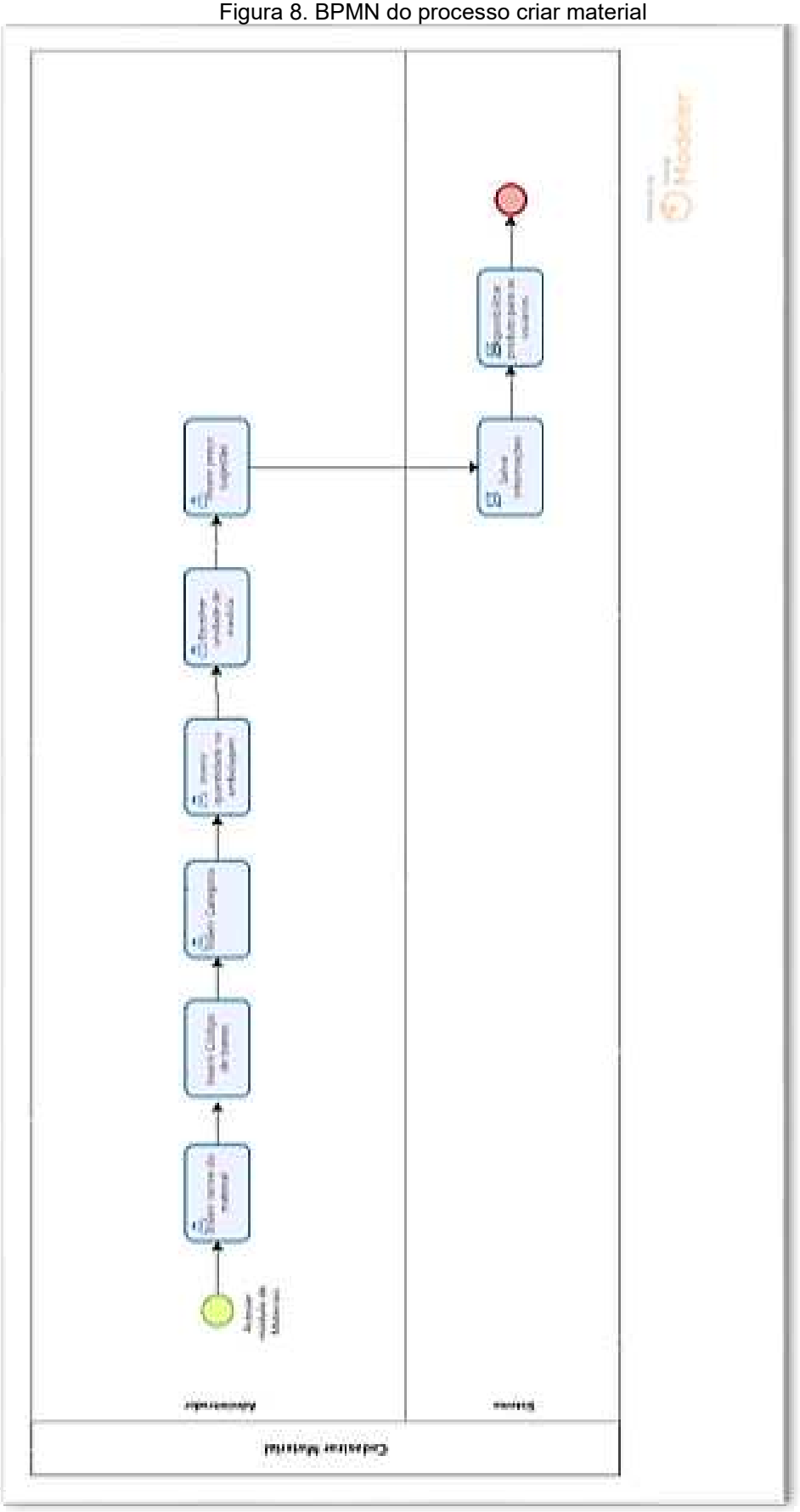
Fonte: Imagem do autor.

Figura 7. BPMN do processo Cadastrar /Comprar lista de compras



Fonte: Imagem do autor.

Figura 8. BPMN do processo criar material



Fonte: Imagem do autor.

10. PROJETO DE BANCO DE DADOS

O planejamento do banco de dados do ChefPrice foi desenvolvido com o objetivo de estruturar e armazenar todas as informações essenciais para o funcionamento do sistema, especialmente relacionadas à gestão de receitas culinárias, insumos (materiais), preços, imagens, usuários e suas interações. Durante as reuniões de definição técnica, a equipe priorizou a organização, rastreabilidade e integridade dos dados. As entidades e relacionamentos foram estruturados de modo a garantir consistência na precificação, histórico de alterações, e escalabilidade futura da aplicação.

10.1. Estruturação das Principais Entidades

- **Usuários** - Responsável por armazenar os dados pessoais do usuário, como nome, e-mail e senha (com confirmação e verificação). Cada usuário é único no sistema e está associado às suas próprias receitas, listas e registros de atividade.
- **Receitas -** Entidade central do sistema, representa as produções culinárias. Contém atributos como título, rendimento, tempo de preparo, status (público ou privado), e data de criação. Cada receita pertence a um único usuário e pode estar relacionada a diversos ingredientes, imagens, etapas e tags.
- **Ingredientes e Produtos** - O sistema diferencia produtos do estoque (entidade Produtos) e os ingredientes efetivamente usados em receitas (entidade Ingredientes_Receitas). Essa separação permite rastrear quais insumos foram utilizados, em que quantidade, unidade de medida e preço, possibilitando o cálculo automático do custo por receita.
- **Histórico de Preços dos Produtos** - Para rastrear alterações de valor ao longo do tempo, a entidade Produtos_Historicos registra a evolução dos preços de cada produto associado a um usuário, permitindo a construção de gráficos de variação e controle de reajustes.

- **Etapas da Receita -** A entidade Etapa_Receitas organiza o passo a passo de cada receita, permitindo que o sistema apresente o modo de preparo de forma clara e sequencial.
- **Imagens e Tipos de Imagem** - Cada receita pode conter uma ou mais imagens. A entidade Imagens armazena os arquivos enviados, com referência ao caminho do arquivo e tipo (armazenado na entidade Tipo_Imagens), além de controle de privacidade e data de criação.
- **Tags e Associação de Tags às Receitas** - Por meio da entidade Receita_Tags_Assoc, o sistema permite que receitas sejam categorizadas com múltiplas tags, facilitando a filtragem e a organização dos dados.
- **Sessões e Tokens de Acesso** - Para garantir a segurança, o banco possui entidades dedicadas ao controle de sessões (Sessoes) e tokens de autenticação (Tokens_Acesso_Pessoal e Tokens_Reset_Senha), permitindo rastrear atividades e controlar acessos.

10.2. Relacionamentos principais

- Usuário possui muitas receitas, produtos, imagens e sessões.
- Cada receita possui muitos ingredientes, etapas, imagens e tags.
- Cada produto pode aparecer em várias receitas, com registro do histórico de preços ao longo do tempo.
- A leitura de cupons fiscais atualiza os valores registrados nos produtos e reflete diretamente nas receitas.

Esse modelo de dados foi projetado para dar suporte completo às funcionalidades do ChefPrice, como o cálculo automatizado de custo por unidade e a leitura de notas fiscais via IA. Ele garante não apenas o armazenamento eficiente, mas também a integridade e consistência das informações que alimentam os módulos do sistema.

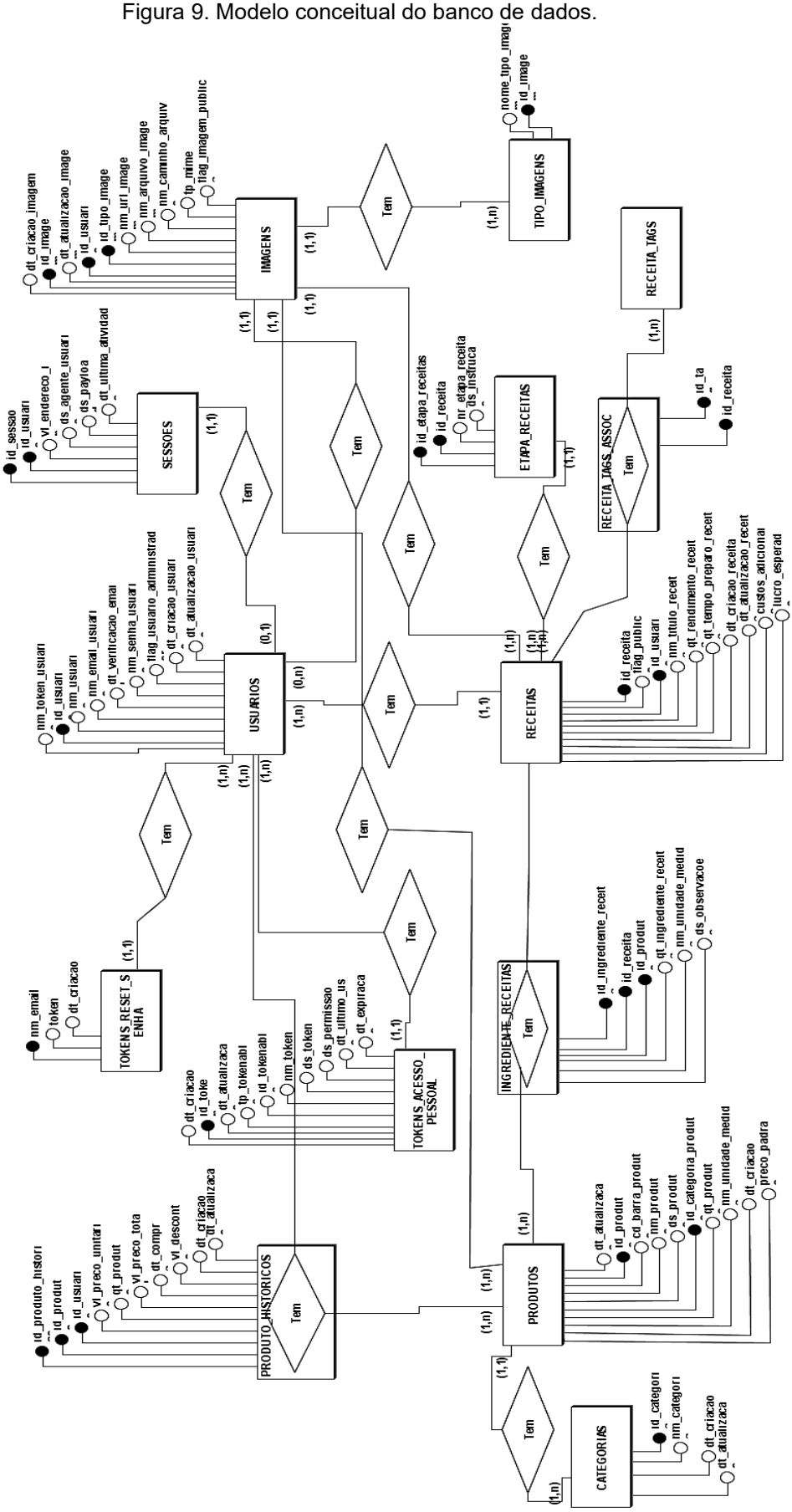
Com isso, foi possível transformar as ações dos usuários (cadastro de receitas, edição de materiais, upload de imagens etc.) em dados rastreáveis e estruturados, assegurando a base necessária para que gráficos, cálculos e relatórios funcionem com precisão e confiabilidade.

10.3. Modelo conceitual

O Modelo Conceitual é o ponto de partida do planejamento. Ele representa os dados de forma abstrata, focando nas entidades (o que você vai armazenar, como "Usuários", "Produtos", "Receitas") e nos relacionamentos entre elas (como um "Usuário" cria uma "Receita").

A Figura 9 é apresentado sobre o modelo conceitual do banco de dados:

Figura 9. Modelo conceitual do banco de dados.



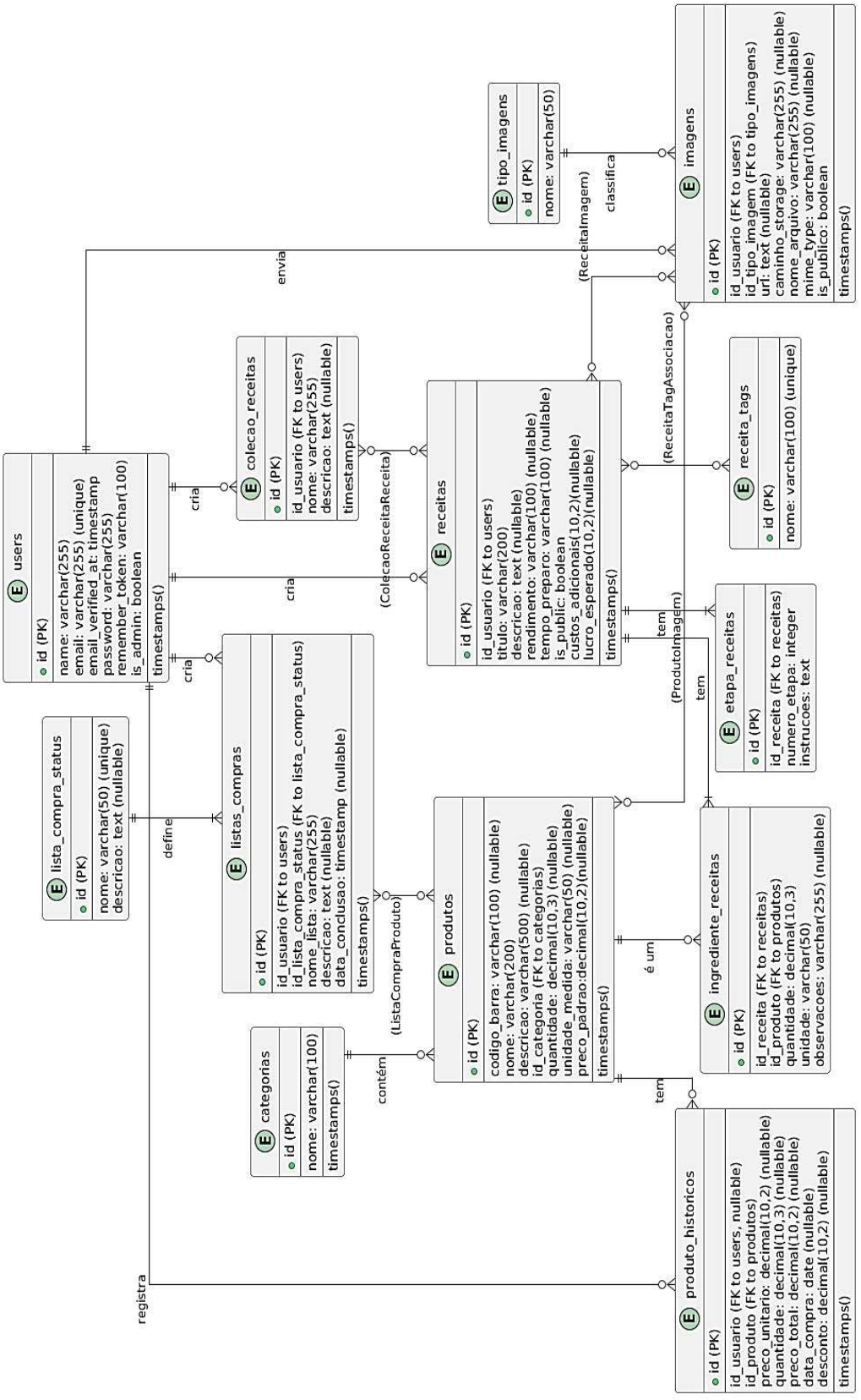
Fonte: Imagem do autor.

10.4. Modelo lógico

O modelo lógico é a continuação do planejamento iniciado no modelo conceitual. Ele representa os dados de forma mais detalhada, estruturando as entidades em tabelas, com seus respectivos atributos (como colunas) e definindo os tipos de dados, chaves primárias e estrangeiras. Nessa etapa, os relacionamentos são formalizados, e começam a ser pensadas as regras de integridade e como as informações se conectam logicamente no banco. Por exemplo, uma tabela "Usuários" pode ter um campo "id", que se relaciona com a tabela "Receitas" por meio de uma chave estrangeira indicando qual usuário criou cada receita.

A Figura 10 é apresentado sobre o modelo lógico do banco de dados:

Figura 10. Modelo lógico do banco de dados



Fonte: Imagem do autor.

11. PROJETO TÉCNICO

O sistema **ChefPrice** foi desenvolvido com uma base tecnológica sólida, utilizando ferramentas e linguagens modernas que garantem **eficiência, segurança e acessibilidade** para pequenos empreendedores do ramo alimentício. As escolhas tecnológicas foram feitas com foco em desempenho, usabilidade e escalabilidade, assegurando que o sistema atenda às demandas atuais e possa evoluir com facilidade.

As principais tecnologias utilizadas incluem:

1. **Visual Studio Code (VS Code)**: Utilizado como ambiente de desenvolvimento integrado (IDE), o VS Code ofereceu suporte avançado à codificação em PHP, Angular, HTML, CSS e JavaScript, além de ferramentas integradas como terminal, Git e extensões específicas para frameworks web.

2. **Angular**: Framework escolhido para o desenvolvimento do Front-end, responsável por proporcionar uma **interface responsiva, moderna e modular**, com navegação fluida e reatividade em tempo real.

3. **PHP com Laravel:** Na API, foi adotado o framework Laravel, que oferece **estrutura robusta, segurança na autenticação, gerenciamento de rotas, controle de acesso, e integração com banco de dados**, facilitando a manutenção e evolução do sistema.

4. **MySQL (via AWS RDS)**: Banco de dados relacional utilizado para armazenar todas as informações do sistema de forma segura e organizada. A instância foi configurada na **nuvem AWS** para garantir **alta disponibilidade, backup automatizado e escalabilidade**.

5. **Whimsical**: Ferramenta utilizada para o design de interfaces e prototipação de alta fidelidade. Auxiliou na criação e validação visual das telas com foco na experiência do usuário (UX), possibilitando testes e iterações antes da implementação.

6. **Git e GitHub**: Utilizados como sistema de controle de versão e repositório remoto, respectivamente. O Git possibilitou rastrear cada alteração feita no código, enquanto o GitHub facilitou a colaboração da equipe e o armazenamento seguro do projeto em nuvem.

7. **Amazon S3**: Utilizado para o **armazenamento de imagens e documentos**, como as fotos de receitas e notas fiscais digitalizadas. A integração com S3 permite alta durabilidade dos arquivos e otimização do carregamento das mídias.

8. **Inteligência Artificial com OCR (Reconhecimento Óptico de Caracteres)**: A funcionalidade mais inovadora do sistema está na **leitura automática de notas fiscais**, utilizando IA para extrair dados por meio de códigos de barras (EAN). Essa informação é cruzada com os produtos cadastrados para **atualizar automaticamente os valores dos insumos**.

11.1. Implementação

11.1.1. Codificação da Aplicação

A seguir, serão exibidos os códigos da aplicação, detalhando a implementação das funcionalidades de conexão ao banco de dados MySQL, gerenciamento de login e uso de sessões para manter a continuidade dos dados do usuário ao longo da navegação.

11.1.1.1. Código de conexão

O código PHP de conexão, utilizado no sistema ChefPrice, tem como objetivo estabelecer a comunicação entre a aplicação web e o banco de dados MySQL, além de preparar o ambiente para manter sessões ativas durante a navegação do usuário. Esse processo é essencial para garantir que os dados do usuário sejam persistidos corretamente enquanto ele interage com funcionalidades como cadastro de receitas, precificação e visualização de listas de compras.

Logo no início do script, o comando *session_start()* é executado. Ele é responsável por iniciar uma sessão PHP, permitindo ao sistema armazenar temporariamente informações do usuário autenticado como nome, ID e permissões, garantindo continuidade de acesso entre diferentes páginas do sistema.

Em seguida, o código define variáveis com as credenciais de acesso ao banco de dados:

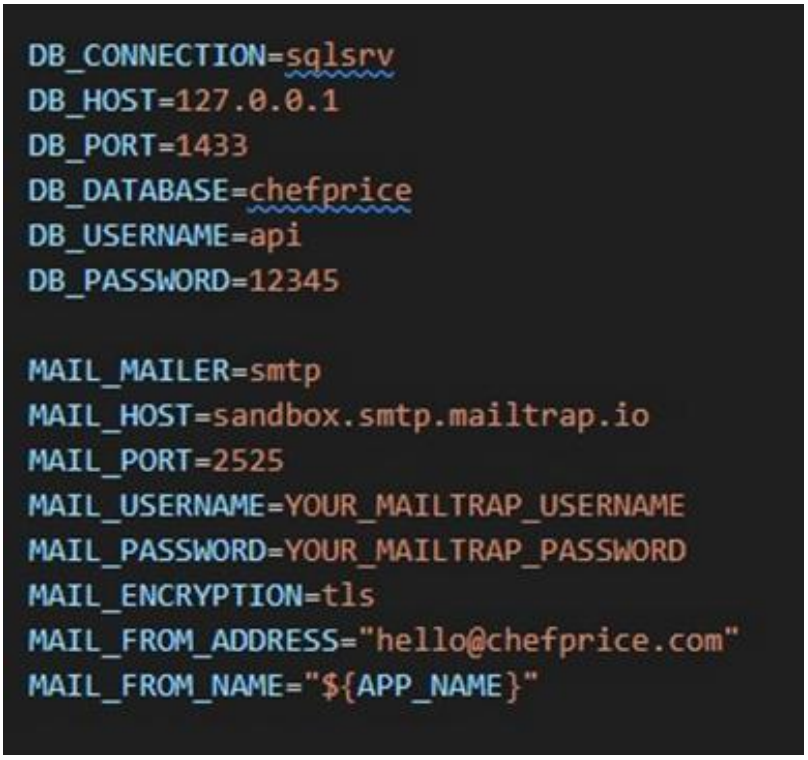
- $servername: define o endereço do servidor, normalmente configurado como "localhost" em ambientes locais;
- $username: representa o usuário do banco de dados, geralmente "root" durante o desenvolvimento;
- $password: a senha do banco, que pode ser vazia em ambientes de testes;
- $dbname: o nome do banco de dados utilizado pela aplicação, como por exemplo "chefprice".

Utilizando esses dados, é instanciado o objeto *mysql*, atribuído à variável *$conn*, que representa a conexão ativa com o banco de dados. Esse objeto será utilizado ao longo da aplicação para executar inserções, consultas, atualizações e exclusões de dados nas tabelas do sistema.

A seguir, é realizada uma verificação de sucesso da conexão. Caso a tentativa falhe, o método *connect_error* é acionado e o sistema encerra a execução com a função *die()*, exibindo uma mensagem de erro que facilita o diagnóstico. Esse trecho de código é essencial para o funcionamento do ChefPrice, pois garante que todas as funcionalidades dependentes do banco de dados — como cadastro de receitas, manipulação de materiais, leitura de nota fiscal e cálculo automático de valores — possam ocorrer de forma segura, confiável e contínua.

A figura 11 é apresentado parte do código de conexão com o banco de dados.

Figura 11. Trecho do código PHP para conexão com o banco de dados



Fonte: Imagem do autor.

11.1.1.2. Código de autenticação de usuário

Logo no início da função login, são definidas as regras de validação dos dados recebidos pela requisição. Nesse caso, os campos esperados são *email* e *password*, ambos obrigatórios e do tipo *string*, sendo o campo *email* também exigido em formato válido. Essa etapa garante que apenas dados coerentes e devidamente formatados sejam processados.

Em seguida, no método *Auth::attempt()* ocorre a verificação das credenciais informadas, se correspondem ou não a um usuário previamente cadastrado no banco de dados. Caso a verificação falhe, uma exceção de validação é lançada, retornando uma mensagem de erro relacionada ao campo *email*, indicando que não foi possível realizar a autenticação.

Se a tentativa de login for realizada com sucesso, o sistema recupera os dados do usuário autenticado por meio do *Auth::user()* e, a partir disso, gera um

novo token de sessão com o método *createToken*, o qual armazenará informações da sessão atual, como permissões, validade e identificação do usuário.

Por fim, a função retorna uma resposta JSON ao cliente, contendo uma mensagem de sucesso, o objeto do usuário autenticado e o token gerado para a sessão atual.

Permite:

- **Login** com validação de credenciais;
- **Recuperação de senha** via e-mail;
- **Cadastro** de novos usuários.

Essa tela garante a segurança de acesso, armazenando senhas e validando e-mails reais.

As figuras 12 e 13 são apresentados partes do código de autenticação do usuário.

Figura 12. Trecho do código PHP para autenticação do usuário

```php
/**
 * Login do usuário.
 */
1 reference | 0 overrides
public function login(Request $request): JsonResponse|mixed
{
    $credentials = $request->validate(rules: [
        'email' => ['required', 'string', 'email'],
        'password' => ['required', 'string'],
    ]);


    if (!Auth::attempt(credentials: $credentials)) {
        throw ValidationException::withMessages(messages: [
            'email' => [trans(key: 'auth.failed')],
        ]);
    }

    $user = Auth::user();

    $token = $user->createToken(name: 'api-token')->plainTextToken;

    return response()->json(data: [
        'message' => 'Login successful.',
        'user' => $user,
        'token' => $token
    ]);
}
```

Fonte: Imagem do autor.

Figura 13. Trecho do código PHP para verificar o usuário na autenticação

logar.php

```php
<?php
require_once "conexao.php";

// Inicia a sessão
session_start();

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $username = trim($_POST['username']);
    $senha = trim($_POST['senha']);

    // Verificar a autenticidade do usuário
    $sql = "SELECT usuario_id, senha FROM usuarios WHERE username = ?";
    $stmt = $conn->prepare($sql);
    $stmt->bind_param("s", $username);
    $stmt->execute();
    $result = $stmt->get_result();

    if ($result->num_rows > 0) {
        $row = $result->fetch_assoc();

        // Verificar a senha
        if (password_verify($senha, $row['senha'])) {
            // Autenticação bem-sucedida, define a sessão com o usuário
            $_SESSION['usuario_id'] = $row['usuario_id']; // Guarda o ID do usuário
            $_SESSION['username'] = $username; // Guarda o username

            // Redireciona para a página inicial
            header('Location: home.php');
            exit;
        } else {
            // Senha incorreta
            header('Location: login.php?erro=senha');
            exit;
        }
    } else {
        // Usuário não encontrado
        header('Location: login.php?erro=usuario');
        exit;
    }

    $stmt->close();
}

$conn->close();
?>
```

Fonte: Imagem do autor.

11.1.1.3. Código de cadastro de usuário

O trecho a seguir, detém a responsabilidade de realizar o cadastro de novos usuários no banco de dados em questão.

Percorrendo o código, é possível perceber que no primeiro trecho da função *register* são definidas as regras dos parâmetros que serão passados para a API, definindo a obrigatoriedade, o tipo dos valores, o tamanho máximo ou mínimo de caracteres, o requisito de formatação e restrições. Essas regras irão garantir que somente dados íntegros e desejados serão salvos no banco de dados.

Após todas as restrições serem definidas, na parte que as precedem, há a validação dos valores que chegaram até essa função para serem salvos. Nesse caso todos os atributos são obrigatórios, o *email* deve ser único para cada usuário e a senha deve ter no mínimo 8 caracteres. Caso os valores atendam às regras, um novo objeto do tipo *User* será criado com os parâmetros *name*, *email* e *password*, e em seguida haverá a escrita do novo usuário no banco de dados. Caso contrário, o usuário receberá um erro de dados inválidos e deverá fazer uma nova requisição.

Depois do cadastro, é enviada uma notificação para validação de *email*, validação esta que será persistida no banco de dados. Isso garante que um endereço de *email* verdadeiro e fidedigno tenha sido utilizado.

Finalmente, um *token* de sessão, que armazenará informações como características do usuário, permissões de acesso e validade da sessão, é gerado e a função irá retornar ao usuário uma mensagem de sucesso, passando também os parâmetros *user* e *token* para a sessão.

Todo o processo descrito acima detalha a segurança e robustez do código.

As figuras 14 e 15 são apresentados partes do código de cadastro do usuário.

Figura 14. Trecho do código PHP para cadastro de usuário

```
5 references | 0 implementations | You, 4 seconds ago | 1 author (You)
class AuthController extends Controller
{
    /**
     * Registro de um novo usuário.
     */
    1 reference | 0 overrides
    public function register(Request $request): JsonResponse|mixed
    {
        $validatedData = $request->validate(rules: [
            'name' => ['required', 'string', 'max:200'],
            'email' => ['required', 'string', 'email', 'max:100', 'unique:users,email'],
            'password' => ['required', 'string', 'confirmed', Password::min(size: 8)],
        ]);

        $user = User::create(attributes: [
            'name' => $validatedData['name'],
            'email' => $validatedData['email'],
            'password' => $validatedData['password'],
        ]);

        $user->sendEmailVerificationNotification();

        $token = $user->createToken(name: 'api-token')->plainTextToken;

        return response()->json(data: [
            'message' => 'Usuário registrado com sucesso.',
            'user' => $user,
            'token' => $token
        ], status: 201); // 201 Created status
    }
```

Fonte: Imagem do autor.

Figura 15. Trecho do código PHP para validar e cadastrar o novo usuário

```php
<?php
require_once "conexao.php";

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // Captura os dados enviados pelo formulário
    $nome = $_POST['nome'];
    $email = $_POST['email'];
    $senha = $_POST['senha'];
    $username = $_POST['username'];

    // Cria um hash da senha para armazenamento seguro
    $hashed_password = password_hash($senha, PASSWORD_DEFAULT);

    // Verifica se o username já existe no banco
    $sql_verificar_username = "SELECT username FROM usuarios WHERE username = ?";
    $stmt_verificar = $conn->prepare($sql_verificar_username);
    $stmt_verificar->bind_param("s", $username);
    $stmt_verificar->execute();
    $stmt_verificar->store_result();

    if ($stmt_verificar->num_rows > 0) {
        // Username já existe, exibe mensagem de erro
        echo "<h3>O username '$username' já está em uso. Por favor, escolha outro.</h3>";
    } else {
        // Prepara a consulta para inserir o novo usuário
        $sql = "INSERT INTO usuarios (nome, email, senha, username) VALUES (?, ?, ?, ?);";
        $stmt = $conn->prepare($sql);
        $stmt->bind_param("ssss", $nome, $email, $hashed_password, $username);

        if ($stmt->execute()) {
            $usuario_id = $conn->insert_id; // Obtém o ID do usuário inserido
            $_SESSION['usuario_id'] = $usuario_id;
            $_SESSION['username'] = $username;

            // Redireciona para a página inicial com mensagem de sucesso
            header('Location: home.php?cadastro=sucesso');
            exit();
        } else {
            // Exibe mensagem de erro caso a inserção falhe
            echo "Erro ao cadastrar usuário: " . $stmt->error;
        }

        // Fecha o statement
        $stmt->close();
    }

    // Fecha o statement de verificação
    $stmt_verificar->close();
}

// Fecha a conexão com o banco de dados
$conn->close();
?>
```

Fonte: Imagem do autor.

11.1.1.4. Código de logout

O código a seguir é responsável por realizar o processo de *logout* do usuário, ou seja, finalizar a sessão ativa.

A função *logout* inicia recuperando o usuário autenticado a partir da requisição recebida. Isso é feito por meio do método *$request->user()*, que identifica automaticamente o usuário vinculado ao *token* enviado na requisição.

Em seguida, é realizada a revogação do token de acesso atual com o método *$user->currentAccessToken()->delete()*. Essa ação invalida o token que estava em uso, encerrando imediatamente a sessão do usuário e impedindo novos acessos com aquele *token*.

Por fim, é retornada uma resposta em formato JSON com uma mensagem de confirmação informando que o usuário foi desconectado com sucesso.

Esse processo garante que, ao encerrar a sessão, o *token* seja removido do sistema, reforçando a segurança da aplicação e evitando acessos não autorizados após o *logout*.

A figura 16 é apresentado o código de *logout* do usuário.

Figura 16. Trecho do código PHP para *logout* do usuário

```php
/**
 * Logout do usuário.
 */
1 reference | 0 overrides
public function logout(Request $request): JsonResponse|mixed
{
    // Pega o usuário autenticado
    $user = $request->user();

    // Revoca o o token atual
    $user->currentAccessToken()->delete();

    return response()->json(['message' => 'Usuário deslogado com sucesso.']);
}
```

Fonte: Imagem do autor.

11.1.1.5. Código para recuperar todas as receitas

O trecho a seguir é utilizado para recuperar todas as receitas cadastradas no sistema, juntamente com seus relacionamentos relevantes.

A função index realiza uma consulta ao banco de dados na tabela associada ao tipo Receita, utilizando o método *with()* para carregar antecipadamente os relacionamentos com usuario, *tags* e imagens. Esse carregamento evita múltiplas consultas desnecessárias, otimizando a performance da aplicação ao retornar todos os dados de forma consolidada.

Além disso, a função aplica o método *latest()* para ordenar os registros com base na data de criação, garantindo que as receitas mais recentes sejam exibidas primeiro. Por fim, o método *get()* finaliza a consulta e retorna os dados encontrados.

A resposta é então formatada em JSON, contendo a lista completa de receitas e suas informações associadas, como etapas, imagens relacionadas, quantidade de ingredientes etc.

- Utiliza *with()* para carregar relacionamentos (usuário, *tags*, imagens);
- Ordena por data de criação (*latest()*);
- Retorna lista em JSON otimizada.

A figura 17 é apresentado parte do código de recuperação de receitas com base em filtros.

Figura 17. Trecho do código PHP para recuperar todas as receitas de acordo com filtros

```
2 references | 0 implementations | You, 3 seconds ago | 1 author (You)
class ReceitaController extends Controller
{
    0 references | 0 overrides
    public function index(): JsonResponse|mixed
    {
        //Retorna todas as receitas com os relacionamentos necessários
        $receitas = Receita::with(relations: ['usuario', 'tags', 'imagens'])->latest()->get();
        return response()->json(data: $receitas);
    }
```

Fonte: Imagem do autor.

11.1.1.6. Código para recuperar receitas baseado no usuário logado

Essa funcionalidade é responsável por garantir que somente o usuário autenticado possa acessar e editar suas próprias receitas dentro do sistema ChefPrice, mantendo a integridade dos dados e a privacidade das informações. Antes de qualquer operação, o sistema verifica se a receita pertence de fato ao usuário autenticado. Isso é feito por meio da comparação entre o *id_usuario* da receita e o ID do usuário obtido pela autenticação (*Auth::id()*). Caso os IDs não coincidam, o sistema retorna uma resposta de erro 403 - Acesso negado, impedindo qualquer edição ou visualização.

Validação dos Dados da Requisição

O sistema aplica regras de validação nos dados recebidos via requisição *PUT* ou *PATCH*, verificando integridade, formatos esperados e preenchimento obrigatório dos campos.

Atualização dos Dados da Receita

Uma vez validados os dados, o sistema executa uma transação para garantir que todas as alterações sejam aplicadas de forma segura. Dentro dessa transação:

Campos como título, descrição, rendimento e tempo_preparo são atualizados se estiverem presentes na requisição. A atualização ocorre apenas com os campos informados, permitindo modificações parciais sem sobrescrever todo o registro.

Atualização dos Relacionamentos (*tags*, imagens, etapas e ingredientes)

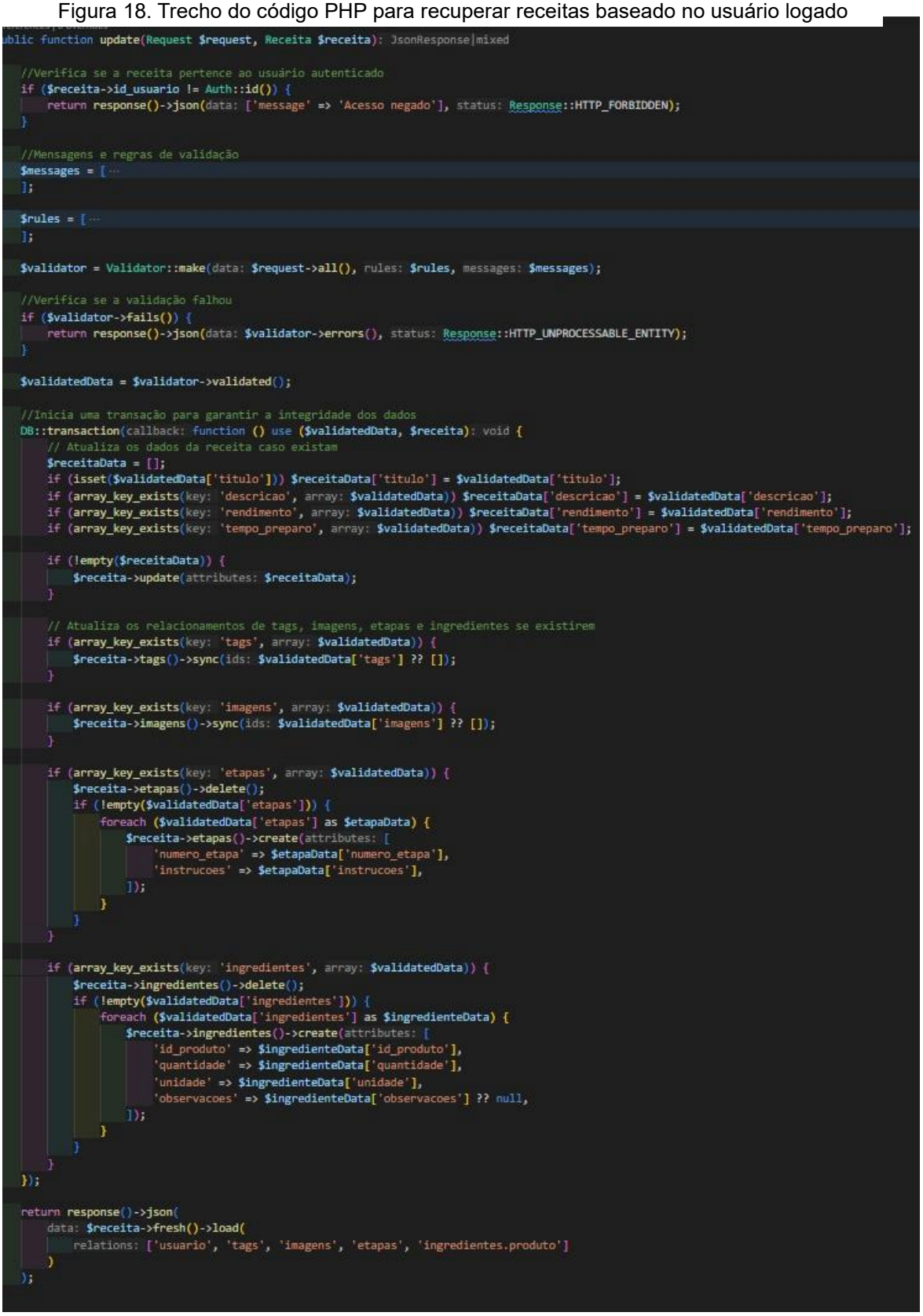
Após atualizar os dados principais da receita, o sistema também gerencia seus relacionamentos. *Tags* e imagens são sincronizadas com os IDs recebidos, substituindo as associações antigas pelas novas. Etapas e ingredientes são totalmente substituídos, os registros existentes são excluídos e recriados com base nos dados atualizados fornecidos pelo usuário.

Por fim, o sistema recarrega todas as relações da receita (usuário, *tags*, imagens, etapas e produtos dos ingredientes) e retorna um JSON com os dados atualizados, assegurando que o Front-end tenha acesso imediato à versão

mais recente da receita modificada. Essa funcionalidade é essencial para manter a segurança e personalização do sistema, garantindo que cada usuário tenha acesso apenas às suas próprias informações e possa gerenciá-las com autonomia, sem risco de interferência externa.

A figura 18 é apresentado o código de recuperação de receitas com base no usuário logado.

Figura 18. Trecho do código PHP para recuperar receitas baseado no usuário logado



```php
public function update(Request $request, Receita $receita): JsonResponse|mixed

    //Verifica se a receita pertence ao usuário autenticado
    if ($receita->id_usuario != Auth::id()) {
        return response()->json(data: ['message' => 'Acesso negado'], status: Response::HTTP_FORBIDDEN);
    }

    //Mensagens e regras de validação
    $messages = [...
    ];

    $rules = [...
    ];

    $validator = Validator::make(data: $request->all(), rules: $rules, messages: $messages);

    //Verifica se a validação falhou
    if ($validator->fails()) {
        return response()->json(data: $validator->errors(), status: Response::HTTP_UNPROCESSABLE_ENTITY);
    }

    $validatedData = $validator->validated();

    //Inicia uma transação para garantir a integridade dos dados
    DB::transaction(callback: function () use ($validatedData, $receita): void {
        // Atualiza os dados da receita caso existam
        $receitaData = [];
        if (isset($validatedData['titulo'])) $receitaData['titulo'] = $validatedData['titulo'];
        if (array_key_exists(key: 'descricao', array: $validatedData)) $receitaData['descricao'] = $validatedData['descricao'];
        if (array_key_exists(key: 'rendimento', array: $validatedData)) $receitaData['rendimento'] = $validatedData['rendimento'];
        if (array_key_exists(key: 'tempo_preparo', array: $validatedData)) $receitaData['tempo_preparo'] = $validatedData['tempo_preparo'];

        if (!empty($receitaData)) {
            $receita->update(attributes: $receitaData);
        }

        // Atualiza os relacionamentos de tags, imagens, etapas e ingredientes se existirem
        if (array_key_exists(key: 'tags', array: $validatedData)) {
            $receita->tags()->sync(ids: $validatedData['tags'] ?? []);
        }

        if (array_key_exists(key: 'imagens', array: $validatedData)) {
            $receita->imagens()->sync(ids: $validatedData['imagens'] ?? []);
        }

        if (array_key_exists(key: 'etapas', array: $validatedData)) {
            $receita->etapas()->delete();
            if (!empty($validatedData['etapas'])) {
                foreach ($validatedData['etapas'] as $etapaData) {
                    $receita->etapas()->create(attributes: [
                        'numero_etapa' => $etapaData['numero_etapa'],
                        'instrucoes' => $etapaData['instrucoes'],
                    ]);
                }
            }
        }

        if (array_key_exists(key: 'ingredientes', array: $validatedData)) {
            $receita->ingredientes()->delete();
            if (!empty($validatedData['ingredientes'])) {
                foreach ($validatedData['ingredientes'] as $ingredienteData) {
                    $receita->ingredientes()->create(attributes: [
                        'id_produto' => $ingredienteData['id_produto'],
                        'quantidade' => $ingredienteData['quantidade'],
                        'unidade' => $ingredienteData['unidade'],
                        'observacoes' => $ingredienteData['observacoes'] ?? null,
                    ]);
                }
            }
        }
    });

    return response()->json(
        data: $receita->fresh()->load(
            relations: ['usuario', 'tags', 'imagens', 'etapas', 'ingredientes.produto']
        )
    );
```

Fonte: Imagem do autor.

11.1.1.7. Código para cadastrar receita

A função *store* irá realizar o cadastro de novas receitas com todos os atributos relacionados no banco de dados apenas mediante validações que irão ocorrer ao longo do código.

Inicialmente na função *store,* são definidas as mensagens dos contextos e restrições para validar os dados recebidos. Ambos estão sendo utilizados na função *Validator*, que definirá o valor da variável *validator*, de acordo com o resultado da validação feita.

No caso de a validação falhar, será retornado um erro no formato de JSON indicando qual dos atributos da requisição está inválido e então o processo será finalizado e a receita não será cadastrada.

Caso haja sucesso na validação, uma nova transação será criada para que a integridade dos dados seja garantida. Nessa transação é feita a definição dos valores dos parâmetros, o *id_usuario* e o *titulo* por exemplo deverão ter um valor atribuído, já a *descricao*, *rendimento* e *tempo_preparo*, podem ser nulos caso nada tenha sido passado para esses campos. Após as validações e atribuições, todos esses parâmetros são passados para a criação de uma nova receita.

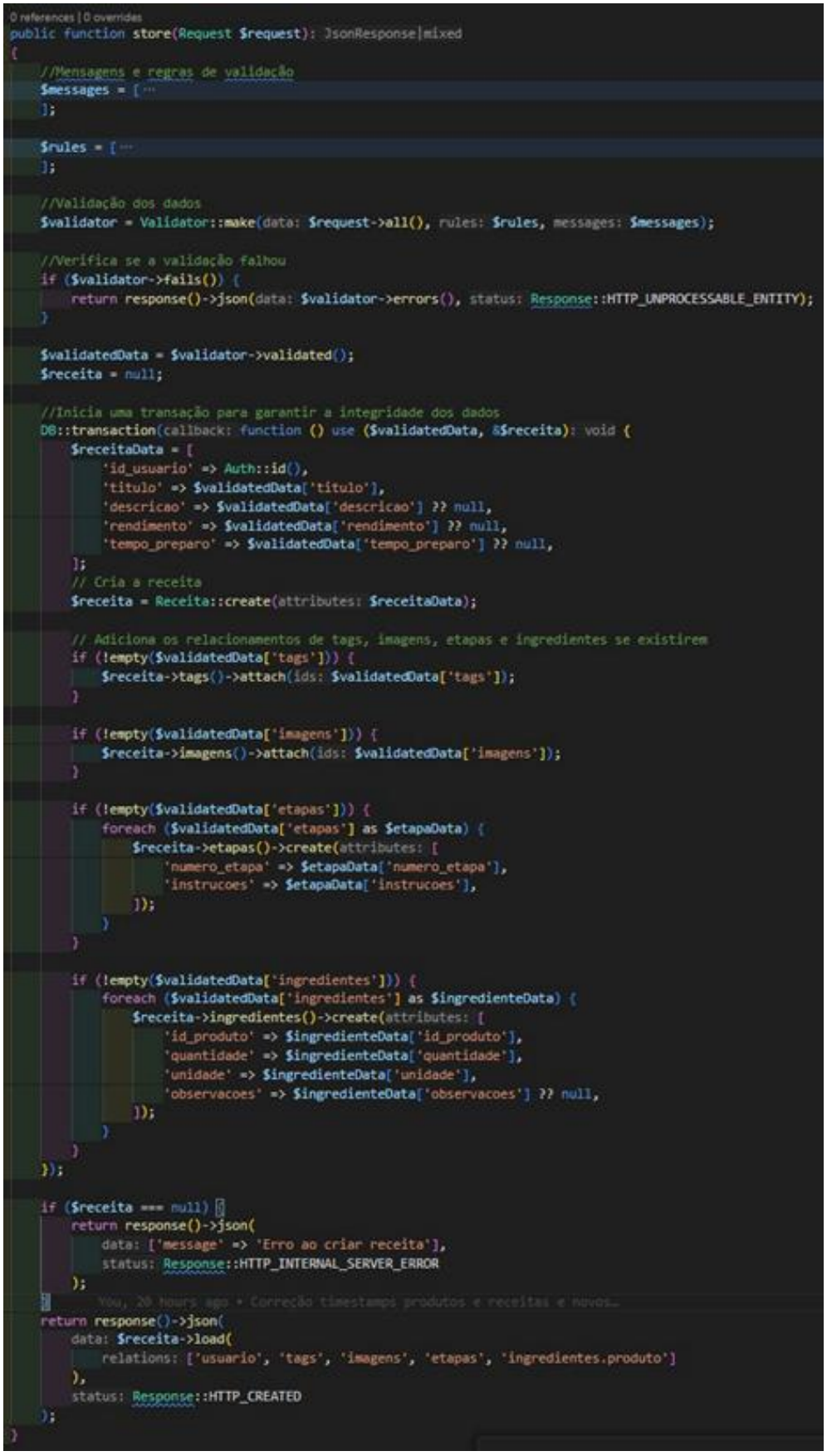
Finalmente, com a nova receita criada, os relacionamentos de *tags*, imagens, etapas da receita, e ingredientes passam por uma verificação de existência, caso essa verificação seja verdadeira, enfim o relacionamento é adicionado.

Ao final do código, ocorre a validação da receita. Se for nula, será retornado um erro e uma mensagem detalhando o ocorrido. Do contrário, os dados atribuídos à receita serão carregados na base de dados e uma mensagem de sucesso será retornada.

- Valida dados com mensagens personalizadas;
- Se falhar, retorna erro detalhado;
- Se bem-sucedido, grava nova receita no banco de dados.

A figura 19 é apresentado o código de cadastro de receita.

Figura 19. Trecho do código PHP para cadastrar receita



```php
0 references | 0 overrides
public function store(Request $request): JsonResponse|mixed
{
    //Mensagens e regras de validação
    $messages = [...
    ];

    $rules = [...
    ];

    //Validação dos dados
    $validator = Validator::make(data: $request->all(), rules: $rules, messages: $messages);

    //Verifica se a validação falhou
    if ($validator->fails()) {
        return response()->json(data: $validator->errors(), status: Response::HTTP_UNPROCESSABLE_ENTITY);
    }

    $validatedData = $validator->validated();
    $receita = null;

    //Inicia uma transação para garantir a integridade dos dados
    DB::transaction(callback: function () use ($validatedData, &$receita): void {
        $receitaData = [
            'id_usuario' => Auth::id(),
            'titulo' => $validatedData['titulo'],
            'descricao' => $validatedData['descricao'] ?? null,
            'rendimento' => $validatedData['rendimento'] ?? null,
            'tempo_preparo' => $validatedData['tempo_preparo'] ?? null,
        ];
        // Cria a receita
        $receita = Receita::create(attributes: $receitaData);

        // Adiciona os relacionamentos de tags, imagens, etapas e ingredientes se existirem
        if (!empty($validatedData['tags'])) {
            $receita->tags()->attach(ids: $validatedData['tags']);
        }

        if (!empty($validatedData['imagens'])) {
            $receita->imagens()->attach(ids: $validatedData['imagens']);
        }

        if (!empty($validatedData['etapas'])) {
            foreach ($validatedData['etapas'] as $etapaData) {
                $receita->etapas()->create(attributes: [
                    'numero_etapa' => $etapaData['numero_etapa'],
                    'instrucoes' => $etapaData['instrucoes'],
                ]);
            }
        }

        if (!empty($validatedData['ingredientes'])) {
            foreach ($validatedData['ingredientes'] as $ingredienteData) {
                $receita->ingredientes()->create(attributes: [
                    'id_produto' => $ingredienteData['id_produto'],
                    'quantidade' => $ingredienteData['quantidade'],
                    'unidade' => $ingredienteData['unidade'],
                    'observacoes' => $ingredienteData['observacoes'] ?? null,
                ]);
            }
        }
    });

    if ($receita === null) {
        return response()->json(
            data: ['message' => 'Erro ao criar receita'],
            status: Response::HTTP_INTERNAL_SERVER_ERROR
        );
    }       You, 20 hours ago • Correção timestamp produtos e receitas e novos...
    return response()->json(
        data: $receita->load(
            relations: ['usuario', 'tags', 'imagens', 'etapas', 'ingredientes.produto']
        ),
        status: Response::HTTP_CREATED
    );
}
```

Fonte: Imagem do autor.

11.1.1.8. Sugerir preço para uma receita

Essa funcionalidade está alocada no Front-end, e consiste em um componente Angular, responsável por apresentar um **modal com os cálculos automáticos de custos e lucros** de uma receita cadastrada no sistema. Toda a lógica é processada no Front-end, permitindo ao usuário visualizar de forma prática e em tempo real.

A lógica de precificação do sistema ChefPrice foi pensada para facilitar a tomada de decisão dos empreendedores na hora de definir o valor de venda de seus produtos. O sistema começa calculando o custo total de produção de uma receita, somando os valores de todos os ingredientes utilizados com base na quantidade e no preço padrão de cada um.

Em seguida, são considerados os chamados **custos adicionais**, que representam despesas indiretas muitas vezes ignoradas, como gás de cozinha, energia elétrica, embalagens e outros insumos que não estão diretamente listados nos ingredientes. Esses custos são aplicados como um percentual sobre o valor dos materiais, tornando o cálculo mais fiel à realidade.

Depois, com base no rendimento da receita, ou seja, em quantas unidades aquele preparo gera, o sistema calcula o **custo por unidade**. Esse valor serve de base para aplicar a **margem de lucro desejada**, configurada pelo próprio usuário.

Com isso, o ChefPrice fornece automaticamente um **valor sugerido de venda** que garante que o produto não seja vendido abaixo do custo, além de calcular o **lucro total estimado** para aquela produção, oferecendo uma visão clara e estratégica da rentabilidade. Essa lógica simples, mas completa, torna o processo de precificação mais justo, confiável e acessível mesmo para quem não tem formação na área financeira.

A figura 20 é apresentado o código de sugestão de preço de venda.

Figura 20. Código TS do modal de sugestão de preço de venda

```typescript
export class ModalCalculoLucroComponent {
  @Output() fecharModal = new EventEmitter<void>();
  mostrarModal: boolean = false;
  public receita?: Receita;

  constructor(private cdr: ChangeDetectorRef) {}

  public abrirModal(receita: Receita) {
    this.receita = receita;
    this.mostrarModal = true;
    this.cdr.markForCheck();
  }

  public fechar() {
    this.mostrarModal = false;
    this.fecharModal.emit();
    this.cdr.markForCheck();
  }

  public get calcularCustoDosMateriais(): number {
    return (
      this.receita?.ingredientes.reduce((total, ingrediente) => {
        return total + Number(ingrediente.quantidade) * (ingrediente.produto?.preco_padrao ?? 0);
      }, 0) ?? 0
    );
  }

  /**
   * Calcula o valor dos custos adicionais da receita.
   * O valor é obtido aplicando o percentual de custos adicionais (caso definido na receita)
   * sobre o custo total dos materiais. Retorna 0 se não houver percentual definido.
   * @returns {number} Valor dos custos adicionais calculados.
   */
  public get calcularCustoAdicionais(): number {
    const percentual = this.receita?.custosAdicionais ?? 0;
    const custoMateriais = this.calcularCustoDosMateriais;
    return (custoMateriais * percentual) / 100;
  }

  /**
   * Calcula o custo por unidade da receita.
   * O custo total é a soma do custo dos materiais e dos custos adicionais.
   * O rendimento é obtido da receita, e se não estiver definido, assume 1.
   * Retorna 0 se o rendimento for menor ou igual a 0.
   * @returns {number} Custo por unidade calculado.
   */
  public get calcularCustoPorUnidade(): number {
    const custoTotal = this.calcularCustoDosMateriais + this.calcularCustoAdicionais;
    const rendimento = this.receita?.rendimento ? parseFloat(this.receita.rendimento) : 1;
    return rendimento > 0 ? custoTotal / rendimento : 0;
  }

  /**
   * Calcula o custo total da receita.
   * O custo total é a soma do custo dos materiais e dos custos adicionais.
   * @returns {number} Custo total da receita calculado.
   */
  public get calcularCustoTotalReceita(): number {
    return this.calcularCustoDosMateriais + this.calcularCustoAdicionais;
  }

  /**
   * Calcula o valor sugerido de venda da receita.
   * O valor é obtido somando o custo por unidade com o lucro esperado (se definido).
   * Se o lucro esperado for 0, retorna apenas o custo por unidade.
   * @returns {number} Valor sugerido de venda calculado.
   */
  public get calcularValorSugeridoVenda(): number {
    const custoPorUnidade = this.calcularCustoPorUnidade;
    const lucroEsperado = this.receita?.lucroEsperado ? this.receita.lucroEsperado : 0;

    // Se o lucro esperado for 0, retorna o custo por unidade
    if (lucroEsperado === 0) {
      return custoPorUnidade;
    }

    // Calcula o valor sugerido de venda com base no custo por unidade e no lucro esperado
    return custoPorUnidade + (custoPorUnidade * lucroEsperado) / 100;
  }

  /**
   * Calcula o lucro total da receita.
   * O lucro é obtido subtraindo o custo total da receita do valor sugerido de venda.
   * @returns {number} Lucro total calculado.
   */
  public get calcularLucroTotal(): number {
    const rendimento = this.receita?.rendimento ? parseFloat(this.receita.rendimento) : 1;

    return (this.calcularValorSugeridoVenda * rendimento) - this.calcularCustoTotalReceita;
  }

}
```

Fonte: Imagem do autor.

11.1.1.9. Recuperar materiais associados ao usuário logado

Aqui foi encontrado um desafio de como faríamos que um produto que é cadastrado pelo administrador do sistema fosse sempre renderizado para o usuário, sem que ele tivesse acesso ao banco de materiais integral vindas do administrador, pois ao subir uma nota fiscal para o banco de dados e a mesma fizesse a atualização do preço do material esse valor não poderia ficar registrado globalmente no sistema de modo que aparecesse para um outro usuário que utilizasse esse mesmo produto em sua conta.

Para resolução desse problema, foi criado um objeto de associação, que carrega para determinado material as características cadastradas pelo usuário e usa como referência o id do usuário e do produto.

A figura 21 é apresentado o código de recuperação de materiais.

Figura 21. Código PHP para recuperação de materiais

```php
public function index(Request $request): JsonResponse|mixed
{
    $query = ProdutoHistorico::query();
    if ($request->has(key: 'id_produto')) {
        $query->where(column: 'id_produto', operator: $request->input(key: 'id_produto'));
    }
    if ($request->has(key: 'id_usuario')) {
        $query->where(column: 'id_usuario', operator: $request->input(key: 'id_usuario'));
    }
    $historicos = $query->with(relations: ['produto', 'usuario'])->latest()->get();
    return response()->json(data: $historicos);
}
```

Fonte: Imagem do autor.

11.1.1.10. Cadastrar lista de compras

O código apresentado refere-se ao método responsável por **armazenar uma nova lista de compras** no sistema ChefPrice. Esse método é ativado por uma requisição HTTP do tipo *POST* e contempla desde a validação de dados até o relacionamento da lista com produtos selecionados pelo usuário.

Inicialmente, o sistema **recupera o usuário autenticado**, responsável pela criação da lista, e tenta localizar no banco de dados um **status padrão "Ativa"** na tabela *lista_compra_status*. Esse status será utilizado como padrão caso o usuário não envie um status específico na requisição.

Em seguida, são definidas regras e mensagens de validação (ainda que estejam vazias no trecho apresentado), e os dados da requisição são verificados com base nessas regras. Caso haja falhas, o sistema retorna uma resposta JSON com os erros encontrados, evitando o processamento de dados inválidos.

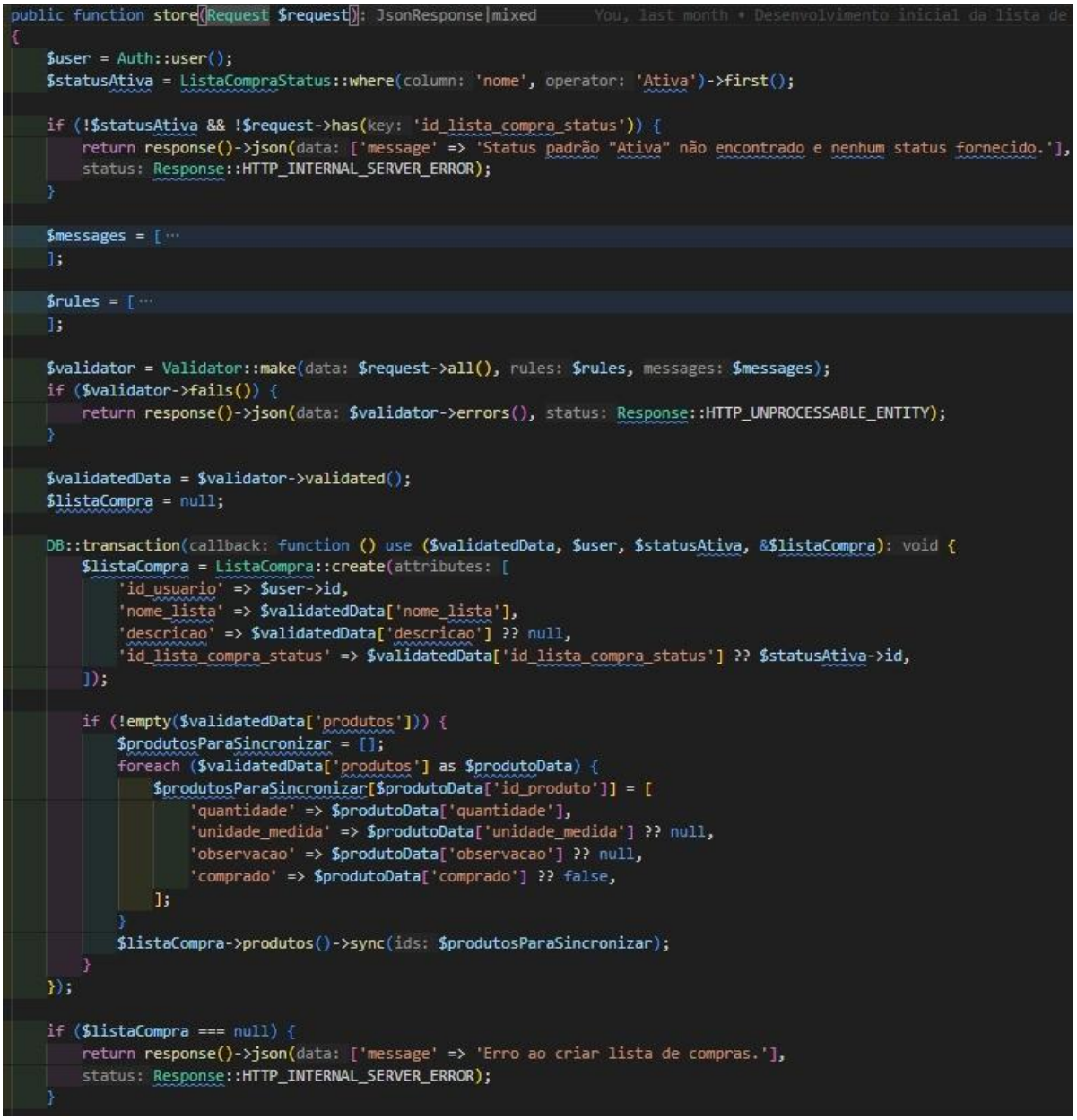
Se a validação for bem-sucedida, os dados são processados dentro de uma **transação de banco de dados**, garantindo que tanto a criação da lista quanto a associação com os produtos só ocorrerão se todas as etapas forem concluídas com sucesso.

Durante essa transação, a lista de compras é criada com os dados recebidos, vinculada ao usuário e ao status apropriado. Se houver produtos incluídos na requisição, o sistema percorre a lista de produtos e **monta um *array* contendo informações adicionais** como quantidade, unidade de medida, observações e se o item já foi comprado. Esses dados são então sincronizados com a lista criada, utilizando o método *sync()*, que realiza o relacionamento entre a lista e os produtos por meio de uma tabela intermediária (*pivot*), armazenando inclusive os atributos extras mencionados.

Ao final do processo, caso a lista seja criada com sucesso, o método finaliza normalmente. Se houver qualquer falha, o sistema retorna um erro HTTP 500 informando que não foi possível criar a lista de compras.

A figura 22 é apresentado trecho do código de cadastro de lista de compras.

Figura 22. Trecho do código PHP para cadastrar lista de compras



```php
public function store(Request $request): JsonResponse|mixed      You, last month • Desenvolvimento inicial da lista de
{
    $user = Auth::user();
    $statusAtiva = ListaCompraStatus::where(column: 'nome', operator: 'Ativa')->first();

    if (!$statusAtiva && !$request->has(key: 'id_lista_compra_status')) {
        return response()->json(data: ['message' => 'Status padrão "Ativa" não encontrado e nenhum status fornecido.'],
        status: Response::HTTP_INTERNAL_SERVER_ERROR);
    }

    $messages = [ ···
    ];

    $rules = [ ···
    ];

    $validator = Validator::make(data: $request->all(), rules: $rules, messages: $messages);
    if ($validator->fails()) {
        return response()->json(data: $validator->errors(), status: Response::HTTP_UNPROCESSABLE_ENTITY);
    }

    $validatedData = $validator->validated();
    $listaCompra = null;

    DB::transaction(callback: function () use ($validatedData, $user, $statusAtiva, &$listaCompra): void {
        $listaCompra = ListaCompra::create(attributes: [
            'id_usuario' => $user->id,
            'nome_lista' => $validatedData['nome_lista'],
            'descricao' => $validatedData['descricao'] ?? null,
            'id_lista_compra_status' => $validatedData['id_lista_compra_status'] ?? $statusAtiva->id,
        ]);

        if (!empty($validatedData['produtos'])) {
            $produtosParaSincronizar = [];
            foreach ($validatedData['produtos'] as $produtoData) {
                $produtosParaSincronizar[$produtoData['id_produto']] = [
                    'quantidade' => $produtoData['quantidade'],
                    'unidade_medida' => $produtoData['unidade_medida'] ?? null,
                    'observacao' => $produtoData['observacao'] ?? null,
                    'comprado' => $produtoData['comprado'] ?? false,
                ];
            }
            $listaCompra->produtos()->sync(ids: $produtosParaSincronizar);
        }
    });

    if ($listaCompra === null) {
        return response()->json(data: ['message' => 'Erro ao criar lista de compras.'],
        status: Response::HTTP_INTERNAL_SERVER_ERROR);
    }
```

Fonte: Imagem do autor.

12. CRITÉRIOS DE INOVAÇÃO

O projeto **ChefPrice** apresenta uma proposta inovadora ao unir **Inteligência artificial**, **empatia com o público-alvo** e a **vivência prática de seus desenvolvedores** para solucionar um problema recorrente enfrentado por pequenos empreendedores: o controle financeiro e a correta precificação de seus produtos.

Um dos principais diferenciais do ChefPrice é o fato de ter sido idealizado e desenvolvido por jovens que **vivenciaram na prática** os desafios que o sistema busca resolver. Essa experiência pessoal proporcionou uma **visão realista e sensível das necessidades do usuário final**, permitindo o desenvolvimento de uma ferramenta verdadeiramente alinhada à realidade desses empreendedores.

O ChefPrice se destaca pela **proximidade com o público-alvo e pela atenção às suas dores reais**. Cada funcionalidade foi cuidadosamente planejada para ser **intuitiva, acessível e eficiente**, eliminando as barreiras que normalmente afastam os usuários da gestão financeira.

Além disso, o sistema consegue **equilibrar simplicidade** integrando tecnologias avançadas como a **inteligência artificial aplicada à leitura automatizada de notas fiscais**. Com uma interface amigável, que pode ser facilmente utilizada até mesmo por pessoas com pouca familiaridade com ferramentas digitais.

Portanto, o ChefPrice se diferencia por **aliar tecnologia, simplicidade e experiência prática**, oferecendo uma solução completa e acessível que entrega valor de forma direta, eficaz e adaptada à realidade dos pequenos empreendedores alimentícios.

12.1. Prompt estruturado para leitura automatizada de cupons fiscais

Uma das inovações mais relevantes do ChefPrice é a utilização da inteligência artificial para extrair automaticamente informações de cupons fiscais a partir de uma imagem. Essa tarefa é realizada por meio de uma requisição à API Gemini, utilizando um prompt altamente detalhado e customizado.

O prompt foi elaborado para simular a atuação de um assistente especialista em leitura de cupons fiscais brasileiros, instruindo a IA a reconhecer padrões típicos do documento, extrair dados estruturados e formatá-los corretamente em JSON.

Essa abordagem foi construída com o objetivo de garantir a máxima precisão na leitura de campos críticos, como:

- Descrição dos produtos;
- Quantidade adquirida;

- Unidade de medida;
- Preço unitário e total;
- Descontos aplicados por item;
- Código de barras;
- Data e hora da venda.

Além da leitura textual, o prompt ainda orienta o modelo a aplicar validações internas, como a comparação entre o número do item e a linha do desconto, a repetição de OCR's com múltiplos filtros e a aplicação de um "sistema de votação" para capturar os valores mais confiáveis, uma técnica que aumenta significativamente a qualidade da leitura automatizada.

A figura 23 é apresentado o prompt utilizado.

Figura 23. Instruções do prompt personalizado para leitura automatizada de cupons fiscais

```
$prompt = <<<PROMPT
    ATENÇÃO: Você é um assistente especializado em analisar cupons fiscais
    brasileiros e extrair informações estruturadas de vendas.
    Você receberá uma imagem de um cupom fiscal brasileiro e deve extrair as
    informações de venda em formato JSON estruturado.
    Você deve analisar o texto da imagem e criar o JSON com as informações
    solicitadas.
    GARANTA QUE TODOS OS ITENS SEJAM EXTRAÍDOS CORRETAMENTE E QUE O JSON ESTEJA
    FORMATADO DE ACORDO COM AS INSTRUÇÕES ABAIXO.
    é CRUCIAL que você siga as instruções rigorosamente e retorne apenas o JSON
    solicitado, sem texto adicional ou explicações.

    LEIA ITEM A ITEM PARA EVITAR MISTURAR INFORMAÇÕES E GARANTIR A EXATIDÃO DOS
    DADOS EXTRAÍDOS.
    Para isso Determine de é necessário ler 2 ou 3 linhas de cada item para
    extrair todas as informações necessárias.
    2 linhas quando não houver desconto e 3 linhas quando houver desconto.

    Utilize multiplas passagens de OCR com filtros de imagem distinto para
    garantir a melhor qualidade de leitura.
    Utilize um sistema de votação para determinar a melhor leitura de cada item.
    Faça isso especialmente para os campos de preço unitário, quantidade e
    desconto, que podem variar entre as linhas.
    Atenção redobrada em quantidade quando for um número decimal, especialmente
    no valor inteiro e nas primeiras casas decimais.

    Para garantir que você está aplicando o desconto no item correto, compare o
    número do item da linha 1 e o número do item na linha 3 (linha de desconto)

    Analise a seguinte imagem de "cupom fiscal" brasileiro.
    Extraia as seguintes informações em formato JSON:
    1. A data principal da venda (geralmente indicada como "VENDA" ou próxima ao
    cabeçalho). Formato: "DD/MM/YYYY - HH:MM".
    2. Uma lista de todos os itens. Para cada item, extraia:
        - "item_code": O número sequencial do item (ex: "001", "002").
        - "barcode": O código de barras do produto ou código numérico se
        disponível (pode ser nulo).
        - "name": A descrição do produto.
        - "quantity": A quantidade comprada (numérico, use ponto como separador
        decimal).
        - "unit_of_measure": A unidade de medida (ex: "UN", "KG", "PC").
        - "unit_price": O preço por unidade (numérico, use ponto como separador
        decimal).
        - "total_price": O preço total para essa linha de item antes de qualquer
        desconto específico do item (numérico, use ponto como separador decimal).
        - "discount": O valor do desconto aplicado especificamente a este item.
        Se uma linha "Desconto sobre item XXX" aparecer abaixo do item, extraia
        este valor como um número positivo. Se não houver desconto, use 0.

    Garanta que todos os valores numéricos (quantity, unit_price, total_price,
    discount) sejam retornados como números.
    Se um campo não puder ser determinado com segurança, retorne null para
    strings ou 0 para números onde apropriado (exceto barcode que pode ser null).
    Priorize a data e hora da operação de VENDA.
    Retorne apenas o JSON, nada além do JSON.
    Não retorne texto adicional, apenas o JSON formatado corretamente com todos
    os itens da lista.
    Sua resposta será processada por um sistema automatizado, então não inclua
    explicações ou formatação adicional.
    Respostas incompletas ou com erros de formatação podem causar falhas no
    processamento.
    Garanta que todos os itens estão no JSON, não utilize "// ... (rest of the
    items)" ou semelhante.

    Muita atenção aos detalhes, especialmente com os campos numéricos. Os
    valores entre parênteses são impostos e devem ser desconsiderados.
```

Fonte: Imagem do autor.

A figura 24 apresenta um trecho da função *processarCupom*, localizada na classe *ProcessamentoImagemService*, responsável pelo processamento automatizado de cupons fiscais por meio da leitura de imagem com auxílio de inteligência artificial (IA).

Esse método é ativado sempre que um usuário realiza o upload de um cupom fiscal. Após o envio, a imagem é encaminhada ao método *ExtrairDadosGemini*, o qual comunica-se com a API Gemini, que por sua vez realiza o reconhecimento de texto (OCR) e retorna uma estrutura JSON com as informações extraídas do cupom.

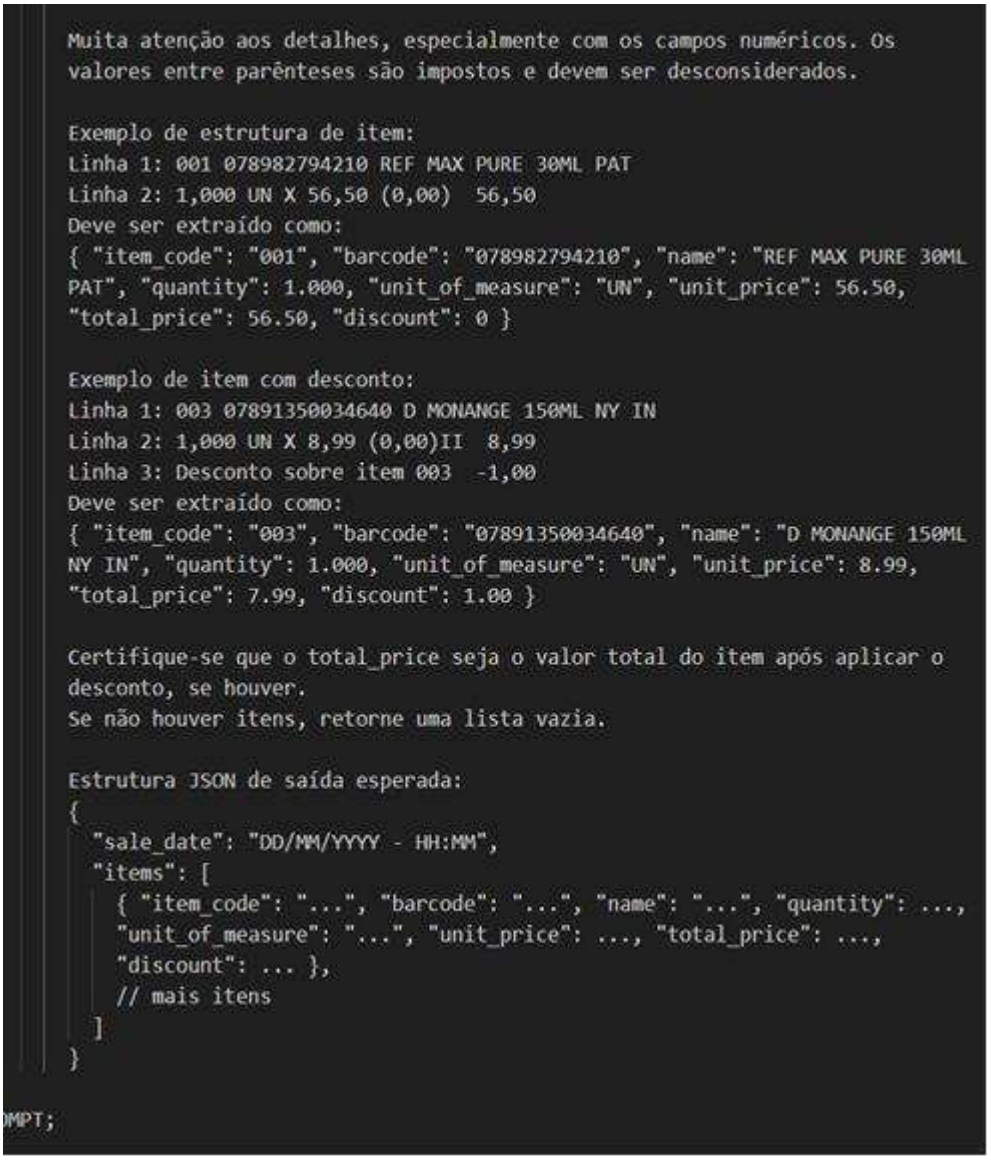
Esse JSON inclui dados como:

- Data e hora da venda;
- Lista de itens comprados, contendo:
- Nome do produto;
- Quantidade;
- Unidade de medida;
- Código de barras (quando disponível);
- Preço unitário e total;
- Valor de desconto (caso aplicável).

Após a extração, o sistema executa diversas validações:

- Validação da resposta da IA
- Verifica se o retorno contém os campos obrigatórios items e *sale_date*. Em caso de ausência ou falha na estrutura, o sistema retorna erro e registra no log.

Figura 24. Prompt utilizado para o treinamento de modelo de IA



```
Muita atenção aos detalhes, especialmente com os campos numéricos. Os
valores entre parênteses são impostos e devem ser desconsiderados.

Exemplo de estrutura de item:
Linha 1: 001 078982794210 REF MAX PURE 30ML PAT
Linha 2: 1,000 UN X 56,50 (0,00)  56,50
Deve ser extraído como:
{ "item_code": "001", "barcode": "078982794210", "name": "REF MAX PURE 30ML
PAT", "quantity": 1.000, "unit_of_measure": "UN", "unit_price": 56.50,
"total_price": 56.50, "discount": 0 }

Exemplo de item com desconto:
Linha 1: 003 07891350034640 D MONANGE 150ML NY IN
Linha 2: 1,000 UN X 8,99 (0,00)II  8,99
Linha 3: Desconto sobre item 003  -1,00
Deve ser extraído como:
{ "item_code": "003", "barcode": "07891350034640", "name": "D MONANGE 150ML
NY IN", "quantity": 1.000, "unit_of_measure": "UN", "unit_price": 8.99,
"total_price": 7.99, "discount": 1.00 }

Certifique-se que o total_price seja o valor total do item após aplicar o
desconto, se houver.
Se não houver itens, retorne uma lista vazia.

Estrutura JSON de saída esperada:
{
  "sale_date": "DD/MM/YYYY - HH:MM",
  "items": [
    { "item_code": "...", "barcode": "...", "name": "...", "quantity": ...,
    "unit_of_measure": "...", "unit_price": ..., "total_price": ...,
    "discount": ... },
    // mais itens
  ]
}

OMPT;
```

Fonte: Imagem do autor.

12.2. Conversão de data

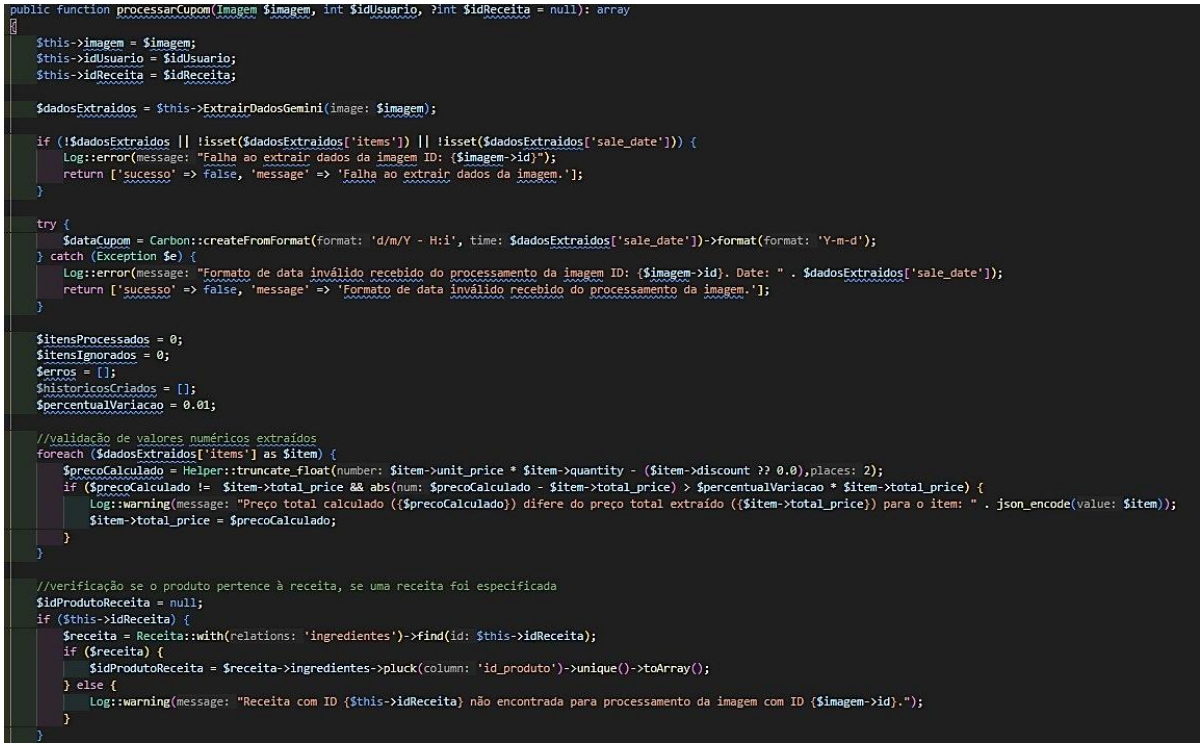
A data retornada é convertida do formato brasileiro (dd/mm/yyyy - hh:mm) para o padrão de armazenamento do banco de dados (yyyy-mm-dd), utilizando a biblioteca *Carbon*.

Quando o cupom está sendo processado dentro do contexto de uma receita específica (parâmetro *$idReceita*), o sistema valida se os produtos extraídos realmente pertencem à composição dessa receita, garantindo integridade nos registros.

Essa abordagem técnica permite ao sistema ChefPrice extrair informações complexas a partir de uma imagem, interpretar dados de forma autônoma, corrigir possíveis distorções matemáticas e, ao final do processo, estruturar essas informações para posterior salvamento no banco de dados.

A figura 25 é apresentado o código de validação e verificação de dados extraídos pela IA do cupom fiscal.

Figura 25. Trecho da função de validação e verificação de consistência dos dados extraídos do cupom fiscal



```php
public function processarCupom(Imagem $imagem, int $idUsuario, ?int $idReceita = null): array
{
    $this->imagem = $imagem;
    $this->idUsuario = $idUsuario;
    $this->idReceita = $idReceita;

    $dadosExtraidos = $this->ExtrairDadosGemini(image: $imagem);

    if (!$dadosExtraidos || !isset($dadosExtraidos['items']) || !isset($dadosExtraidos['sale_date'])) {
        Log::error(message: "Falha ao extrair dados da imagem ID: {$imagem->id}");
        return ['sucesso' => false, 'message' => 'Falha ao extrair dados da imagem.'];
    }

    try {
        $dataCupom = Carbon::createFromFormat(format: 'd/m/Y - H:i', time: $dadosExtraidos['sale_date'])->format(format: 'Y-m-d');
    } catch (Exception $e) {
        Log::error(message: "Formato de data inválido recebido do processamento da imagem ID: {$imagem->id}. Date: " . $dadosExtraidos['sale_date']);
        return ['sucesso' => false, 'message' => 'Formato de data inválido recebido do processamento da imagem.'];
    }

    $itensProcessados = 0;
    $itensIgnorados = 0;
    $erros = [];
    $historicosCriados = [];
    $percentualVariacao = 0.01;

    //validação de valores numéricos extraídos
    foreach ($dadosExtraidos['items'] as $item) {
        $precoCalculado = Helper::truncate_float(number: $item->unit_price * $item->quantity - ($item->discount ?? 0.0),places: 2);
        if ($precoCalculado !=  $item->total_price && abs(num: $precoCalculado - $item->total_price) > $percentualVariacao * $item->total_price) {
            Log::warning(message: "Preço total calculado ({$precoCalculado}) difere do preço total extraído ({$item->total_price}) para o item: " . json_encode(value: $item));
            $item->total_price = $precoCalculado;
        }
    }

    //verificação se o produto pertence à receita, se uma receita foi especificada
    $idProdutoReceita = null;
    if ($this->idReceita) {
        $receita = Receita::with(relations: 'ingredientes')->find(id: $this->idReceita);
        if ($receita) {
            $idProdutoReceita = $receita->ingredientes->pluck(column: 'id_produto')->unique()->toArray();
        } else {
            Log::warning(message: "Receita com ID {$this->idReceita} não encontrada para processamento da imagem com ID {$imagem->id}.");
        }
    }
```

Fonte: Imagem do autor.

Essa camada de automação reduz drasticamente o esforço manual do usuário, promove maior confiabilidade nos dados armazenados e representa um dos principais diferenciais técnicos do projeto.

12.3. Armazenamento automatizado de dados com validação transacional

A figura 26 ilustra a etapa final do método *processarCupom*, responsável por **armazenar no banco de dados os itens extraídos automaticamente do cupom fiscal**, após validação feita com inteligência artificial. Esse processo é

realizado de forma transacional, garantindo que qualquer falha anule as inserções parciais.

Figura 26. Processamento e persistência de itens extraídos do cupom fiscal

```php
DB::beginTransaction();
try {
    foreach ($dadosExtraidos['items'] as $item) {
        $produto = $this->indexOrStoreProduto(item: $item);      You, 12 minutes ago • Refatoração do serviço de processamento de imag…
        if (!isset($item->barcode) || empty($item->barcode)) {
            Log::warning(message: "Item sem código de barras encontrado: " . json_encode(value: $item));
            $erros[] = "Item sem código de barras encontrado: " . json_encode(value: $item);
            $itensIgnorados++;
            continue;
        }

        if (!$produto) {
            Log::warning(message: "Produto não encontrado ou criado para o item: " . ($item->name ?? $item->barcode) . " na imagem ID ({$imagem->id})");
            $erros[] = "Produto não encontrado/criado para: " . ($item->name ?? $item->barcode);
            $itensIgnorados++;
            continue;
        }

        if ($idProdutoReceita !== null && !in_array(needle: $produto->id, haystack: $idProdutoReceita)) {
            Log::info(message: "Produto {$produto->nome} (ID: {$produto->id}) do cupom (Imagem ID: {$imagem->id}) não pertence à receita ID {$this->idReceita}. Pulando.");
            $itensIgnorados++;
            continue;
        }

        $quantidade = (float)str_replace(search: ',', replace: '.', subject: $item->quantity);
        $precoUnitario = (float)str_replace(search: ',', replace: '.', subject: $item->unit_price);
        $desconto = isset($item->discount) ? (float)str_replace(search: ',', replace: '.', subject: $item->discount) : 0.0;
        $precoTotalCalculado = ($quantidade * $precoUnitario) - $desconto;

        $historico = ProdutoHistorico::create(attributes: [
            'id_usuario' => $this->idUsuario,
            'id_produto' => $produto->id,
            'preco_unitario' => $precoUnitario,
            'quantidade' => $quantidade,
            'preco_total' => $precoTotalCalculado,
            'data_compra' => $dataCupom,
            'desconto' => $desconto,
        ]);
        $historicosCriados[] = $historico->id;
        $itensProcessados++;
    }
    DB::commit();
} catch (Exception $e) {
    DB::rollBack();
    Log::error(message: "Erro ao salvar históricos de produto para imagem ID {$imagem->id}: " . $e->getMessage());
    return ['sucesso' => false, 'message' => 'Erro ao salvar dados no banco.', 'errors' => [$e->getMessage()]];
}

Log::info(message: "Processamento do cupom concluído para imagem ID {$imagem->id}. Itens processados: {$itensProcessados}, Itens ignorados: {$itensIgnorados}.");
return [
    'sucesso' => true,
    'mensagem' => "Processamento do cupom concluído. {$itensProcessados} itens processados, {$itensIgnorados} itens ignorados.",
    'itensProcessados' => $itensProcessados,
    'itensIgnorados' => $itensIgnorados,
    'errosProcessamento' => $erros
];
```

Fonte: Imagem do autor.

Nesse trecho, o sistema percorre a lista de itens extraídos pela IA (já validados anteriormente), realiza o tratamento e normalização dos dados, e efetua o registro no banco por meio da criação de objetos na tabela *produto_historico*.

A lógica é envolta por uma transação (*DB:beginTransaction()*), garantindo a **consistência da operação**: caso ocorra qualquer exceção durante o processamento, todas as alterações são desfeitas com um *DB::rollBack()*.

Cada item passa pelas seguintes etapas:

1. **Busca ou criação do produto no banco de dados**, com base no código de barras. Se o item não possuir código ou o produto não for localizado/criado, ele é ignorado e um aviso é registrado no log.

2. **Verificação de vínculo com uma receita específica** (caso o processamento esteja atrelado a uma receita previamente cadastrada). Se o produto extraído não fizer parte da receita em questão, também é ignorado.
3. **Conversão dos valores** de quantidade, preço unitário e desconto, com substituição do separador decimal (vírgula para ponto), garantindo compatibilidade com o formato esperado pelo banco.
4. **Cálculo do valor total** do item, considerando desconto:
   total = (quantidade × preço unitário) − desconto
5. **Registro da operação no banco de dados** com a criação do histórico da compra (*ProdutoHistorico::create*), armazenando os seguintes campos:
   - ID do usuário;
   - ID do produto;
   - Preço unitário;
   - Quantidade adquirida;
   - Preço total calculado;
   - Data da compra;
   - Valor do desconto aplicado.

Durante o processo, o sistema contabiliza quantos itens foram **processados com sucesso**, quantos foram **ignorados por inconsistência** e **registra mensagens detalhadas no log** do sistema, tanto em casos de erro como de sucesso.

Ao final do processamento, o sistema retorna uma resposta estruturada em JSON, contendo:

- Status da operação;
- Número de itens processados e ignorados;
- Mensagem descritiva de sucesso ou falha;
- Lista de erros registrados (se houver).

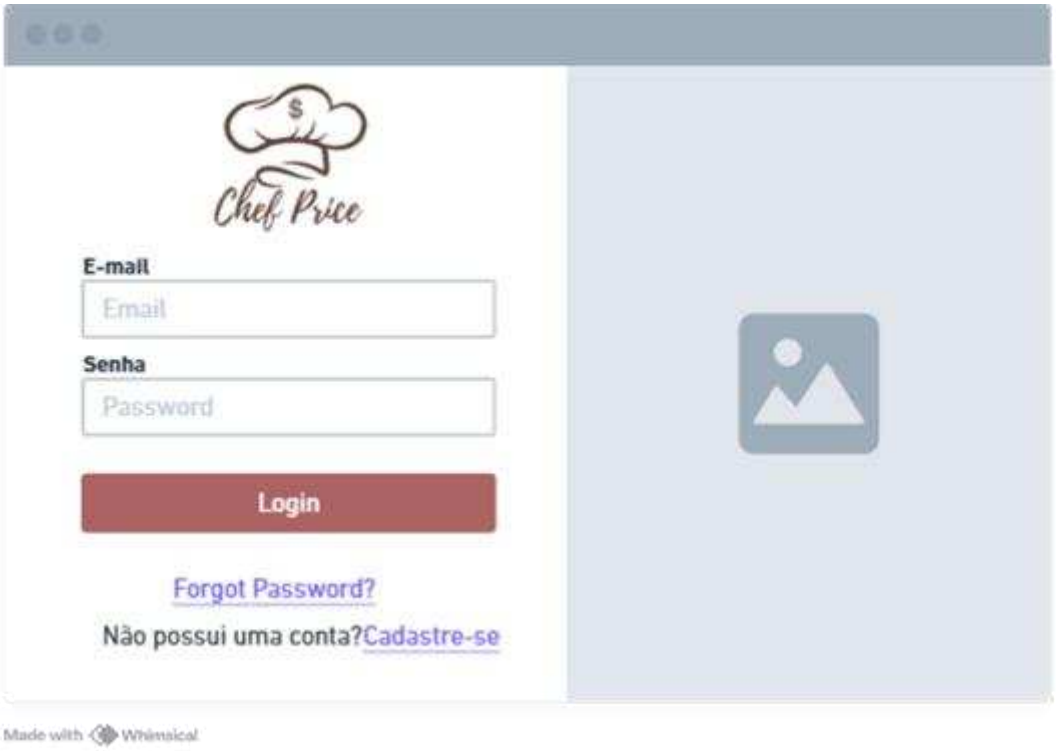
Esse processo demonstra o cuidado da arquitetura do sistema ChefPrice com a **segurança dos dados, integridade das informações e rastreabilidade das ações**, fortalecendo sua proposta de inovação baseada em automação inteligente, confiável e acessível.

13. PROTÓTIPO DAS TELAS

A prototipação das telas é uma etapa crucial para elucidar a melhor maneira de posicionar os componentes gráficos. Um dos principais critérios levados em consideração foi *affordance*, termo utilizado na área de experiência do usuário (UX), que significa reduzir a carga de esforço mental do próprio usuário, melhorando a experiência e usabilidade. Para isso as telas devem ser intuitivas e possuir um *design* leve, principalmente se tratando do posicionamento e características dos componentes, como cor, tamanho etc.

Tela de login

Figura 27. Protótipo da tela de login



Fonte: Imagem do autor.

Tela de cadastro de usuário

Figura 28. Protótipo de tela de cadastro de usuário

Fonte: Imagem do autor.

Tela de recuperação de senha

Figura 29. Protótipo de tela de recuperação de senha

Fonte: Imagem do autor.

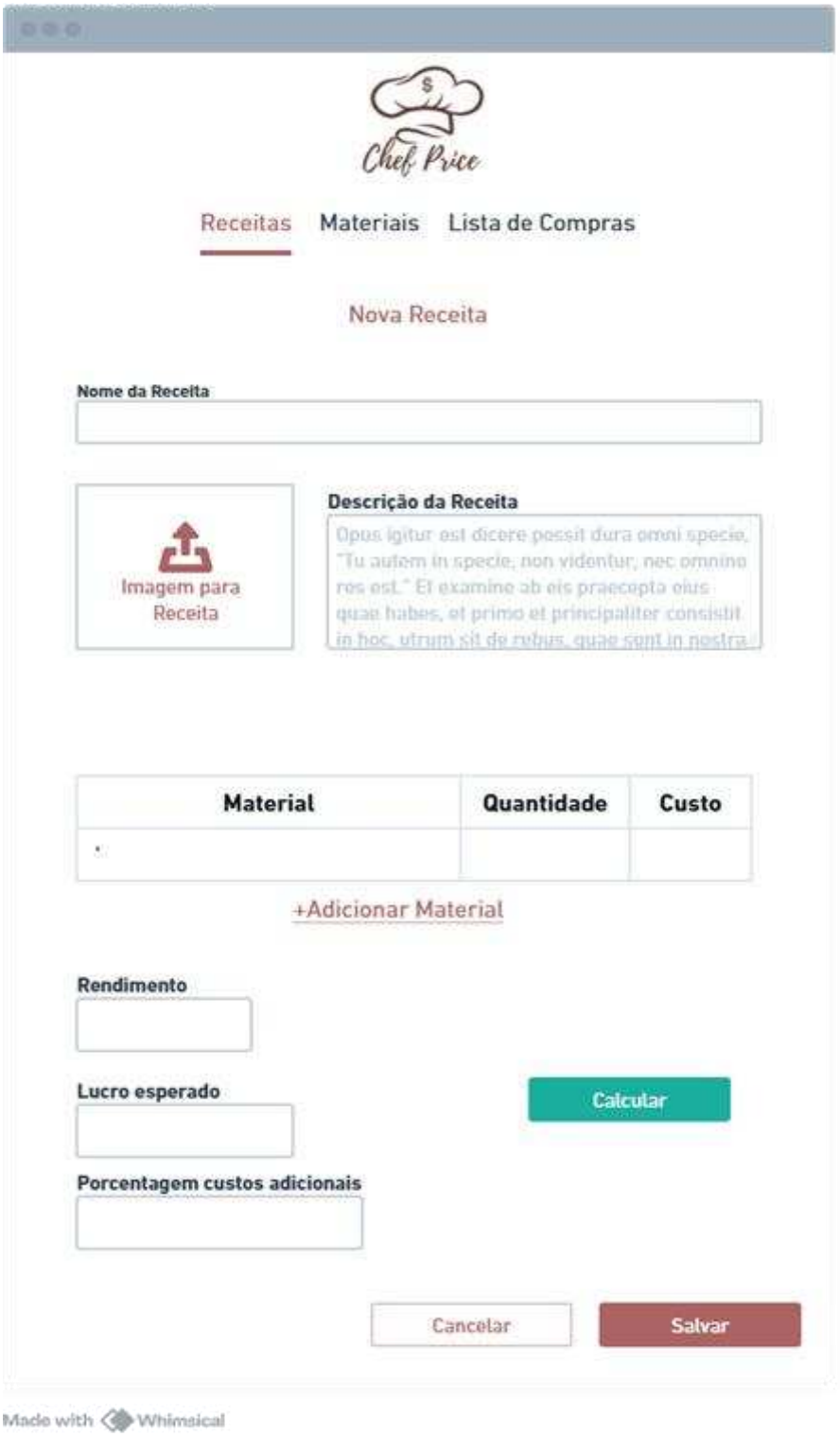
Tela de Inicial do sistema

Figura 30. Protótipo da tela inicial do sistema

Fonte: Imagem do autor.

Tela de cadastro de receita

Figura 31. Protótipo da tela de cadastro de receita



Fonte: Imagem do autor.

Modal de cálculo de valor da receita

Figura 32. Protótipo de modal de cálculo de valor da receita

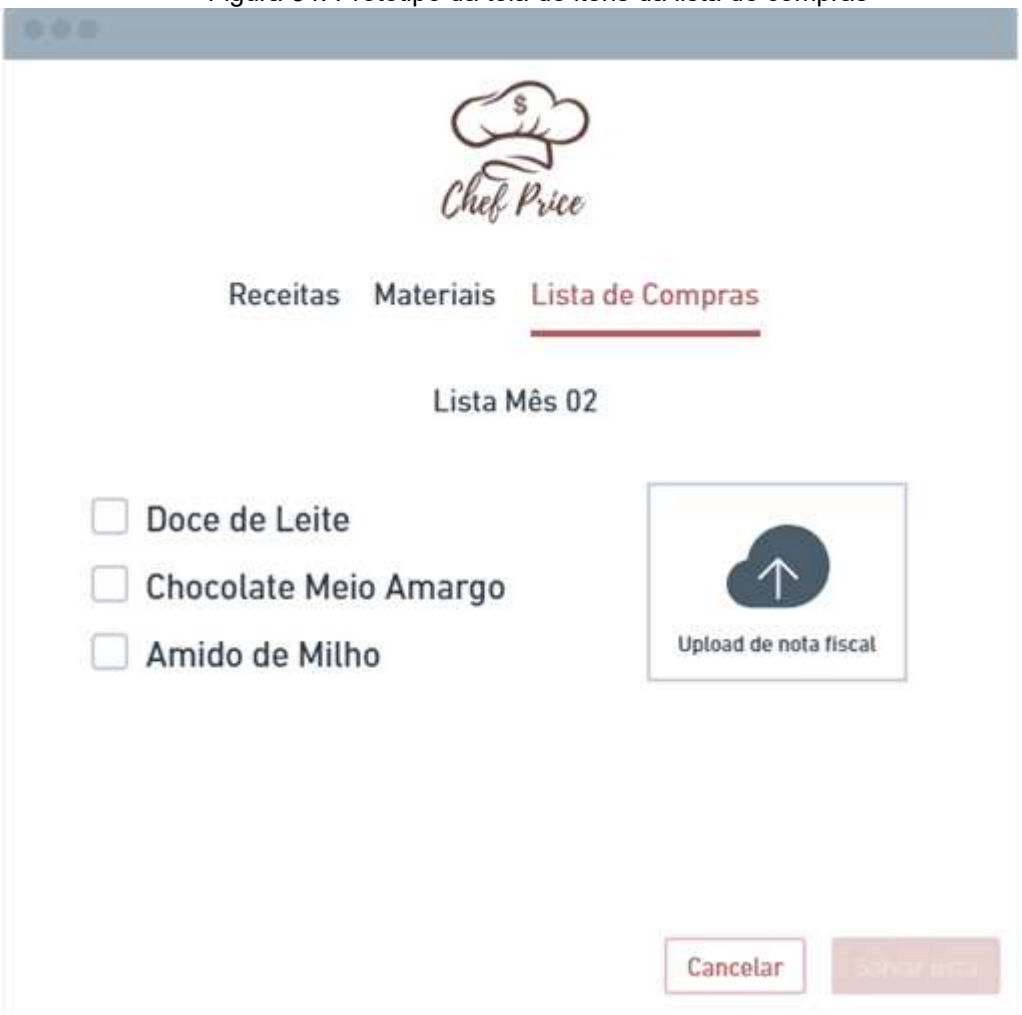
Fonte: Imagem do autor.

Tela de lista compras

Figura 33. Protótipo de listas de compras cadastradas

Fonte: Imagem do autor.

Figura 34. Protótipo da tela de itens da lista de compras



Fonte: Imagem do autor.

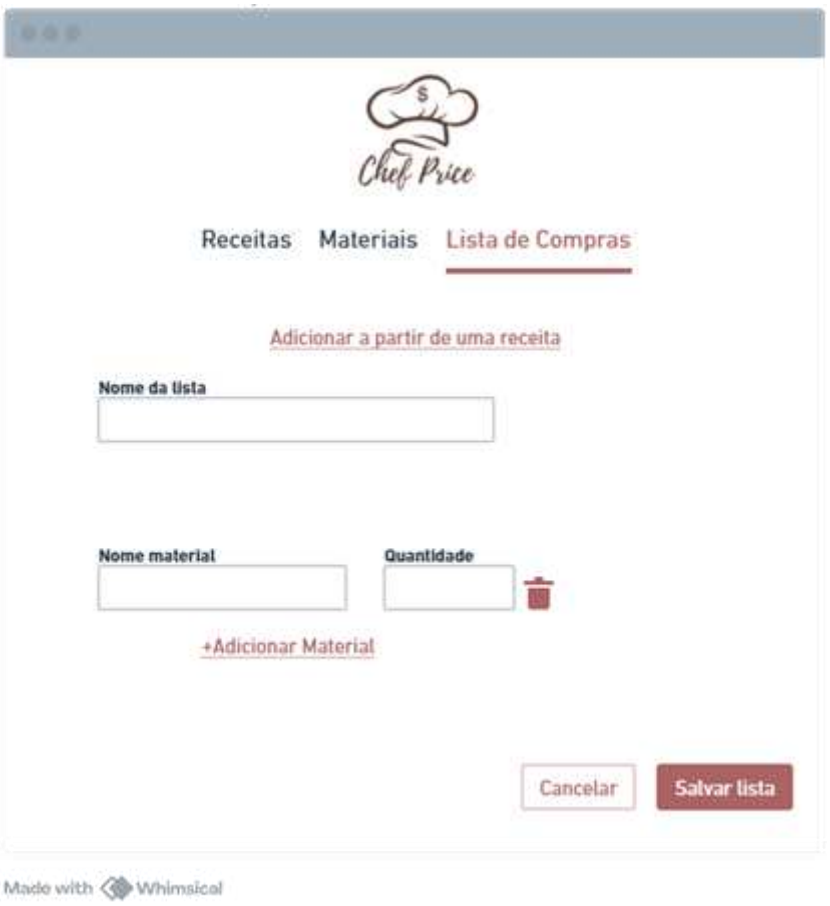
Tela de materiais de usuário administrador

Figura 35. Protótipo da tela de lista de materiais

Fonte: Imagem do autor.

Tela de cadastro de lista de compras

Figura 36. Protótipo da tela de cadastro de lista de compras

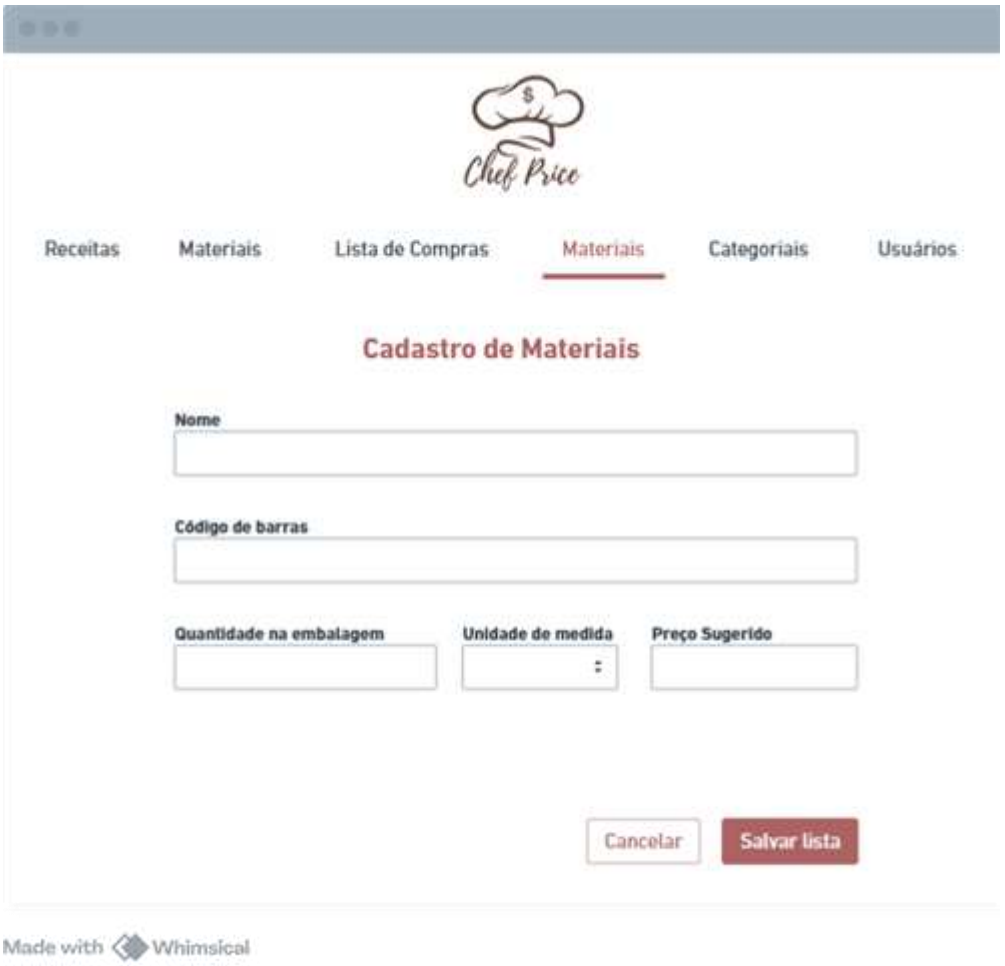


Fonte: Imagem do autor.

Tela de cadastro de materiais

O acesso a essa tela de cadastro de materiais tem o acesso limitado para usuários administradores.

Figura 37. Protótipo da tela de cadastro de materiais

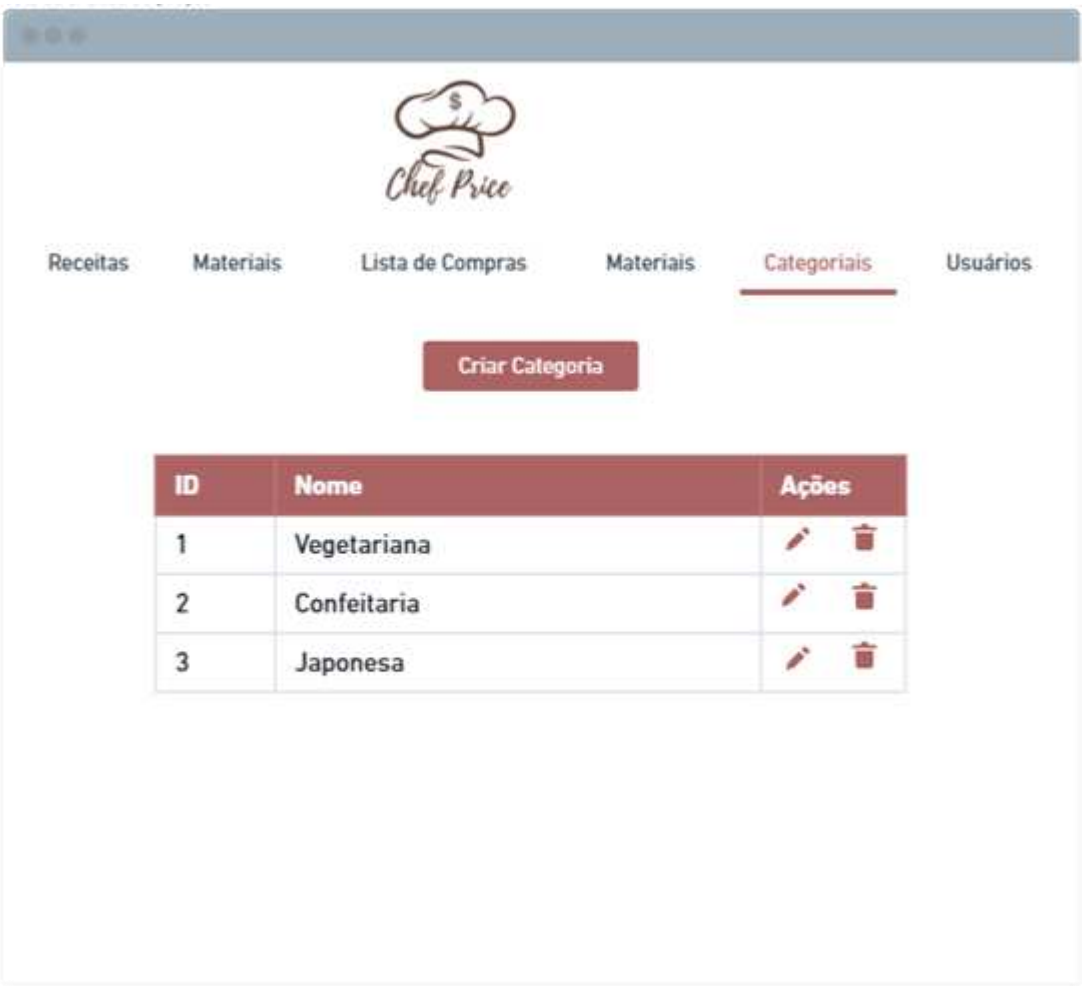


Fonte: Imagem do autor.

Tela de categorias

O acesso a tela de categorias tem o acesso limitado para usuários administradores.

Figura 38. Protótipo da tela de categorias



Fonte: Imagem do autor.

Tela de gerenciamento de usuários

O acesso a tela de gerenciamento de usuários tem o acesso limitado para usuários administradores.

Figura 39. Protótipo da tela de gerenciamento de usuários

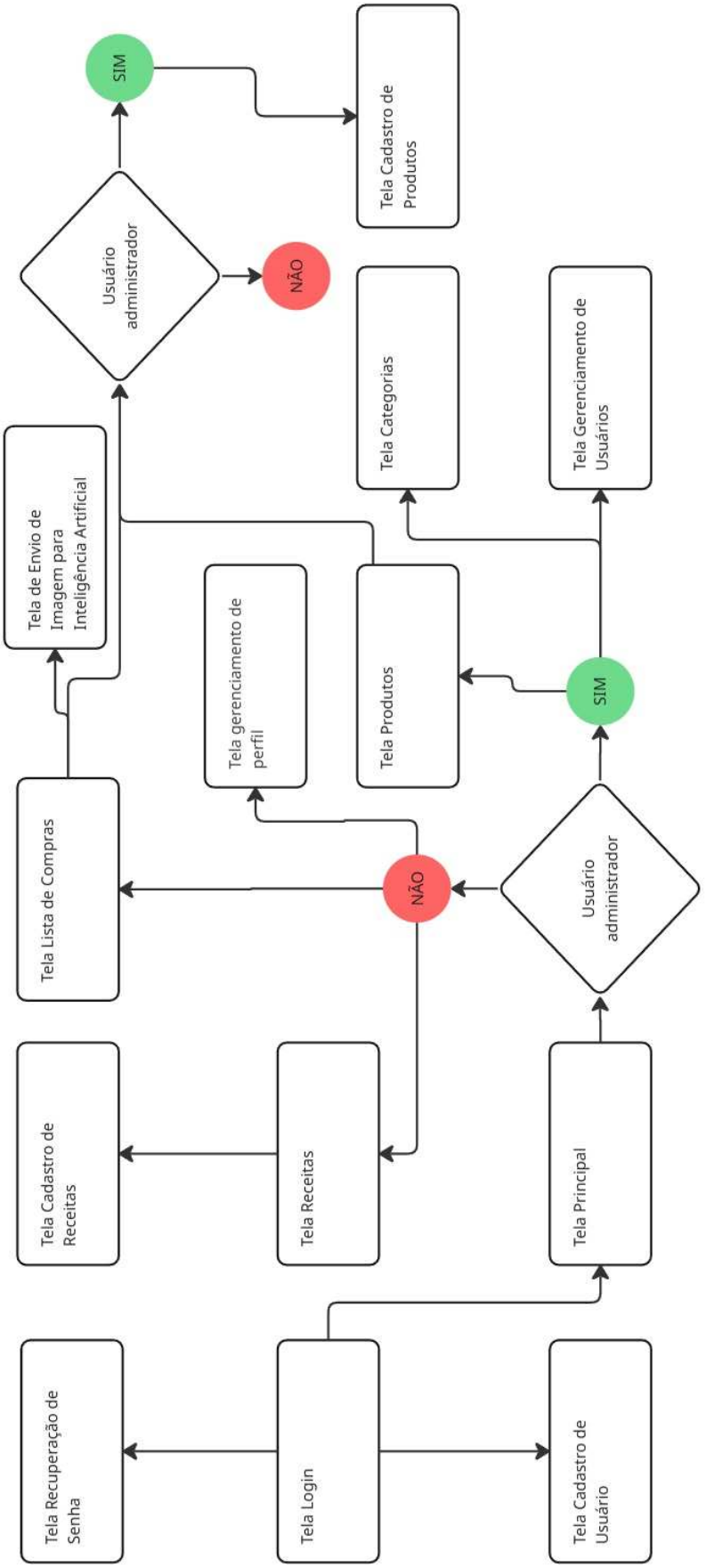
Fonte: Imagem do autor.

14. FLUXO DE TELAS

O diagrama de fluxo de telas simplifica o entendimento da aplicação, e permite que o caminho para chegar a um determinado ponto seja visualizado, demonstrando inclusive restrições para telas que só devem ser acessadas por usuários permitidos.

A figura 40 é apresentado sobre o fluxo de usuário do sistema.

Figura 40. Fluxograma de navegação de tela



Fonte: Imagem do autor.

15. AVALIAÇÃO DO PROJETO FINAL

O projeto ChefPrice foi avaliado com base em critérios essenciais como funcionalidade, usabilidade, desempenho e segurança, apresentando resultados positivos em todos os aspectos analisados. A aplicação demonstrou cumprir com êxito os objetivos definidos em sua concepção, oferecendo uma ferramenta prática e eficaz para auxiliar pequenos empreendedores do setor alimentício na gestão de custos, precificação e organização de insumos.

Durante os testes, o sistema se destacou pela sua interface intuitiva e responsiva, desenvolvida com foco na experiência do usuário. O design simples e moderno, aliado à navegação fluida, garantiu facilidade de uso mesmo para usuários com pouca familiaridade com sistemas digitais.

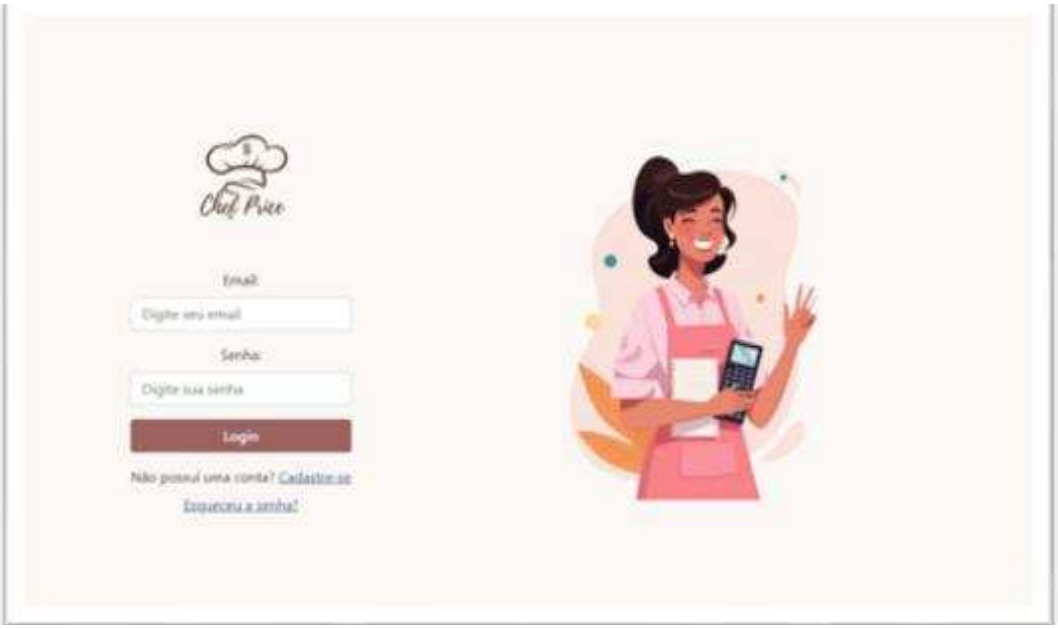
As principais funcionalidades, como o cadastro e cálculo automático de receitas, a gestão de materiais com histórico de preços, a lista de compras integrada e o dashboard de análise de preços, operaram corretamente, proporcionando uma experiência estável e confiável. A integração com inteligência artificial para leitura de notas fiscais também se mostrou funcional, agilizando o processo de atualização de valores de insumos.

Portanto, o ChefPrice foi considerado um projeto bem-sucedido, alcançando plenamente seus objetivos. Além de sua relevância técnica, a aplicação possui forte potencial para impactar positivamente a realidade de seus usuários, promovendo uma gestão financeira mais eficiente e contribuindo para o crescimento sustentável de pequenos negócios alimentícios.

16. TELAS FINAIS DO SISTEMA

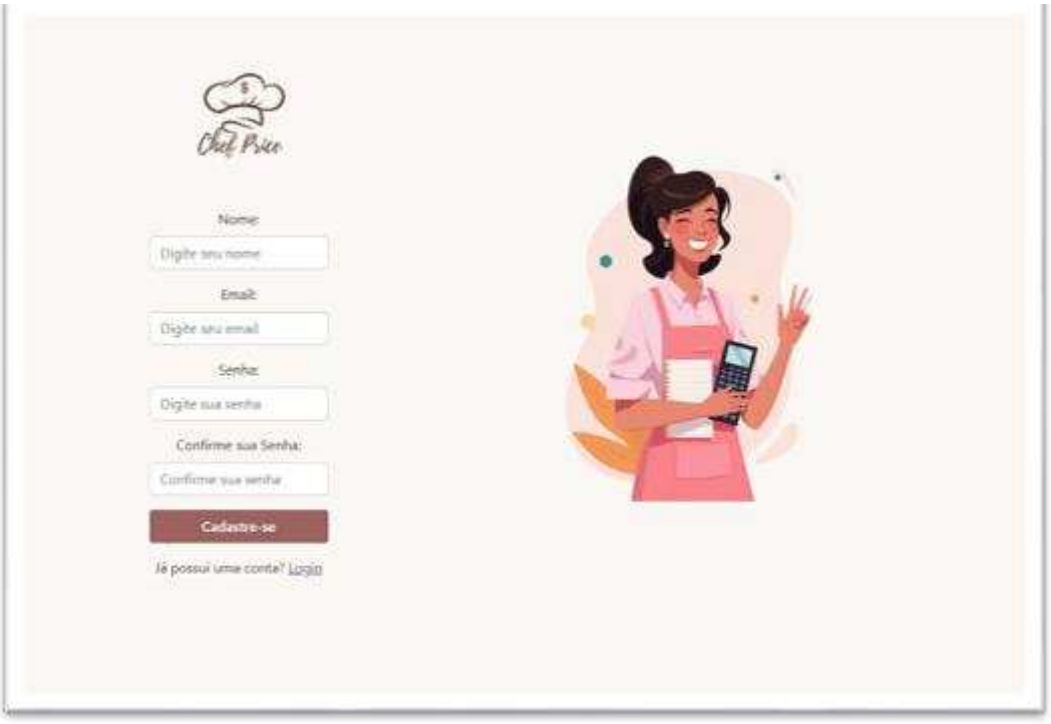
As telas finalizadas e inspiradas nos Wireframes criados concretizam toda a idealização desse projeto.

Figura 41. Tela de login



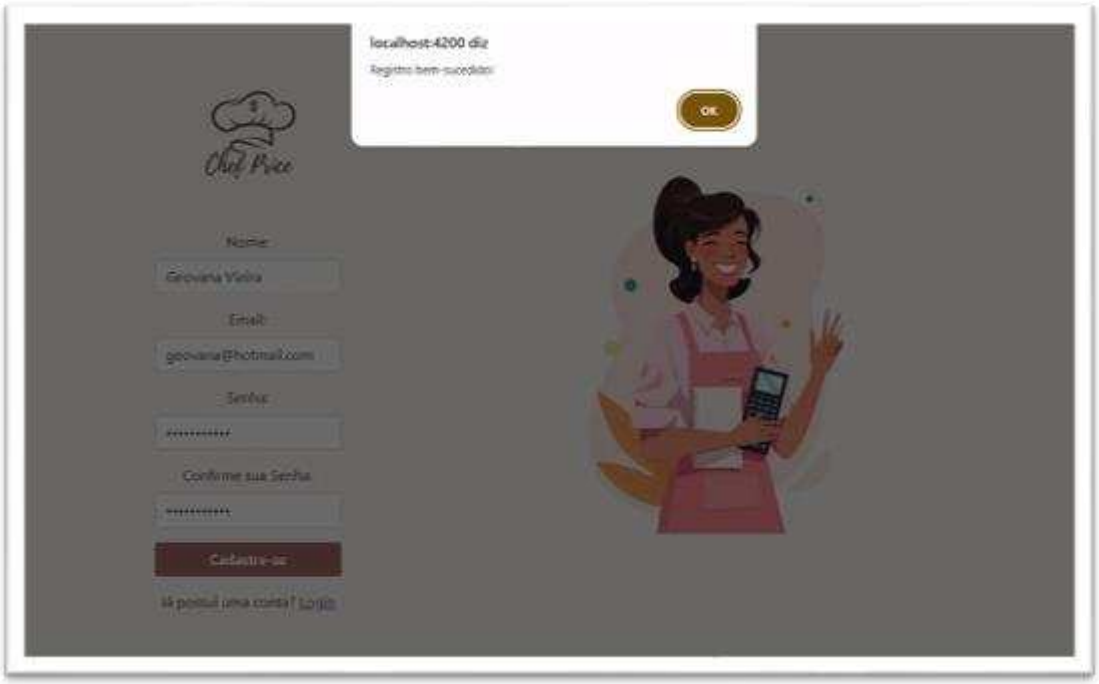
Fonte: Imagem do autor.

Figura 42. Tela de cadastro



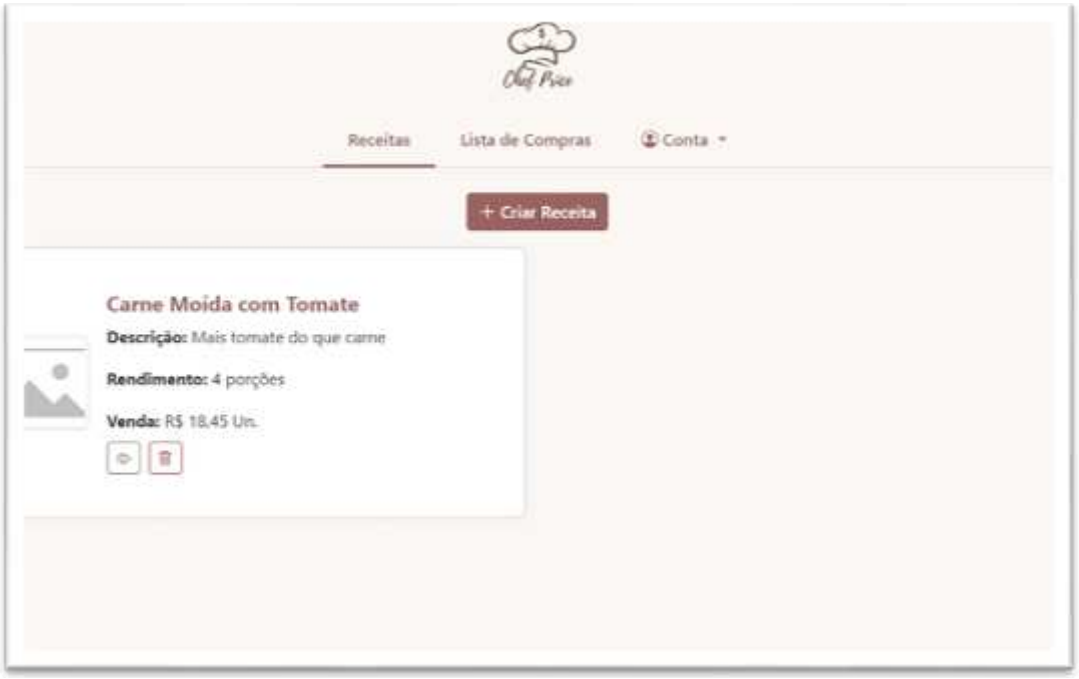
Fonte: Imagem do autor.

Figura 43. Mensagem de sucesso de cadastro de usuário



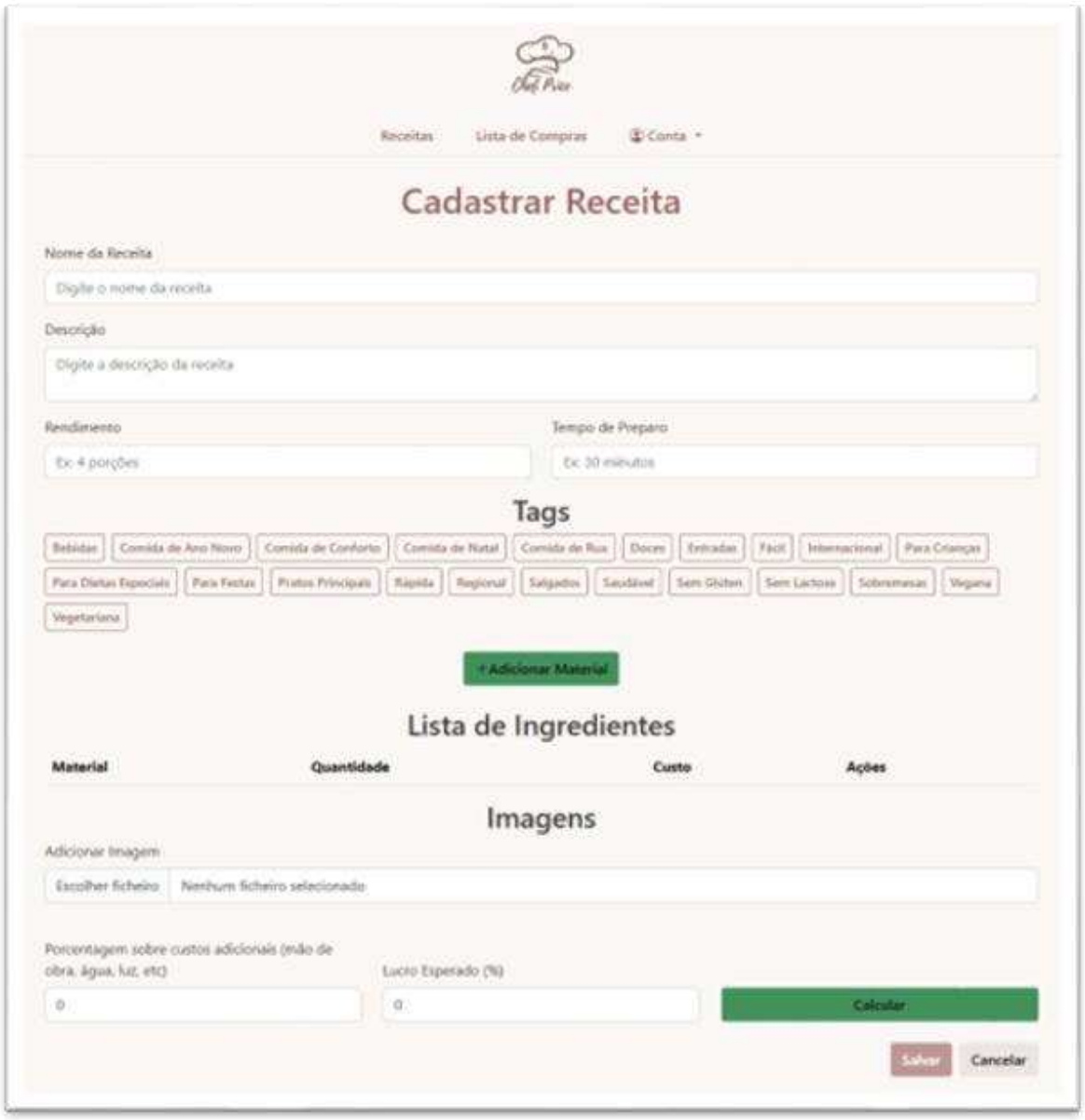
Fonte: Imagem do autor.

Figura 44. Tela inicial



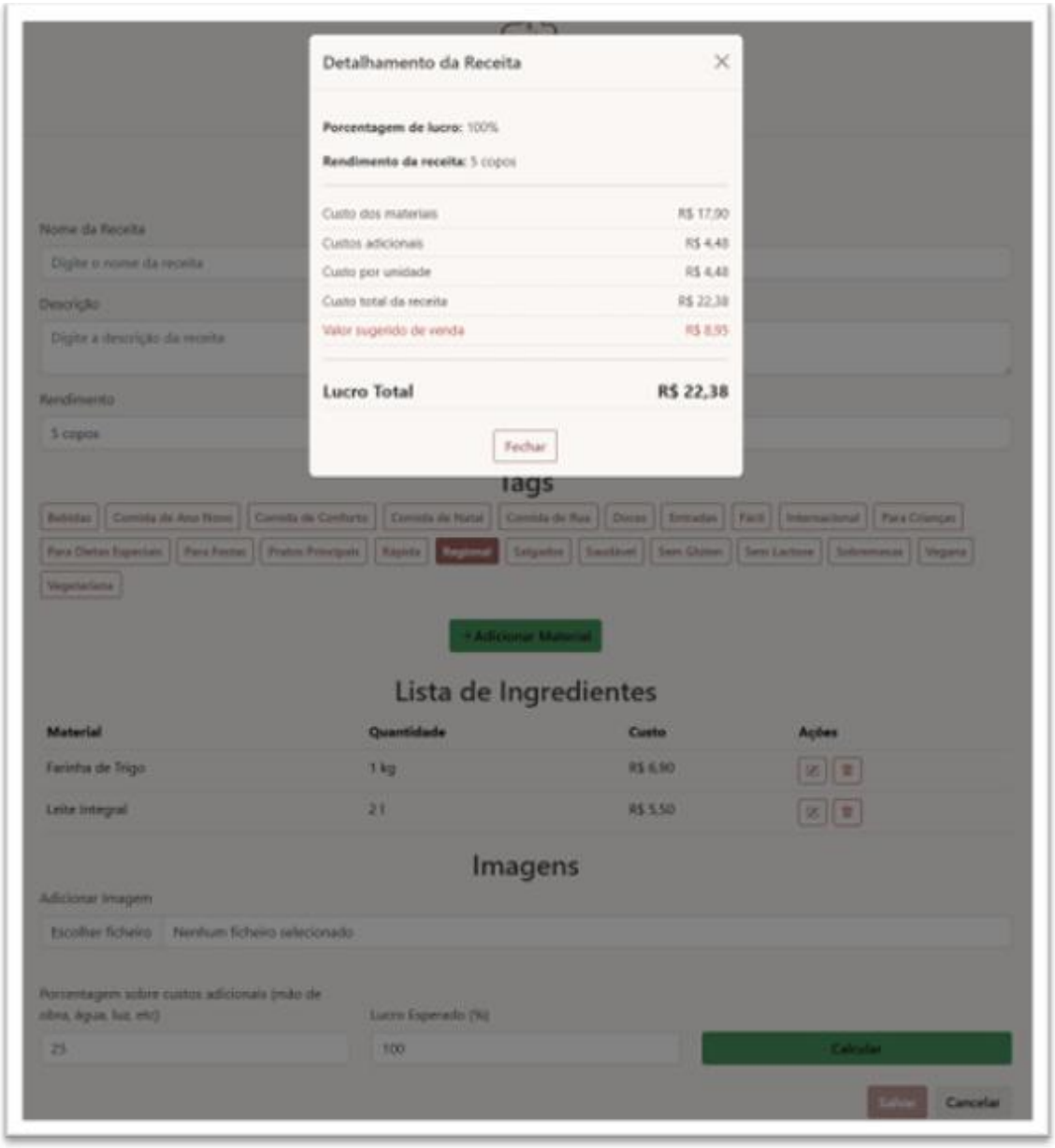
Fonte: Imagem do autor.

Figura 45. Tela de cadastro de receita



Fonte: Imagem do autor.

Figura 46. Modal de cálculo de receita

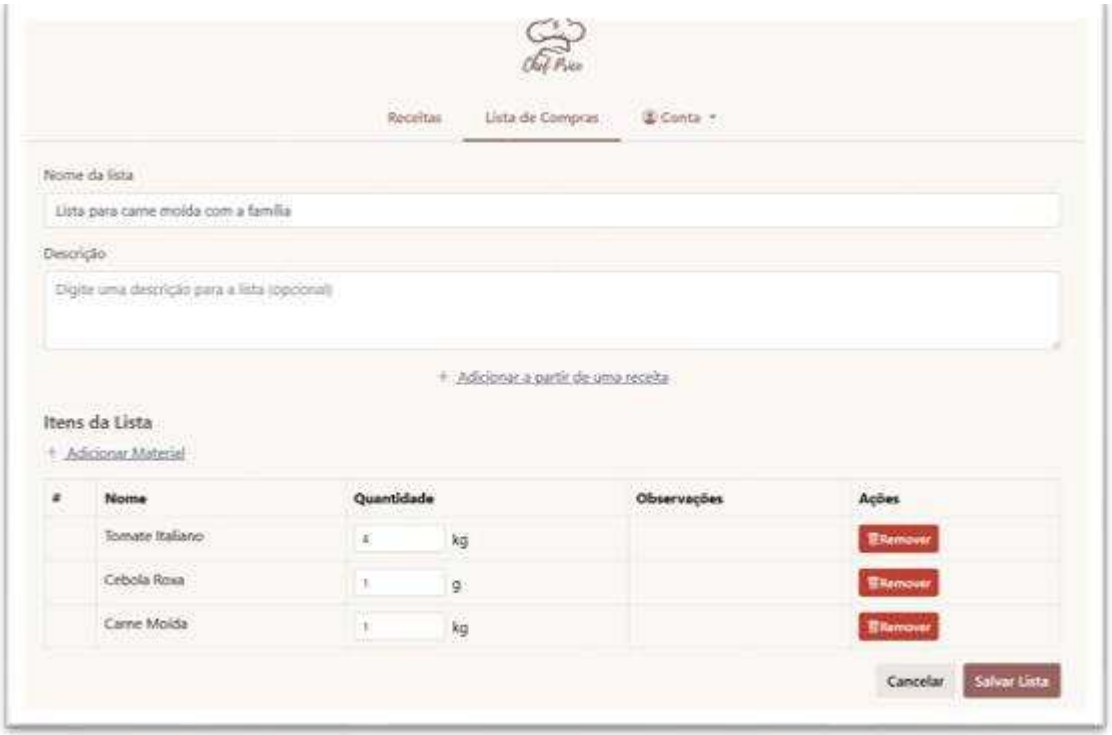


Fonte: Imagem do autor.

Figura 47. Tela de lista de compras

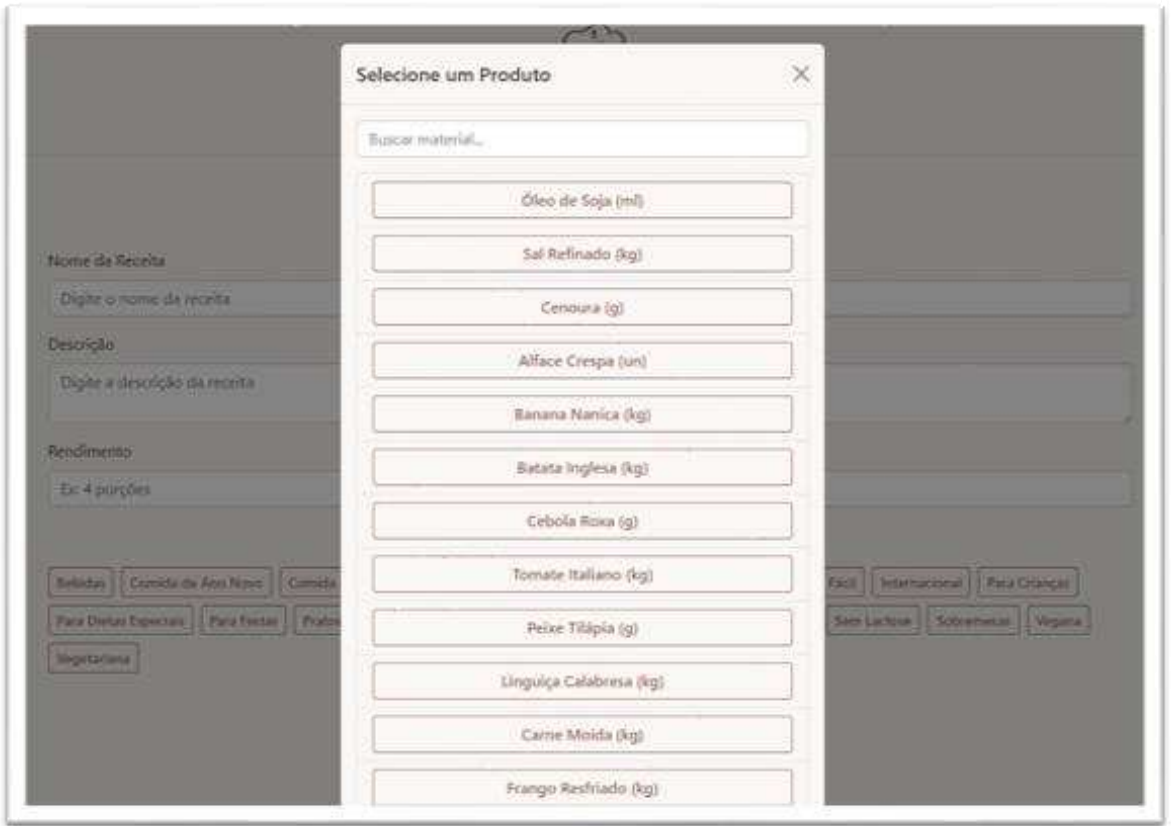
Fonte: Imagem do autor.

Figura 48.Tela de cadastro de lista de compras



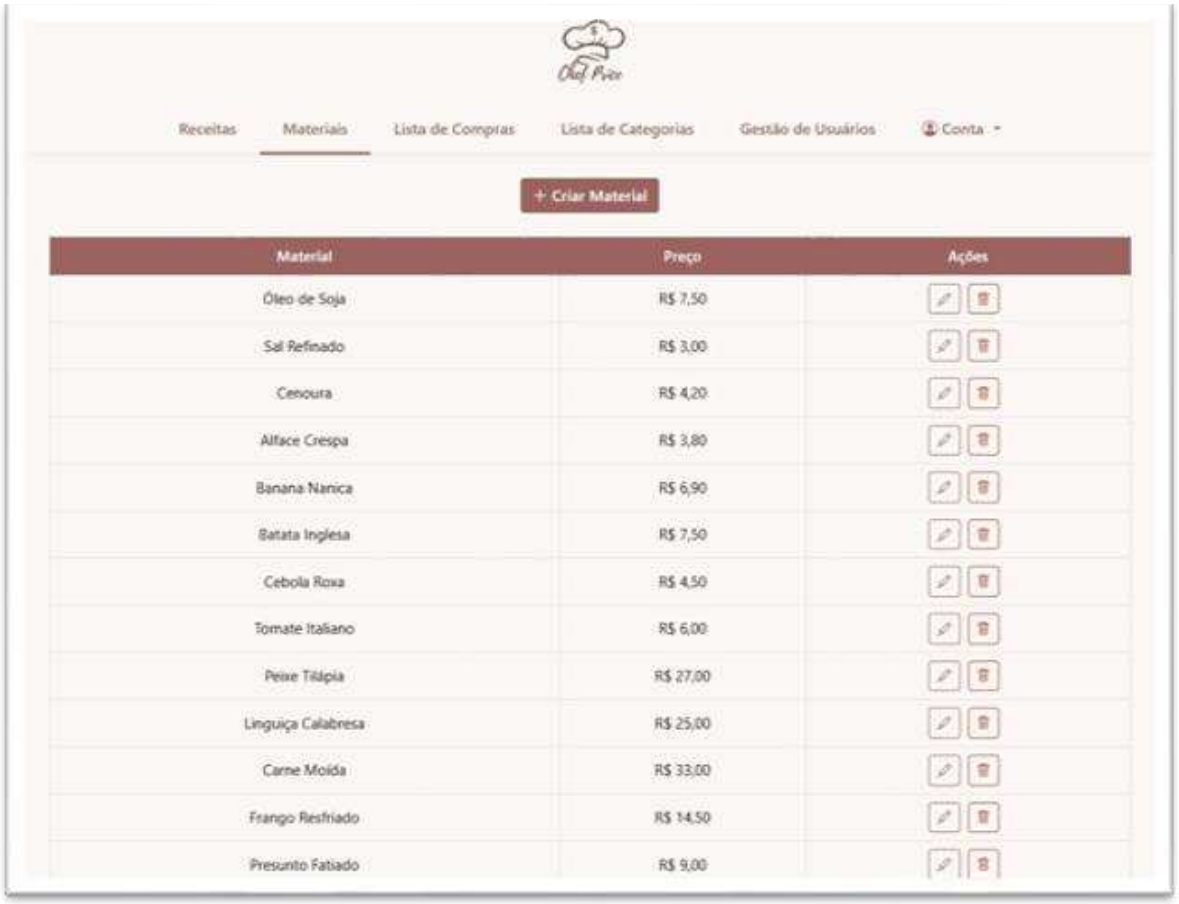
Fonte: Imagem do autor.

Figura 49.Tela de seleção de produto



Fonte: Imagem do autor.

Figura 50. Tela de materiais cadastrados



Fonte: Imagem do autor.

Figura 51. Tela de categorias cadastradas

Fonte: Imagem do autor.

Figura 52. Tela de gestão de usuários

Fonte: Imagem do autor.

17. DIVULGAÇÃO DO PRODUTO

A divulgação do **ChefPrice** será realizada por meio de uma estratégia de marketing digital orgânica e colaborativa, com foco na **rede de parceiros que contribuíram diretamente com o projeto** desde sua concepção. A proposta é utilizar canais acessíveis e autênticos para alcançar o público-alvo de forma eficaz, valorizando a credibilidade construída ao longo do desenvolvimento do sistema.

A principal frente de divulgação será feita **pelos próprios empreendedores que participaram das entrevistas, testes e validações de funcionalidades** durante o projeto. Esses parceiros, que já demonstraram interesse e necessidade por uma solução como o ChefPrice, atuarão como promotores iniciais do sistema, compartilhando suas experiências com colegas de profissão, grupos de confeitaria, panificação e demais nichos do setor alimentício.

Além disso, será incentivado o uso das **redes sociais dos parceiros** (como *Instagram*, *Facebook* e *WhatsApp Business*) para disseminar vídeos curtos, depoimentos reais e demonstrações do sistema em uso. Esse tipo de

conteúdo terá caráter espontâneo e autêntico, gerando identificação com o público e aumentando o alcance da ferramenta de forma orgânica.

A médio prazo, com base na resposta inicial, será avaliada a expansão da estratégia com:

- Criação de perfis oficiais nas redes sociais para publicação de dicas, tutoriais e atualizações do sistema;
- Parcerias com microinfluenciadores da área de confeitaria e empreendedorismo;
- Criação de materiais educativos (como e-books e blog posts) que abordem temas como precificação, controle de insumos e crescimento de pequenos negócios.

Com essa abordagem focada na **comunidade que inspirou o projeto**, espera-se consolidar o **ChefPrice** como uma solução reconhecida, relevante e próxima da realidade de seus usuários, destacando-se no mercado pela inovação, acessibilidade e utilidade prática.

18. CONCLUSÃO

Com base nos resultados obtidos até aqui, o ChefPrice se consolida como uma ferramenta essencial de apoio à gestão financeira para pequenos empreendedores da área alimentícia. No entanto, é possível avançar ainda mais na promoção de autonomia e eficiência desses profissionais por meio de melhorias futuras que ampliem o escopo analítico da plataforma.

Uma das principais inovações propostas para etapas futuras do projeto é a criação de gráficos de variação de preços com base nos valores dos materiais registrados no sistema ao longo dos meses. Essa funcionalidade permitirá ao usuário visualizar claramente a flutuação dos preços dos insumos, identificar tendências e compreender os melhores períodos e locais para realizar suas compras. Além disso, ao integrar o local de compra com a análise histórica de valores, o sistema poderá indicar em qual mercado o produto costuma ter menor custo, fornecendo uma recomendação estratégica para maximização do lucro.

A importância desse recurso torna-se ainda mais evidente quando relacionada ao debate público sobre desigualdade tributária. Como mostra o artigo

"Imposto alto para quem? Uma análise da desigualdade na tributação estadual a partir de renda, gênero e raça", publicado na Revista de Sociologia e Política (2024), o sistema tributário brasileiro, ao se apoiar majoritariamente em impostos indiretos como o ICMS, penaliza mais severamente os pequenos empreendedores e os grupos em maior vulnerabilidade social. Produtos essenciais, como alimentos e energia elétrica, frequentemente recebem as maiores alíquotas, aumentando o impacto na renda de quem já tem menos margem para absorver esses custos.

Dessa forma, ao oferecer análises automatizadas de preços com base em dados reais e atualizados, o ChefPrice assume um papel social relevante: ajudar seus usuários a enfrentarem a regressividade tributária na prática cotidiana, fornecendo informações que permitem decisões de compra mais conscientes e vantajosas. A incorporação de ferramentas como dashboards visuais e alertas de mais bem mercado para compra de insumos representa um passo à frente na democratização do acesso a recursos analíticos que antes estavam restritos a grandes empresas.

REFERÊNCIAS

Azevedo e Leone – Gestão financeira em pequenas empresas: AZEVEDO, José Gilmar de; LEONE, Rodrigo José Guerra. *Práticas de gestão financeira em micro e pequenas empresas.* Revista Ciências da Administração, Fortaleza, v. 17, n. 1, p. 55–83, jan./abr. 2011.

Silva – Extração de dados com IA em controle externo (TCU): SILVA, Luís André Dutra e. *Uso de técnicas de inteligência artificial para subsidiar ações de controle.* Revista do TCU, n. 137, p. 124–129, set./dez. 2016.

Bazzotti e Garcia – Sistemas de informação para decisão: BAZZOTTI, Cristiane; GARCIA, Elias. *A importância do sistema de informação gerencial na gestão empresarial para tomada de decisões.* Revista Ciências Sociais em Perspectiva, Cascavel, v. 2, n. 1, p. 21–32, 2003.

APÊNDICES

APÊNDICE A – MODELO DO FORMULÁRIO APLICADO

A seguir, apresentam-se os resultados obtidos com a aplicação do formulário de pesquisa utilizado durante o desenvolvimento do sistema **ChefPrice**. O objetivo da coleta foi compreender **os hábitos de precificação, controle de insumos e expectativas em relação ao uso de tecnologia** por parte de pequenos produtores de alimentos.

As respostas coletadas auxiliaram diretamente na **definição das funcionalidades do sistema**, especialmente no que diz respeito ao **cálculo de preço de venda**, **leitura automática de nota fiscal**, e **análise de variação de custos**.

1. Como você costuma definir o preço dos seus produtos alimentícios?

- 47,6% dos respondentes afirmam fazer um **cálculo aproximado com base em custos e aplicação de margem**.
- 33,3% **não costumam fazer cálculo**, baseando-se apenas no que consideram justo.
- Apenas 19% utilizam **planilhas ou sistemas próprios** para realizar esse controle.

Interpretação: A maioria dos empreendedores tem noção de custo, mas realiza estimativas manuais. Isso evidencia a oportunidade de entregar uma ferramenta mais precisa e automática, como o ChefPrice.

2. Você costuma guardar e analisar notas fiscais de compras para verificar se o valor de determinado insumo aumentou ou diminuiu com o tempo?

- 57,1% disseram que **fazem esse acompanhamento manualmente**.
- 33,3% **não fazem, mas gostariam de acompanhar de forma prática**.
- 14,3% **não guardam nem analisam notas fiscais**.

Interpretação: Há uma demanda clara por **automatização da leitura de notas fiscais** e acompanhamento de preços. Isso reforça a importância da funcionalidade de leitura via IA presente no ChefPrice.

3. Quão útil seria para você um sistema que armazena automaticamente os valores de cada compra a partir de uma foto da nota fiscal e mostra a variação?

- 100% dos participantes consideraram essa funcionalidade **muito útil**.
- Nenhum participante respondeu “pouco útil” ou “não vejo utilidade”.