

# Projeto Arquitetural de Software

**CookFood**

**Discentes:** Jéssica Cavalcante, João Victor Saraiva, Eduardo Felipe

**Docente:** Fabrício Schlag

---

## Índice

1. Introdução
  - 1.1. Objetivos
  - 1.2. Público Alvo
  - 1.3. Organização do Documento
2. Descrição Geral do Sistema
  - 2.1. Identificação e Missão do Sistema
  - 2.2. Descrição dos Interessados do Sistema
  - 2.3. Objetivos e Características do Sistema
  - 2.4. Regras de Negócio
3. Requisitos do Sistema
  - 3.1. Requisitos Funcionais
  - 3.2. Requisitos Não Funcionais
  - 3.3. Prototipação
  - 3.4. Métricas e Cronograma
4. Analise e Design
  - 4.1. Arquitetura do Sistema
  - 4.2. Padrões de Design
  - 4.3. Modelo do Domínio
  - 4.4. Diagrama de Classes
    - 4.4.1. Entidades
    - 4.4.2. Atributos e Métodos das Classes
    - 4.4.3. Relacionamentos
    - 4.4.4. Diagrama
  - 4.5. Diagrama de Sequência
    - 4.5.1. Diagrama de Cadastro e Autenticação de login

- 4.5.2. Diagrama de Busca de Receita
- 4.5.3. Diagrama de Adição de Receita
- 4.5.4. Diagrama de Visualização de Receita
- 4.5.5. Diagrama de Avaliação e Comentário

#### 4.6. Modelo de Dados

- 4.6.1. Modelo Lógico da Base de Dados
- 4.6.2. Criação Física do Modelo de Dados
- 4.6.3. Dicionário de Dados

#### 4.7. Ambiente de Desenvolvimento

#### 4.8. Sistemas e Componentes Externos Utilizados

### 5. Implementação

- 5.1. Configuração Inicial do Ambiente
- 5.2. Desenvolvimento de Microserviços
- 5.3. Orquestração e API Gateway

### 6. Teste

- 6.1. Estratégias de Teste
- 6.2. Pipeline de Testes Automatizados
- 6.3. Teste de Comunicação entre Microserviços

### 7. Verificação e Validação de Artefatos

- 7.1. Verificação dos Artefatos
  - 7.1.1. Verificação da Documentação
  - 7.1.2. Verificação de Modelagem
  - 7.1.3. Verificação dos Protótipos
  - 7.1.4. Revisão de Códigos

- 7.2. Validação de Artefatos
  - 7.2.1. Validação dos Requisitos
  - 7.2.2. Validação dos Funcional
  - 7.2.3. Validação Não Funcional

### 8. Implantação

- 8.1. Model Avaliação
- 8.2. View Avaliação
- 8.3. Controller Avaliação
- 8.4. Serializer Avaliação

- 8.5. Rotas (URLs)
- 8.6. Tabela Organizacional Padrão para Avaliação
- 9. Manual do Usuário
  - 9.1. Introdução
  - 9.2. Pré-Requisitos
  - 9.3. Primeiro Passos
  - 9.4. Funcionalidades Principais
  - 9.5. Solução de Problemas
- 10. Conclusão
- 11. Bibliografia

## 1. Introdução

Este documento tem como objetivo fornecer uma visão geral do projeto de desenvolvimento de um "site e aplicativo de receitas" Cookfood, abrangendo suas funcionalidades, especificações técnicas e etapas de implementação. Ele servirá como um guia para os envolvidos no desenvolvimento do software, garantindo uma compreensão clara de seus requisitos, arquitetura, e processo de evolução ao longo do ciclo de vida do projeto.

O sistema será desenvolvido para o mercado de gastronomia, com foco em usuários que desejam acessar, compartilhar e gerenciar receitas de forma prática e interativa. A aplicação busca oferecer uma plataforma robusta e intuitiva tanto para cozinheiros amadores quanto para profissionais, atendendo às demandas de um público diversificado que busca inspiração e organização no preparo de refeições.

Este documento destina-se a desenvolvedores, gerentes de projeto e demais partes interessadas, e abrange as informações essenciais sobre as características e o funcionamento da solução, orientando o time nas fases de planejamento, desenvolvimento, testes e implantação. As seções a seguir descrevem os requisitos necessários para a implementação do sistema, abrangendo aspectos técnicos e operacionais que garantam uma plataforma eficiente, acessível e segura para todos os tipos de usuários.

### 1.1. Objetivos

- Descrever o escopo do sistema
- Apresentar uma visão detalhada dos requisitos funcionais
-

Apresentar uma visão detalhada dos requisitos não funcionais

- Descrever os principais casos de uso
- Definir as interações dos usuários com o sistema

## 1.2. Público alvo

O CookFood é projetado para atender diferentes perfis de usuários apaixonados por culinária. O sistema é ideal para cozinheiros amadores e entusiastas que buscam inspiração para novas receitas ou desejam compartilhar suas próprias criações. Ele também atrai profissionais da área gastronômica, como chefs e estudantes de gastronomia, que podem usar a plataforma para descobrir novas receitas e compartilhar seu conhecimento.

Além disso, o CookFood atende pessoas que procuram receitas específicas, seja por preferências alimentares, dietas, restrições ou até mesmo pela necessidade de preparar algo com os ingredientes que já têm em casa. A plataforma também é uma excelente ferramenta para blogueiros e criadores de conteúdo culinário, que podem compartilhar receitas e conectar-se com uma comunidade de apaixonados por gastronomia.

## 1.3. Organização do documento

O documento está organizado em 6 tópicos. No 1º tópico, tem-se a apresentação geral do documento, descrevendo seus propósitos e seu contexto. No 2º temos a descrição do problema, onde descrevemos detalhadamente o escopo do sistema, explanando suas necessidades e desafios tendo uma visão geral do sistema proposto, destacando suas principais funcionalidades e benefícios. No 3º tópico, são apresentados os casos de uso e requisitos funcionais, com suas identificações e descrições, detalhando as funcionalidades específicas que o sistema deve oferecer para atender às necessidades da empresa. No 4º definimos as regras de negócios que o sistema deve manipular, incluindo os tipos de dados necessários para o correto funcionamento do sistema. O 5º tópico, enumera e descreve os requisitos não funcionais do sistema e identifica restrições que possam impactar o desenvolvimento ou implementação do sistema. Diante disso, temos o 6º tópico onde mostramos o protótipo inicial do sistema, o qual servirá para testar o usuário e coletar feedbacks. No 7º e último tópico, tem a bibliografia do documento, onde são listadas as referências utilizadas na elaboração do documento.

## 2. Descrição do sistema

Nesta seção, será feita uma descrição detalhada do escopo que o sistema de catálogo de receitas online da CookFood está inserido, bem como suas características específicas do problema no contexto em que o sistema está inserido.

## 2.1. Identificação e missão do Sistema

O **CookFood** é uma plataforma online que oferece um catálogo de receitas, permitindo que usuários acessem, compartilhem e gerenciem receitas de forma interativa e intuitiva. O sistema está inserido no mercado da gastronomia, atendendo tanto cozinheiros amadores quanto profissionais. Sua principal funcionalidade é proporcionar uma interface amigável e robusta para que os usuários explorem uma vasta gama de receitas categorizadas, além de criar e compartilhar suas próprias receitas com a comunidade.

O sistema abrange funcionalidades que facilitam a navegação, organização e personalização da experiência do usuário, além de possibilitar a interação social entre os membros da plataforma.

## 2.2. Descrição dos interessados do sistema

<b>Interessados</b>	<b>Descrição</b>
Usuários Finais	São os principais utilizadores da plataforma, incluindo cozinheiros amadores, profissionais e entusiastas que buscam, compartilham ou interagem com receitas no sistema.
Desenvolvedores de Software	Responsáveis pela implementação e manutenção técnica do sistema, garantindo que as funcionalidades sejam entregues de forma eficiente e conforme os requisitos estabelecidos.
Patrocinadores	São investidores ou empresas que apoiam financeiramente o desenvolvimento e manutenção do sistema, buscando retorno através de parcerias, publicidade ou monetização da plataforma.
Fornecedores de Conteúdo	Incluem blogueiros, chefs e outros criadores que contribuem com receitas e conteúdos relacionados, enriquecendo o catálogo do <b>CookFood</b> e atraindo mais usuários.
Entidades Reguladoras	Organizações responsáveis por garantir que o sistema esteja em conformidade com as legislações vigentes, como a proteção de dados (LGPD, GDPR) e normas de publicidade de produtos alimentícios.

## 2.3. Objetivos e características esperadas

Esta seção deve listar, de acordo com a visão dos interessados no sistema, todos os objetivos e características fundamentais do sistema a ser desenvolvido e os respectivos benefícios esperados para o negócio ou para o interessado.

<b>Interessados</b>	<b>Objetivo ou característica desejada</b>	<b>Benefício adquirido</b>
Usuários Finais	Acessar, compartilhar e interagir com receitas de forma simples e intuitiva.	Ampla diversidade de receitas e uma comunidade colaborativa para aprendizado e inspiração.
Desenvolvedores de Software	Desenvolver um sistema eficiente, escalável e fácil de manter.	Criar um produto de qualidade, de fácil manutenção e que atenda às expectativas dos usuários.
Patrocinadores	Investir no desenvolvimento do sistema para obter retorno financeiro.	Ganho de visibilidade, parcerias comerciais e potencial de monetização da plataforma.
Fornecedores de Conteúdo	Compartilhar suas receitas e conteúdos para ganhar visibilidade e engajamento.	Maior exposição, interação com público interessado e possível geração de receitas.
Entidades Reguladoras	Garantir que o sistema esteja em conformidade com as leis e normas aplicáveis.	Cumprimento das regulamentações e garantia de que os dados e atividades na plataforma sejam seguros e legais.

## 2.4. Regras de Negócio

<b>Ref.</b>	<b>Descrição</b>	<b>Casos de uso</b>
RN01	O sistema deve armazenar informações básicas dos usuários, como nome, e-mail, senha	UC01, UC07

	(devidamente criptografada) e foto de perfil (opcional). Além disso, deve manter dados de preferências como receitas favoritas e histórico de interações no sistema.	
RN02	Cada receita no sistema deve conter informações detalhadas, incluindo título, descrição, ingredientes, modo de preparo, tempo de preparo, tempo de cozimento, porções, categoria da receita (sobremesa, prato principal, etc.), restrições alimentares (vegana, sem glúten, etc.), e uma imagem ilustrativa.	UC05
RN03	O sistema deve armazenar os comentários feitos pelos usuários nas receitas, além de permitir que eles atribuam uma nota de 1 a 5 estrelas. Esses dados devem ser vinculados ao perfil do usuário e à receita correspondente.	UC04
RN04	As receitas devem estar organizadas em categorias, como pratos principais, sobremesas, entradas, entre outros. Também deve haver informações para filtros de busca, como tempo de preparo, dificuldade, tipo de dieta e ingredientes principais.	UC02
RN05	O sistema deve coletar dados sobre o uso geral, como o número de receitas acessadas, compartilhadas e avaliadas.	UC06, UC04
RN06	O sistema deve manter um registro das receitas compartilhadas por usuários, seja por redes sociais ou por link direto, permitindo monitorar o nível de engajamento com o conteúdo.	UC06

### 3. Casos de uso e requisitos funcionais

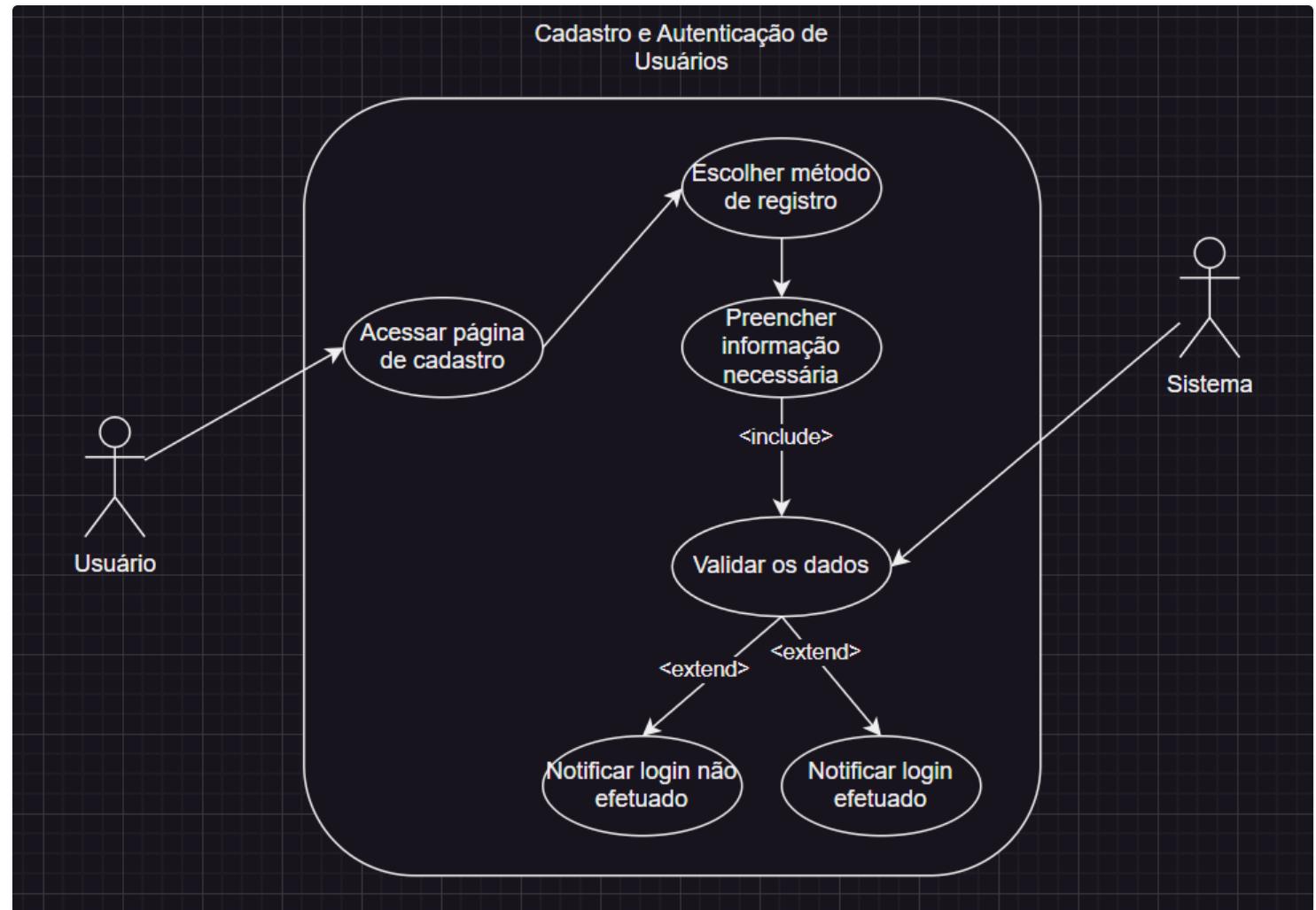
Casos de uso e requisitos funcionais são elementos cruciais na especificação do sistema para a CookFood. Os casos de uso descrevem as interações entre os atores e o sistema, identificando as principais funcionalidades e fluxos de trabalho. Já os requisitos funcionais estabelecem as características e comportamentos esperados do sistema, como a capacidade de hospedar receitas, gerar interação com o usuário e fornecer informações em tempo real. Esses elementos

são fundamentais para garantir que o sistema atenda às necessidades dos usuários e proporcione uma experiência eficiente e eficaz.

### 3.1.1. Diagramas de caso de uso

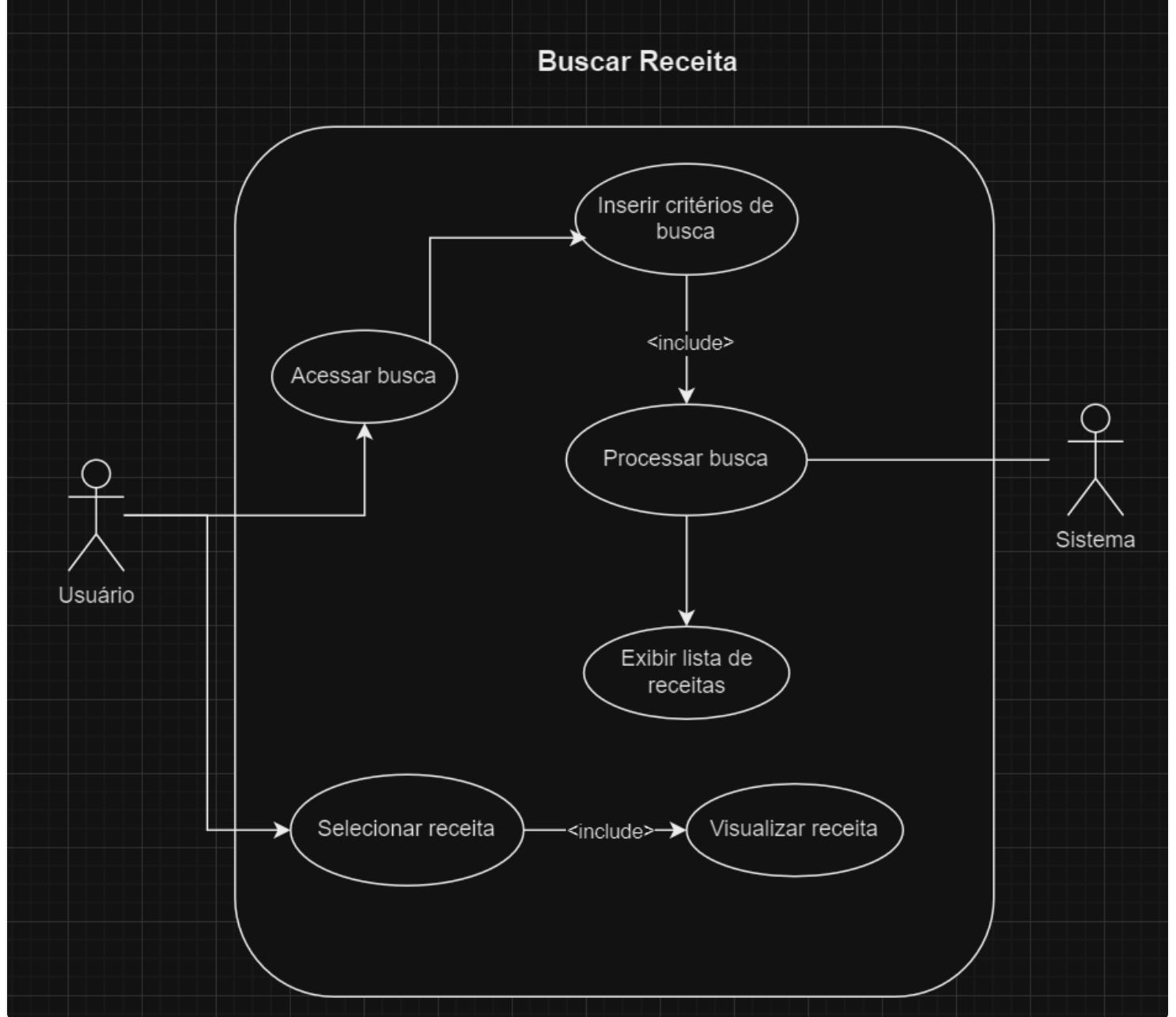
Um Caso de Uso UML descreve como os usuários (atores) interagem com o sistema para atingir objetivos específicos. Ele é composto por diagramas que mostram as funcionalidades principais, os atores envolvidos, e as relações entre eles.

#### UC01 - Cadastro e Autenticação de Usuário



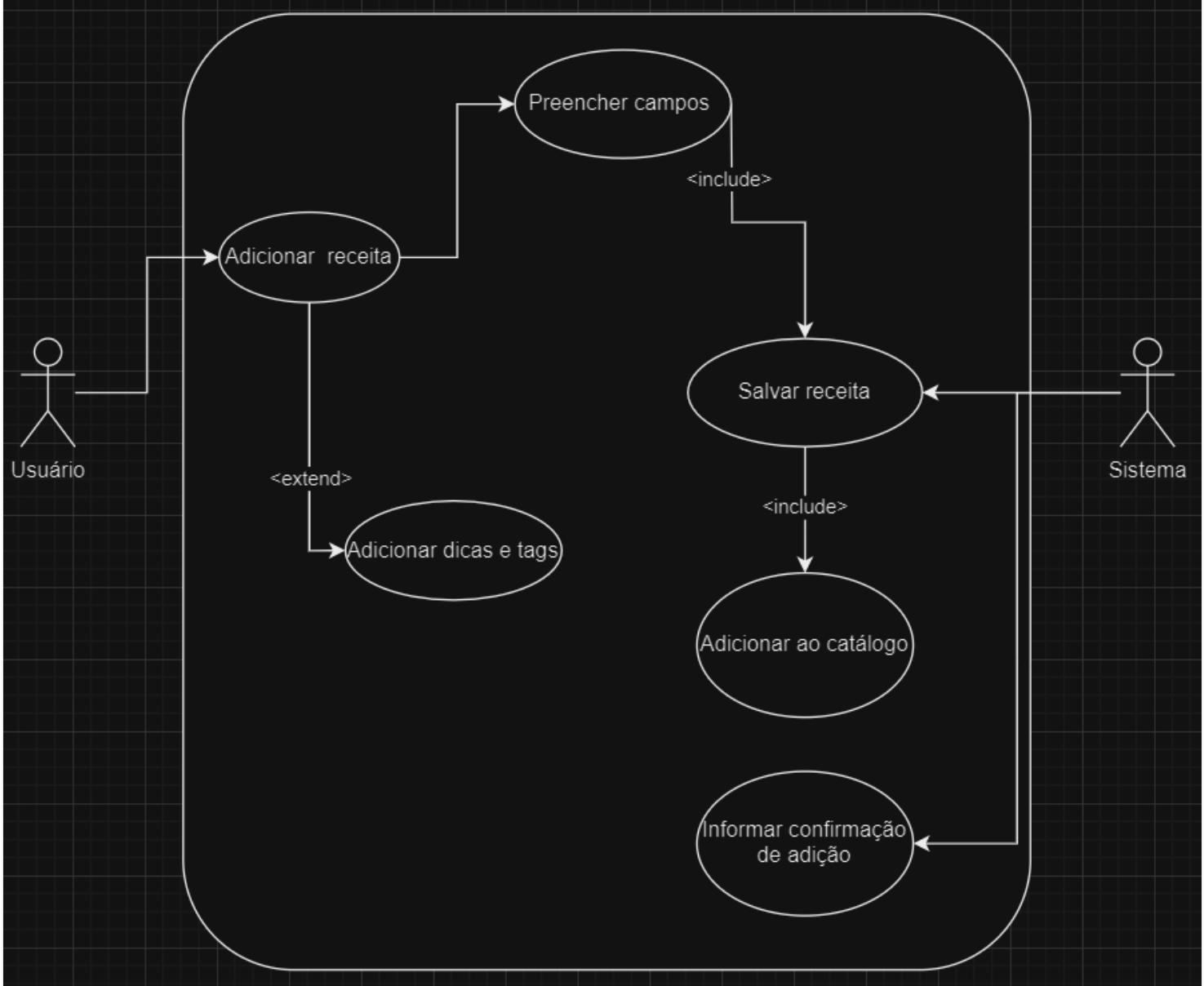
#### UC02 - Buscar Receita

## Buscar Receita



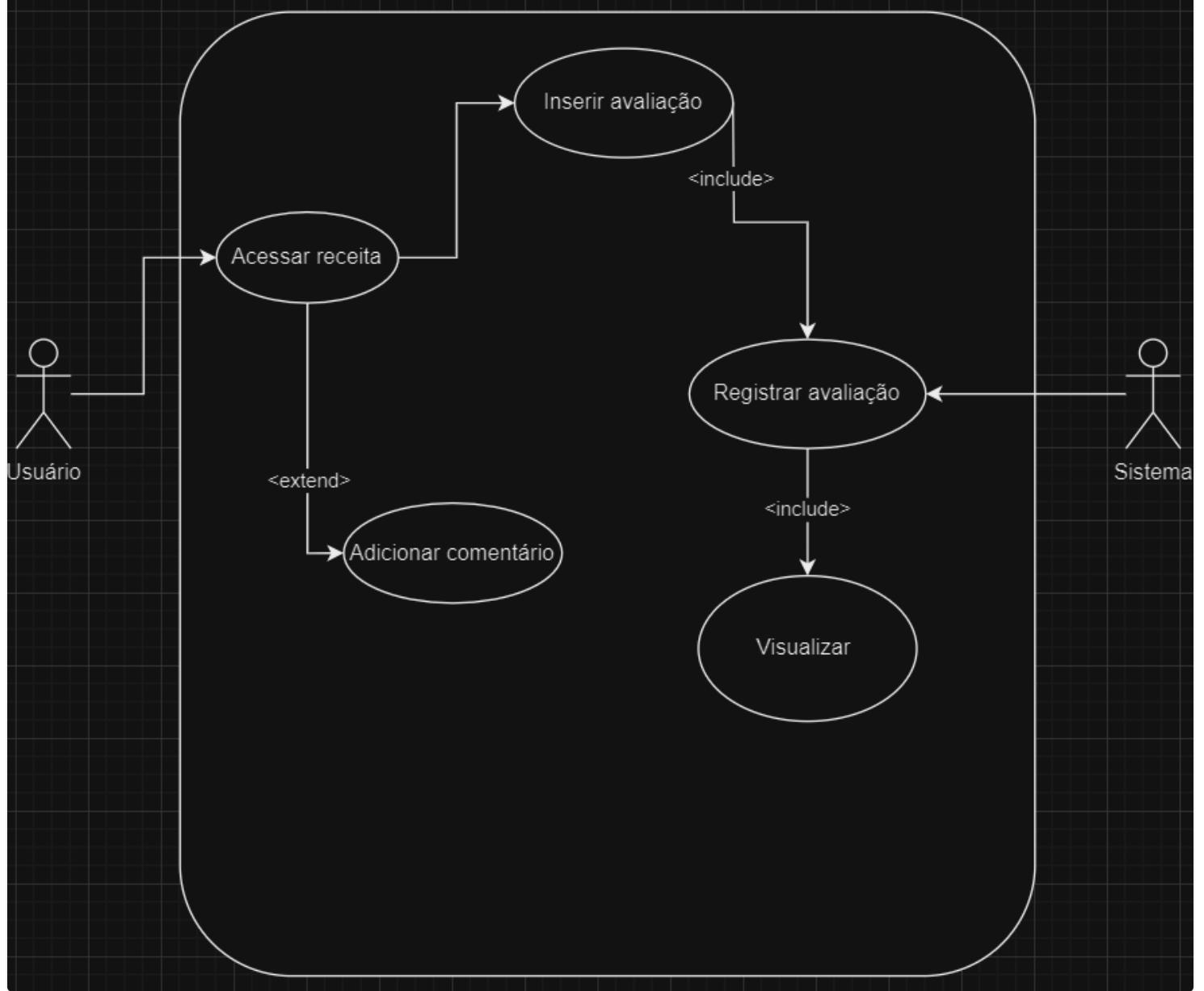
**UC03** - Adição de receita

## Adição de receita



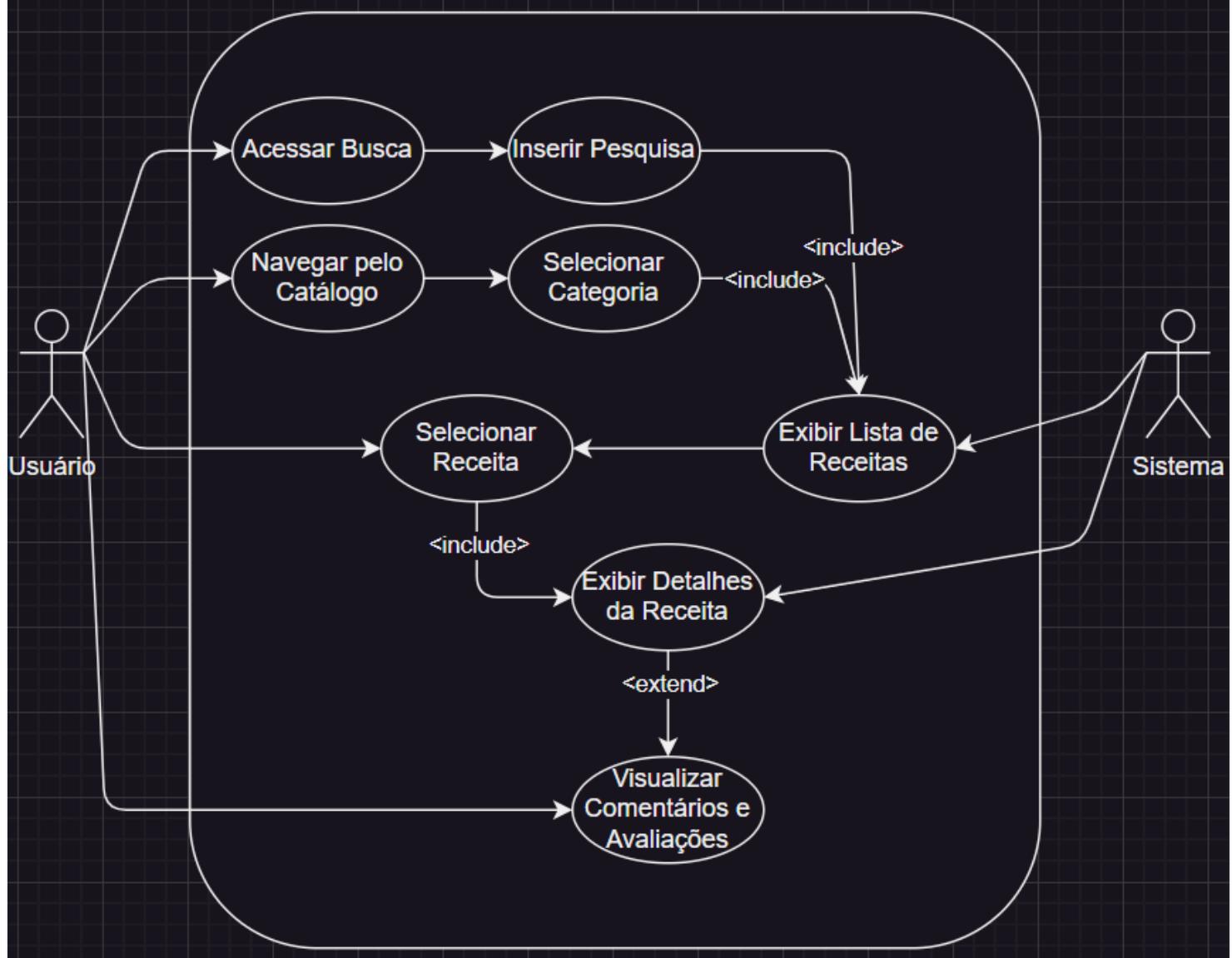
**UC04** - Avaliação e comentário de Receitas

## Avaliação e comentário de Receitas



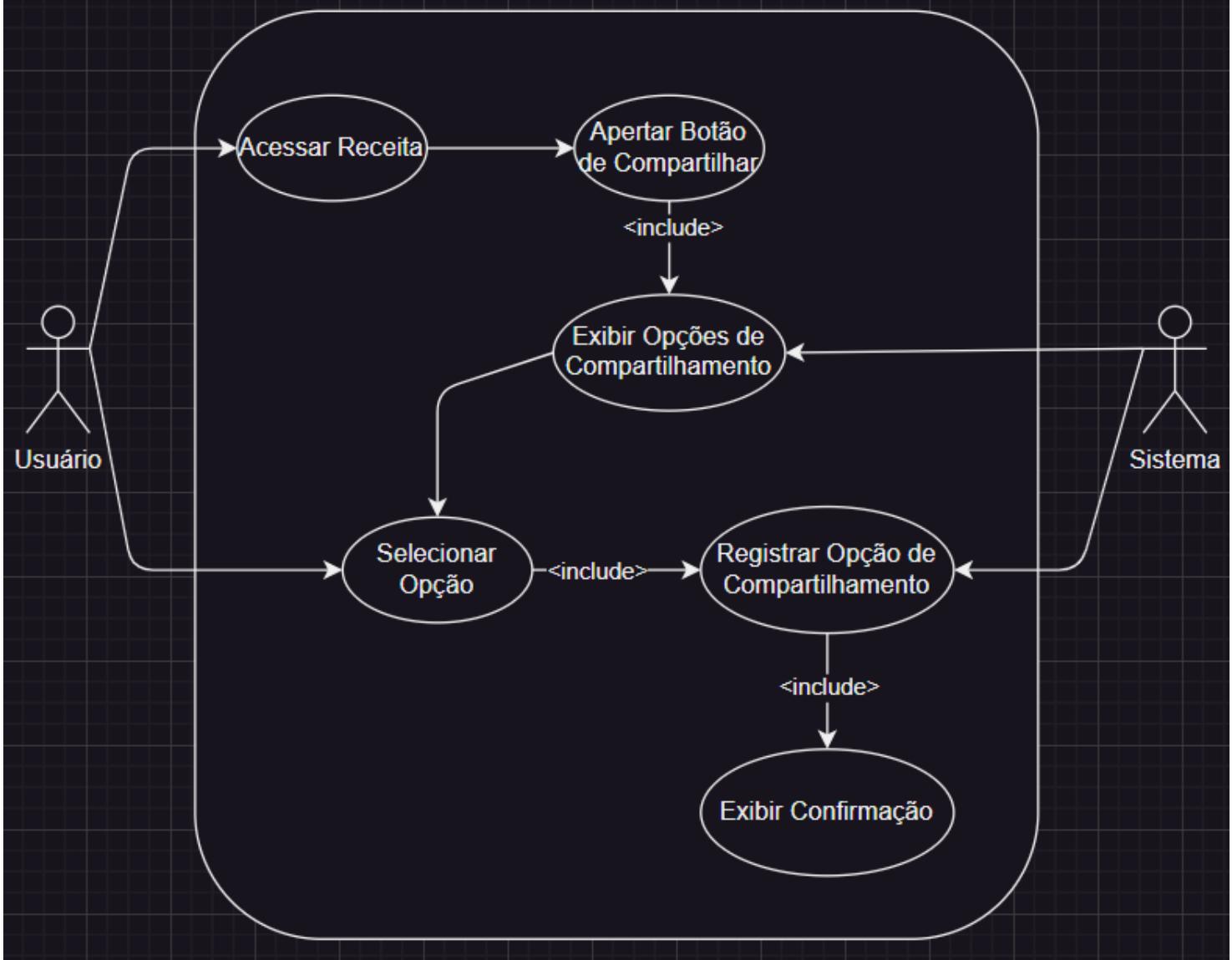
**UC05** - Visualização de Receitas

## Visualização de Receitas

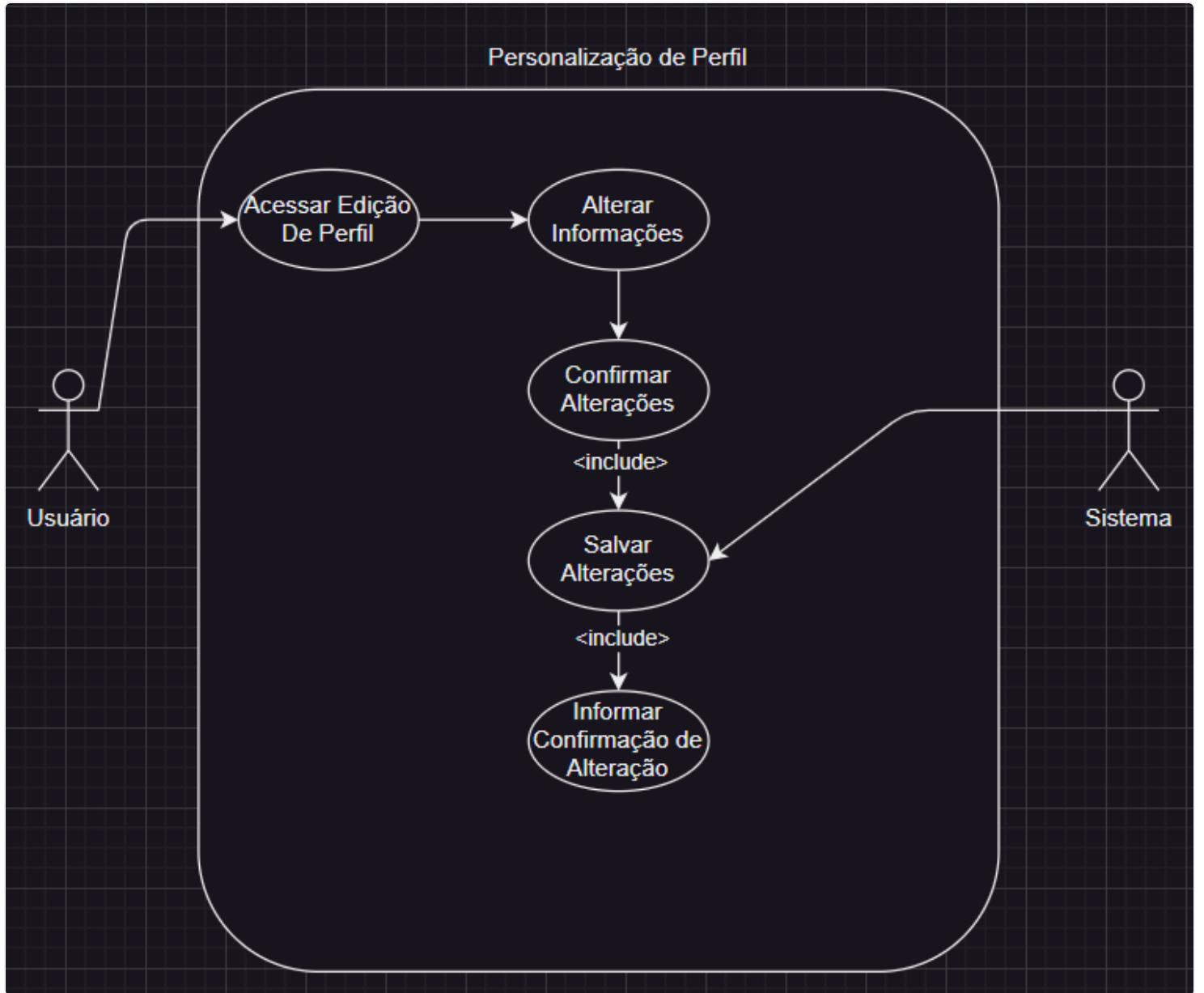


**UC06** - Compartilhar Receita

## Compartilhar Receita



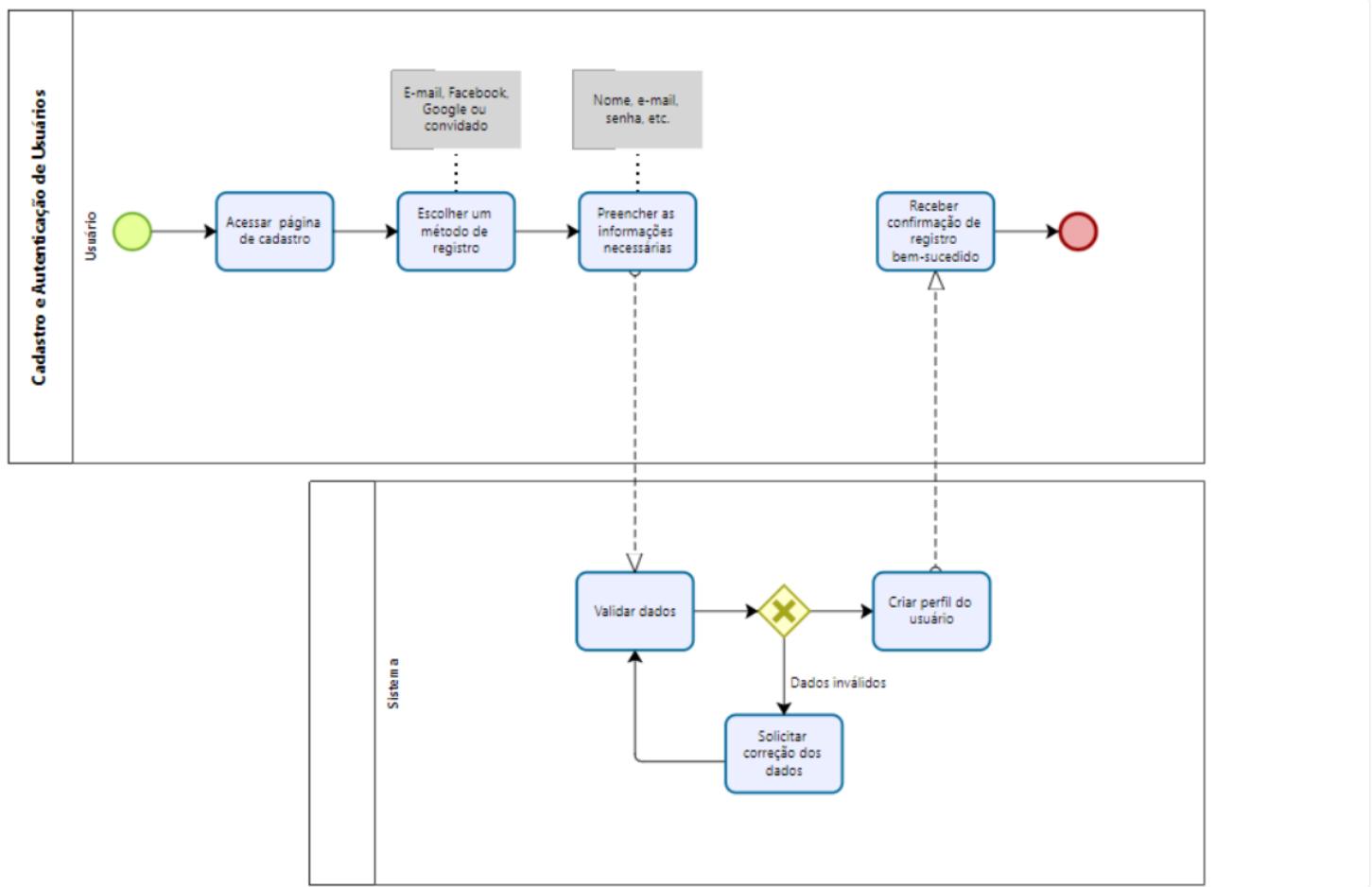
**UC07** - Personalização de Perfil



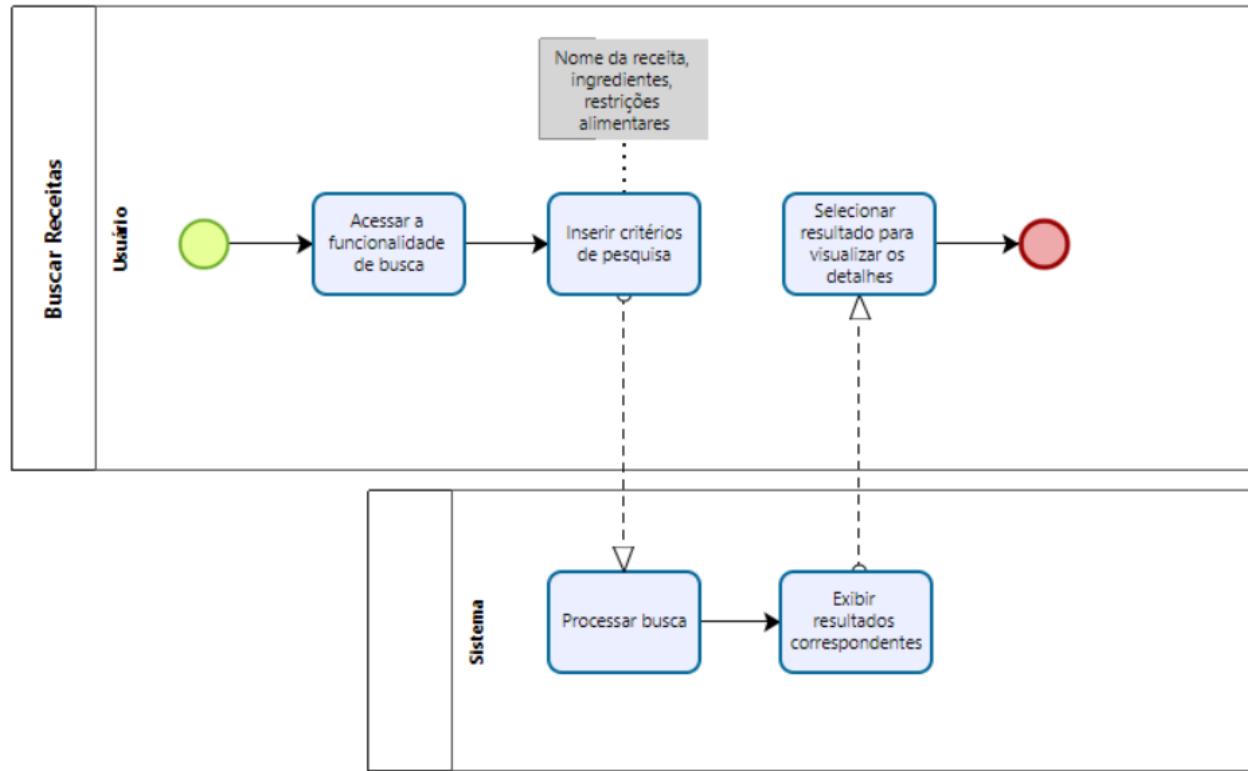
### 3.1.2. Diagramas BPMN

O BPMN é uma notação padrão utilizada para modelar e representar visualmente processos de negócios, permitindo que sejam compreendidos tanto por profissionais de negócios quanto por desenvolvedores técnicos.

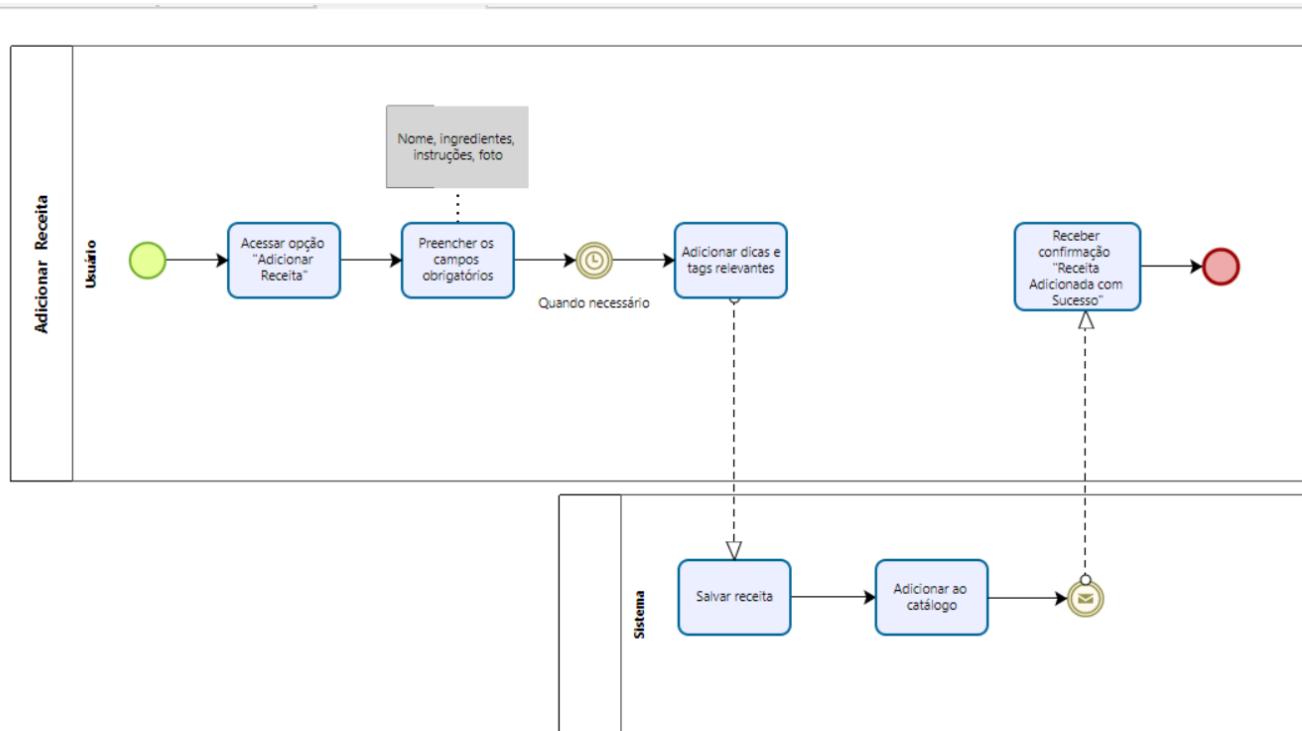
No projeto CookFood, a modelagem de processos através do BPMN será fundamental para garantir que todas as interações e fluxos de atividades, desde o cadastro de receitas até a gestão de conteúdo e moderação, sejam claramente definidos e otimizados. Através desta notação, será possível mapear de forma detalhada os principais processos do sistema, identificar possíveis gargalos e garantir que todas as etapas de interação do usuário com a plataforma sejam eficazes e eficientes.



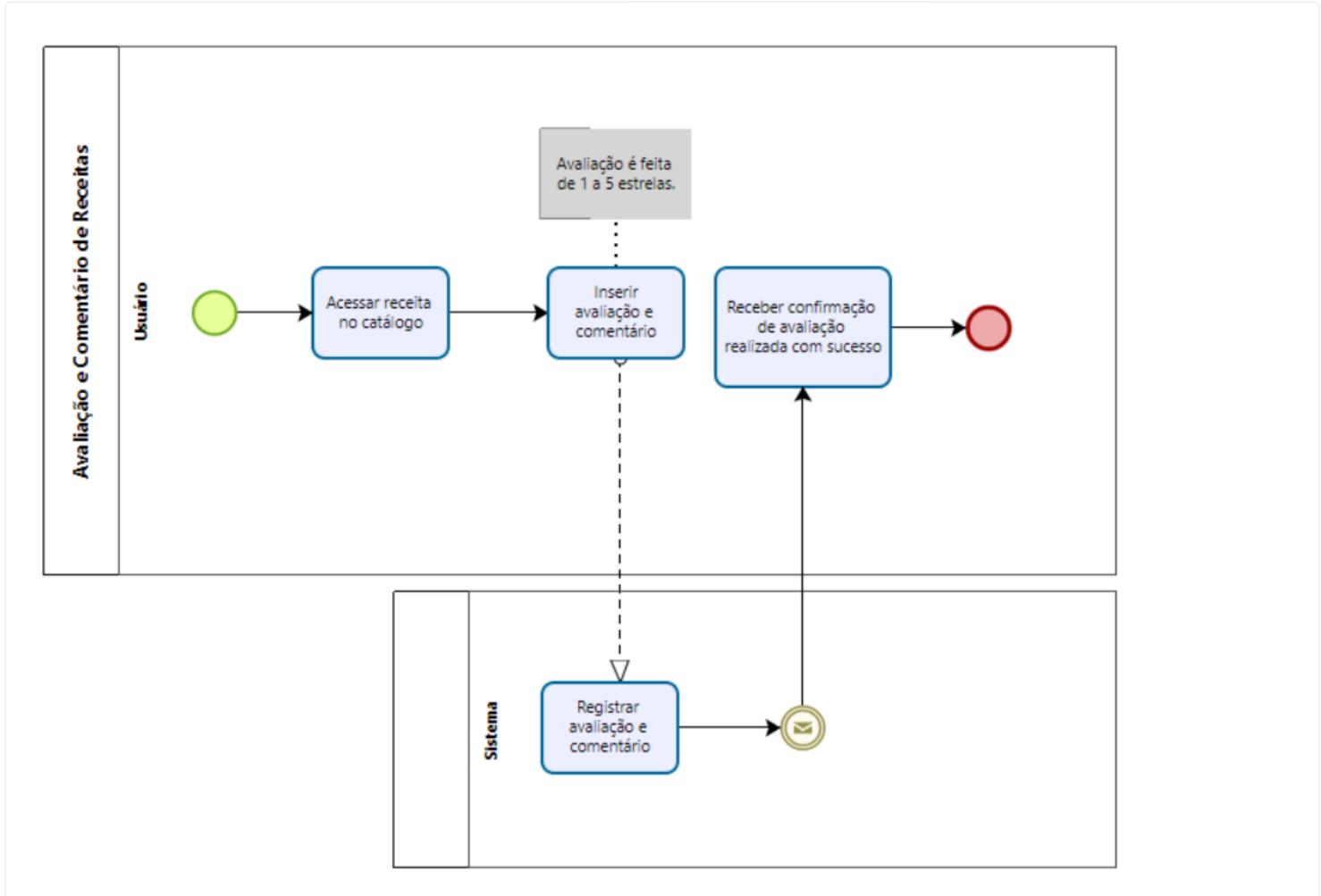
## BPMN02 - Buscar Receitas



### BPMN03 - Adição de Receita



## BPMN04 - Avaliação e Comentário de Receitas



### 3.2. Descrição de caso de uso

UC01 - Cadastro e Autenticação de Usuários	
<b>Atores</b>	Usuário
<b>Definição</b>	O usuário pode se registrar no sistema por meio de e-mail, contas de redes sociais ou como convidado.
<b>Gatilho</b>	O usuário apertou o botão de criar conta.
<b>Pré Condições</b>	<ul style="list-style-type: none"> <li>• O usuário deve ter acesso à internet.</li> <li>• O usuário deve possuir um endereço de e-mail válido ou uma conta em uma rede social suportada (Facebook, Google).</li> </ul>
<b>Pós Condições</b>	<ul style="list-style-type: none"> <li>• O sistema deve criar um perfil de usuário.</li> </ul>

	<ul style="list-style-type: none"> <li>• O usuário deve receber uma confirmação de registro bem-sucedido.</li> <li>• O usuário deve estar autenticado e logado no sistema.</li> </ul>
<b>Requisitos Funcionais</b>	RF01, RF07
<b>Fluxo principal</b>	<ol style="list-style-type: none"> <li>1. O usuário acessa a página de cadastro.</li> <li>2. O usuário escolhe um método de registro (e-mail, Facebook, Google ou convidado).</li> <li>3. O usuário preenche as informações necessárias (nome, e-mail, senha, etc.).</li> <li>4. O sistema valida os dados e cria o perfil do usuário.</li> <li>5. O usuário recebe uma confirmação de registro bem-sucedido.</li> </ol>
<b>Fluxo Alternativo</b>	<ol style="list-style-type: none"> <li>1. O usuário acessa a página de cadastro.</li> <li>2. O usuário escolhe um método de registro (e-mail, Facebook, Google ou convidado).</li> <li>3. O usuário preenche as informações necessárias (nome, e-mail, senha, etc.).</li> <li>4. O sistema valida os dados e cria o perfil do usuário.</li> <li>5. O usuário não recebe a confirmação de registro devido a um problema no servidor.</li> <li>6. O sistema exibe uma mensagem de erro informando que o registro não pôde ser concluído e solicita que o usuário tente novamente mais tarde.</li> <li>7. O usuário tenta novamente após algum tempo e o registro é concluído com sucesso.</li> </ol>

<b>UC02 - Buscar Receitas</b>	
<b>Atores</b>	Usuário
<b>Definição</b>	O usuário pode buscar receitas usando critérios específicos.
<b>Gatilho</b>	O usuário apertou o botão de buscar ou na lupa.

<b>Pré Condições</b>	<ul style="list-style-type: none"> <li>• O usuário deve estar autenticado.</li> </ul>
<b>Pós Condições</b>	<ul style="list-style-type: none"> <li>• O sistema deve exibir uma lista de receitas correspondentes aos critérios de busca inseridos pelo usuário.</li> <li>• O usuário deve poder visualizar os detalhes das receitas selecionadas.</li> </ul>
<b>Requisitos Funcionais</b>	RF02, RF03
<b>Fluxo principal</b>	<ol style="list-style-type: none"> <li>1. O usuário acessa a funcionalidade de busca.</li> <li>2. O usuário insere critérios de pesquisa (nome da receita, ingredientes, restrições alimentares).</li> <li>3. O sistema processa a busca e exibe uma lista de receitas correspondentes.</li> <li>4. O usuário pode selecionar uma receita para visualizar os detalhes.</li> </ol>

<b>UC03 - Adição de Receita</b>	
<b>Atores</b>	Usuário
<b>Definição</b>	O usuário pode adicionar sua própria receita ao catálogo.
<b>Gatilho</b>	O usuário apertou o botão de adicionar receita.
<b>Pré Condições</b>	<ul style="list-style-type: none"> <li>• O usuário deve estar autenticado.</li> </ul>
<b>Pós Condições</b>	<ul style="list-style-type: none"> <li>• O sistema deve salvar a receita e adicioná-la ao catálogo.</li> <li>• O usuário deve receber uma confirmação de que a receita foi adicionada com sucesso.</li> <li>• A nova receita deve estar visível e acessível para outros usuários no catálogo.</li> </ul>
<b>Requisitos Funcionais</b>	RF04
<b>Fluxo principal</b>	<ol style="list-style-type: none"> <li>1. O usuário acessa a opção "Adicionar Receita".</li> </ol>

	<ol style="list-style-type: none"> <li>2. O usuário preenche os campos obrigatórios (nome, ingredientes, instruções, foto).</li> <li>3. O usuário pode adicionar dicas e tags relevantes.</li> <li>4. O sistema salva a receita e a adiciona ao catálogo.</li> <li>5. O usuário recebe uma confirmação de que a receita foi adicionada com sucesso.</li> </ol>
<b>Fluxo Alternativo</b>	<ol style="list-style-type: none"> <li>1. O usuário tenta adicionar uma receita, mas os dados inseridos são inválidos.</li> <li>2. O sistema exibe uma mensagem de erro informando ao usuário sobre os dados inválidos e solicita a correção dos campos.</li> <li>3. O usuário corrige os erros e tenta novamente.</li> <li>4. O sistema valida os dados corrigidos e a receita é adicionada com sucesso.</li> </ol>

<b>UC04 - Avaliação e Comentário de Receitas</b>	
<b>Atores</b>	Usuário
<b>Definição</b>	O usuário pode avaliar e comentar sobre receitas de outros usuários.
<b>Gatilho</b>	O usuário apertou o botão de adicionar comentário ou avaliação.
<b>Pré Condições</b>	<ul style="list-style-type: none"> <li>• O usuário deve estar autenticado.</li> </ul>
<b>Pós Condições</b>	<ul style="list-style-type: none"> <li>• O sistema deve registrar a avaliação e o comentário do usuário.</li> <li>• A avaliação e o comentário devem ser exibidos na página da receita correspondente.</li> <li>• O usuário deve receber uma confirmação de que a avaliação foi realizada com sucesso.</li> <li>• O comentário deve estar visível para outros usuários autenticados.</li> <li>• A média das avaliações da receita deve ser atualizada automaticamente.</li> </ul>
<b>Requisitos Funcionais</b>	RF05

<b>Fluxo principal</b>	<ol style="list-style-type: none"> <li>1. O usuário acessa uma receita no catálogo.</li> <li>2. O usuário insere uma avaliação (de 1 a 5 estrelas).</li> <li>3. O usuário pode adicionar um comentário sobre a receita.</li> <li>4. O sistema registra a avaliação e o comentário.</li> <li>5. O usuário visualiza a confirmação de que a avaliação foi realizada com sucesso.</li> </ol>
<b>Fluxo Alternativo</b>	<ol style="list-style-type: none"> <li>1. O usuário acessa uma receita no catálogo.</li> <li>2. O usuário insere uma avaliação (de 1 a 5 estrelas).</li> <li>3. O usuário tenta adicionar um comentário sobre a receita, mas a conexão com a internet é perdida.</li> <li>4. O sistema exibe uma mensagem de erro informando que a conexão foi perdida e que o comentário não pôde ser registrado.</li> <li>5. O usuário reconecta à internet e tenta novamente adicionar o comentário.</li> <li>6. O sistema registra a avaliação e o comentário.</li> <li>7. O usuário visualiza a confirmação de que a avaliação foi realizada com sucesso.</li> </ol>

### UC05 - Visualização de Receitas

<b>Atores</b>	Usuário
<b>Definição</b>	O usuário pode visualizar as receitas disponíveis no catálogo.
<b>Gatilho</b>	O usuário seleciona uma receita na lista de busca ou navega pelo catálogo de receitas.
<b>Pré Condições</b>	<ul style="list-style-type: none"> <li>• O usuário deve estar autenticado.</li> <li>• O sistema deve ter receitas cadastradas e disponíveis para visualização.</li> </ul>
<b>Pós Condições</b>	<ul style="list-style-type: none"> <li>• O sistema deve exibir os detalhes completos da receita selecionada, incluindo ingredientes, modo de preparo, tempo de preparo, e avaliações.</li> </ul>

	<ul style="list-style-type: none"> <li>• O usuário deve poder visualizar comentários e avaliações de outros usuários sobre a receita.</li> </ul>
<b>Requisitos Funcionais</b>	RF03
<b>Fluxo principal</b>	<ol style="list-style-type: none"> <li>1. O usuário acessa a funcionalidade de busca ou navega pelo catálogo de receitas.</li> <li>2. O usuário insere critérios de pesquisa ou seleciona uma categoria de receitas.</li> <li>3. O sistema exibe uma lista de receitas correspondentes aos critérios de busca ou categoria selecionada.</li> <li>4. O usuário seleciona uma receita da lista.</li> <li>5. O sistema exibe os detalhes completos da receita selecionada.</li> <li>6. O usuário visualiza os ingredientes, modo de preparo, tempo de preparo, e avaliações da receita.</li> <li>7. O usuário pode visualizar comentários e avaliações de outros usuários sobre a receita.</li> </ol>
<b>Fluxo Alternativo</b>	<ol style="list-style-type: none"> <li>1. O usuário tenta visualizar uma receita, mas há uma falha na conexão com o servidor.</li> <li>2. O sistema exibe uma mensagem informando que não foi possível recuperar os dados da receita devido a um erro de conexão e solicita que o usuário tente novamente mais tarde.</li> <li>3. O usuário reconecta e tenta visualizar a receita novamente.</li> <li>4. O sistema exibe os detalhes completos da receita após a conexão ser restaurada.</li> </ol>

<b>UC06 - Compartilhar Receita</b>	
<b>Atores</b>	Usuário
<b>Definição</b>	O usuário pode compartilhar uma receita com outros usuários da plataforma.
<b>Gatilho</b>	O usuário apertou o botão de adicionar comentário ou avaliação.

<b>Pré Condições</b>	<ul style="list-style-type: none"> <li>O usuário deve estar autenticado.</li> </ul>
<b>Pós Condições</b>	<ul style="list-style-type: none"> <li>O sistema deve registrar a ação de compartilhamento da receita.</li> <li>A receita compartilhada deve estar visível para outros usuários da plataforma.</li> <li>O usuário deve receber uma confirmação de que a receita foi compartilhada com sucesso.</li> </ul>
<b>Requisitos Funcionais</b>	RF08
<b>Fluxo principal</b>	<ol style="list-style-type: none"> <li>O usuário acessa a receita que deseja compartilhar.</li> <li>O usuário aperta o botão de compartilhar receita.</li> <li>O sistema exibe opções de compartilhamento (e-mail, Google, redes sociais, link direto).</li> <li>O usuário seleciona a opção desejada e confirma o compartilhamento.</li> <li>O sistema registra a ação de compartilhamento e exibe uma confirmação de sucesso ao usuário.</li> </ol>
<b>Fluxo Alternativo</b>	<ol style="list-style-type: none"> <li>O usuário tenta compartilhar uma receita, mas há um problema na integração com as redes sociais (Facebook, Google, etc.).</li> <li>O sistema exibe uma mensagem de erro informando que a receita não pôde ser compartilhada nas redes sociais e oferece a opção de copiar um link direto para compartilhar manualmente.</li> <li>O usuário escolhe compartilhar a receita via link direto, e o sistema gera o link.</li> <li>O sistema confirma que o link foi gerado com sucesso e pode ser copiado.</li> </ol>

#### UC07 - Personalização de Perfil

<b>Atores</b>	Usuário
---------------	---------

<b>Definição</b>	O usuário pode personalizar seu perfil, ajustando informações pessoais e preferências dentro da plataforma.
<b>Gatilho</b>	O usuário acessa a opção de editar perfil.
<b>Pré Condições</b>	<ul style="list-style-type: none"> <li>• O usuário deve estar autenticado.</li> </ul>
<b>Pós Condições</b>	<ul style="list-style-type: none"> <li>• O sistema deve salvar as alterações feitas pelo usuário no perfil.</li> <li>• O usuário deve receber uma confirmação de que as alterações foram salvas com sucesso.</li> </ul>
<b>Requisitos Funcionais</b>	RF06
<b>Fluxo principal</b>	<ol style="list-style-type: none"> <li>1. O usuário acessa a opção de editar perfil.</li> <li>2. O usuário altera as informações desejadas (e-mail, nome, foto de perfil, preferências de notificação).</li> <li>3. O usuário confirma as alterações.</li> <li>4. O sistema salva as alterações no perfil do usuário.</li> <li>5. O usuário recebe uma confirmação de que as alterações foram salvas com sucesso.</li> </ol>
<b>Fluxo Alternativo</b>	<ol style="list-style-type: none"> <li>1. O usuário tenta alterar suas informações de perfil, mas alguns dos dados inseridos são inválidos (por exemplo, um endereço de e-mail no formato incorreto).</li> <li>2. O sistema exibe uma mensagem de erro informando sobre os dados incorretos e solicita a correção dos campos.</li> <li>3. O usuário corrige os dados e tenta salvar novamente.</li> <li>4. O sistema valida os dados corrigidos e atualiza as informações de perfil com sucesso.</li> </ol>

### 3.3. Requisitos Funcionais

<b>Ref.</b>	<b>Descrição</b>	<b>Casos de uso</b>
<b>RF01 - Requisito Funcional de</b>	O sistema deve permitir que os usuários se registrem com um e-mail válido, senha ou utilizando	UC01

<b>Usuário</b>	login por redes sociais.	
<b>RF02 - Requisito Funcional de Busca</b>	O sistema deve permitir que os usuários busquem receitas através de filtros como nome da receita, ingredientes, categoria (sobremesas, pratos principais, etc.), tempo de preparo e restrições alimentares..	UC02
<b>RF03 - Requisito Funcional de Visualização</b>	O sistema deve exibir detalhes das receitas, incluindo lista de ingredientes, modo de preparo, tempo de cozimento, porções e uma imagem ilustrativa.	UC05
<b>RF04 - Requisito Funcional de Inclusão de Receita</b>	O sistema deve permitir que os usuários cadastrados adicionem suas próprias receitas, fornecendo informações como título, ingredientes, modo de preparo, tempo de preparo e uma imagem.	UC02
<b>RF05 - Requisito Funcional de Avaliação</b>	O sistema deve permitir que os usuários deixem comentários e avaliem receitas de outros usuários, com uma escala de 1 a 5 estrelas e façam comentários.	UC04
<b>RF06 - Requisito Funcional de Personalização</b>	O sistema deve permitir que o usuário personalize o seu perfil, informando os seus dados e preferências dentro da plataforma.	UC07
<b>RF07 - Requisito Funcional de Login</b>	O sistema deve possibilitar o login de usuários cadastrados, com validação de credenciais e recuperação de senha.	UC01
<b>RF08 - Requisito Funcional de Compartilhamento</b>	O sistema deve permitir que os usuários compartilhem receitas nas redes sociais ou por links diretos.	UC06

### 3.4. Requisitos e restrições não funcionais

Os requisitos e restrições não funcionais são elementos essenciais na especificação do sistema para CookFood. Esses, abrangem aspectos como desempenho, segurança, usabilidade e compatibilidade com diferentes plataformas. Além disso, restrições relacionadas à arquitetura de software, plataformas de hardware e documentação são consideradas. Esses requisitos e restrições são de grande importância para garantir eficiência, confiabilidade e adequação do

sistema pelo que foi concordado entre a CookFood e a equipe de desenvolvimento. Esses requisitos são listados a baixo.

### 3.4.1. Desempenho:

- **RNF01 - Requisito de Desempenho de Usuários:** O sistema deve suportar até 500 usuários simultâneos sem degradação perceptível do desempenho.
- **RNF02 - Requisito de Desempenho de Páginas:** As páginas de receitas devem carregar em no máximo 3 segundos em uma conexão de internet banda larga (10 Mbps).
- **RNF03 - Requisito de Desempenho de Busca:** O tempo de resposta para busca de receitas não deve exceder 2 segundos, considerando um banco de dados com até 10.000 receitas.

### 3.4.2. Segurança:

- **RNF04 - Requisito de Segurança de Senha:** Todas as senhas de usuários devem ser armazenadas de forma criptografada utilizando algoritmos seguros como bcrypt ou scrypt.
- **RNF05 - Requisito de Segurança de LGPD:** O sistema deve estar em conformidade com as normas da LGPD (Lei Geral de Proteção de Dados) e GDPR (Regulamento Geral de Proteção de Dados), garantindo a privacidade e segurança dos dados dos usuários.
- **RNF06 - Requisito de Segurança de Atividade:** As sessões de usuários devem expirar automaticamente após 15 minutos de inatividade, com a possibilidade de reautenticação.

### 3.4.3. Usabilidade:

- **RNF07 - Requisito de Usabilidade de UX:** O sistema deve ser intuitivo e fácil de usar, permitindo que novos usuários aprendam a utilizá-lo sem a necessidade de um tutorial extenso.
- **RNF08 - Requisito de Usabilidade de Responsividade:** A interface deve ser responsiva, funcionando de forma otimizada em dispositivos móveis (resoluções de tela de 360x640 px) e desktops (resoluções de tela de 1920x1080 px).
- **RNF09 - Requisito de Usabilidade de Acessibilidade:** O sistema deve oferecer acessibilidade para pessoas com deficiência, atendendo aos padrões WCAG 2.1 AA (Web Content Accessibility Guidelines).

### 3.4.4. Compatibilidade:

- **RNF10 - Requisito de Compatibilidade de Navegador:** O sistema deve ser compatível com os navegadores mais utilizados, incluindo Google Chrome, Mozilla Firefox, Safari e Microsoft Edge, nas duas últimas versões de cada.
- **RNF11 - Requisito de Compatibilidade de SO:** A aplicação móvel deve estar disponível tanto para Android (versão mínima 8.0) quanto para iOS (versão mínima 12.0).

### **3.4.5. Confiabilidade:**

- **RNF12 - Requisito de Confiabilidade de Inatividade:** O sistema deve ter uma disponibilidade mínima de 99,9%, permitindo apenas 43 minutos de inatividade planejada ou não planejada por mês.
- **RNF13 - Requisito de Confiabilidade de Backup :** O sistema deve realizar backups diáriamente automáticos dos dados, com possibilidade de restauração em até 24 horas no caso de falha.

### **3.4.6. Escalabilidade:**

- **RNF14 - Requisito de Escalabilidade de Usuários:** A arquitetura do sistema deve ser escalável, permitindo que o número de usuários ativos cresça de 100 para 10.000 sem necessidade de reescrever componentes centrais.
- **RNF15 - Requisito de Escalabilidade de Sistema:** O sistema deve suportar a adição de novas categorias e filtros de busca sem impactar no desempenho.

### **3.4.7. Manutenibilidade:**

- **RNF16 - Requisito de Manutenibilidade de Sistema :** O código do sistema deve seguir boas práticas de desenvolvimento, como a separação de responsabilidades (MVC, por exemplo), para facilitar a manutenção e evolução.
- **RNF17 - Requisito de Manutenibilidade de Documentação:** O sistema deve incluir documentação técnica para desenvolvedores, descrevendo a arquitetura, APIs e fluxos principais.

### **3.4.8. Portabilidade:**

- **RNF18 - Requisito de Portabilidade de Nuvem:** O sistema deve ser portável para diferentes ambientes de hospedagem em nuvem, incluindo AWS, Google Cloud e Azure.
- **RNF19 - Requisito de Portabilidade de Dispositivos Móveis:** A aplicação móvel deve permitir ser portabilizada facilmente para outras plataformas de desenvolvimento, como Flutter ou React Native.

## **3.5. Prototipação**

A prototipação de telas é uma etapa fundamental no processo de design de software, onde são criados modelos visuais e interativos que representam a aparência e a funcionalidade de um aplicativo ou site antes de sua implementação. Essa prática permite que designers e desenvolvedores visualizem e testem a estrutura, navegação e interação do produto, facilitando a identificação de problemas de usabilidade e a coleta de feedback de usuários e stakeholders. Os protótipos podem variar em fidelidade, desde esboços simples em papel até maquetes digitais interativas, e são essenciais para alinhar expectativas e garantir que o produto final atenda às necessidades dos usuários.

A prototipação de telas foi realizada utilizando o Figma, uma ferramenta de design colaborativa amplamente utilizada para a criação de interfaces de usuário. O Figma permite que toda a equipe trabalhe simultaneamente no mesmo projeto, facilitando a colaboração em tempo real e a iteração rápida das ideias. Com ele, conseguimos criar protótipos interativos e dinâmicos, simulando com precisão a experiência do usuário final. Além disso, a plataforma oferece funcionalidades como a criação de componentes reutilizáveis e feedback instantâneo, otimizando o processo de design e garantindo maior eficiência no desenvolvimento do projeto.

**Receitas mais acessadas**



**Café da Manhã**



**Almoço**



**Sobremesas**



**Jantar**





Criado por: Bruno Davis, Eduardo Felipe e Filipe Monteiro

**Tiramisú**



**Ingredientes:**

- 450 g de queijo mascarpone
- ½ colher de sopa de rum (opcional)
- ½ xícara de açúcar
- 3 ovos grandes gemas e claras separadas
- 2 xícaras de café expresso frio em uma tigela rasa
- 2 pacotes de biscoito champagne
- 1-2 colheres de sopa de cacau em pó
- 1-2 colheres de sopa de raspas de chocolate

**Modo de Preparo:**

- Em uma tigela de tamanho médio, bata as gemas e acrescente o açúcar e o rum.
- Em outra tigela, bata as claras em neve.
- Use uma colher para colocar o queijo mascarpone nas claras e misture devagar.
- Com cuidado acrescente as gemas.
- Molhe cada biscoito champagne no café – de maneira rápida para molhar o biscoito sem encharcar.
- Faça uma camada com os biscoitos molhados em uma travessa quadrada.
- Coloque uma camada da mistura de mascarpone e por cima outra camada de biscoitos molhados no café.
- Termine com uma camada final de mascarpone.
- Deixe na geladeira por pelo menos 6 horas – de um dia para o outro é o ideal.
- Jogo o cacau em pó por cima e acrescente as raspas de chocolate na hora de servir.



Criado por: Bruno Davis, Eduardo Felipe e Filipe Monteiro

## 4. Análise e Design

Análise e Design de Software são etapas fundamentais no processo de desenvolvimento de sistemas, voltadas para transformar as necessidades de negócio e os requisitos dos usuários em uma solução técnica estruturada e eficiente.

A Análise de Software foca em entender o que o sistema deve fazer. Nesta fase, os requisitos funcionais e não funcionais são coletados e detalhados para garantir que o sistema proposto atenderá às necessidades dos usuários. A análise permite identificar os problemas específicos que o sistema precisa resolver, suas restrições e as expectativas dos stakeholders. O resultado da análise é um conjunto bem definido de requisitos e modelos de entendimento do domínio, que são usados para guiar o design e a implementação.

O Design de Software (ou Projeto de Software) transforma esses requisitos em uma solução arquitetônica detalhada. Nessa fase, são definidas a arquitetura do sistema, as interfaces, os componentes e a forma como esses elementos interagem entre si. O design busca garantir que o sistema seja eficiente, escalável, modular e fácil de manter. Existem dois níveis de design:

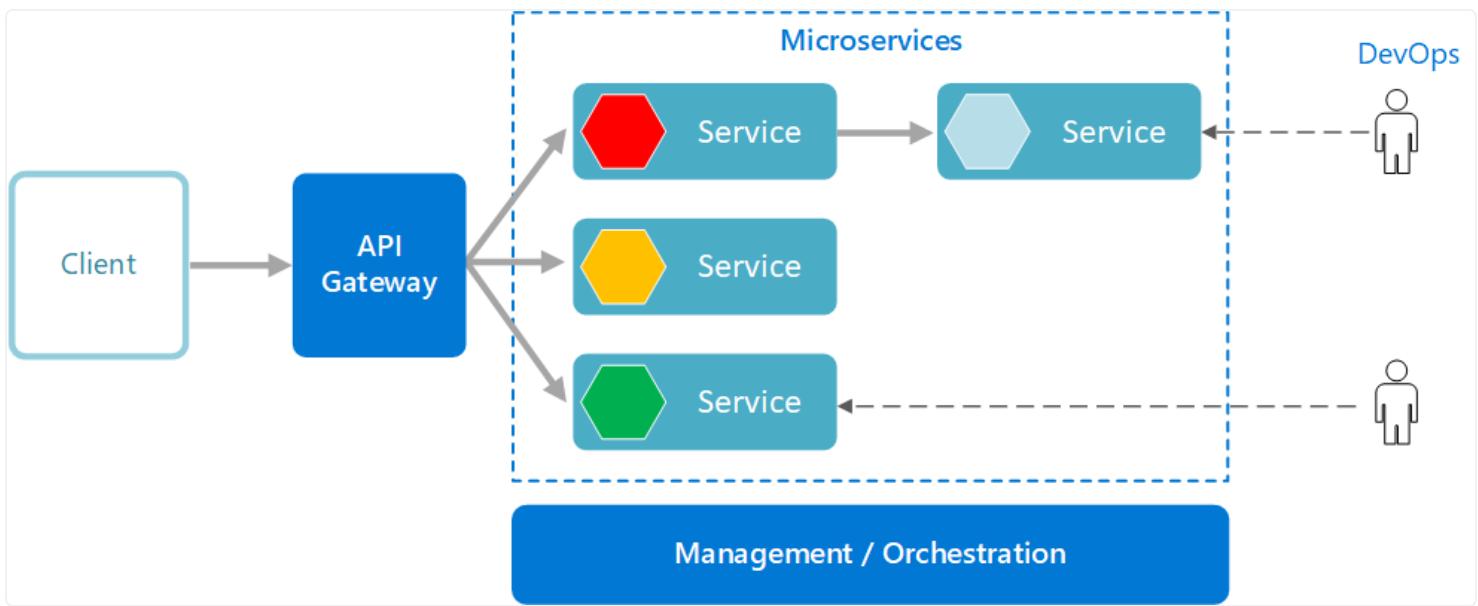
- **Design de Alto Nível (Arquitetura):** Definição da estrutura geral do sistema, como escolha entre arquitetura monolítica ou de microserviços, organização em camadas, etc.
- **Design de Baixo Nível (Detalhado):** Descrição dos componentes específicos e de como eles se comunicarão, incluindo classes, métodos e padrões de projeto a serem usados.

### 4.1. Arquitetura do Sistema

Para o CookFood, iremos utilizar uma Arquitetura de Microserviços. Este modelo distribui a aplicação em múltiplos serviços pequenos e independentes, cada um com uma responsabilidade específica, que se comunicam por meio de APIs. Essa abordagem é ideal para sistemas como o CookFood devido à flexibilidade, escalabilidade e facilidade de manutenção que oferece.

A arquitetura de microserviços permite que cada componente, como o serviço de gerenciamento de receitas, o serviço de autenticação de usuários e o serviço de notificações, seja desenvolvido, implantado e escalado separadamente. Isso facilita a evolução do sistema e garante um desempenho consistente, mesmo com o crescimento do número de usuários ou de novas funcionalidades. Além disso, essa arquitetura favorece a integração de novas tecnologias ou atualizações sem interferir em toda a aplicação.

A divisão e explicação de cada serviço está no item 5.2. Desenvolvimento de Microserviços



## 4.2. Padrões de Projeto

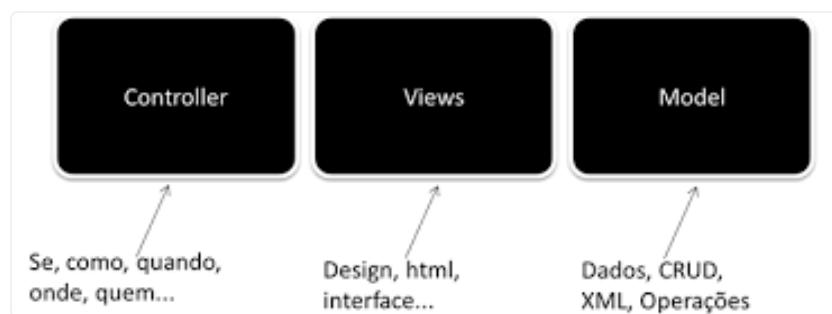
Os Padrões de Design são soluções gerais para problemas recorrentes no desenvolvimento de software. Eles são utilizados para organizar e estruturar o código de forma que ele seja reutilizável, escalável e fácil de manter. No projeto CookFood, adotaremos o padrão MVC (Model-View-Controller) como base para organizar a lógica de apresentação e interação com o usuário, promovendo uma separação clara entre dados, interface e controle. Além disso, essa estrutura será combinada com uma arquitetura de Microserviços, visando a modularidade e a independência entre componentes.

O padrão MVC é amplamente utilizado em desenvolvimento de software, especialmente em sistemas que possuem uma interface de usuário interativa. Ele organiza a aplicação em três componentes principais:

- **Model (Modelo):** Representa os dados e a lógica de negócios da aplicação. No CookFood, o Model incluirá entidades como Receitas, Usuários, Categorias, e Comentários. Ele é responsável por acessar e manipular os dados, garantindo que as operações realizadas sobre eles respeitem as regras do sistema.
- **View (Visão):** Define a interface do usuário, apresentando as informações e interações necessárias. As Views no CookFood serão as páginas e componentes visuais onde os usuários poderão buscar e visualizar receitas, criar novas receitas e interagir com o conteúdo.
- **Controller (Controlador):** Serve como intermediário entre o Model e a View, controlando o fluxo de dados. No CookFood, os Controllers receberão as ações dos usuários (como buscar uma receita ou criar uma nova), interagirão com o Model para manipular os dados e, finalmente, atualizarão a View para exibir os resultados.

Dentro de cada microserviço, o padrão MVC organizará a lógica interna do serviço. Por exemplo, no Serviço de Receitas, o Model será responsável pela manipulação dos dados de receitas, o Controller cuidará das requisições que chegam (como busca e criação de novas receitas), e a View (quando aplicável, como uma API ou página específica) será responsável por retornar a resposta para o usuário ou para outro serviço.

Ao combinar MVC e Microserviços, o CookFood ganha uma estrutura robusta e eficiente para o desenvolvimento de suas funcionalidades. O padrão MVC organiza o código interno de cada microserviço, enquanto a arquitetura de microserviços garante flexibilidade e independência entre os módulos. Dessa forma, conseguimos um sistema que é tanto escalável quanto fácil de evoluir, adaptando-se rapidamente às novas necessidades dos usuários e exigências do mercado.



### 4.3. Modelo do Domínio

O Modelo do Domínio define as principais entidades do sistema e suas relações, oferecendo uma visão clara dos dados e conceitos fundamentais no CookFood. Esse modelo serve como um guia para o desenvolvimento, garantindo que todos compreendam os elementos centrais e como eles interagem.

Para o CookFood, o modelo do domínio inclui entidades como:

- **Usuário:** Representa os usuários da plataforma, com atributos como nome, e-mail, senha (criptografada), foto de perfil e configurações de notificação.
- **Receita:** É o conteúdo principal, incluindo título, descrição, ingredientes, modo de preparo, tempo de preparo, porções e categoria.
- **Categoria:** Classifica as receitas em grupos como sobremesas, pratos principais, etc., facilitando a busca e organização.
- **Comentário:** Reflete a interação dos usuários com as receitas, permitindo que avaliem e comentem, com data e vínculo ao usuário e receita.
- **Favoritos:** Lista de receitas favoritas de cada usuário, facilitando o acesso rápido às suas preferências.

Essas entidades e seus relacionamentos formam a base dos dados e processos do sistema, permitindo que funcionalidades como busca, cadastro e gerenciamento de receitas sejam

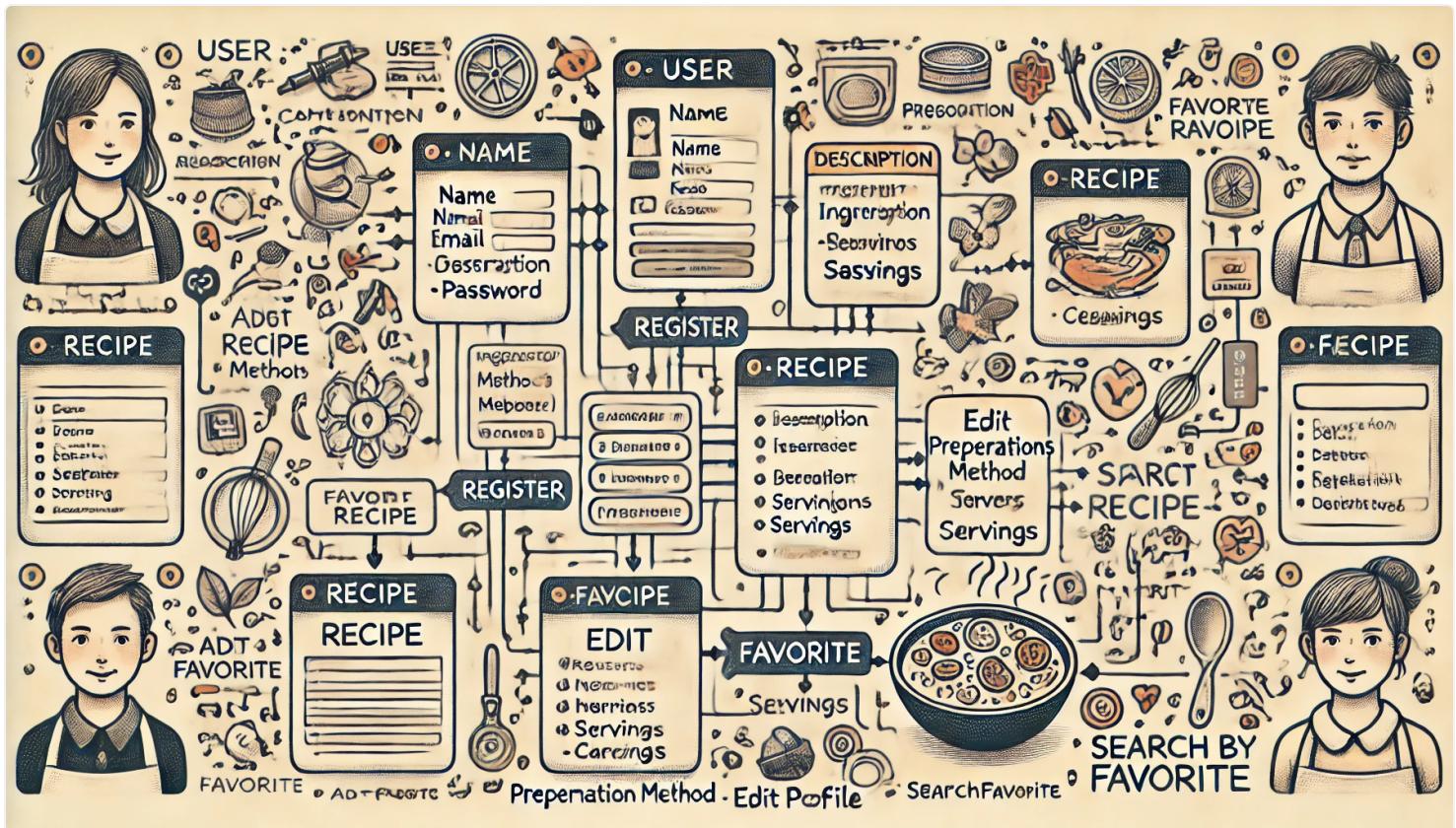
implementadas com clareza e consistência.

#### 4.4. Diagrama de Classes

O Diagrama de Classe é uma das principais ferramentas da UML (Unified Modeling Language) para representar a estrutura de um sistema de software, especialmente em projetos orientados a objetos. Ele descreve as principais classes do sistema, seus atributos, métodos, e as relações entre elas, como associações, dependências e heranças. Em essência, o diagrama de classe oferece uma visão detalhada de como as entidades no sistema estão conectadas e de como elas interagem para realizar as funcionalidades planejadas.

Para o projeto CookFood, o diagrama de classe é fundamental. Ele nos ajuda a visualizar e estruturar os elementos essenciais da aplicação, como Usuário, Receita, Categoria, Comentário, e Favorito, e a definir claramente os atributos e métodos de cada um. Essa organização permite uma base sólida para o desenvolvimento, deixando evidente quais funcionalidades cada classe precisa oferecer, além de facilitar a comunicação entre os membros da equipe, garantindo uma visão compartilhada sobre o sistema.

Além disso, o diagrama de classe apoia diretamente a etapa de implementação. Com uma estrutura clara e bem definida, os desenvolvedores podem se concentrar em cada classe individualmente, enquanto têm uma visão de como ela se relaciona com as demais. No CookFood, o diagrama mostra, por exemplo, que a classe Usuário pode interagir com a classe Favorito para armazenar receitas favoritas, ou com a classe Comentário para gerenciar as interações dos usuários. Assim, o diagrama de classe serve como uma "planta baixa" do sistema, facilitando tanto o desenvolvimento quanto a manutenção futura do projeto.



#### 4.4.1. Entidades

1. Usuário
2. Receita
3. Categoria
4. Comentário
5. Favorito

#### 4.4.2. Atributos e Métodos das Classes

##### 1. Classe Usuário

- **Atributos:**

- `nome`: String
- `email`: String
- `senha`: String
- `fotoPerfil`: String

- **Métodos:**

- `cadastrar()`: Cadastra um novo usuário.

- `login()`: Realiza a autenticação do usuário.
- `editarPerfil()`: Permite ao usuário editar seu perfil.
- `adicionarFavorito(receita: Receita)`: Adiciona uma receita aos favoritos do usuário.

## 2. Classe Receita

- **Atributos:**

- `titulo`: String
- `descricao`: String
- `ingredientes`: List<String>
- `modoPreparo`: String
- `tempoPreparo`: Integer
- `porcoes`: Integer
- `categoria`: Categoria

- **Métodos:**

- `adicionarReceita()`: Adiciona uma nova receita.
- `editarReceita()`: Permite editar uma receita existente.
- `buscarPorCategoria(categoria: Categoria)`: Retorna receitas de uma categoria específica.

## 3. Classe Categoria

- **Atributos:**

- `nome`: String

- **Métodos:**

- `criarCategoria()`: Cria uma nova categoria.
- `editarCategoria()`: Edita uma categoria existente.

## 4. Classe Comentário

- **Atributos:**

- `texto`: String
- `dataComentario`: Date

- `usuario`: Usuário

- `receita`: Receita

- **Métodos:**

- `adicionarComentario()`: Adiciona um comentário a uma receita.

- `removerComentario()`: Remove um comentário da receita.

## 5. Classe Favorito

- **Atributos:**

- `usuario`: Usuário

- `receitasFavoritas` : List<Receita>

- **Métodos:**

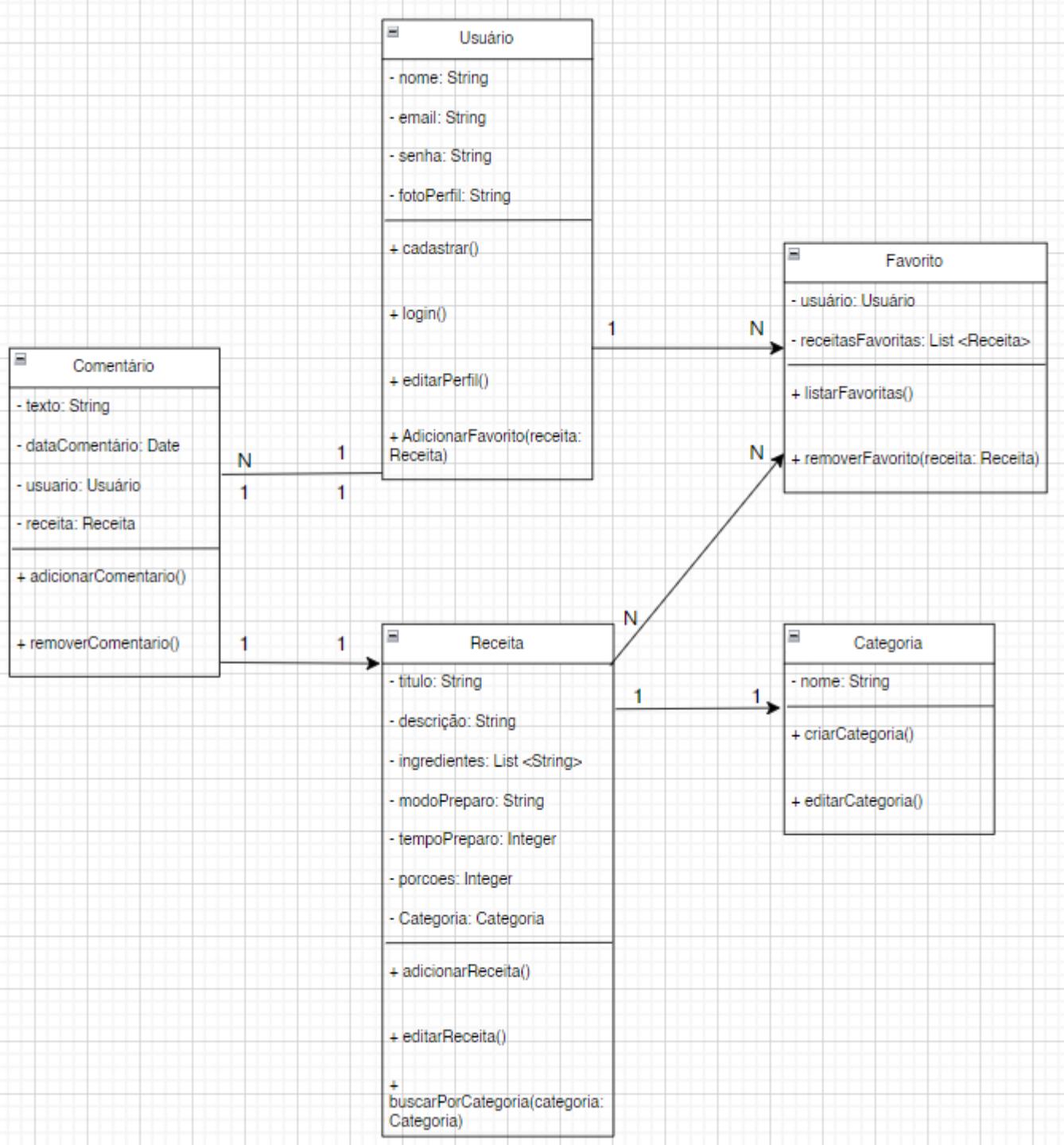
- `listarFavoritos()` : Lista todas as receitas favoritas do usuário.

- `removerFavorito(receita: Receita)` : Remove uma receita dos favoritos do usuário.

### 4.4.3. Relacionamentos

- Usuário tem uma associação com Comentário (um usuário pode fazer vários comentários).
- Receita está associada a Categoria (uma receita pertence a uma categoria específica).
- Comentário associa Usuário e Receita (um comentário pertence a um usuário e uma receita).
- Favorito conecta Usuário e Receita (um usuário pode ter várias receitas em sua lista de favoritos).

### 4.4.4. Diagrama



## 4.5. Diagrama de Sequência

O diagrama de sequência é uma ferramenta de modelagem usada para representar a interação entre objetos e componentes de um sistema ao longo do tempo. Esse tipo de diagrama descreve visualmente a ordem de execução das operações ou métodos, indicando o fluxo de mensagens entre as entidades envolvidas em um caso de uso específico.

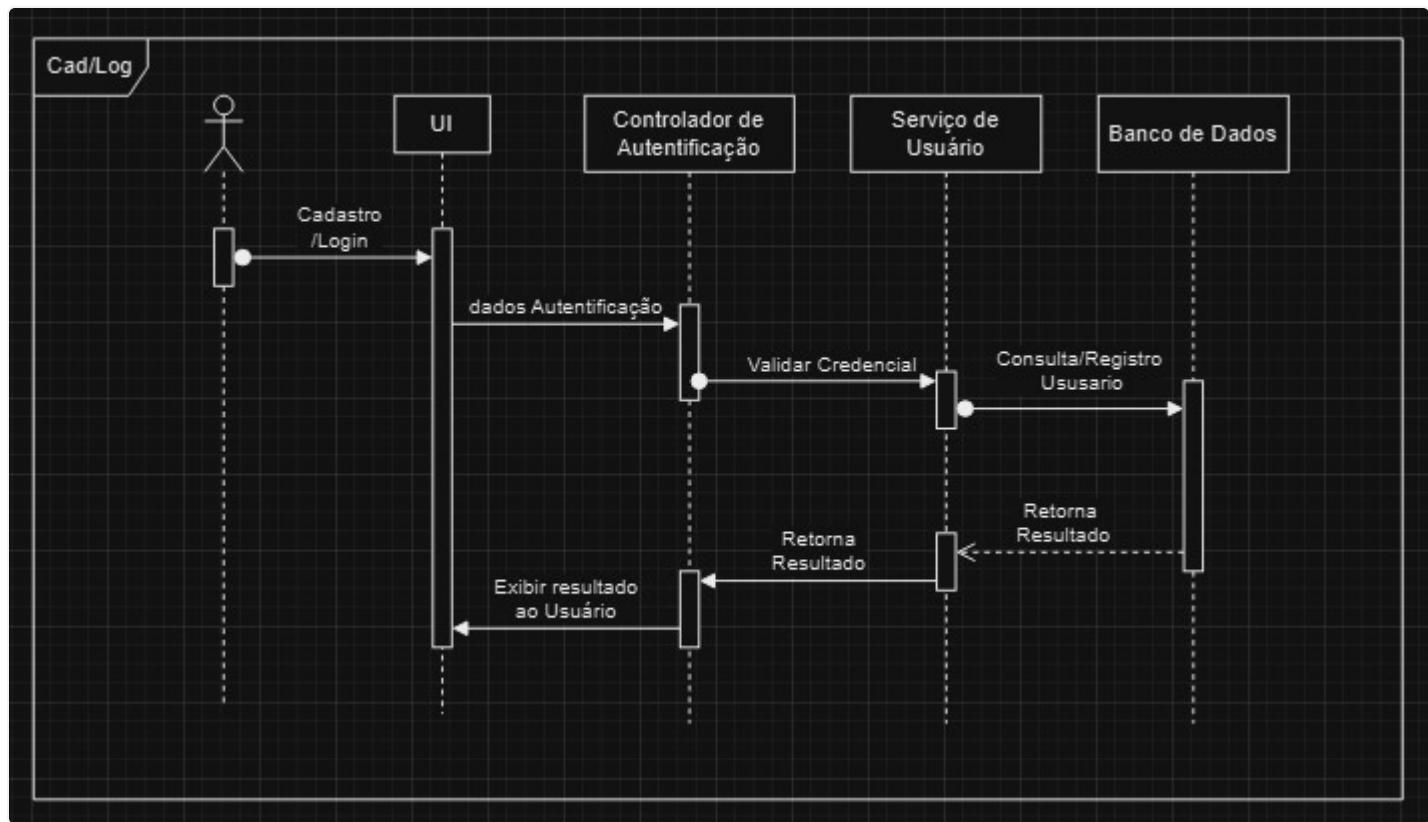
No projeto CookFood, o diagrama de sequência é utilizado para ilustrar o comportamento dinâmico do sistema em cenários essenciais, como o fluxo de busca, visualização, adição de receitas, e personalização de perfil. Em um diagrama de sequência:

1. **Objetos/Componentes** envolvidos são dispostos horizontalmente na parte superior.
2. **Linhas de vida** estendem-se verticalmente a partir de cada objeto, representando sua existência durante o fluxo.
3. **Mensagens** entre os objetos são indicadas por setas horizontais, mostrando as chamadas de métodos e a troca de informações.

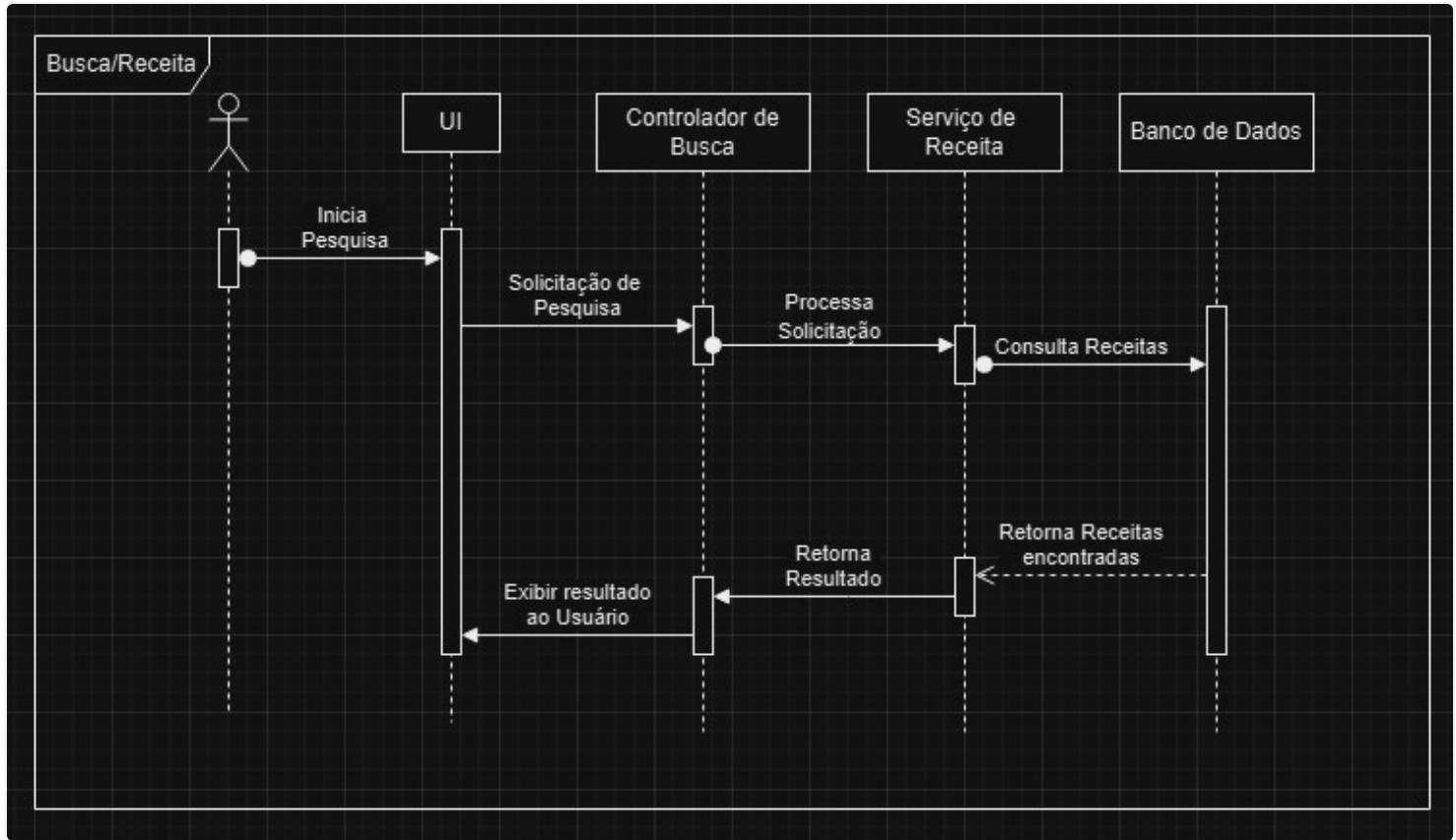
O diagrama ajuda a equipe de desenvolvimento a compreender a sequência exata de operações, evitando ambiguidades e facilitando a implementação dos requisitos do sistema.

A seguir colocarei os principais casos de diagrama de sequência. Esses casos descrevem interações entre o usuário e o sistema, especialmente nos fluxos essenciais para garantir que a plataforma funcione como esperado.

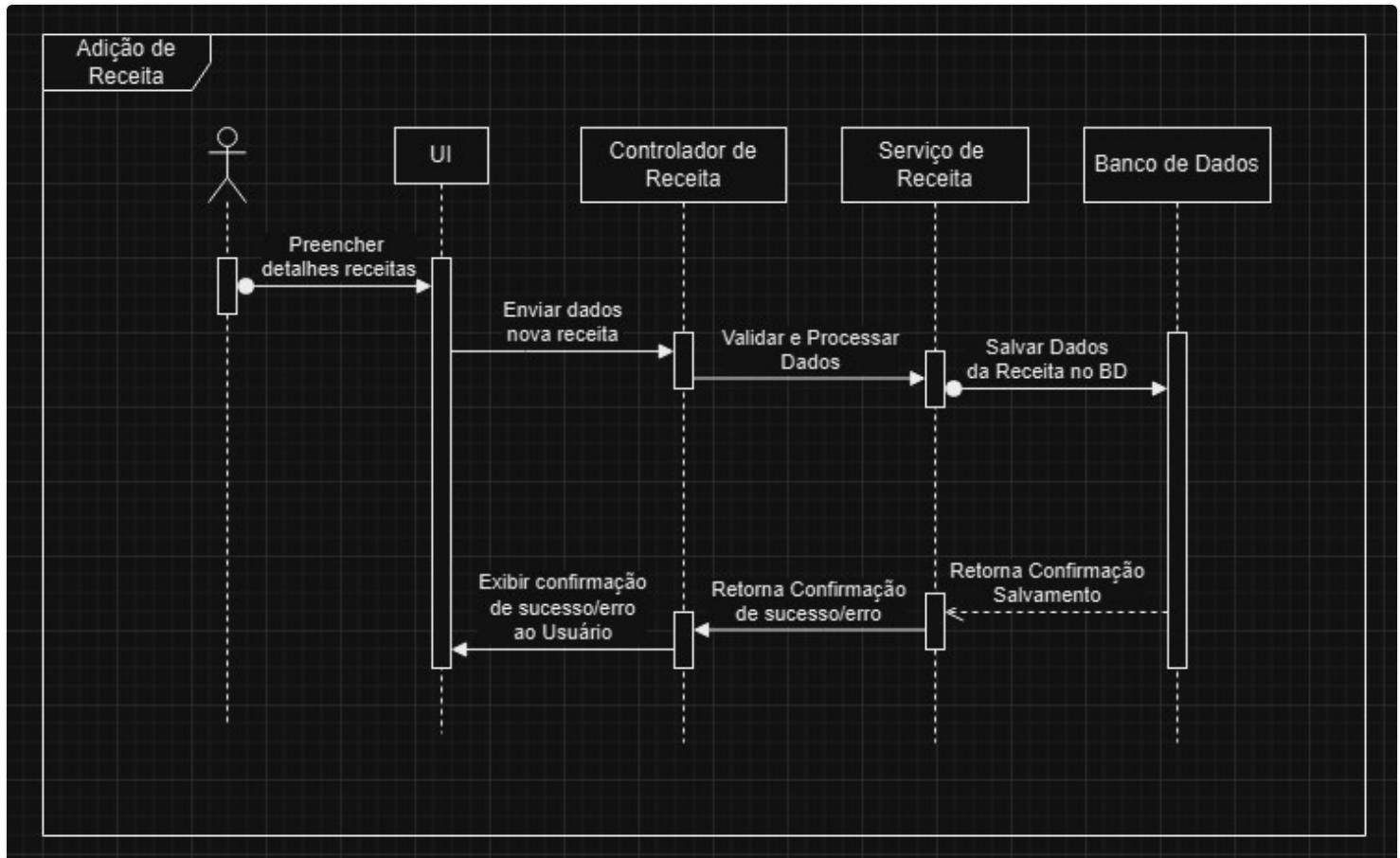
#### 4.5.1. Cadastro e Autenticação de Usuário



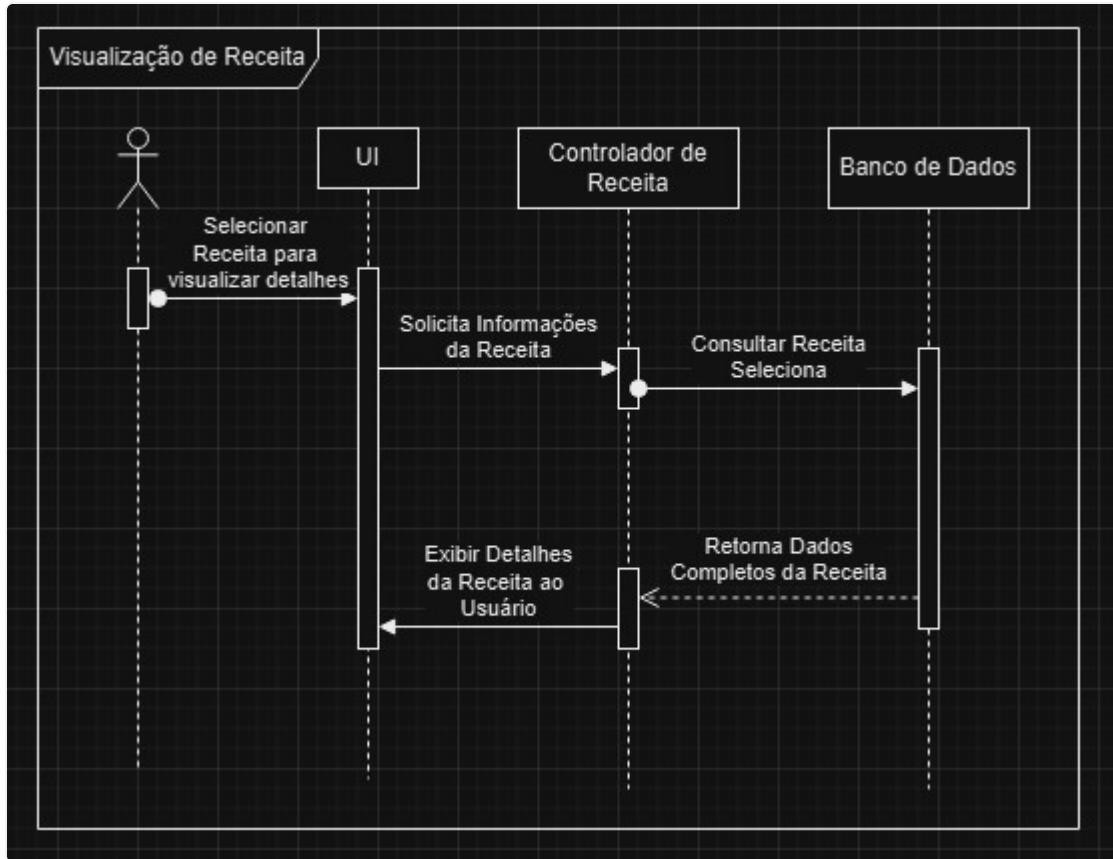
#### 4.5.2. Busca de Receita



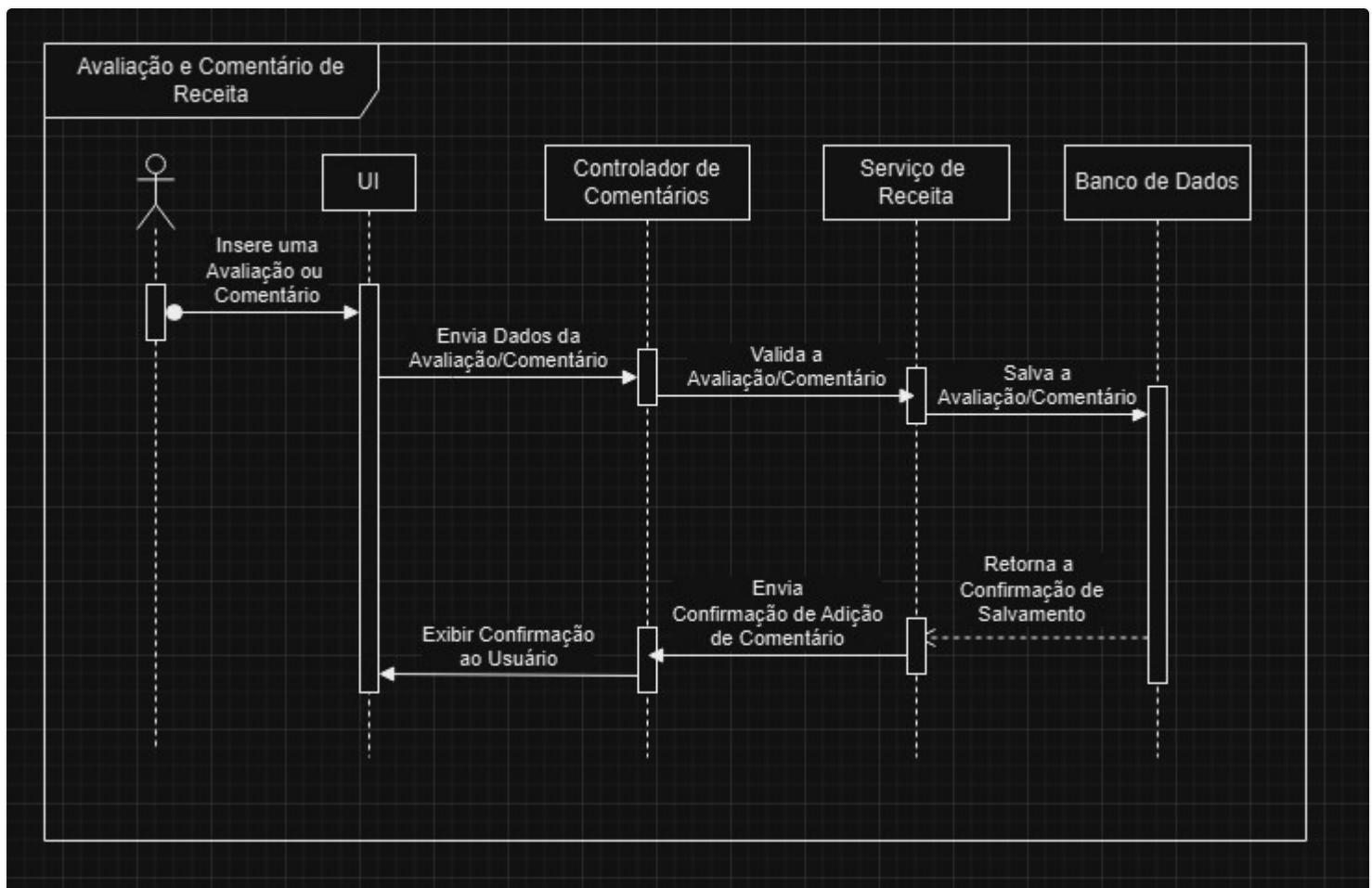
#### 4.5.3. Adição de receita



#### 4.5.4. Visualização de Receita



#### 4.5.5. Avaliação e Comentário de Receita



## 4.6. Modelo de Dados

### 4.6.1. Modelo Lógico da Base de Dados

O modelo lógico será baseado nos principais requisitos do sistema, estruturado para suportar a funcionalidade de gestão de receitas. As entidades principais incluem:

- **Usuário:** Armazena informações como ID, nome, e-mail, senha (criptografada) e preferências.
- **Receita:** Inclui título, descrição, ingredientes, modo de preparo, tempo de preparo, categoria e avaliações.
- **Avaliação e Comentário:** Relaciona usuários e receitas, registrando avaliações (1 a 5 estrelas) e comentários.
- **Categoria e Tags:** Define filtros para busca de receitas.

As relações entre essas entidades serão normalizadas para evitar redundância e assegurar a integridade dos dados.

### 4.6.2. Criação Física do Modelo de Dados

O banco de dados será implementado em um sistema de gerenciamento relacional, como PostgreSQL ou MySQL. Seguem os principais comandos:

#### 4.6.2.1 Tabela Usuários:

```
CREATE TABLE usuarios (
    id SERIAL PRIMARY KEY,
    nome VARCHAR(50),
    email VARCHAR(50) UNIQUE,
    senha VARCHAR(10),
    preferencias JSON );
```

#### 4.6.2.2 Tabela de Receitas:

```
CREATE TABLE receitas (
    id SERIAL PRIMARY KEY,
    titulo VARCHAR(200),
    descricao TEXT,
    ingredientes JSON,
    modo_preparo TEXT,
    tempo_preparo INT,
    categoria_id INT REFERENCES categorias(id),
```

```
usuario_id INT REFERENCES usuarios(id);
```

#### 4.6.2.3 Tabela de Avaliação:

```
CREATE TABLE avaliacoes (
    id SERIAL PRIMARY KEY,
    receita_id INT REFERENCES receitas(id),
    usuario_id INT REFERENCES usuarios(id),
    estrelas INT CHECK (estrelas BETWEEN 1 AND 5),
    comentario TEXT );
```

#### 4.6.3. Dicionário de Dados

Campo	Descrição	Tipo de Dado	Tabela
<i>id</i>	Identificador único	SERIAL	Usuários, Receitas, Avaliações
<i>nome</i>	Nome completo do usuário	VARCHAR(50)	Usuários
<i>titulo</i>	Título da receita	VARCHAR(50)	Receitas
<i>estrelas</i>	Avaliação de 1 a 5	INT	Avaliações
<i>ingredientes</i>	Lista de ingredientes	JSON	Receitas

#### 4.7. Ambiente de Desenvolvimento

O ambiente de desenvolvimento será configurado para suportar agilidade e colaboração:

- **Linguagem de programação:** Python (Django) para backend, e ReactJS para frontend.
- **Banco de Dados:** PostgreSQL para armazenamento relacional.
- **Ferramentas de Controle de Versão:** Git com GitHub para versionamento e integração contínua.
- **Ferramentas de CI/CD:** Jenkins e Docker para automação de build e deploy.
- **IDE:** Visual Studio Code, com extensões para linting e debug.

#### 4.8. Sistemas e Componentes Externos Utilizados

##### 4.8.1. APIs Externas:

- Google OAuth para autenticação ou Firebase.
- Cloudinary para armazenamento de imagens de receitas.

#### **4.8.2. Bibliotecas e Frameworks:**

- Bootstrap para estilização responsiva.
- Axios para chamadas de API no frontend.

#### **4.8.3. Plataforma de Hospedagem:**

- AWS para backend e banco de dados.
- Netlify ou Vercel para deploy do frontend.

### **5. Implementação**

#### **5.1. Configuração Inicial do Ambiente**

- **Gerenciamento de Código:**
  - Reppositórios Git separados para cada microserviço.
  - Utilização de GitHub Actions para pipelines de CI/CD.
- **Containerização:**
  - Uso de Docker para isolar cada microserviço em contêineres.
  - Docker Compose para orquestração em ambiente local.
- **Infraestrutura de Mensageria:**
  - RabbitMQ ou Kafka para comunicação assíncrona entre microserviços.
- **Monitoramento e Logs:**
  - Prometheus para monitoramento de métricas.
  - Grafana para visualização de dashboards.
  - ELK Stack (Elasticsearch, Logstash, Kibana) para centralização e análise de logs.

#### **5.2. Desenvolvimento de Microserviços**

Cada microserviço será desenvolvido com foco na independência e responsabilidade única.

- **Microserviço de Autenticação:**
  - Gerenciar o registro e login de usuários, utilizando JWT para autenticação segura.

- Comunicação síncrona com o gateway API e comunicação assíncrona com outros microserviços.
- **Microserviço de Receitas:**
  - CRUD de receitas.
  - Integração com o microserviço de buscas para indexação de novos dados.
- **Microserviço de Avaliação e Comentários:**
  - Gerenciar avaliações e comentários relacionados às receitas.
  - Comunicação com o microserviço de receitas para exibição de avaliações médias.
- **Microserviço de Busca:**
  - Implementação com ElasticSearch para buscas rápidas e filtragem eficiente.
  - Atualização assíncrona por mensagens recebidas de outros microserviços.

### 5.3. Orquestração e API Gateway

- Implementação de um API Gateway (por exemplo, Kong ou NGINX) para:
  - Gerenciar autenticação e autorização.
  - Direcionar solicitações para os microserviços corretos.
- Definição de contratos de API utilizando OpenAPI/Swagger para padronizar endpoints e facilitar o consumo.

## 6. Teste

### 6.1. Estratégias de Teste

A abordagem de testes será baseada em uma pirâmide de testes, com os seguintes níveis:

1. **Testes Unitários:**
  - Cobertura mínima de 80% do código.
  - Utilização de ferramentas como Pytest (Python) e Jest (JavaScript).
2. **Testes de Integração:**
  - Testar a interação entre microserviços utilizando ambientes simulados (Mock APIs).
  - Ferramentas: Postman, Pact (Consumer-driven contract testing).
3. **Testes Funcionais:**

- Simular fluxos completos do usuário, garantindo que os requisitos funcionais sejam atendidos.

#### 4. Testes de Performance:

- JMeter ou Gatling para identificar gargalos e verificar o desempenho com carga escalável.

#### 5. Testes de Resiliência e Recuperação:

- Injectar falhas simuladas (Chaos Engineering) para testar a robustez do sistema.
- Ferramentas: Chaos Monkey.

### 6.2. Pipeline de Testes Automatizados

- Configuração de pipelines CI/CD para execução automática dos testes. Etapas principais:
  - **Build:** Compilação do código e geração de artefatos.
  - **Testes Automatizados:** Executar testes unitários, de integração e funcionais.
  - **Análise de Qualidade:** Uso de SonarQube para análise estática de código.

### 6.3. Teste de Comunicação entre Microserviços

- Validação de chamadas síncronas e assíncronas, garantindo que mensagens sejam entregues corretamente entre os serviços. Ferramentas:
  - **WireMock** para mockar APIs externas.
  - **RabbitMQ Management Plugin** para inspecionar filas e mensagens.

## 7. Verificação e Validação de Artefatos

A verificação e validação dos artefatos do projeto CookFood são processos essenciais para garantir que o produto desenvolvido atende às especificações e requisitos definidos. Eles garantem a qualidade e conformidade do sistema em relação às expectativas dos stakeholders e às normas de desenvolvimento de software.

### 7.1. Verificação dos Artefatos

#### 7.1.1. Verificação da Documentação

##### Revisão dos Requisitos Funcionais e Não Funcionais:

- **Verificado:** O documento apresenta requisitos funcionais, como registro de usuários (RF01), busca de receitas (RF02), avaliação de receitas (RF05), e outros. Eles estão detalhados e

vinculados aos casos de uso.

### Casos de Uso e Regras de Negócio:

- **Verificado:** Casos de uso como "Cadastro e Autenticação de Usuário" (UC01) e "Busca de Receitas" (UC02) foram descritos com fluxo principal e alternativo.

### 7.1.2. Verificação de Modelagem e Diagramas

- **Verificado:** Os diagramas de casos de uso e de interação foram apresentados, cobrindo os principais fluxos do sistema.
- **Verificado:** Entidades como **Receita**, **Usuário** e **Categoria** estão descritas no modelo de domínio.

### 7.1.3. Verificação dos Protótipos

#### Prototipação de Interfaces:

- **Verificado:** As telas apresentadas no Figma refletem os principais fluxos de navegação, como cadastro de receitas e interação com a comunidade.

### 7.1.4. Revisão de Código e Componentes

#### Revisão de Código Fonte:

- **Verificado:** O código deve ser revisado para garantir que segue o padrão arquitetural definido (microserviços e MVC). O uso de boas práticas, como modularidade, validação de entradas e criptografia, deve ser garantido.

## 7.2. Validação dos Artefatos

### 7.2.1. Validação dos Requisitos com Stakeholders

- **Verificado:** Apresentar os requisitos documentados, casos de uso e protótipos para os stakeholders, incluindo desenvolvedores e potenciais usuários. Coletar feedback sobre a adequação do sistema às necessidades e ajustar os requisitos conforme necessário.

### 7.2.2. Validação Funcional

#### Testes Funcionais:

- **Verificado:** Conduzir testes para validar as principais funcionalidades, como registro de usuário, busca de receitas e avaliações. Verificar se as operações ocorrem conforme o esperado e dentro dos parâmetros definidos.

#### Testes de Aceitação:

- **Verificado:** Realizar testes com usuários reais para validar se as funcionalidades são úteis, intuitivas e atendem às expectativas dos usuários finais.

### 7.2.3. Validação Não Funcional

#### Testes de Desempenho:

- **Verificado:** Testar o sistema com 500 usuários simultâneos para verificar a degradação do desempenho. Os tempos de carregamento das páginas e de busca devem estar dentro dos limites especificados.

#### Testes de Segurança:

- **Verificado:** Validar a conformidade com a LGPD, garantindo que os dados dos usuários são criptografados e estão protegidos.

#### Testes de Usabilidade:

- **Verificado:** Realizar testes com diferentes dispositivos e resoluções de tela para garantir que a interface seja responsiva e acessível.

### 7.3. Relatório e Ajustes

#### Relatório de Resultados:

- Documentar todas as verificações e validações realizadas, descrevendo pontos fortes e não conformidades identificadas.

#### Implementação dos Ajustes:

- Corrigir as não conformidades observadas durante a validação, realizar novos testes para validar as correções e garantir que os artefatos estão prontos para a próxima fase ou entrega.

## 8. Implantação

Para um exemplo de implementação vamos ilustrar os caso de avaliação de receitas no sistema. O microserviço de Avaliação e Comentários é responsável por gerenciar as interações dos usuários com as receitas, permitindo que eles avaliem pratos com estrelas (de 1 a 5) e compartilhem seus comentários sobre a experiência. Além disso, esse microserviço calcula a média das avaliações de cada receita e interage com o microserviço de receitas para exibir essas informações aos usuários de forma integrada.

A implementação segue a arquitetura MVC (Model-View-Controller), promovendo a separação de responsabilidades e facilitando a escalabilidade e manutenção do sistema. Cada camada do microserviço desempenha um papel fundamental: o Model define as estruturas de dados e interage com o banco de dados, o Controller centraliza a lógica de negócios e validações, enquanto o View processa as requisições e retorna respostas no formato JSON por meio de uma API REST.

Por meio desse microserviço, o sistema CookFood busca oferecer uma experiência rica e colaborativa aos usuários, permitindo que compartilhem suas opiniões e ajudem outros a encontrar as melhores receitas. A seguir, será apresentada a estrutura detalhada do microserviço, exemplificando como as camadas trabalham em conjunto para atingir esses objetivos.

## 8.1. Model (Modelo)

O Model é responsável por gerenciar os dados da aplicação, interagir com o banco de dados e validar informações. Em Django, ele é implementado utilizando as classes de modelos.

```
# models.py
from django.db import models

class Avaliacao(models.Model):
    receita_id = models.IntegerField() # ID da receita (referência ao microserviço de receitas)
    usuario_id = models.IntegerField() # ID do usuário (referência à tabela de usuários)
    estrelas = models.IntegerField(choices=[(i, i) for i in range(1, 6)]) # Estrelas (1 a 5)
    comentario = models.TextField(blank=True, null=True) # Comentário opcional
    data_criacao = models.DateTimeField(auto_now_add=True) # Data da avaliação

    def __str__(self):
        return f"Avaliação {self.estrelas} estrelas para Receita {self.receita_id}"
```

## 8.2. View (Visualização)

O View é responsável por processar as requisições HTTP, interagir com o modelo e retornar uma resposta ao cliente. No Django Rest Framework (DRF), isso é implementado por [ViewSets](#) ou [APIView](#).

```
# views.py
from rest_framework import viewsets, status
from rest_framework.response import Response
from .models import Avaliacao
from .serializers import AvaliacaoSerializer
import requests

class AvaliacaoViewSet(viewsets.ViewSet):
    # Listar avaliações por receita
    def list(self, request, receita_id=None):
        avaliacoes = Avaliacao.objects.filter(receita_id=receita_id)
        serializer = AvaliacaoSerializer(avaliacoes, many=True)
        return Response(serializer.data)

    # Criar nova avaliação
    def create(self, request):
        serializer = AvaliacaoSerializer(data=request.data)
        if serializer.is_valid():
            serializer.save()
            return Response(serializer.data, status=status.HTTP_201_CREATED)
        return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)

    # Calcular média de avaliações de uma receita
```

```

def media_avaliacoes(self, request, receita_id=None):
    avaliacoes = Avaliacao.objects.filter(receita_id=receita_id)
    media = avaliacoes.aggregate(models.Avg('estrelas'))['estrelas__avg']
    return Response({"receita_id": receita_id, "media_estrelas": media})

# Obter informações da receita (via comunicação com outro microserviço)
def detalhes_receita(self, receita_id):
    try:
        response = requests.get(f"http://microservico-receitas/api/receitas/{receita_id}/")
        if response.status_code == 200:
            return response.json()
    except requests.exceptions.RequestException as e:
        print(f"Erro ao comunicar com o microserviço de receitas: {e}")
    return None

```

Chamar o controller:

```

from .controllers import AvaliacaoController

class AvaliacaoViewSet(viewsets.ViewSet):
    def media_avaliacoes(self, request, receita_id=None):
        media = AvaliacaoController.calcular_media(receita_id)
        return Response({"receita_id": receita_id, "media_estrelas": media})

```

## 8.3. Controller

No Django, o Controller está embutido nas views, mas a separação lógica entre o fluxo de controle e as operações nos modelos é recomendada. Aqui o **Controller** define as regras de negócio, como validações ou cálculos.

Por exemplo, no método de cálculo de média de avaliações, o **Controller** centraliza essa lógica:

```

# controllers.py (arquivo opcional para separar a lógica)
from django.db.models import Avg
from .models import Avaliacao

class AvaliacaoController:
    @staticmethod
    def calcular_media(receita_id):
        avaliacoes = Avaliacao.objects.filter(receita_id=receita_id)
        media = avaliacoes.aggregate(Avg('estrelas'))['estrelas__avg']
        return media

```

## 8.4. Serializers

Os serializers convertem os objetos do banco de dados (Modelos) para JSON e vice-versa.

```

# serializers.py
from rest_framework import serializers
from .models import Avaliacao

class AvaliacaoSerializer(serializers.ModelSerializer):
    class Meta:
        model = Avaliacao
        fields = ['id', 'receita_id', 'usuario_id', 'estrelas', 'comentario', 'data_criacao']

```

## 8.5. Rotas (URLs)

Configura as rotas para acessar as funções do microserviço:

```

# urls.py
from django.urls import path
from .views import AvaliacaoViewSet

avaliacao_list = AvaliacaoViewSet.as_view({'get': 'list', 'post': 'create'})
avaliacao_media = AvaliacaoViewSet.as_view({'get': 'media_avaliacoes'})

urlpatterns = [
    path('avaliacoes/', avaliacao_list, name='avaliacoes'),
    path('avaliacoes/<int:receita_id>/media/', avaliacao_media, name='media_avaliacoes'),
]

```

## 8.6. Tabela Organizacional Padrão MVC para Avaliação de Receita

Camada	Arquivo	Responsabilidade
Model	<a href="#">model.py</a>	Define o modelo de dados Avaliação e interage com o banco de dados.
View	<a href="#">views.py</a>	Define a API, processa requisições e retorna respostas em JSON.
Controller	<a href="#">controllers.py</a>	Regras de negócio, como cálculo de médias de avaliações.
Serializers	<a href="#">serializers.py</a>	Converte dados do modelo para JSON e valida entradas de dados da API.
URLs	<a href="#">urls.py</a>	Define rotas para endpoints do microserviço.

## 9. Manual do Usuário

### 9.1. Introdução

Este manual destina-se a orientar os usuários na utilização da plataforma CookFood, destacando as funcionalidades principais e fornecendo instruções claras para um uso eficiente.

### 9.2. Pré-requisitos

- **Acesso à Internet:** A plataforma requer conexão estável.
- **Dispositivo Compatível:** Disponível para navegadores modernos (Google Chrome, Firefox, Safari, Edge) e dispositivos móveis com Android (8.0 ou superior) ou iOS (12.0 ou superior).

### 9.3. Primeiros Passos

#### 1. Cadastro:

- Acesse a página inicial e clique em "Criar Conta".
- Escolha uma das opções: e-mail, redes sociais (Google ou Facebook) ou acesso como convidado.
- Preencha os dados solicitados e confirme.

#### 2. Login:

- Insira seu e-mail e senha cadastrados ou utilize o login via redes sociais.

#### 3. Configuração de Perfil:

- Clique em "Meu Perfil" para personalizar informações e preferências.

### 9.4. Funcionalidades Principais

#### • Buscar Receitas:

- Use a barra de busca para encontrar receitas por nome, ingredientes ou categorias.
- Utilize filtros para refinar sua pesquisa (tempo de preparo, restrições alimentares, etc.).

#### • Adicionar Receitas:

- Clique em "Adicionar Receita" e preencha os campos obrigatórios: título, ingredientes, modo de preparo e imagem.

#### • Avaliar e Comentar Receitas:

- Acesse uma receita, insira uma avaliação de 1 a 5 estrelas e deixe um comentário.

#### • Compartilhar Receitas:

- Clique em "Compartilhar" e escolha entre redes sociais ou link direto.

#### • Receitas Favoritas:

- Marque receitas como favoritas para acesso rápido em sua área de perfil.

## 9.5. Solução de Problemas

- **Esqueci minha senha:**

- Acesse a opção "Esqueci minha senha" na tela de login e siga as instruções.

- **Problemas de carregamento:**

- Verifique sua conexão com a Internet.
  - Se persistir, entre em contato pelo suporte no e-mail: [suporte@cookfood.com](mailto:suporte@cookfood.com).

## 10. Conclusão

O projeto CookFood representa um avanço significativo na forma como receitas são acessadas e compartilhadas, promovendo uma experiência digital intuitiva e rica para os amantes da culinária.

A adoção de uma arquitetura de microserviços assegura escalabilidade e manutenção eficiente, enquanto a implementação de recursos como busca avançada e personalização de perfil aprimora a usabilidade.

O desenvolvimento orientado por práticas ágeis e uso de tecnologias modernas garante que o sistema seja robusto, seguro e de fácil adaptação às demandas futuras.

Com este documento, buscamos consolidar uma base sólida para a continuidade do projeto, assegurando que as equipes de desenvolvimento e operação tenham os recursos necessários para entregar uma solução de qualidade que atenda às expectativas de seus usuários e stakeholders.

## 11. Bibliografia

Essas normas abrangem aspectos importantes do desenvolvimento de software, desde a definição de processos e ciclos de vida até a avaliação da qualidade do produto e do processo.

- **ABNT NBR ISO/IEC 12207** - Engenharia de Software: Processos de Ciclo de Vida de Software.

Esta norma pode ser aplicada na seção que descreve os processos de desenvolvimento, testes e implantação do sistema CookFood, garantindo que todas as etapas do ciclo de vida do software sejam seguidas conforme os padrões internacionais.

- **ABNT NBR ISO/IEC 25010** - Engenharia de Software: Modelos de Qualidade de Produto.

Pode ser aplicada na definição dos requisitos de qualidade do software, assegurando que o produto final atenda aos critérios de qualidade estabelecidos.

- **ABNT NBR ISO/IEC 15288** - Engenharia de Sistemas: Processos de Ciclo de Vida de Sistemas.

Pode ser aplicada na descrição dos processos de desenvolvimento do sistema, garantindo uma abordagem sistemática e disciplinada ao desenvolvimento do software.

- **ABNT NBR ISO/IEC 15504** - Avaliação de Processo de Software.

Pode ser aplicada na avaliação dos processos de desenvolvimento do software, ajudando a identificar áreas de melhoria e assegurar a qualidade do processo de desenvolvimento.