**DOCUMENTAÇÃO**

**RF01 – ReceitinhasFederais**

**OBJETIVOS:**

Armazenamento de receitas com Título;

Armazenamento de receitas com Ingredientes;

Armazenamento de receitas com Modo de preparo;

Armazenamento de receitas com Tempo de preparo;

Armazenamento de receitas com Autor;

Armazenamento de receitas com Categoria;

Armazenamento de receitas com O grau de dificuldade da receita;

Armazenamento de receitas com a Quantidade de pratos produzida;

**Tecnologias Utilizadas:**

* Linguagem de Programação C#;
* Word;
* Picsart;

**Diagrama de casos de uso:**Diagrama

Descrição gerada automaticamente

Usuário - é o ator que interage com o sistema de receitas. O usuário pode buscar, adicionar e remover.

Casos de Uso:

Consultar receitas - permite ao usuário pesquisar receitas no sistema. Esse caso de uso é importante porque permite ao usuário encontrar receitas que atendam às suas necessidades.

Cadastrar receitas - permite ao usuário criar uma nova receita no sistema. Esse caso de uso é importante porque permite ao usuário compartilhar receitas com outros usuários.

Remover receitas - permite ao usuário excluir uma receita do sistema. Esse caso de uso é importante porque permite ao usuário remover receitas desnecessárias ou impróprias.

Cada caso de uso representa uma funcionalidade do sistema que pode ser realizada pelo usuário. Cada caso de uso tem seus próprios campos específicos, que são definidos nas etapas de análise de requisitos e levantamento de informações. Esses campos podem incluir informações como nome da receita, categoria da receita, ingredientes necessários, tempo de preparo e número de porções.

A modelagem de casos de uso é importante porque ajuda a identificar os requisitos de sistema. Os requisitos de sistema são as características que o sistema deve ter para atender às necessidades dos usuários. A modelagem de casos de uso permite que os requisitos sejam identificados e organizados em uma estrutura clara e compreensível.

Em resumo, o diagrama de caso de uso é uma ferramenta importante na modelagem de sistemas de software. Ele ajuda a identificar os requisitos de sistema, definir as funcionalidades do sistema e mostrar como os usuários interagem com essas funcionalidades. Com o diagrama de caso de uso, os desenvolvedores podem criar um sistema que atenda às necessidades dos usuários de maneira eficiente e eficaz.

**Diagrama de classes:**

**Tabela

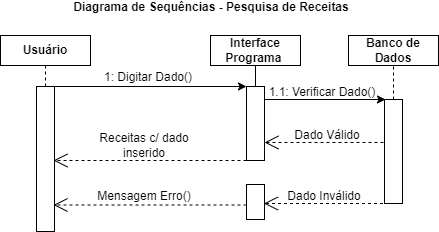
Descrição gerada automaticamente**

**Explicação descritiva do Diagrama de Classes Receita**

* Titulo: Campo do tipo string que representa o título da receita.
* Ingredientes: Campo do **tipo string** que representa os ingredientes necessários para preparar a receita.
* ModoPreparo: Campo do **tipo string** que representa o modo de preparo da receita.
* GrauDificuldade: Campo do **tipo string** que representa o grau de dificuldade da receita, como fácil, médio ou difícil.
* Autor: Campo do **tipo string** que representa o nome do autor da receita.
* TempoPreparo: Campo do **tipo string** que representa o tempo necessário para preparar a receita, por exemplo, "30 minutos" ou "1 hora".
* Categoria: Campo do **tipo string** que representa a categoria da receita, como sobremesa, prato principal ou entrada.
* QntdPratos: Campo do **tipo int** que representa a quantidade de pratos que a receita pode fazer.

Esses campos definem os atributos de uma classe **Receita**, que pode ser usada para representar receitas culinárias no sistema. Cada instância da classe Receita terá seus próprios valores atribuídos a esses campos, permitindo que várias receitas diferentes sejam armazenadas e gerenciadas pelo sistema.

**Diagrama de sequência da pesquisa de receitas:**



O usuário acessa a tela inicial do programa e visualiza a lista de receitas disponíveis.

Para pesquisar uma receita específica, o usuário digita um valor na **textbox** presente na interface.

Quando o usuário digita o valor na **textbox**, o programa verifica se o valor é válido e se há alguma receita com os valores inseridos na **textbox** correspondente no banco de dados.

Caso o valor seja válido, o programa realiza uma busca por todas as receitas cadastradas no banco de dados que contenham o valor digitado pelo usuário. A busca é realizada nas variáveis das receitas, como título, ingredientes ou modo de preparo.

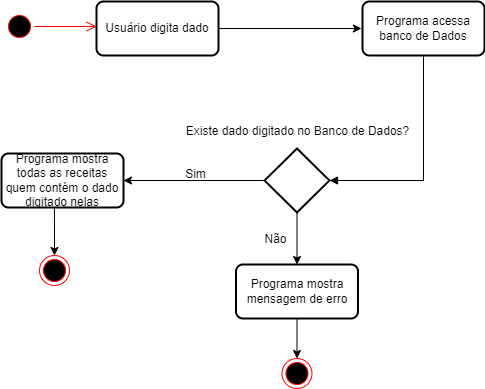
Quando o programa encontra uma ou mais receitas que correspondem ao valor pesquisado, ele as retorna para a interface.

A interface exibe as receitas encontradas em uma lista para o usuário.

Caso o valor digitado pelo usuário não corresponda a nenhuma informação contida dentro da receita no banco de dados, o programa exibe uma mensagem de erro na interface.

O usuário pode então realizar uma nova pesquisa digitando outro valor na **textbox**.

**Diagrama de atividades da pesquisa de receitas:**



Explicando o diagrama, quando o usuário acessa a tela inicial do programa, ele pode pesquisar uma receita, o usuário digita um valor na **textbox** e confirma a pesquisa. Em seguida, o programa verifica se o valor digitado é válido e se há alguma receita correspondente no banco de dados.

Caso o valor seja inválido, o programa exibe uma mensagem de erro na interface.

Caso o valor seja válido, o programa realiza uma busca por todas as receitas que contenham o valor digitado e vai adicionando elas na lista de visualização das receitas positivas quanto ao dado inserido.

Após a busca, o programa exibe todas as receitas encontradas em uma lista para o usuário. O usuário pode selecionar uma das receitas da lista para visualizá-la detalhadamente ou realizar uma nova pesquisa digitando outro valor na **textbox**.

**Documentação de requisitos:**

**Documento de Requisitos - Sistema de Gerenciamento de Receitas**

**Introdução**

Este documento tem como objetivo especificar os requisitos funcionais e não-funcionais para o desenvolvimento de um sistema de gerenciamento de receitas. O sistema permitirá que o usuário adicione novas receitas, visualize as receitas já cadastradas e exclua as receitas desejadas. A implementação do sistema será realizada em linguagem de programação C#.

**Requisitos Funcionais**

**2.1. Adicionar Receita**

O sistema deve permitir que o usuário adicione uma nova receita ao banco de dados. Ao adicionar uma nova receita, o usuário deve informar os seguintes campos obrigatórios:

Título: uma **string** que representa o título da receita;

Ingredientes: uma **string** que representa a lista de ingredientes necessários para a receita;

Modo de Preparo: uma **string** que representa o passo a passo do preparo da receita;

Grau de Dificuldade: uma **string** que representa o grau de dificuldade da receita (fácil, médio ou difícil);

Autor: uma **string** que representa o autor da receita;

Tempo de Preparo: uma **string** que representa o tempo necessário para preparar a receita;

Categoria: uma **string** que representa a categoria da receita (exemplo: sobremesa, prato principal, entrada, etc.);

Quantidade de Pratos: um **inteiro** que representa a quantidade de pratos que a receita rende;

O sistema deve validar se todos os campos obrigatórios foram preenchidos corretamente antes de permitir a adição da nova receita.

**2.2. Visualizar Receitas**

O sistema deve permitir que o usuário visualize todas as receitas cadastradas no banco de dados. As receitas devem ser exibidas em uma lista, mostrando as seguintes informações:

**Título**

**Autor**

**Categoria**

**Ingredientes**

**Tempo de Preparo**

**Modo de Preparo**

**Grau de Dificuldade**

**Quantidade de Porções**

O usuário poderá selecionar uma receita específica da lista para visualizar mais informações sobre a mesma, como a lista de ingredientes, o modo de preparo e a quantidade de pratos que rende.

**2.3. Excluir Receita**

O sistema deve permitir que o usuário exclua uma receita específica do banco de dados. O usuário deverá selecionar a receita que deseja excluir digitando o **Título**, e o sistema deve exibir uma mensagem de confirmação antes de realizar a exclusão.

**Requisitos Não-Funcionais**

**3.1. Segurança**

O sistema deve garantir a segurança dos dados armazenados no banco de dados, evitando o acesso não autorizado e protegendo as informações dos usuários.

**3.2. Usabilidade**

O sistema deve ser fácil de usar e intuitivo, permitindo que o usuário realize todas as operações de forma simples e rápida.

**3.3. Performance**

O sistema deve ser capaz de processar grandes quantidades de dados sem comprometer a performance, garantindo que o tempo de resposta seja adequado para as necessidades dos usuários.

**3.4. Portabilidade**

O sistema deve ser desenvolvido de forma a permitir sua execução em diferentes plataformas e sistemas operacionais, sem comprometer a funcionalidade e a usabilidade.

**Conclusão**

Este documento especificou os requisitos funcionais e não-funcionais para o desenvolvimento de um sistema de gerenciamento de receitas.