ETEC PROFESSOR CAMARGO ARANHA

DESENVOLVIMENTO DE SISTEMA

Julia Peres Cardoso – RM 2301084

Fabrício Rodrigues Pisni – RM 2301305

Marco Antonio Calixto Alves

Arthur Moura Cardoso

Davi Ramos Silva – RM 2301455

**TESTE E QUALIDADE DE SOFTWARE – CRUD**

SÃO PAULO

2025

ETEC PROFESSOR CAMARGO ARANHA

DESENVOLVIMENTO DE SISTEMA

Julia Peres Cardoso – RM 2301084

Fabrício Rodrigues Pisni – RM 2301305

Marco Antonio Calixto Alves

Arthur Moura Cardoso

Davi Ramos Silva – RM 2301455

**TESTE E QUALIDADE DE SOFTWARE – CRUD**

Projeto Prático TCC, ministrado pelos professores Davi Vilar e Luis Lima

SÃO PAULO

2025

**Diagrama Caso de Uso – Sistema de Irrigação**

Diagrama

O conteúdo gerado por IA pode estar incorreto.

O diagrama de casos de uso apresentado na Figura acima representa as principais funcionalidades do sistema proposto, bem como a interação entre o usuário e o sistema. Este sistema tem como objetivo o gerenciamento e automação de baias de irrigação, permitindo ao usuário configurar, monitorar e controlar o processo de irrigação de forma prática e eficiente.

O único ator identificado é o Usuário, que pode representar tanto um agricultor, quanto um técnico ou administrador responsável pelo gerenciamento das baias. A seguir, são descritas as principais funcionalidades do sistema:

* Realizar Cadastro / Efetuar Login: O usuário pode se registrar no sistema, sendo essa ação estendida pelo processo de autenticação (login), garantindo a segurança e individualização do acesso.
* Manter Baia: Esse caso de uso agrupa todas as ações relacionadas à administração das baias de irrigação. Está relacionado diretamente com operações de CRUD (Criar, Ler, Atualizar e Deletar) das baias. Entre as funcionalidades específicas estão:
  1. Acessar Baia, que pode ser estendido para Visualizar Sensores ou Gerenciar Irrigação.
  2. Adicionar Baia, que por sua vez requer o fornecimento de um novo IP para vincular o equipamento ao sistema.
* Gerenciar Irrigação: Após acessar uma baia, o usuário pode configurar os parâmetros de irrigação, como:
  1. Configurar Duração e Configurar Frequência, ambos com a ação obrigatória de Salvar Configurações.
  2. Essa funcionalidade garante controle detalhado sobre os intervalos e tempos de irrigação aplicados à baia selecionada.
* Visualizar Sensores: tem como objetivo permitir ao usuário o acompanhamento em tempo real dos dados captados pelos sensores instalados em cada baia de irrigação
  1. Verificar Funcionamento: Permite que o usuário verifique o status atual da baia, sensores e funcionamento geral do sistema.
  2. Identificar Processo de Irrigação: O usuário define o tipo de processo a ser utilizado:
  3. Automático, no qual o sistema se baseia em critérios pré-configurados.
  4. Manual, onde o controle é feito diretamente pelo usuário.
  5. Ambos os processos incluem a etapa de Configurar Processo, com base nas preferências e necessidades do usuário.

As relações entre os casos de uso utilizam as estereotipagens «include» (inclusão obrigatória) e «extend» (extensão opcional), conforme as necessidades de cada funcionalidade. Essa modelagem garante modularidade, clareza e reuso dos componentes do sistema.

O diagrama evidencia a preocupação em criar um sistema funcional e flexível, voltado para facilitar o manejo de baias de irrigação em ambientes rurais ou hortas urbanas, especialmente com foco na integração com dispositivos físicos, como sensores e controladores (ex: Arduino), por meio da definição de IPs e configurações específicas de cada baia.

Fundamentação Teórica

No contexto da modelagem UML, especialmente ao representar operações CRUD (Criar, Ler, Atualizar e Deletar), é comum utilizar o termo **"Manter"** para agrupar essas funcionalidades em um único caso de uso. Essa abordagem visa simplificar o diagrama de casos de uso, evitando a proliferação de casos de uso individuais para cada operação CRUD.

O uso do caso de uso "Manter" é apropriado quando se deseja representar operações de manutenção de dados que não necessitam de detalhamento individual no diagrama de casos de uso. Por exemplo, em vez de criar casos de uso separados para "Criar Usuário", "Atualizar Usuário", "Excluir Usuário" e "Consultar Usuário", pode-se consolidar essas operações em um único caso de uso denominado "Manter Usuário". Essa prática é recomendada para manter o diagrama mais limpo e compreensível, especialmente em sistemas com múltiplas entidades que requerem operações CRUD similares.

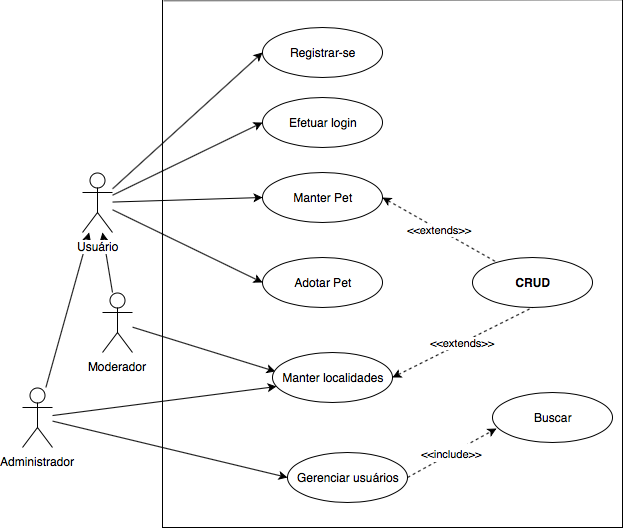
Apesar das vantagens, é essencial avaliar o contexto do sistema para determinar se a consolidação das operações CRUD em um único caso de uso é adequada. Em situações onde cada operação possui regras de negócio distintas ou complexidades específicas, pode ser mais apropriado modelá-las separadamente para capturar todos os requisitos funcionais de forma precisa.

Considere um sistema de gerenciamento de produtos. Em vez de criar casos de uso separados para cada operação CRUD, pode-se ter um único caso de uso:

* **Manter Produto**: Engloba as funcionalidades de adicionar, visualizar, atualizar e remover produtos do sistema.

Essa abordagem proporciona uma visão mais coesa das operações relacionadas à entidade "Produto", facilitando a comunicação entre os membros da equipe de desenvolvimento e os stakeholders.

Em resumo, utilizar o caso de uso "Manter" para representar operações CRUD é uma prática eficaz para simplificar diagramas de casos de uso, desde que aplicada com discernimento e considerando as particularidades de cada sistema.



Exemplo ilustrativo

Em diagramas de casos de uso da UML (Unified Modeling Language), as relações **include**, **extend** e **generalização** (representada por setas) são utilizadas para modelar diferentes tipos de interações entre casos de uso. Aqui está uma explicação sobre quando e como usar cada uma:

**Include (Incluir)**

* **Quando usar**: Utilize a relação *include* quando um caso de uso comum é compartilhado por vários outros casos de uso. Isso ajuda a evitar repetição de descrições e promove a reutilização de funcionalidades.
* **Exemplo**: Se vários casos de uso requerem autenticação do usuário, você pode criar um caso de uso separado chamado "Autenticar Usuário" e incluí-lo nos outros casos de uso que necessitam dessa funcionalidade.

**Extend (Estender)**

* **Quando usar**: A relação *extend* é apropriada quando um caso de uso opcional ou condicional estende o comportamento de outro caso de uso. Isso permite modelar variações no comportamento principal com base em condições específicas.
* **Exemplo**: Um caso de uso "Processar Pedido" pode ser estendido por "Aplicar Desconto" apenas se certas condições forem atendidas, como o cliente ter um código de desconto válido.

**Generalização (Seta com linha cheia e ponta vazada)**

* **Quando usar**: A generalização é usada para mostrar que um caso de uso é uma especialização de outro. Isso é útil quando diferentes casos de uso compartilham comportamentos comuns, mas também têm comportamentos específicos.
* **Exemplo**: Um caso de uso "Gerenciar Conta" pode ter casos de uso especializados como "Gerenciar Conta Pessoal" e "Gerenciar Conta Empresarial", cada um com suas particularidades.

**Dicas Adicionais**

* **Include** é para reutilização obrigatória de funcionalidade.
* **Extend** é para adicionar comportamento opcional ou condicional.
* **Generalização** é para modelar hierarquias de casos de uso com comportamentos comuns e específicos.

Para uma compreensão mais aprofundada, você pode consultar recursos como o artigo "Diferença entre include e extend" de Rogério Zambon, que oferece exemplos práticos e explicações detalhadas sobre o uso dessas relações em diagramas de casos de uso.

Opções para Representar CRUD em Diagramas de Casos de Uso:

1. **1. Caso de Uso Único para CRUD:**
   * Nomenclatura: "Manter Cliente", "Gerenciar Produtos", etc.
   * Vantagens: Simplifica o diagrama, reduz a complexidade e facilita a compreensão.
   * Desvantagens: A descrição do caso de uso deve ser mais detalhada para abranger todas as operações.
   * Exemplo: O caso de uso "Manter Cliente" pode incluir subfluxos para criar, ler, atualizar e excluir clientes.
2. **2. Casos de Uso Individuais:**
   * Nomenclatura: "Criar Cliente", "Ler Cliente", "Atualizar Cliente", "Excluir Cliente".
   * Vantagens: Cada caso de uso é específico e, portanto, mais fácil de entender.
   * Desvantagens: O diagrama fica mais complexo e extenso, com um caso de uso para cada operação.
3. **3. Casos de Uso com "Extends":**
   * Utilização: Cria um caso de uso genérico (ex: "Administrar Usuários") e usa o relacionamento "Extends" para casos de uso específicos (ex: "Criar Usuário".
   * Vantagens: Permite reutilização de código e evita redundância.
   * Desvantagens: Pode tornar o diagrama mais complicado para quem não está familiarizado com o conceito de "Extends".

**Importância da especificação de casos de uso**

A importância dos casos de uso está em sua capacidade de fornecer uma visão clara e concisa do que o sistema fará e como ele será utilizado. Isso ajuda a garantir que os [requisitos do sistema](https://analisederequisitos.com.br/requisitos-funcionais-e-nao-funcionais/) estejam claramente definidos e que as expectativas do usuário sejam atendidas.

Casos de uso podem ser usados como base para o planejamento de testes de aceitação, garantindo que o sistema esteja em conformidade com os requisitos do usuário.

Outra vantagem dos casos de uso é que eles podem ser facilmente compreendidos por pessoas sem conhecimento técnico. Isso significa que eles podem ser usados para comunicação com *stakeholders* do projeto, incluindo [gerentes de projeto](https://analisederequisitos.com.br/o-que-faz-gerente-de-projetos/), usuários finais e desenvolvedores.

Por fim, os casos de uso podem ajudar a identificar e resolver problemas de requisitos precocemente no processo de desenvolvimento. Isso pode economizar tempo e dinheiro, pois problemas identificados tardiamente podem exigir retrabalho significativo

Identificar os atores envolvidos

Identificar os atores envolvidos: quem são os usuários ou sistemas que interagem com o caso de uso? É importante definir seus papéis e responsabilidades no processo.

Determinar pré-condições necessárias

Descrever as pré-condições: quais são as condições que devem ser atendidas para que o caso de uso ocorra?

Por exemplo, se o caso de uso envolve uma compra online, é preciso que o usuário esteja logado e tenha selecionado os itens que deseja comprar.

Detalhar o fluxo principal

Identificar as etapas do fluxo principal: quais são as ações que o usuário ou sistema devem realizar para completar o caso de uso? É importante detalhar cada etapa e as condições de sucesso ou falha.

Detalhar os fluxos alternativos ou de exceções

Descrever as alternativas de fluxo: quais são as ações alternativas que o usuário pode tomar durante o caso de uso? Por exemplo, se o usuário não encontra o produto desejado, ele pode buscar por outros produtos ou abandonar a compra.

Definir o resultado esperado

Definir as pós-condições: o que acontece após a conclusão do caso de uso? É importante definir as condições que devem ser atendidas para que o caso de uso seja considerado concluído com sucesso.