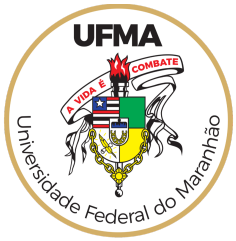


Universidade Federal do Maranhão

Centro de Ciência e Tecnologia - CCET

Bacharelado Interdisciplinar em Ciência e Tecnologia

Alexandre Wesley dos Santos Silva

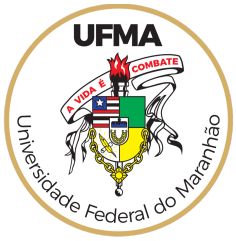
Alyson Clenardo Souza Pereira

Guilherme de Aquino Pacheco

João Manoel Torres Padilha

PROJETO DE DESENVOLVIMENTO DE SOFTWARE: SISTEMA DE GESTÃO DE LOCADORA DE VEÍCULOS

São Luís - Ma, novembro de 2023



Universidade Federal do Maranhão

Centro de Ciência e Tecnologia - CCET

Bacharelado Interdisciplinar em Ciência e Tecnologia

Alexandre Wesley dos Santos Silva

Alyson Clenardo Souza Pereira

Guilherme de Aquino Pacheco

João Manoel Torres Padilha

PROJETO DE DESENVOLVIMENTO DE SOFTWARE: SISTEMA DE GESTÃO DE LOCADORA DE VEÍCULOS

Trabalho apresentado para a disciplina **EECP0011 - Projeto e Desenvolvimento de Software** do curso CIÊNCIA E TECNOLOGIA da Universidade Federal do Maranhão - UFMA, ministrada pelo professor **THALLES LEVI AZEVEDO VALENTE.**

São Luís - Ma, novembro de 2023

Sumário

1. Introdução

Em um mercado altamente dinâmico, a "MasterLocadora" enfrenta desafios operacionais que demandam uma abordagem inovadora para garantir sua relevância e competitividade. Atualmente, a empresa depende de um sistema de gestão manual que, embora tenha sido uma solução em um passado distante, agora se revela inadequado diante das crescentes complexidades do setor de locação de veículos.

Os problemas associados a agendamentos duplicados, falta de transparência nas informações e atrasos na manutenção preventiva são sintomas de um método antiquado que não apenas afeta a eficiência interna, mas também compromete a experiência do cliente. A necessidade de uma transformação digital torna-se evidente, e é nesse contexto que a "MasterLocadora" inicia uma ambiciosa iniciativa de modernização.

O objetivo é claro: implementar um sistema de software eficiente e integrado que não apenas corrija as deficiências operacionais, mas também posicione a empresa como líder em inovação no mercado de locação de veículos. Essa jornada não é apenas uma resposta aos desafios atuais, mas uma visão estratégica para o futuro, onde a eficiência operacional, a transparência nas operações e a excelência na experiência do cliente se tornam os pilares fundamentais para o sucesso duradouro da "MasterLocadora".

2. Objetivo

O objetivo principal deste trabalho é modernizar os processos de gestão de aluguel de veículos da "MasterLocadora" por meio da implementação de um sistema de software eficiente e integrado. Essa iniciativa visa aumentar a eficiência operacional, proporcionar transparência nas operações e aprimorar a experiência do cliente. O foco é não apenas corrigir deficiências operacionais existentes, como agendamentos duplicados e falta de transparência, mas também garantir a competitividade da empresa no mercado de locação de veículos.

3. Processo de Desenvolvimento

O processo de desenvolvimento de software é uma jornada intrincada que visa criar, aprimorar e manter sistemas digitais. Inicia-se com a fase de requisitos, onde são identificadas e compreendidas as necessidades do sistema. Em seguida, entra-se na etapa de projeto, onde esses requisitos são traduzidos em uma arquitetura técnica que define a estrutura do software.

A implementação é o estágio em que o código ganha vida, transformando-se em funcionalidades executáveis. Nesse ponto, o software passa por rigorosos testes para garantir que atende aos requisitos e funcione sem falhas. Com a validação concluída, inicia-se a fase de implantação, levando o software do ambiente de desenvolvimento para o ambiente de produção, onde usuários finais podem interagir com ele.

Contudo, o desenvolvimento de software não é uma jornada com fim. A manutenção contínua se torna uma parte integral, cuidando de atualizações, correções de bugs e melhorias para garantir que o software permaneça relevante e funcional ao longo do tempo.

A importância desse processo é ampla. Não apenas resolve problemas específicos e atende às necessidades do usuário, mas também impulsiona inovações, automatiza processos para melhorar a eficiência operacional e mantém a competitividade no mercado. Além disso, o desenvolvimento de software influencia diretamente a experiência do usuário, determinando a usabilidade e a eficácia do software no mundo cada vez mais digital e interconectado em que vivemos. Em resumo, o desenvolvimento de software é um ciclo constante de planejamento, criação, teste e aprimoramento, garantindo a entrega de soluções tecnológicas adaptadas às demandas em constante evolução.

4. Requisitos Funcionais

Requisitos funcionais são especificações que detalham as ações específicas que um sistema de software deve executar para atender às necessidades do usuário. Sua importância reside na orientação clara para o design e implementação, assegurando que o software atenda às expectativas do cliente. Esses requisitos servem como um guia essencial para desenvolvedores, promovendo uma abordagem estruturada e eficiente no processo de criação de software.

Através de entrevista e coleta de dados foram elencados os seguintes requisitos funcionais:

RF1. Cadastro de Veículos:

- O sistema deve permitir o cadastramento de veículos, incluindo informações como placa, modelo, ano, quilometragem atual, e dados de revisão como data da última revisão e quilometragem correspondente. Além disso, deve ser possível associar cada veículo a uma unidade específica da locadora e definir a taxa de locação diária por classe de veículo.

RF2. Cadastro de Clientes:

- O sistema deve possibilitar o registro completo dos clientes, incluindo nome, telefone, endereço e CPF, facilitando o processo de atendimento e comunicação com os clientes.

RF3. Cadastro de Locação:

- O sistema deve permitir o cadastro detalhado das locações, incluindo informações como funcionário responsável, unidade da locadora, quilometragem de saída e chegada, detalhes do carro alugado (incluindo a placa), datas de saída e chegada, bem como os valores da locação.

RF4. Cadastro de Funcionários:

- O sistema deve permitir o registro completo dos funcionários da locadora, incluindo nome, cargo e unidade de trabalho de cada funcionário, garantindo um controle eficaz sobre a equipe.

RF5. Cadastro de Unidade da Locadora:

- O sistema deve possibilitar o cadastro das unidades da locadora, incluindo informações de localização, nome e detalhes sobre os veículos disponíveis em cada unidade, bem como a pessoa responsável por cada unidade.

RF6. Relatórios:

RF6.1 Relatório de Locação:

- O sistema deve gerar relatórios de locações com a capacidade de aplicar filtros, permitindo segmentar as locações por unidade, datas de saída e chegada, cliente, funcionário e veículo, assim como por valor de locação.

RF6.2 Relatório de Veículos:

- O sistema deve gerar relatórios sobre veículos disponíveis e alugados, segmentados por unidade, com um filtro especial para identificar veículos com mais de 30 mil quilômetros rodados, indicando sua possibilidade de revenda.

5. Requisitos Não Funcionais

Requisitos não funcionais definem critérios além das funcionalidades específicas do sistema, como desempenho e segurança. São essenciais para garantir padrões de qualidade, impactando a eficiência global do software e a satisfação do usuário. Complementam os requisitos funcionais, fornecendo uma visão abrangente para o desenvolvimento de um sistema eficaz**.**

Através de entrevista e coleta de dados foram elencados os seguintes requisitos não funcionais:

RNF1. Segurança e Autenticação:

- O sistema deve oferecer segurança com um sistema de login e senha, garantindo acesso restrito e exclusivo para gestores, funcionários e clientes. As informações devem ser protegidas por criptografia.

RNF2. Interface do Usuário:

- O sistema deve fornecer uma interface com boa experiência do usuário e de uso intuitivo.

RNF3. Desempenho:

- O sistema deve ser capaz de lidar com um grande volume de transações simultâneas, garantindo resposta rápida mesmo durante períodos de alta demanda, como feriados ou estações turísticas.

RNF4. Manutenibilidade:

- O código-fonte do sistema deve ser bem estruturado e comentado para facilitar a manutenção contínua. Além disso, deve ser implementado um sistema eficiente de controle de versão para acompanhar as alterações no código.

6. Casos de Uso

Casos de uso descrevem as interações entre usuários e um sistema, destacando como o sistema responde a ações específicas do usuário. Sua importância reside na representação visual e compreensão clara de como os usuários interagem com o software, ajudando a definir requisitos funcionais. Os casos de uso servem como uma ferramenta crucial na comunicação entre equipes de desenvolvimento e stakeholders, alinhando as expectativas e fornecendo uma visão prática do sistema em ação. Essa abordagem centrada no usuário contribui para a criação de software mais intuitivo, eficiente e alinhado com as necessidades reais dos usuários finais.

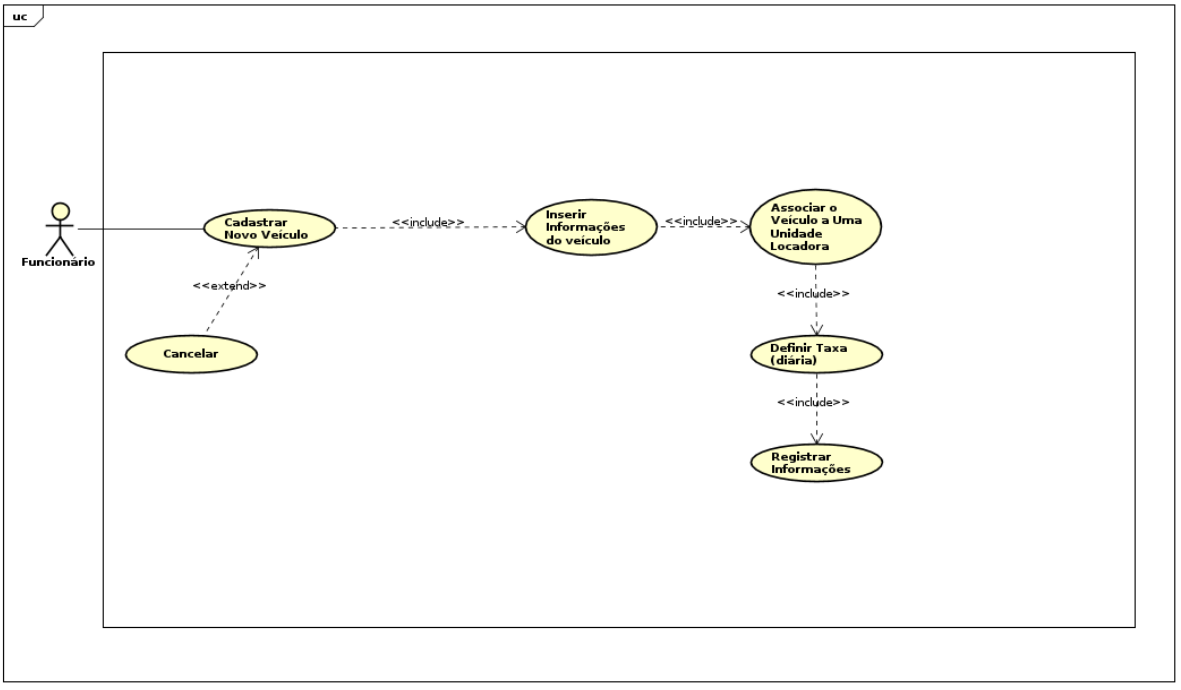
Para cada caso de uso pensado, foi elaborada uma descrição para identificar de melhor forma e especificar suas características.

6.1. Caso de uso Cadastrar Veículo

- **Caso de Uso:** Cadastrar Veículo
- **Objetivo:** Cadastrar um novo veículo no sistema.
- **Atores:** Funcionário
- **Pré-condição:** Nenhuma

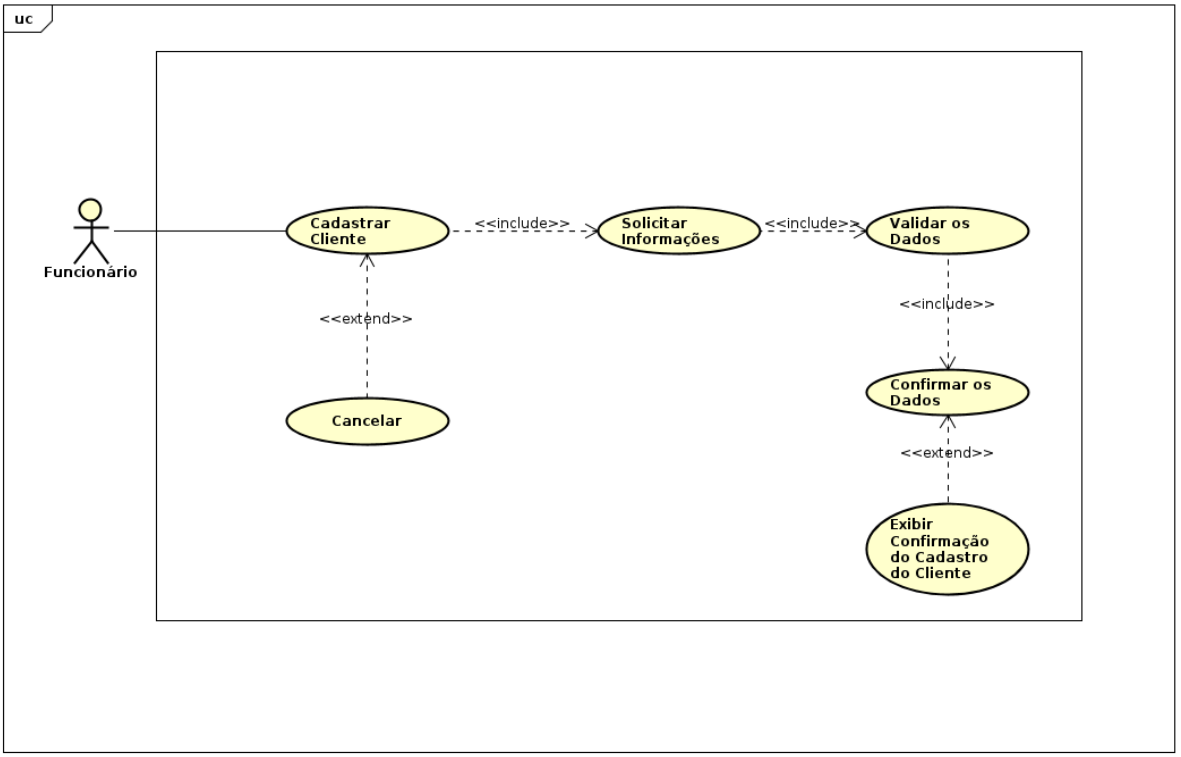
- **Pós-condição:** Veículo cadastrado no sistema com todas as informações fornecidas.
- **Fluxo Principal de Eventos:**
    1. O funcionário seleciona a opção de cadastrar um novo veículo.
    2. O sistema solicita as informações do veículo, como placa, modelo, ano, quilometragem atual, data da última revisão e quilometragem correspondente.
    3. O funcionário fornece as informações solicitadas.
    4. O sistema associa o veículo a uma unidade específica da locadora.
    5. O funcionário define a taxa de locação diária por classe de veículo.
    6. O sistema registra as informações e confirma o cadastro do veículo.
    7. O caso de uso termina.

- **Fluxos Alternativos:**
    1. **Cancelamento:** Em qualquer momento, o funcionário pode cancelar a operação, e o sistema descarta as informações inseridas.
- **Cenário principal:** Cadastro de um novo veículo.
- **Cenários secundários**: Cancelamento.



6.2. Caso de uso Cadastrar Cliente

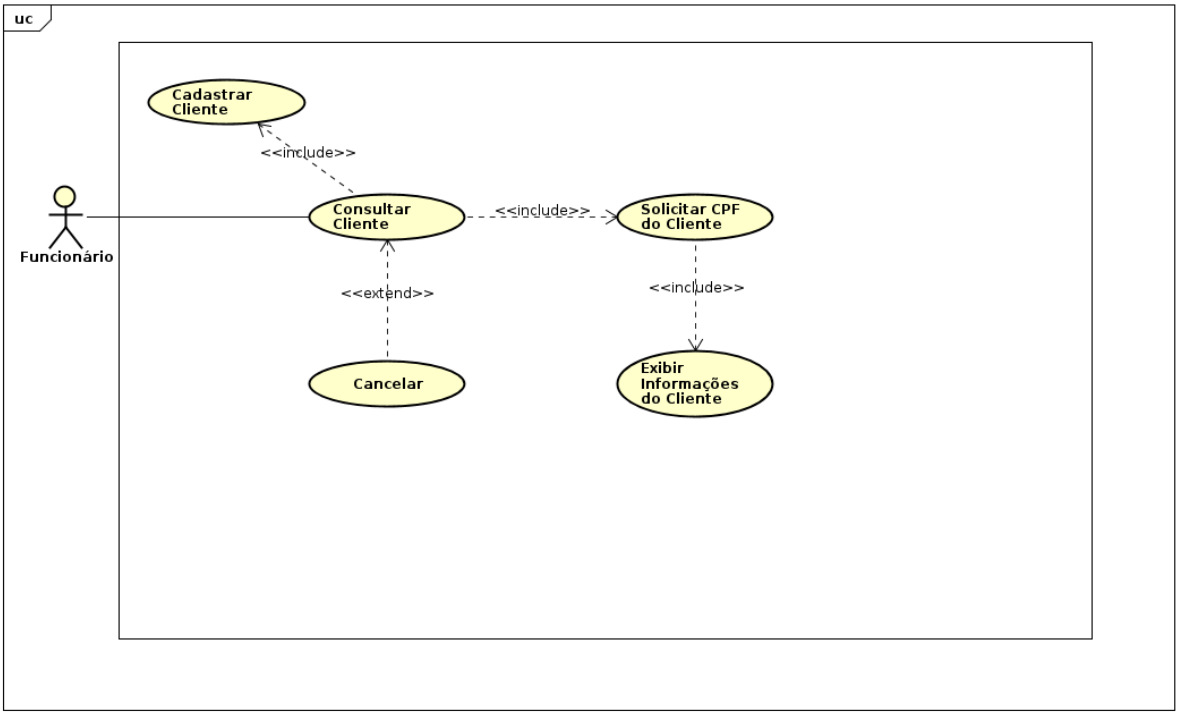
- **Caso de Uso:** Cadastrar Cliente
- **Objetivo**:Registrar um novo cliente no sistema com informações completas para facilitar o atendimento e a comunicação.
- **Atores**:Funcionário
- **Pré-condição:**Nenhuma
- **Pós-condição:**Cliente cadastrado no sistema com todas as informações fornecidas.
- **Fluxo Principal de Eventos:**
    1. O funcionário seleciona a opção de cadastrar um novo cliente.
    2. O sistema solicita as informações do cliente, como nome, telefone, endereço e CPF.
    3. O funcionário fornece as informações solicitadas.
    4. O sistema valida os dados e confirma o cadastro do cliente.
    5. O caso de uso termina.
- **Fluxos Alternativos:**
    1. **Cancelamento**: Em qualquer momento, o funcionário pode cancelar a operação, e o sistema descarta as informações inseridas.



6.3. Caso de Uso Consultar Cliente

- **Caso de Uso**: Consultar Cliente
- **Objetivo**:Consultar as informações de um cliente existente no sistema.
- **Atores:**Funcionário
- **Pré-condição**:Cliente cadastrado no sistema.
- **Pós-condição**:Nenhuma
- **Fluxo Principal de Eventos:**
    1. O funcionário seleciona a opção de consultar cliente.
    2. O sistema exibe um formulário de busca.
    3. O funcionário insere o CPF do cliente a ser consultado.
    4. O sistema apresenta as informações do cliente.
    5. O caso de uso termina.

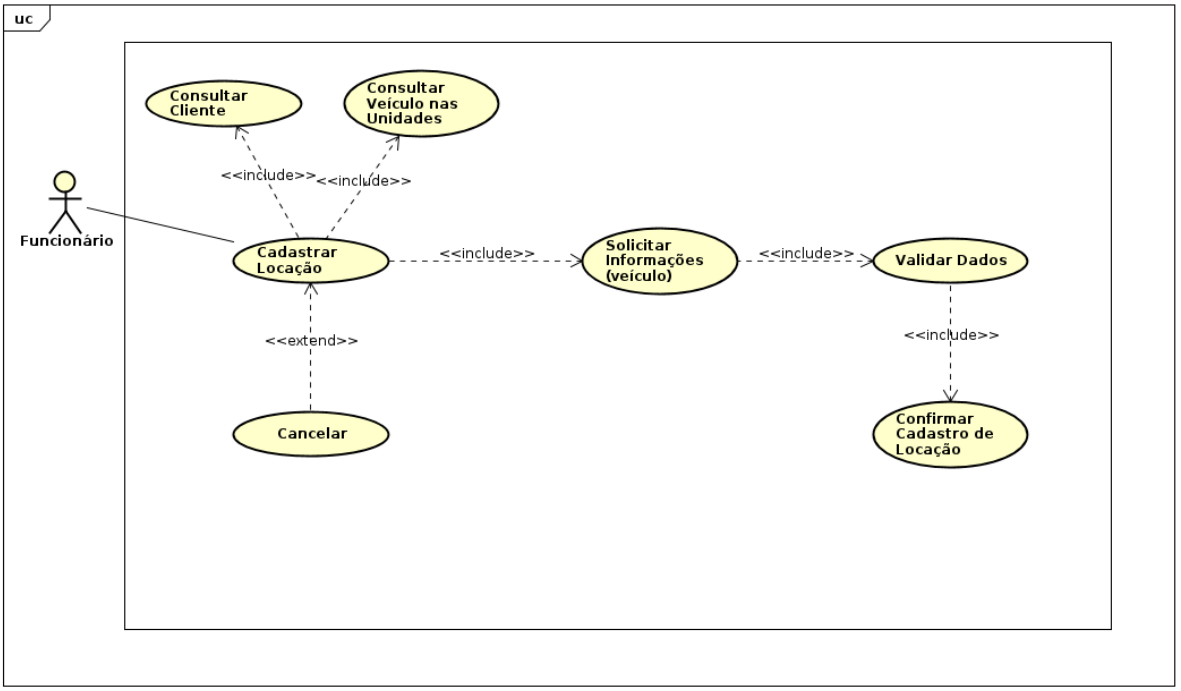
- **Fluxos Alternativos:**
    1. **Cancelamento:** Em qualquer momento, o funcionário pode cancelar a operação, e o sistema retorna ao menu principal.



6.4. Caso de Uso Cadastrar Locação

- **Caso de Uso**: Cadastrar Locação
- **Objetivo**: Registrar detalhes completos de uma locação no sistema
- **Atores:** Funcionário
- **Pré-condição:** Cliente cadastrado no sistema, veículo disponível para locação.
- **Pós-condição:** Locação registrada no sistema com todas as informações fornecidas.
- **Fluxo Principal de Eventos:**
    1. O funcionário seleciona a opção de cadastrar uma nova locação.
    2. O sistema solicita as informações necessárias, como:
    3. O funcionário fornece as informações solicitadas.
    4. O sistema valida os dados e confirma o cadastro da locação.
    5. O caso de uso termina.

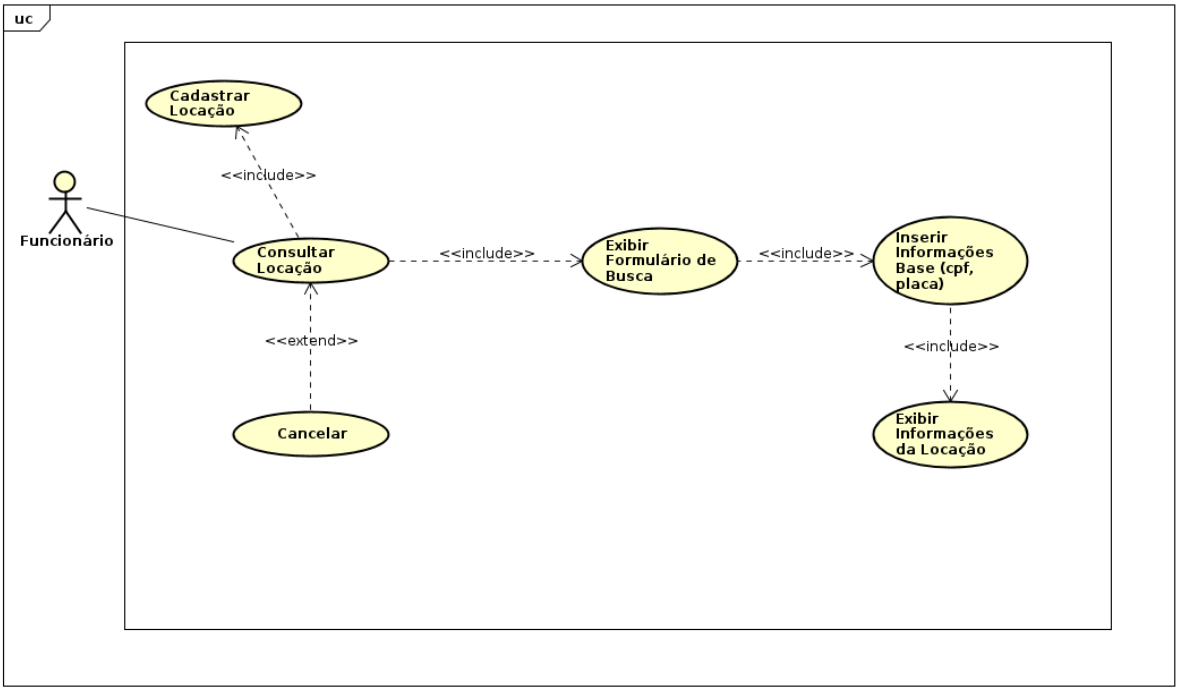
- **Fluxos Alternativos:**
    1. **Cancelamento:** Em qualquer momento, o funcionário pode cancelar a operação, e o sistema descarta as informações inseridas.



6.5. Caso de Uso Consultar Locação

- **Caso de Uso:** Consultar Locação
- **Objetivo:** Consultar as informações de uma locação existente no sistema.
- **Atores:** Funcionário
- **Pré-condição:** Locação registrada no sistema.
- **Pós-condição:** Nenhuma
- **Fluxo Principal de Eventos:**
    1. O funcionário seleciona a opção de consultar a locação.
    2. O sistema exibe um formulário de busca.
    3. O funcionário insere informações relevantes, como o número da locação, nome do cliente, ou placa do veículo.
    4. O sistema apresenta as informações da locação.
    5. O caso de uso termina.

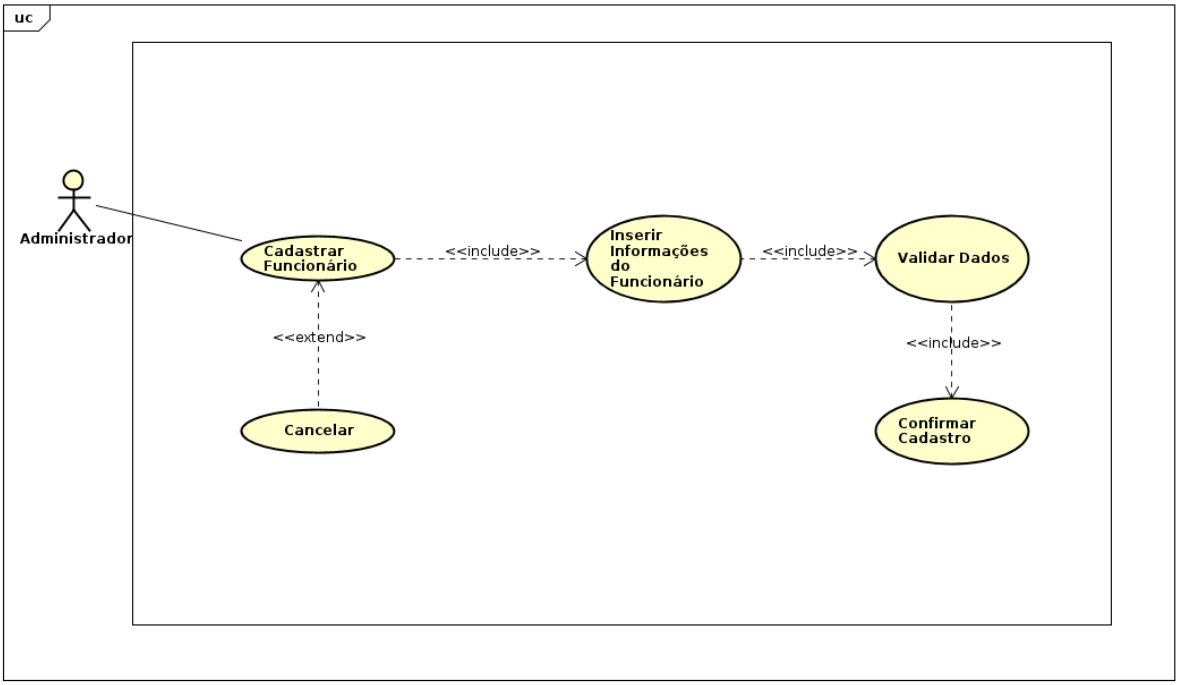
- **Fluxos Alternativos:**
    1. **Cancelamento:** Em qualquer momento, o funcionário pode cancelar a operação, e o sistema retorna ao menu principal.



6.6. Caso de Uso Cadastrar Funcionário

- **Caso de Uso:** Cadastrar Funcionário
- **Objetivo:** Registrar informações completas de um novo funcionário.
- **Atores:** Administrador.
- **Pré-condição:** Nenhuma
- **Pós-condição:** Funcionário cadastrado no sistema com todas as informações fornecidas.
- **Fluxo Principal de Eventos:**
    1. O administrador seleciona a opção de cadastrar um novo funcionário.
    2. O sistema solicita as informações do funcionário.
    3. O administrador fornece as informações solicitadas.
    4. O sistema valida os dados e confirma o cadastro do funcionário.
    5. O caso de uso termina.

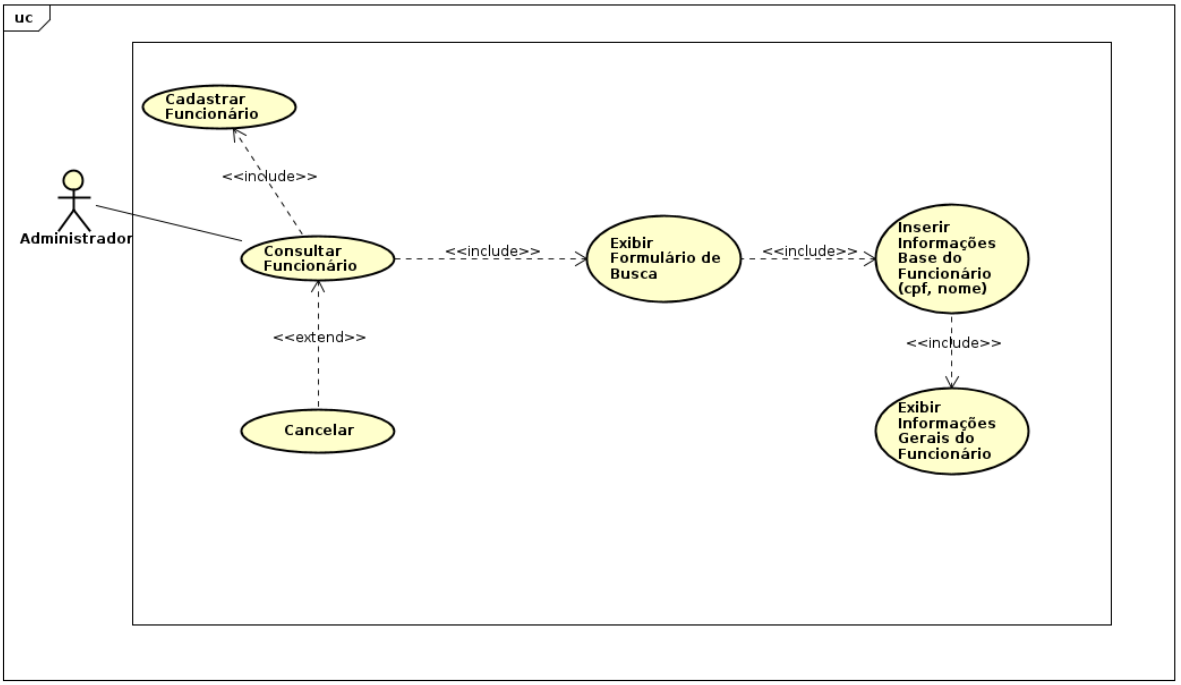
- **Fluxos Alternativos:**
    1. **Cancelamento:** Em qualquer momento, o administrador pode cancelar a operação, e o sistema descarta as informações inseridas.



6.7. Caso de Uso Consultar Funcionário

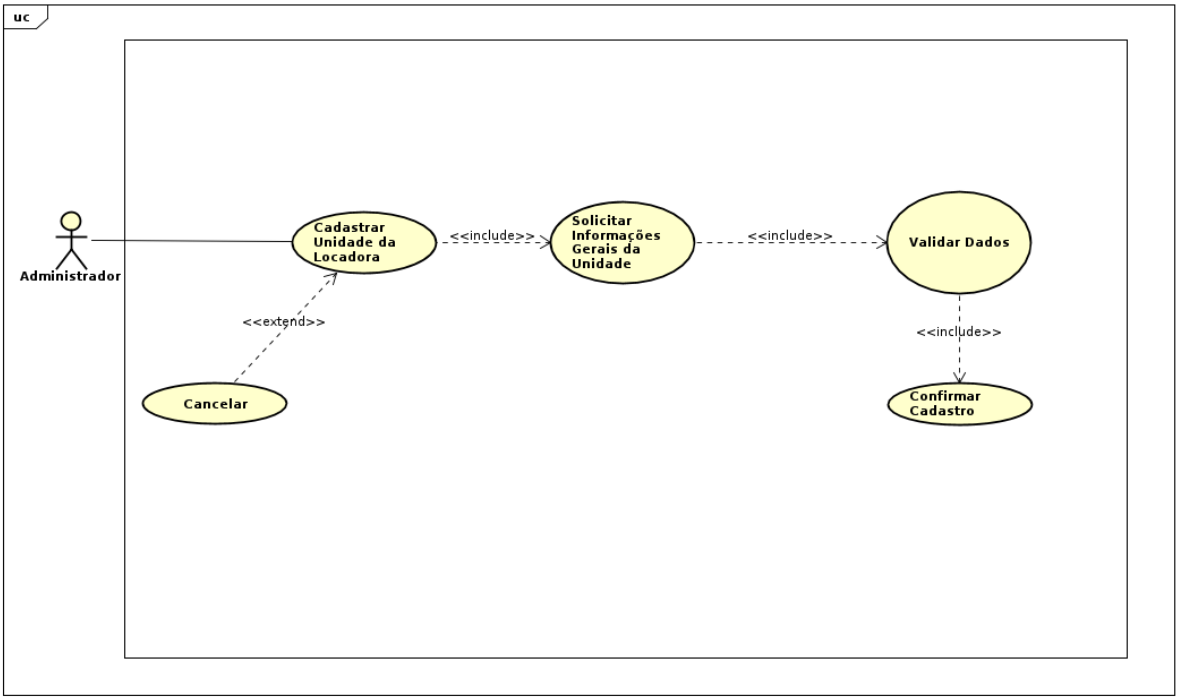
- **Caso de Uso:** Consultar Funcionário
- **Objetivo:** Consultar as informações de um funcionário existente no sistema.
- **Atores:** Administrador.
- **Pré-condição:** Funcionário cadastrado no sistema.
- **Pós-condição:** Nenhuma
- **Fluxo Principal de Eventos:**
    1. O administrador seleciona a opção de consultar o funcionário.
    2. O sistema exibe um formulário de busca.
    3. O administrador insere informações relevantes, como o nome ou cargo do funcionário.
    4. O sistema apresenta as informações do funcionário.
    5. O caso de uso termina.

- **Fluxos Alternativos:**
    1. **Cancelamento:** Em qualquer momento, o administrador pode cancelar a operação, e o sistema retorna ao menu principal.



6.8. Caso de Uso Cadastrar Unidade da Locadora

- **Caso de Uso:** Cadastrar Unidade da Locadora
- **Objetivo:** Registrar as informações de uma nova unidade da locadora no sistema.
- **Atores:** Administrador
- **Pré-condição:** Nenhuma
- **Pós-condição:** Unidade da locadora cadastrada no sistema com todas as informações fornecidas.
- **Fluxo Principal de Eventos:**
    1. O administrador seleciona a opção de cadastrar uma nova unidade da locadora.
    2. O sistema solicita as informações da unidade, como localização, nome e pessoa responsável.
    3. O administrador fornece as informações solicitadas.
    4. O sistema valida os dados e confirma o cadastro da unidade.
    5. O caso de uso termina.
- **Fluxos Alternativos:**
    1. **Cancelamento**: Em qualquer momento, o administrador pode cancelar a operação, e o sistema descarta as informações inseridas.

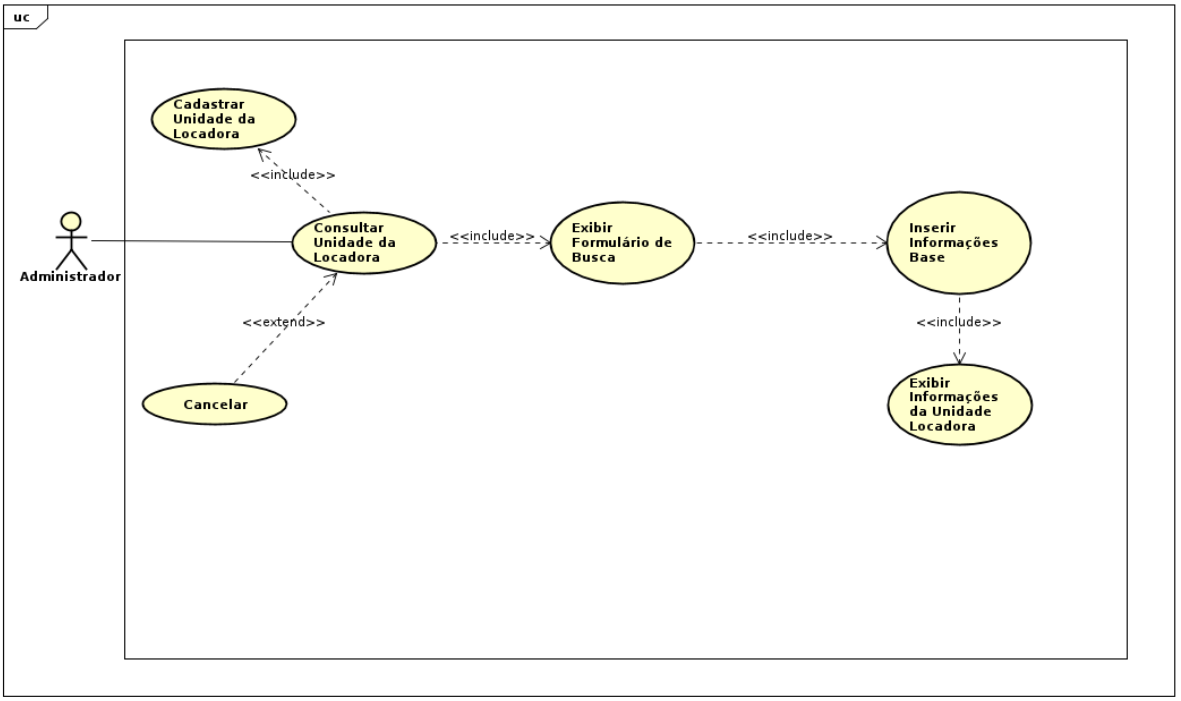


6.9. Caso de Uso: Consultar Unidade da Locadora

- **Caso de Uso**: Consultar Unidade da Locadora
- **Objetivo:** Permitir que o administrador consulte as informações de uma unidade da locadora existente no sistema.
- **Atores:** Administrador
- **Pré-condição:** Unidade da locadora cadastrada no sistema.
- **Pós-condição:** Nenhuma
- **Fluxo Principal de Eventos:**
    1. O administrador seleciona a opção de consultar unidade da locadora.
    2. O sistema exibe um formulário de busca.
    3. O administrador insere informações relevantes, como o nome ou localização da unidade.
    4. O sistema apresenta as informações da unidade da locadora.
    5. O caso de uso termina.

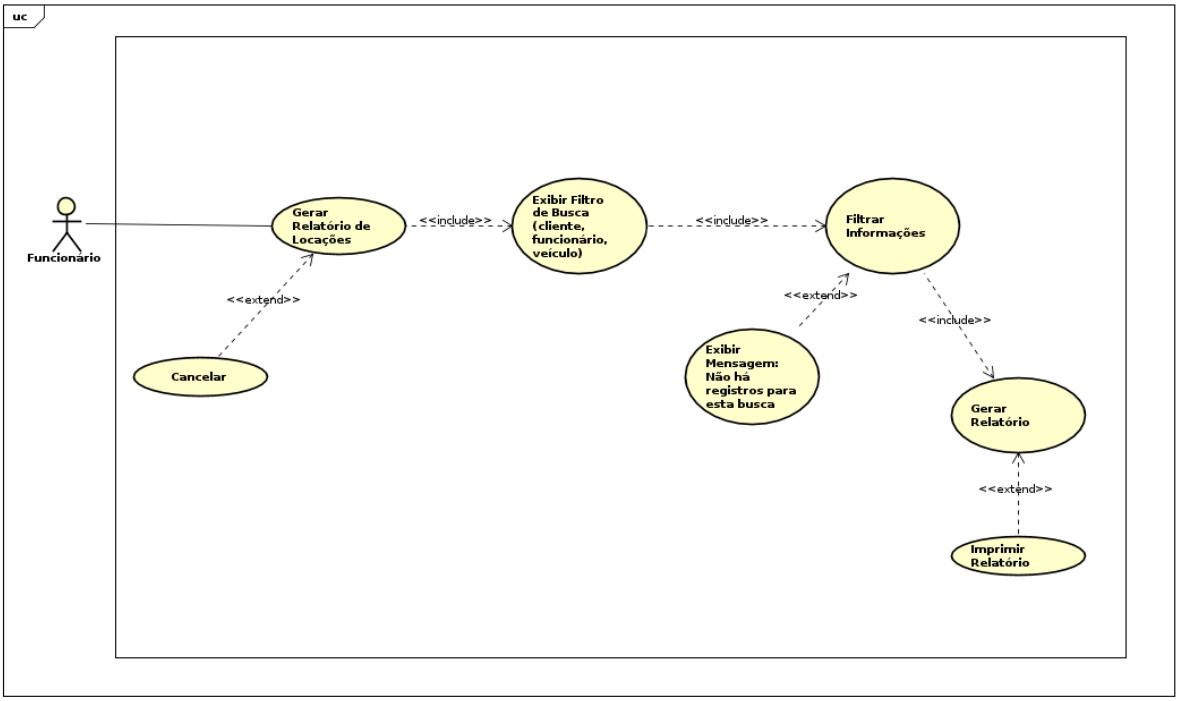
Fluxos Alternativos:

1. **Cancelamento:** Em qualquer momento, o administrador pode cancelar a operação, e o sistema retorna ao menu principal.



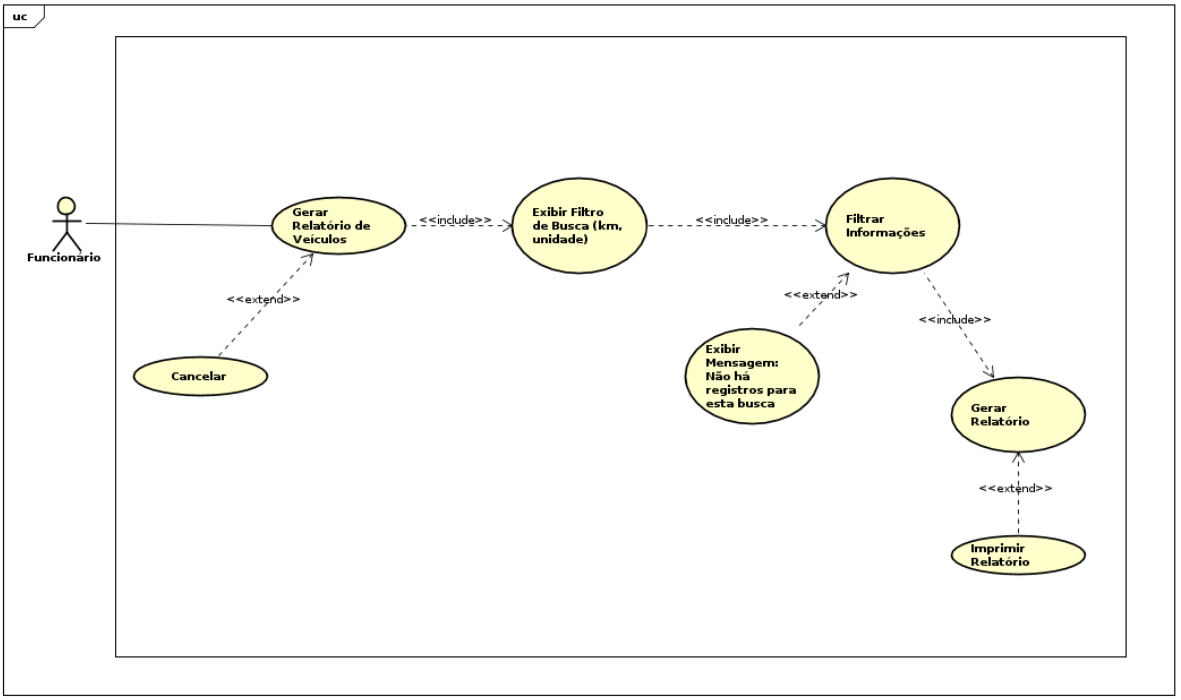
6.10. Caso de Uso Gerar Relatório de Locações

- **Caso de Uso:** Gerar Relatório de Locações
- **Objetivo:** Permitir que os funcionários gerem relatórios de locações.
- **Atores:** Funcionário
- **Pré-condição:** Existência de dados cadastrados.
- **Pós-condição:** Nenhuma
- **Fluxo Principal de Eventos:**
    1. O funcionário acessa a opção de gerar relatório de locações.
    2. O sistema exibe opções de filtros, como unidade, datas de saída e chegada, cliente, funcionário, veículo e valor de locação.
    3. O funcionário seleciona os filtros desejados.
    4. O sistema gera o relatório de locações com base nos filtros aplicados.
    5. O funcionário visualiza e imprime o relatório.
    6. O caso de uso termina.
- **Fluxos Alternativos:**
    1. **Cancelamento:** Em qualquer momento, o funcionário pode cancelar a operação, e o sistema retorna ao menu principal.



6.11. Caso de Uso Gerar Relatório de Veículos

- **Caso de Uso:** Gerar Relatório de Veículos
- **Objetivo:** Permitir que os funcionários gerem relatórios sobre veículos disponíveis e alugados.
- **Atores:** Funcionário
- **Pré-condição:** Existência de dados cadastrados.
- **Pós-condição:** Nenhuma
- **Fluxo Principal de Eventos:**
    1. O funcionário acessa a opção de gerar relatório de veículos.
    2. O sistema exibe opções de filtros, como unidade e condição dos veículos (com mais de 30 mil quilômetros rodados).
    3. O funcionário seleciona os filtros desejados.
    4. O sistema gera o relatório de veículos com base nos filtros aplicados.
    5. O funcionário visualiza e imprime o relatório.
    6. O caso de uso termina.
- **Fluxos Alternativos:**
    1. **Cancelamento:** Em qualquer momento, o funcionário pode cancelar a operação, e o sistema retorna ao menu principal.



7. Diagrama de Classes

Um diagrama de classes é uma representação visual de classes, interfaces, associações e seus relacionamentos em um sistema orientado a objetos. Ele é uma parte essencial da modelagem de sistemas usando a linguagem de modelagem unificada (UML). Uma classe é um modelo ou blueprint para criar objetos. Ela define atributos (propriedades) e métodos (comportamentos) que os objetos criados a partir dela terão. Por exemplo, a classe "Carro" pode ter atributos como "modelo" e "ano", e métodos como "ligar" e "desligar".

Atributos são características ou propriedades de uma classe que descrevem o estado dos objetos dessa classe. Por exemplo, uma classe "Pessoa" pode ter atributos como "nome", "idade" e "endereço".

Métodos são funções ou procedimentos associados a uma classe que definem seu comportamento. Eles representam as ações que os objetos dessa classe podem realizar. Por exemplo, uma classe "Círculo" pode ter métodos para calcular a área e o perímetro.

A associação representa a relação entre duas ou mais classes. Ela pode indicar a conexão entre objetos de diferentes classes. Por exemplo, uma associação entre as classes "Professor" e "Disciplina" pode indicar que um professor leciona uma disciplina.

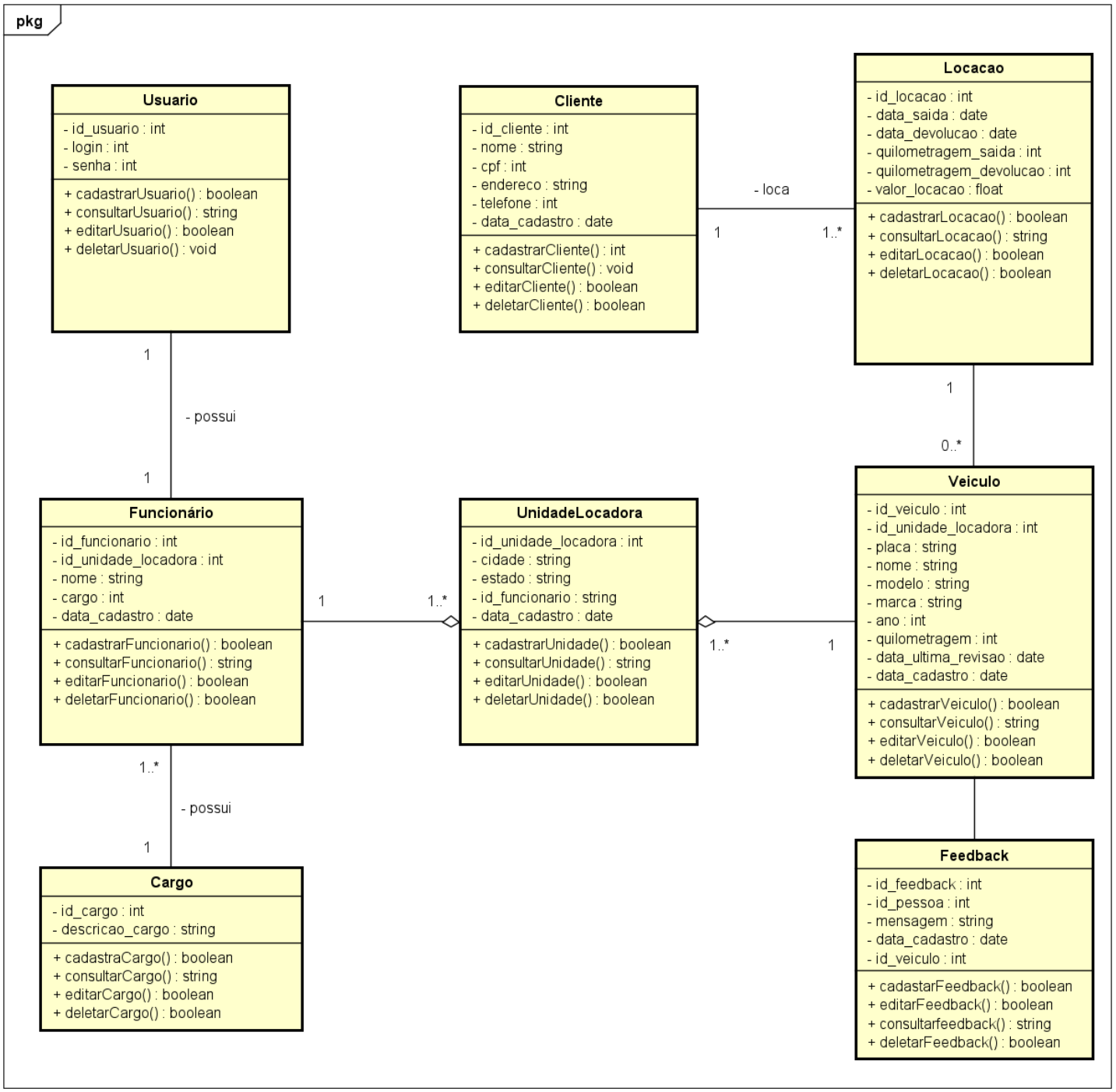
Herança é um mecanismo que permite que uma classe herda atributos e métodos de outra classe. A classe que fornece a herança é chamada de superclasse, e a classe que herda é chamada de subclasse. Isso promove a reutilização de código. Por exemplo, uma classe "Estudante" pode herdar da classe "Pessoa".

Agregação é uma forma de associação que indica que uma classe é composta por outras classes, mas essas classes podem existir independentemente. Por exemplo, uma classe "Universidade" pode ser agregada a várias classes "Departamento".

Composição é uma forma mais forte de agregação, indicando que uma classe é composta por outras classes e que essas partes não podem existir independentemente da classe principal. Por exemplo, uma classe "Carro" pode ser composta por uma classe "Motor".

Dependência ocorre quando uma classe usa outra, mas não há uma relação estrutural entre elas. Por exemplo, se a classe "Pedido" utiliza a classe "Cliente" como um parâmetro em um de seus métodos, há uma dependência entre essas classes.

A imagem a seguir ilustra como as classes interagem entre si.

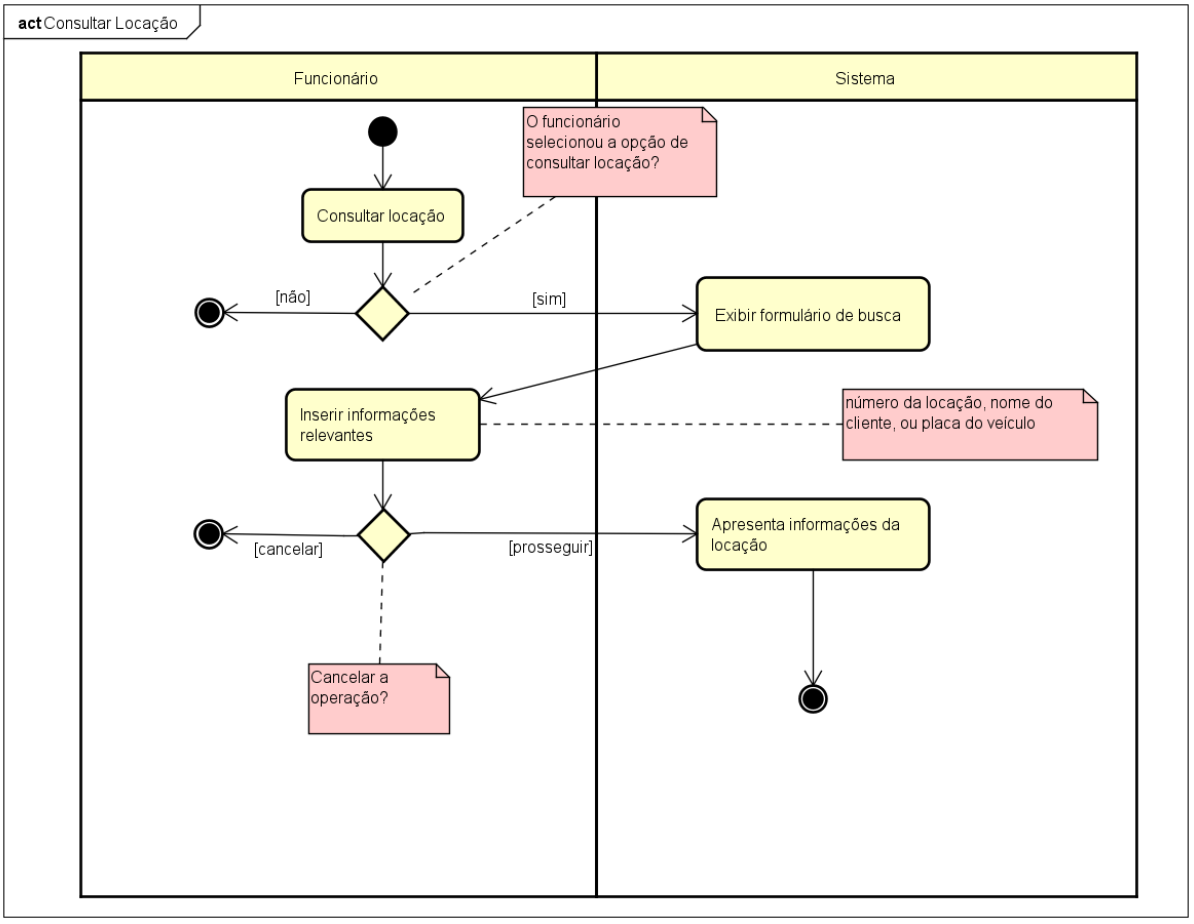

pkg
Usuario
- id_usuario : int
- login : int
- senha : int
+ cadastrarUsuario() : boolean
+ consultarUsuario() : string
+ editarUsuario() : boolean
+ deletarUsuario() : void
Cliente
- id_cliente : int
- nome : string
- cpf : int
- endereco : string
- telefone : int
- data_cadastro : date
+ cadastrarCliente() : int
+ consultarCliente() : void
+ editarCliente() : boolean
+ deletarCliente() : boolean
Locacao
- id_locacao : int
- data_saida : date
- data_devolucao : date
- quilometragem_saida : int
- quilometragem_devolucao : int
- valor_locacao : float
+ cadastrarLocacao() : boolean
+ consultarLocacao() : string
+ editarLocacao() : boolean
+ deletarLocacao() : boolean
- loca
1
1..*
1
- possui
1
1
0..*
Funcionário
- id_funcionario : int
- id_unidade_locadora : int
- nome : string
- cargo : int
- data_cadastro : date
+ cadastrarFuncionario() : boolean
+ consultarFuncionario() : string
+ editarFuncionario() : boolean
+ deletarFuncionario() : boolean
1
1..*
UnidadeLocadora
- id_unidade_locadora : int
- cidade : string
- estado : string
- id_funcionario : string
- data_cadastro : date
+ cadastrarUnidade() : boolean
+ consultarUnidade() : string
+ editarUnidade() : boolean
+ deletarUnidade() : boolean
1..*
1
Veiculo
- id_veiculo : int
- id_unidade_locadora : int
- placa : string
- nome : string
- modelo : string
- marca : string
- ano : int
- quilometragem : int
- data_ultima_revisao : date
- data_cadastro : date
+ cadastrarVeiculo() : boolean
+ consultarVeiculo() : string
+ editarVeiculo() : boolean
+ deletarVeiculo() : boolean
1..*
- possui
1
Cargo
- id_cargo : int
- descricao_cargo : string
+ cadastraCargo() : boolean
+ consultarCargo() : string
+ editarCargo() : boolean
+ deletarCargo() : boolean
Feedback
- id_feedback : int
- id_pessoa : int
- mensagem : string
- data_cadastro : date
- id_veiculo : int
+ cadastarFeedback() : boolean
+ editarFeedback() : boolean
+ consultarfeedback() : string
+ deletarFeedback() : boolean

8. Diagrama de Atividades

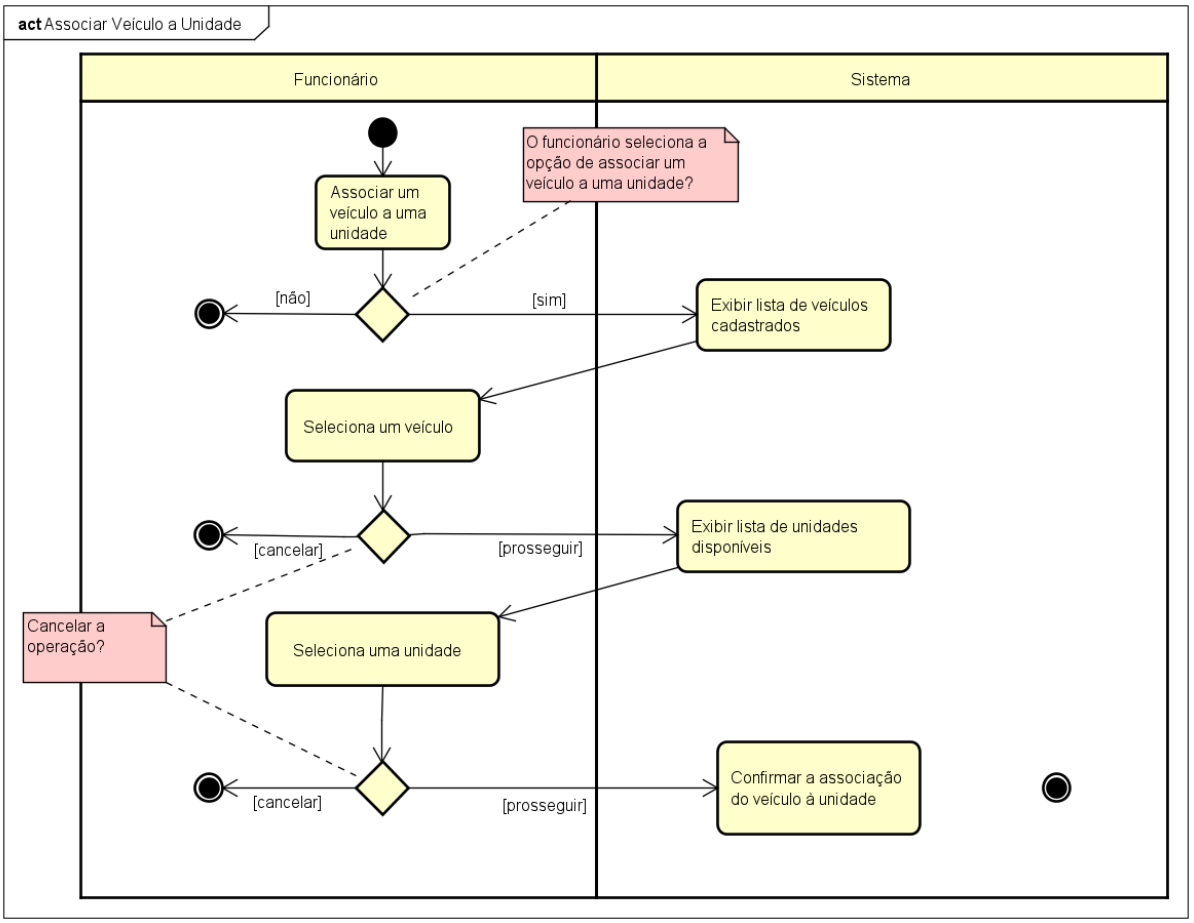
O diagrama de atividade fornece uma visualização do comportamento de um sistema descrevendo a sequência de ações em um processo. Os diagramas de atividades são semelhantes a fluxogramas porque mostram o fluxo entre as ações em uma atividade; no entanto, os diagramas de atividades também podem mostrar fluxos paralelos ou simultâneos e fluxos alternativos.

Na sequência estão os diagramas de atividade para cada caso de uso descrito anteriormente.

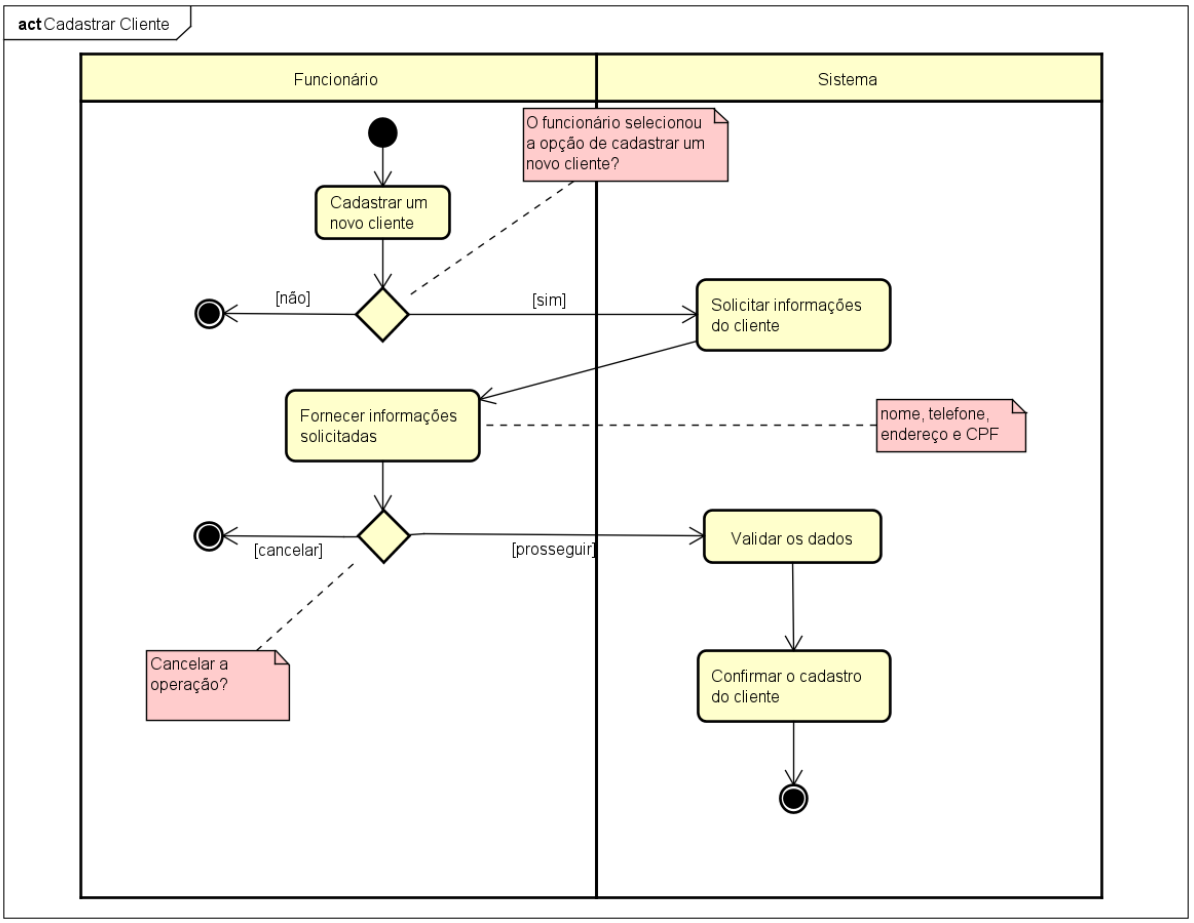
8.1. Caso de Uso Cadastrar Veículo



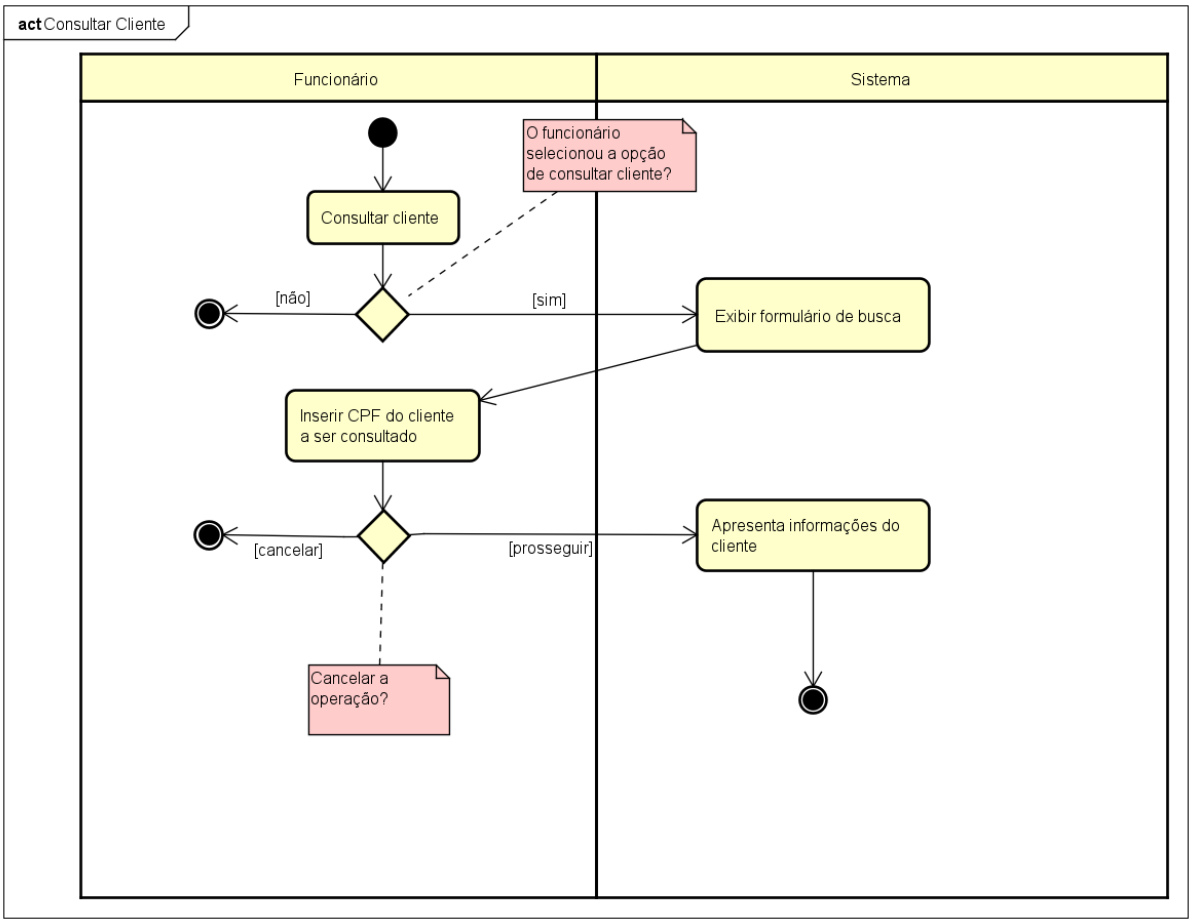
8.2. Caso de Uso Associar Veículo a Unidade



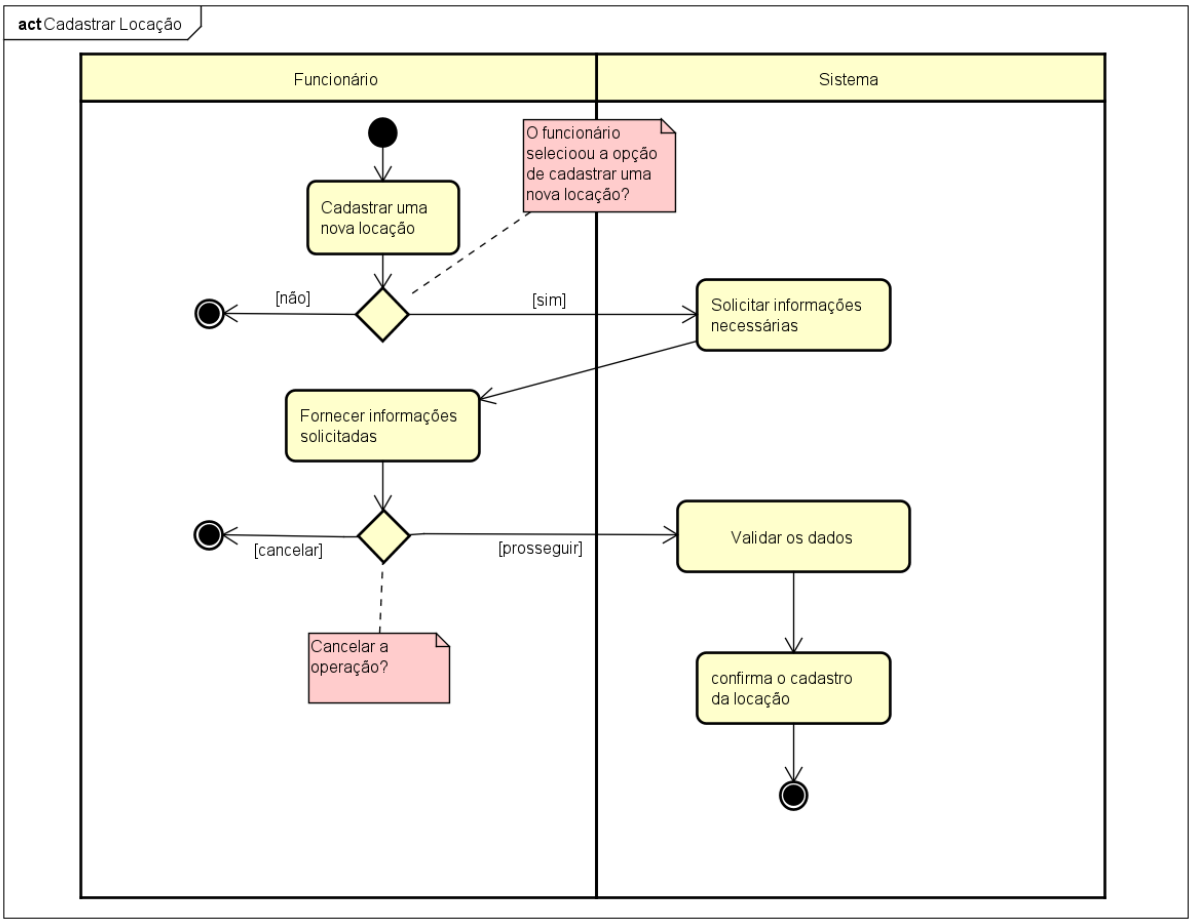
8.3. Caso de uso Cadastrar Cliente



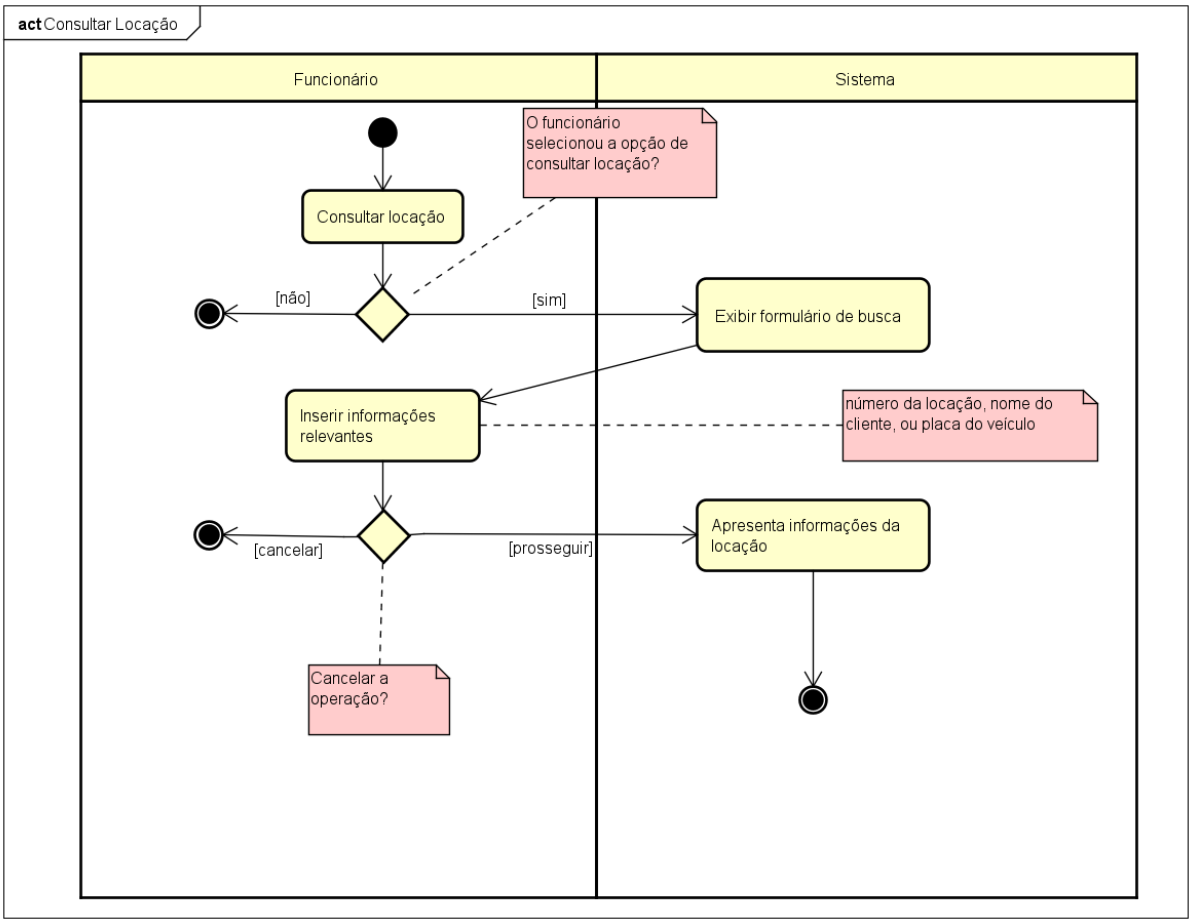
8.4. Caso de Uso Consultar Cliente



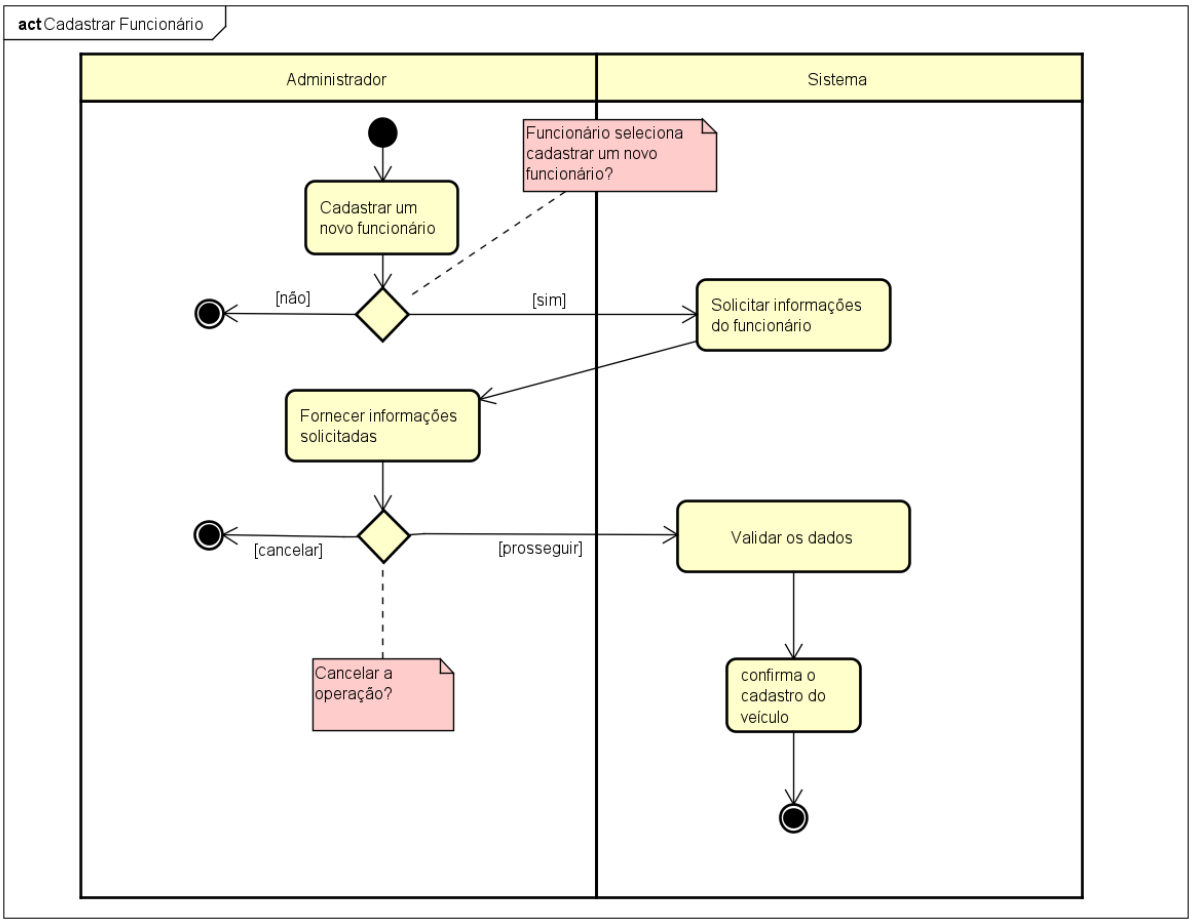
8.5. Caso de Uso Cadastrar Locação



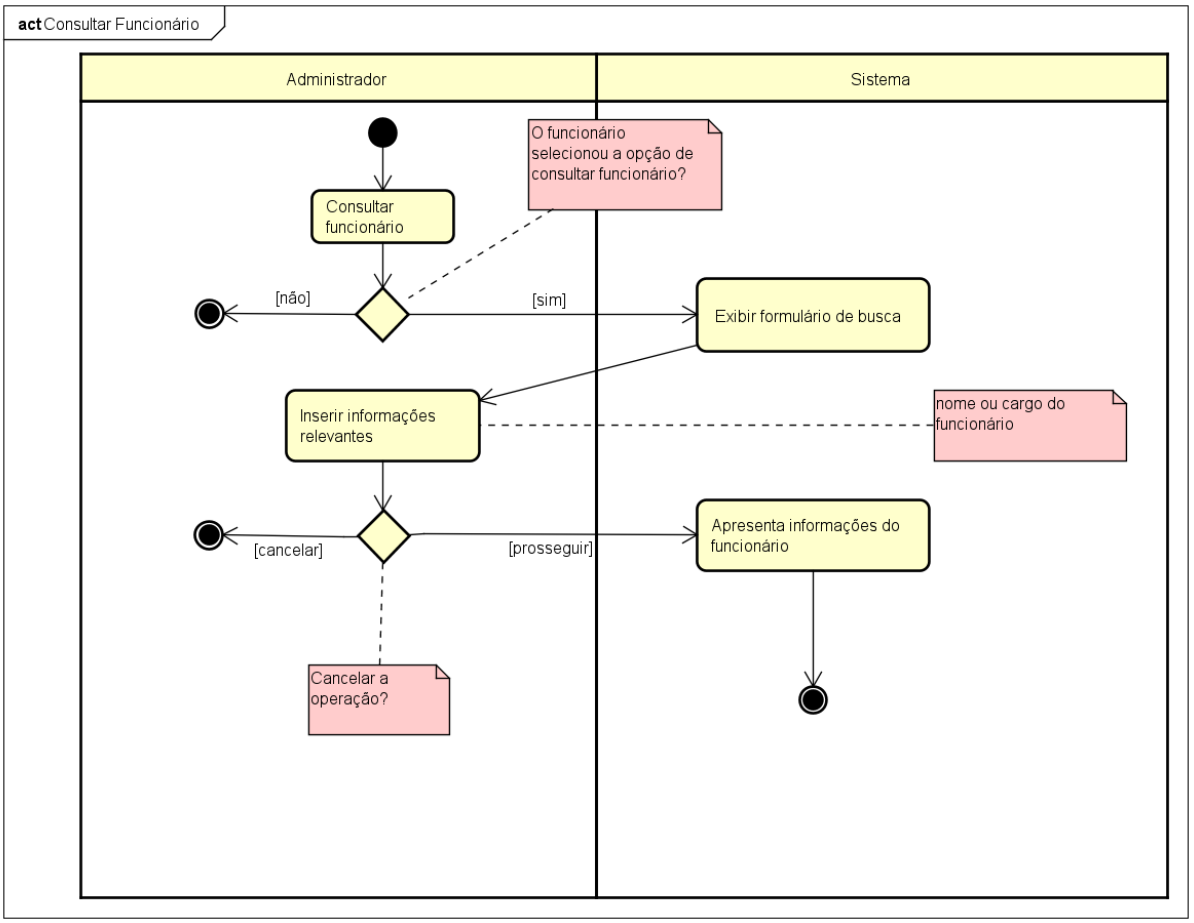
8.6. Caso de Uso Consultar Locação



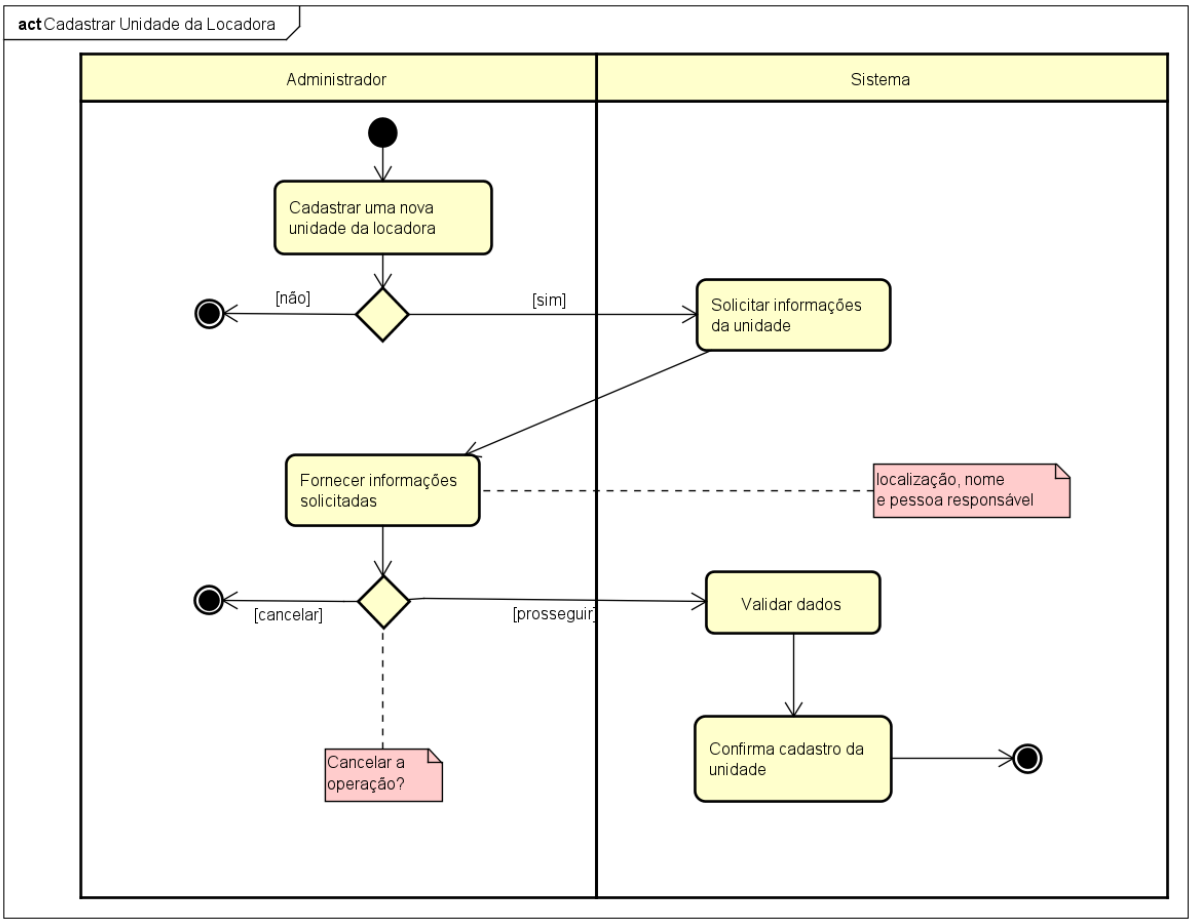
8.7. Caso de Uso Cadastrar Funcionário



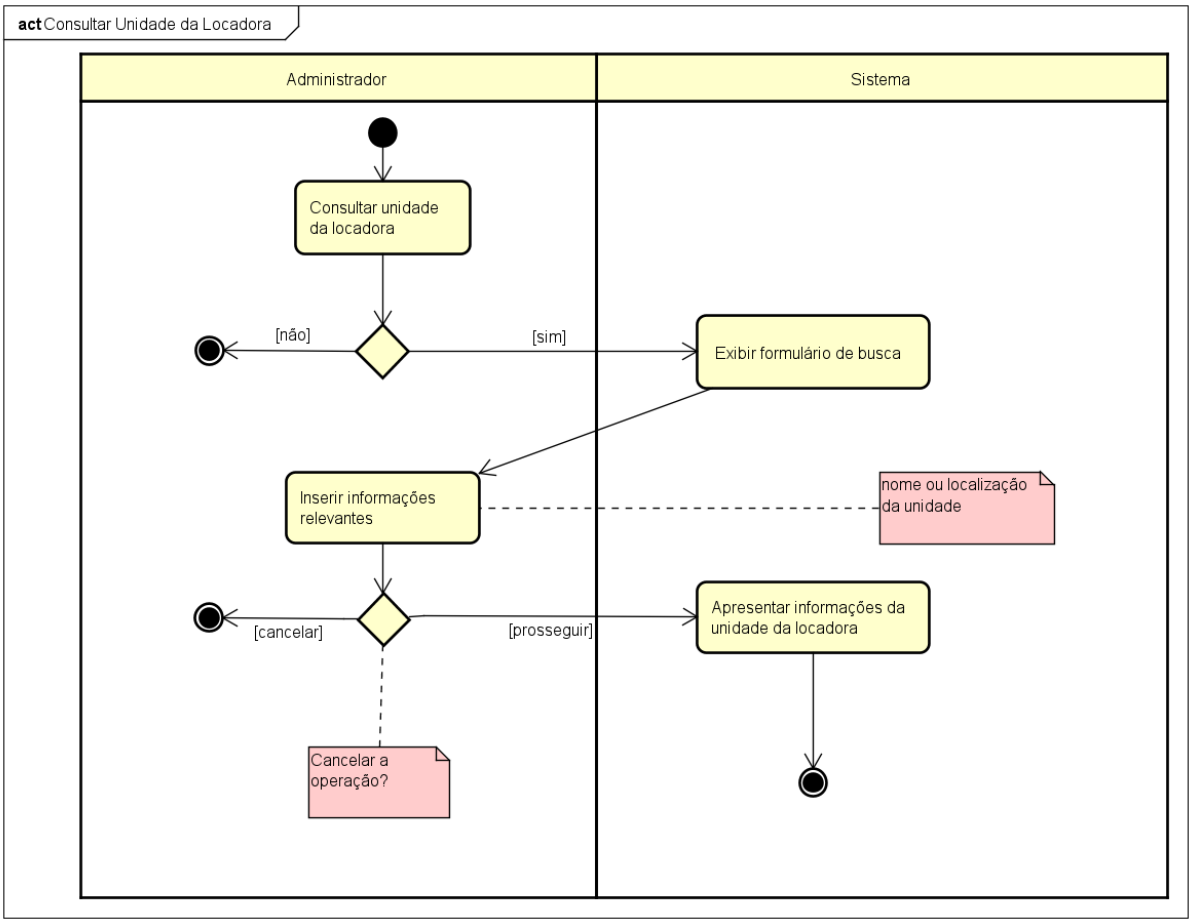
8.8. Caso de Uso Consultar Funcionário



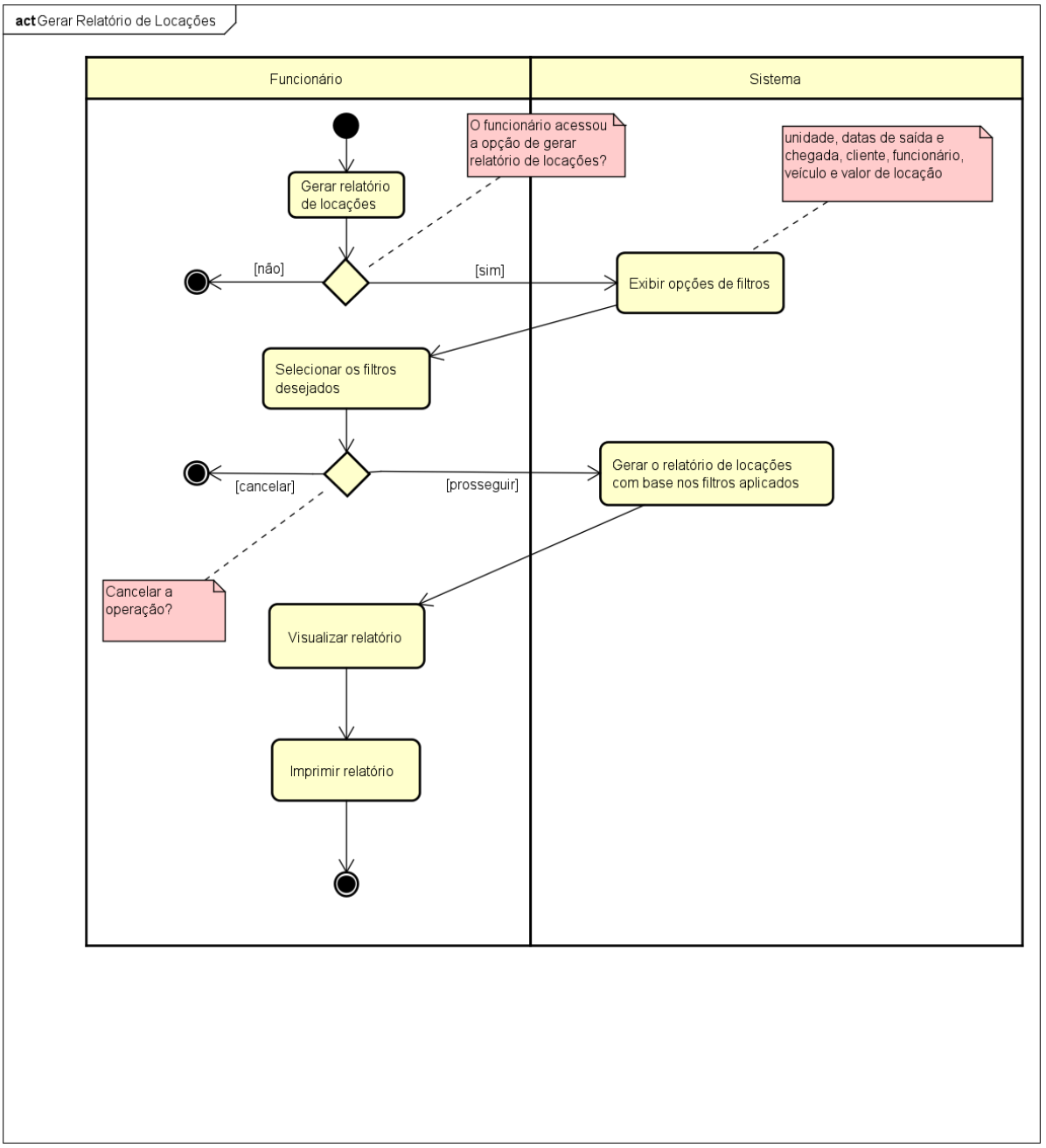
8.9. Caso de Uso Cadastrar Unidade da Locadora



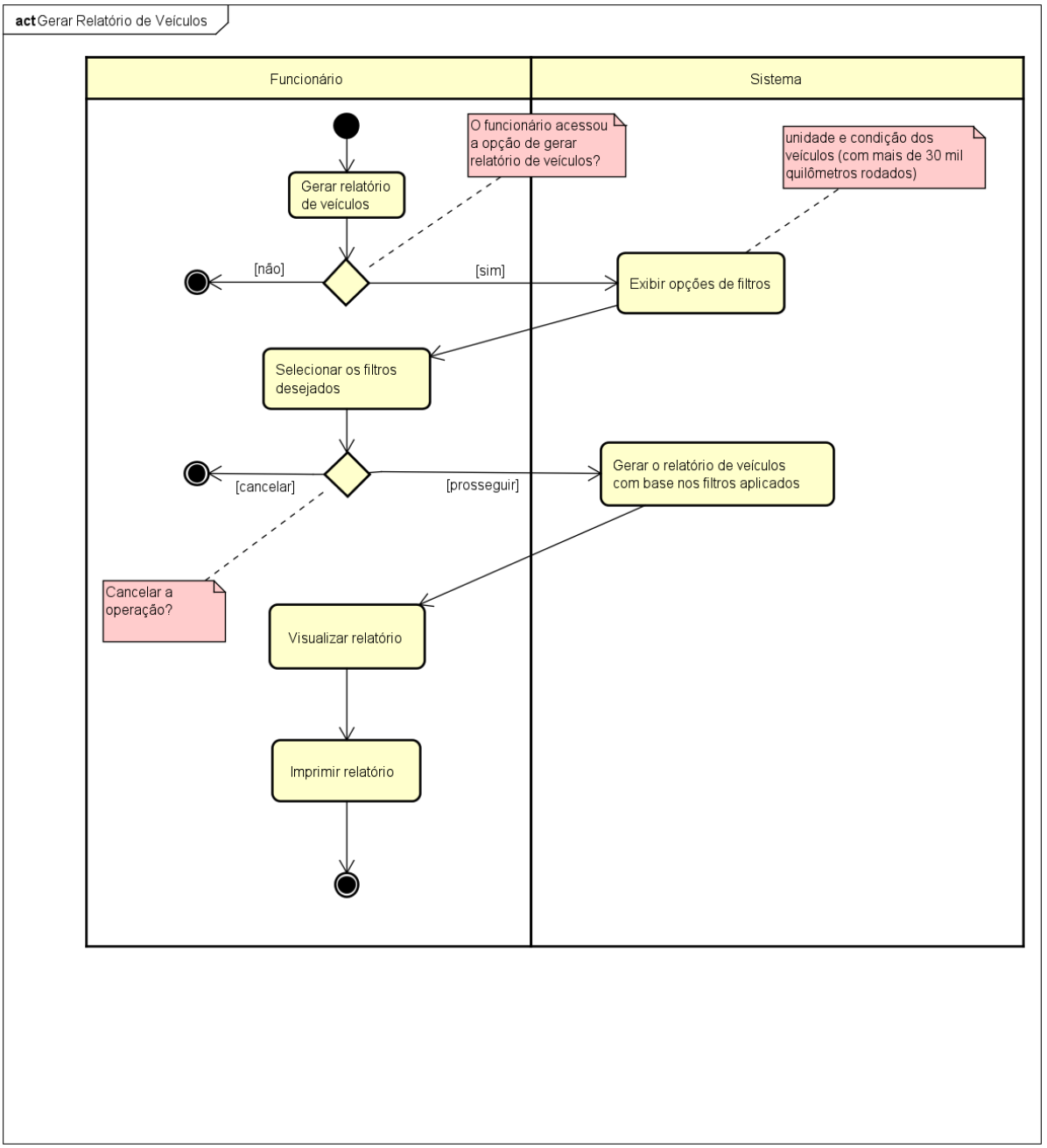
8.10. Caso de Uso Consultar Unidade da Locadora



8.11. Caso de Uso Gerar Relatório de Locações



8.12. Caso de Uso Gerar Relatório de Veículos



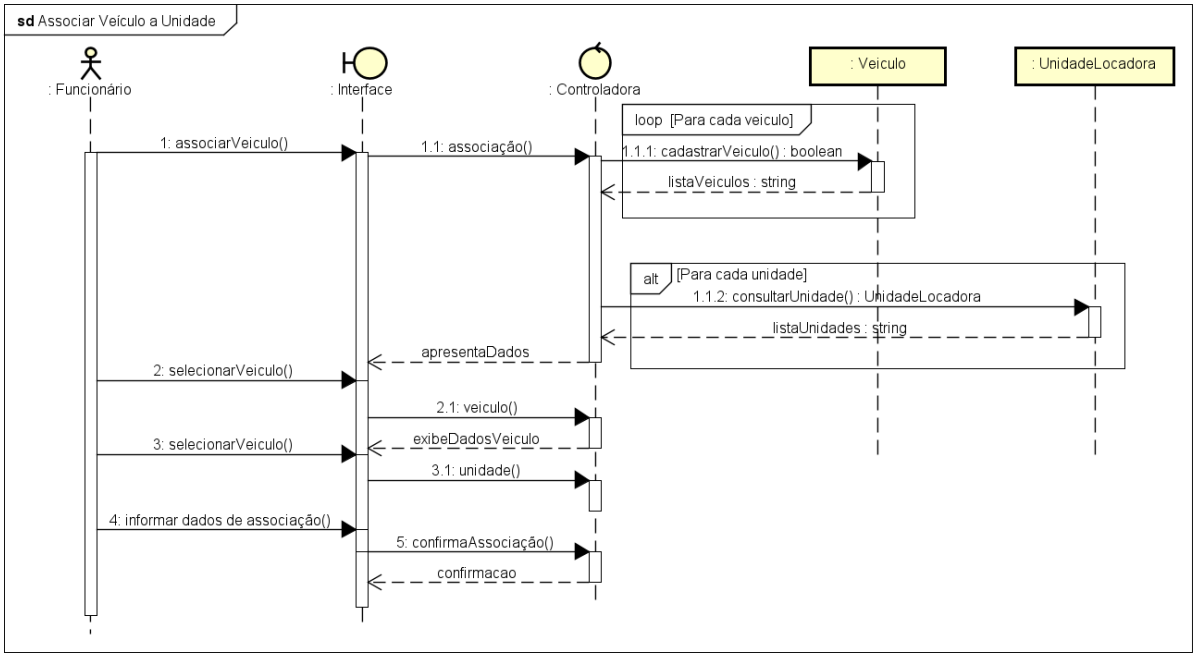
9. Diagrama de Sequência

Um diagrama de sequência é uma ferramenta de modelagem da UML (Unified Modeling Language - Linguagem de Modelagem Unificada) que representa a interação entre objetos em um sistema ao longo do tempo. Ele destaca a ordem das mensagens trocadas entre os objetos e a execução dos métodos.

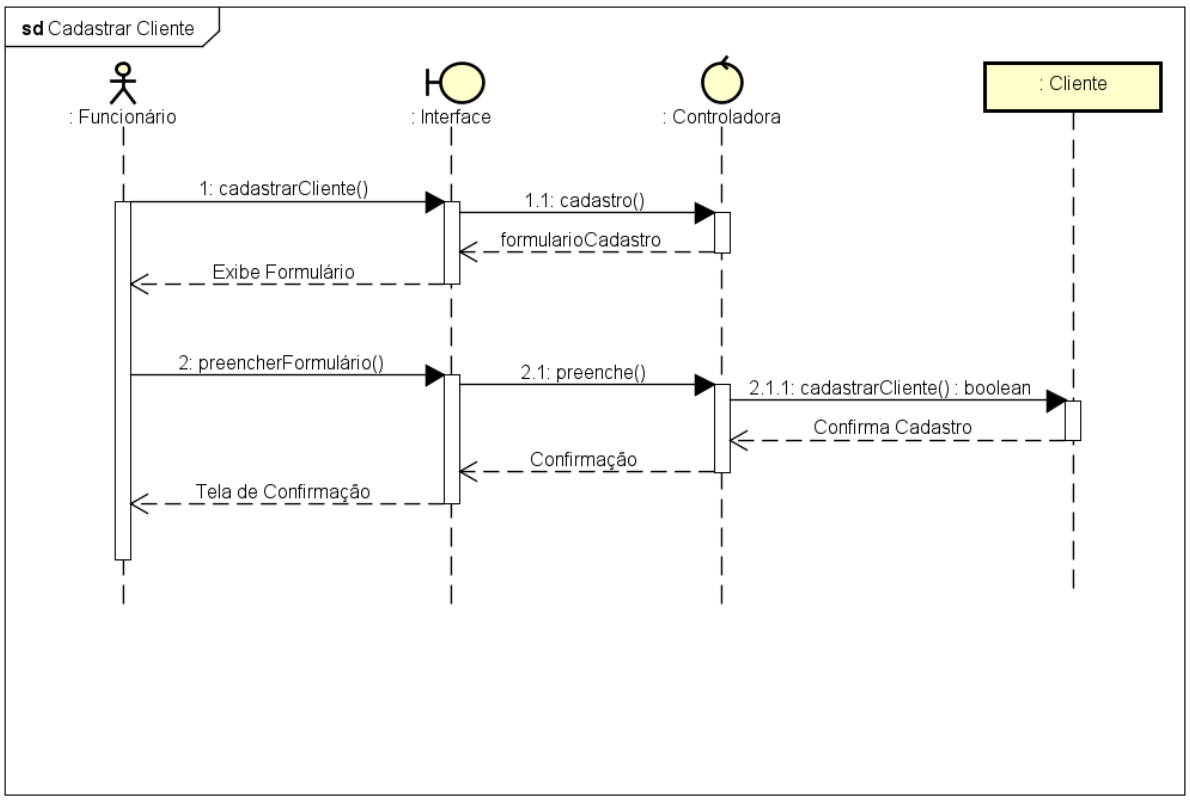
No diagrama de sequência, os objetos são representados por retângulos verticais, e as mensagens entre eles são mostradas por setas horizontais. A ordem das mensagens reflete a ordem cronológica em que as interações ocorrem. Além disso, é possível indicar a ativação e desativação de objetos ao longo do tempo.

Este tipo de diagrama é frequentemente utilizado durante a fase de design de sistemas para visualizar e compreender as interações entre diferentes partes do sistema. Ele é especialmente útil para representar cenários complexos e entender o fluxo de controle entre os objetos.

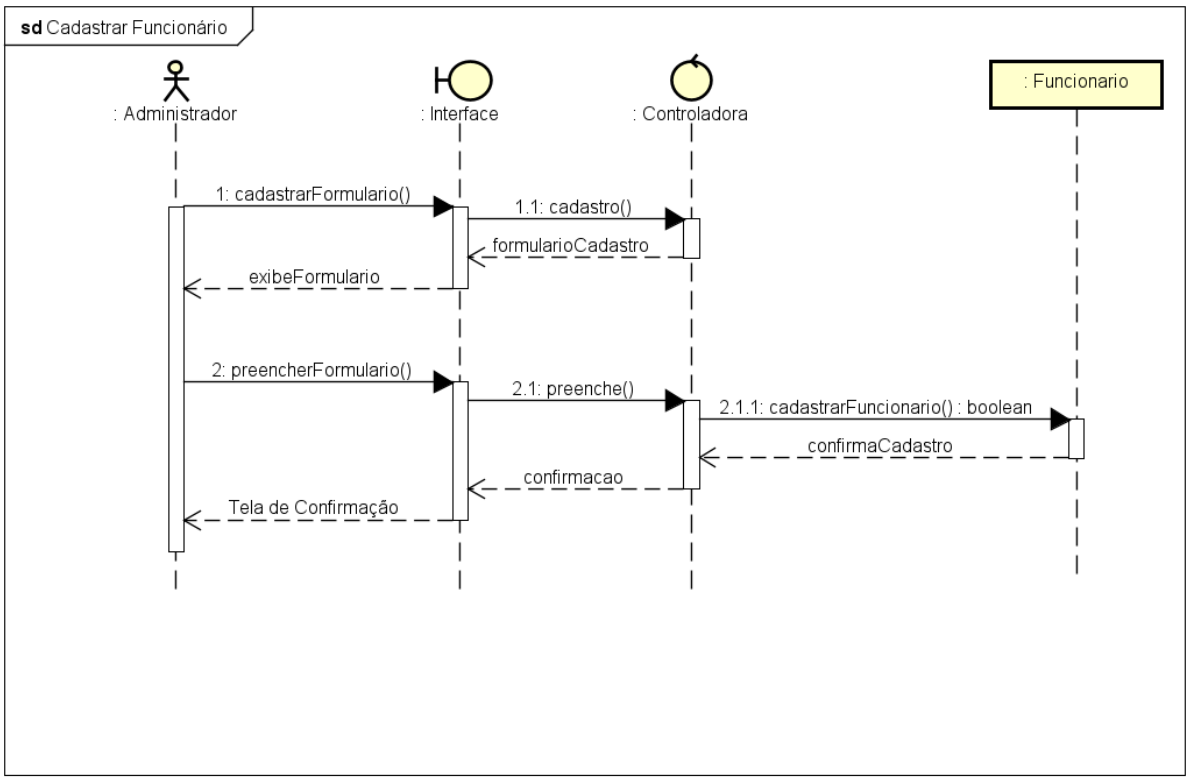
9.1. Associar Veículo a Unidade



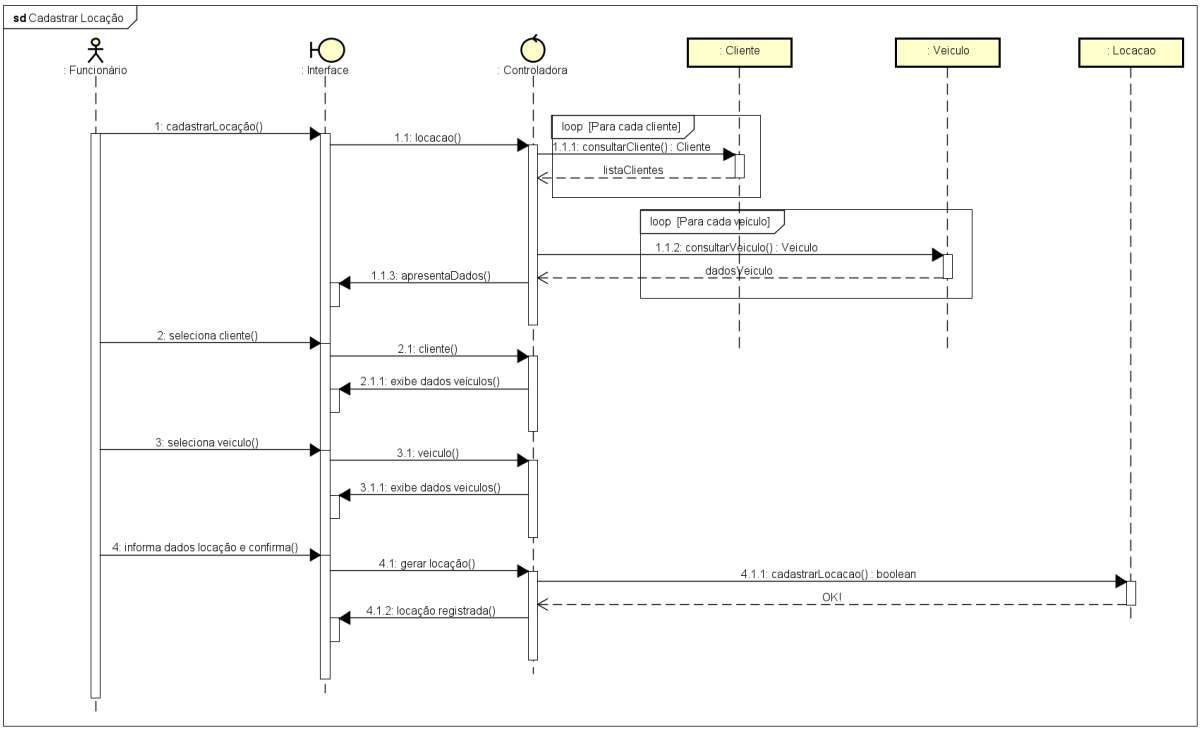
9.2. Cadastrar Cliente



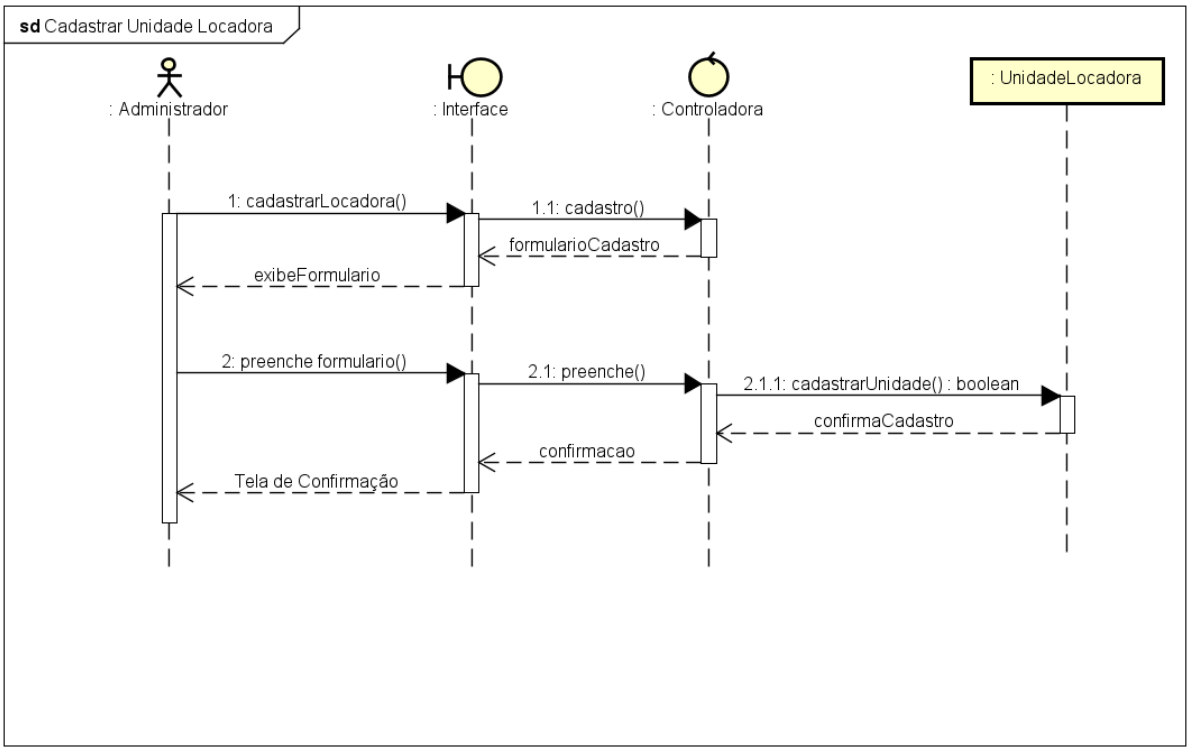
9.3. Cadastrar Funcionário



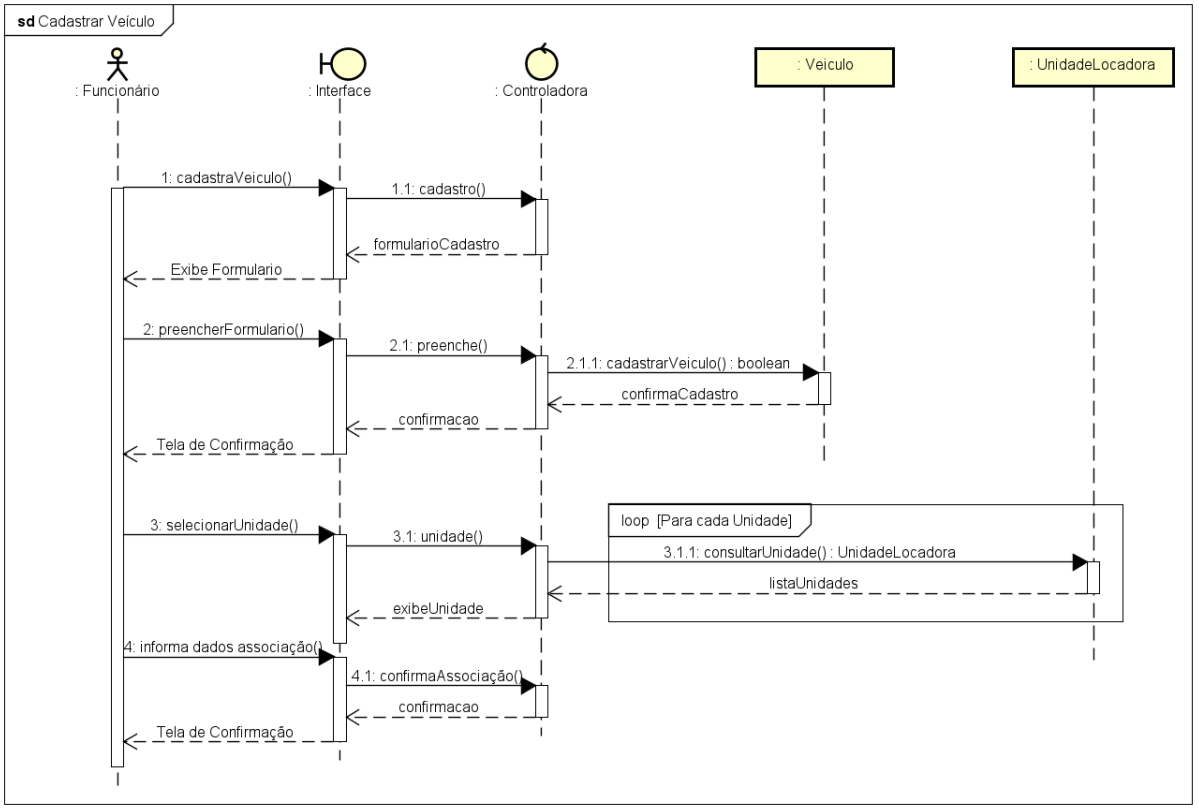
9.4. Cadastrar Locação



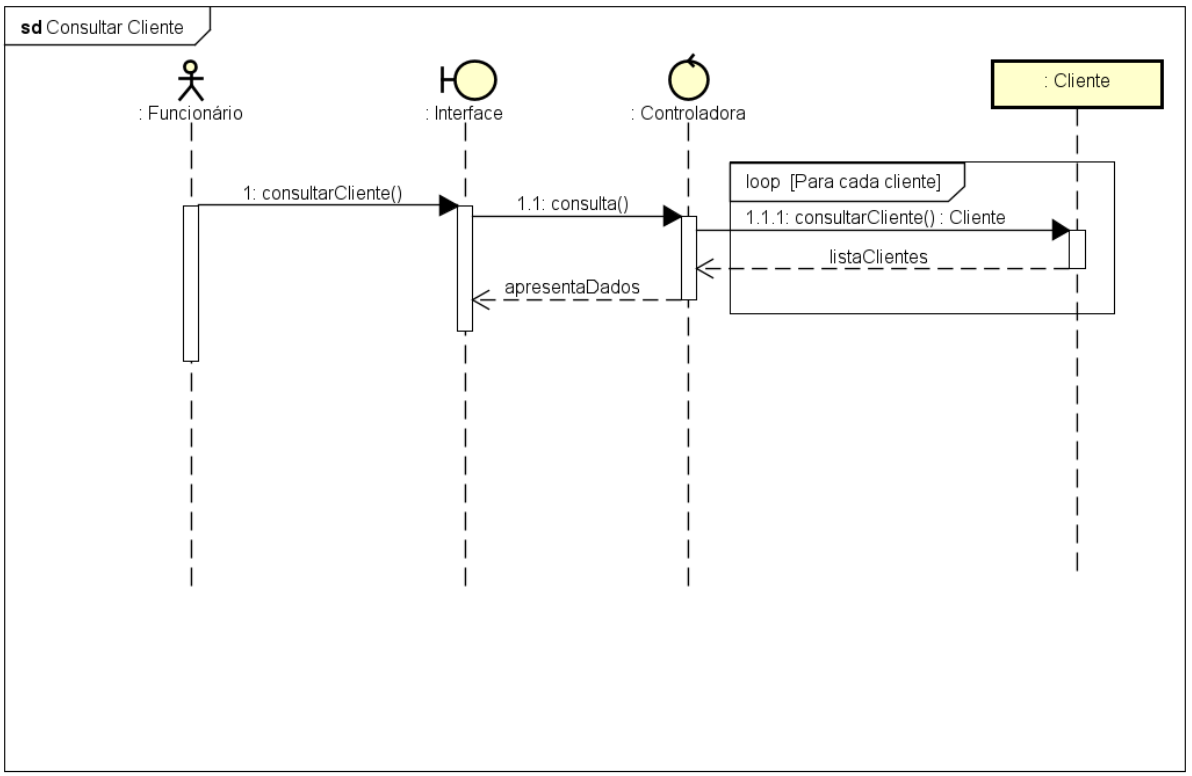
9.5. Cadastrar Unidade Locadora



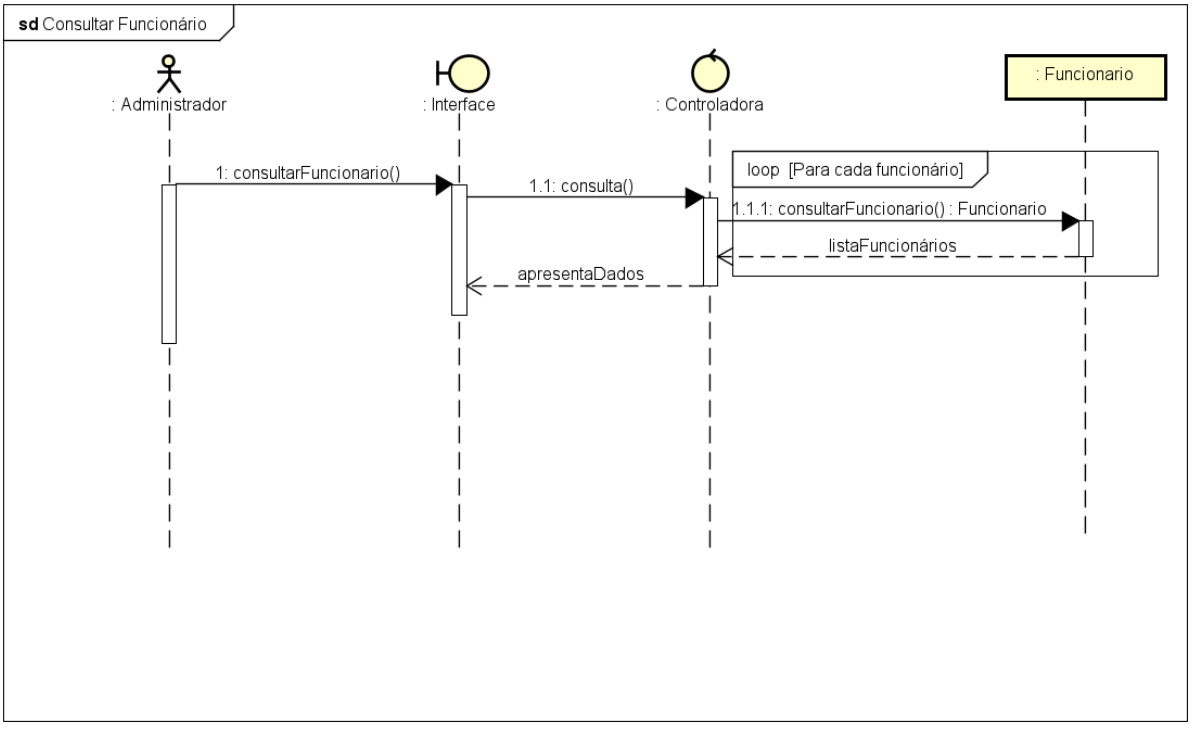
9.6. Cadastrar Veículo



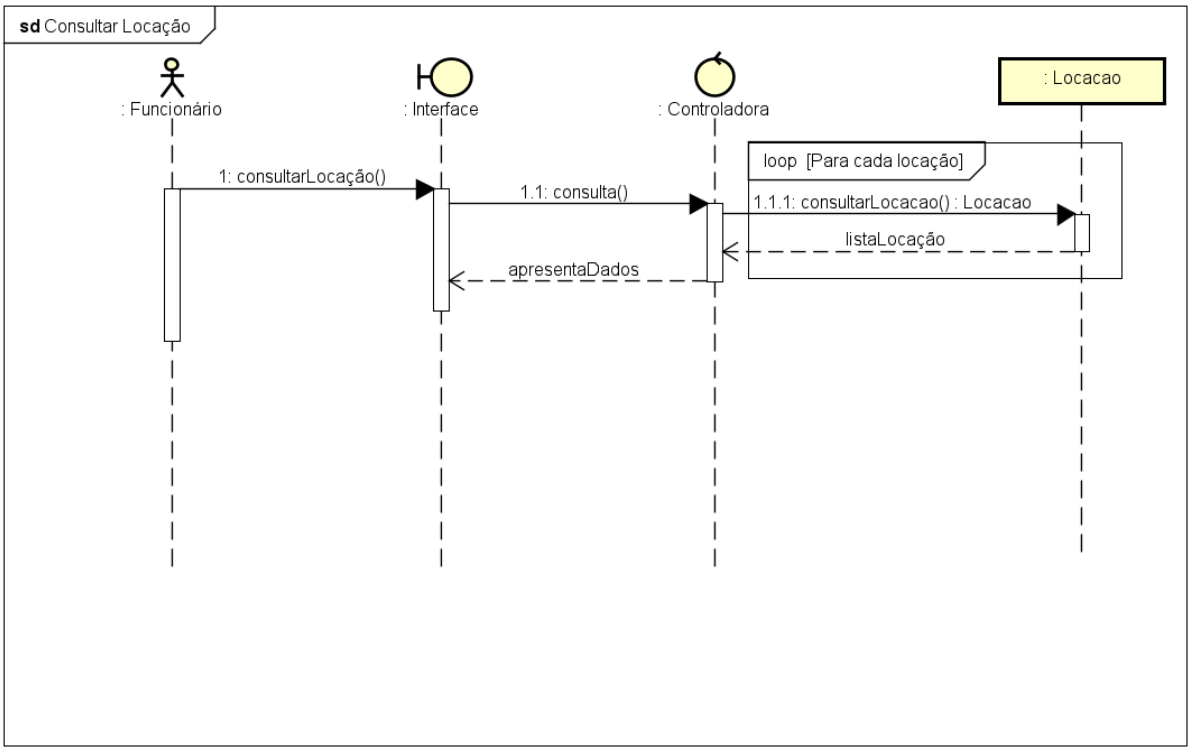
9.7. Consultar Cliente



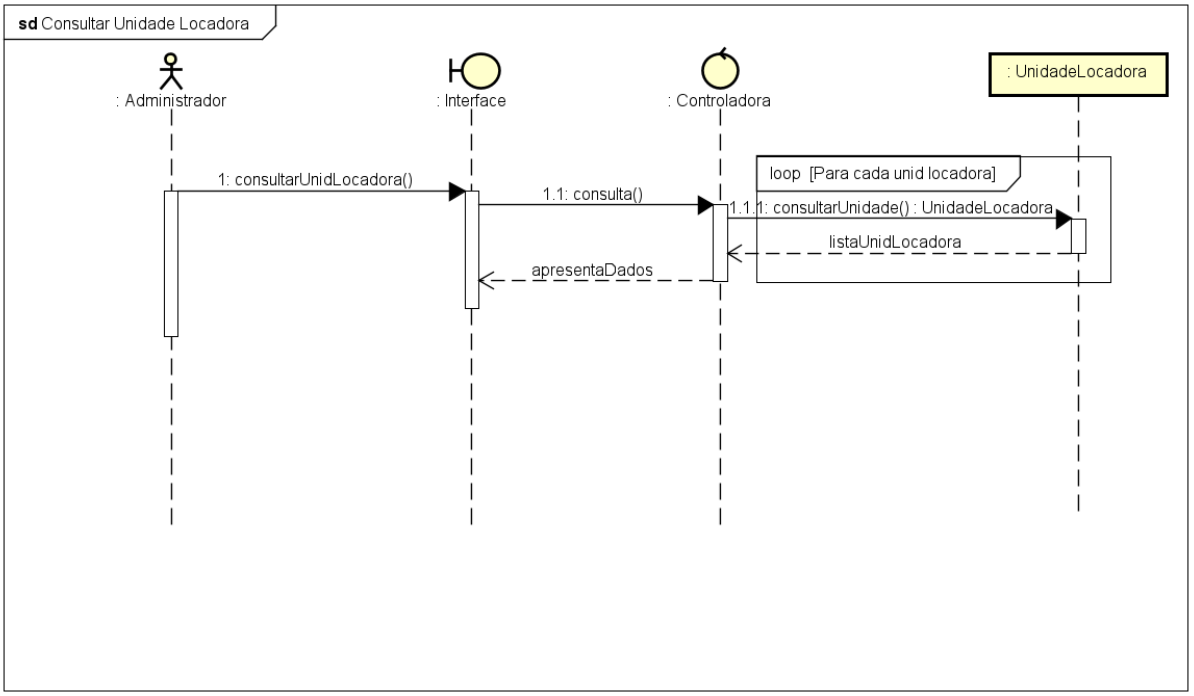
9.8. Consultar Funcionário



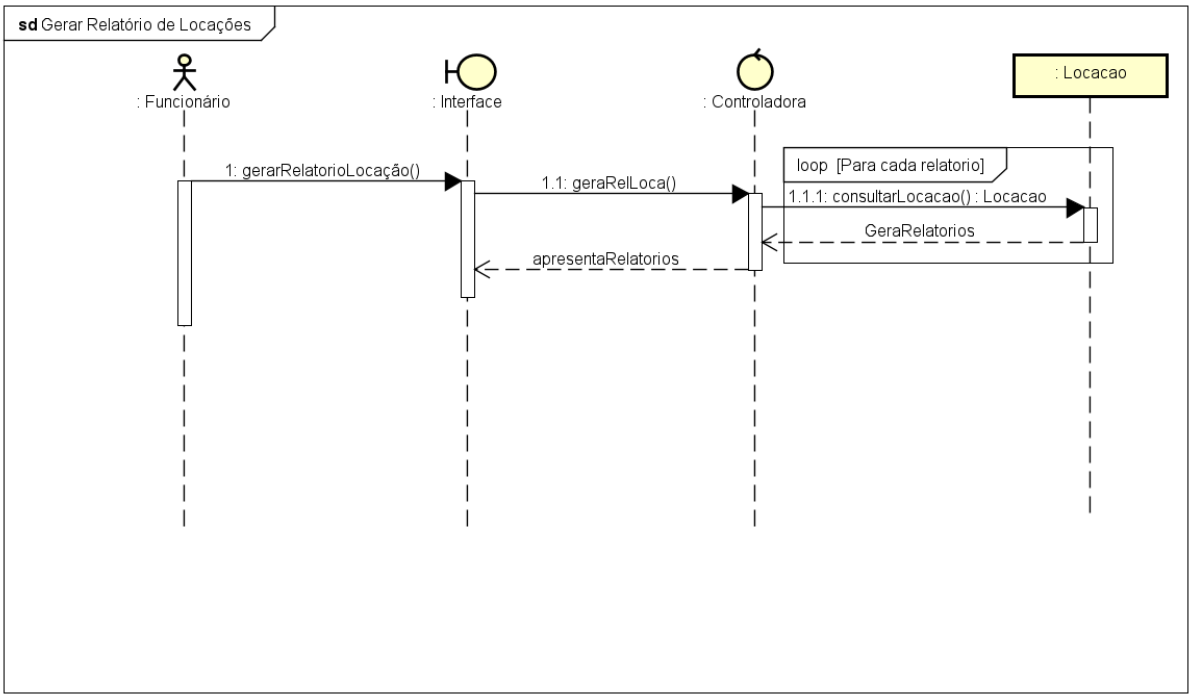
9.9. Consultar Locação



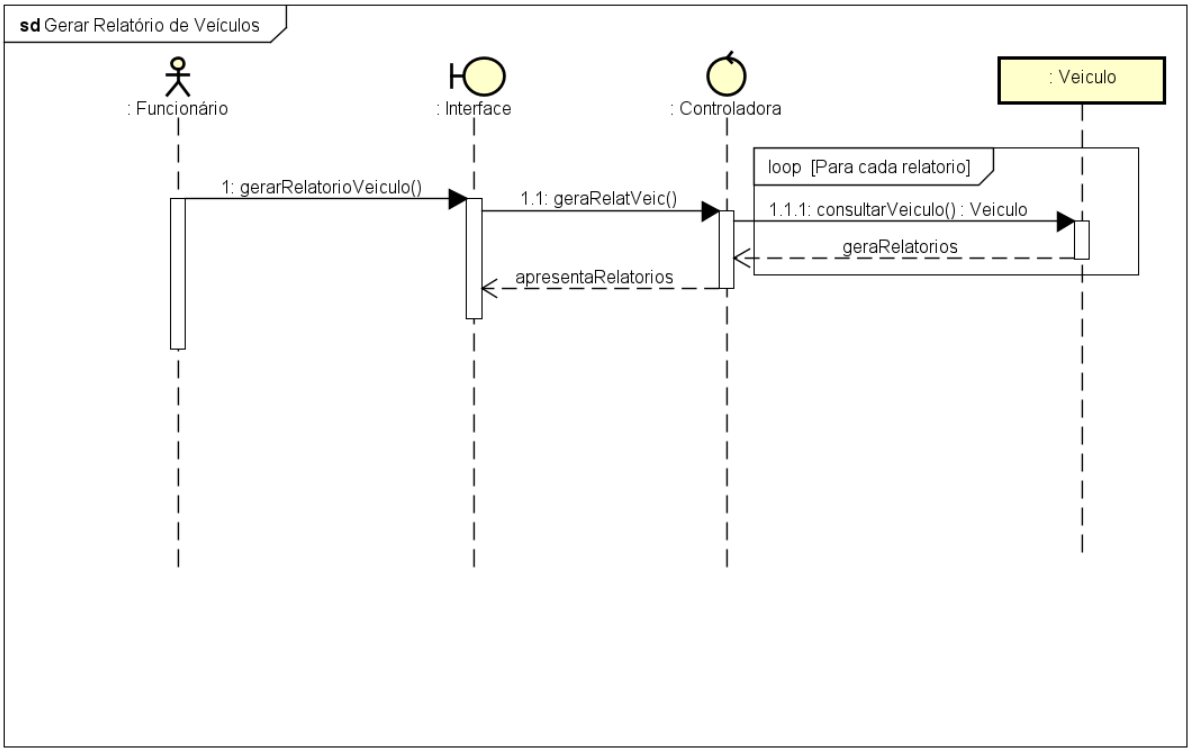
9.10. Consultar Unidade Locadora



9.11. Gerar Relatório de Locação



9.12. Gerar Relatório de Veículos



10. Tecnologias de implementação

10.1. Laravel

O Laravel é um framework PHP que segue o padrão de arquitetura MVC (Model-View-Controller). O MVC é um padrão de design que organiza o código-fonte de uma aplicação em três componentes principais: *Model*, *View* e *Controller*. Esses componentes ajudam a separar as preocupações e melhoram a manutenção, escalabilidade e legibilidade do código.

10.2. MySQL

O Laravel utiliza o Eloquent ORM (Object-Relational Mapping) para interagir com bancos de dados MySQL e fornecer uma camada de abstração que simplifica operações de banco de dados. Cada modelo no Laravel corresponde a uma tabela no banco de dados, e as instâncias desses modelos representam registros individuais na tabela. O Eloquent também oferece métodos para realizar operações comuns no banco de dados, como buscar registros, criar novos registros, atualizar registros existentes e excluir registros.

10.3. Bootstrap

Framework front-end de código aberto desenvolvido pela equipe do Twitter. Ele fornece um conjunto de ferramentas para o desenvolvimento de interfaces web responsivas e estilizadas rapidamente. O Bootstrap inclui um conjunto abrangente de estilos CSS que garantem uma aparência consistente em todos os elementos. Além disso, é possível personalizar a aparência do Bootstrap usando variáveis SASS ou LESS

10.4. GIT

O Git é um sistema de controle de versão distribuído, o que significa que um clone local do projeto é um repositório de controle de versão completo. Esses repositórios locais totalmente funcionais facilitam o trabalho offline ou remoto. Esse paradigma é diferente do controle de versão centralizado, no qual os clientes devem sincronizar o código com um servidor antes de criar novas versões do código.

Conclusão

Em suma este trabalho abrangente sobre o processo de desenvolvimento de software e as diferentes ferramentas de modelagem utilizadas, fica evidente a importância de uma abordagem estruturada e detalhada para garantir o sucesso na construção de sistemas de alta qualidade. Ao levantar os requisitos funcionais e não funcionais, identificamos as necessidades essenciais do sistema, garantindo que as expectativas dos usuários sejam atendidas e que as características fundamentais, como desempenho, segurança e usabilidade, sejam devidamente consideradas.

Os casos de uso fornecem uma boa visão dos diferentes cenários de interação entre usuários e sistema, enquanto o diagrama de classes representa a estrutura estática do sistema, mostrando as relações entre as entidades principais. Os diagramas de atividades e de sequência complementam essa visão ao destacar o fluxo de atividades e interações temporais, respectivamente.

Em conjunto, essas ferramentas de modelagem oferecem uma base sólida para o entendimento e comunicação entre as equipes de desenvolvimento, analistas de sistemas e stakeholders. A análise de requisitos, a representação visual das classes e interações, juntamente com os fluxos de atividades, contribuem para um processo mais transparente e eficiente. A utilização dessas técnicas não apenas facilita a compreensão do sistema, mas também serve como guia valioso ao longo de todas as fases do desenvolvimento, desde a concepção até a implementação.

Concluímos, portanto, que a combinação cuidadosa dessas ferramentas é essencial para o desenvolvimento de software bem-sucedido, proporcionando uma base sólida para a criação de sistemas que atendam não apenas aos requisitos declarados, mas também às expectativas e necessidades em constante evolução dos usuários e do ambiente em que o software será utilizado.

Referências

1. IBM. Diagramas de caso de uso. IBM Rational Software Modeler, Versão 7.5.0. Disponível em: https://www.ibm.com/docs/pt-br/rsm/7.5.0?topic=diagrams-use-case. Acesso em: 15 de novembro de 2023.

2. IBM. Diagramas de Classe. IBM Rational Software Modeler, Versão 7.5.0. Disponível em: https://www.ibm.com/docs/pt-br/rsas/7.5.0?topic=structure-class-diagrams. Acesso em: 15 de novembro de 2023.

3. IBM. Diagramas de Atividades. IBM Rational Software Modeler, Versão 7.5.0. Disponível em: https://www.ibm.com/docs/pt-br/rational-soft-arch/9.7.0?topic=diagrams-activity. Acesso em: 15 de novembro de 2023.

4. IBM. Diagramas de Sequência. IBM Rational Software Modeler, Versão 7.5.0. Disponível em: https://www.ibm.com/docs/pt-br/rsm/7.5.0?topic=uml-sequence-diagrams. Acesso em: 15 de novembro de 2023.
5. Microsoft. Learn/DevOps/O que é o Git.
   Disponível em:
   https://learn.microsoft.com/pt-br/devops/develop/git/what-is-git. Acesso em: 15 de novembro de 2023.