Documentação de Projeto

para o sistema

<AutoClick>

Versão 1.0

Projeto de sistema elaborado pelo(s) aluno(s)
Lucas Giovine, Thiago Cury, Vitor Rebula e Pedro Porto
como parte da disciplina **Projeto de Software**.

<21/05/2025>

Tabela de Conteúdo

1. Introdução **1**
2. Modelos de Usuário e Requisitos **1**
2.1 Descrição de Atores 1
2.2 Modelo de Casos de Uso e Histórias de Usuários 1
2.3 Modelagem funcional com contratos 1
3. Modelos de Projeto **1**
3.1 Arquitetura 1
3.2 Diagrama de Componentes e Implantação. 2
3.3 Diagrama de Classes 2
3.4 Diagramas de Sequência 2
3.5 Diagramas de Comunicação 2
3.6 Diagramas de Estados 2
4. Modelos de Dados **2**

Histórico de Revisões

Nome	Data	Razões para Mudança	Versão
Adicionando ID nos casos de uso	22/05/2025	Ausência de ID's para cada caso de uso	1.1.0

1. Introdução

Este documento agrega: 1) a elaboração e revisão de modelos de domínio e 2) modelos de projeto para o sistema <AutoClick>. A referência principal para a descrição geral do problema, domínio e requisitos do sistema é o documento de especificação que descreve a visão de domínio do sistema.

2. Modelos de Usuário e Requisitos

2.1 Descrição de Atores

O sistema AutoClick possui três atores principais:

- Gerente**:** usuário responsável pela gestão e administração do sistema. Possui permissões para cadastrar e gerenciar usuários (vendedores e clientes) e veículos, além de ter acesso a relatórios e indicadores de desempenho.
- Vendedor**:** funcionário que atua diretamente na operação de vendas. Tem como funções principais o cadastro de veículos, realizar vendas e gerar relatórios.
- Cliente**:** usuário que acessa o sistema com o objetivo de buscar veículos disponíveis, visualizar informações detalhadas dos veículos, realizar propostas de compra e acompanhar o progresso das suas negociações.

2.2 Modelo de Casos de Uso e Histórias de Usuários

Nesta subseção é apresentado o diagrama de casos de uso do sistema, bem como histórias de usuário que refletiram na criação dos mesmos.

Cliente

- *Como cliente, desejo visualizar os veículos disponíveis no sistema para conhecer as opções antes de decidir uma compra.*
- *Como cliente, desejo filtrar veículos por modelo, ano, preço e marca para encontrar facilmente um veículo que atenda às minhas necessidades.*
- *Como cliente, desejo realizar a compra de um veículo pelo sistema para agilizar meu processo de aquisição sem precisar ir até a loja.*

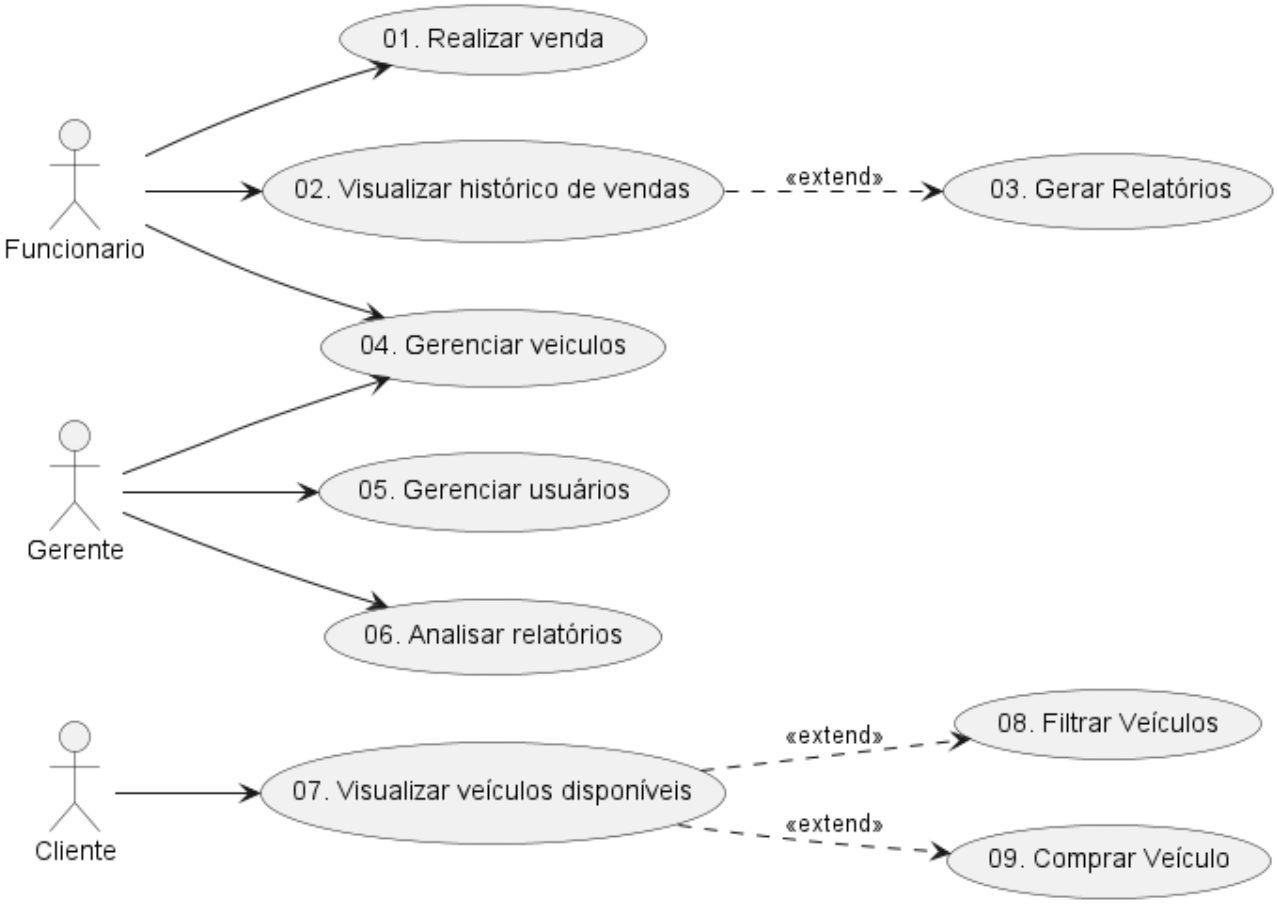
Funcionário

- *Como funcionário, desejo registrar as vendas realizadas no sistema para garantir que as negociações fiquem devidamente armazenadas e organizadas.*
- *Como funcionário, desejo gerenciar o catálogo de veículos, podendo adicionar, editar ou remover veículos, para manter a base de dados atualizada.*
- *Como funcionário, desejo acessar o histórico de vendas para acompanhar vendas anteriores e oferecer suporte aos clientes.*

- *Como funcionário, desejo gerar relatórios de vendas e desempenho para acompanhar meus resultados e melhorar minha performance profissional.*

Gerente

- *Como gerente, desejo gerenciar os usuários do sistema, podendo cadastrar, editar e remover funcionários, para manter o controle de acesso e a organização da equipe.*
- *Como gerente, desejo acessar relatórios de vendas e desempenho da empresa para ter uma visão geral dos resultados e tomar decisões estratégicas.*
- *Como gerente, desejo gerenciar o catálogo de veículos, assim como os funcionários, para garantir que todas as informações estejam corretas e atualizadas.*



2.3 Modelagem funcional com contratos

Nesta subseção é apresentado a modelagem funcional com contratos de 3 dos principais Casos de Uso, descritos na Seção 2.3, em formato para cada contrato de operação.

Contrato	Realizar venda
Operação	realizarVenda(clienteId, veiculoId, funcionarioId)
Referências cruzadas	UC01 – Realizar Venda
Pré-condições	cliente != null veiculo != null veiculo.status == "disponível" funcionario != null
Pós-condições	venda != null veiculo.status == "vendido" vendas.contains(venda) == true

Contrato	Listar veículos
Operação	listarVeiculosDisponiveis(filtros)
Referências cruzadas	UC07 – Visualizar Veículos Disponíveis, UC08 – Filtrar Veículos
Pré-condições	veiculos.size > 0
Pós-condições	resultado = veiculos.filter(filtros) resultado != null

Contrato	Gerenciar veículo
Operação	gerenciarVeiculo(acao, dadosVeiculo)
Referências cruzadas	UC04 – Gerenciar Veículos
Pré-condições	funcionario != null OR gerente != null (acao == "editar" OR acao == "excluir") ⇒ veiculo != null
Pós-condições	(acao == "cadastrar") ⇒ veiculos.contains(dadosVeiculo) == true (acao == "editar") ⇒ veiculo.atualizado == true (acao == "excluir") ⇒ veiculos.contains(veiculo) == false

3. Modelos de Projeto

3.1 Arquitetura

O sistema foi desenvolvido utilizando o padrão arquitetural MVC (Model-View-Controller), amplamente adotado em aplicações web com o framework Spring Boot. A divisão de responsabilidades é feita da seguinte forma:

- Model: Responsável pela representação dos dados e das regras de negócio. Aqui estão as entidades (models) que mapeiam as tabelas do banco de dados e os objetos de transferência de dados (DTOs) que são utilizados para otimizar a comunicação entre as camadas.

- Controller: Recebe as requisições HTTP, processa os dados de entrada, valida informações e repassa as ações para a camada de serviço ou diretamente para a camada de negócio. É responsável por expor os endpoints da API REST.

- Service: Contém as regras de negócio e orquestra as operações entre Controller, Repository e outros serviços.

- Repository (DAO): Camada responsável pelo acesso aos dados. No Spring Boot, utiliza-se o Spring Data JPA, que fornece abstrações para operações CRUD e consultas no banco de dados, eliminando a necessidade de implementação manual dos DAOs tradicionais.

Adicionalmente, são utilizados DTOs (Data Transfer Objects) para encapsular os dados que transitam entre cliente e servidor, separando a representação interna das entidades do modelo de dados exposto via API.

3.2 Diagrama de Componentes e Implantação.

Diagramas de componentes do sistema. Diagrama de implantação mostrando onde os componentes estarão alocados para a execução.

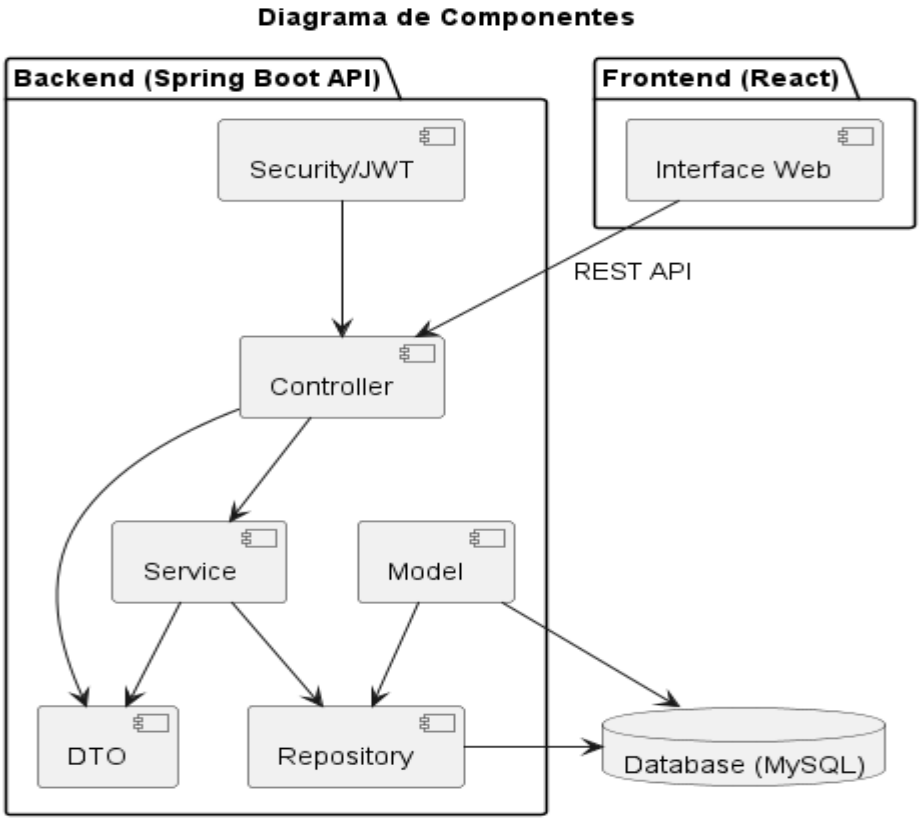
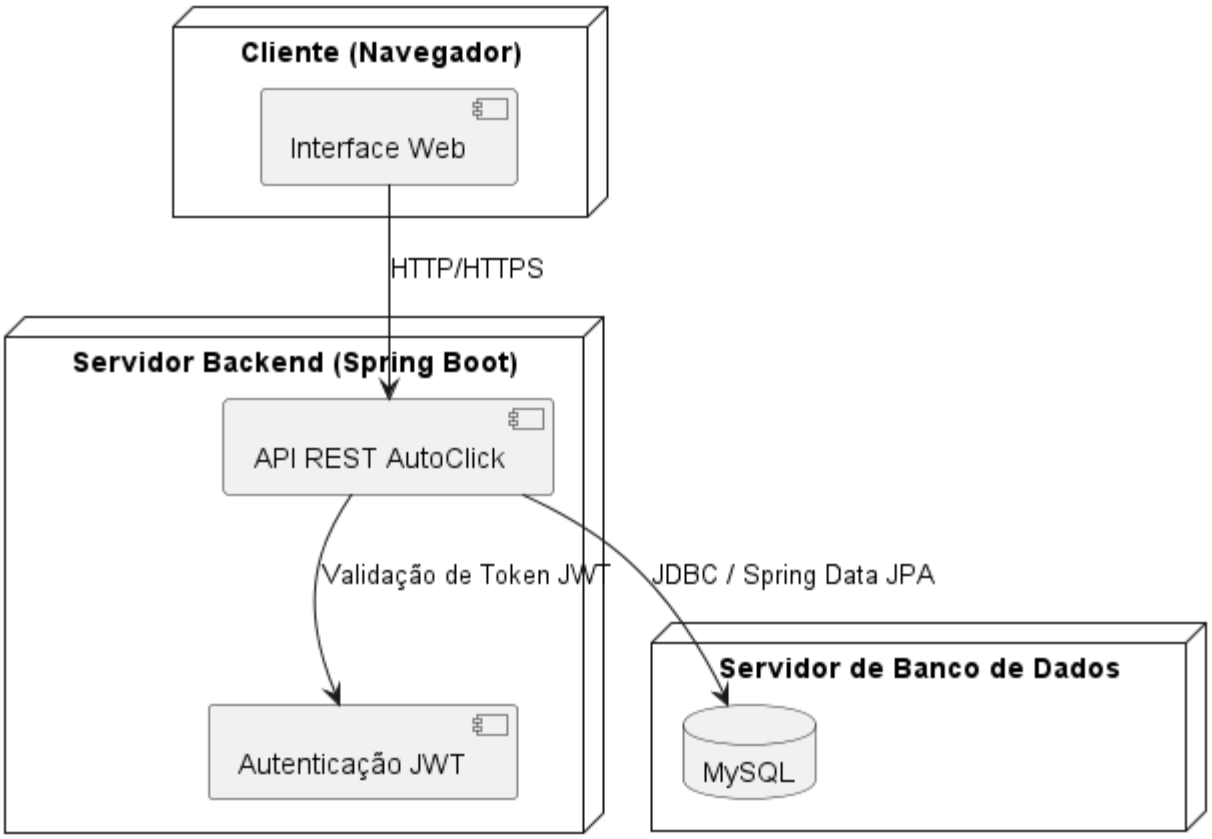
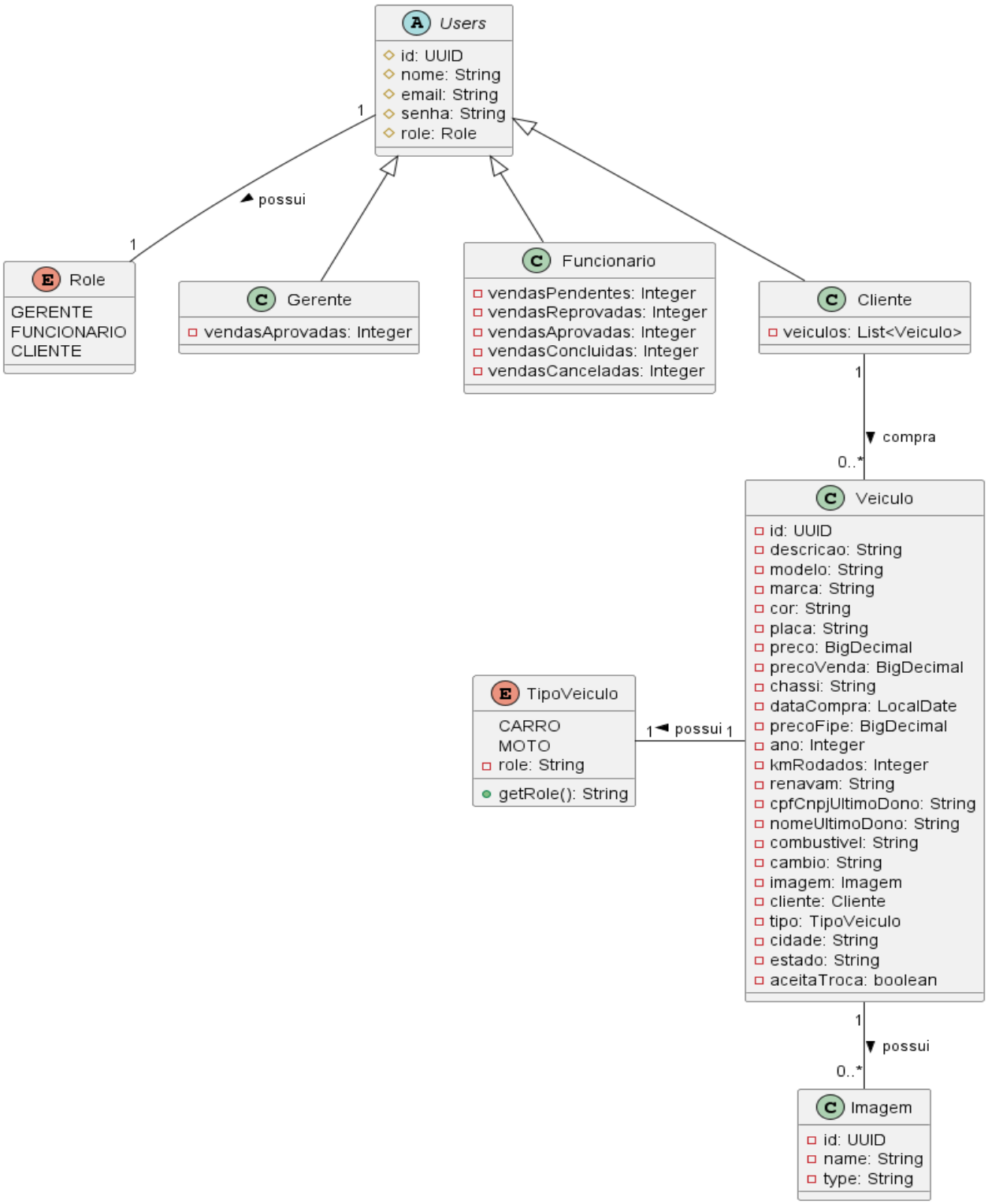


Diagrama de Implantação



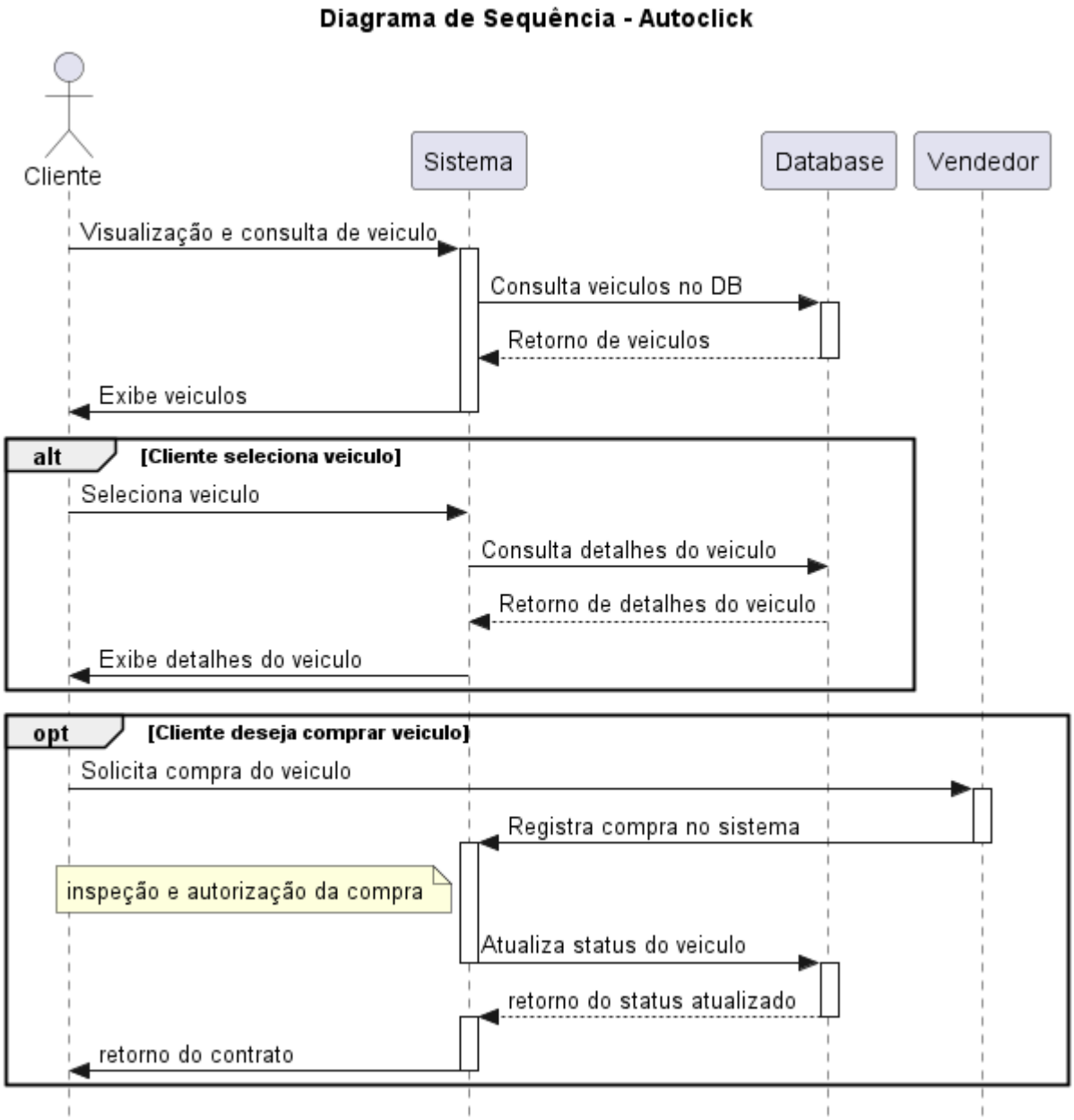
3.3 Diagrama de Classes

Diagrama de classes do sistema, apresentando as Classes e estruturas modeladas para implementação da solução.



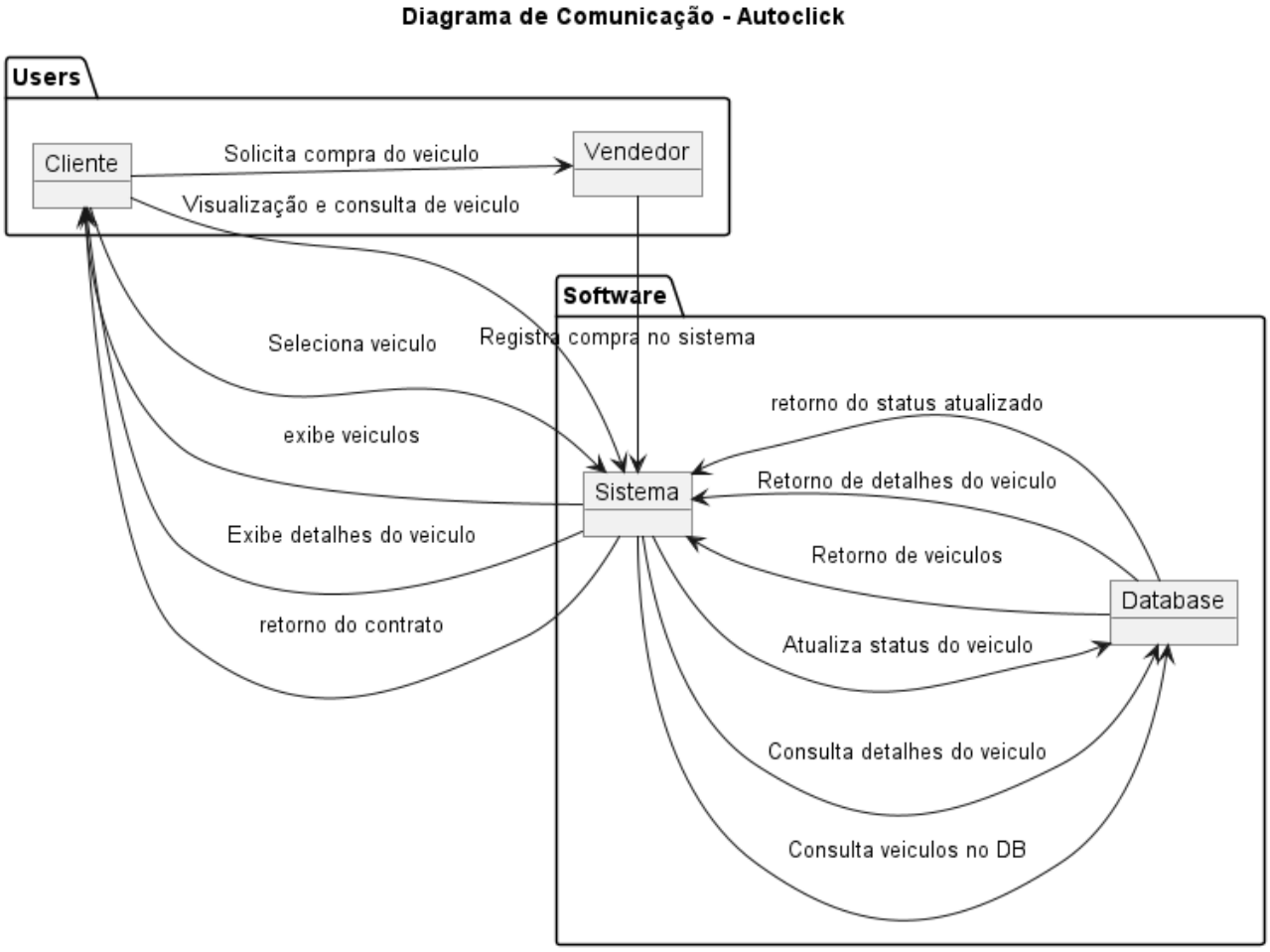
3.4 Diagramas de Sequência

Diagramas de sequência para realização de casos de uso.



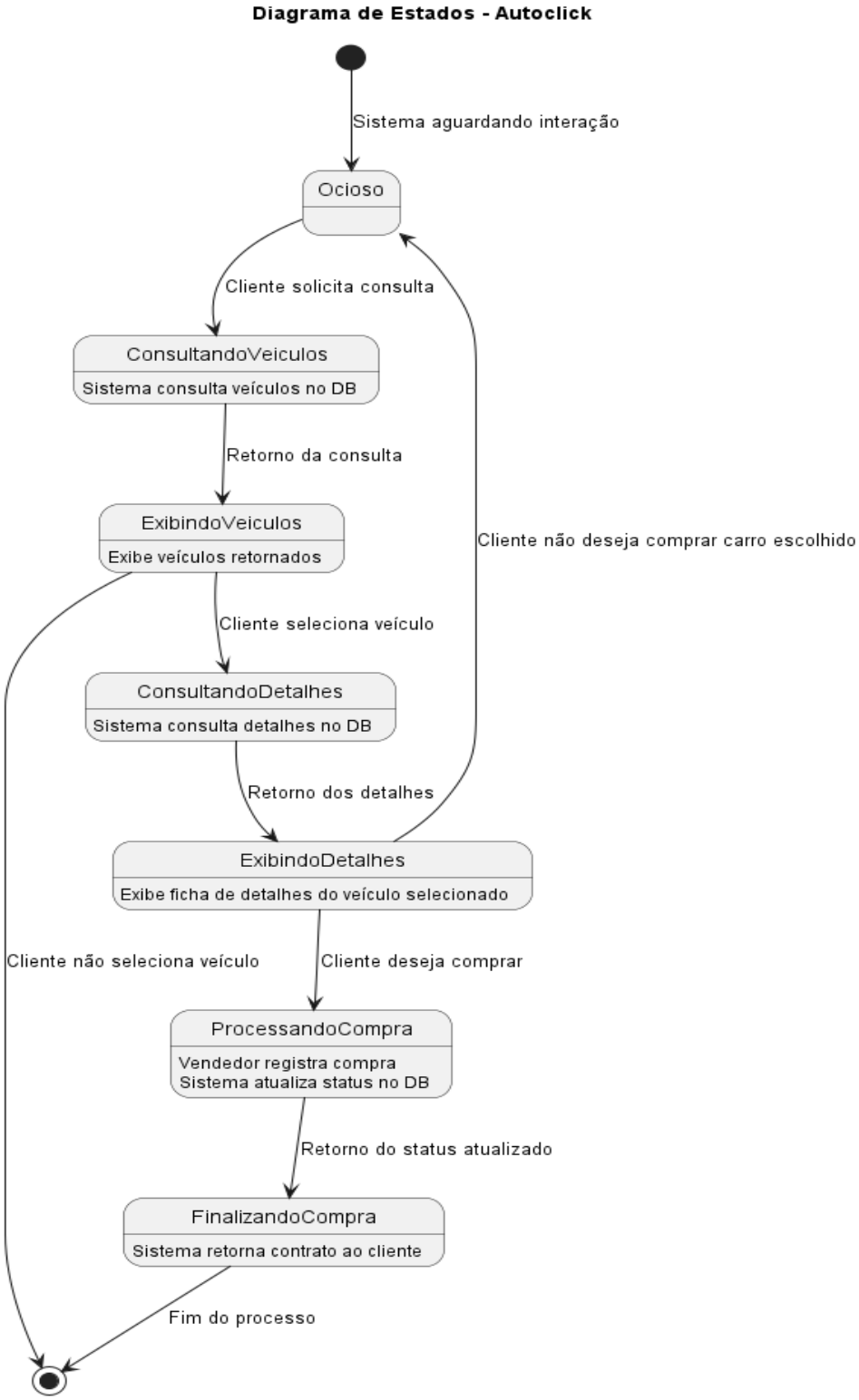
3.5 Diagramas de Comunicação

Diagramas de comunicação para realização de casos de uso.



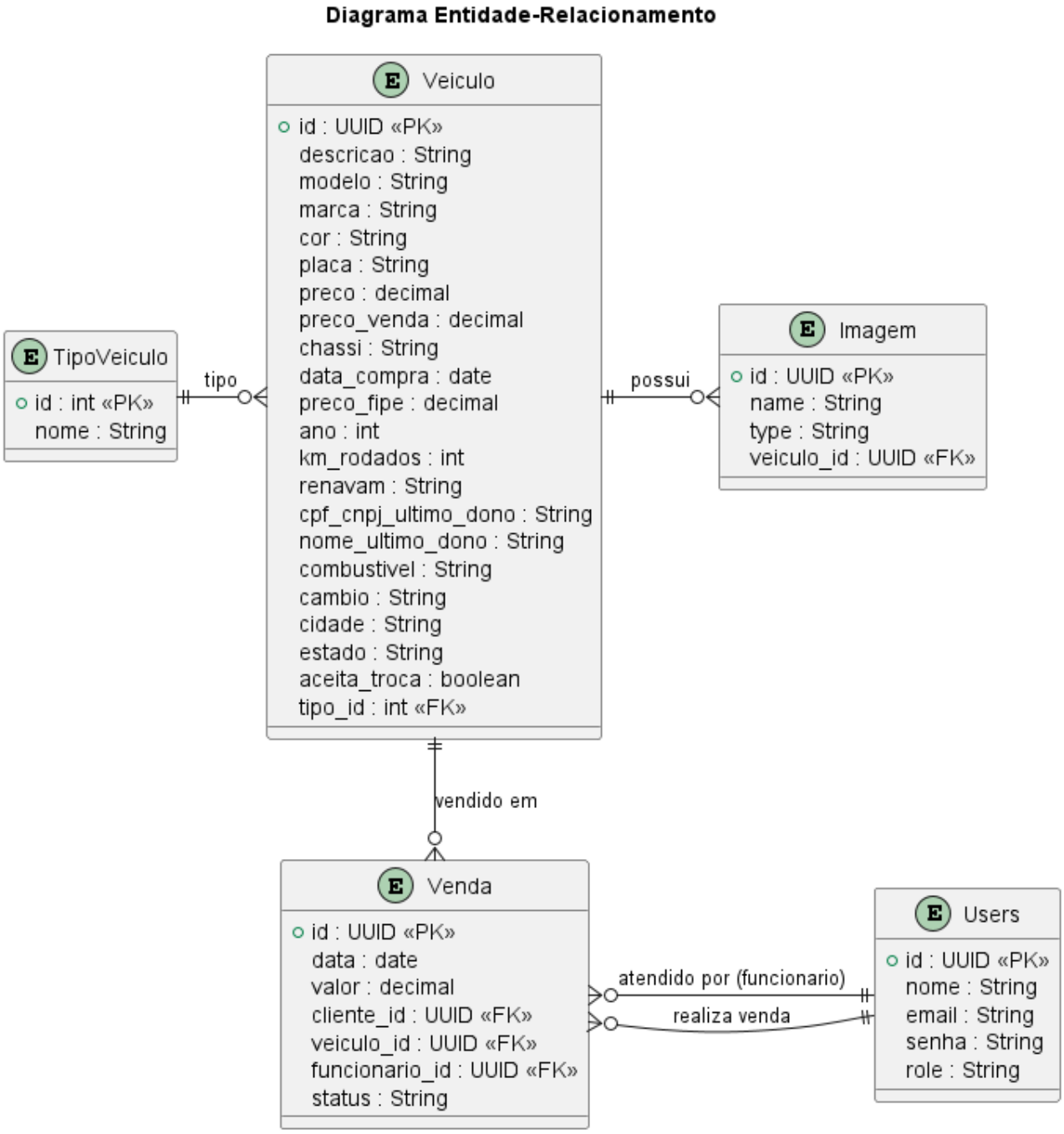
3.6 Diagramas de Estados

Diagramas de estados do sistema.



4. Modelos de Dados

Abaixo segue o diagrama ER (Entidade-Relacionamento) e uma breve explicação do mapeamento de entidades utilizado.



O sistema utiliza JPA/Hibernate para mapeamento objeto-relacional. A herança entre Users, Gerente, Funcionario e Cliente é implementada com a estratégia SINGLE_TABLE, utilizando uma coluna role para diferenciar os tipos de usuários. As entidades possuem relacionamentos refletidos no banco

através de chaves estrangeiras (@ManyToOne, @OneToMany). Listas, como imagens associadas a veículos, são modeladas como relações 1:N. O enum TipoVeiculo é mapeado como uma tabela auxiliar de domínio. Todos os identificadores são UUIDs para garantir unicidade global.