RELATÓRIO TÉCNICO: SISTEMA DE GESTÃO LOGÍSTICA E TRANSPORTADORA

Equipe de Desenvolvimento:

- **Sérgio Eduardo**
- **Michel**
- **Igor**

1. INTRODUÇÃO

Este projeto visa o desenvolvimento de um sistema de software robusto para a gestão de uma transportadora. O foco principal da solução é agilizar o processo de pedidos, prover rastreamento de entregas em tempo real e otimizar o gerenciamento de frota.

A solução foi desenvolvida utilizando a linguagem Java, adotando a arquitetura MVC (Model-View-Controller) para garantir a separação de responsabilidades e a escalabilidade do código. Além disso, foram implementadas práticas rigorosas de segurança da informação, destacando-se a utilização de criptografia RSA para a proteção de dados sensíveis.

2. REQUISITOS DO SISTEMA

O escopo do projeto foi definido com base no levantamento de necessidades dos principais stakeholders: Cliente, Funcionário e Gerente. Abaixo estão listados os requisitos funcionais, não-funcionais e regras de negócio implementadas.

ID	Tipo	Descrição
RF001	Funcional	Rastreio da Entrega: O cliente deve visualizar o status do pedido em tempo real.
RF003	Funcional	Cadastro do Usuário: O sistema deve permitir o cadastro com validação de dados.
RF004	Funcional	Cadastro de Pedidos: Registro de novas solicitações de transporte.

ID	Tipo	Descrição
RF006	Funcional	Cálculo de Frete: Estimativa de valor baseada em parâmetros definidos.
RNF002	Não-Funcional	Autenticação: O sistema deve possuir login seguro.
RNF003	Não-Funcional	Segurança de Dados: Dados sensíveis (senhas) devem ser criptografados com algoritmo RSA.
RD001	Regra de Negócio	Regra de Carga: Impedir cadastro de cargas que excedam o limite legal do veículo (ex: VUC 3.000kg).

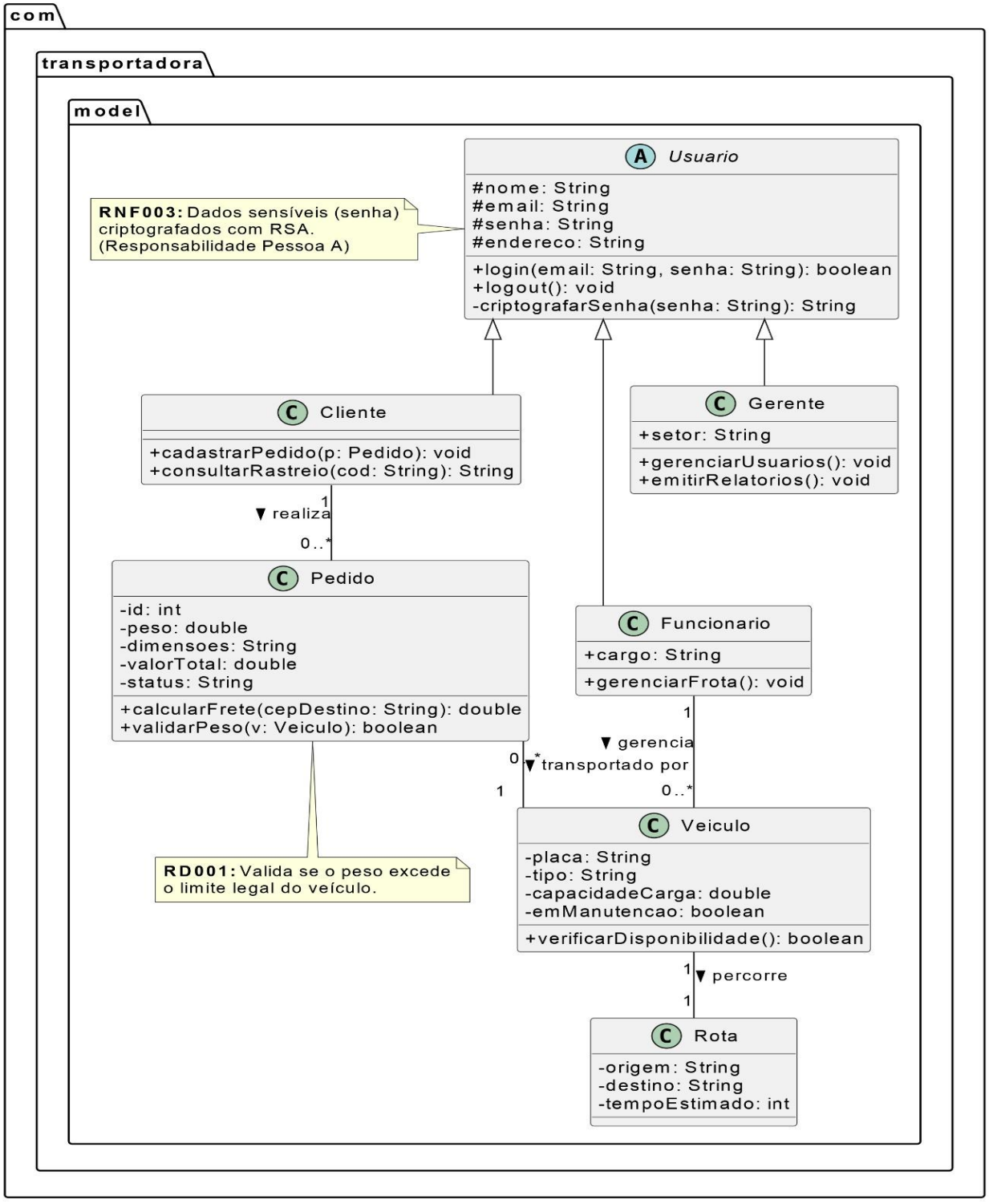
3. MODELAGEM E ARQUITETURA DO SISTEMA

A modelagem do sistema foi estruturada para garantir a integridade dos dados e a segurança das operações.

3.1. Diagrama de Classes Detalhado

O Diagrama de Classes representa a estrutura estática do sistema. Foi adotada uma estratégia de herança através da classe abstrata Usuario, centralizando os atributos comuns (Nome, Email, Senha) para os atores Cliente, Gerente e Funcionário.

Diagrama de Classes Detalhado - Projeto Transportadora



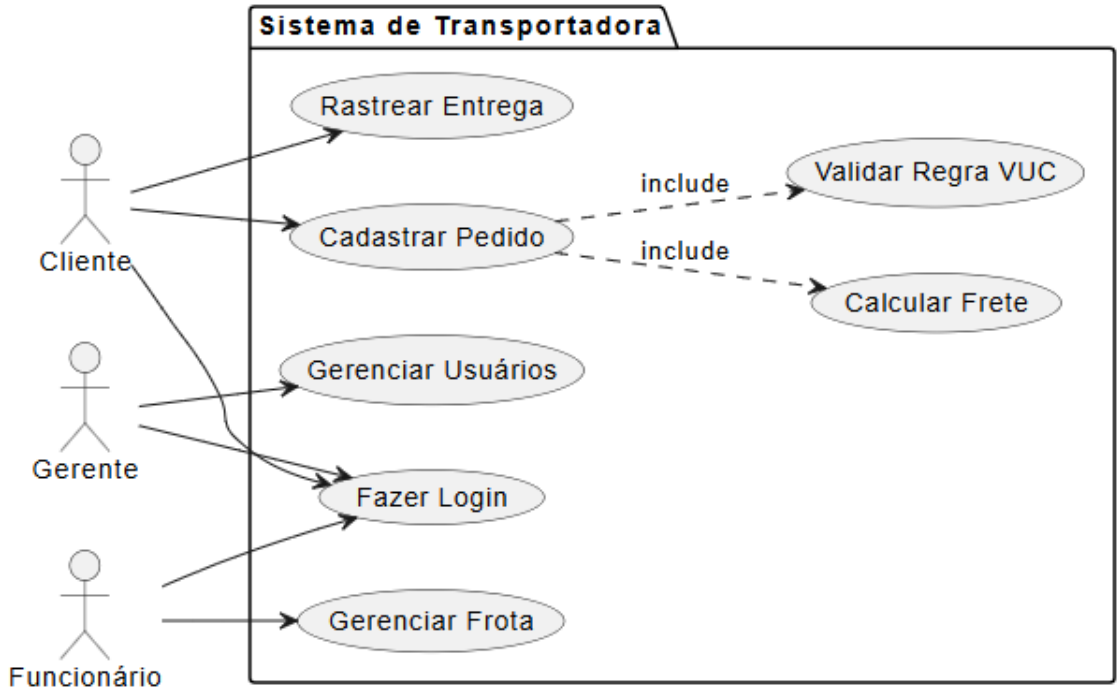
Decisões Arquiteturais:

- ***Segurança (RNF003): A classe Usuario encapsula o método de criptografia RSA, garantindo que a senha nunca transite ou seja armazenada em texto plano.***

- ***Validação de Negócio (RD001): As classes Pedido e Veiculo possuem uma associação direta (1 para 0..1). Isso é crucial para validar se o peso da carga é compatível com a capacidade do veículo alocado.***

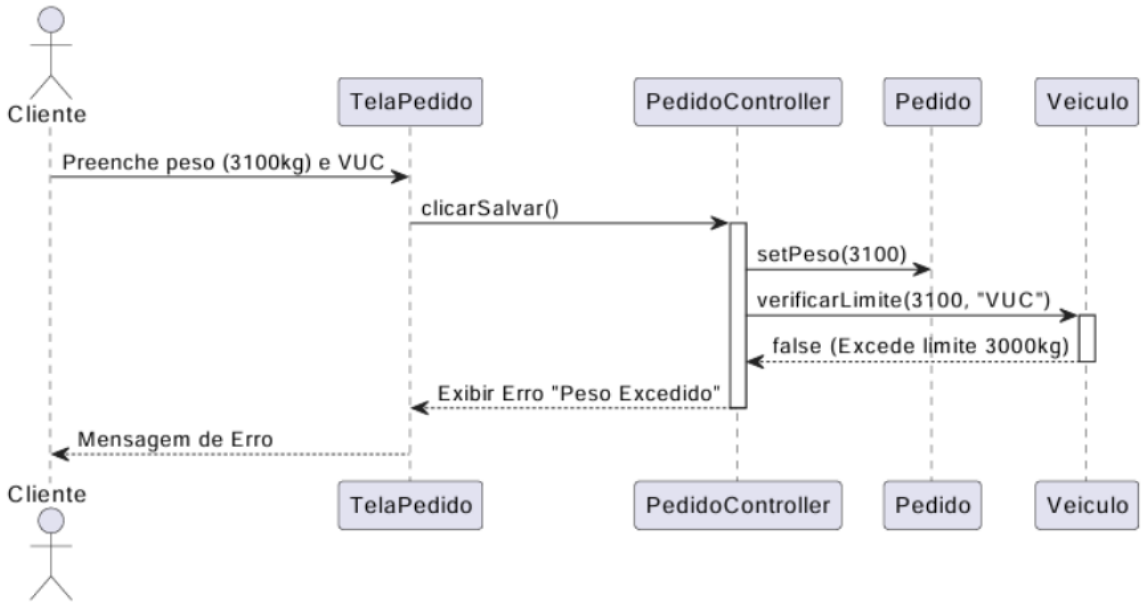
3.2. Diagrama de Casos de Uso (Módulo de Pedidos)

O Módulo de Pedidos é central para a operação logística. O diagrama abaixo destaca o processo de cadastro, que inclui duas funcionalidades essenciais via include: o cálculo de frete e a validação da regra de carga.



3.3. Diagrama de Sequência e Objetos

Para ilustrar o fluxo de interação entre o Cliente e o Sistema durante a criação de um pedido e a validação de regras, utilizou-se o diagrama de sequência abaixo:



4. IMPLEMENTAÇÃO DOS MÓDULOS

4.1. Módulo de Frota (Veículos e Rotas)

As classes Veículo e Rota formam a base operacional. A implementação garante que todo novo veículo inicie com o status DISPONIVEL e que sua capacidade de carga seja imutável após a criação, assegurando a confiabilidade para a regra de negócio RD001.

4.2. Dicionário de Dados – Classe Pedido

Abaixo, a estrutura de dados principal para o tráfego de informações de pedidos:

Atributo	Tipo de Dado	Descrição
id	int	Identificador único do pedido.
peso	double	Peso da carga em kg (validado pela RD001).
dimensoes	String	Dimensões físicas da carga.
valorTotal	double	Valor total somando nota fiscal e frete.

Atributo	Tipo de Dado	Descrição
status	String	Status atual do pedido (ex: Em Trânsito).

5. PLANO DE TESTES E EVIDÊNCIAS

Os testes unitários foram realizados utilizando o framework JUnit 4 e gerenciados via Maven. O objetivo foi validar tanto a lógica de negócio crítica quanto os requisitos de segurança.

5.1. Testes de Segurança (Autenticação e RSA)

- **Casos de Teste: CT-001 e CT-002.**
- **Objetivo: Verificar criptografia de senha e bloqueio de credenciais inválidas.**
- **Resultado: Os testes passaram com tempo de execução de 0.701s. O sistema comprovou criptografar a senha corretamente antes da comparação no banco de dados.**

```
-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running com.transportadora.model.LoginTest
Executando teste: Verifica��o RSA...
Executando teste: Falha de Login...
Executando teste: Login com Sucesso...
Tests run: 3, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.701 s -- in com.transportadora.model.LoginTest

Results:

Tests run: 3, Failures: 0, Errors: 0, Skipped: 0

------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  8.978 s
Finished at: 2025-11-27T22:03:38-03:00
------------------------------------------------------------------------
```

5.2. Testes de Regra de Negócio (Validação de Carga)

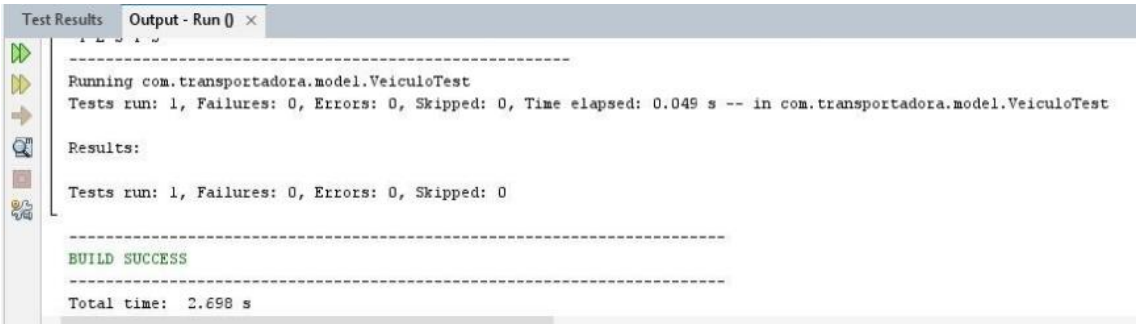
- **Caso de Teste:** CT-003 (Validação da Regra RD001).
- **Objetivo:** Garantir que o sistema bloqueie pedidos com peso superior à capacidade do veículo (ex: Carga de 3.100kg em VUC de 3.000kg).

- **Resultado:** Sucesso. O sistema registrou 2 execuções com zero falhas, confirmando o bloqueio de cargas excedentes.

```
-------------------------------------------------------
Running com.transportadora.model.test.PedidoTest
Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.062 s -- in com.transportadora.model.test.PedidoTest

Results:

Tests run: 2, Failures: 0, Errors: 0, Skipped: 0

------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  2.179 s
Finished at: 2025-12-05T18:51:04-03:00
------------------------------------------------------------------------
```


5.3. Testes Operacionais (Gestão de Frota)

- **Objetivo:** Validar status inicial e persistência da capacidade de carga.
- **Procedimento:** Instanciação de veículo e verificação via getStatus() e getCapacidadeCarga().
- **Resultado:** Sucesso. A criação de veículos segue estritamente as regras definidas, garantindo integridade para a designação de rotas.



6. CONCLUSÃO

O desenvolvimento do Sistema de Gestão Logística atingiu os objetivos propostos na etapa atual. A arquitetura MVC permitiu o desenvolvimento modular, facilitando a integração entre as partes de Segurança, Pedidos e Frota. Os testes unitários comprovaram que as regras de negócio críticas (como a limitação de carga RD001) e os requisitos não-funcionais de segurança (Criptografia RSA) foram implementados com sucesso e estão operacionais.