Carga Explosiva

Documento de Arquitetura de Software

Versão 1.1

Documento apresentado na disciplina
Projeto de Software para o curso de
Graduação em Ciência da Computação
da Universidade Federal Fluminense,
ministrado pelo docente Vânia de
Oliveira Neves

Barbara Keren N. C. Guarino
Caio Guimarães Cardoso
Gabriel Ramalho Braga
Gyselle Regina Gonçalves de Mello

Sumario

Histórico de Revisão

Data	Versão	Descrição	Autor
11/04/2024	01.00.00	Criação do Documento	Barbara
10/05/2024	01.01.00	Conclusão Parcial do Documento	Barbara, Gyselle, Gabriel e Caio
30/05/024	01.10.00	Conclusão Final do Documento	Barbara, Gyselle, Gabriel e Caio

Introdução

Este documento apresenta uma visão abrangente da arquitetura do sistema de gestão de frota desenvolvido pela [equipe]. O sistema visa proporcionar uma solução completa para empresas de transporte e logística, permitindo o gerenciamento eficiente de veículos, motoristas e manutenção. Esta introdução fornecerá uma visão geral do conteúdo do documento, destacando os principais aspectos da arquitetura e sua importância para o sucesso do sistema.

Finalidade

Este documento oferece uma visão geral arquitetural abrangente do sistema [nome do sistema], usando diversas visões arquitetônicas para representar diferentes aspectos do sistema. O objetivo deste documento é capturar e comunicar as decisões arquitetônicas significativas que foram tomadas em relação ao sistema.

Definições, Acrônimos e Abreviações

Definições, Acrônimos e Abreviações	**Descrição**
Frota	Conjunto de veículos pertencentes a uma empresa ou organização
Usuário	Pessoa autorizada a interagir com o sistema de gestão de frota.
API	Interface de Programação de Aplicativos.
CRUD	Create, Read, Update, Delete (Criar, Ler, Atualizar, Excluir)
SQL	Structured Query Language (Linguagem de Consulta Estruturada).
UI	User Interface (Interface do Usuário).
GPS	Global Positioning System (Sistema de Posicionamento Global).
SMTP	Simple Mail Transfer Protocol (Protocolo de Transferência de Correio Simples).
SOLID	É um acrônimo para: Single Responsibility Principle, Open-Closed Principle, Liskov Substitution Principle, Interface Segregation Principle e Dependency Inversion Principle

Definições, Acrônimos e Abreviações	Descrição
GRASP	General Responsibility Assignment Software Patterns
GoF	Padrões de Projeto GoF (Gang of Four)

Escopo

Este Documento de Arquitetura de Software se aplica ao sistema Carga Explosiva, que será desenvolvido pela Carga Explosiva visando empresas de transportes e logísticas.

O sistema trata-se de um software para gestão de frota com suporte para desktops e smartphones, com o objetivo de facilitar o gerenciamento eficiente dos veículos da frota, assim como o gerenciamento de motoristas e da oficina própria que a empresa possui para pequenas manutenções corretivas e preventivas.

O sistema incluirá funcionalidades abrangentes como o cadastro e gestão detalhada de veículos, permitindo acompanhar informações como modelo, placa, ano de fabricação, número do chassi, capacidade de carga, histórico de manutenções, histórico de multas, histórico de sinistros, quilometragem, histórico de abastecimento, entre outros dados vinculados ao veículo.

Outra funcionalidade crucial será o gerenciamento dos motoristas. O sistema permitirá o cadastro completo dos motoristas, incluindo dados pessoais, documentos, habilitação, treinamento, histórico de condução e experiência profissional. Cada motorista poderá ser atribuído a veículos específicos, facilitando o controle das escalas de trabalho e dos períodos de condução.

No que diz respeito a manutenção, o sistema oferece recursos de agendamentos e registros de manutenção, permitindo que os usuários possam agendar serviços preventivos e corretivos, planejar manutenções periódicas com base em quilometragem ou tempo de uso, registrar todas as atividades de manutenção realizadas em cada veículo, além de acompanhar custos, materiais e peças utilizadas em cada manutenção.

Além disso, o sistema controlará o estoque e serviços prestados pela oficina que a empresa possui, permitindo o registro de entradas, saídas, transferências e ajustes dos serviços prestados, assim como, de estoque de peças e materiais. Alertas automáticos serão gerados em caso de estoque baixo, facilitando a reposição de itens essenciais.

Por fim, o sistema fornecerá uma variedade de relatórios e análises sobre o desempenho da frota e das operações de manutenção. Esses relatórios serão personalizáveis e permitirão uma avaliação detalhada do custo total de propriedade, da eficiência operacional, do consumo de combustível e da produtividade dos motoristas.

Em síntese, o sistema de gestão de frota será uma ferramenta essencial para otimizar as operações das empresas de transporte e logística, garantindo um controle detalhado, levando a tomadas de decisão embasadas em dados e redução significativa de custos operacionais.

Requisitos Arquiteturalmente Significantes

Nesta seção, serão apresentados os requisitos arquitetônicos significativos para o sistema de gestão de frota. Esses requisitos são essenciais para a definição da arquitetura do software e são baseados nas necessidades funcionais, qualidades desejadas e restrições do sistema. Ao identificar e descrever esses requisitos, busca-se garantir que a arquitetura dos sistemas atenda plenamente às expectativas de integração, desempenho, segurança e usabilidade, contribuindo assim para o sucesso e eficiência do sistema como um todo.

Requisitos	**Descrição**
Integração com Dispositivos Móveis	Necessidade de suportar desktops e smartphones, garantindo integração e sincronização eficientes entre diferentes plataformas.
Segurança	Proteção dos dados contra acessos não autorizados e autenticados, garantindo confidencialidade e integridade.
Desempenho	Garantia de um desempenho eficiente, especialmente para operações em tempo real.
Flexibilidade e Manutenibilidade	Capacidade de suportar mudanças e atualizações sem impactar negativamente a estabilidade ou funcionalidade do sistema.
Disponibilidade	Garantia de disponibilidade do sistema durante a maior parte do tempo.
Persistência	Necessidade de armazenar e recuperar dados de forma eficiente e confiável.
Armazenamento de Arquivos e Imagens	Capacidade de armazenar e recuperar arquivos e imagens relacionados à frota e oficina.
Localização em Tempo Real	Necessidade de rastrear a localização em tempo real dos veículos da frota.
Interface de Usuário Dinâmica e Amigável	Exigência de um interface intuitiva e responsiva para os usuários interagirem facilmente com o sistema.
Reuso de Componentes	Capacidade de reutilizar componentes de interface do usuário e lógica de negócios para minimizar a duplicação de código e facilitar a manutenção, seguindo padrões de design e boas práticas

Requisitos	Descrição
	de desenvolvimento.
Interoperabilidade	O sistema deve se comunicar e interagir com outros sistemas, por exemplo, os sistemas de gerar relatório e de localização em tempo real.

Metas e Restrições Arquiteturais

A arquitetura do sistema de gestão de frota será guiada por uma abordagem centrada na integração, confiabilidade e usabilidade. O sistema será projetado para atender às necessidades complexas e multifacetadas de gerenciamento de frota, integrando-se de forma eficiente com sistemas legados e oferecendo alto desempenho em condições operacionais e diversas. A robustez durante um longo período de manutenção será uma prioridade, garantindo que o sistema permaneça estável e fácil de manter ao longo do tempo. Além disso, a segurança dos dados e das comunicações será uma preocupação central, assegurando a integridade e confidencialidade das informações em todas as integrações do sistema. A filosofia da arquitetura busca criar uma solução flexível, modular e altamente adaptável, capaz de evoluir com as demandas do mercado e as necessidades em constante mudança dos usuários.

As metas arquitetônicas definem os objetivos que orientarão o design e o desenvolvimento do sistema de gestão de frota, visando garantir sua eficácia, confiabilidade e usabilidade. A arquitetura do sistema será projetada com foca na:

Metas	Descrição
Confiabilidade	A arquitetura deve ser projetada para garantir a confiabilidade do sistema, minimizando o tempo de inatividade não planejado e garantindo a disponibilidade contínua dos serviços.
Usabilidade	A arquitetura deve priorizar a usabilidade, proporcionando uma interface de usuário intuitiva e amigável para garantir uma experiência de uso positiva para os usuários.
Segurança	A arquitetura deve garantir a segurança dos dados e das comunicações, protegendo o sistema contra ameaças externas e internas, e garantindo a integridade e confidencialidade dos dados.
Manutenibilidade	A arquitetura deve ser projetada para facilitar a manutenção do sistema, permitindo atualizações e modificações sem impactar negativamente a estabilidade ou a funcionalidade do sistema.
Portabilidade	A arquitetura deve garantir a portabilidade do sistema, permitindo

Metas	Descrição
	que ele seja executado em diferentes plataformas e ambientes de execução, como desktops e dispositivos móveis.
Modularidade	A arquitetura deve ser modular, permitindo a divisão do sistema em componentes independentes, facilitando a manutenção e evolução do sistema e promovendo o reuso de componentes.
Tolerância a Falhas	A arquitetura deve ser projetada para lidar com falhas de componentes individuais, garantindo a disponibilidade contínua do sistema e minimizando o impacto de falhas no funcionamento do sistema.
Eficiência	A arquitetura deve garantir a eficiência do sistema, otimizando o uso de recursos e garantindo um desempenho adequado mesmo em condições de carga elevada ou recursos limitados.
Interação	A arquitetura deve facilitar a integração com sistemas externos, permitindo a troca de dados e a comunicação sem problemas entre diferentes sistemas e serviços.

Para garantir o sucesso do sistema de gestão de frota, é essencial identificar e considerar cuidadosamente as restrições arquitetônicas que podem impactar o design e a implementação da solução. Essas restrições representam limitações e requisitos não negociáveis que devem ser levados em consideração ao definir a arquitetura do sistema.

Restrições	Descrição
Conformidade com Padrões	A arquitetura deve aderir a padrões de desenvolvimento de software reconhecidos e amplamente aceitos, garantindo a interoperabilidade e a manutenção do sistema.
Tecnologia de Plataforma	O sistema deve ser desenvolvido utilizando tecnologias que suportem tanto desktop quanto dispositivos móveis, garantindo uma experiência consistente em todas as plataformas.
Segurança	A arquitetura deve priorizar a segurança, incorporando práticas e mecanismos robustos para proteger os dados contra acesso não autorizado e ataques cibernéticos.
Desempenho	O sistema deve ser otimizado para garantir um desempenho eficiente, minimizando os tempos de resposta e maximizando a capacidade de processamento em todas as condições de uso.
Confiabilidade	A arquitetura deve ser projetada para garantir a confiabilidade do

Restrições	Descrição
	sistema, minimizando o tempo de inatividade não planejado e as interrupções nas operações de frota.

Considerando as metas e restrições definidas, a arquitetura do sistema de gestão de frota será projetada para atender não apenas aos requisitos funcionais, mas também às expectativas de confiabilidade, usabilidade e segurança. Ao identificar e abordar esses elementos desde o início do processo de design, buscamos garantir que o sistema seja robusto, capaz de manter a disponibilidade contínua das operações da frota, e capaz de evoluir com as necessidades do negócio e as demandas do mercado.

Arquitetura do Software

A arquitetura de software desempenha um papel fundamental na concepção e implementação de sistemas robustos e eficientes. Para o desenvolvimento do sistema de gestão de frota, optamos por adotar uma abordagem que combina elementos da arquitetura **Cliente-Servidor** com a estrutura em **Camadas**. Essa mistura de padrões arquiteturais visa aproveitar as vantagens de cada abordagem, proporcionando uma solução que atende às necessidades complexas e multifacetadas do sistema.

A mistura das arquiteturas Cliente-Servidor e em Camadas oferece uma abordagem flexível para o desenvolvimento do software em desenvolvimento. Nesta abordagem, o sistema é dividido em duas partes principais: o cliente e o servidor. No lado cliente, são desenvolvidos aplicativos desktop e móveis que atuam como interfaces de usuário para interação com o sistema. No lado servidor, a lógica de processamento e armazenamento de dados é organizada em camadas distintas: apresentação, lógica de negócios e persistência.

Camadas no Lado do Cliente:

Camada de Segurança (Autenticação do Cliente): Esta camada é responsável por autenticar os usuários do sistema, garantindo que apenas usuários autorizados possam acessar as funcionalidades do sistema. Arqui, os usuários fornecem suas credenciais de login para se autenticar no sistema antes de poderem interagir com a interface do usuário.

Camada de Apresentação (Interface do Usuário): Nesta camada, são desenvolvidas as interfaces do usuário que o clientes interagem para acessar e utilizar o sistema. Aqui, os aplicativos desktop e móveis são criados para proporcionar uma experiência intuitiva e responsiva aos usuários, permitindo que eles visualizem e interajam com as informações de maneira eficiente.

Camada de Aplicação (Lógica do Cliente): Esta camada é responsável por processar a entrada do usuário e preparar as solicitações para serem enviadas ao servidor. Aqui, a lógica de

negócio relacionada à interação do cliente com o sistema é implementada, garantindo que as operações sejam realizadas de acordo com as regras estabelecidas e que os dados sejam formatados corretamente antes de serem enviados.

Camada de Comunicação (Comunicação de Rede): Nesta camada, são estabelecidas e gerenciadas as comunicações entre o cliente e o servidor. Aqui, são implementados os métodos necessários para enviar e receber dados do servidor, garantindo uma comunicação eficiente e confiável entre as partes.

Camadas no Lado do Servidor:

Camada de Segurança (Autenticação e Autorização): Esta camada é responsável por garantir a autenticação e autorização dos usuários, bem como a proteção dos dados contra acessos não autorizados. Ela inclui a implementação de mecanismos de autenticação, como login com nome de usuário e senha, e autorização baseada em papéis para controlar o acesso às funcionalidades do sistema. Além disso, ela gerencia sessões e tokens de autenticação para garantir a segurança contínua do sistema.

Camada de Apresentação (Recebimento de Requisições): Responsável por receber as solicitações dos clientes e encaminhá-las para o processamento adequado. Aqui, os endpoints de API são implementados para lidar com as requisições dos clientes, garantindo que cada solicitação seja roteada para o destino correto dentro do revisor.

Camada de Aplicação (Lógica de Negócios): Esta camada é responsável por processar as solicitações recebidas dos clientes e realizar operações no sistema. Aqui, lógica de negócios é implementada para executar as operações necessárias, como cadastro de veículos, atribuição de motoristas, entre outros, de acordo com as regras de negócio estabelecidas.

Camada de Persistência (Acesso a Dados): Responsável por interagir com o banco de dados para armazenar e recuperar informações sobre veículos, motoristas, manutenção, entre outros. Aqui, são implementados os métodos para realizar operações de CRUD no banco de dados, garantindo que os dados sejam armazenados e recuperados de forma eficiente e segura.

A mistura das arquiteturas Cliente-Servidor e em Camadas oferece uma solução robusta e flexível para o sistema de gestão de frota. Ao combinar elementos de centralização, segurança e controle do servidor com a flexibilidade, reutilização de componentes e manutenibilidade das camadas separadas, conseguimos desenvolver um sistema que atende às necessidades complexas e em constante evolução do nosso cenário operacional. Esta abordagem arquitetural permite confiabilidade e eficiência, garantindo uma experiência positiva para os usuários finais e uma base sólida para futuras expansões e atualizações do sistema.

Mecanismos Arquiteturais

Os mecanismos arquiteturais são elementos essenciais para garantir o funcionamento adequado e eficiente de um sistema de software. Eles representam as estratégias e técnicas utilizadas para lidar com aspectos cruciais da arquitetura, como persistência de dados, comunicação entre componentes e tratamento de erros. Nesta seção, serão listados e descritos diversos mecanismos arquitetônicos utilizados no sistema, destacando suas finalidades, atributos e funções.

Mecanismo Arquitetural	**Finalidade**	**Atributos**	**Funções**
Persistência de Dados	Garantir a persistência e recuperação eficiente de dados no sistema, permitindo que informações importantes sejam armazenadas de forma confiável.	Confiabilidade, Eficiência, Consistência	Realizar operações de CRUD no banco de dados, gerenciar transações, otimizar consultas e garantir a integridade dos dados.
Comunicação entre Componentes	Facilitar a comunicação e troca de dados entre os diversos componentes do sistema, permitindo a integração e interação entre eles.	Confiabilidade, Eficiência, Segurança.	Implementar protocolos de comunicação, gerenciar mensagens, garantir a entrega e recebimento de dados de forma assíncrona ou síncrona.
Tratamento de Erros	Lidar com situações de exceção e erros de forma adequada, minimizando impactos negativos no funcionamento do sistema e proporcionando uma experiência de usuário consistente.	Robustez, Resiliência, Monitoramento.	Capturar exceções, registrar erros, notificar usuários ou administradores, implementar estratégias de fallback e recuperação.
Autenticação e Autorização	Garantir que apenas usuários autorizados tenham acesso ao sistema e a determinadas funcionalidades, protegendo os dados contra acessos não autorizados.	Segurança, Privacidade, Confiabilidade.	Verificar credenciais de usuário, gerenciar sessões e token de autenticação, controlar permissões de acesso a recursos e funcionalidades.

Mecanismo Arquitetural	Finalidade	Atributos	Funções
Geração de Relatórios	Permitir a geração e personalização de relatórios sobre o desempenho da frota e das operações de manutenção, fornecendo insights valiosos para tomada de decisões.	Flexibilidade, Eficiência, Customização.	Coletar dados relevantes, processar informações, formatar e apresentar relatórios de maneira clara e compreensível.
Filtragem e Validação de Dados de Entrada	Prevenir ataques de segurança, como inserção de códigos maliciosos, e garantir a integridade dos dados processados pelo sistema, filtrando e validando informações de entrada.	Segurança, Integridade, Confiabilidade.	Validar formatos de dados, sanitizar entradas para evitar SQL Injection, XSS e outros ataques, rejeitar dados suspeitos ou malformados.
Notificação e Alerta	Notificar os usuários sobre eventos importantes e alertar sobre situações críticas, garantindo uma resposta rápida a problemas e necessidades urgentes de acordo com a necessidade de cada usuário.	Responsividade, Confiabilidade, Eficiência.	Enviar notificações por e-mail, mensagens push, configurar alertas automáticos para situações como estoque baixo, vencimento de documentos, manutenções pendentes, entre outros.
Backup e Recuperação	Garantir a integridade e disponibilidade dos dados do sistema, possibilitando a recuperação de informações em caso de falhas ou desastres.	Segurança, Confiabilidade, Resiliência.	Realizar backups regulares dos dados do sistema, armazenar cópias de segurança em locais seguros, implementar procedimentos de recuperação de desastres.
Controle de Acesso	Gerenciar os direitos de acesso dos usuários ao sistema, garantindo que cada usuário tenha permissões adequadas de acordo com sua função e responsabilidades.	Segurança, Confidencialidade, Conformidade.	Definir políticas de acesso, controlar e restringir o acesso a recursos e funcionalidades com base nas permissões atribuídas a cada usuário,

Mecanismo Arquitetural	Finalidade	Atributos	Funções
			monitorar atividades de acesso para garantir conformidade com as políticas de segurança.

Os mecanismos arquiteturais desempenham um papel crucial na definição e implementação de uma arquitetura de software robusta e eficiente. Eles fornecem as bases necessárias para lidar com desafios complexos e garantir o funcionamento adequado do sistema. Ao entender e aplicar corretamente esses mecanismos, é possível criar um sistema confiável, seguro, capaz de atender às necessidades do negócio e dos usuários finais.

Decisões e Justificativas

Nesta seção, apresentamos as decisões arquiteturas fundamentais tomadas durante o processo de concepção do sistema, bem como as justificativas que embasam essas escolhas. Ao longo do desenvolvimento do projeto, foram consideradas diversas alternativas e abordagens arquitetônicas, levando em conta os requisitos, metas e restrições.

Decisão	Justificativa
Escolhas da Arquitetura Cliente-Servidor em Camadas	A escolha da arquitetura Cliente-Servidor em Camadas, permite uma clara separação de responsabilidades entre o cliente e o servidor, facilitando o desenvolvimento, manutenção e segurança do sistema. Além disso, facilita a implementação de segurança em diferentes níveis, ademais, proporciona maior flexibilidade para adaptação a diferentes plataformas e ambientes de execução, como desktops e dispositivos móveis.
Implementação de Camadas de Segurança em Cliente e Servidor	Essa decisão é crucial para atender aos requisitos de segurança, autenticação e autorização. A implementação de medidas de segurança em ambas as camadas ajuda a proteger os dados contra acessos não autorizados e garantir a integridade e confidencialidade das informações, contribuindo para a confiabilidade do sistema.
Desempenho e Disponibilidade	Em um ambiente operacional de gestão de frota, onde operações em tempo real podem ser críticas, o desempenho e a disponibilidade são fundamentais. A arquitetura Cliente-Servidor em Camadas facilita a otimização do desempenho e garante a disponibilidade contínua do sistema, minimizando o tempo de

Decisão	Justificativa
	inatividade não planejado.
Usabilidade e Manutenibilidade	Uma interface de usuário amigável é essencial para garantir uma experiência positiva para os usuários do sistema. A separação em camadas facilita o desenvolvimento de uma interface de usuário intuitiva e responsiva, contribuindo para a usabilidade e manutenibilidade do sistema ao longo do tempo.
Adoção de Práticas de Reuso de Componentes	Essa decisão apoia a meta de manutenção e a necessidade de flexibilidade. Ao reutilizar componentes, é possível minimizar a duplicidade de código, facilitando a manutenção e evolução do sistema ao longo do tempo, sem comprometer a estabilidade ou funcionalidade do sistema.

Representação Arquitetural

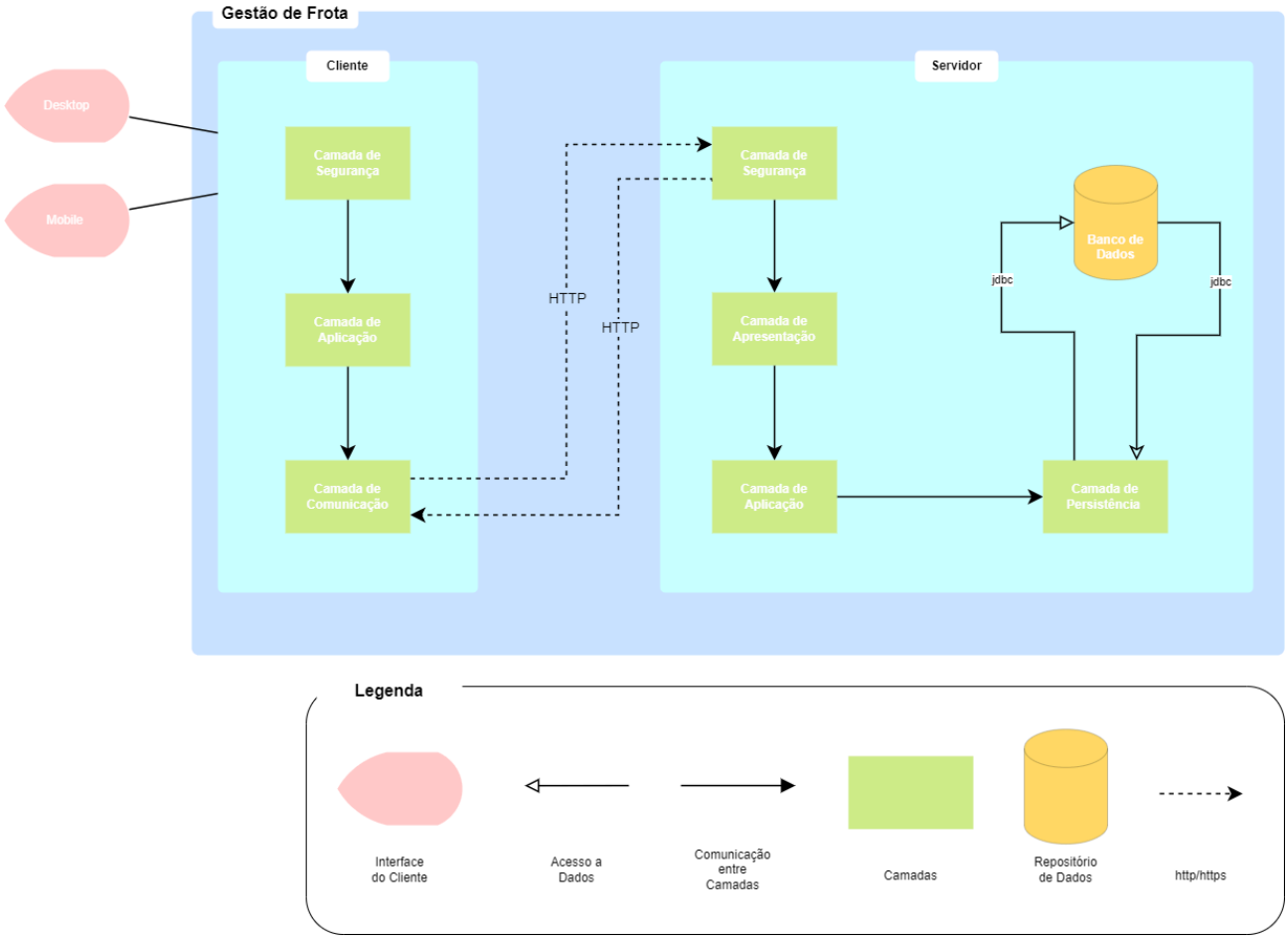


Imagem I *- Visão Geral da Arquitetura do Software*

Visões Arquiteturais

O sistema de gestão de frota é descrito por meio de várias visões arquitetônicas, conforme o modelo “4+1” proposto por Kruchten [KRU41]. Cada visão oferece uma perspectiva única do sistema, permitindo uma compreensão abrangente de sua estrutura e funcionamento.

Cenarios

Os cenários descrevem situações de uso do sistema de gestão de frota que estão diretamente ligadas à arquitetura. Eles não definem a estrutura do sistema, mas são essenciais para entender como o sistema opera e porque a arquitetura foi projetada da maneira como é. Cada cenário destaca uma interação específica entre os usuários e o sistema, ilustrando como os componentes arquiteturais trabalham juntos para atender as necessidades dos usuários. Esses cenários são fundamentais para uma compreensão abrangente da arquitetura do sistema e sua aplicação no contexto operacional real.

Casos de Uso

Nesta seção, serão apresentados os casos de uso do sistema, fornecendo uma visão das interações entre os usuários e o sistema.

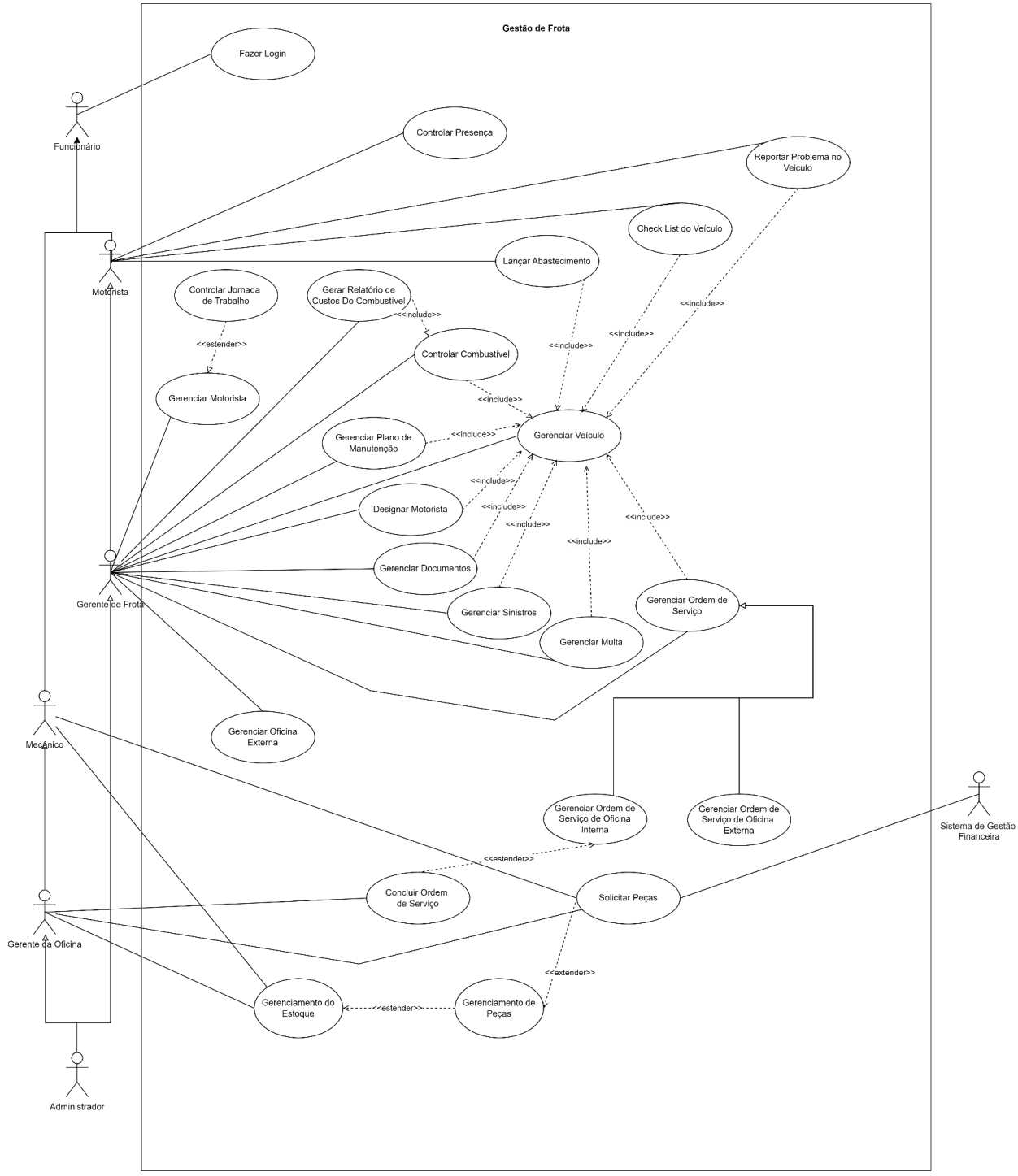


Imagem II *- Diagrama de Casos de Uso*

Diagrama de Sequência do Sistema

Por meio de diagramas de sequência, será ilustrada visualmente a interação entre os atores (usuários, administradores, motoristas) e o sistema. Esses diagramas fornecerão uma representação clara das etapas envolvidas na execução de cada caso de uso, destacando as trocas de mensagens e o fluxo de controle entre os componentes do sistema.

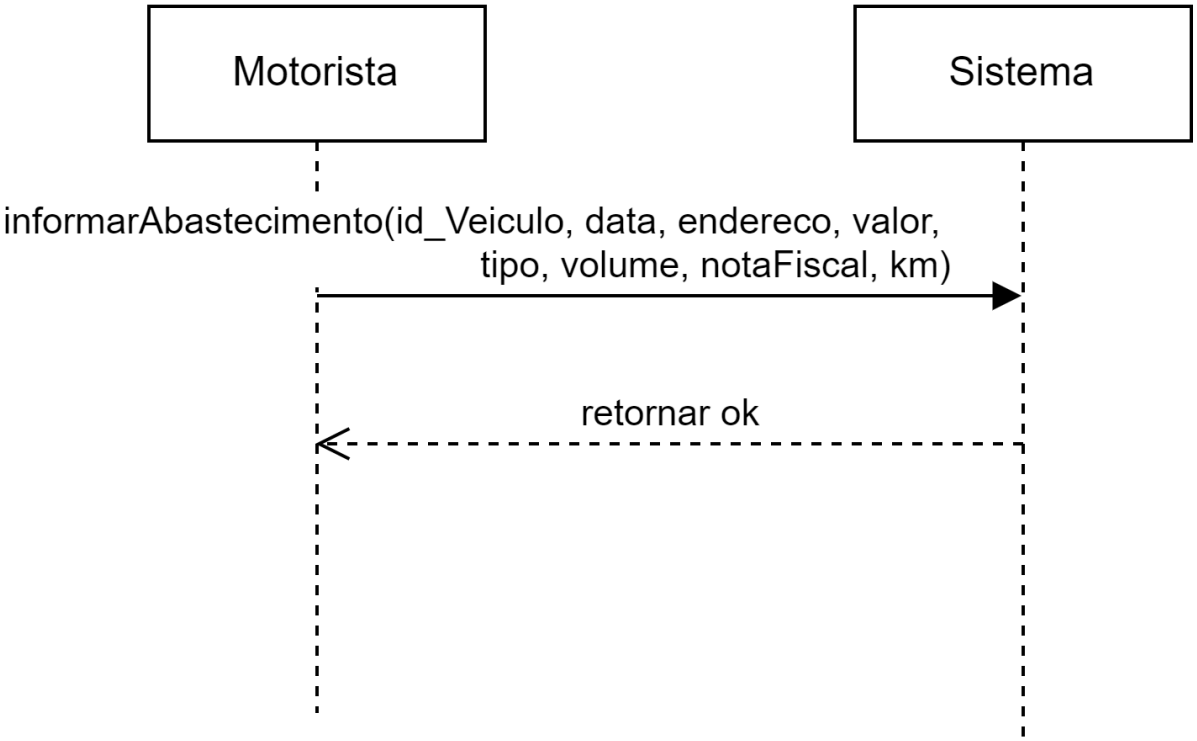


Imagem III *- Diagrama de Sequência do Sistema - Lançar Abastecimento*

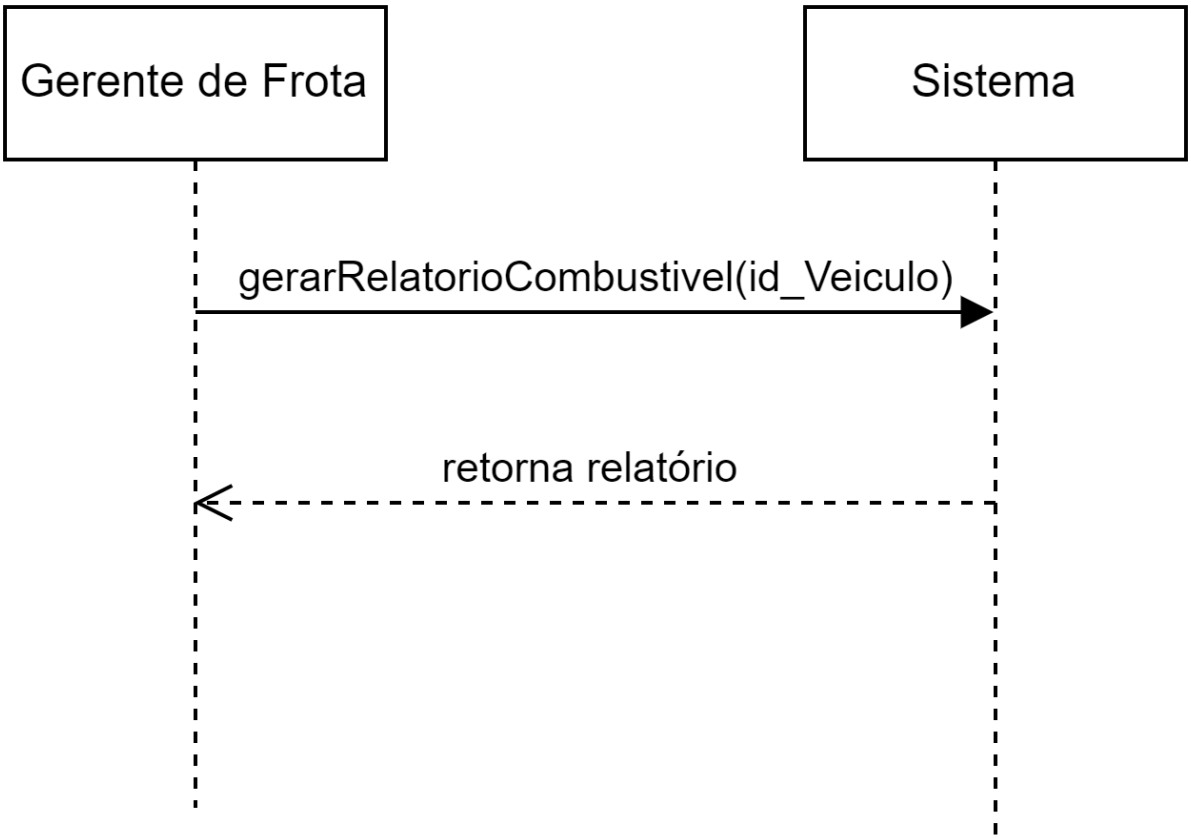


***Imagem IV** - Diagrama de Sequência do Sistema - Gerar Relatório de Custo de Combustível*

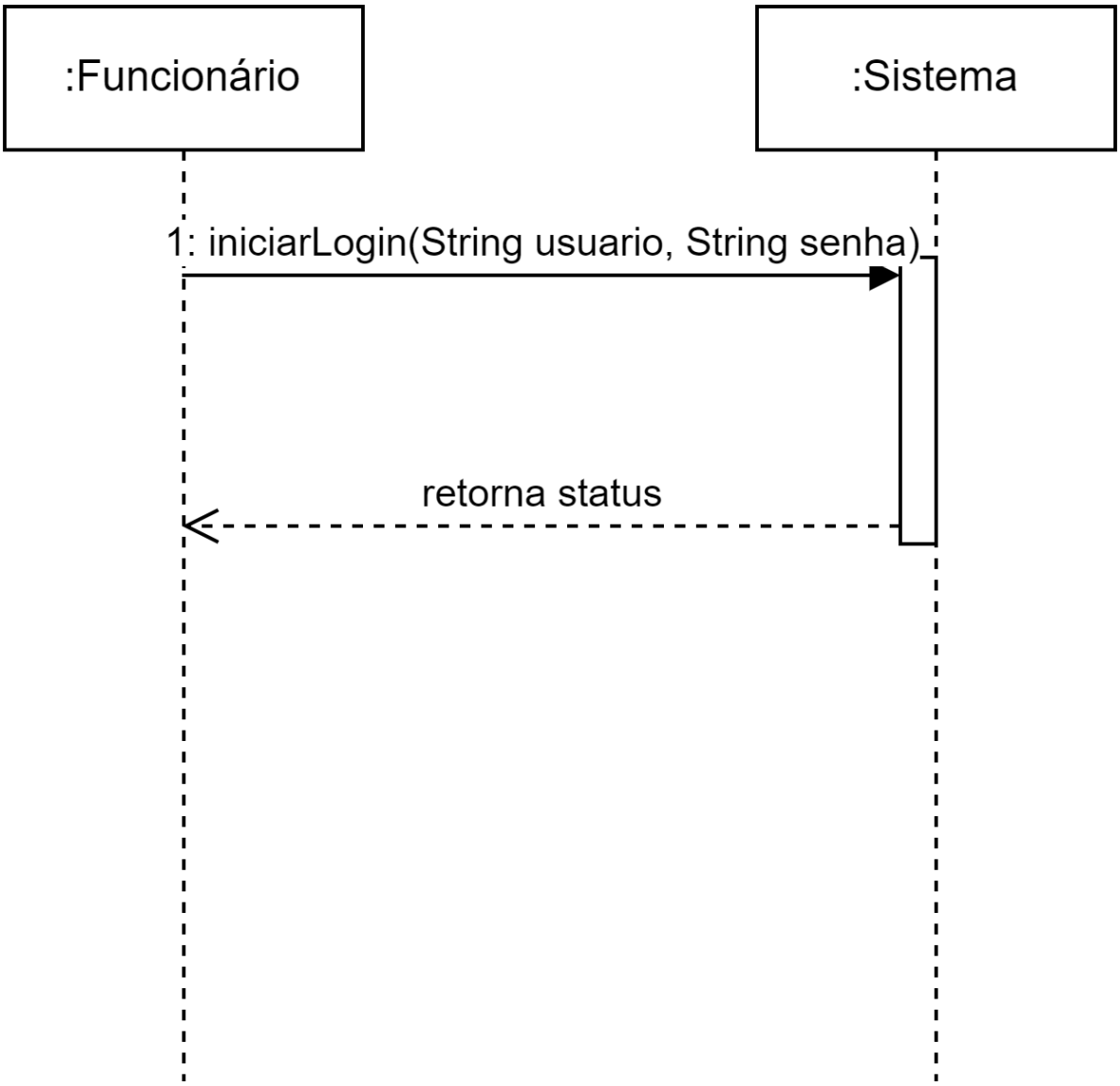


***Imagem V** - Diagrama de Sequência do Sistema - Fazer Login*

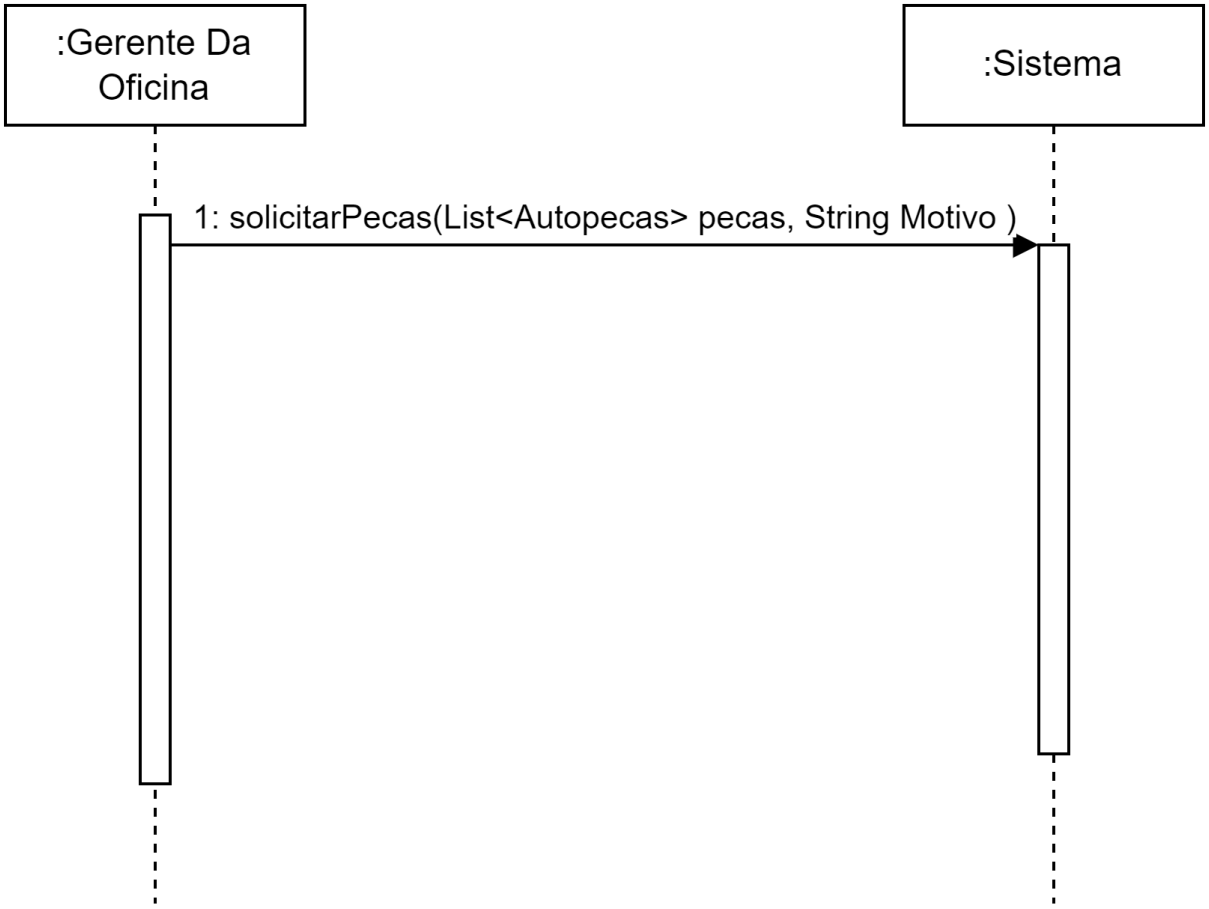


Imagem VI *- Diagrama de Sequência do Sistema - Solicitar Peças*

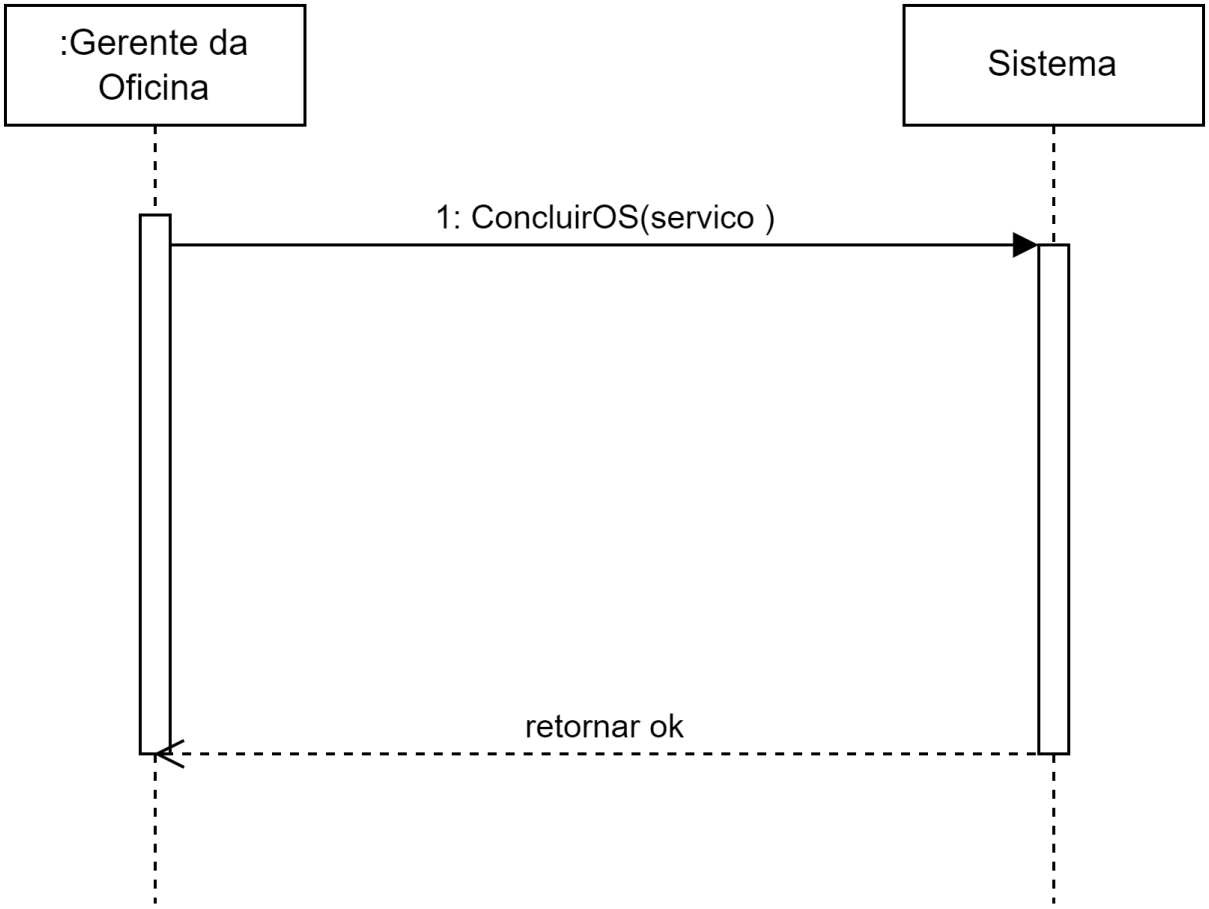


Imagem VII *- Diagrama de Sequência do Sistema - Concluir Os*

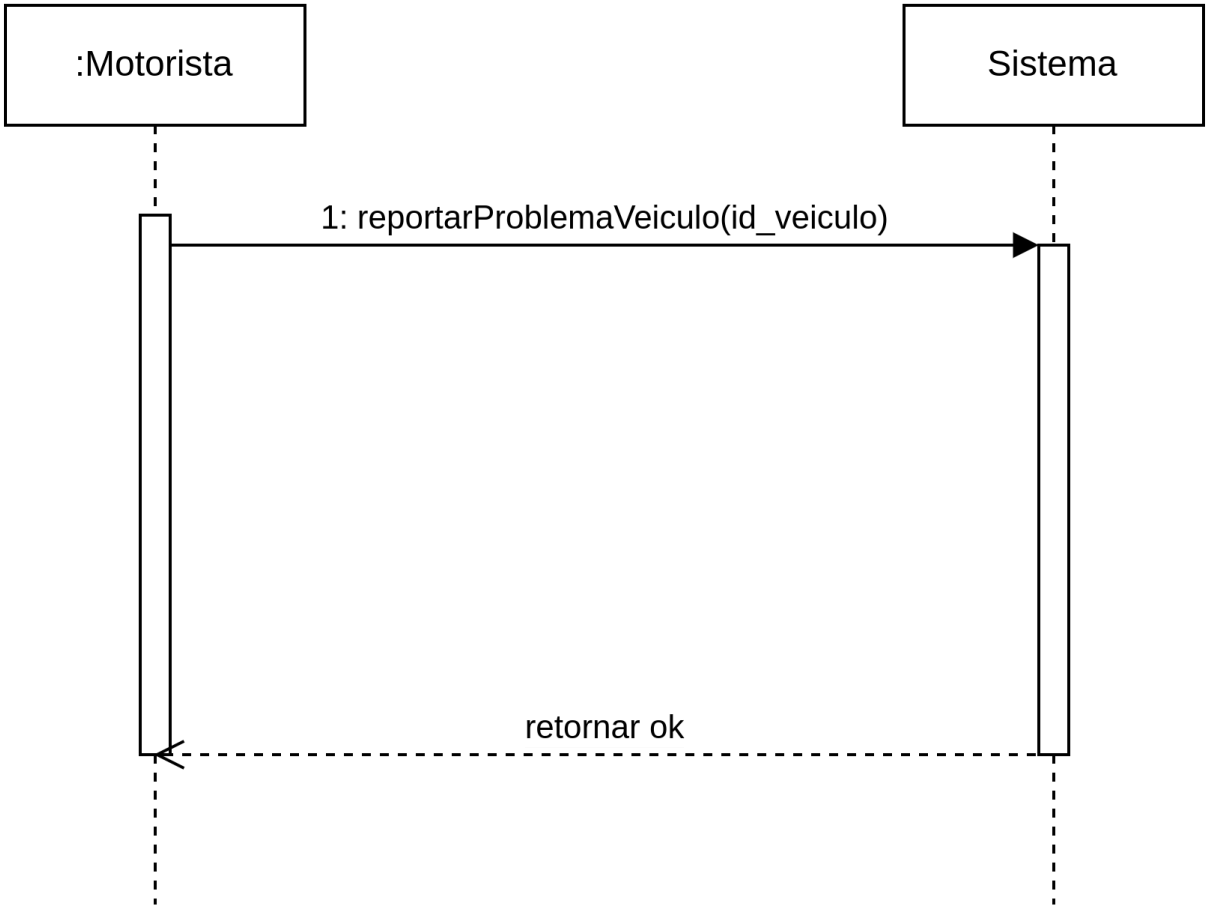


Imagem VIII *- Diagrama de Sequência do Sistema - Relatar Problema Veículo*

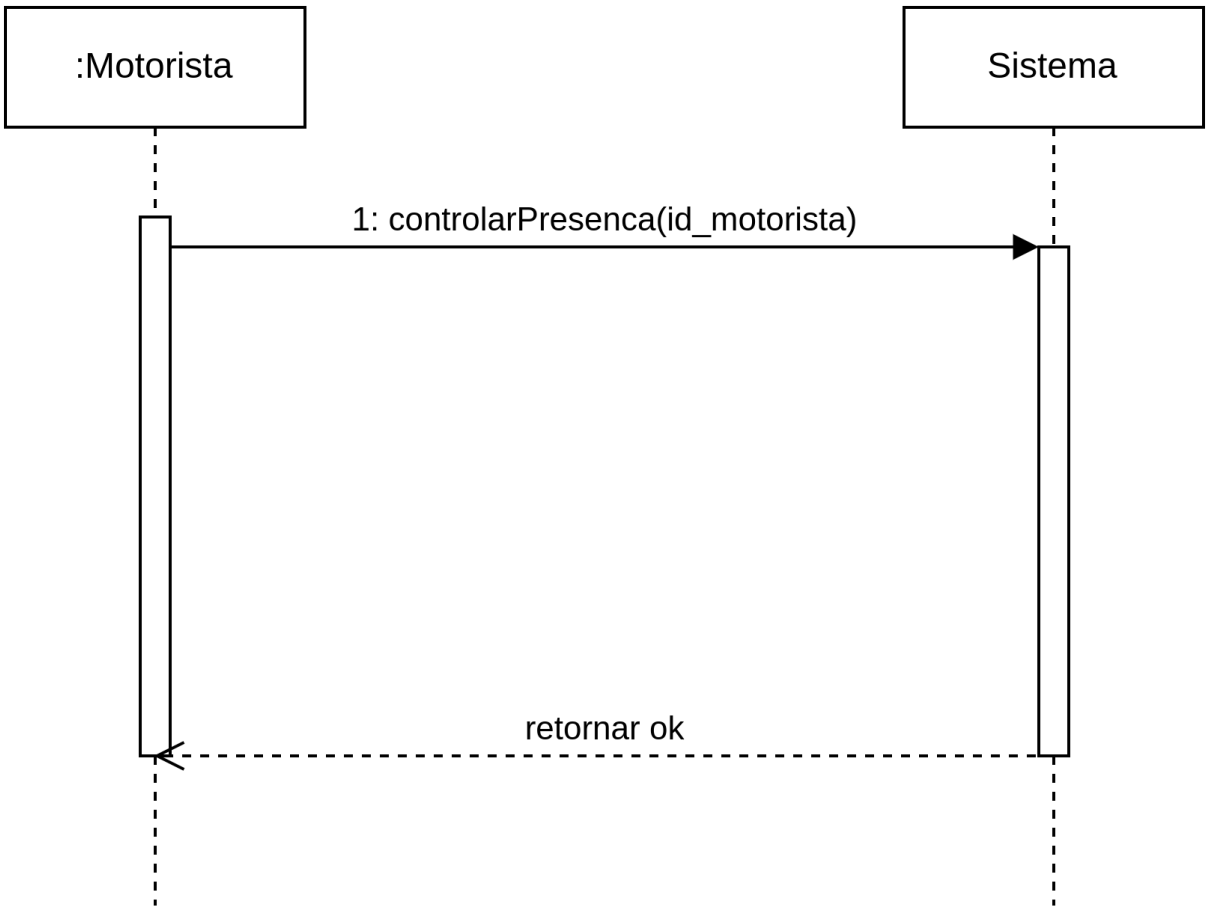


***Imagem IX** - Diagrama de Sequência do Sistema - Controlar Presença*

Contratos das Operações

A seção de contrato de operação define os termos e condições para o uso do sistema Carga Explosiva.

Solicitar Peças	
Nome da Operação	solicitarPecas
Parâmetros de Entrada	- List<ItemDoEstoque> pecas - String motivo
Referências Cruzadas	Casos de Uso: “Solicitar Peças”
Pré-Condições	- Estar logado como Gerente

Solicitar Peças	
Pós-Condições	- Uma instância da classe Pedido é criada

Gerar Relatório de Custo de Combustível	
Nome da Operação	gerarRelatorioCombustivel
Parâmetros de Entrada	- id_Veiculo
Referências Cruzadas	Casos de Uso: “Gerar Relatório de Custo de Combustível”
Pré-Condições	- Estar logado como Gerente - Veículo está cadastrado - Existir pelo menos um abastecimento cadastrado ao veículo informado
Pós-Condições	- Uma instância da classe Relatório do Veículo é criada

Lançar Abastecimento	
Nome da Operação	informarAbastecimento
Parâmetros de Entrada	- id_Veiculo - Date data - List<String> endereco - Float valor - char tipo - Float volume - String notaFiscal - Float km
Referências Cruzadas	Casos de Uso: “Lançar Abastecimento”
Pré-Condições	- Esta logado como Motorista - Motorista está designado para veículo
Pós-Condições	- Uma instância da classe Abastecimento é criada

Reportar Problema no Veículo	
Nome da Operação	reportarProblemaVeiculo
Parâmetros de Entrada	- id_veiculo

Reportar Problema no Veículo	
Referências Cruzadas	Casos de Uso: “Reportar Problema no Veículo”
Pré-Condições	- Estar logado como Motorista - Veículo está cadastrado
Pós-Condições	- O veículo foi identificado - Um objeto “ProblemaRelatado” foi criado e associado ao motorista

Controlar Presença	
Nome da Operação	controlarPresenca
Parâmetros de Entrada	- id_motorista
Referências Cruzadas	Casos de Uso: “Controlar Presença”
Pré-Condições	- Estar logado como Motorista
Pós-Condições	- Um objeto “ControlePresenca” foi criado e associado ao motorista

Fazer Login	
Nome da Operação	fazerLogin
Parâmetros de Entrada	- usuario - senha
Referências Cruzadas	Casos de Uso: “Fazer Login”
Pré-Condições	- Estar online - Estar na tela inicial - Estar deslogado
Pós-Condições	- Usuário logado

Concluir OS	
Nome da Operação	ConcluirOS
Parâmetros de Entrada	- Os Interna

Concluir OS	
Referências Cruzadas	Casos de Uso: “Concluir OS”
Pré-Condições	- Estar logado como Gerente da Oficina - OS Interna existe - O Gerente conhece a OS - OS Interna não está concluída
Pós-Condições	- Os Interna concluída

Designar Motorista	
Nome da Operação	designarMotorista
Parâmetros de Entrada	- id_motorista - id_veiculo
Referências Cruzadas	Casos de Uso: “Designar Motorista”
Pré-Condições	- Estar logado como Gerente
Pós-Condições	- Criar uma instância de AtribuicaoVeiculo_motorista

Visão Lógica

Nesta seção, será apresentada a visão lógica do sistema de Gestão de Frota, destacando as principais entidades e seus relacionamentos. O diagrama de classe de análise fornecerá uma representação estruturada da arquitetura do sistema, destacando as principais entidades e suas interações.

Diagrama de Classe de Análise

O diagrama de classe de análise apresentará as principais entidades do sistema, como Veículo, Motorista, entre outras, e seus relacionamentos. Essa visão lógica será fundamental para compreender a estrutura estática do sistema e identificar os principais pontos de integração e interação entre os componentes.

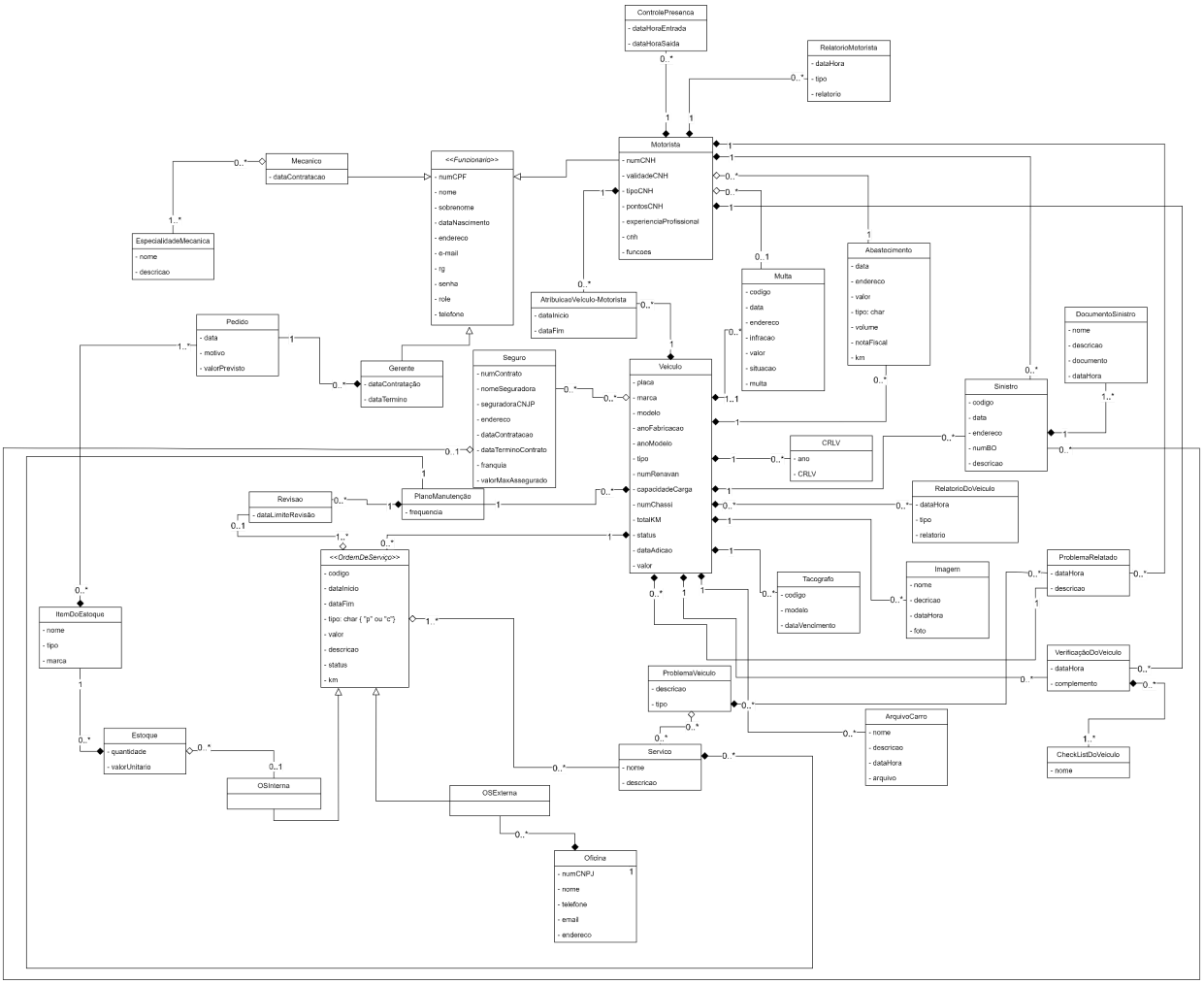


***Imagem X**- Diagrama de Classe de Análise*

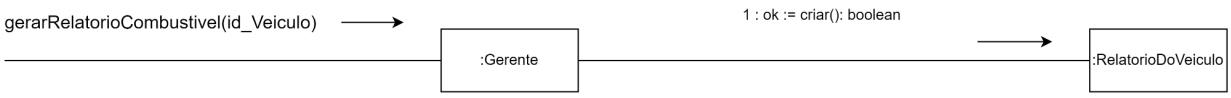
Visão Processo

Nesta seção, serão descritas as interações entre os objetos ou componentes do sistema Carga Explosiva, fornecendo uma visão detalhada dos processos internos. Os diagramas de interação serão utilizados para representar visualmente essas interações e destacar os fluxos de controle e trocas de mensagens entre os componentes do sistema.

Diagrama de Interação

Os Diagramas de Interação são essenciais para entender a dinâmica da comunicação entre os componentes do sistema. Nesta seção, apresentamos os diagramas que detalham como os processos e threads interagem para realizar as funcionalidades do sistema. Esses diagramas fornecem uma

visão clara da troca de mensagens, chamadas de métodos e eventos, permitindo uma análise aprofundada do comportamento em tempo de execução e ajudando a identificar possíveis pontos de contenção ou falhas na comunicação.



Anexo XI *- Diagrama de Interação - Gerar Relatório de Custo de Combustível*

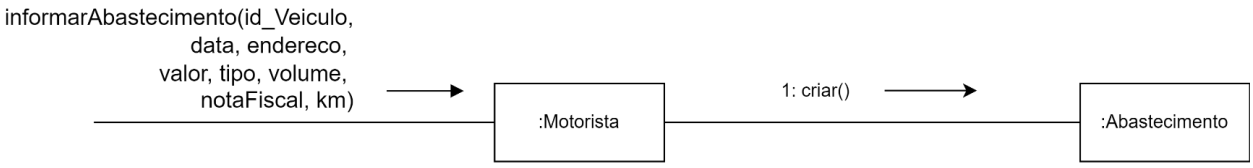


Imagem XII *- Diagrama de Interação - Lançar Abastecimento*

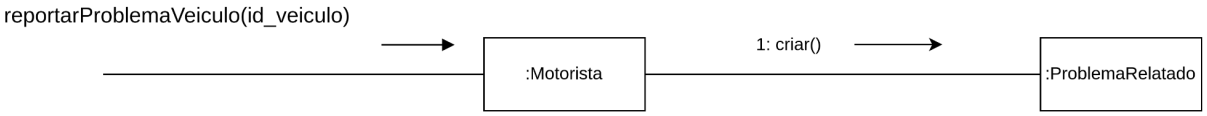


Imagem XIII *- Diagrama de Interação - Reportar Problema Veículo*

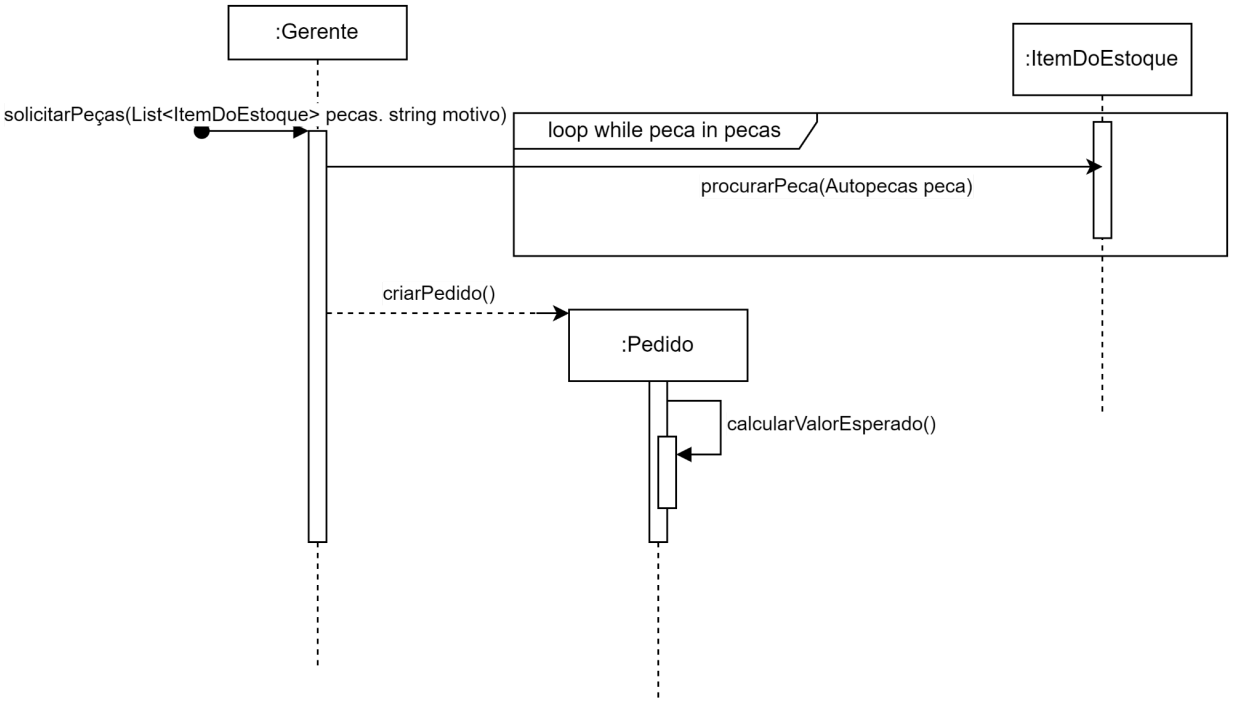


Imagem XIV *- Diagrama de Sequência - Solicitar Peças*

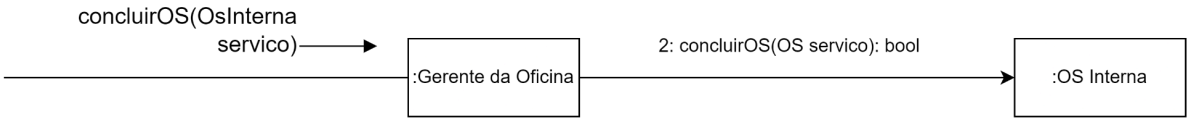


Imagem XV*- Diagrama de Interação - Concluir OS*

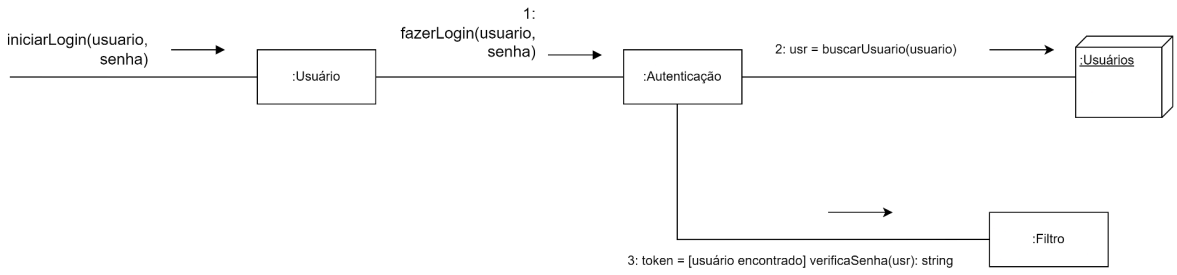


Imagem XVI *- Diagrama de Interação - Fazer Login*

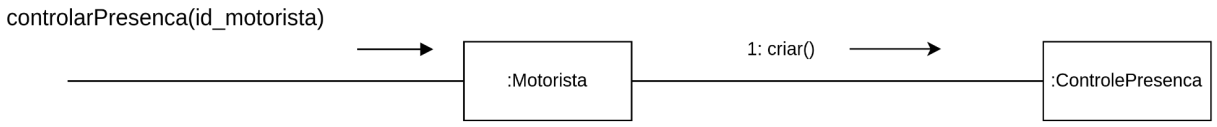


***Imagem XVII** - Diagrama de Interação - Controlar Presença*

Diagrama de Interação (Refatorado)

Os diagramas de interação serão refatorados para incorporar os princípios SOLID e os padrões GRASP, visando melhorar a coesão e o baixo acoplamento entre os componentes do sistema. Essa refatoração será fundamental para garantir a qualidade e a flexibilidade do design do sistema, facilitando sua manutenção e extensão futuras.

Lançar Abastecimento

Utilizando o padrão GRASP “Controller”, onde o Motorista chama um controller para registrar um abastecimento e ele fica responsável por criar uma nova instância da classe Abastecimento.

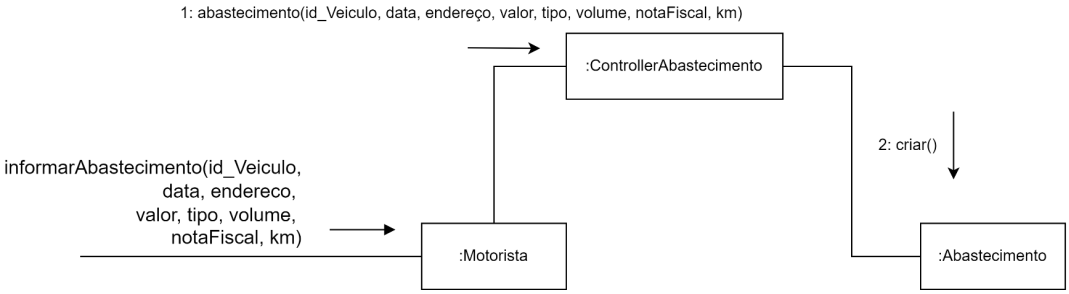


***Imagem XVIII**- Diagrama de Interação - Lançar Abastecimento*

Gerar Relatório de Custo de Combustível

Utilização do padrão GRASP “Controller”, onde foi criado uma classe ControllerRelatorioCombustivel que fica responsável por gerar todos os relatórios referentes ao Abastecimento do Veículos e ele fica responsável por criar uma instância da classe RelatorioDoVeiculo.

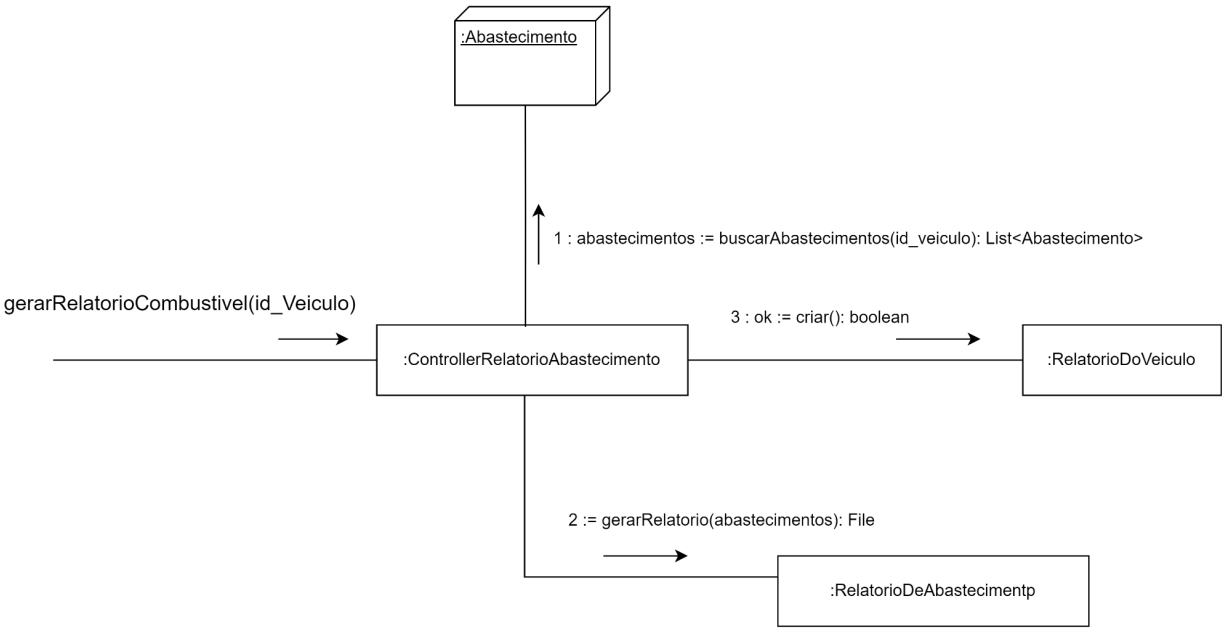


***Imagem XIX**- Diagrama de Interação (Refatorado) - Gerar Relatório de Custo de Combustível*

Concluir OS

No diagrama de interação de Concluir OS foi utilizado o padrão GRASP Especialista e sofreu alteração para incluir o padrão GoF Facade.

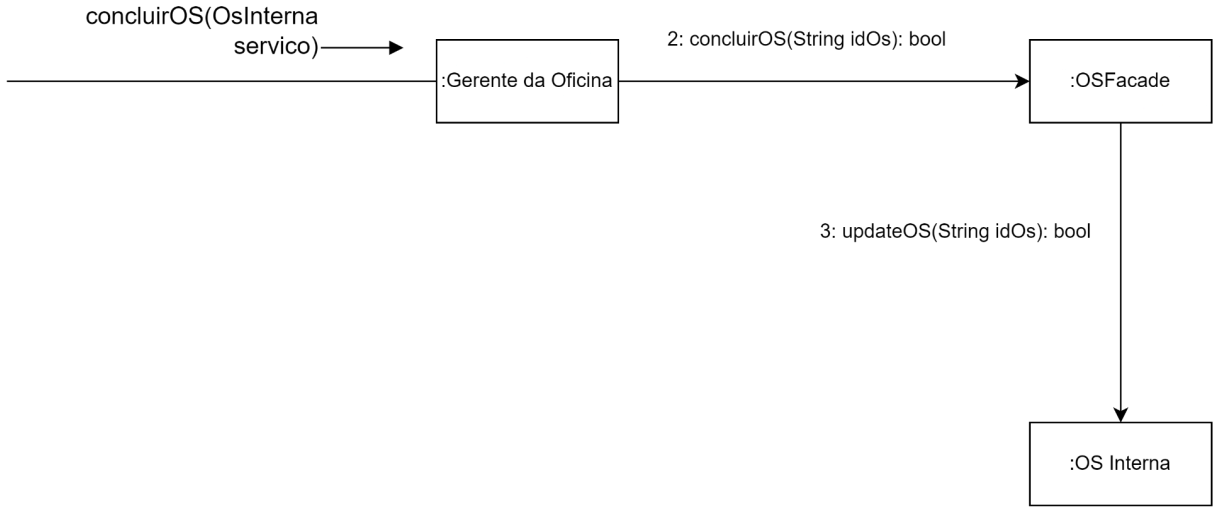


***Imagem XX**- Diagrama de Interação (Refatorado) - Concluir OS*

Fazer Login

No diagrama de interação de fazer Login foi utilizado o padrão Grasp Controler para interação com banco de dados e outras lógicas desse processo.

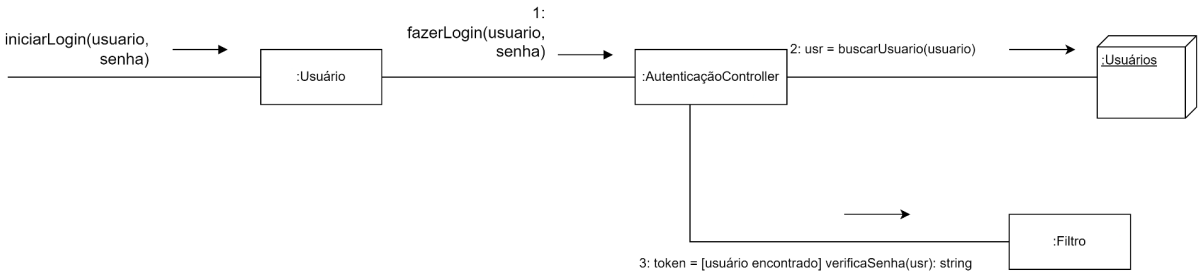


***Imagem XXI**- Diagrama de Interação (Refatorado) - Fazer Login*

Visão de Desenvolvimento

Nesta seção, será apresentada a visão de desenvolvimento do sistema de Gestão de Frota, destacando a estrutura interna das classes e os princípios e padrões aplicados no design do sistema. Os diagramas de classes detalhados fornecerão uma representação detalhada da estrutura interna das classes, enquanto a lista de tecnologias utilizadas destaca as principais tecnologias e ferramentas adotadas no desenvolvimento do sistema.

Diagrama de Classes Detalhado

O diagrama de classes detalhado apresenta a estrutura interna das classes do sistema, representando seus atributos, métodos e relacionamentos. Esses diagramas serão fundamentais para compreender a implementação do sistema e identificar possíveis pontos de melhoria no design.

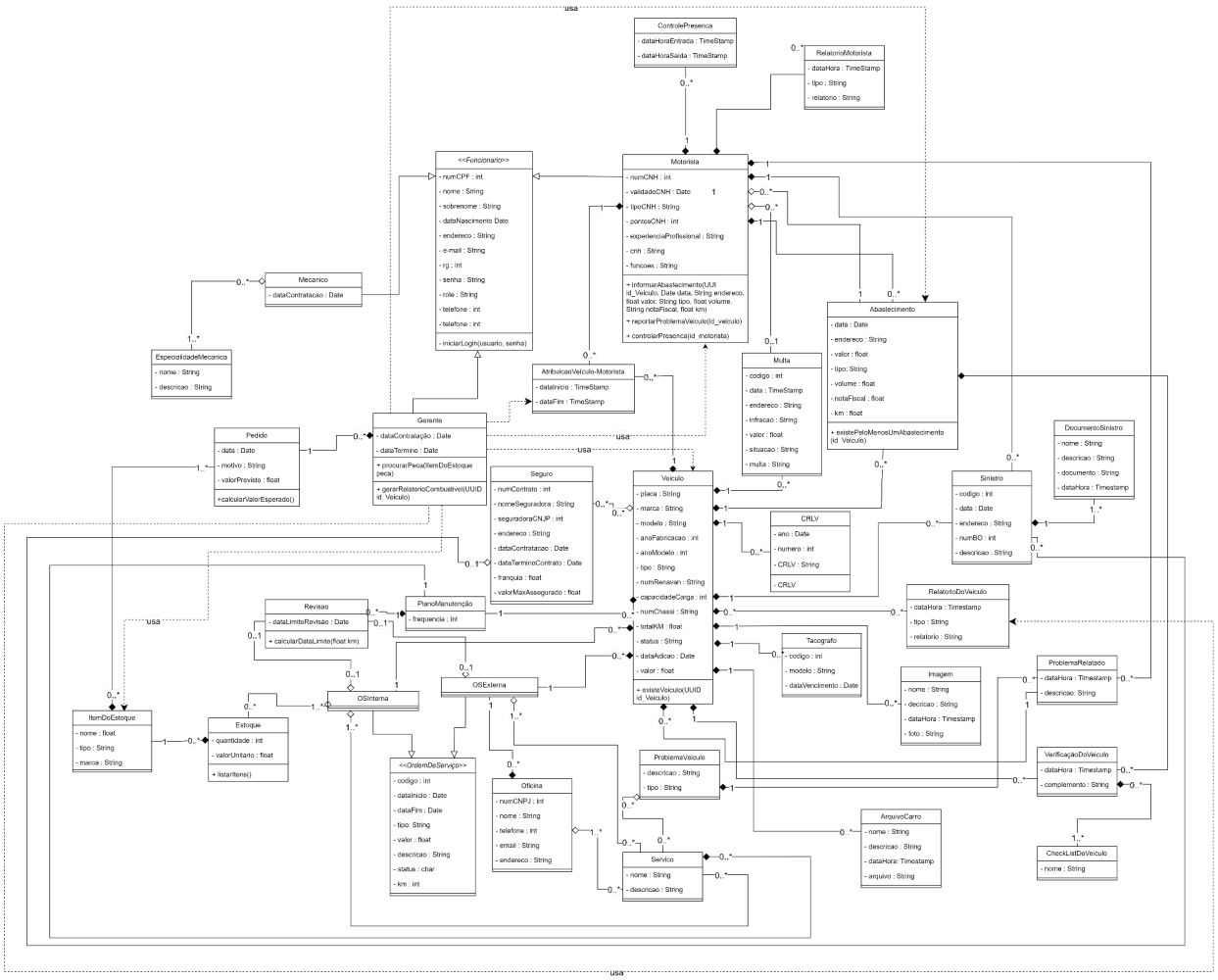


Imagem XXII *- Diagrama de Classe Detalhado*

Diagrama de Classes Detalhado (Refatorado)

O diagrama de classe detalhado será refatorado para incorporar os princípios SOLID e os padrões GRASP e os Padrões GoF, visando melhorar a qualidade do design do sistema. Essa refatoração será fundamental para garantir a coesão, o baixo acoplamento e a reusabilidade do código, facilitando sua manutenção e extensão futuras.

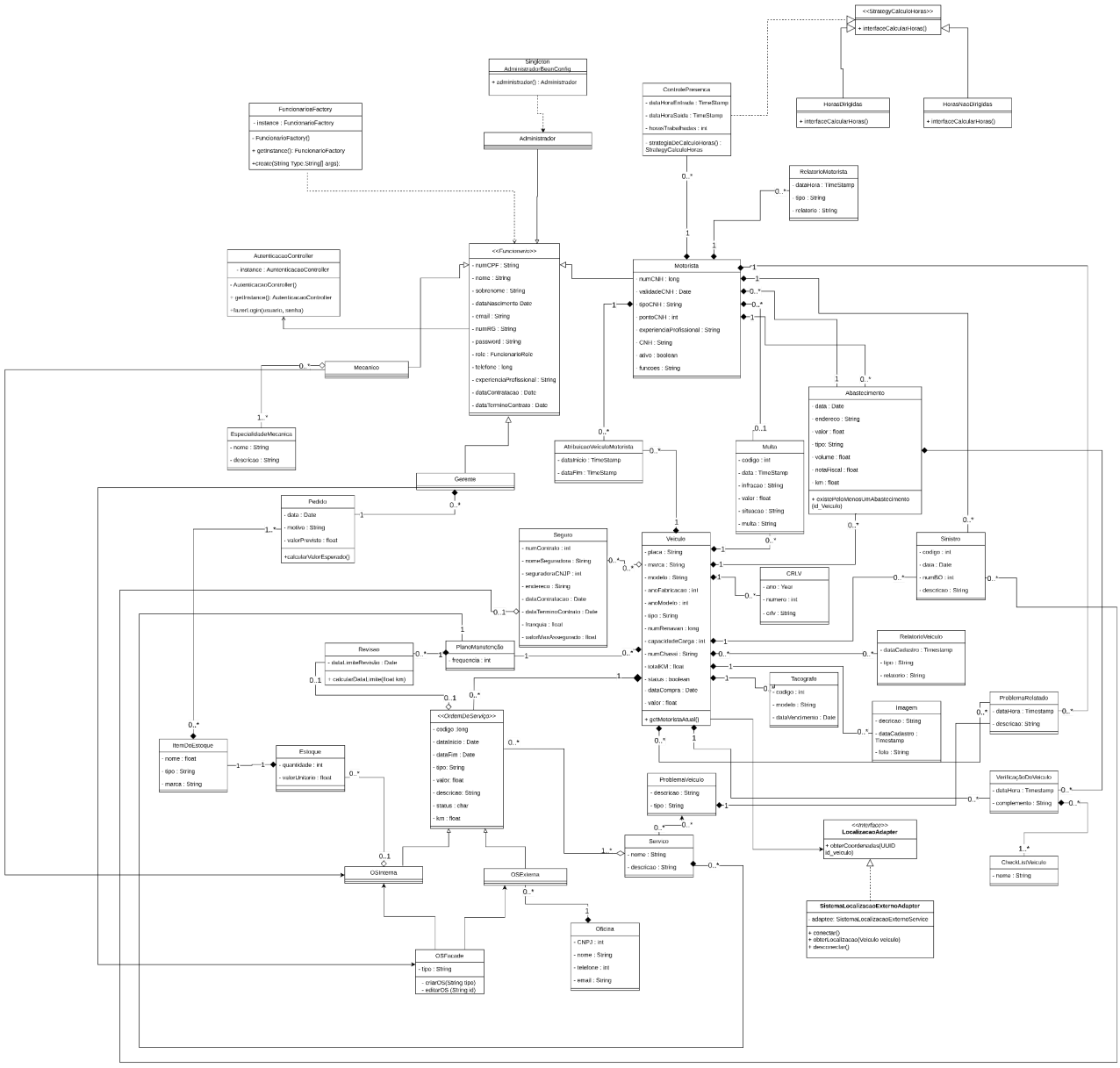


Imagem XXIII *- Diagrama de Classe Detalhado (Refatorado)*

Padrões Adicionados

Padrão	**Descrição**	**Utilização**
Singleton	Garante a existência de apenas uma instância de uma classe e oferece um ponto global de	Na classe Administrador, o padrão Singleton foi implementado para assegurar que apenas uma instância

Padrão	Descrição	Utilização
	acesso a essa instância.	de Administrador exista no sistema, garantindo que todas as operações relacionadas a administradores sejam gerenciadas de forma consistente por essa única instância. Isso evita duplicação de dados e centraliza o controle sobre as funcionalidades administrativas do sistema.
Strategy	Define uma família de algoritmos, encapsula cada um deles e permite que sejam intercambiáveis.	O padrão Strategy foi aplicado na classe ControlePresença, onde duas estratégias diferentes foram definidas para calcular as horas trabalhadas dos funcionários. Cada estratégia encapsula uma lógica específica de cálculo de horas, permitindo que a estratégia adequada seja selecionada dinamicamente com base nas condições ou requisitos específicos. Isso proporciona flexibilidade e reutilização de código na gestão de presença dos funcionários.
Facade	Fornece uma interface unificada para um conjunto de interfaces em um subsistema.	No contexto do sistema, o padrão Facade foi utilizado para criar uma fachada entre diferentes tipos de usuários (gerente de frota, gerente mecânico, mecânico) e as ordens de serviço (interna e externa). A fachada simplifica a interação entre usuários e ordens de serviço, encapsulando a complexidade subjacente e proporcionando uma interface única e simplificada para os usuários interagirem com o sistema de gestão de frota.
Adapter	Permite que interfaces incompatíveis trabalhem juntas através de um adaptador.	Um padrão Adapter foi implementado para integrar um sistema externo de localização em tempo real ao sistema de gestão de frota. O adaptador atua como uma ponte entre a interface do sistema externo e a interface requerida pelo

Padrão	Descrição	Utilização
		sistema interno, traduzindo e adaptando os dados recebidos de forma que possam ser utilizados sem alterar as interfaces existentes do sistema interno. Isso facilita a interoperabilidade e a integração de sistemas heterogêneos.
Factory	Define uma interface para criar um objeto, mas permite às subclasses decidirem qual classe instanciar.	Na criação de funcionários no sistema, o padrão Factory foi utilizado para encapsular a lógica de criação de diferentes tipos de funcionários. Cada fábrica concreta implementa a interface de fábrica e decide qual classe concreta de funcionário irá instanciar com base nos parâmetros ou requisitos específicos de criação. Isso promove a flexibilidade na criação de objetos e permite adicionar novos tipos de funcionários sem modificar o código existente.

Esses padrões foram escolhidos e implementados no diagrama de classe do projeto "Carga Explosiva" para promover a reutilização de código, a flexibilidade na adaptação a requisitos futuros e a simplificação da complexidade no sistema. Cada padrão oferece soluções específicas para os desafios encontrados na arquitetura e no design do software, seguindo as melhores práticas de engenharia de software.

Tecnologias Utilizadas

Para o desenvolvimento do sistema Carga Explosiva, foram selecionadas tecnologias e ferramentas específicas que garantem a eficiência, e manutenção do sistema. A escolha dessas tecnologias foi baseada em suas capacidades de atender aos requisitos funcionais e não funcionais do projeto, bem como na experiência da equipe de desenvolvimento. A seguir, apresentamos uma descrição detalhada das tecnologias utilizadas em cada parte do sistema.

Planejado Inicialmente

Antes do início do desenvolvimento do sistema Carga Explosiva, foram planejadas diversas tecnologias para atender aos requisitos de um sistema robusto e eficiente. Este plano inicial refletia a

intenção de utilizar tecnologias que oferecessem alta performance, facilidade de manutenção e uma experiência de usuário consistente tanto em dispositivos móveis quanto em desktops. As escolhas foram guiadas pela necessidade de desenvolver um sistema moderno que pudesse atender às demandas complexas do setor de transporte e logística.

Vamos detalhar as tecnologias e ferramentas que inicialmente foram consideradas para o projeto e como essas escolhas refletem as melhores práticas e tendências do mercado no momento do planejamento.

Área	Tecnologia	Descrição
Front-End (Cliente)	Flutter	Flutter é um framework open-source desenvolvido pelo Google, projetado para criar interfaces de usuário nativas para dispositivos móveis a partir de um único código base. Utilizando a linguagem Dart, Flutter oferece performance de alto nível e rapidez no desenvolvimento de interfaces responsivas e visualmente atrativas.
Back-End (Servidor)	Spring Boot	Spring Boot é um framework Java que simplifica o desenvolvimento de aplicações Java EE e microsserviços. Com configuração automática e um ambiente de execução pronto para uso, Spring Boot oferece produtividade e facilidade na criação de sistemas robustos.
Banco de Dados	MySQL	MySQL é um sistema de gerenciamento de banco de dados relacional open-source, conhecido por sua confiabilidade, desempenho e ampla compatibilidade com diversas tecnologias de back-end, tornando-o ideal para aplicações web e corporativas.

Utilizadas no Desenvolvimento

Durante o desenvolvimento do sistema Carga Explosiva, foram feitas adaptações estratégicas para garantir eficiência e alinhamento às necessidades do projeto. Inicialmente planejado para utilizar Flutter tanto para interfaces de usuário de desktop quanto para aplicativos móveis, a decisão

de implementar o front-end com React foi motivada por considerações de tempo e agilidade no desenvolvimento. Posteriormente, reconhecendo a necessidade de performance, foi mantido o uso de Spring Boot para o back-end, alinhando-se com as tecnologias previstas. Esta abordagem assegura não apenas a continuidade dos padrões de qualidade e segurança, mas também preparou o caminho para futuras iterações que poderão incluir a adoção de Flutter para expandir as funcionalidades do sistema em plataformas móveis e desktops.

Front-End (Cliente)

Para a camada de front-end do sistema Carga Explosiva, foi escolhido inicialmente o Flutter para desenvolver interfaces de usuário tanto para desktop quanto para aplicativos móveis. No entanto, devido a restrições de tempo e considerando a familiaridade da equipe com o ecossistema React, a implementação foi realizada utilizando React. Esta escolha se mostrou eficiente para garantir um desenvolvimento ágil e a entrega de uma interface dinâmica e responsiva.

Tecnologia	**Descrição**
React	React é um framework JavaScript amplamente adotado para o desenvolvimento de interfaces de usuário dinâmicas e responsivas. Utilizando uma abordagem baseada em componentes, React permite dividir a interface em elementos reutilizáveis, chamados de componentes. Isso não apenas facilita a organização e manutenção do código, mas também promove uma maior modularidade do sistema. Além disso, o React utiliza um modelo de programação declarativo, o que simplifica a atualização e renderização eficiente da interface com base nas mudanças de estado da aplicação.
Zod	Zod é uma biblioteca utilizada para validar esquemas de dados em JavaScript/TypeScript. No contexto do projeto Carga Explosiva, Zod desempenha um papel crucial ao garantir que os dados manipulados pela aplicação estejam em conformidade com as expectativas do sistema. Ele permite definir e validar estruturas de dados de maneira precisa, assegurando a integridade e consistência dos dados em todas as interações do usuário com o sistema.
React-Hook-Form	React-Hook-Form é uma biblioteca que simplifica o gerenciamento de formulários no

Tecnologia	Descrição
	React. Ela utiliza hooks do React para oferecer uma solução eficiente e de alto desempenho na captura e validação de dados de entrada dos usuários. Ao utilizar React-Hook-Form, desenvolvedores podem criar formulários complexos de forma simplificada, garantindo ao mesmo tempo uma experiência de usuário fluida e responsiva.
React-Router-Dom	React-Router-Dom é uma biblioteca que facilita a navegação entre diferentes componentes em um aplicativo React. Ela permite definir rotas de maneira declarativa, associando cada rota a um componente específico. Isso é essencial para estruturar e gerenciar a navegação dentro do sistema Carga Explosiva, possibilitando aos usuários acessarem diferentes páginas e fluxos de maneira intuitiva e eficiente.
Axios	Axios é uma biblioteca amplamente utilizada para realizar requisições HTTP baseadas em promises tanto no navegador quanto no Node.js. No contexto do projeto, Axios é essencial para a comunicação entre o front-end desenvolvido em React e o back-end implementado em Spring Boot. Ele facilita a integração entre as camadas de front-end e back-end, permitindo o envio e recebimento de dados de forma síncrona e assíncrona de maneira eficiente e segura.
Bootstrap	Bootstrap é um framework front-end que oferece uma coleção de componentes e estilos predefinidos, permitindo aos desenvolvedores criar interfaces responsivas e esteticamente agradáveis de maneira rápida e eficiente. Utilizando o Bootstrap no projeto Carga Explosiva, a equipe pôde aproveitar componentes como grids, botões, formulários e outros elementos visuais prontos para uso, economizando tempo no desenvolvimento e garantindo consistência visual em toda a aplicação.

Tecnologia	Descrição
FontAwesomeIcons	FontAwesomeIcons é um conjunto de ícones vetoriais que adicionam elementos visuais significativos ao sistema. Com uma vasta biblioteca de ícones disponíveis, FontAwesomeIcons permite aos desenvolvedores integrar ícones facilmente em interfaces de usuário, melhorando a usabilidade e a experiência do usuário final. Esses ícones são utilizados de forma a complementar a interface do sistema Carga Explosiva, tornando a navegação e interação mais intuitivas e visualmente agradáveis.

Back-End (Servidor)

A camada de Back-End do sistema Carga Explosiva foi implementada com foco na robustez, segurança, utilizando o framework Spring Boot e diversas tecnologias associadas. Inicialmente planejado para integrar perfeitamente com o front-end desenvolvido em Flutter, a mudança para React devido a restrições de tempo não afetou a escolha e implementação das ferramentas no Back-End. Este ambiente foi projetado para suportar as operações críticas de gerenciamento de frota, oferecendo um sistema confiável e eficiente para empresas de transporte e logística.

Para a implementação do Back-End, foi escolhido o Spring Boot, um framework Java amplamente reconhecido por sua simplicidade e eficiência no desenvolvimento de aplicações empresariais. **Spring Boot** simplifica o desenvolvimento de aplicações Java ao oferecer uma configuração pronta para o uso, baseada na convenção sobre configuração. Ele promove uma abordagem modular e robusta, facilitando a criação de microserviços e aplicações de grande escala. A escolha do Spring Boot para o Carga Explosiva foi guiada pela sua capacidade de integrar-se facilmente com uma variedade de tecnologias e bibliotecas, permitindo um desenvolvimento ágil e eficiente. Junto ao Spring Boot foram utilizadas outras tecnologias/ferramentas para ajudar no desenvolvimento.

Tecnologias	Descrição
Spring Web	Spring Web, parte integrante do ecossistema Spring, foi utilizado para criar aplicativos web baseados em Spring MVC. Esta tecnologia proporciona suporte completo para desenvolver endpoints RESTful, essenciais para a comunicação entre o front-end e o back-end no sistema Carga Explosiva.

Tecnologias	Descrição
Lombok	Lombok é uma biblioteca Java que reduz o código boilerplate (repetitivo) em classes POJO (Plain Old Java Objects). Ele automatiza a geração de getters, setters, construtores e outros métodos padrões, permitindo que os desenvolvedores foquem mais na lógica de negócios e menos na escrita de código repetitivo.
MySQL Driver	Para conectar o Spring Boot ao banco de dados MySQL, foi utilizado o MySQL Driver. Este driver JDBC oferece uma interface para estabelecer conexões seguras e eficientes com o banco de dados, garantindo que as operações de persistência de dados sejam executadas de forma confiável e otimizada.
Spring Data JPA	Spring Data JPA facilita a implementação de repositórios baseados em JPA (Java Persistence API) no Spring. Ele simplifica o acesso e manipulação de dados no banco de dados através de consultas declarativas, reduzindo a complexidade do código e melhorando a manutenibilidade da aplicação.
Spring Security	Spring Security foi adotado para oferecer recursos avançados de autenticação e controle de acesso ao sistema "Carga Explosiva". Esta poderosa ferramenta permite configurar políticas de segurança robustas, como autenticação baseada em tokens, autorizações granulares e proteção contra vulnerabilidades comuns de segurança.
JWT Tokens	JWT (JSON Web Tokens) Tokens são utilizados para autenticação stateless no sistema, gerenciando tokens de acesso seguro entre o front-end e o back-end. Esta abordagem permite que o sistema Carga Explosiva mantenha a integridade e a segurança das sessões de usuário de maneira eficiente.
Spring Validation	O Spring Validation é utilizado para validar dados de entrada no servidor, garantindo que

Tecnologias	Descrição
	apenas dados válidos sejam processados e armazenados. Esta ferramenta é essencial para garantir a integridade dos dados manipulados pelo sistema Carga Explosiva e para proporcionar uma experiência consistente aos usuários finais.

Esta combinação de tecnologias no Back-End garante que o sistema "Carga Explosiva" seja capaz de atender às exigências de desempenho e segurança necessárias para operações críticas de gestão de frota, proporcionando uma experiência confiável e eficiente para os usuários finais.

Bando de Dados

O banco de dados escolhido para o sistema Carga Explosiva foi o MySQL, um dos sistemas de gerenciamento de banco de dados relacional mais utilizados no mundo. O MySQL oferece uma combinação robusta de desempenho e confiabilidade, tornando-o ideal para aplicações empresariais como a gestão de frota. Sua estrutura relacional permite armazenar e gerenciar dados de forma eficiente, enquanto suas capacidades de escalabilidade horizontal e vertical garantem que o sistema possa crescer conforme a demanda. A escolha do MySQL foi fundamentada em sua ampla adoção pela comunidade, suporte robusto e integração perfeita com o ecossistema de desenvolvimento do Spring Boot, garantindo assim uma base de dados estável e otimizada para as necessidades do sistema Carga Explosiva.

Ambiente de Desenvolvimento

No processo de desenvolvimento do sistema Carga Explosiva, foram selecionadas ferramentas essenciais para garantir eficiência, colaboração e controle sobre o código-fonte e seu ciclo de vida. A seguir, detalhamos o ambiente de desenvolvimento utilizado:

IDE IntelliJ: O IntelliJ foi escolhido como ambiente de desenvolvimento integrado (IDE) para o projeto, oferecendo uma plataforma robusta para desenvolver tanto o front-end quanto o back-end. Com suporte avançado para linguagens como Java e JavaScript, o IntelliJ proporciona ferramentas poderosas para escrita de código, depuração, refatoração e gerenciamento de projetos. Sua interface intuitiva e extensibilidade garantem uma experiência de desenvolvimento fluente e produtiva para toda a equipe.

Git: Utilizado como sistema de controle de versão distribuído, o Git desempenha um papel fundamental no gerenciamento do código-fonte do projeto "Carga Explosiva". Através do Git, os desenvolvedores podem colaborar de forma eficaz, rastrear alterações no código e coordenar o trabalho em equipe de maneira organizada.

Linha de Produção de Software

A linha de produção de software do sistema Carga Explosiva é cuidadosamente projetada para garantir um desenvolvimento eficiente e modular, permitindo a personalização e escalabilidade do sistema conforme as necessidades específicas das empresas de transporte e logística. O diagrama a seguir ilustra a estrutura da linha de produção do software, que inclui características mandatórias, opcionais e alternativas, facilitando a compreensão das diferentes partes e funcionalidades do sistema.

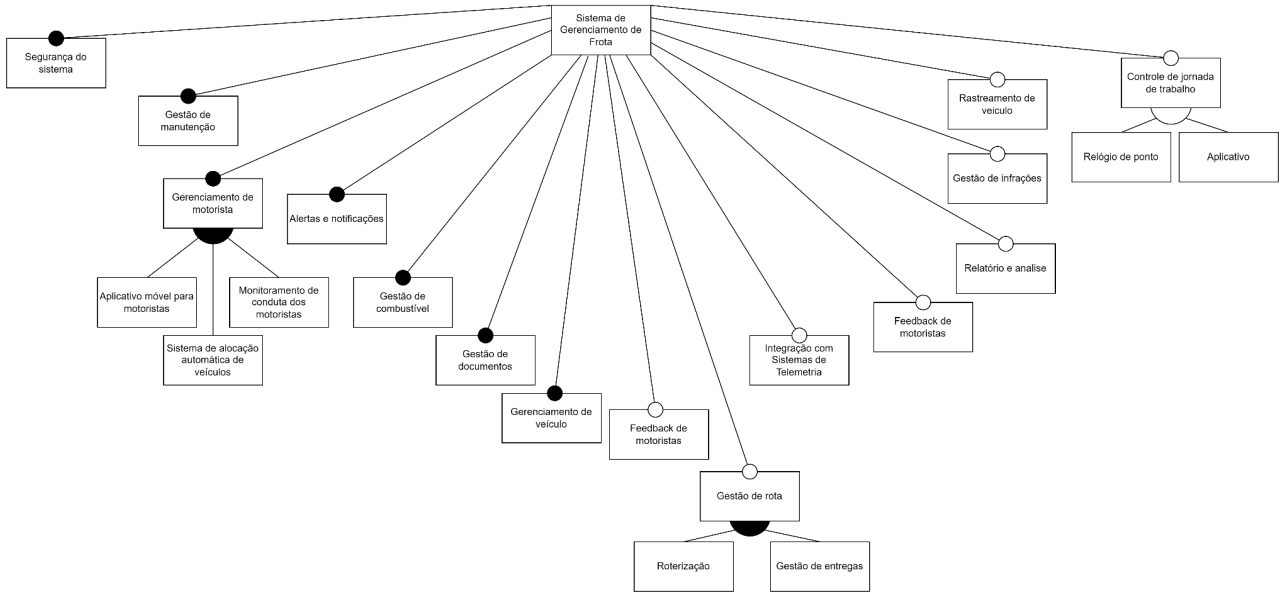


Imagem XVII *- Diagrama de Linha de Produção de Software para Sistema de Gestão de Frota*

Dimensionamento e Performance

Esta seção aborda as considerações cruciais relacionadas ao dimensionamento e desempenho do sistema de Carga Explosiva. Esses aspectos são essenciais para garantir que o sistema atenda às demandas operacionais esperadas, oferecendo uma experiência de usuário satisfatória e eficiência operacional.

Volume

Para o sistema de Carga Explosiva, é essencial entender o volume previsto de usuários, acessos diários e mensais. Essas métricas são fundamentais para dimensionar a infraestrutura e garantir que o sistema mantenha um desempenho consistente e responsivo, atendendo às demandas operacionais

sem comprometer a experiência do usuário. A análise cuidadosa desses dados orienta a implementação de estratégias de otimização e escalabilidade, assegurando a eficiência operacional em todas as etapas do projeto.

- Número estimado de usuários: Entre 100 e 500 usuários simultâneos.
- Número estimado de acessos diários: Aproximadamente 70% dos usuários acessam diariamente.
- Número estimado de acessos por mês: Aproximadamente 8.000 acessos mensais.
- Tempo de seção de um usuário: Em média, os usuários mantêm sessões ativas por até 8 horas.

Performance

Para garantir uma experiência de usuário satisfatória e a eficiência do sistema, são estabelecidos os seguintes objetivos de desempenho:

- **Tempo máximo de resposta para operações críticas**: As transações críticas do sistema, como cadastro de veículos, atribuição de motoristas e registro de manutenções, devem ter um tempo máximo de resposta de até 3 segundos para garantir a responsividade do sistema.
- **Tempo máximo de processamento de relatórios**: A geração de relatórios personalizáveis sobre o desempenho da frota e das operações de manutenção deve ser concluída dentro de um intervalo de tempo aceitável, com um tempo máximo de processamento de 10 segundos para relatórios típicos e 20 segundos para relatórios mais complexos.
- **Otimização de consultas e operações de banco de dados**: Devem ser implementados práticas de otimização de consultas e operações de banco de dados para garantir uma recuperação eficiente de dados e minimizar os tempos de resposta em todas as interações com o banco de dados.

Qualidade

Nesta seção, mostra como a arquitetura de software contribui para diversos aspectos de qualidade do sistema, incluindo extensibilidade, confiabilidade, portabilidade, segurança e outros.

Item	Descrição
Extensibilidade	A arquitetura do sistema foi projetada com extensibilidade em mente, permitindo que novas funcionalidades sejam facilmente adicionadas e integradas ao sistema existente. Isso é alcançado por meio da modularidade da arquitetura em camadas que facilita a identificação e isolamento de componentes específicos para

Item	Descrição
	modificação ou substituição sem afetar outras partes do sistema. Além disso, a adoção de padrões arquitetônicos, como a separação clara entre a lógica de negócios e a apresentação, facilita a extensão do sistema sem comprometer sua estabilidade ou integridade.
Confiabilidade	A confiabilidade do sistema é garantida por meio de várias medidas arquiteturais. A separação em camadas facilita o isolamento de falhas e a implementação de estratégias de recuperação de falhas em níveis específicos da arquitetura. Além disso, a redundância de componentes críticos e a implementação de mecanismos de monitoramento e registro de erros contribuem para a detecção precoce e a mitigação de falhas, garantindo a disponibilidade contínua do sistema.
Portabilidade	A arquitetura do sistema foi projetada para garantir sua portabilidade em diferentes plataformas e ambientes de execução, incluindo desktops e dispositivos móveis. Isso é alcançado por meio da adoção de tecnologias e padrões de desenvolvimento que são compatíveis com múltiplas plataformas, permitindo que o sistema seja executado de forma consistente e eficiente em diferentes ambientes. Além disso, a separação clara entre a lógica de negócios e a apresentação facilita a adaptação do sistema a diferentes requisitos e restrições de plataforma.
Segurança	A segurança do sistema é uma prioridade e é abordada em várias camadas da arquitetura. Medidas de segurança, como autenticação e autorização de usuários, são implementadas tanto no cliente quanto no servidor para proteger os dados contra acessos não autorizados. Além disso, a criptografia de dados em trânsito e em repouso, juntamente com implementação de práticas de segurança recomendadas, garantem a integridade e confidencialidade das informações do sistema.
Usabilidade	A usabilidade do sistema é aprimorada pela arquitetura centrada no usuário, que prioriza a criação de interfaces intuitivas e responsivas. A separação entre a lógica de apresentação e a lógica de negócios permite o desenvolvimento de interfaces de usuário flexíveis e personalizáveis, adaptadas às necessidades e preferências dos usuários finais. Além disso, a implementação de práticas de design centradas no usuário garante uma experiência de uso positiva, promovendo a eficiência e a satisfação do usuário.
Manutenibilidade	A manutenibilidade e sistema é garantida pela modularidade da

Item	Descrição
	arquitetura, que facilita a identificação e correção de defeitos, bem como a implantação e a adoção de padrões de codificação consistentes facilitam a compreensão e a manutenção do código-fonte, garantindo a estabilidade e a evolução contínua do sistema ao longo do tempo.

A qualidade do sistema é fundamental para garantir sua eficácia, confiabilidade e usabilidade. Através da definição de requisitos de qualidade claros e mensuráveis, podemos assegurar que o sistema atenda às expectativas dos usuários e às necessidades do negócio. Ao adotar práticas de desenvolvimento de software robustas, realizar testes rigorosos e implementar processos de garantia de qualidade eficazes, podemos mitigar riscos, identificar e corrigir defeitos precocemente, e fornecer um produto final de alta qualidade. Investir na qualidade desde as fases iniciais do projeto é essencial para evitar retrabalho, garantir a satisfação do cliente e alcançar o sucesso a longo prazo do sistema.

Documentação

Nesta seção, você encontrará os links para os documentos online que complementam e detalham as informações apresentadas neste documento de arquitetura do sistema Carga Explosiva.

Documento	Link
Pasta do Trabalho	Carga Explosiva
Documento de Arquitetura de Software	Documento de Arquitetura de Software.d…
Diagrama de Casos de Uso	Documento Draw.io
Diagrama de Classe de Análise	Documento Draw.io
Diagramas de Sequência do Sistema	Documento Draw.io
Diagramas de Interação	Documento Draw.io
Diagrama de Classe Detalhado	Documento Draw.io
Diagrama de Visão Geral da Arquitetura	Documento Draw.io
Diagrama de Linha de Produção de Software	Documento Drow.io