**Documento de Visão de Projeto**



**Fiel Leitor**

**Histórico de Versões**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Data** | **Versão** | **Descrição** | **Autor** | **Revisor** |
| 29/09/25 | 1.0 | Modelagem e desenvolvimento | Mahmoud | - |

|  |  |
| --- | --- |
| **Cliente** | FATEC - Interno |
| **Documento** | Documento de Visão de Projeto: *Fiel Leitor* |
| **Data** | 29 de setembro de 2025 |
| **Autor** | **Mahmoud Fauzi Ghazal**  mahmoud.f.ghazal@gmail.com |

**Página de Assinaturas**

|  |  |  |
| --- | --- | --- |
| Revisado e Aprovado por: |  |  |
|  |  | 29/09/25 |

**Índice**

Camada de Apresentação 7

Camada de Persistência 11

Diagrama de caso de uso geral do sistema 5

Escopo 4

Objetivo 4

Pacote Model 10

Qualidade 13

Realização dos Casos de Uso Significativos 11

Representação Arquitetural 4

Tamanho e Performance 13

Visão de Dados 13

Visão de Implementação 13

Objetivo

Este documento trata principalmente da documentação das necessidades de negócios, da justificativa do projeto, do entendimento atual das necessidades do cliente e descreve resumidamente o novo produto, serviço ou resultado que deve satisfazer esses requisitos.

Tem o objetivo de alinhar as expectativas dos interessados para formalizar o início do projeto. Apresentar uma visão arquitetural dos sistemas da linha **SISCAD**. O intuito é salientar diferentes aspectos deste produto, obtidos a partir de decisões arquiteturais realizadas no âmbito dos sistemas da linha **SISCAD**.

* 1. Escopo

O escopo deste documento trata do desenvolvimento de um módulo que atenda todas as necessidades de uma biblioteca abrangendo qualquer instituição de ensino, sendo que este módulo é uma parte de um sistema maior.

O escopo deste documento é documentar as partes significativas do ponto de vista da arquitetura do modelo de design, como sua divisão em subsistemas e pacotes. Além disso, mostra sua divisão em classes e utilitários de classe.

* 1. Referências

Listar documentos relacionados a este documento de visão. Podem ser tanto documentos que foram utilizados para confeccionar este documento ou fazem parte deste documento, como documentos que estendem este documento.

Para a construção deste documento foram utilizadas as seguintes referências:

* Documentos DVP Exemplo fornecido pelo professor Rodrigo Rocha.

Este documento influencia os seguintes documentos:

* Documento de Requisitos

Necessidades de Negócio

O sistema de e-commerce para venda de livros é necessário para disponibilizar o catálogo de obras de forma organizada, permitindo pesquisas, compras online seguras e acompanhamento de pedidos. Além disso, deve oferecer controle de estoque, geração de relatórios de vendas e integração com meios de pagamento e entrega, garantindo eficiência operacional e melhor experiência ao cliente.

Objetivo do Projeto

Desenvolver uma plataforma para soluções web capaz de:

* armazenar informações em uma base de dados
* utilizar o protocolo HTTP

Facilitando a compra de livros pelos clientes, agilizando o processamento de pedidos e pagamentos, e permitindo à administração acompanhar vendas e controlar o estoque de forma eficiente.

Declaração Preliminar de Escopo

Esta seção descreve, em alto nível, o escopo do projeto. Os requisitos serão melhor detalhados nos documentos de Requisitos e Dicionário WBS.

* 1. Descrição

Descrição narrativa do produto do projeto.

* 1. Produtos a serem entregues

Os seguintes itens são considerados produtos do projeto, na sua etapa 1.

* Sistema do módulo de catalogo, etapa 1, implementado de acordo com a especificação feita na fase de análise. (código objeto e código fonte).
* Documentos de especificação do sistema, concebido na fase de elaboração.
* Plataforma Web funcional.
  1. Requisitos

É importante salientar que o documento de visão estará sendo feito no momento de iniciação do projeto. No qual, ainda não se tem todo o detalhamento. Portanto, os requisitos devem descritos em alto nível e depois detalhados em outro documento, caso o projeto seja aprovado.

* + 1. Requisitos Funcionais

O sistema deve ser capaz de efetuar o cadastro, exclusão, alteração e consulta de livros do acervo da biblioteca.

O sistema deve ser capaz de efetuar o controle dos empréstimos.

O sistema deve ser capaz de fazer o controle de atraso na devolução de livros que foram vendas a um aluno ou professor.

O sistema deve ser capaz de produzir relatórios referentes as operações do sistema.

O sistema deve ser capaz de realizar a consulta a históricos.

* + 1. Requisitos Não Funcionais

Utilizar linguagem Java

Utilizar o banco de dados PostGreSQL

A arquitetura da solução obedecer ao design pattern MVC.

O sistema deve rodar nos seguintes browsers:

* Google Chrome

Premissas

As premissas são informações que consideramos verdadeiras ou reais, e que vão sendo elaboradas gradativamente no decorrer do processo de Planejamento. Elas são documentas inicialmente no documento de visão e serão usadas como entradas em outros processos mais à frente.

Premissas são coisas que o gerente de projeto toma como verdade para basear o seu planejamento. Uma premissa considerada, mas não documentada, pode ocasionar o insucesso do projeto.

* O projeto será orientado pelo professor Rodrigo Rocha.

Influência das Partes Interessadas

Os interessados no projeto podem influenciar fortemente os objetivos do projeto. Muitos deles têm forte influência negativa e podem jogar areia no seu projeto. Na fase de Iniciação é importante que você possa identificar quais são as forças que poderão contribuir com seu projeto e quais serão as negativas para que você possa identificar formas de neutralizá-las. Procure conversar com os stakeholders do projeto e obter detalhes mais específicos sobre os objetivos e resultados pretendidos. Nessa etapa, o objetivo é obter uma visão geral do projeto, ou seja, a definição — que deve conter informações suficientes para descrever tanto o projeto, seus requisitos e objetivos, quanto como saber o que deve ser feito para ser concluído com êxito. Resumindo quais são os fatores críticos de sucesso de seu projeto.

* **FATEC Mogi das Cruzes:** instituição responsável pelo curso, patrocinadora do projeto. Interessada em avaliar a competência do aluno Mahmoud Ghazal na execução do projeto e promover aprendizado prático em desenvolvimento de sistemas de e-commerce.
* **Rodrigo Rocha:** avaliador do projeto. Interessado em acompanhar o desempenho do aluno, verificando se os objetivos do projeto foram atingidos e se as boas práticas de desenvolvimento foram aplicadas corretamente.
* **Mahmoud Ghazal:** aluno responsável pelo projeto. Interessado em demonstrar suas habilidades no desenvolvimento de uma plataforma de venda de livros, aplicando conhecimentos teóricos na prática e aprendendo com a experiência.

Representação Arquitetural

O sistema será desenvolvido a partir de uma arquitetura multicamadas que adota princípios dos padrões de projeto GoF e J2EE, com separação clara entre camadas de apresentação, negócio e persistência de dados.

A arquitetura tem como objetivo garantir manutenibilidade, reuso de componentes e desacoplamento entre as partes do sistema.

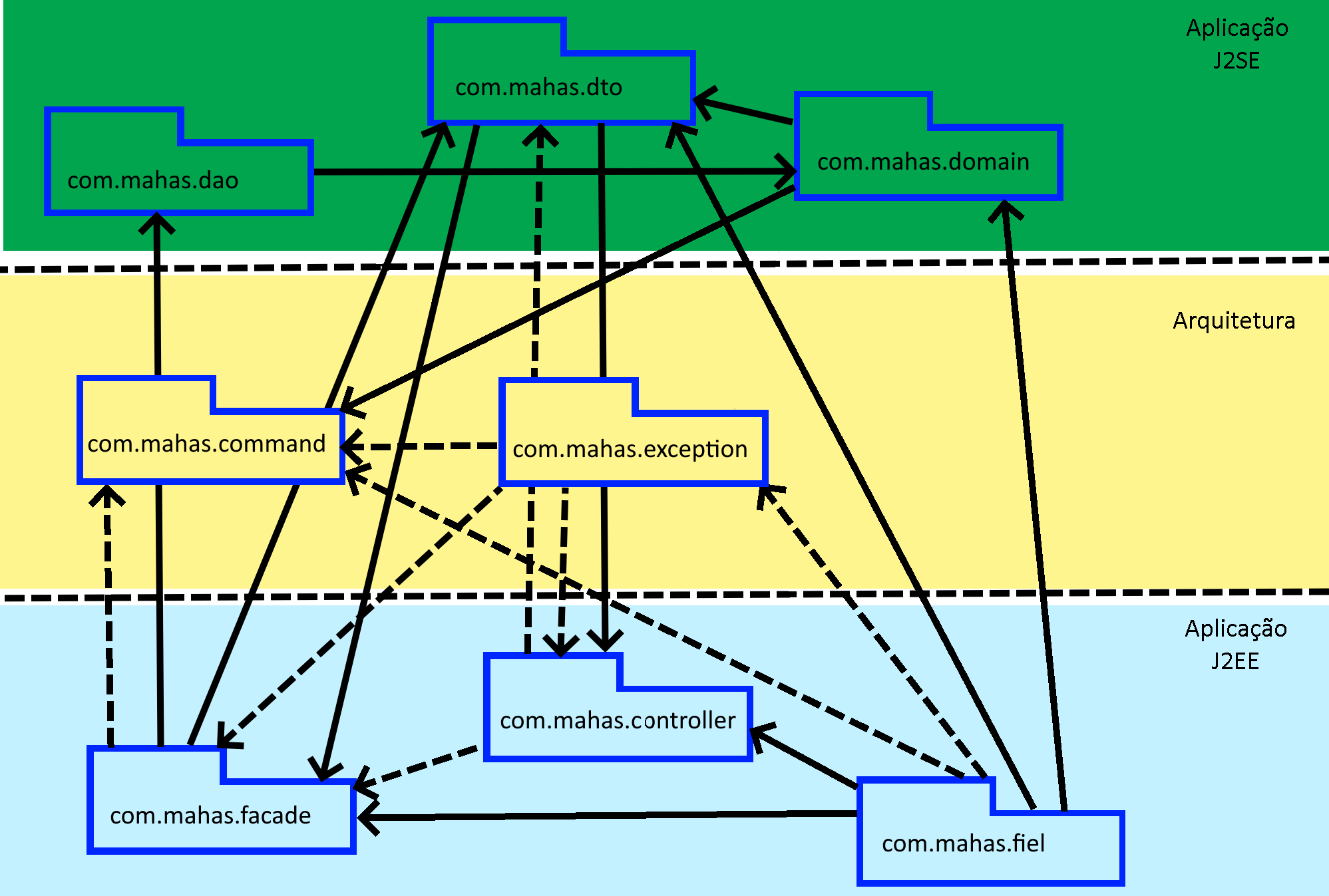
O **Container WEB** será responsável pela camada de apresentação do sistema. Diferente da arquitetura tradicional baseada em JSF, neste projeto a camada de apresentação será implementada utilizando o **framework Next.js (React),** que segue o modelo **MVC** no front-end, garantindo a separação entre **View (UI), Controller (tratamento de eventos e requisições)** e **Model (representação dos dados consumidos da API).**

As **Classes de Domínio** representam os **Value Objects**, contendo apenas atributos e métodos de acesso (getters/setters), e são utilizadas para transportar informações entre as diferentes camadas do sistema.

As **Classes de Negócio** estarão representadas na camada de **Fachada (Facade).** Essa camada é responsável pela aplicação das regras de negócio, como, por exemplo, cadastrar usuários ou processar transações. A Fachada também estabelece o relacionamento com os **DAOs**, que têm a responsabilidade de persistir e recuperar os objetos no banco de dados.

A arquitetura proposta também dá suporte a aplicações **J2SE**, permitindo a reutilização das camadas de Fachada e DAO sem necessidade de execução em um servidor de aplicações. Entretanto, aplicações **J2EE** podem ser executadas em servidores como **WildFly, GlassFish ou Payara**, garantindo escalabilidade e suporte a distribuições mais complexas.

O diagrama a seguir (Figura 2) representará a organização das classes dentro dos pacotes, tanto da arquitetura quanto da aplicação que fará uso dessa arquitetura. É importante atentar para a padronização de nomes de pacotes e para a separação entre aplicação e infraestrutura, bem como para o tipo de aplicação que se pretende desenvolver, **J2SE ou J2EE**.



**Figura 2 - Diagrama em pacotes**

É importante ressaltar que o uso da arquitetura para aplicações **J2SE** não implica em importar para a aplicação o pacote com.mahas.fiel.controller de interface web, mas o uso de uma aplicação **J2EE** implica em importar pacotes específicos de comunicação entre camadas (com.mahas.fiel.facade e com.mahas.fiel.dao), garantindo a correta utilização do padrão arquitetural proposto.

* 1. Restrições Arquiteturais

Foram identificadas algumas orientações / restrições pertinentes ao desenvolvimento deste subsistema:

* Utilização do JDK 21 do Java;
* Utilização do framework Spring para desenvolvimento da camada de negócio e controle de transações;
* Utilização do framework de mapeamento objeto-relacional Hibernate;
  1. Objetivos e Restrições Arquiteturais

Alguns requisitos registrados que impactam diretamente a arquitetura do sistema fiel são:

* A consideração das bases de dados Sistemas Legados do MEC como fontes de dados de autenticação;

Considerando premissas definidas para os Sistemas fiel pode-se citar as seguintes restrições:

* Utilização da Linguagem Java
* Considerar a utilização de software Livre, quando possível
* Necessidade de comunicação com sistemas legados do MEC
* O Sistema de Gerenciamento de Banco de Dados a ser considerado em implementações de âmbito corporativo será o SGBD Oracle.

1. Visão de Use Case

Esta seção apresenta os Casos de Uso arquiteturalmente significativos, que foram selecionados considerando-se o pacote do Modelo de Casos de Uso que representa o sistema fiel.

A classificação dos casos de uso, em termos de significância, foi realizada com base na observação de pelo menos um dos seguintes critérios:

* Casos de uso que estendem outros Casos de Uso
* Casos de Uso que são incluídos em outros Casos de Uso e
* Casos de uso que acessam sistemas externos
  1. Diagrama de Caso de Manter Dados de Atributos

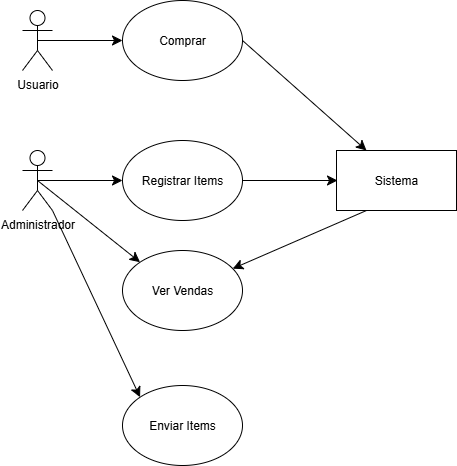


Figura 4.1 Diagrama de Caso de Uso de Fiel Leitor

* 1. Descrição dos Casos de Uso Arquiteturalmente Significativos

**Enviar Vendas**

Esse caso de uso envolve o envio de informações de vendas realizadas para um sistema externo (como um sistema de faturamento, ERP ou servidor remoto). Pode incluir integração com APIs, autenticação, tratamento de falhas de rede e persistência de estado.

**Ver Vendas**

Permite aos usuários consultar vendas realizadas, com filtros como data, cliente ou status. Pode incluir paginação, ordenação e visualização de detalhes de cada venda.

**Registrar Item**

Permite aos usuários consultar vendas realizadas, com filtros como data, cliente ou status. Pode incluir paginação, ordenação e visualização de detalhes de cada venda.

**Comprar**

Esse é o caso de uso principal do ponto de vista do usuário final, envolvendo a escolha de itens, adição ao carrinho, cálculo de totais, aplicação de descontos e finalização da compra.

1. Visão de Lógica

Esta visão apresenta elementos de design significativos do ponto de vista da arquitetura, descrevendo a organização do Sistema file em pacotes, bem como a organização desses pacotes em camadas.

O Diagrama com as camadas do sistema fiel é ilustrado na figura 5.1.

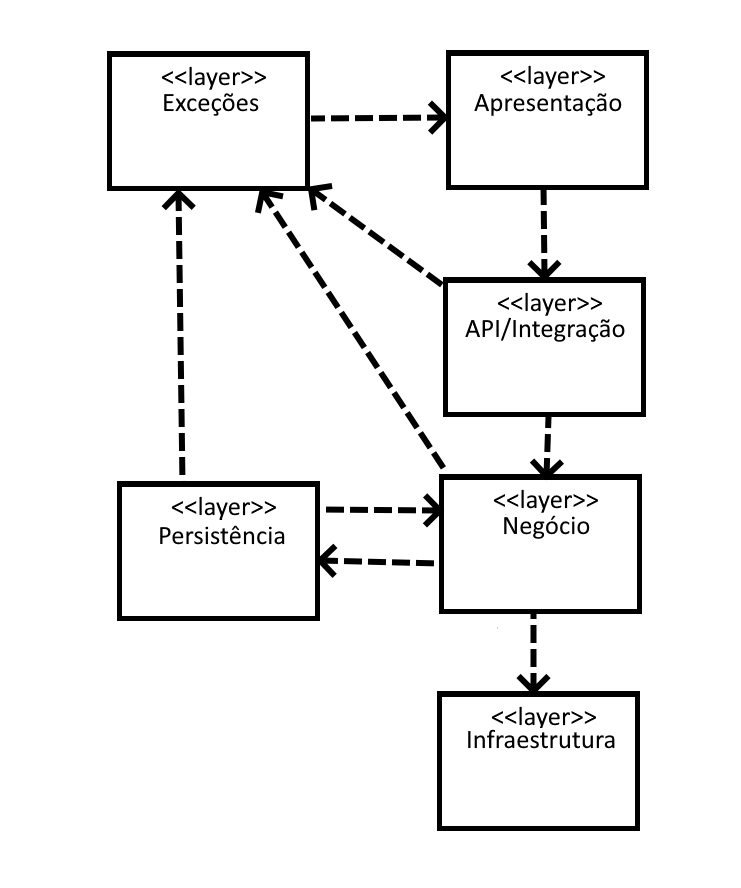


Figura 5.1 – Diagrama de camadas do Fiel

**Apresentação**: Contém as interfaces gráficas implementadas em **Next.js**, através das quais os usuários interagem com o Fiel. Essa camada envia e recebe dados do backend por meio de chamadas a APIs REST.

**API/Integração:** Contém os controllers do Spring Boot, responsáveis por receber as requisições do frontend e encaminhá-las à camada de Negócio, retornando as respostas em formato JSON.

**Negócio**: Contém as classes que centralizam a execução das funcionalidades do Fiel, incluindo fachadas e comandos que implementam as regras de negócio.

**Persistência**: Contém as classes responsáveis por armazenar e recuperar os dados, incluindo entidades de modelo, objetos de transporte (DTOs) e repositórios (DAOs) conectados ao banco de dados relacional.

**Infraestrutura:** Contém a classe principal do sistema, que inicializa o Spring Boot, configura os pacotes e gerencia o ciclo de vida da aplicação.

**Exceções:** Contém classes de exceção personalizadas utilizadas de forma transversal em todas as camadas.

* 1. Camada de Apresentação

Nesta camada, temos os pacotes que contêm todos os arquivos relacionados à interface com o usuário, implementados em Next.js. Isso inclui páginas React/JSX para exibição de dados e navegação entre telas, CSS/SCSS ou módulos de estilo para layout e aparência, imagens e ícones usados na interface, e TypeScript para lógica de interface e tipagem. Além disso, está camada interage com a API/Integração do backend, representada pelos Controllers do Spring Boot, que recebem as requisições HTTP do frontend e chamam a camada de Negócio.

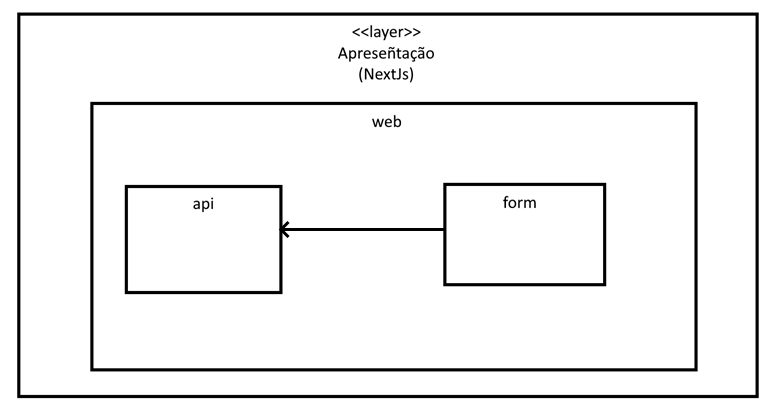


Figura 5.3: Camada de Apresentação

* 1. Camada de API/Integração

Nesta camada estão os pacotes responsáveis pela integração entre o frontend e o backend, implementados em Spring Boot e compostos principalmente pelos Controllers. Essa camada expõe APIs REST que recebem as requisições HTTP enviadas pela camada de Apresentação desenvolvida em Next.js, validam os dados recebidos e os encaminham para a camada de Negócio, retornando as respostas em formato JSON para consumo pelo frontend. Além disso, a camada de API/Integração também pode concentrar a configuração de rotas, o controle de segurança com autenticação e autorização, bem como o tratamento de exceções globais relacionadas às requisições. Por desempenhar o papel de porta de entrada do sistema, ela garante a comunicação clara e padronizada entre a interface do usuário e as regras de negócio que sustentam o funcionamento da aplicação.

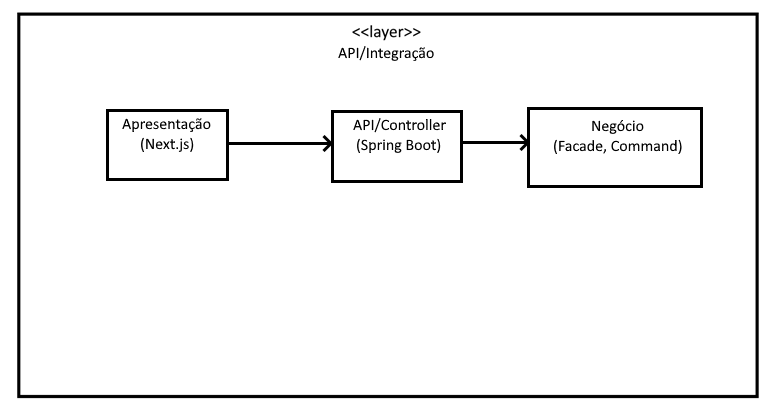


Figura 5.3: Camada de Apresentação

## 

* 1. Camada de Negócio

Nesta camada estão localizadas as classes responsáveis por centralizar e executar as funcionalidades principais do sistema, representando as regras de negócio que dão significado à aplicação. Ela contém **fachadas** que expõem serviços de forma organizada e **comandos** que implementam a lógica necessária para atender às operações solicitadas pelos usuários através da camada de API/Integração.

A camada de Negócio atua como o **núcleo do sistema**, garantindo a separação entre a lógica de processamento e as demais responsabilidades, de forma que mudanças na interface gráfica ou nos mecanismos de integração não impactem diretamente o funcionamento interno das regras.

Além disso, essa camada faz uso de **objetos de transporte (DTOs)** e **entidades de domínio**, que permitem trafegar informações de maneira estruturada, assegurando **clareza, consistência e integridade** durante a execução dos processos.

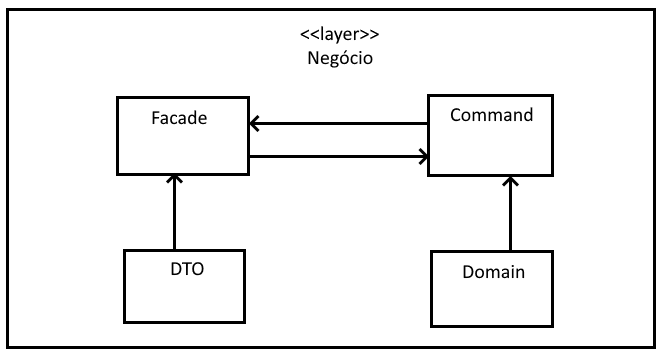


Figura 5.4: Camada de Negócios

* + 1. Pacote Controller

A figura 5.5 ilustra as principais classes de controle.

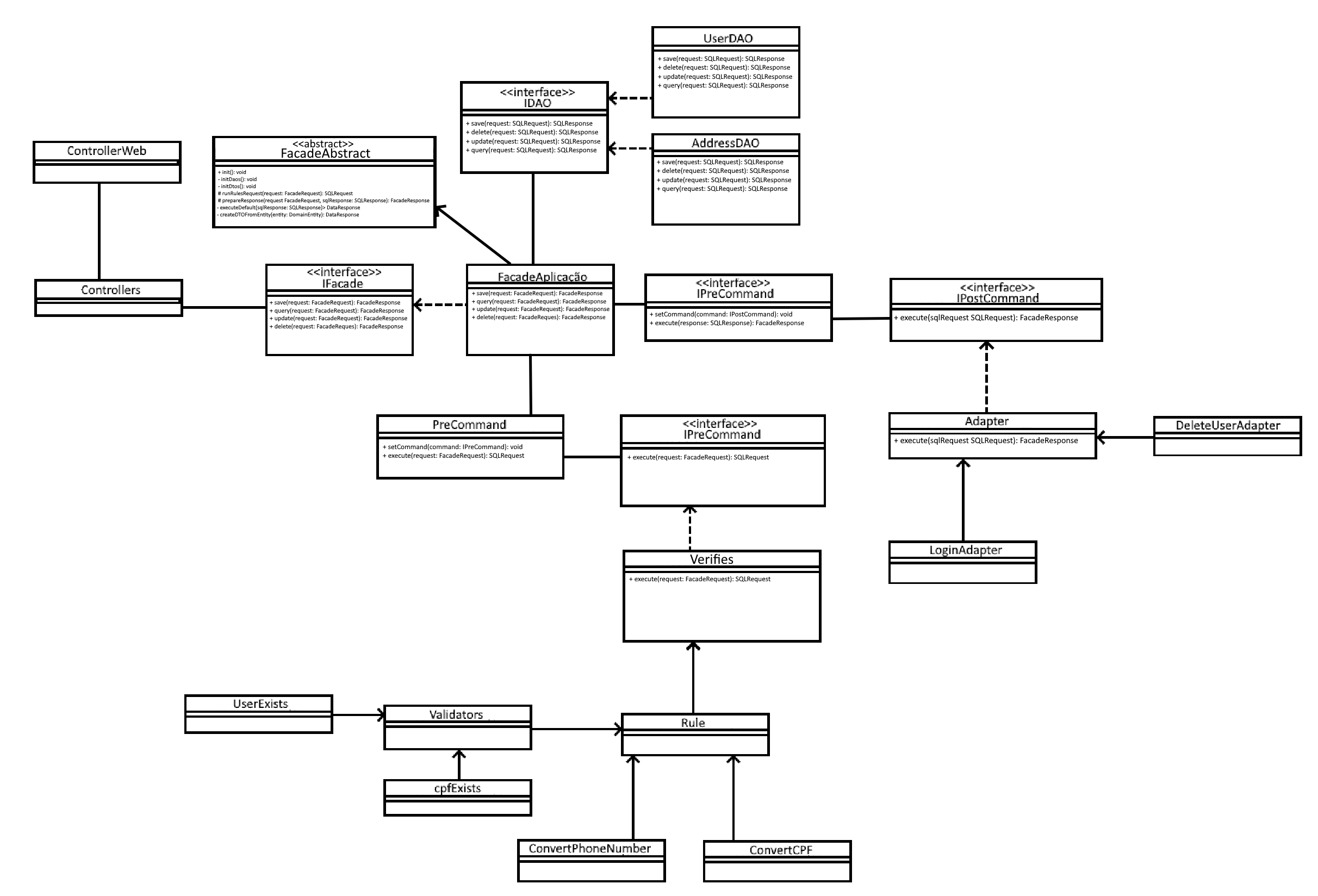


Figura 5.5: Classes de controle

* + 1. Pacote Model

A figura 5.6. ilustra as principais classes do modelo.

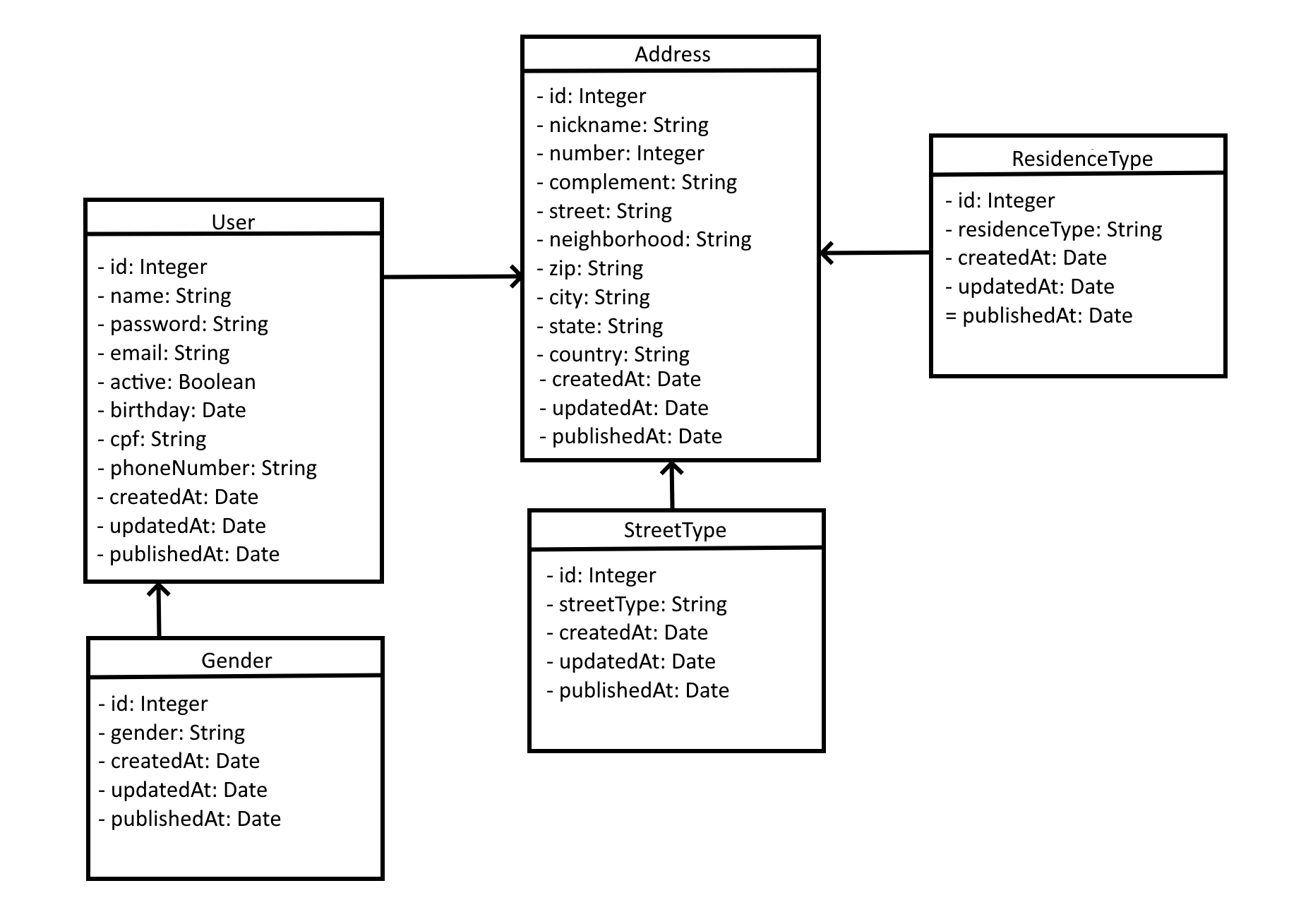


Figura 5.6: Classes do Modelo

## 

* 1. Camada de Persistência

Nesta camada estão as classes responsáveis por **armazenar, consultar e recuperar os dados da aplicação**. Ela é composta por **entidades de modelo (Domain)**, que representam as tabelas e estruturas do banco relacional, por **objetos de transporte (DTOs)** usados para trafegar informações entre a camada de Negócio e outras camadas, e por **repositórios/DAOs**, que implementam a comunicação direta com o banco de dados.

A camada de Persistência tem como principal objetivo **isolar o acesso aos dados** do restante do sistema, garantindo que a lógica de negócio não dependa de detalhes de infraestrutura. Dessa forma, mudanças na base de dados (como a troca de tecnologia ou ajustes em queries) não afetam diretamente as regras de negócio.

Além disso, essa camada trabalha de forma integrada com o mecanismo de mapeamento objeto-relacional (ORM), permitindo operações consistentes de **CRUD (Create, Read, Update, Delete)** sobre os objetos do domínio.

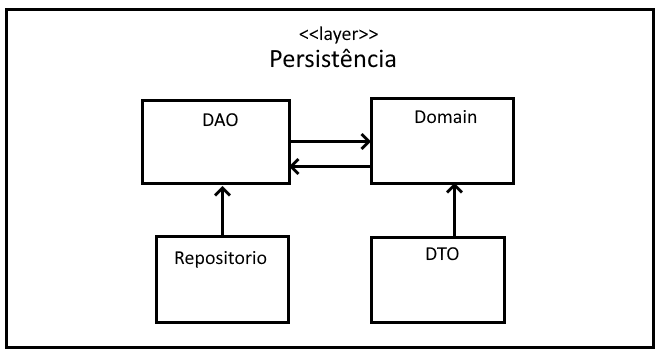


Figura 5.7: Camada de Persistência.

* 1. Camada de Infraestrutura

Nesta camada está localizada a **base técnica que sustenta a aplicação**. Ela contém a classe principal do sistema, responsável por **inicializar o Spring Boot**, configurar pacotes, gerenciar dependências e controlar o ciclo de vida da aplicação.

Além disso, a camada de Infraestrutura pode conter configurações de **segurança, logging, gerenciamento de conexões, filas de mensagens, cache** e outros recursos que dão suporte às demais camadas.

O papel dessa camada **é fornecer serviços e recursos técnicos** de forma transparente para o restante do sistema, sem misturar lógica de negócio ou regras de apresentação. Assim, alterações em frameworks, bibliotecas ou integrações de baixo nível pode ser feitas sem impactar diretamente as outras camadas.

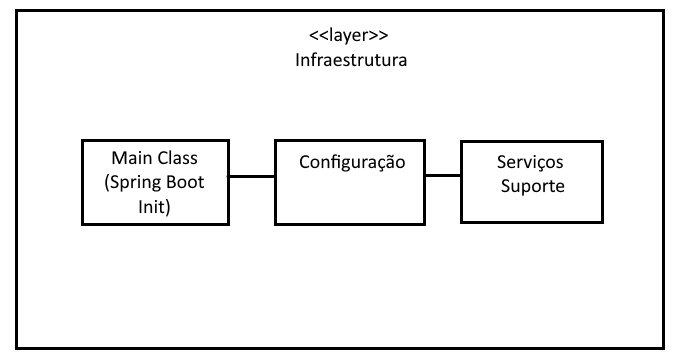


Figura 5.7: Camada de Infraestrutura.

* 1. Camada de Exceções

Nesta camada estão localizadas as **classes de exceção personalizadas**, utilizadas de forma transversal em todo o sistema. O objetivo dessa camada é **centralizar o tratamento de erros e condições inesperadas**, garantindo que as falhas sejam comunicadas de maneira clara e estruturada entre as diferentes camadas da aplicação.

As exceções personalizadas permitem **padronizar mensagens de erro**, encapsular informações relevantes e diferenciar tipos distintos de problemas (como falhas de validação, erros de negócio, problemas de integração ou falhas de persistência).

Essa camada contribui para a **robustez e a manutenibilidade do sistema**, já que separa o tratamento de falhas da lógica principal das demais camadas, além de permitir que os erros sejam propagados e tratados de forma adequada pela camada de API/Integração e apresentados corretamente na camada de Apresentação.

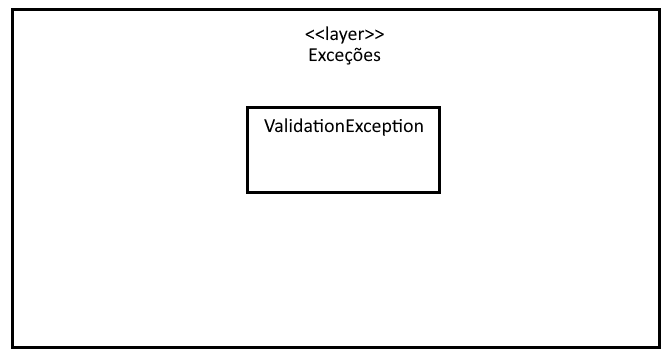


Figura 5.7: Camada de Exceções.

1. Qualidade

O sistema XXX será usado para o controle do comércio exterior brasileiro, conseqüentemente tratando de altos volumes financeiros e um grande número de operações de importação/ exportação diariamente.

Eventuais erros e/ou falhas na sua operação podem levar a prejuízos significativos tanto em termos financeiros quanto na imagem da UMC, portanto na fase de design deve-se levar em consideração como fatores prioritários a confiabilidade e robustez do sistema.

Adicionalmente, o sistema fiel pode ser alvo de ataques de “hackers” para roubar ou simplesmente corromper informações, possibilidade aumentada pela interface do sistema disponível na Internet, para evitar que tais ataques sejam bem sucedidos uma infra-estrutura de segurança deve ser especificada e projetada.

Maiores informações sobre questões relacionadas aos requisitos de qualidade do sistema fiel podem ser obtidas no documento de requisitos não funcionais.

1. Referências

Unified Modeling Language: <http://www.omg.org/technology/documents/formal/uml.htm>

RUP. Rational Unified Process.