

# DOCUMENTO DE VISÃO DE PROJETO



## E-commerce Sarajevo



Tecnologia em Análise e Desenvolvimento de Sistemas | Tecnologia em Agronegócio | Tecnologia em Recursos Humanos

## Histórico de Versões

Data	Versão	Descrição	Autor	Revisor
13/04/25	1.0	Modelagem e desenvolvimento	Fábio	-

**Cliente** FATEC - Interno  
**Documento** Documento de Visão de Projeto: *E-commerce Sarajevo*  
**Data** 13 de abril de 2025  
**Autor** **Fábio Casagrande**  
fabiocasagrande999@gmail.com

## Página de Assinaturas

Revisado e  
Aprovado por:

13/04/25

## Índice

1. Objetivo.....	4
1.1. Escopo .....	4
1.2. Referências .....	4
2. Necessidades de Negócio.....	4
3. Objetivo do Projeto.....	4
4. Declaração Preliminar de Escopo .....	5
4.1. Descrição .....	5
4.2. Produtos a Serem Entregues.....	5
4.3. Requisitos.....	5
4.3.1. Requisitos Funcionais .....	5
4.3.2. Requisitos Não Funcionais.....	5
4.3.3. Regras de Negócio .....	6
5. Premissas .....	6
6. Influência das Partes Interessadas .....	6
7. Representação Arquitetural.....	7
7.1. Restrições Arquiteturais.....	9
7.2. Objetivos e Restrições Arquiteturais .....	9
8. Visão de Use Case .....	9
8.1. Diagrama de Casos de Uso Arquiteturalmente Significativos .....	10
8.1.1. Diagrama de Casos de Uso Manter Catálogo de Livros .....	10
8.1.2. Diagrama de Casos de Uso Realizar Pedido .....	11
8.1.3. Diagrama de Casos de Uso Histórico de Pedidos .....	12
8.2. Descrição de Casos de Uso Arquiteturalmente Significativos .....	13
8.2.1. Manter Catálogo de Livros .....	13
8.2.2. Realizar Pedido.....	13
8.2.3. Histórico de Pedidos .....	13
9. Visão de Lógica .....	14
9.1. Camada de Apresentação .....	16
9.1.1. Camada de Negócio .....	17
9.1.2. Pacote Controller .....	18
9.1.3. Pacote Model .....	19
9.1.4. Camada de Persistência .....	20
9.1.5 Realização de Casos de Uso Significativos.....	21
9.1.5.1 Manter Catálogo de Livros.....	21
9.1.5.2 Realizar Pedidos .....	22
9.1.5.3. Histórico de Pedidos.....	23
10. Visão de Implantação.....	24
11. Visão de Implementação.....	25
12. Visão de Dados.....	26
13. Tamanho e Performance.....	28
14. Qualidade .....	29
15. Cronograma Macro .....	30
16. Referências.....	31

## 1. Objetivo

Este documento tem como principal finalidade registrar as necessidades de negócio, justificar o desenvolvimento do sistema proposto e apresentar o entendimento atual sobre os requisitos do cliente. Visa descrever de forma resumida o novo sistema de e-commerce de livreria, que deverá satisfazer os requisitos levantados. Além disso, busca alinhar as expectativas dos stakeholders e formalizar o início do projeto, apresentando uma visão geral da arquitetura do sistema.

### 1.1. Escopo

O escopo deste documento abrange o desenvolvimento de uma aplicação web de e-commerce voltada para a venda de livros. O sistema atenderá tanto clientes quanto administradores, oferecendo funcionalidades como cadastro e busca de livros, gerenciamento de pedidos, controle de estoque, e funcionalidades administrativas. Este módulo será construído utilizando o framework Spring Boot, fazendo parte de uma solução integrada de comércio eletrônico.

Este documento visa ainda descrever os principais aspectos arquiteturais da aplicação, incluindo sua estrutura em camadas, divisão em pacotes e componentes, e as principais decisões técnicas envolvidas no projeto.

### 1.2. Referências

Para a elaboração deste documento, foram consideradas as seguintes referências:

- Levantamento de requisitos funcionais, não funcionais e regras de negócio definidos previamente.
- Material didático e orientações fornecidas pelo professor da disciplina.

Este documento serve de base para o seguinte artefato:

- Documento de Requisitos

## 2. Necessidades de Negócio

Um sistema informatizado de e-commerce voltado para a comercialização de livros é necessário para que livrerias possam expandir sua presença digital, automatizar processos comerciais e oferecer uma experiência de compra eficiente aos seus clientes. Através da solução, será possível gerenciar o catálogo de livros disponíveis, realizar vendas online, acompanhar o status de pedidos e controlar o estoque de forma integrada.

Além disso, o sistema permitirá que administradores tenham acesso a relatórios gerenciais com base nas transações realizadas, auxiliando na tomada de decisões estratégicas e operacionais. Com a informatização dessas rotinas, espera-se melhorar o atendimento ao cliente, reduzir erros manuais e ampliar o alcance comercial da livreria.

## 3. Objetivo do Projeto

Desenvolver uma plataforma web de e-commerce especializada na venda de livros, que possibilite:

- Gerenciar o carrinho de compras e realizar pedidos online;
- Controlar o estoque de forma automatizada;

- Permitir o cadastro e gerenciamento de clientes;
- Emitir relatórios de vendas, pedidos e movimentações do sistema;
- Utilizar arquitetura baseada em microserviços com Spring Boot e persistência de dados em banco relacional.

A solução deverá ser acessível por meio de navegadores modernos, com interface amigável e responsiva, visando proporcionar agilidade nas operações da livraria e uma boa experiência ao usuário final.

## 4. 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.

### 4.1. Descrição

O sistema proposto é uma plataforma web de e-commerce voltada à comercialização de livros. A solução visa atender pequenas e médias livrarias que desejam digitalizar suas operações, oferecendo funcionalidades essenciais para a gestão de produtos, pedidos, usuários e controle de estoque. O sistema será desenvolvido em Java com uso do framework Spring Boot, com foco em desempenho, segurança e escalabilidade.

### 4.2. Produtos a serem entregues

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

- Plataforma de e-commerce de livros, etapa 1, implementada de acordo com a especificação elaborada na fase de análise (código objeto e código fonte);
- Documentação de especificação técnica e funcional, elaborada na fase de concepção;

### 4.3. Requisitos

#### 4.3.1. Requisitos Funcionais

- O sistema deve permitir o cadastro, edição, exclusão e visualização de livros no catálogo;
- O sistema deve permitir a realização de pedidos por parte dos clientes;
- O sistema deve gerenciar o status dos pedidos: pendente, em preparação, enviado, entregue;
- O sistema deve permitir o cadastro e autenticação de usuários (clientes e administradores);
- O sistema deve permitir o controle de estoque dos livros;
- O sistema deve gerar relatórios administrativos com dados de vendas, estoque e pedidos;
- O sistema deve possibilitar a busca e filtragem de livros por critérios como título, autor, gênero, preço, etc.

#### 4.3.2. Requisitos Não Funcionais

- O sistema deverá ser desenvolvido utilizando a linguagem Java;
- O framework Spring Boot será utilizado na construção do backend da aplicação;
- O banco de dados utilizado será o MySQL;
- A arquitetura do sistema deverá seguir o padrão MVC (Model-View-Controller);
- A interface deverá ser responsiva, adaptando-se a diferentes tamanhos de tela.
- A aplicação deverá ser compatível com os principais navegadores modernos:
  - Google Chrome
  - Mozilla Firefox

- Microsoft Edge

#### 4.3.3. Regras de Negócio

- Um livro só poderá ser comprado se houver estoque disponível;
- Um pedido só poderá ser concluído mediante confirmação de pagamento;
- Os administradores poderão cadastrar, editar e remover livros do catálogo;
- O sistema deve registrar data e hora dos pedidos realizados;
- Clientes só poderão acessar o histórico de pedidos mediante login autenticado;
- Apenas usuários com perfil de administrador poderão acessar os relatórios de vendas e estoque.

## 5. Premissas

As premissas descritas abaixo foram consideradas verdadeiras durante a fase de iniciação do projeto e servirão de base para o planejamento detalhado a ser realizado posteriormente. Premissas não documentadas podem comprometer o sucesso do projeto, portanto, este registro inicial é fundamental.

- O projeto será orientado pelo professor Rodrigo Rocha.
- A equipe de desenvolvimento estará disponível para reuniões semanais de alinhamento durante o semestre letivo.
- A aplicação será implementada utilizando a linguagem Java com o framework Spring Boot.
- O banco de dados MySQL será utilizado e disponibilizado para o projeto sem custo adicional.
- O escopo atual do projeto contempla apenas a etapa inicial da plataforma, podendo ser expandido em fases futuras.
- Os interessados no projeto (stakeholders) estarão disponíveis para fornecer feedback e validações parciais durante o processo de desenvolvimento.

## 6. Influência das Partes Interessadas

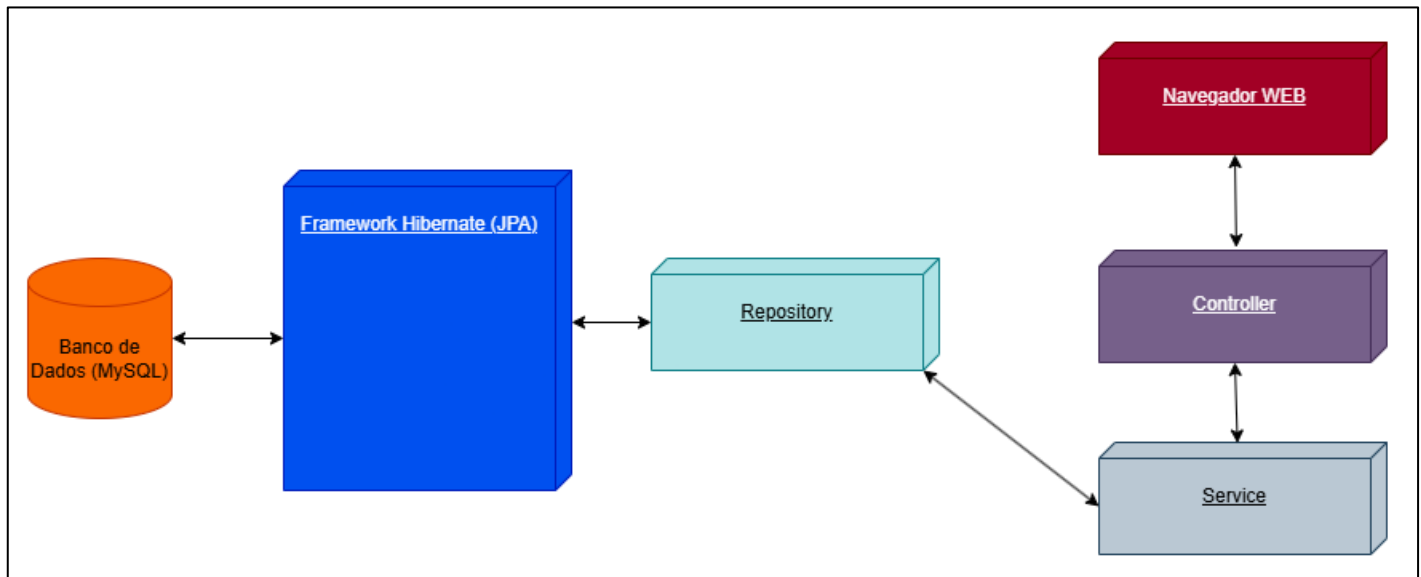
Durante a fase de iniciação do projeto, foram identificadas as seguintes partes interessadas, bem como suas possíveis influências sobre os objetivos, escopo e execução do projeto:

- **Professor Rodrigo Rocha:** orientador do projeto. Possui influência direta sobre a condução das atividades, qualidade da documentação e validação dos marcos do projeto. Seu apoio e direcionamento são fundamentais para o andamento e aprovação do projeto.
- **Fábio Casagrande:** responsável pela análise, projeto, implementação e testes do sistema. Tem interesse em aplicar boas práticas de engenharia de software e garantir a entrega do produto conforme os requisitos.
- **Usuários finais (clientes da livraria):** esperam uma interface intuitiva, funcionalidade completa no processo de compra e segurança nas transações. Sua aceitação será um dos fatores críticos de sucesso.
- **Administradores da plataforma (donos da livraria):** interessados na eficiência do sistema para gerenciar produtos, pedidos e relatórios. Sua colaboração é essencial para validações e melhorias nos processos.

A identificação dessas partes interessadas permite antecipar necessidades, alinhar expectativas e definir estratégias para mitigar riscos e fortalecer os fatores críticos de sucesso.

## 7. Representação Arquitetural

Os sistemas serão desenvolvidos tendo como base a arquitetura ilustrada na Figura 1. Toda a arquitetura segue padrões tradicionais de projeto definidos pelo GoF, assim como conceitos da arquitetura J2EE, que continuam válidos como referência, mesmo com o uso de tecnologias modernas como SpringBoot.



**Figura 1 - Modelo Arquitetural**

A camada de apresentação será abrigada no Container WEB, e será composta por controladores e páginas que implementam o padrão Model-View-Controller (MVC). Para esta camada, será utilizada uma solução baseada em Spring MVC e templates HTML com Thymeleaf.

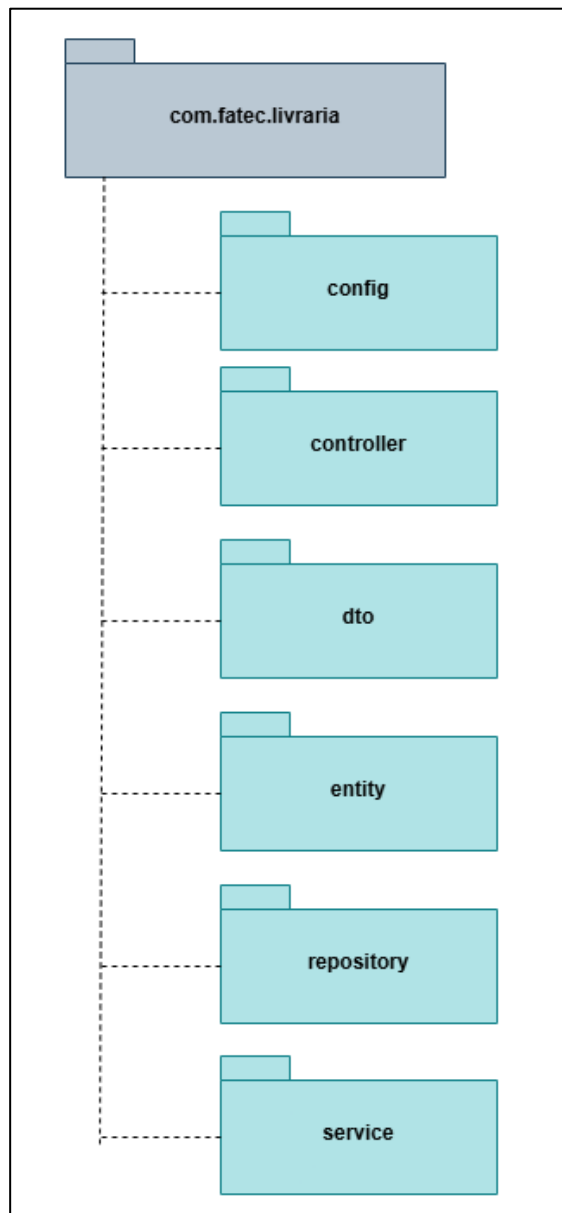
As Classes de Domínio representam as entidades do sistema, contendo atributos e métodos de acesso (getters e setters), que refletem os dados manipulados pela aplicação (como livros, usuários, pedidos, etc).

As Classes de Negócio são responsáveis pelas regras de negócio do sistema e serão implementadas como serviços na arquitetura Spring (anotated com @Service). Estas classes fazem uso de repositórios JPA para realizar operações de persistência no banco de dados.

A camada de persistência será implementada com o Hibernate (JPA), e terá a função de intermediar o acesso ao banco de dados relacional (MySQL), conforme mostrado na figura.

Neste projeto, será adotada uma arquitetura modular baseada no framework Spring Boot, utilizando componentes como @Controller, @Service e @Repository para garantir uma separação clara de responsabilidades entre as camadas da aplicação. A persistência de dados será realizada por meio do Hibernate (JPA), que facilita a comunicação com o banco de dados relacional. Essa estrutura promove organização, manutenibilidade e facilita a escalabilidade do sistema.

O diagrama a seguir, Figura 2, representa a organização das classes dentro dos pacotes da aplicação baseada em Spring Boot. A estrutura foi pensada de forma a garantir a separação de responsabilidades e a facilidade na manutenção do sistema.



**Figura 2 - Diagrama em pacotes**

A aplicação será estruturada com pacotes distintos para as camadas de apresentação (controller), regra de negócio (service), persistência de dados (repository), modelo de dados (model), além de pacotes auxiliares como dto e config.

Essa abordagem elimina a necessidade de diferenciação entre aplicações J2SE e J2EE, bem como o uso de componentes complexos como EJBs. A arquitetura aproveita os recursos nativos do Spring Boot, como a anotação `@Service`, `@Repository`, e `@RestController`, e se alinha aos padrões modernos de desenvolvimento de aplicações Java corporativas.

Essa padronização permite maior reutilização de código, clareza arquitetural e aderência às boas práticas recomendadas pela comunidade Java.



## 7.1. Restrições Arquiteturais

Foram identificadas algumas orientações e restrições pertinentes ao desenvolvimento deste sistema:

- Utilização do JDK 21 (Java 21) como plataforma base de desenvolvimento;
- Utilização do Spring Boot como framework principal da aplicação, oferecendo suporte completo para desenvolvimento modular, injeção de dependências e controle transacional;
- Utilização do framework Hibernate (JPA) para mapeamento objeto-relacional (ORM);
- Utilização do SGBD MySQL como banco de dados relacional;
- Utilização de fetch API (JavaScript) para realizar requisições assíncronas entre o front-end e a aplicação backend;
- A aplicação será executada em servidores compatíveis com Spring Boot embutido, eliminando a necessidade de servidores de aplicação tradicionais como Tomcat externo ou Oracle Application Server;

## 7.2. Objetivos e Restrições Arquiteturais

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

- A necessidade de garantir um modelo de persistência desacoplado, baseado em JPA/Hibernate, facilitando a portabilidade entre diferentes SGBDs;
- A utilização do Spring Boot como plataforma de desenvolvimento principal, devido à sua robustez, flexibilidade e integração com múltiplos módulos (segurança, dados, validação etc.);
- A padronização da linguagem Java em sua versão mais recente e estável (Java 21), com foco em desempenho e segurança;
- A adoção de práticas modernas de desenvolvimento web, com comunicação entre cliente e servidor realizada via requisições assíncronas (fetch/AJAX);
- A preferência por software livre e de código aberto, respeitando as premissas de baixo custo e facilidade de manutenção;
- A utilização do SGBD MySQL como repositório de dados oficial da aplicação.

## 8. 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 da Livraria Virtual.

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;
- Casos de uso que acessam sistemas externos ou persistem dados importantes.

## 8.1. Diagrama de Casos de Uso Arquiteturalmente Significativos

### 8.1.1. Diagrama de Casos de Uso Manter Catálogo de Livros

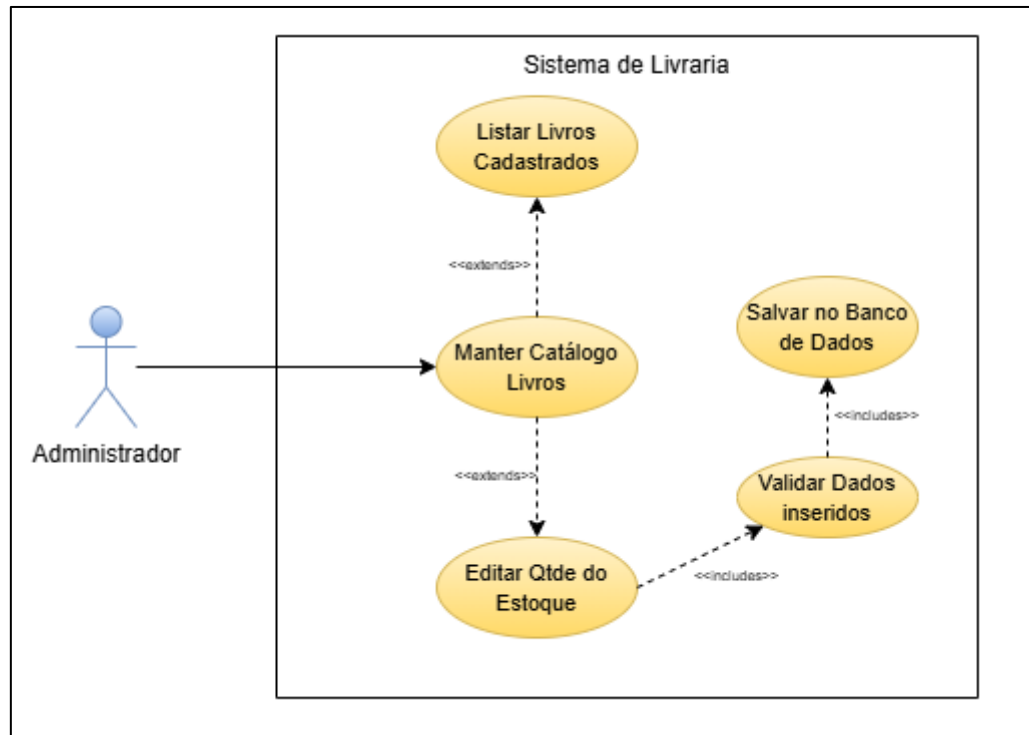


Figura 3.1 - Caso de Uso de Manter Catálogo de Livros

### 8.1.2. Diagrama de Casos de Uso Realizar Pedido

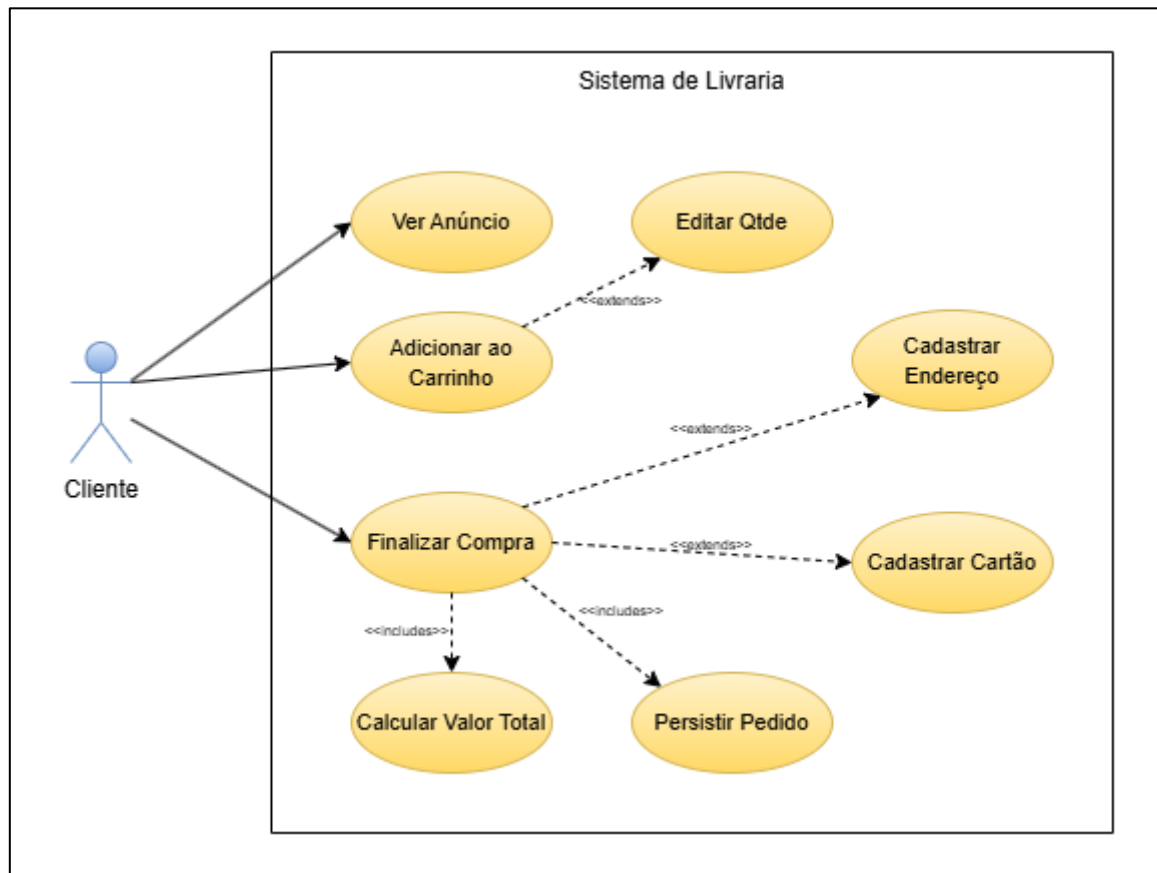


Figura 3.2 - Caso de Uso de Realizar Pedido

### 8.1.3. Diagrama de Casos de Uso Histórico de Pedidos

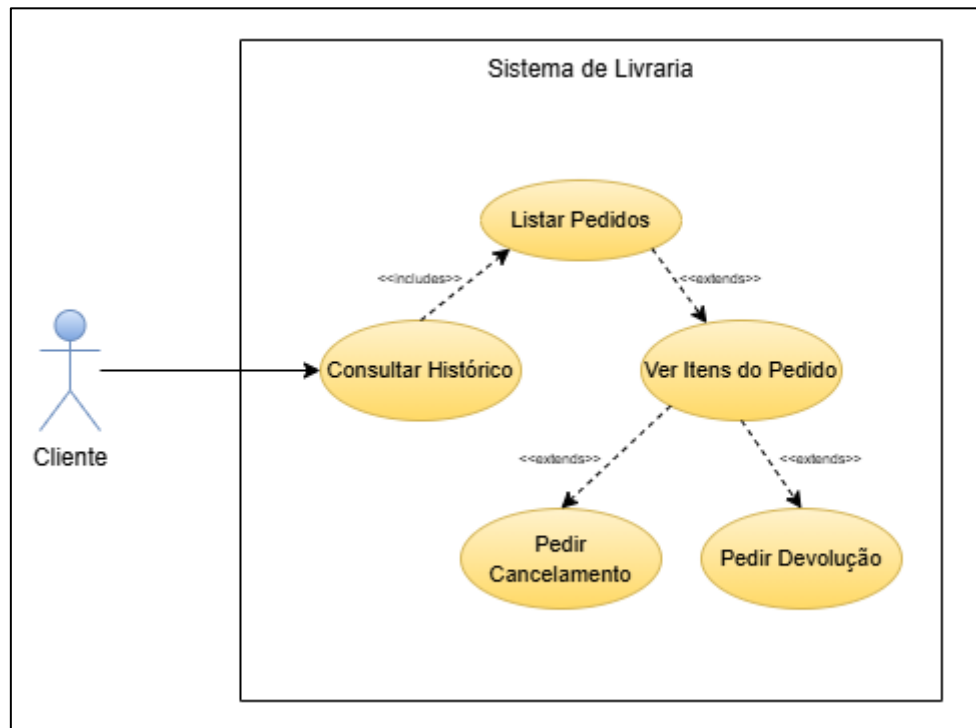


Figura 3.3 - Caso de Uso de Histórico de Pedidos

## 8.2. Descrição dos Casos de Uso Arquiteturalmente Significativos

### 8.2.1. Manter Catálogo de Livros

Este caso de uso permite que o administrador cadastre, edite e exclua livros do acervo da livraria. O sistema deve validar os dados inseridos e persistir as alterações no banco de dados.

- Ator principal: Administrador
- Fluxo principal:
  - Administrador acessa o painel de administração.
  - Escolhe a opção "Gerenciar Estoque".
  - Preenche ou edita o campo Estoque.
  - Submete os dados.
  - Sistema valida e persiste no banco via camada Service e Repository.
- Envolve as camadas: Controller → Service → Repository → Hibernate (JPA) → MySQL

### 8.2.2. Realizar Pedido

O usuário navega pelo catálogo, adiciona livros ao carrinho e finaliza a compra. Este caso envolve controle de estoque, cálculo de preço e persistência do pedido.

- Ator principal: Usuário
- Fluxo principal:
  - Usuário navega e adiciona livros ao carrinho.
  - Acessa o carrinho e clica em "Finalizar Pedido".
  - Preenche os dados de pagamento e envio.
  - Sistema valida os dados, cria o pedido e atualiza o estoque.
- Envolve: Controller → Service (com lógica de negócio e cálculo) → Repository

### 8.2.3. Histórico de Pedidos

Permite que o usuário logado consulte os pedidos realizados, com detalhes como data, status e itens comprados. Pode envolver paginação e filtros.

- Ator principal: Usuário
- Fluxo principal:
  - Usuário acessa "Meus Pedidos".
  - Sistema recupera os pedidos do banco e exibe em ordem cronológica.
- Pode envolver: chamadas assíncronas via fetch (AJAX), paginação e cache.

## 9. Visão de Lógica

Esta visão apresenta elementos de design significativos do ponto de vista da arquitetura, descrevendo a organização do sistema de e-commerce de livros em camadas lógicas, seguindo o padrão MVC (Model-View-Controller) adotado pelo framework Spring Boot.

O sistema está organizado em três camadas principais, conforme ilustrado na Figura 5.1:

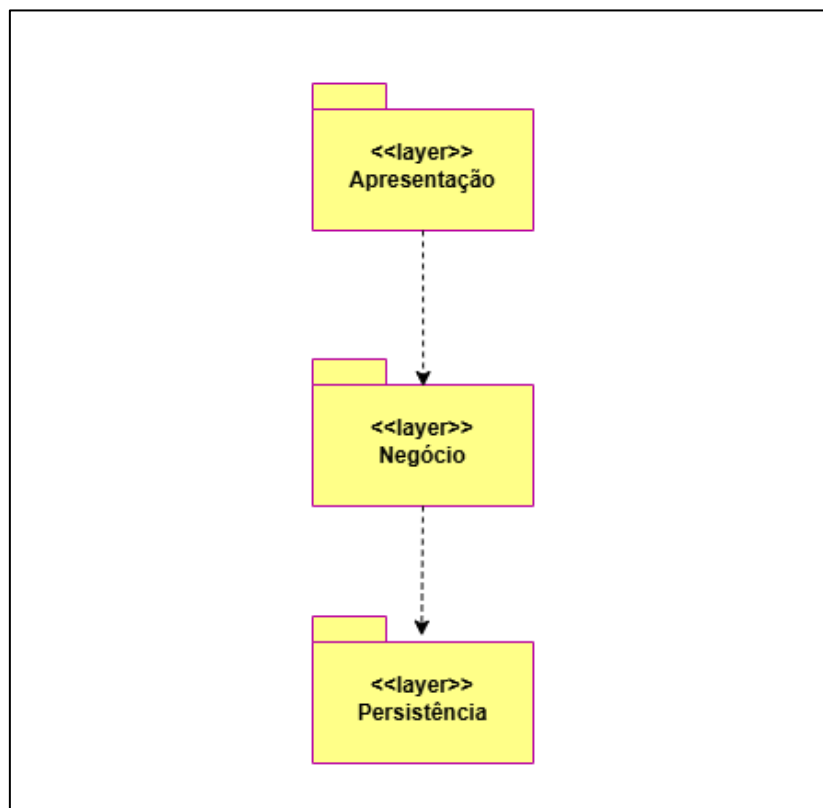


Figura 4.1 – Diagrama de Camadas do E-commerce Sarajevo

**Apresentação:** Esta camada é composta pelos controladores REST do Spring (@RestController) e pela interface do usuário baseada em HTML, CSS e JavaScript. A comunicação com o backend é feita por meio de chamadas assíncronas via fetch (AJAX), permitindo operações como cadastro de livros, finalização de pedidos e consulta de histórico.

**Negócio:** Nesta camada estão concentradas as regras de negócio da aplicação. É representada pelas classes anotadas com @Service, que orquestram a lógica da aplicação, fazem validações, cálculos (como preços e descontos), e coordenam o acesso às entidades via repositórios.

**Persistência:** Responsável pela comunicação com o banco de dados relacional (MySQL), esta camada é composta por interfaces @Repository e pelas entidades @Entity que representam o modelo de domínio. A persistência dos dados é feita por meio do framework JPA (Hibernate).

A Figura 4.2 ilustra o diagrama de camadas do sistema, complementando a visão apresentada na Figura 4.1, agora com as principais tecnologias utilizadas no desenvolvimento do projeto de e-commerce de livros.

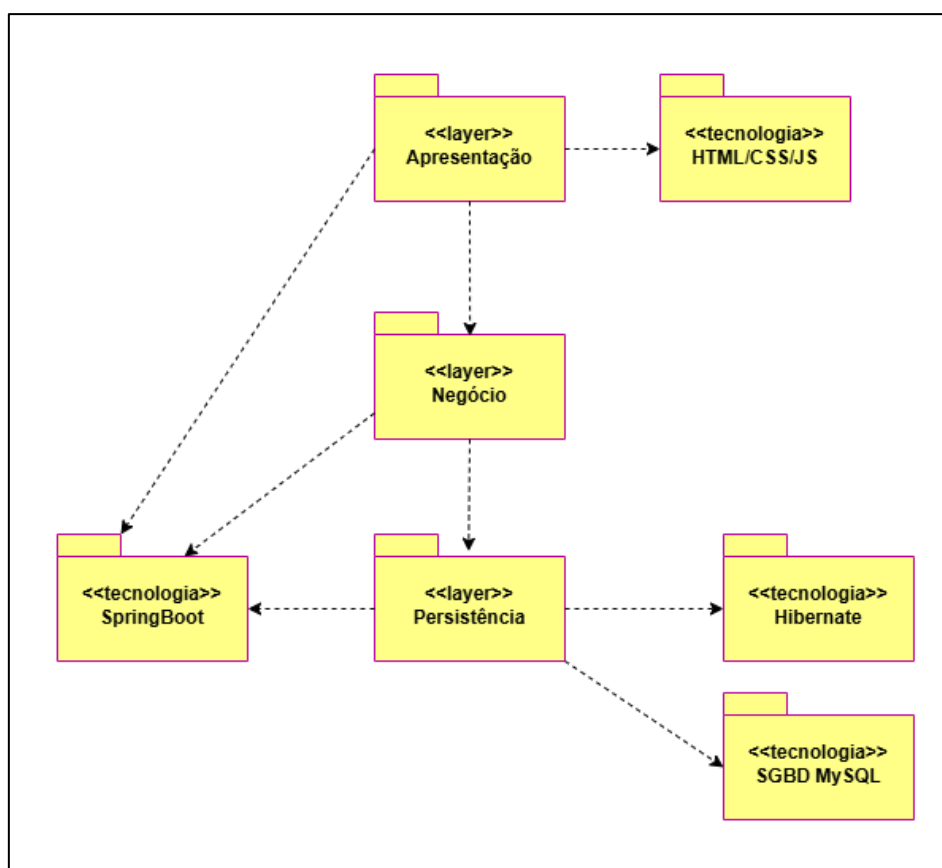
A arquitetura segue uma estrutura em camadas bem definida, composta por: Apresentação, Negócio e Persistência. Cada camada possui responsabilidades distintas e interage de forma desacoplada por meio de interfaces bem definidas.

Na camada de apresentação, os componentes se dividem em duas partes:

- Porção cliente: composta por HTML, CSS e JavaScript executados no navegador do usuário. Essa parte é responsável por renderizar as interfaces, realizar validações básicas (como campos obrigatórios) e interagir com o servidor por meio de requisições assíncronas utilizando a API fetch, viabilizando uma experiência dinâmica com chamadas AJAX.
- Porção servidora: implementada com @RestController do Spring Boot, essa parte expõe endpoints REST que recebem e processam requisições do front-end. Os dados trafegam no formato JSON, promovendo uma integração leve e eficiente entre cliente e servidor.

A camada de negócio centraliza as regras da aplicação e é composta por serviços (@Service), que orquestram os dados e a lógica necessária para atender às funcionalidades do sistema. Essa camada atua como intermediária entre o controlador REST e a persistência.

Por fim, a camada de persistência é responsável pelo acesso ao banco de dados relacional MySQL, utilizando o framework Spring Data JPA com suporte ao Hibernate para realizar o mapeamento objeto-relacional (ORM). A comunicação entre as camadas segue o fluxo: Controller → Service → Repository → Banco de Dados.



**Figura 4.2 – Camadas do E-commerce Sarajevo com dependência de tecnologias**

## 9.1. Camada de Apresentação

A camada de apresentação é responsável por gerenciar a interação entre o usuário e o sistema. Ela é composta por dois principais componentes:

- **Front-end (Cliente):** Contém os arquivos responsáveis pela interface visual acessada no navegador do usuário, como HTML, CSS, imagens e scripts JavaScript. As requisições assíncronas feitas via fetch permitem interações dinâmicas sem recarregar a página, como inserções e consultas ao catálogo de livros.
- **Back-end (Servidor):** Implementado com o framework Spring Boot, a lógica de controle da apresentação é representada por classes anotadas com `@RestController`. Esses controladores expõem endpoints REST que recebem as requisições do front-end, processam os dados e retornam as respostas no formato JSON.

Na Figura 5.3, é ilustrada a organização básica da camada de apresentação. O pacote web concentra tanto os controladores (controller) responsáveis pela lógica da interface, quanto os arquivos de interface em si (form), que incluem os recursos visuais e scripts de interação.

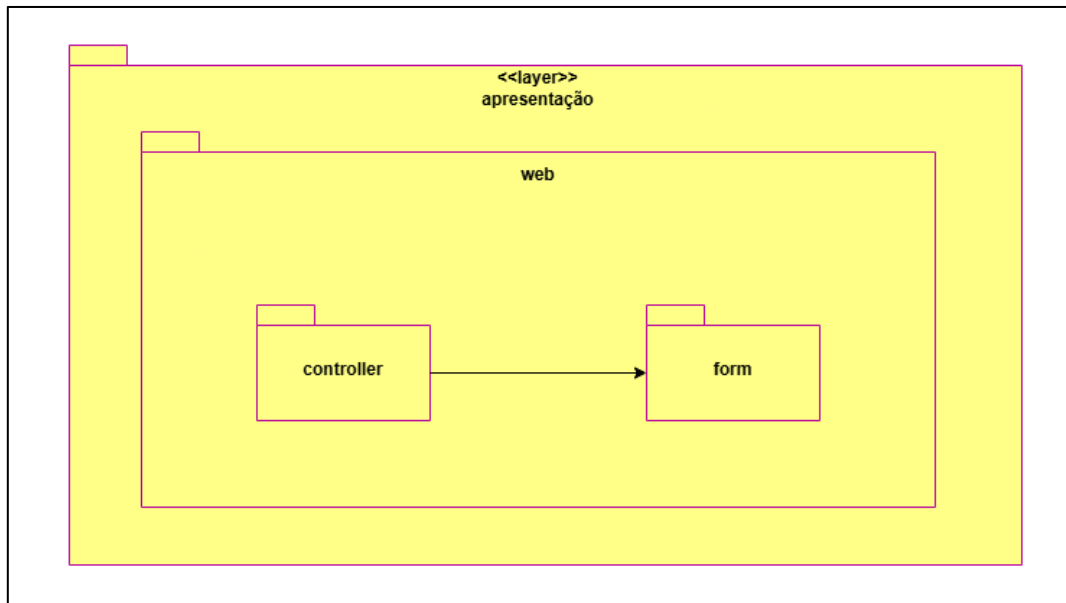


Figura 4.3 – Camada de Apresentação



### 9.1.1. Camada de Negócio

A camada de negócio é responsável por encapsular toda a lógica central do sistema, garantindo a integridade das regras e operações que regem os processos da livreria. Ela é composta pelos seguintes pacotes:

- **service:** Contém as classes responsáveis por coordenar as operações de negócio, como processar pedidos, aplicar regras de desconto ou validar informações antes da persistência. Essas classes geralmente são anotadas com `@Service`.
- **model (ou domain):** Reúne as classes que representam as entidades do sistema, como Livro, Usuario, Pedido, entre outras. Essas classes são anotadas com `@Entity` e mapeadas com a JPA (Hibernate).
- **filter (ou specification):** Contém classes utilitárias para construção de filtros dinâmicos usados em consultas, por exemplo, para busca de livros por critérios como título, autor ou preço.

A Figura 4.4 representa a estrutura da camada de negócio e os relacionamentos entre seus componentes.

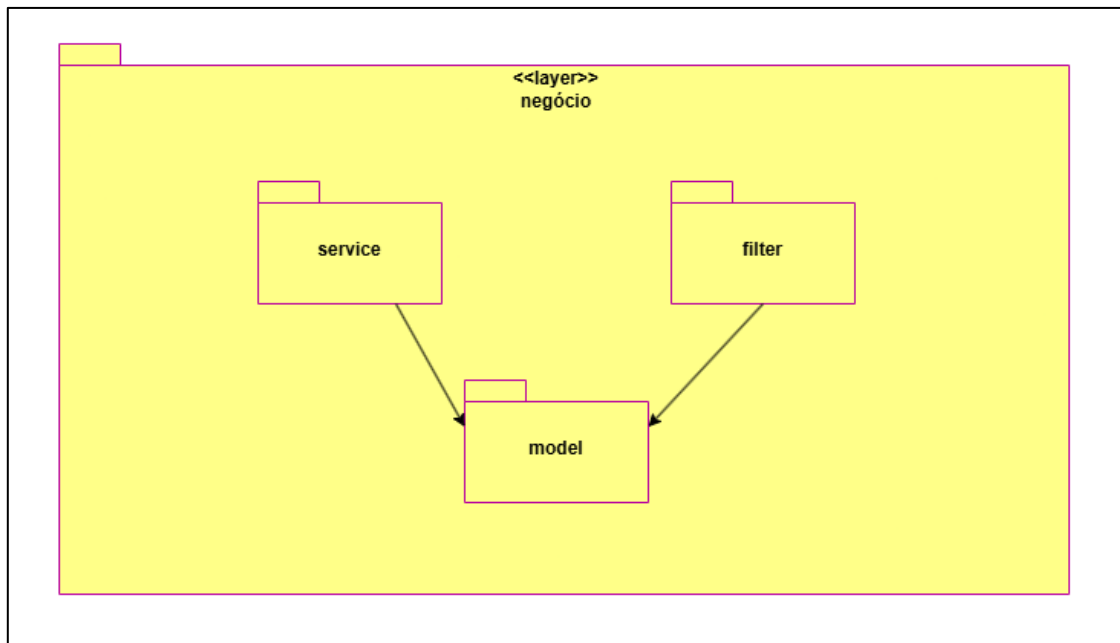
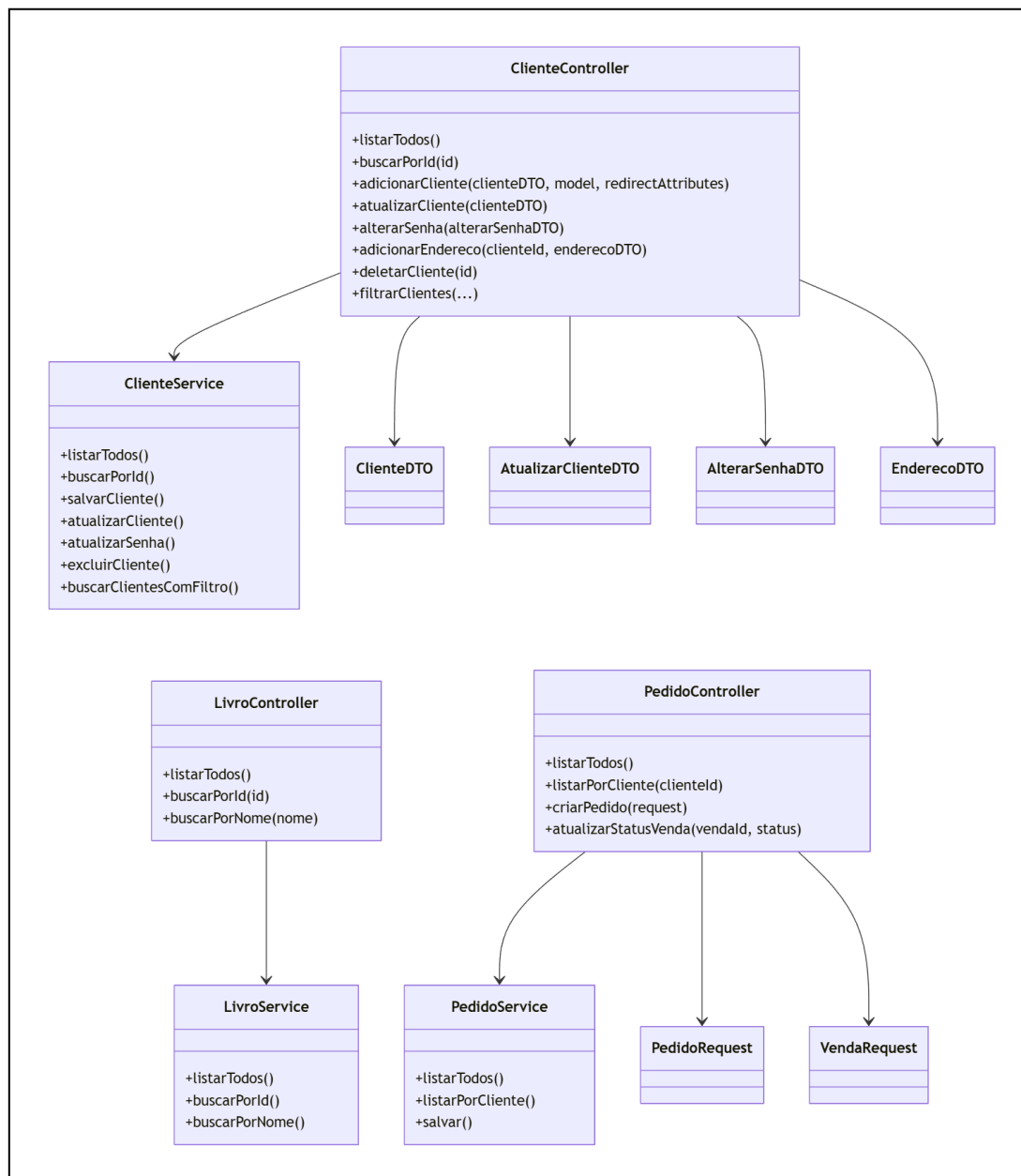


Figura 4.4 – Camada de Negócios

### 9.1.2. Pacote Controller

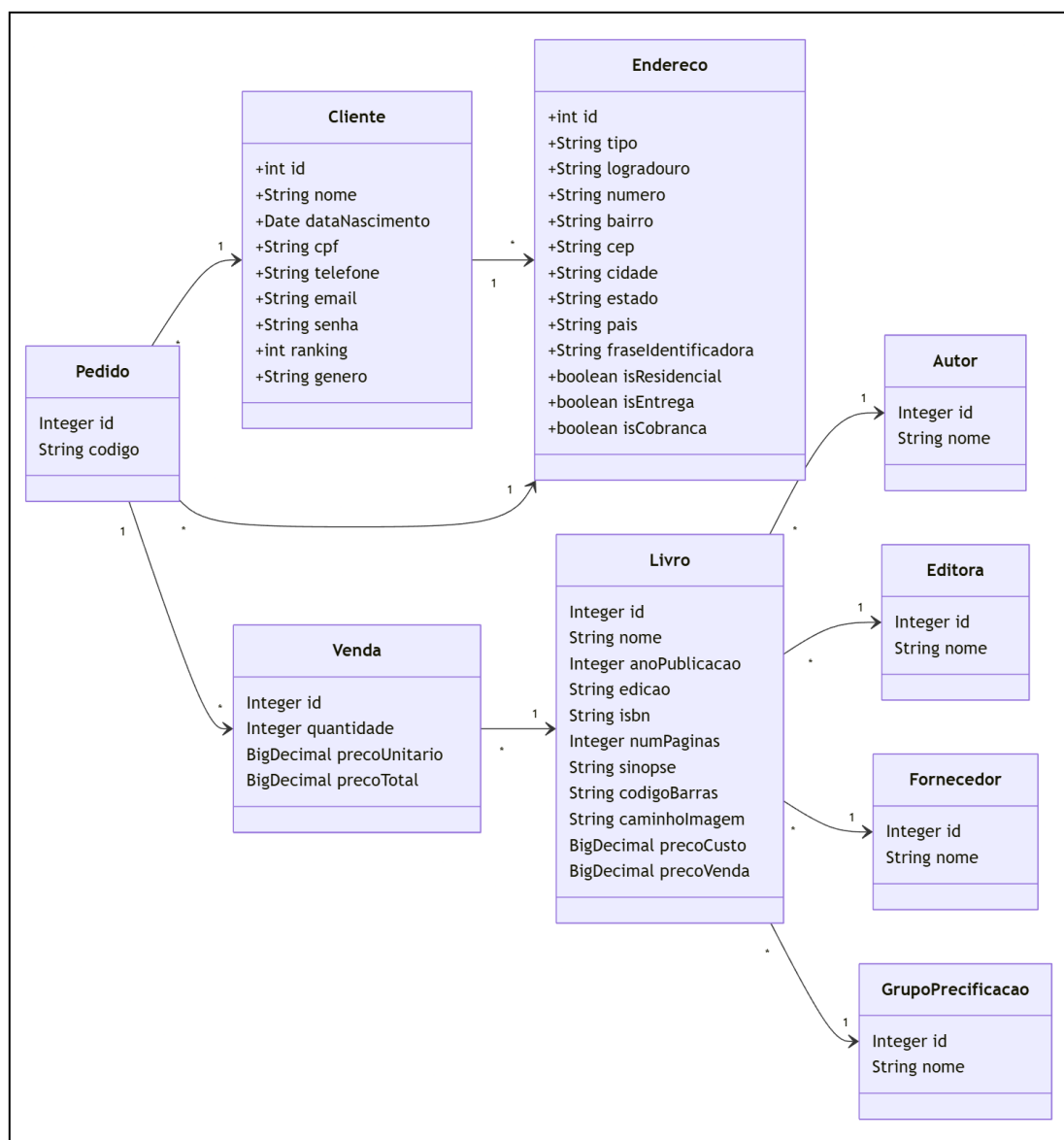
A figura 4.5 apresenta as classes de controle mais relevantes do sistema.



**Figura 4.5 – Classes de Controle**

### 9.1.3. Pacote Model

A figura 4.6. apresenta as classes do modelo mais relevantes do sistema.



**Figura 4.6 – Classes do Modelo**

#### 9.1.4. Camada de Persistência

A camada de persistência é responsável por todas as operações de acesso ao banco de dados relacional. Ela contém as interfaces que realizam a comunicação com o banco por meio do Spring Data JPA, que abstrai a maior parte do código de acesso a dados.

Essa camada é composta principalmente por:

- repository: Contém interfaces estendendo JpaRepository ou CrudRepository, responsáveis pelas operações de persistência, consulta e exclusão de entidades como Livro, Usuario e Pedido.
- hibernate (implícito): A persistência é feita com o uso do Hibernate como provedor JPA, embora as classes que interagem diretamente com ele fiquem ocultas pela abstração do Spring Data. O mapeamento entre objetos Java e tabelas do banco é configurado via anotações JPA nas entidades da camada de modelo (@Entity, @Id, @ManyToOne, etc.).

A Figura 4.7 representa a estrutura da camada de persistência e sua organização.

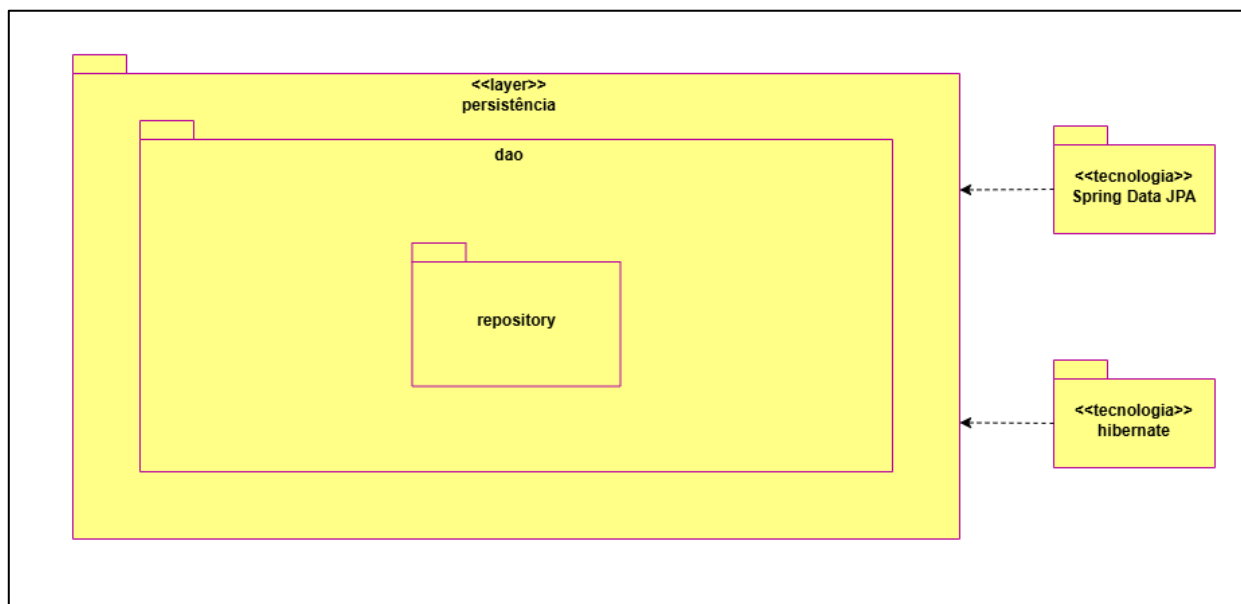


Figura 4.7 – Camada de Persistência

## 9.1.5. Realização de Casos de Uso Significativos

### 9.1.5.1. Manter Catálogo de Livros

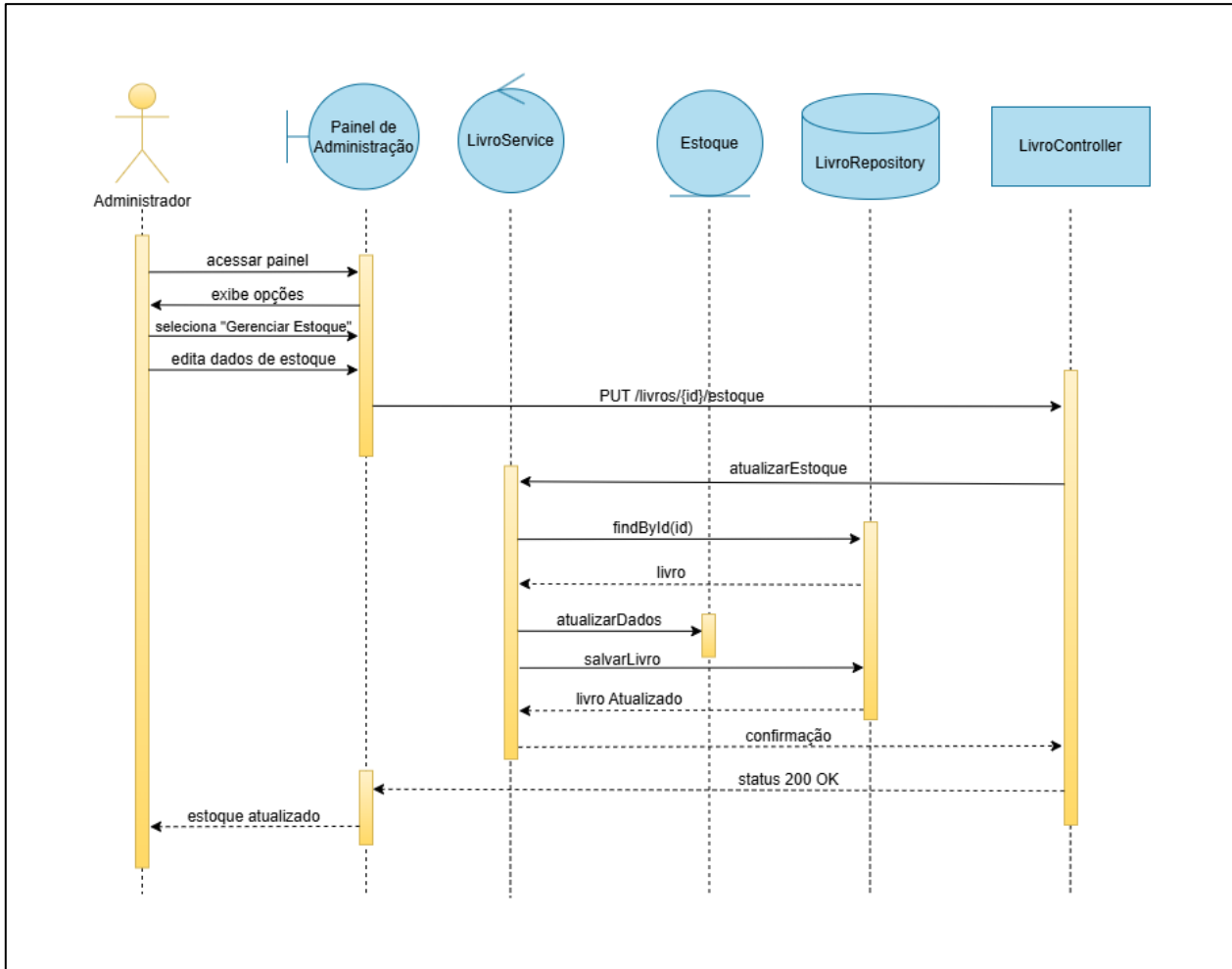


Figura 5.1 – Diagrama de Sequência Manter Catálogo de Livros

### 9.1.5.2. Realizar Pedido

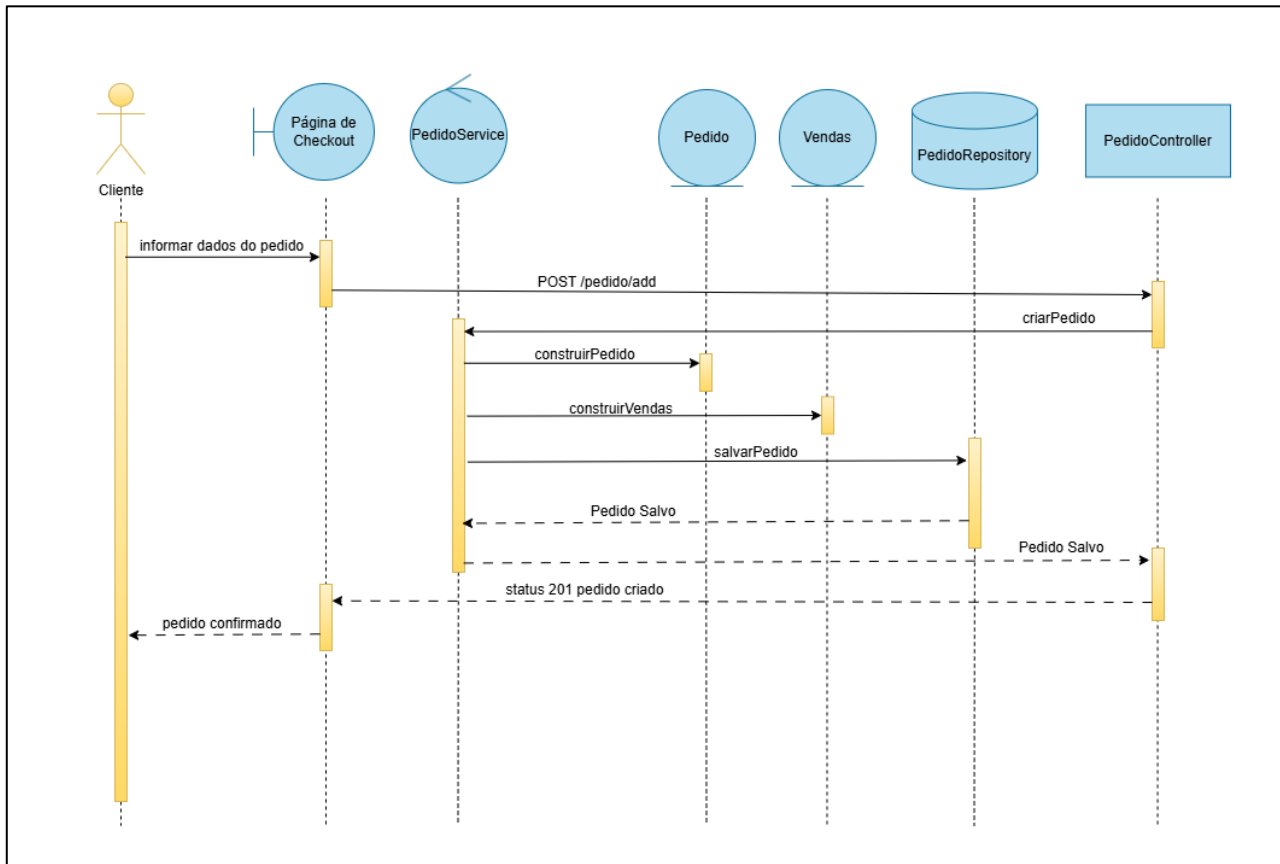


Figura 5.2 – Diagrama de Sequência Realizar Pedido

### 9.1.5.3. Histórico Pedidos

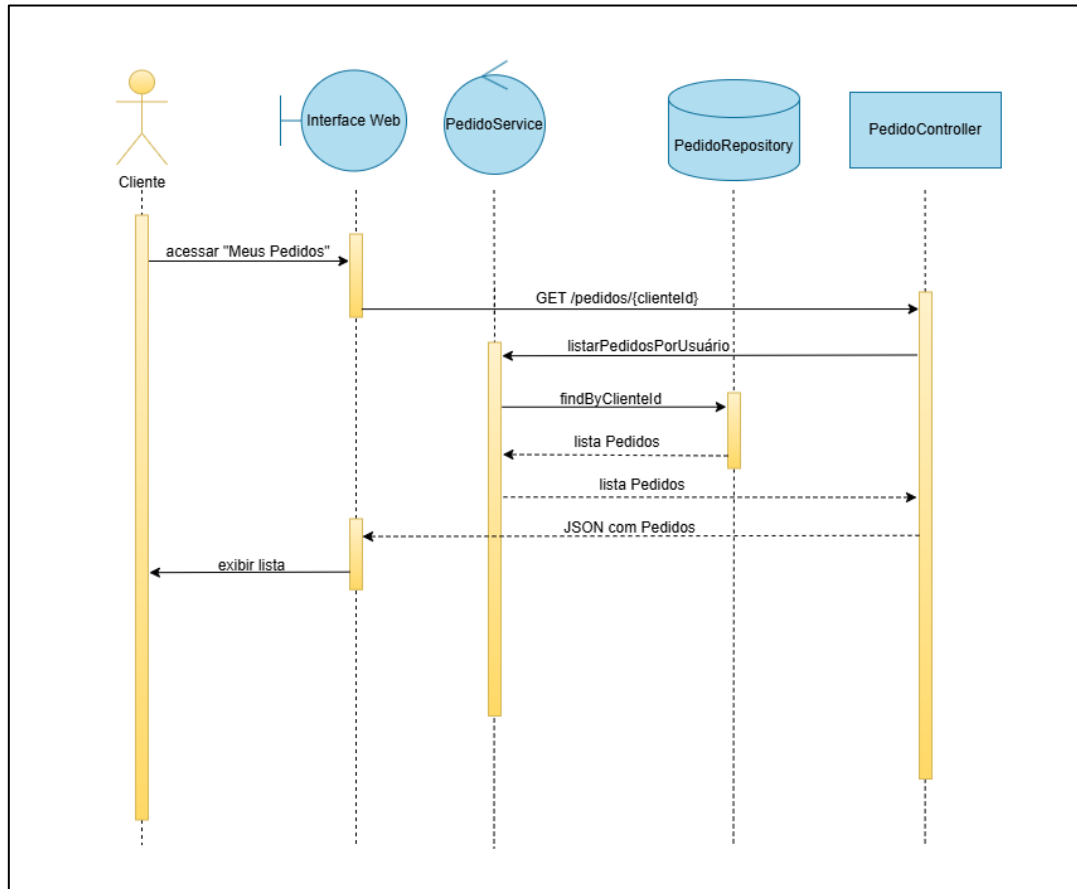
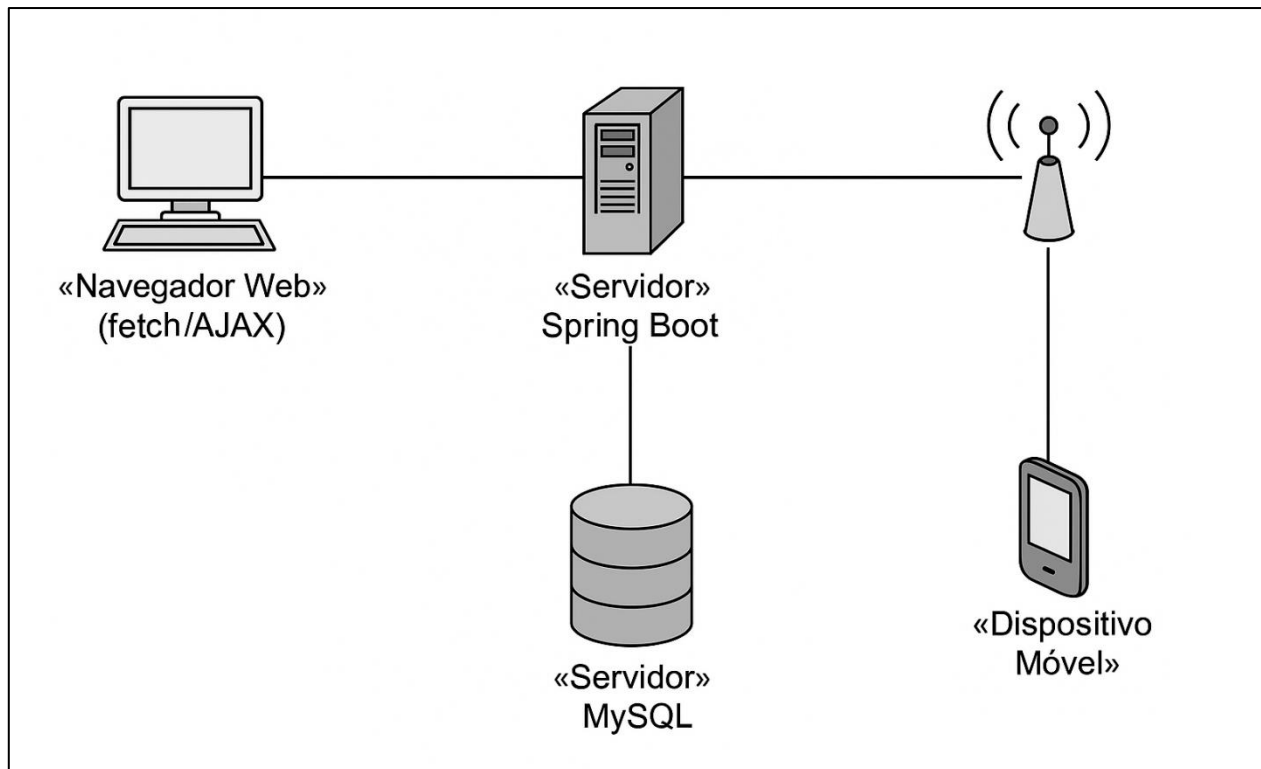


Figura 5.3 – Diagrama de Sequência Histórico Pedidos

## 10. Visão de Implantação

Esta seção apresenta a arquitetura física onde o sistema da livraria será implantado, destacando os nós físicos, dispositivos de acesso, servidores e suas interconexões. A visão de implantação descreve os componentes de software sendo executados sobre os elementos de hardware, oferecendo uma compreensão clara de como o sistema será operacionalizado no ambiente real.

A Figura 6 ilustra o modelo de implantação do sistema.



**Figura 6 – Visão de Implantação do Sistema de Livraria**

Na Figura 6 observa-se os seguintes nós físicos:

- **Navegadores Web (Usuários/Admins):** Representam os dispositivos utilizados pelos clientes e administradores para interagir com o sistema por meio da interface web. A comunicação com o backend ocorre via chamadas assíncronas (fetch/AJAX) sobre o protocolo HTTP(S).
- **Dispositivos Móveis:** Representam outro meio de acessar a aplicação, através de um navegador do próprio dispositivo móvel.
- **Servidor:** Responsável por hospedar a aplicação principal da livraria, processando as requisições dos clientes (terminais e dispositivos móveis). Este servidor executa os componentes de backend desenvolvidos em Java (Spring Boot), além de expor as APIs REST que interagem com o banco de dados.
- **Servidor DB:** Responsável por armazenar todos os dados persistentes da aplicação, como informações de livros, pedidos, clientes, vendas e estoque. O sistema utiliza o banco de dados MySQL, acessado via JPA/Hibernate pela camada de repositório da aplicação.



## 11. Visão de Implementação

A estrutura de implementação do sistema da livraria segue uma arquitetura em camadas bem definida, fortemente alinhada à Visão Lógica já apresentada neste documento. Dessa forma, evita-se a redundância de diagramas nesta seção.

O sistema é implementado utilizando o framework Spring Boot, que favorece a separação de responsabilidades e a organização modular do código. A decomposição do software contempla as seguintes camadas principais:

- **Camada de Apresentação (Controller):** Responsável por receber requisições HTTP (principalmente via fetch/AJAX) e direcioná-las às camadas de serviço. Os controladores expõem endpoints REST que representam os casos de uso da aplicação.
- **Camada de Serviço (Service):** Contém as regras de negócio e a lógica de processamento dos dados. Atua como intermediária entre os controladores e a camada de persistência.
- **Camada de Acesso a Dados (Repository):** Utiliza o Spring Data JPA para abstrair o acesso ao banco de dados relacional MySQL. Essa camada define interfaces que interagem com entidades persistentes por meio do Hibernate.
- **Camada de Modelo (Model/Entity):** Representa as entidades de domínio do sistema, como Cliente, Livro, Pedido, Venda e Estoque. Essas classes são mapeadas com anotações JPA para persistência automática.

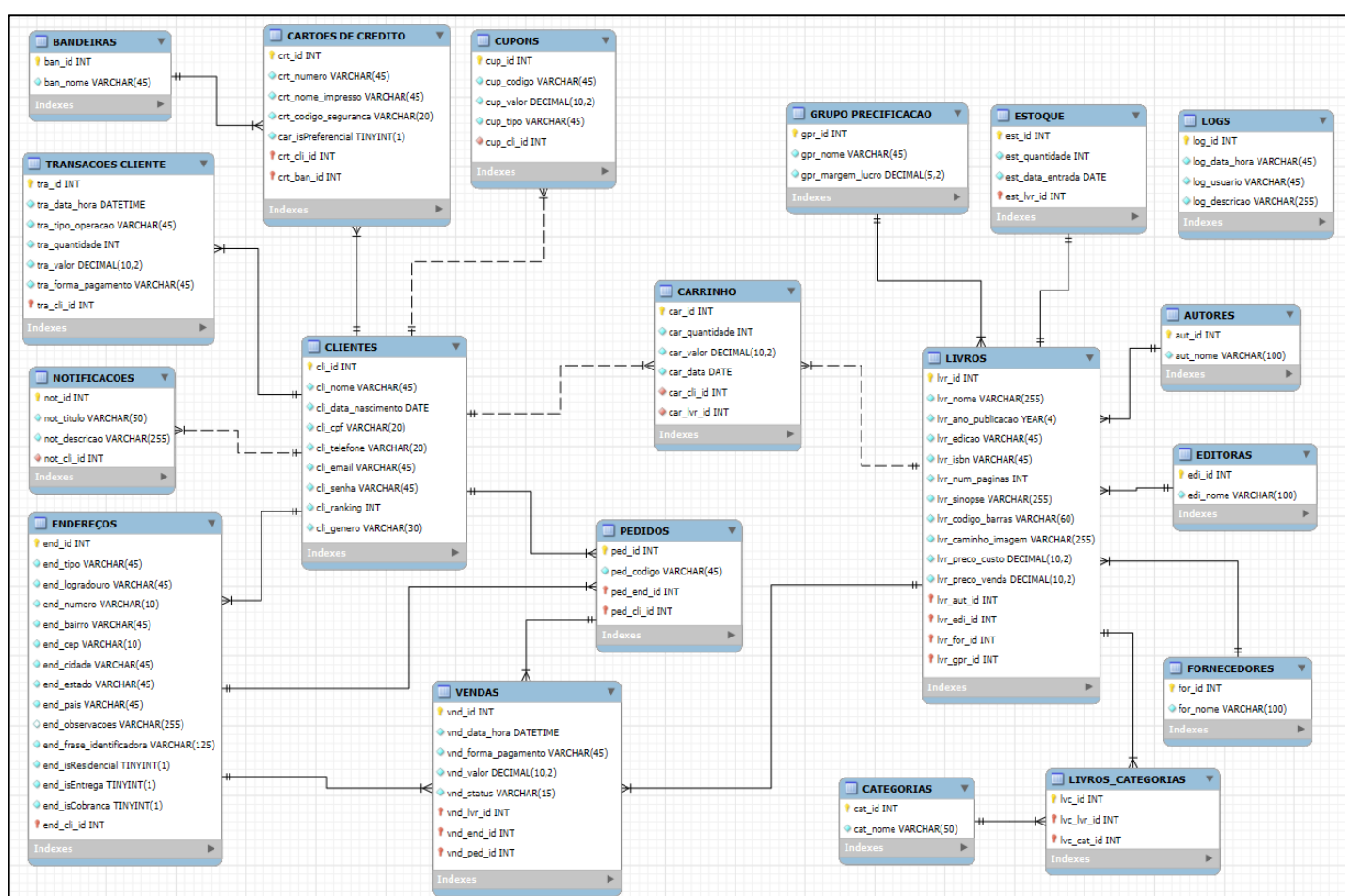
A implementação segue boas práticas de modularidade, favorecendo manutenção, testes unitários e escalabilidade. O código-fonte é organizado em pacotes separados conforme cada camada, o que reforça a coesão interna e a separação de responsabilidades.

## 12. Visão de Dados

O sistema da livraria utiliza um banco de dados relacional MySQL para persistência de dados, em conjunto com o framework Hibernate, que realiza o mapeamento objeto-relacional (ORM) entre as entidades Java e as tabelas do banco. O controle de transações é gerenciado pelo Spring Framework, assegurando a integridade e consistência dos dados em todas as operações realizadas.

O Modelo Entidade-Relacionamento (MER) da base de dados foi desenvolvido no MySQL Workbench. Este modelo contempla as principais entidades do sistema, como Cliente, Livro, Pedido, Venda, Estoque, entre outras, e define suas respectivas relações, como cardinalidade e integridade referencial.

A figura a seguir ilustra o Modelo Entidade-Relacionamento da base de dados:



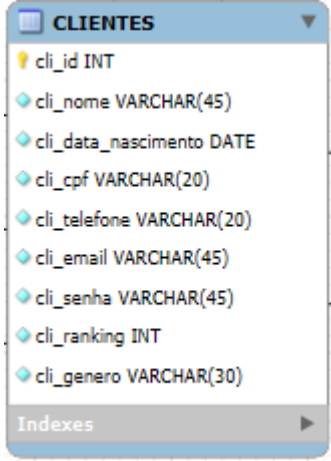
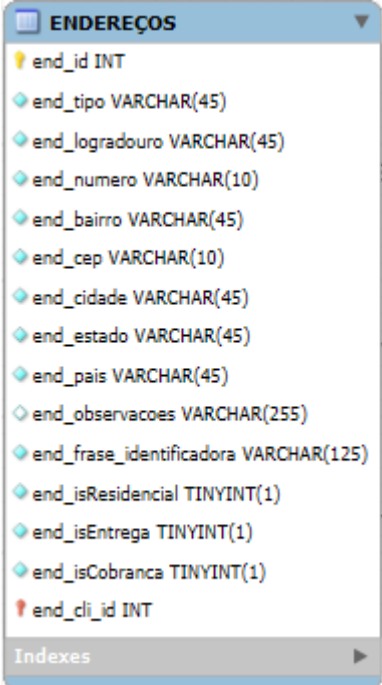
**Figura 7 – Modelo Entidade-Relacionamento do Banco de Dados do Sistema**

Nesta seção, apresentamos uma visão simplificada do mapeamento objeto-relacional realizado no sistema da livraria. O mapeamento foi feito de forma direta, ou seja, as classes de modelo da aplicação Java foram construídas com atributos que correspondem exatamente às colunas das tabelas no banco de dados MySQL.

Esse alinhamento direto entre o modelo de classes (implementado com Java e anotado com JPA/Hibernate) e o modelo relacional (MER do MySQL) facilita a manutenção do sistema, bem como o entendimento da estrutura de dados tanto por desenvolvedores quanto por administradores de banco.

A Tabela 1 apresenta alguns exemplos de mapeamentos entre as classes de modelo da aplicação e suas respectivas entidades no banco de dados, destacando que esse padrão é seguido de forma consistente em todo o sistema.

**Tabela 1: Mapeamento Objeto-Relacional**

Classe	Entidade
<div> <div>Cliente</div> <div> +int id  +String nome  +Date dataNascimento  +String cpf  +String telefone  +String email  +String senha  +int ranking  +String genero </div> </div>	
<div> <div>Endereco</div> <div> +int id  +String tipo  +String logradouro  +String numero  +String bairro  +String cep  +String cidade  +String estado  +String pais  +String observacoes  +String fraseIdentificadora  +boolean isResidencial  +boolean isEntrega  +boolean isCobranca </div> </div>	

<div data-bbox="317 351 649 911"> <p><b>Livro</b></p> <ul style="list-style-type: none"> <li>Integer id</li> <li>String nome</li> <li>Integer anoPublicacao</li> <li>String edicao</li> <li>String isbn</li> <li>Integer numPaginas</li> <li>String sinopse</li> <li>String codigoBarras</li> <li>String caminhaImagem</li> <li>BigDecimal precoCusto</li> <li>BigDecimal precoVenda</li> </ul> </div>	<div data-bbox="965 288 1319 967"> <p><b>LIVROS</b></p> <ul style="list-style-type: none"> <li>lv_id INT</li> <li>lv_nome VARCHAR(255)</li> <li>lv_ano_publicacao YEAR(4)</li> <li>lv_edicao VARCHAR(45)</li> <li>lv_isbn VARCHAR(45)</li> <li>lv_num_paginas INT</li> <li>lv_sinopse VARCHAR(255)</li> <li>lv_codigo_barras VARCHAR(60)</li> <li>lv_caminho_imagem VARCHAR(255)</li> <li>lv_preco_custo DECIMAL(10,2)</li> <li>lv_preco_venda DECIMAL(10,2)</li> <li>lv_aut_id INT</li> <li>lv_edi_id INT</li> <li>lv_for_id INT</li> <li>lv_gpr_id INT</li> </ul> <p>Indexes</p> </div>
--	---

### 13. Tamanho e Performance

O sistema da livraria foi projetado para operar em ambiente web, atendendo tanto administradores quanto usuários finais (clientes) que acessam a plataforma para consultar livros, realizar pedidos e acompanhar seu histórico de compras. Embora não se trate de um sistema com exigência de processamento intensivo ou de alta concorrência, é esperado que ele suporte uma base de dados moderada e um número razoável de acessos simultâneos.

Durante períodos promocionais, lançamentos ou eventos sazonais como o Natal e a Black Friday, é possível que o sistema passe por picos de utilização, com aumento significativo na quantidade de requisições, especialmente nas funcionalidades de pesquisa de livros, finalização de pedidos e consulta ao histórico de compras.

O backend do sistema é implementado em Java com Spring Boot, utilizando Hibernate para mapeamento objeto-relacional e persistência em um banco de dados MySQL. O front-end se comunica com o backend via requisições assíncronas (fetch/AJAX), o que melhora a experiência do usuário e evita recargas completas de página.

A arquitetura adotada permite boa escalabilidade vertical (em servidores mais potentes) e horizontal, caso se opte no futuro por uma estrutura em nuvem com balanceamento de carga. A aplicação também faz uso de paginação em consultas, cache em pontos estratégicos e controle de transações eficientes para garantir desempenho satisfatório mesmo em cenários de carga mais elevada.

Informações detalhadas sobre requisitos de desempenho e limitações esperadas de carga podem ser consultadas no documento de Requisitos.

## 14. Qualidade

O sistema da livraria foi desenvolvido com foco em oferecer uma experiência de uso fluida, confiável e segura, atendendo tanto administradores quanto clientes finais. Como o sistema lida com informações sensíveis, como dados pessoais, endereços e histórico de pedidos, além de transações comerciais envolvendo valores financeiros, a qualidade do software é um fator essencial para seu sucesso.

Durante o processo de design e implementação, foram priorizados atributos como:

- **Confiabilidade:** o sistema deve se comportar de maneira previsível e estável, mesmo em condições adversas ou de alta carga.
- **Segurança:** boas práticas de segurança foram adotadas, como validação de dados no backend, controle de autenticação/autorização e prevenção contra ataques comuns (ex: SQL Injection e Cross-Site Scripting).
- **Escalabilidade e desempenho:** a arquitetura foi planejada para permitir crescimento futuro e responder bem a variações de uso, especialmente em datas promocionais.
- **Manutenibilidade:** a estrutura modular com camadas separadas (Controller, Service, Repository) facilita futuras correções, melhorias ou expansões.
- **Usabilidade:** a interface web com chamadas assíncronas proporciona navegação ágil, minimizando recargas de página e melhorando a experiência do usuário.

Além disso, considerando que o sistema é acessado pela internet, mecanismos de proteção foram considerados, como camadas de autenticação para usuários e administradores, proteção de endpoints e boas práticas na configuração do servidor.

Maiores informações sobre os requisitos de qualidade podem ser encontradas no documento de Requisitos, que detalha aspectos como desempenho, disponibilidade, portabilidade e segurança da aplicação.

## 15. Cronograma Macro.

O cronograma macro do projeto da livraria online apresenta uma visão geral das principais entregas ao longo do desenvolvimento. Os prazos estão organizados por atividades técnicas e funcionais, com base nas funcionalidades essenciais do sistema.

<b>Resultado</b>	<b>Prazo Estimado</b>
Protótipo do CRUD de Clientes	Semana 2
Implementação do CRUD de Clientes	Semana 3
Testes automatizados do CRUD de Clientes	Semana 5
Protótipo completo do sistema	Semana 11
Implementação da funcionalidade de inserção de venda	Semana 12
Implementação da funcionalidade de devolução	Semana 13
Testes automatizados da inserção de venda	Semana 14
Testes automatizados da funcionalidade de devolução	Semana 15
Refatorações e ajustes finais	Semana 16

Observação: Os prazos apresentados são estimativas iniciais, com base na fase atual do projeto. Um cronograma mais detalhado será definido posteriormente durante o planejamento, podendo ser ajustado conforme surgirem novas prioridades ou requisitos.

## 16. Referências

DIAGRAMS.NET. *Draw.io – Diagramas Online*. Disponível em: <https://www.diagrams.net>. Acesso em: 13 abr. 2025.

HIBE, Hibernate ORM. *Hibernate Documentation*. Disponível em: <https://hibernate.org/orm/documentation/>. Acesso em: 13 abr. 2025.

MERMAID. *Mermaid Live Editor*. Disponível em: <https://mermaid.live>. Acesso em: 13 abr. 2025.

MYSQL. *MySQL Workbench – MySQL Developer Tools*. Disponível em: <https://www.mysql.com/products/workbench/>. Acesso em: 13 abr. 2025.

OMG – OBJECT MANAGEMENT GROUP. *Unified Modeling Language (UML)*. Disponível em: <http://www.omg.org/spec/UML>. Acesso em: 13 abr. 2025.

OPENAI. *ChatGPT – Assistente de Inteligência Artificial*. Disponível em: <https://chat.openai.com>. Acesso em: 13 abr. 2025.

SPRING.IO. *Spring Framework Documentation*. Disponível em: <https://spring.io/projects/spring-framework>. Acesso em: 13 abr. 2025.

FATEC MOGI DAS CRUZES. *Template de Documento de Visão de Projeto (DVP)*. Fornecido pelo docente da disciplina de Laboratório de Engenharia de Software, 2025.

FATEC MOGI DAS CRUZES. *Documento de Requisitos do Sistema da Livraria*. Fornecido pelo docente da disciplina de Laboratório de Engenharia de Software, 2025.