



UNIVERSIDADE ESTADUAL DO NORTE DO PARANÁ

Centro de Ciências Tecnológicas

Curso de Ciência da Computação

PROF. ANDRÉ LUIS ANDRADE MENOLLI

DISCENTES: DAVI RODRIGUES DAVANZO

JOANA SHIZU ONO DE CAMARGO

TRABALHO FINAL

BANDEIRANTES PR

2024

Parte A: Arquitetura Proposta para o Projeto

Visão Conceitual da Arquitetura

A arquitetura proposta para o projeto é baseada no padrão **MVC (Model-View-Controller)**, que separa a aplicação em três camadas principais para garantir modularidade, escalabilidade e manutenção. Essa divisão permite que a lógica de negócios, a interface com o usuário e o acesso ao banco de dados sejam implementados de forma independente, promovendo separação de responsabilidades.

A aplicação segue o seguinte esquema arquitetural:

1. Camada Model:

- Responsável pelas entidades do sistema e suas regras de negócio.
- Representa os objetos principais do domínio, como Aluno, Livro, Emprestimo, Devolucao, Reserva, etc.
- Fornece métodos para manipular os dados de forma encapsulada.

2. Camada Controller (ou Service):

- Responsável por processar as solicitações do sistema.
- Contém os serviços (Service) que encapsulam a lógica de negócios e controlam o fluxo entre o Model e a View.
- Exemplo: EmprestimoService, DevolucaoService, LivroService.

3. Camada DAO (Data Access Object):

- Responsável pela persistência dos dados no banco de dados.
- Implementa um padrão genérico com a interface GenericDAO, especializada em DAOs como AlunoDAO, LivroDAO, EmprestimoDAO, entre outros.
- Realiza operações como save, findById, findAll, update e delete.

4. Camada View:

- Responsável pela interação com o usuário.
- Recebe comandos e envia requisições para o Controller, exibindo mensagens de erro ou sucesso.

Descrição dos Elementos da Arquitetura e as Dependências

Elementos Principais

1. Model:

- **Entidades:**

- Aluno: Representa os alunos que podem emprestar livros.
- Livro: Representa os livros disponíveis ou emprestados pela biblioteca.
- Emprestimo: Representa o registro de um empréstimo realizado.
- Devolucao: Representa a devolução de um empréstimo, calculando atrasos e multas.
- Reserva: Registra a intenção de um aluno em reservar um livro.
- Titulo: Define atributos específicos de um título de livro (como área e autor).

- **Regras de Negócio:**

- Validação de dados (exemplo: verificação de pendências de alunos).
- Atualização de disponibilidade de livros após devoluções.

2. Controller (Service):

- **Responsáveis por encapsular lógica de negócios:**

- **Exemplo:** EmprestimoService realiza empréstimos verificando a disponibilidade dos livros e atualizando o status do Emprestimo no banco de dados.

- Coordena as interações entre a **View**, o **Model**, e o **DAO**.

3. DAO:

- Define métodos específicos de acesso ao banco de dados usando a interface genérica GenericDAO:

- save: Salvar uma nova entidade.
- findById: Consultar uma entidade por ID.
- findAll: Consultar todas as entidades.
- update: Atualizar uma entidade existente.
- delete: Remover uma entidade.

- **Exemplo:** LivroDAO é especializado em operações relacionadas à entidade Livro.

4. View:

- Interage com o usuário (funcionário da biblioteca).
- Coleta dados necessários (como ID do aluno ou lista de livros).
- Apresenta mensagens de erro ou confirmação com base nas respostas do Controller.

Dependências

- **View -> Sistema (Controller):**

- A View chama o Sistema para realizar operações, como registrar empréstimos ou devoluções.

- **Controller -> DAO:**

- Os Controllers (Services) utilizam os DAOs para acessar o banco de dados e manipular entidades.

- **DAO -> DataSource:**

- Os DAOs interagem com o banco de dados por meio da conexão gerenciada pelo DataSource.

- **Model -> Controller e DAO:**

- As entidades do Model são trafegadas entre o Controller e o DAO para validação e persistência.

Padrões Arquiteturais Escolhidos

Padrão MVC (Model-View-Controller)

Por que MVC foi escolhido?

1. Separar Responsabilidades:

- A arquitetura separa claramente a lógica de negócio (Model), a interface com o usuário (View), e o controle do fluxo de informações (Controller).

2. Facilidade de Manutenção e Extensão:

- Mudanças na interface gráfica (View) não afetam a lógica de negócios ou a estrutura de persistência.
- Novos requisitos podem ser facilmente incorporados sem comprometer outras camadas.

3. Testabilidade:

- Cada camada pode ser testada de forma independente, permitindo maior confiabilidade e controle de qualidade.

Padrão DAO (Data Access Object)

Por que DAO foi escolhido?

1. Encapsulamento do Acesso ao Banco de Dados:

- Todas as operações de persistência são realizadas por uma camada específica (DAO), isolando o acesso ao banco.


2. Reutilização de Código:

- A interface genérica GenericDAO permite que diferentes DAOs compartilhem a mesma estrutura básica, reduzindo redundâncias.


3. Facilidade de Substituição:

- Mudanças na tecnologia de persistência (por exemplo, trocar Hibernate por JPA) podem ser feitas de forma isolada na camada DAO.

b. Adicionar funcionalidades para cadastrar Livros e Alunos (inclusive com interface gráfica).



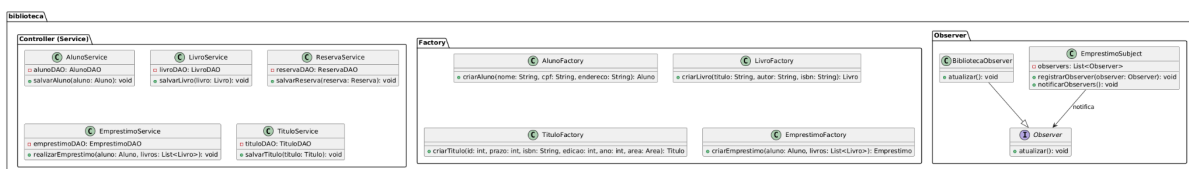
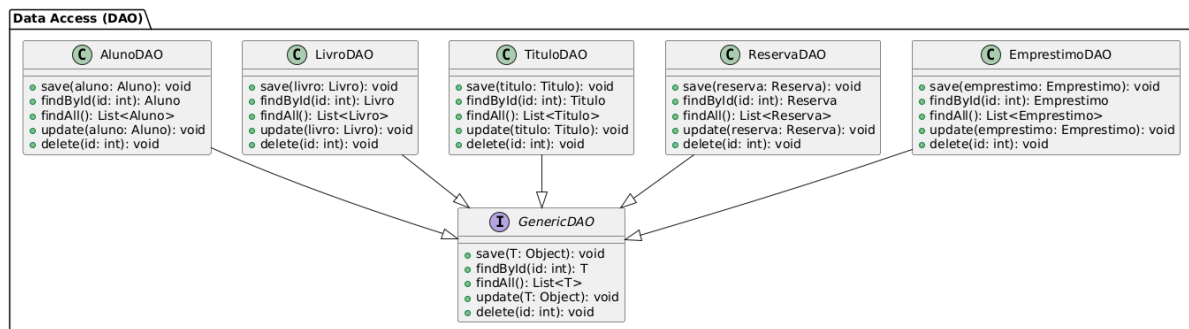
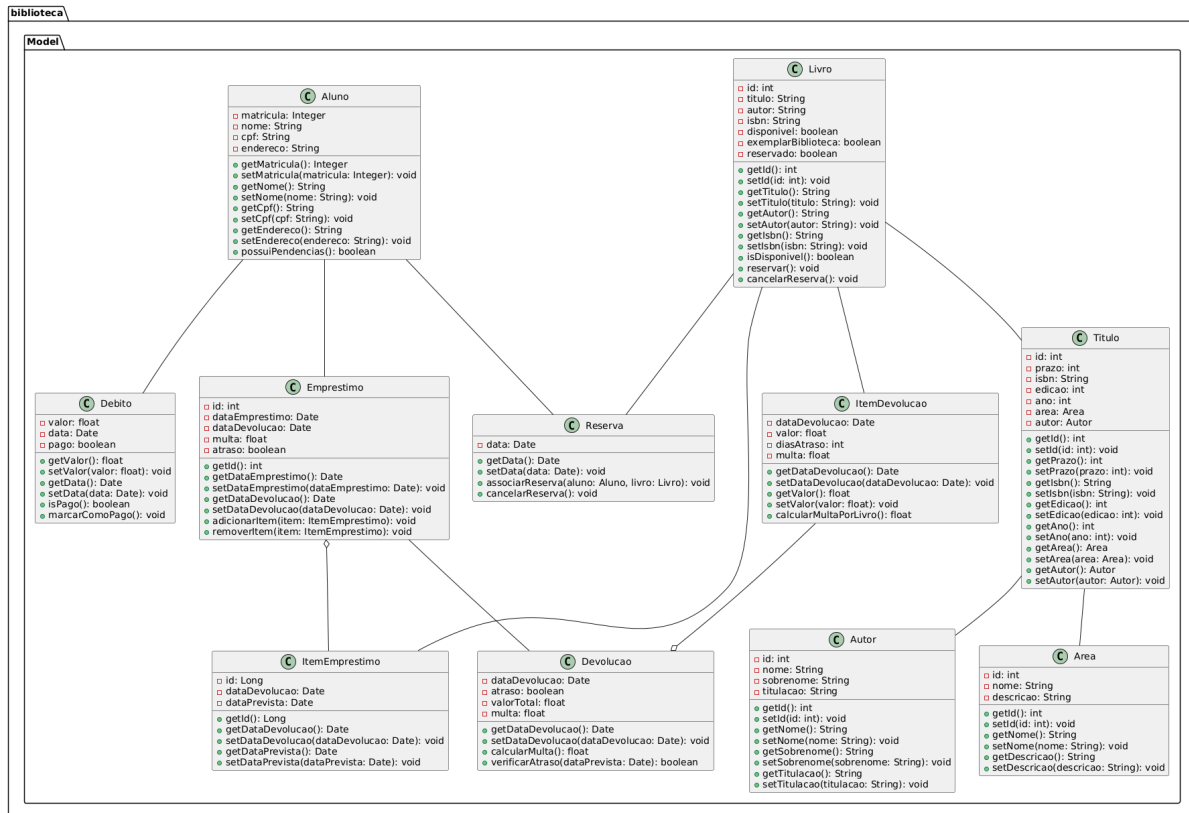
A screenshot of a graphical user interface window titled "Cadastrar Livro". The window has a dark gray title bar with standard window controls (minimize, maximize, close) on the right. The main area is light gray and contains three text input fields stacked vertically. The first field is labeled "Título do Livro:", the second "ISBN:", and the third "Autor:". Below these fields is a button labeled "Cadastrar Livro". The "Título do Livro:" field is currently selected, indicated by a blue border.



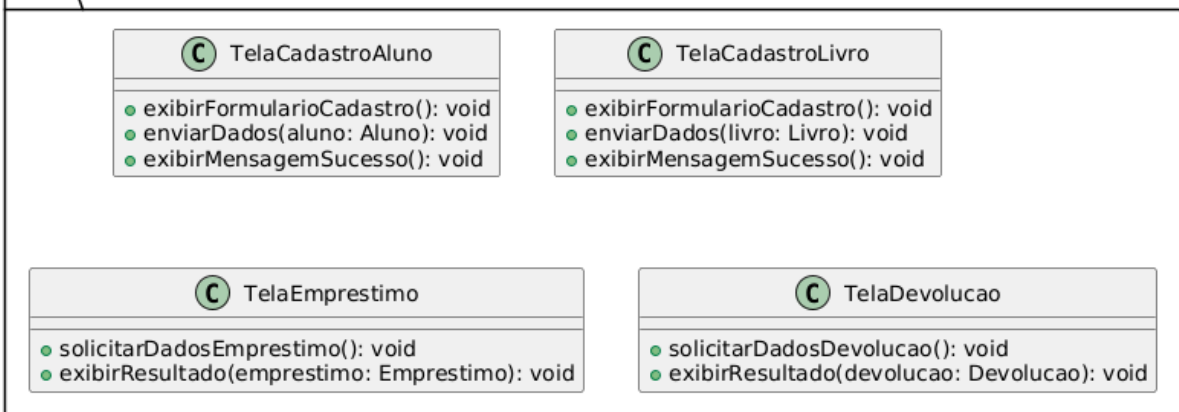
A screenshot of a graphical user interface window titled "Cadastrar Aluno". The window has a dark gray title bar with standard window controls (minimize, maximize, close) on the right. The main area is light gray and contains three text input fields stacked vertically. The first field is labeled "Matrícula:", the second "Nome do Aluno:", and the third "CPF:". Below these fields is a button labeled "Cadastrar Aluno". The "Matrícula:" field is currently selected, indicated by a blue border.

c. Finalizar a implementação do Caso de Uso Emprestar Livro

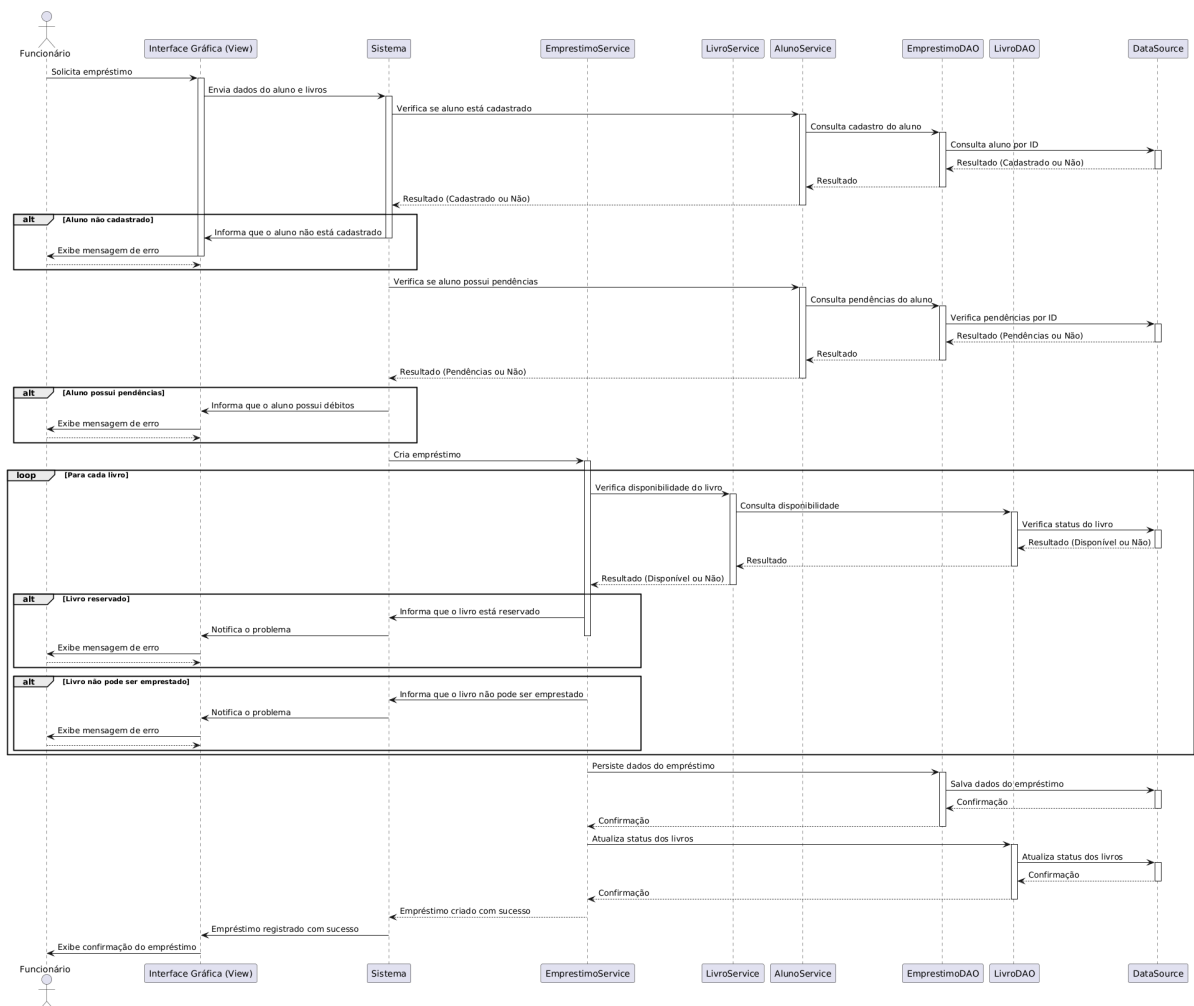
Diagrama de Classe



View



Finalização do Diagrama de Sequência



d. Caso de Uso: Devolver Livro

Ator Principal

- Funcionário

Objetivo

Permitir a devolução de livros emprestados, verificando se houve atraso e calculando possíveis multas.

Fluxo Principal (Caminho Básico)

1. O funcionário inicia o processo de devolução.
2. O sistema solicita a identificação do empréstimo e dos itens (livros) a serem devolvidos.
3. O funcionário informa os dados do empréstimo e os itens de devolução.
4. O sistema verifica se os itens estão associados ao empréstimo informado.
5. O sistema calcula a data limite para devolução e verifica se houve atraso.
6. Caso não haja atraso:
 - O sistema atualiza o estado do livro como disponível.
 - O sistema registra a devolução no banco de dados.
7. Caso haja atraso:
 - O sistema calcula a multa correspondente.
 - O sistema notifica o funcionário sobre o valor da multa.
 - O sistema registra a devolução e a multa no banco de dados.
8. O sistema confirma a devolução para o funcionário.

Fluxo Alternativo

3.a. Empréstimo não encontrado:

3.a.1 O sistema informa que o empréstimo não foi localizado. 3.a.2 O sistema encerra o caso de uso.

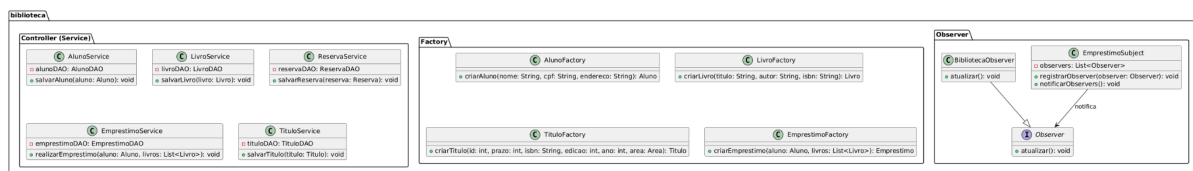
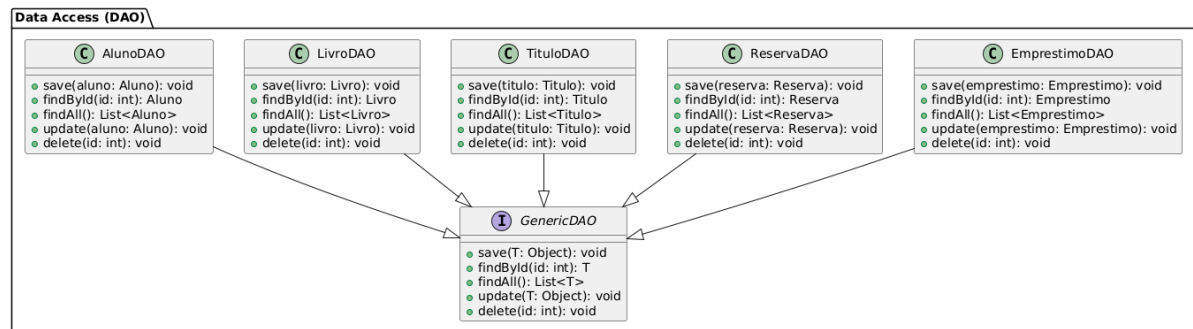
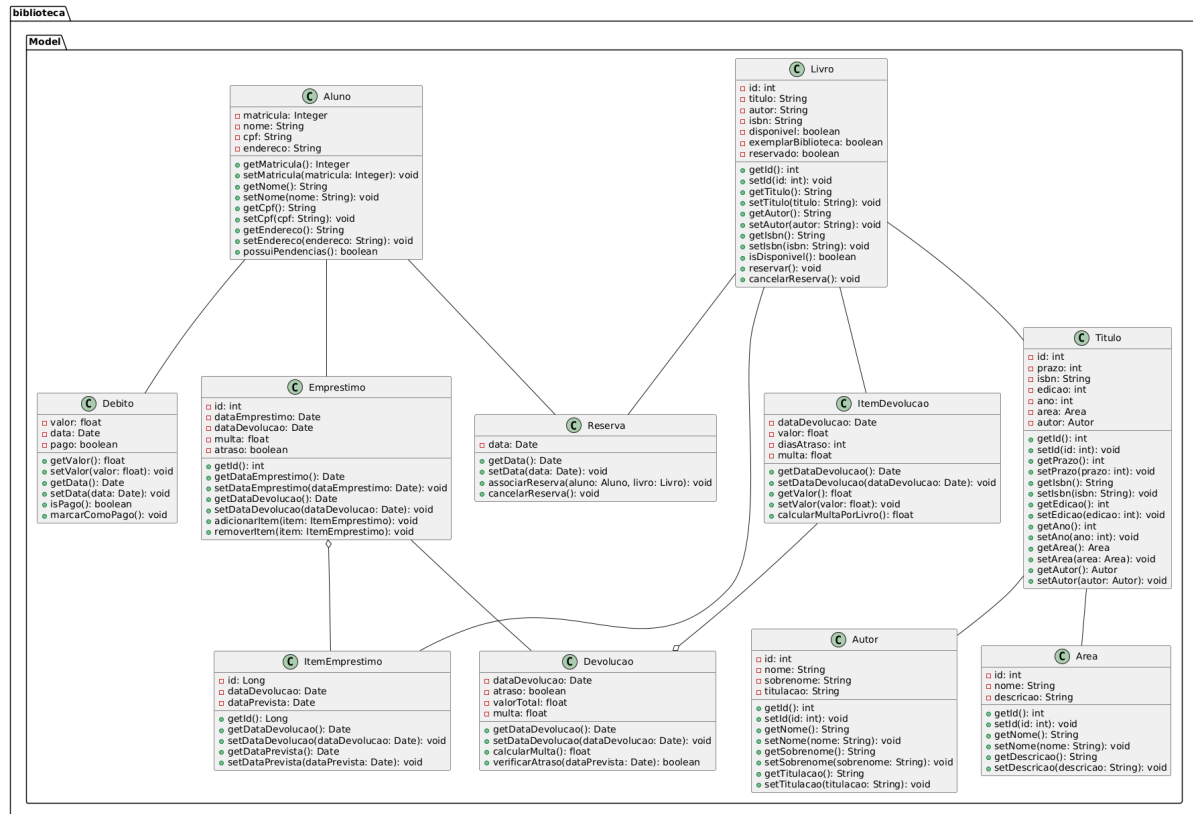
4.a. Livro não associado ao empréstimo:

4.a.1 O sistema informa que o item não pertence ao empréstimo especificado. 4.a.2 O sistema solicita que o funcionário corrija os dados informados.

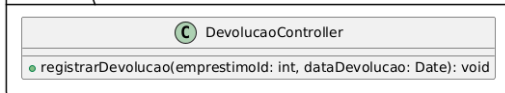
7.a. Multa não paga imediatamente:

7.a.1 O sistema registra a multa como pendente para o aluno associado ao empréstimo.

Diagrama de Classe



Controller



View

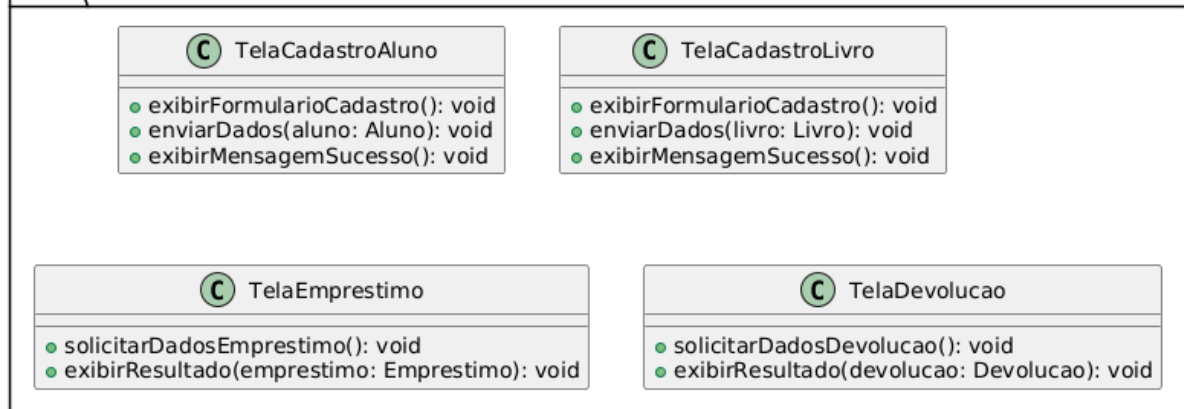
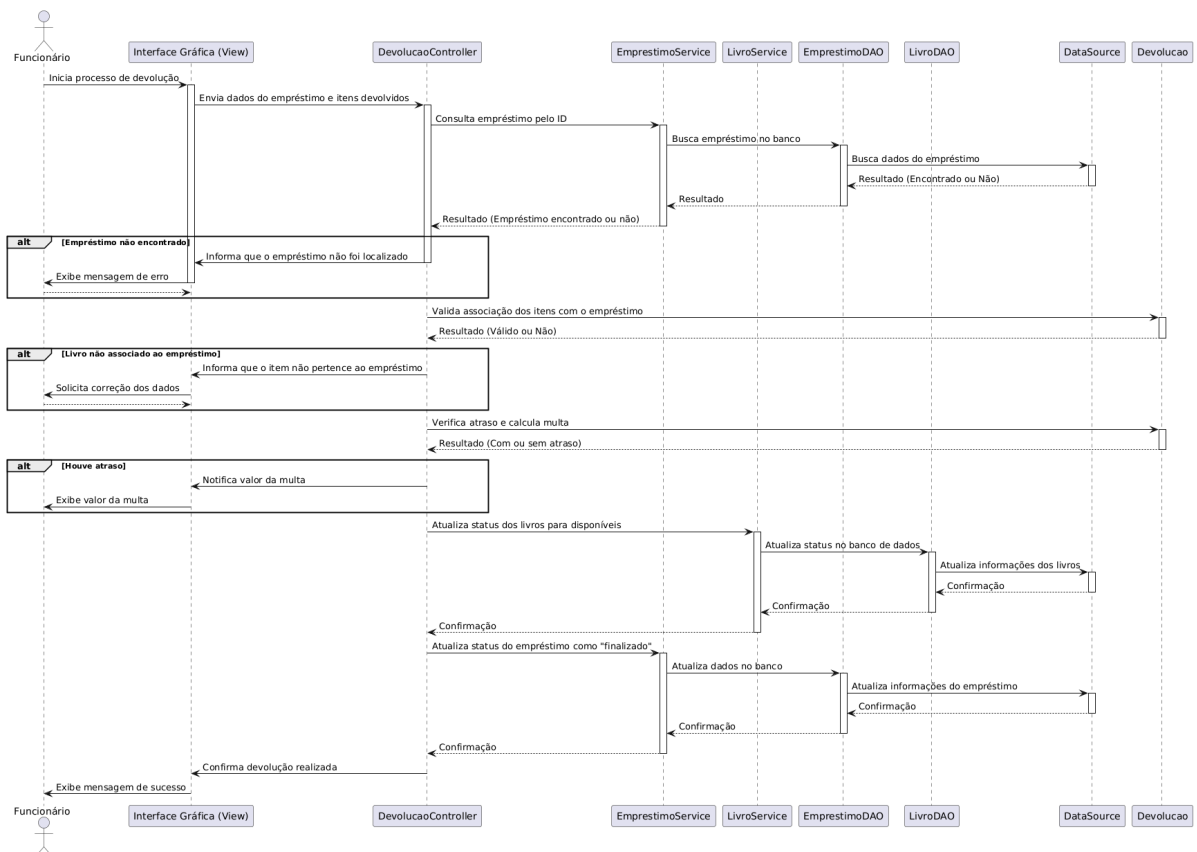


Diagrama de Sequência



e. Três Padrões de Projeto no Sistema

O sistema contempla pelo menos três padrões de projeto, implementados em diferentes camadas:

1. **Padrão Factory** (Camada Model):

- O padrão Factory é utilizado para a criação de objetos de maneira encapsulada, reduzindo a complexidade de inicialização e promovendo reutilização de código.
- **Exemplo:** Classes AlunoFactory, LivroFactory, EmprestimoFactory.

2. **Padrão Observer** (Camada Service):

- Implementado para notificar mudanças em objetos importantes para outras partes do sistema.
- **Exemplo:** Classes BibliotecaObserver e EmprestimoSubject, que permitem que alterações em empréstimos notifiquem outros componentes.

3. **Padrão DAO (Data Access Object)** (Camada DAO):

- Utilizado para isolar a lógica de acesso ao banco de dados, centralizando as operações de persistência em uma camada específica.
- **Exemplo:** Interface GenericDAO e suas implementações como AlunoDAO, LivroDAO, e DevolucaoDAO.

f. Persistência com o Padrão DAO

A camada de persistência do projeto utiliza o padrão DAO para gerenciar as operações de acesso ao banco de dados. A interface GenericDAO garante uma estrutura unificada para as operações CRUD (Create, Read, Update, Delete). Cada entidade possui um DAO específico que implementa a interface genérica, permitindo flexibilidade e manutenção centralizada.

Vantagens do uso do padrão DAO:

1. **Modularidade:** Cada DAO é independente e responsável por uma única entidade.
2. **Reutilização:** O código genérico da interface GenericDAO reduz redundância.
3. **Flexibilidade:** Mudanças no banco de dados ou na tecnologia de persistência podem ser feitas sem afetar outras partes do sistema.

g. Custo de Suporte a Outros SGBDs

A adaptação do sistema para suportar um outro tipo de SGBD (Sistema Gerenciador de Banco de Dados) requer os seguintes ajustes e custos potenciais:

1. Alterar configuração do DataSource:

- Atualizar as credenciais e configurações de conexão no arquivo de configuração (ex.: persistence.xml para JPA).
- Custo: Baixo, pois é uma alteração pontual.

2. Compatibilidade de Dialectos SQL:

- Caso o SGBD utilize um dialeto SQL diferente, ajustes podem ser necessários nas consultas nativas.
- Custo: Moderado, dependendo da extensão das diferenças.

3. Driver JDBC:

- Instalação do driver apropriado para o novo SGBD.
- Custo: Baixo, pois a maioria dos drivers é gratuita e fácil de configurar.

4. Testes de Integração:

- Executar testes para verificar a compatibilidade do sistema com o novo SGBD.
- Custo: Moderado, devido ao tempo de execução e verificação.

5. Treinamento da Equipe:

- Caso o novo SGBD seja desconhecido pela equipe, pode ser necessário treinamento.
- Custo: Alto, dependendo do nível de experiência necessário.