

Ana Beatriz Reis Alves

GERENCIAPOST

SISTEMA DE GERENCIAMENTO DE POSTS

Instituto Federal de Educação, Ciência e Tecnologia Baiano – Campus Catu

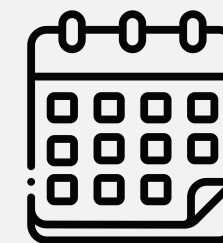
01 INTRODUÇÃO

02 OBJETIVOS

03 FLUXO DO SITE

04 DIAGRAMAS DE CLASSE

05 RESPONSABILIDADES

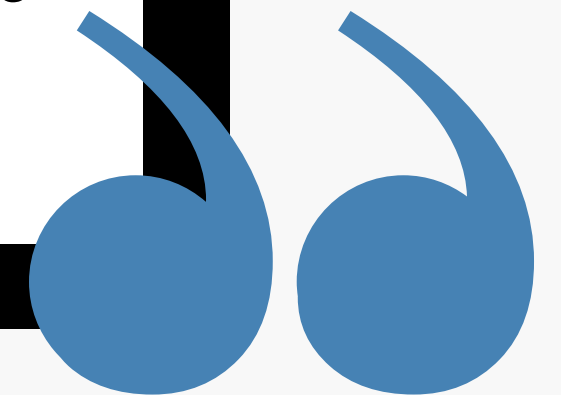


SUMÁRIO




INTRODUÇÃO

O projeto "GerenciaPost" é uma plataforma para gerenciar posts de diferentes tipos (texto, imagem, vídeo) de maneira eficiente e escalável. O sistema permite a criação, edição, exclusão e exibição de posts, proporcionando uma interface simples e amigável para os usuários. A estrutura do código foi organizada utilizando diversos padrões de projeto para promover a modularidade, facilidade de manutenção e extensibilidade.



OBJETIVOS


Desenvolver um sistema de gerenciamento de posts utilizando padrões de projeto para garantir a escalabilidade, manutenção eficiente e fácil implementação de novos recursos.



1.

POSTS FLEXIVEIS

Implementar tipos de posts flexíveis (texto, imagem, vídeo).



2.

EXTENSIBILIDADE

Garantir que o sistema seja fácil de estender para novos tipos de posts ou funcionalidades



3.

FACILIDADE DE USO

Criar uma interface simplificada para gerenciamento de posts .



4.

REUTILIZAÇÃO

Adotar padrões de projeto que promovam a reutilização de código e redução de acoplamento.

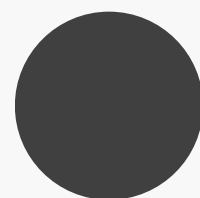


DIAGRAMA DE FLUXO

Criação de Posts

O usuário inicia a criação de um post, escolhendo entre os tipos disponíveis (Texto, Imagem, Vídeo). O sistema cria o post conforme o tipo selecionado, utilizando a classe apropriada.

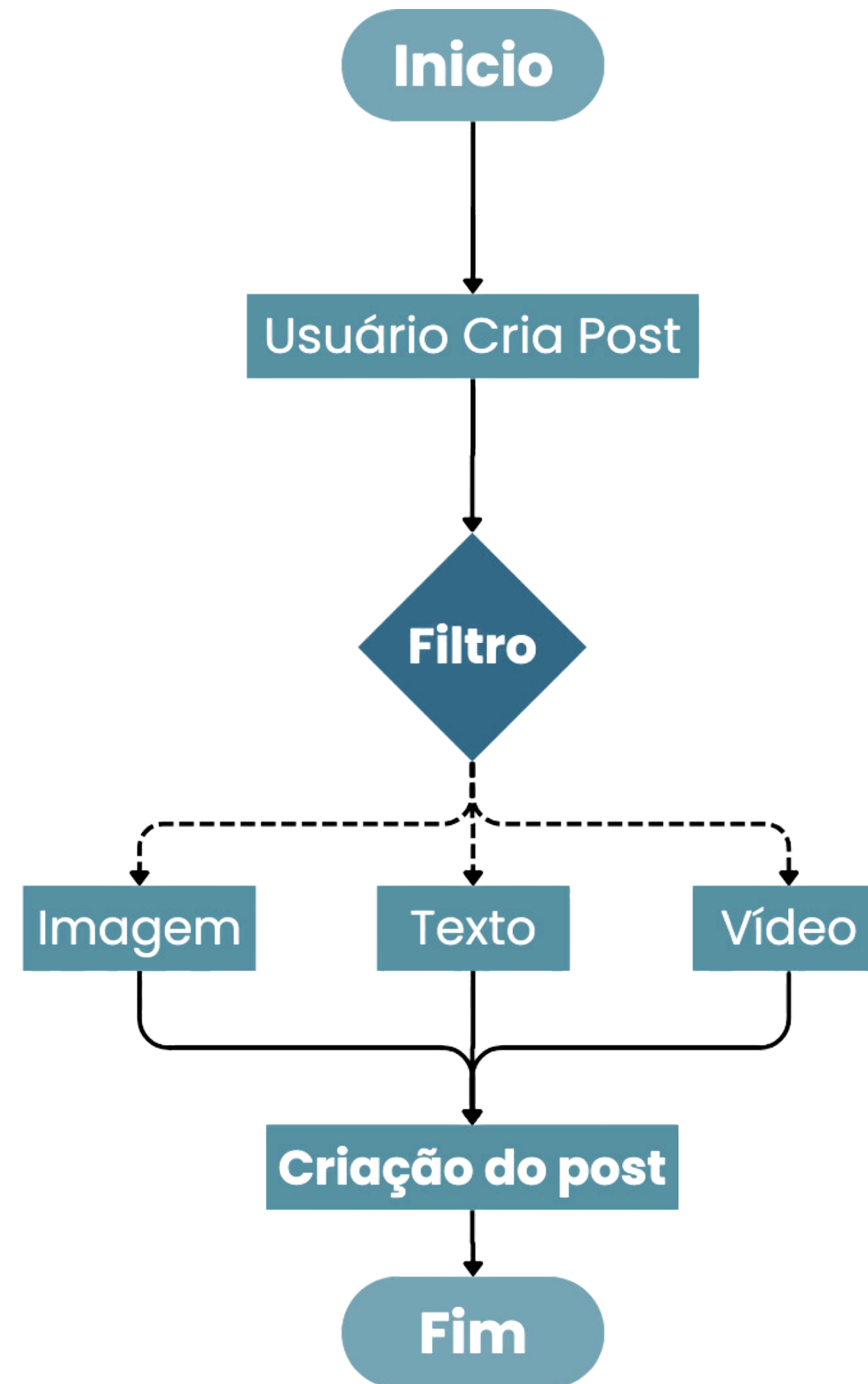
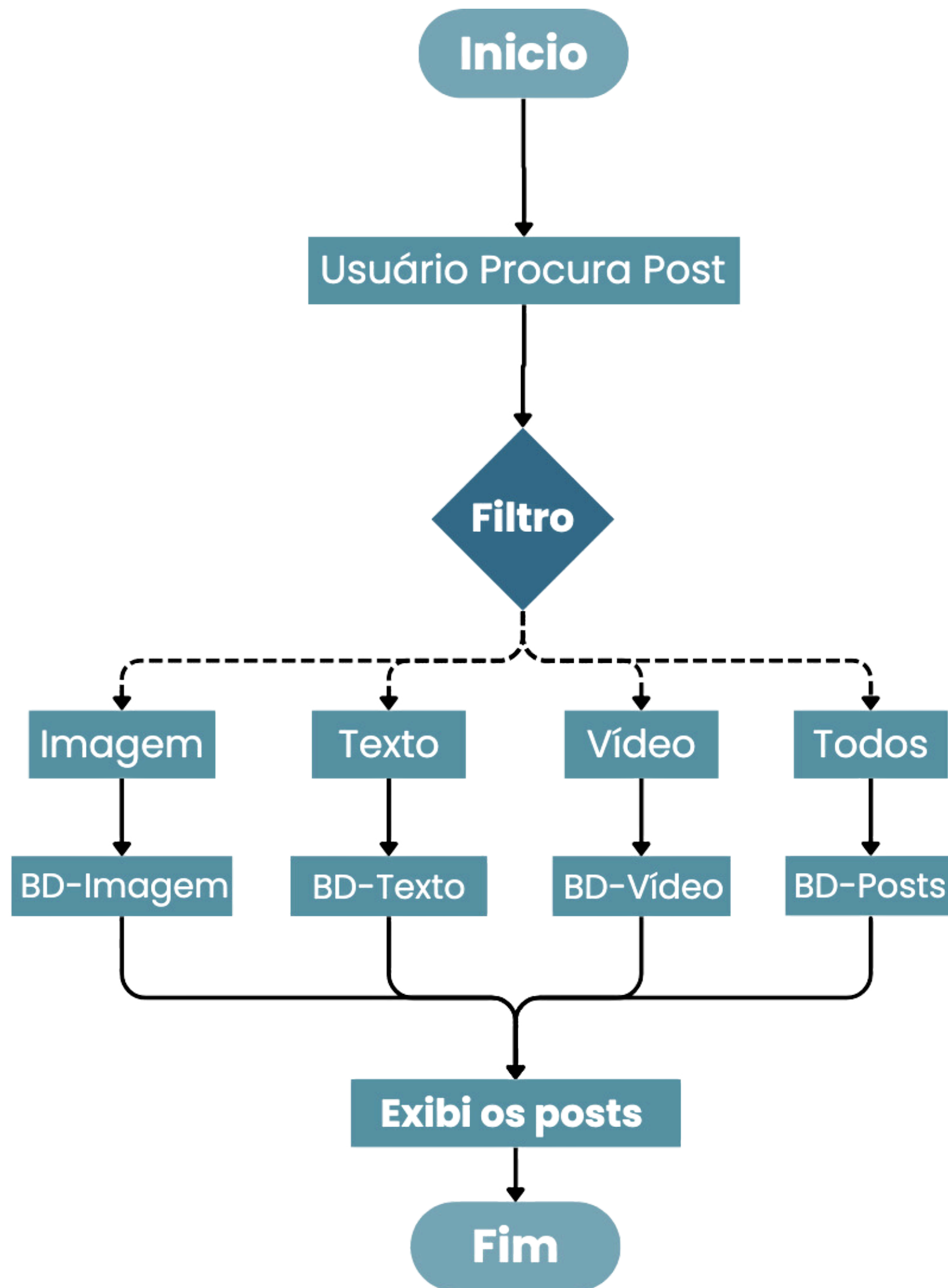


DIAGRAMA DE FLUXO



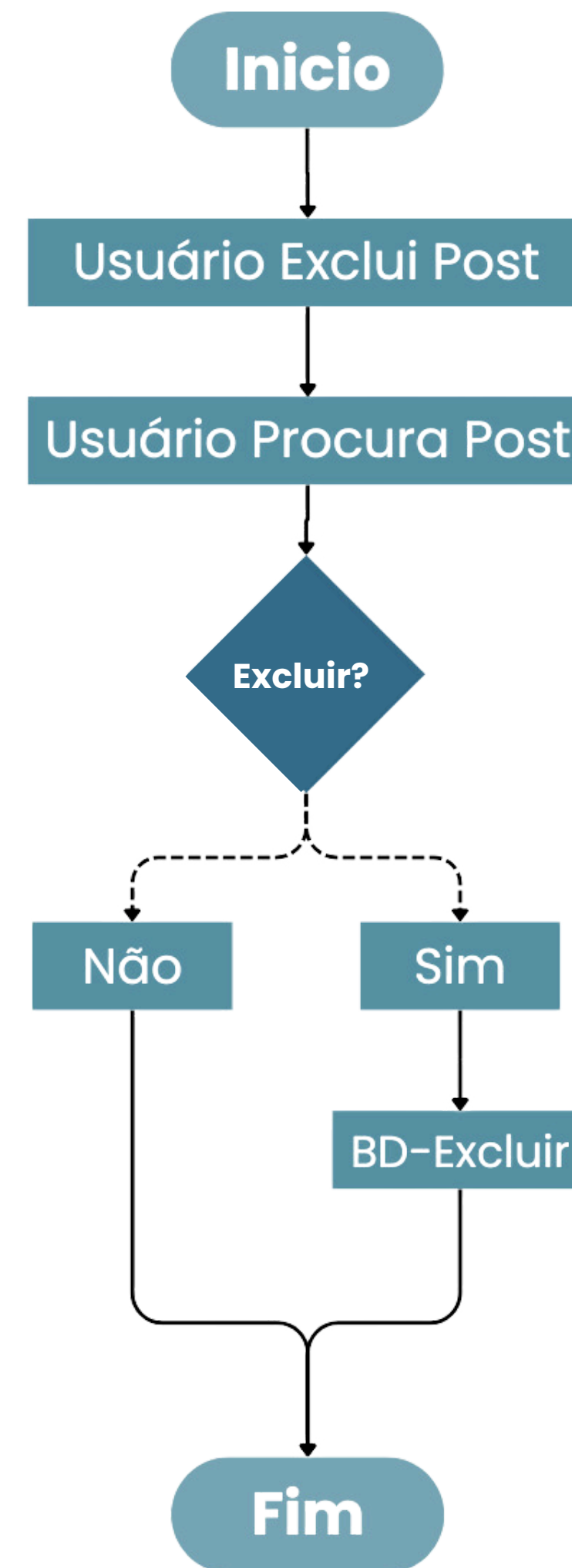
Procurar Posts

O usuário busca posts existentes, podendo aplicar filtros para refinar a busca por tipo de conteúdo (Texto, Imagem, Vídeo ou Todos). O sistema realiza a pesquisa no banco de dados com base nos critérios fornecidos.

DIAGRAMA DE FLUXO

Excluir Posts

O usuário localiza um post existente usando filtros, e tem a opção de excluí-lo. A exclusão é realizada e registrada no banco de dados.



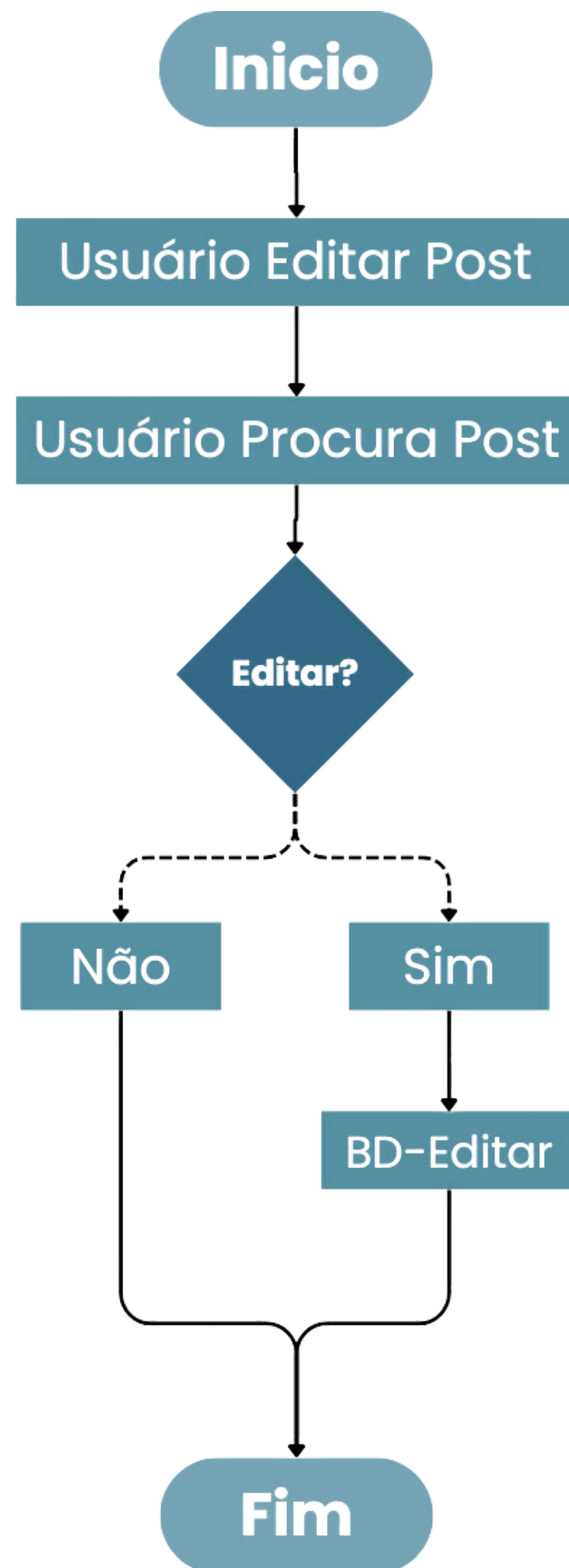


DIAGRAMA DE FLUXO

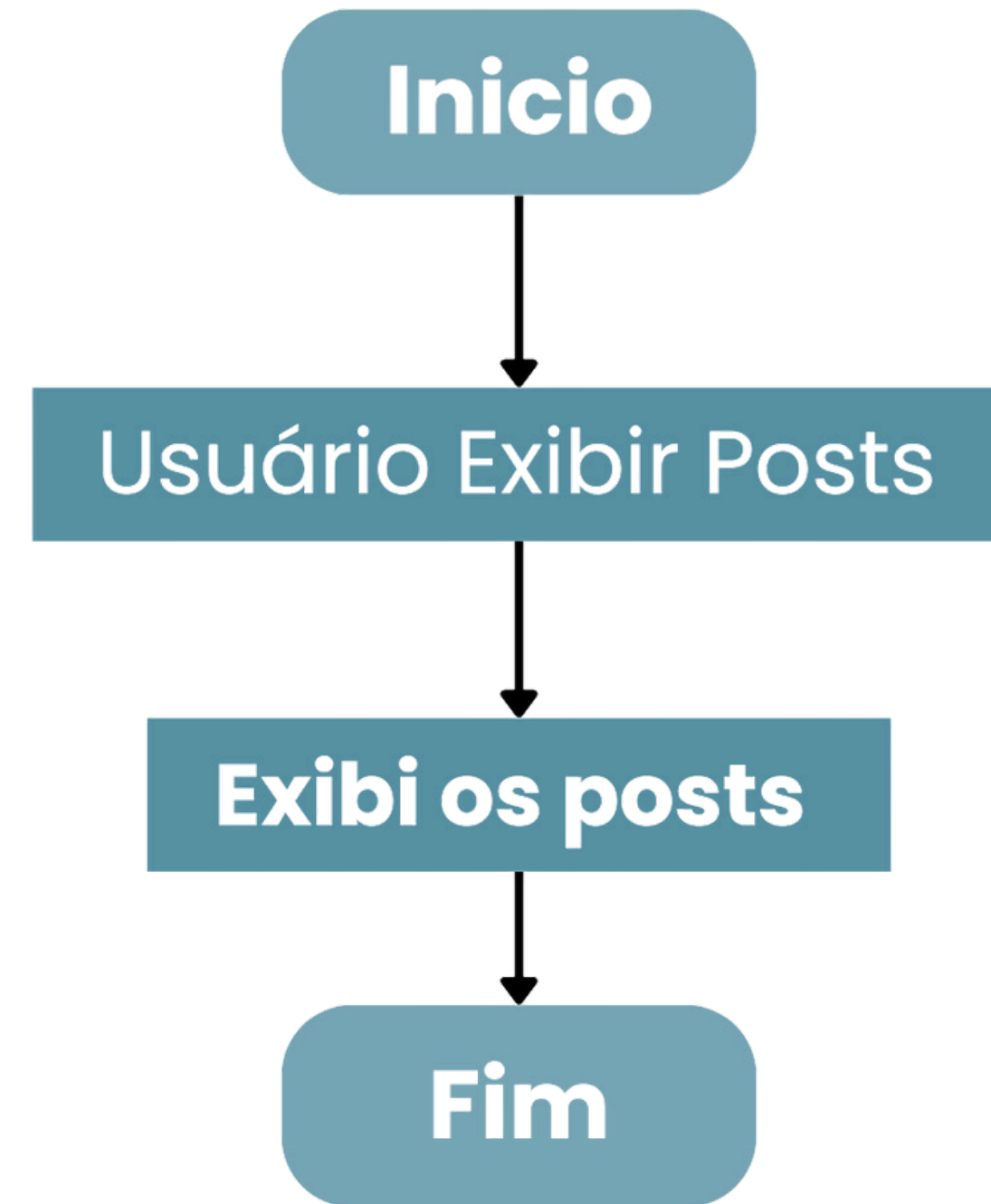
Editar Posts

O usuário encontra um post usando filtros e escolhe editá-lo. As alterações feitas no conteúdo do post são salvas no banco de dados.

DIAGRAMA DE FLUXO

Exibir Posts

Os posts são exibidos na página inicial. O post é carregado do banco de dados e apresentado na interface para leitura.



DESIGN PATTERNS

UTILIZADOS NO PROJETO

1

FACTORY METHOD

Criação

Utilizado na Criação
de posts

2

FAÇADE

Estrutural

Utilizado como
fachada

3

OBSERVER

Comportamental

Utilizado nas notificações
sobre mudanças

4

STRATEGY

Comportamental

Utilizado para exibição e
validação dos posts

5

SINGLETON

Criação

Usado na conexão com
o banco de dados.

SINGLETON

É aplicado à classe `Database`, garante que haja apenas uma instância da classe e, consequentemente, uma única conexão com o banco de dados.

- O método `getInstance()` retorna a única instância da classe `Database`.
- Os métodos `__clone()` e `__wakeup()` previnem a clonagem da instância, respectivamente. Esses métodos são privados e vazios, impedindo que novas instâncias sejam criadas a partir da existente.

Benefício: Evita a abertura de múltiplas conexões, otimizando recursos e evitando sobrecarga no banco de dados.

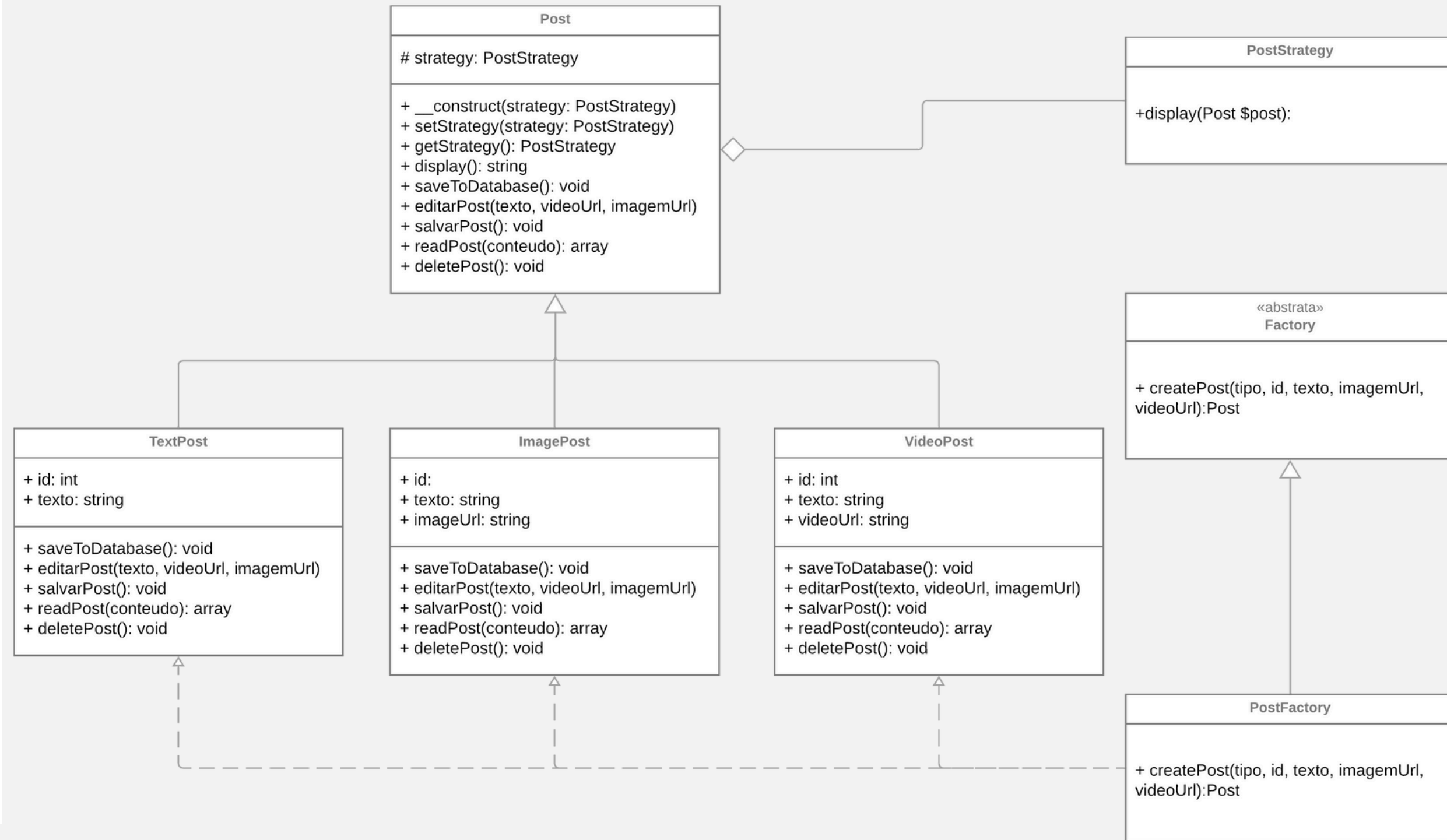
Database
- \$instance: Database - \$connection: PDO
- __construct() + getInstance(): Database - __clone() - __wakeup()

FACTORY METHOD

É utilizado para criar objetos de maneira flexível e dinâmica, permitindo que o código seja independente das classes concretas que estão sendo instanciadas. Implementação no projeto:

- A classe **PostFactory** é responsável por criar diferentes tipos de posts, como **TextPost**, **ImagePost** e **VideoPost**.
- As classes concretas são classes que implementam a **classe abstrata Post**.
- **Factory** é o creator, **PostFactory** é o concretecreeator, **Post** é o product e as classes **TextPost**, **ImagePost** e **VideoPost** são os concreteProduct.

Benefício: Promove a reutilização de código e reduz o acoplamento entre as classes concretas.

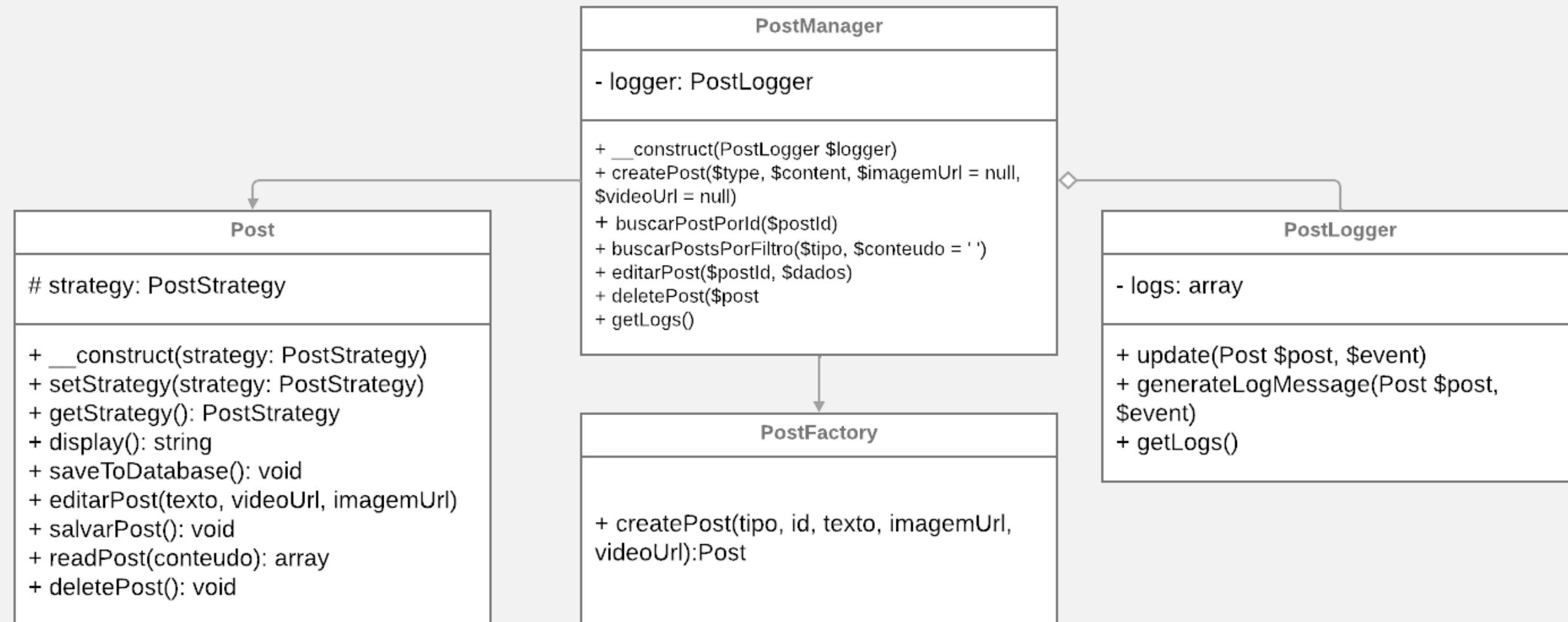


FAÇADE

Fornece uma interface simplificada para interações complexas entre classes ou subsistemas, escondendo a complexidade do back-end para quem usa o front-end.

- A classe **PostManager** atua como uma fachada, combinando as funcionalidades do **Factory Method**, das estratégias de validação e das operações de banco de dados em uma interface única e simples.
- Essa classe é usada para facilitar as operações relacionadas à **criação, atualização, exclusão e exibição de posts**.

Benefício: Simplifica a integração entre front-end e back-end e melhora a legibilidade do código.

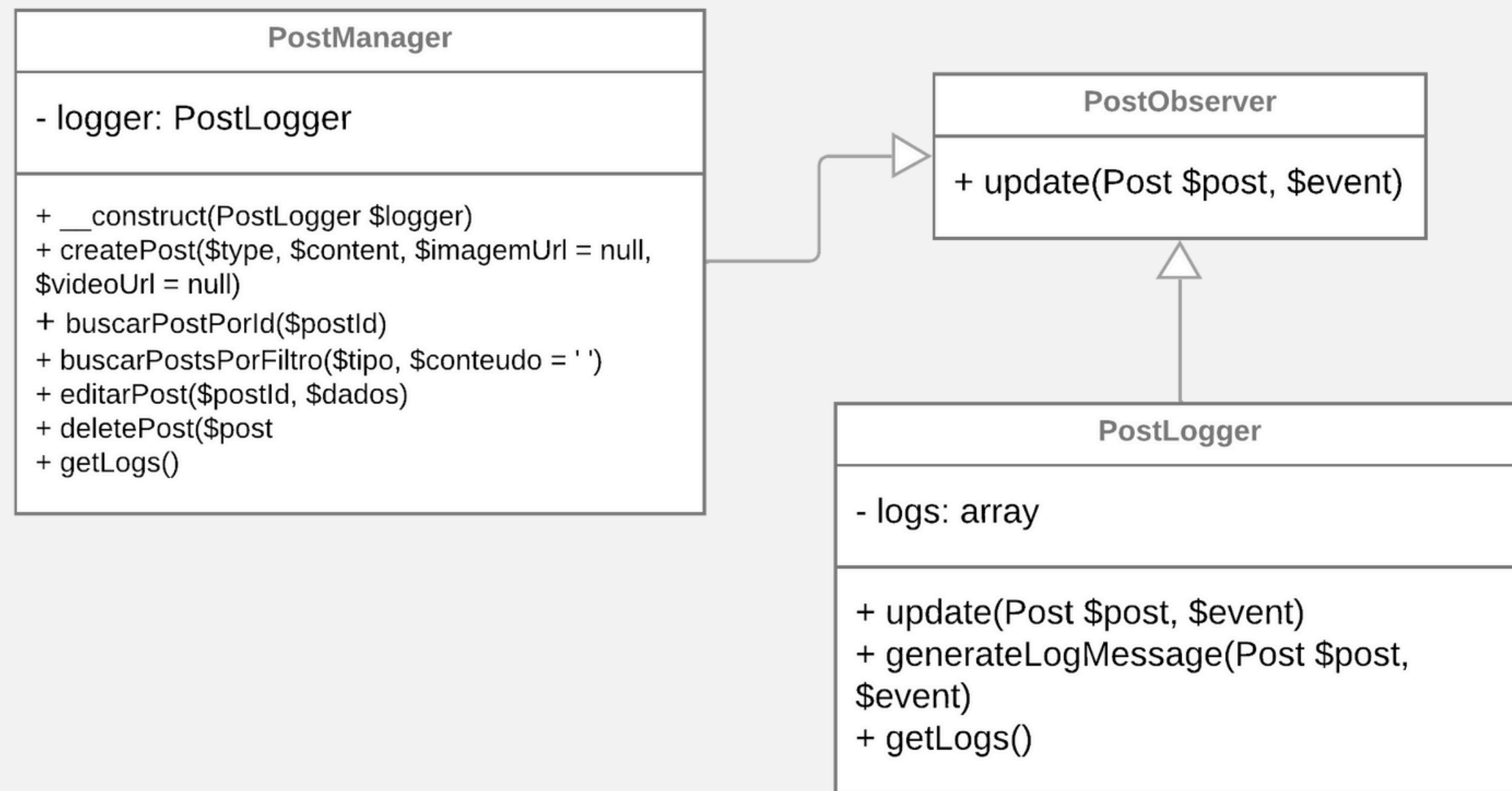


OBSERVER

É utilizado para **notificar** no sistema mudanças em um objeto de interesse, promovendo um sistema de eventos desacoplado. Implementação no projeto:

- A interface **PostObserver** define a estrutura básica para observadores.
- A classe **PostLogger** implementa essa interface e é notificada sempre que um post é criado, registrando o evento em um log.

Benefício: Permite adicionar novos comportamentos (como notificações ou auditorias) sem alterar as classes principais do sistema.



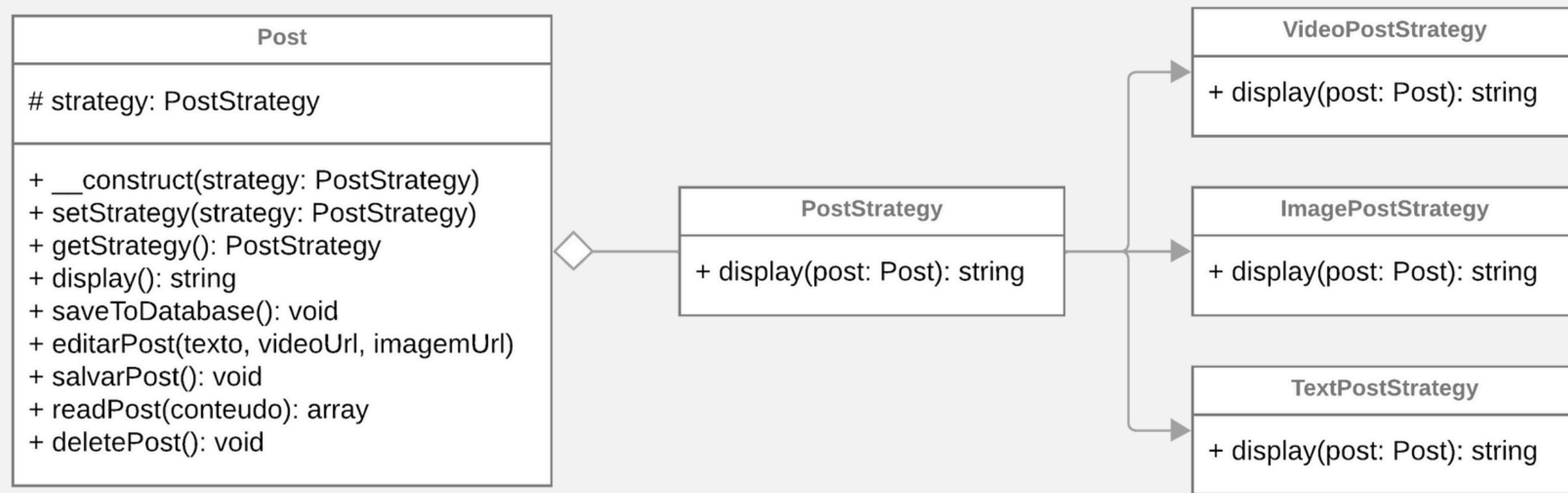
STRATEGY

Permite definir diferentes algoritmos ou comportamentos e selecioná-los em tempo de execução, dependendo do contexto.

Implementação no projeto:

- A interface **PostStrategy** define a estrutura para estratégias de **validação e exibição de posts**.
- **TextPostStrategy**, **ImagePostStrategy** e **VideoPostStrategy** implementam comportamentos específicos para cada tipo de post.
- Cada tipo de post utiliza a estratégia correspondente para exibir seus dados.

Benefício: Facilita a extensão do sistema para novos comportamentos sem modificar o código existente.



RESPONSABILIDADES

ANA BEATRIZ

Dev do projeto

1. Desenvolvimento do Front-end e Back-end
2. Implementação do padrão de projeto
Factory Method
3. Implementação do padrão de projeto
Facade
4. Implementação do padrão de projeto
Observer
5. Implementação do padrão de projeto
Strategy
6. Implementação de scripts JavaScript
7. Configuração do Banco de Dados





OBRIGADO

SISTEMA DE GERENCIAMENTO DE POSTS