Especificação de Requisitos de Software (ERS) - V2.1

Course	Padrões de projeto
Last Edited	@1 de dezembro de 2025 20:26

1. Fundamentos Arquiteturais e Restrições

1.1. Arquitetura Orientada a Padrões

A solução adota o **Padrão Fachada (Facade)** para fornecer uma interface de alto nível e simplificada para o cliente (CLI/GUI), isolando-o da complexidade de coordenação dos subsistemas1111. Para garantir modularidade, manutenibilidade e extensibilidade, a arquitetura interna dos subsistemas utiliza explicitamente os seguintes Padrões de Projeto:

- **Data Access Object (DAO):** No Subsistema de Usuários, para abstrair a persistência RDB.
- **Strategy:** No Subsistema de Dados, para permitir a troca de algoritmos de interpretação de imagens.
- **Factory Method:** No Subsistema de Alerta, para centralizar a criação de objetos de notificação (E-mail, API).
- **Adapter:** Utilizado pelo Subsistema de Dados (`LeitorImagemSHA`) e pelo Subsistema de Notificações (`NotificadorEmail`) para isolar o código de interfaces externas (SHA e biblioteca SMTP, respectivamente).

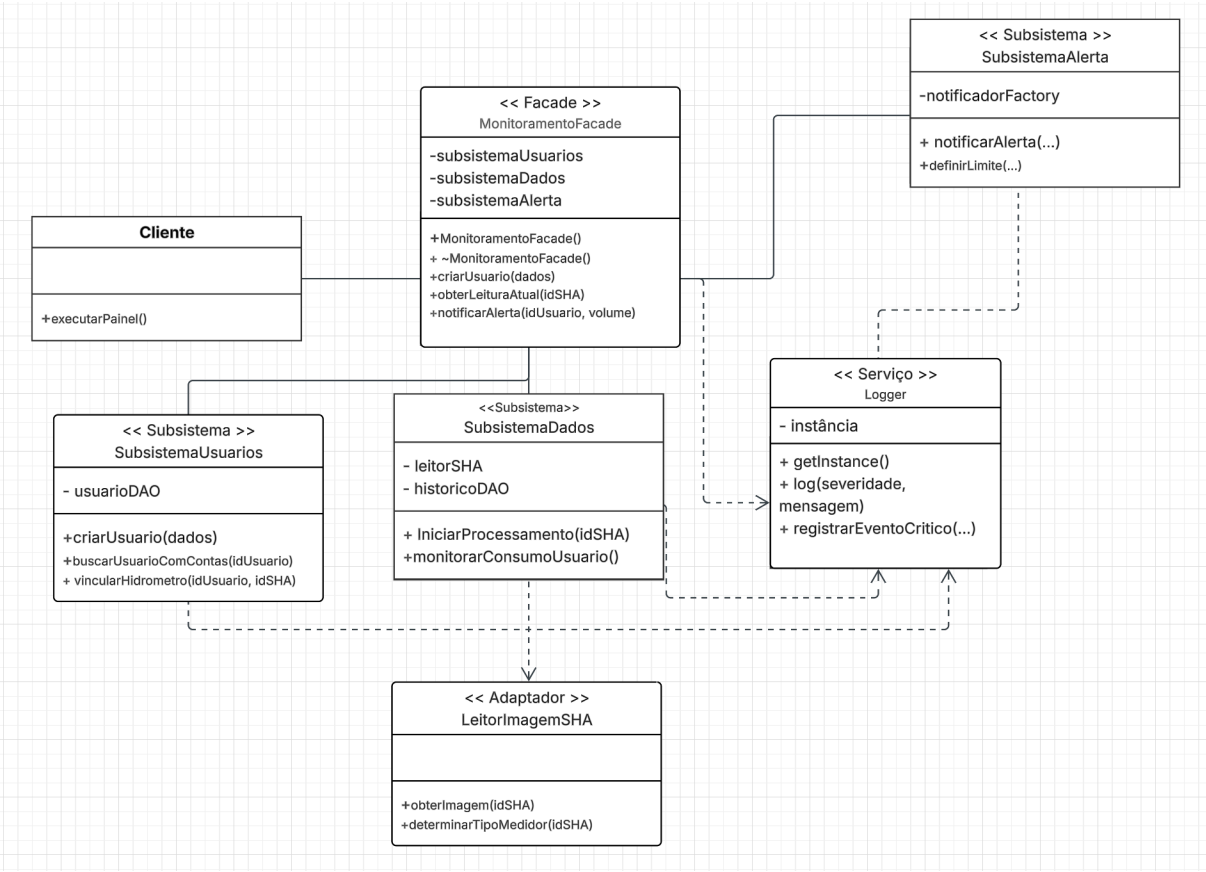
1.2. Restrições e Soluções (SHA)

As restrições de comunicação com o Simulador de Hidrômetros (SHA) são fundamentais e definem o design do `SubsistemaDados`.

Restrição Essencial	Consequência Arquitetural
O Painel não interfere e **não acessa** métodos internos dos SHA7777.	O `SubsistemaDados` e a `MonitoramentoFacade` não possuem dependências diretas de código com a interface de controle do SHA.

O consumo é obtido **exclusivamente pela leitura e interpretação de imagens** geradas pelos SHA.	A classe `LeitorImagemSHA` (Padrão Adapter) é o único componente que interage com a fonte externa (diretório monitorado) para buscar a imagem. A lógica de extração é delegada ao Padrão Strategy (OCR/Visão Computacional).

2. Diagrama geral:



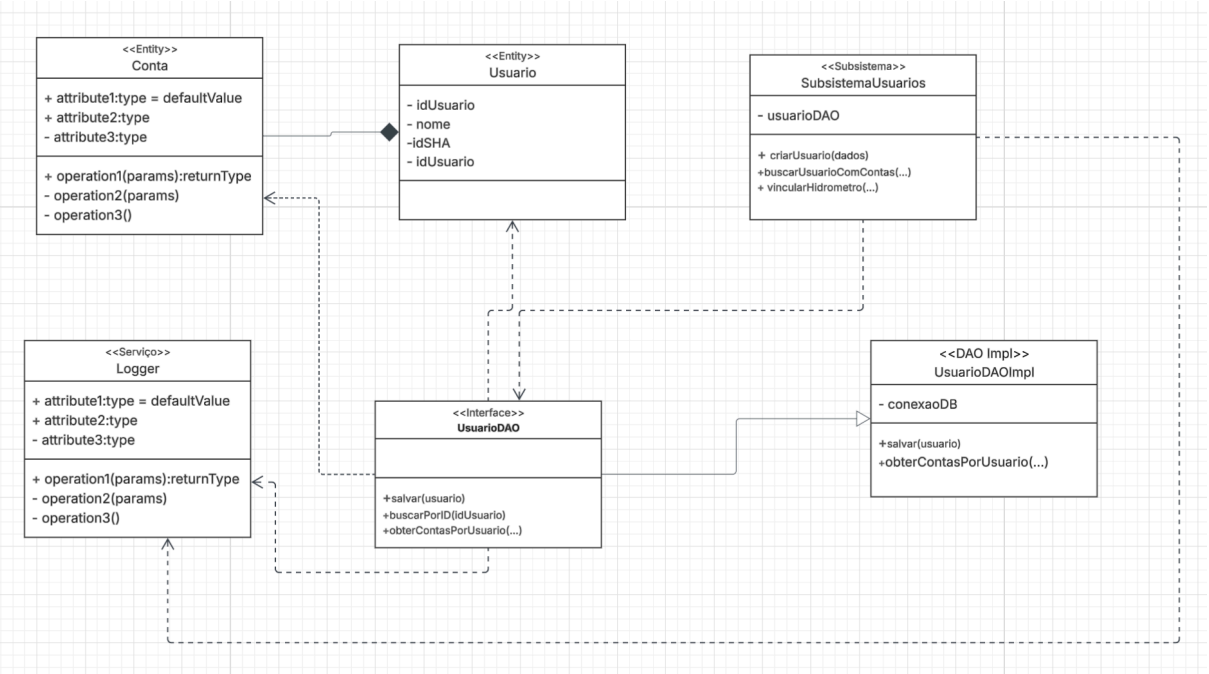
3. Detalhamento do Subsistema de Usuários (Padrão DAO)

O `SubsistemaUsuarios` gerencia o CRUD (Criação, Leitura, Atualização, Deleção) de usuários, perfis e o mapeamento de Contas/SHAs. Sua principal função é atuar como a **camada de lógica de negócio** acima da camada de persistência.

3.1. Funções e Justificativa do DAO

O Padrão **DAO (Data Access Object)** é empregado para isolar completamente a lógica de acesso ao Banco de Dados Relacional (RDB), garantindo que a classe `SubsistemaUsuarios` permaneça independente de *drivers* SQL ou ORMs específicos.

3.2. Diagrama



---

4. 📉 Detalhamento do Subsistema de Dados (Padrões Strategy e Adapter)

Este subsistema é o mais crítico em termos de restrições16. Ele deve ser capaz de processar imagens de diferentes tipos de medidores.

4.1. Funções e Justificativa da Strategy

O **Padrão Strategy** permite que o algoritmo de **interpretação da imagem (OCR/Visão Computacional)** possa ser trocado em tempo de execução. Isso é vital, pois medidores analógicos (ponteiros) exigem uma lógica de processamento de imagem diferente dos medidores digitais17.

4.2 Funções e Justificativa do Template Method:

4.2.1 Funções do Template Method no Painel

O Template Method (implementado pelas classes `FluxoProcessamentoBase` e `LeituraSHAProcessador`) tem três funções principais:

Função	Detalhe da Implementação

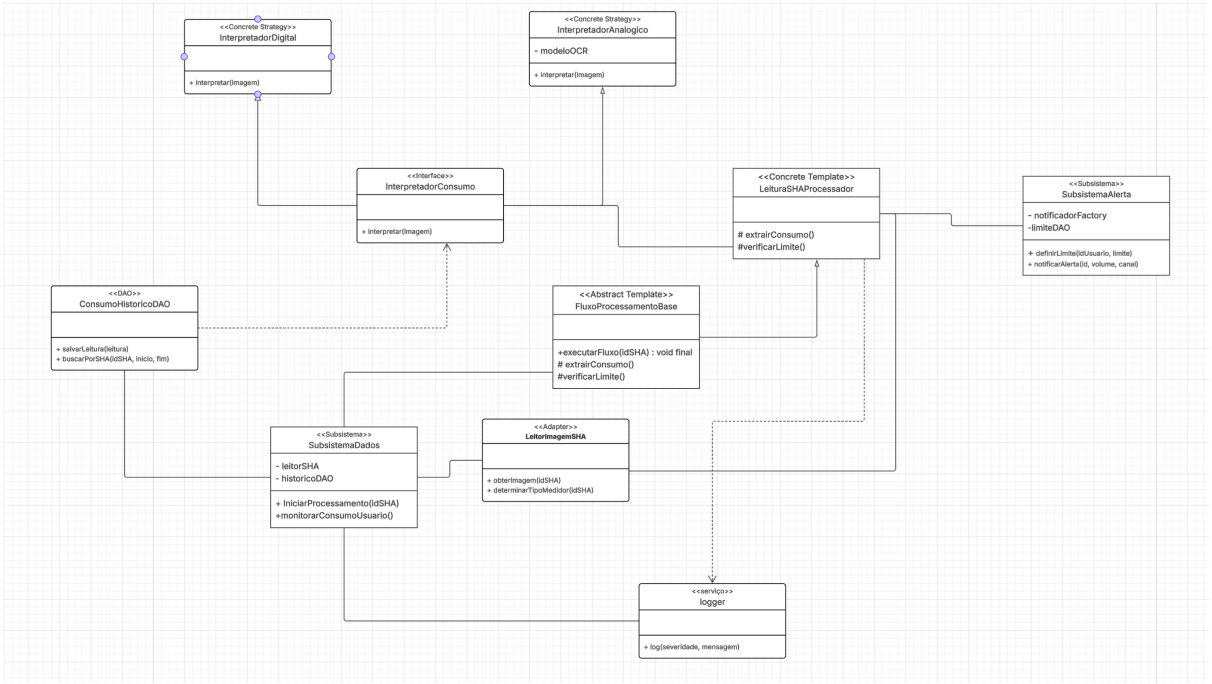
Definir o Esqueleto (Método Template)	O método `+executarFluxo(idSHA) : void final` define a sequência obrigatória de passos que o sistema deve seguir toda vez que processa uma leitura, sem permitir que essa ordem seja alterada: **Obter Imagem → Extrair Consumo → Salvar Histórico → Verificar Alerta.**
Garantir a Ordem Fixa (Final)	Ao ser definido como `final` (em C++), o método garante que nenhuma subclasse (`LeituraSHAProcessador`) possa sobrescrever a ordem das chamadas, eliminando o risco de erros de lógica, como verificar o limite antes de salvar o consumo.
Delegar o Trabalho Variável (Hooks)	A classe abstrata (`FluxoProcessamentoBase`) define **métodos hook** protegidos (`#extrairConsumo()` , `#verificarLimite()`) que as subclasses concretas devem implementar. Isso move a complexidade de como a extração é feita — chamando a Strategy — para a subclasse, mantendo a classe base enxuta.

2. Justificativa Arquitetural (Por que usá-lo?)

O uso do Template Method é justificado por resolver diretamente o problema de controle do fluxo de dados:

- **Controle Rigoroso do RF 2.1:** Resolve o requisito de **garantir a ordem de execução** do processo de monitoramento. Sem ele, a Fachada teria que chamar quatro métodos sequenciais, e a ordem poderia ser quebrada acidentalmente por um desenvolvedor.
- **Princípio Aberto/Fechado (OCP):** A classe base (`FluxoProcessamentoBase`) está **fechada para modificação** (a ordem é fixa), mas está **aberta para extensão** (novas classes que implementam o Template podem surgir se o fluxo de leitura mudar no futuro, como adicionar um passo de autenticação da imagem).
- **Centralização da Coordenação:** A lógica complexa de coordenação do Subsistema de Dados, que antes estava confusa (e foi removida do `SubsistemaDados`), agora está centralizada e encapsulada no `FluxoProcessamentoBase` , promovendo o **Princípio da Responsabilidade Única (SRP)**.

4.3 . Diagrama:



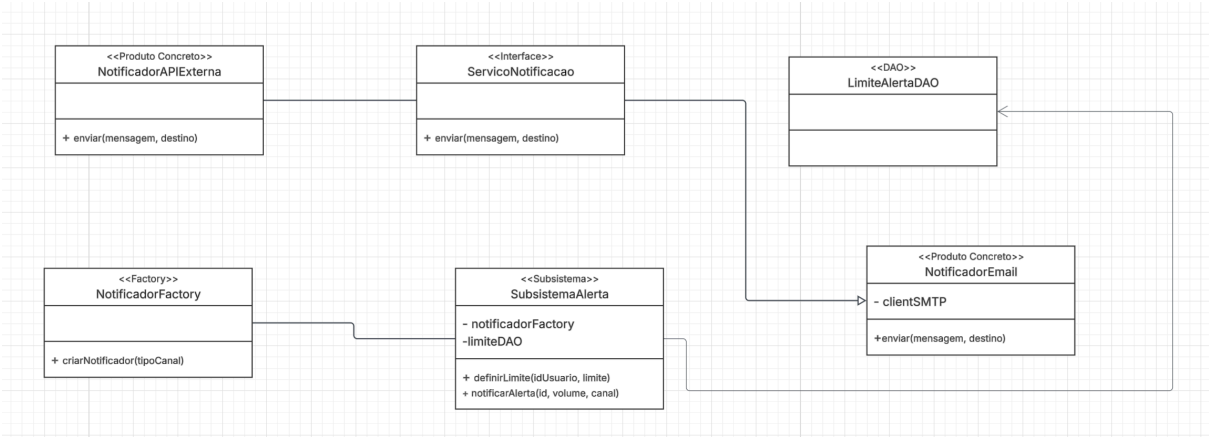
5. Detalhamento do Subsistema de Alerta (Padrões Factory Method e Adapter)

O `SubsistemaAlerta` garante o envio de notificações de consumo excessivo para múltiplos destinos (CAGEPA, Usuário, Sistemas Externos)21.

5.1. Funções e Justificativa da Factory

O **Padrão Factory Method** é essencial para a criação dos objetos de notificação (`NotificadorEmail` , `NotificadorAPIExterna`). A **`NotificadorFactory`** encapsula a complexidade de instanciar o canal correto, permitindo que o sistema adicione novos canais (como SMS) sem modificar o código do `SubsistemaAlerta` .

5.2. Diagrama


<<Produto Concreto>>
NotificadorAPIExterna
+ enviar(mensagem, destino)
<<Interface>>
ServicoNotificacao
+ enviar(mensagem, destino)
<<DAO>>
LimiteAlertaDAO
<<Produto Concreto>>
NotificadorEmail
- clientSMTP
+enviar(mensagem, destino)
<<Factory>>
NotificadorFactory
+ criarNotificador(tipoCanal)
<<Subsistema>>
SubsistemaAlerta
- notificadorFactory
-limiteDAO
+ definirLimite(idUsuario, limite)
+ notificarAlerta(id, volume, canal)