

Especificação de Requisitos

Fachada do Sistema de Monitoramento de Hidrômetros (SSMH)

Versão Final

1. Introdução

1.1. Propósito

Este documento especifica os requisitos para uma **Fachada (Facade)** de um sistema de software. O objetivo é criar um ponto de entrada único e simplificado para um "Painel de Monitoramento" de hidrômetros, atendendo às necessidades de uma concessionária de água fictícia (ex: CAGEPA).

1.2. Escopo (Os 3 Requisitos Mínimos)

O escopo deste projeto está focado em **três funcionalidades principais** que a Fachada deve fornecer:

Gestão de Usuários: Permitir o CRUD (Criar, Ler, Atualizar, Deletar) de usuários e suas contas de água.

Monitoramento: Permitir a consulta de consumo de água por intervalo de tempo, seja de um hidrômetro (SHA) ou de um usuário (agregando seus SHAs).

Alertas: Enviar um alerta quando o consumo de um usuário ultrapassar um limite pré-definido.

2. Visão Geral da Arquitetura

A Fachada simplifica a comunicação. O "Cliente" (que pode ser um Painel Gráfico (GUI) ou uma Linha de Comando (CLI)) só precisa conversar com a Fachada, sem se preocupar com a complexidade dos sistemas internos.

Diagrama de Arquitetura: [Cliente (GUI / CLI)] -> [FACHADA] -> [Subsistemas]

Para funcionar, a Fachada irá coordenar três subsistemas de backend:

1. **Subsistema de Usuários:** Responsável por guardar os dados cadastrais, faturas e quais SHAs pertencem a quem.

2. **Subsistema de Monitoramento:** Responsável por receber e armazenar as leituras de consumo dos hidrômetros.
 3. **Subsistema de Alertas:** Responsável por verificar as regras de consumo e disparar as notificações.
-

3. Especificação da Fachada

A Fachada deve expor os seguintes conjuntos de funcionalidades para o Cliente (Painel).

3.1. Funcionalidade 1: Gestão de Usuários e Contas (Padrão Command)

(Requisito Atendido: "Gerir, fazer um CRUD, de Usuários e suas respectivas contas de água.")

Métodos Expostos pela Fachada:

Nome do Método	Descrição
<code>executarComandoUsuario(comando)</code>	Novo método. A Fachada recebe um objeto <code>UserCommand</code> (ex: <code>CriarUsuarioCommand</code> , <code>DeletarUsuarioCommand</code>) e o executa. Isso substitui o CRUD direto e permite o recurso de "Desfazer".

3.2. Funcionalidade 2: Monitoramento de Consumo (Padrão Composite)

(Requisito Atendido: "Monitorar por intervalos de tempo o consumo de água individualizado de um hidrômetro (SHA) ou de um Usuário...")

Métodos Expostos pela Fachada:

Função	Descrição
--------	-----------

monitorarConsumo(monitoravel, data_inicio, data_fim)	<p>Novo método unificado. Recebe um objeto que implementa a interface ConsumoMonitoravel (que pode ser um ConsumoHidrometro [Leaf] ou um ConsumoUsuario [Composite]) e calcula o consumo de forma transparente.</p>
--	--

3.3. Funcionalidade 3: Sistema de Alertas (Padrão Strategy)

(Requisito Atendido: "Alertar quando um usuário ultrapassar um determinado volume de consumo.")

Mudança Principal: Os métodos da fachada deixam de aceitar apenas um número fixo (*limite_litros_dia*) e passam a aceitar um **tipo de estratégia** e seus parâmetros. Isso permite que, no futuro, sejam criadas regras como "Detecção de Vazamento" sem mudar a assinatura do método da Fachada.

Métodos Expostos pela Fachada:

Nome do Método	Descrição
configurarRegraDeAlerta(id_usuario, tipo_estrategia, valor_parametro)	<p>(Modificado) Define qual estratégia de análise será usada para o usuário.</p> <p>Parâmetros:</p> <p>tipo_estrategia: Define o algoritmo (ex: "LIMITE_DIARIO", "MEDIA_MOVEL", "DETECCAO_VAZAMENTO").</p> <p>valor_parametro: O valor de ajuste (ex: "70" para limite, ou "24h" para vazamento).</p>

listarAlertasAtivos()	Retorna uma lista de alertas que foram disparados (para o painel da CAGEPA).
configurarCanalNotificacao(tipo_notificacao)	<p>(Novo) Define a estratégia de envio de mensagens para o sistema como um todo.</p> <p>Exemplo: O cliente pode alternar de "WINDOWS_POPUP" para "CONSOLE_LOG" ou "EMAIL" em tempo de execução.</p>

4. Especificação dos Subsistemas

A Fachada não faz o trabalho sozinha; ela delega as tarefas para estes subsistemas.

4.1. Subsistema 1: Gestão de Usuários (Padrão Strategy e Command)

Responsabilidade: Gerenciar as entidades de negócio (Usuários, Faturas, Vínculos), atuando como uma camada de abstração sobre os dados. **Delega a persistência real** para uma estratégia de armazenamento configurável.

Padrão de Projeto Aplicado:

- **Padrão de Projeto Aplicado:** Strategy (para persistência) e Command (para execução de ações).

O que faz:

1. Atua como Receptor do Padrão Command. Ele recebe o comando e realiza a lógica de negócio, delegando a persistência à **ArmazenamentoStrategy**.

Definição das Estratégias (Novas Classes/Interfaces):

- **Interface (Contrato):** ArmazenamentoStrategy
 - Métodos: salvar(entidade), buscar(id), atualizar(id, dados), deletar(id).
- **Estratégia Concreta A:** ArmazenamentoSqlite (Conecta via JDBC/ORM a um banco Sqlite).
- **Estratégia Concreta B:** ArmazenamentoVolatil (Usa HashMaps em memória RAM para testes unitários ultra-rápidos).

Métodos Internos (Atualizado):

Método da Estratégia (ArmazenamentoStrategy)	Usado por (Método da Fachada)
storage_salvarUsuario(dados)	fachada.criarUsuario()
storage_buscarUsuarioPorId(id)	fachada.buscarUsuario()
storage_atualizarUsuario(id, dados)	fachada.atualizarUsuario()
storage_removerUsuario(id)	fachada.deletarUsuario()
storage_associarHidrometro(id_user, id_sha)	fachada.vincularHidrometro()
storage_listarHidrometros(id_user)	fachada.monitorarConsumoPorUsuario()
storage_buscarFaturas(id_user)	fachada.listarContasDeAgua()
storage_definirImplementacao(tipo)	Inicialização do Sistema (Define se vai rodar em modo "Teste/Memória" ou "Produção/SQL").

4.2. Subsistema 2: Coleta e Monitoramento (Padrões Adapter e Composite)

- **Responsabilidade:** É o "historiador" de consumo (dados de série temporal).
- **Padrão de Projeto Aplicado:** Adapter (para OCR) e Composite (para agregação de consumo).
- **O que faz:**
 - Usa o Adapter (**ProcessadorOCR**) para traduzir imagens em números.
 - É responsável por construir a estrutura Composite de consumo (**ConsumoUsuario** contendo **ConsumoHidrometro**).

- **Exemplo de Tecnologia:** Um banco de dados relacional simples ou arquivo de configurações.
- **Métodos Internos (usados pela Fachada):**

Função no Subsistema TSDB	Descrição de Uso
<code>tsdb_consultarConsumo(id_sha, inicio, fim)</code>	Usado por <code>fachada.monitorarConsumoPorHidrometro()</code> .
<code>tsdb_consultarConsumoAgregado(lista_de_shas, inicio, fim)</code>	Usado por <code>fachada.monitorarConsumoPorUsuario()</code> (para consultar vários SHAs de uma vez, de forma eficiente).
<code>tsdb_calcularConsumoRecente(id_sha, periodo_horas)</code>	Usado internamente pelo Subsistema de Alertas para verificar as regras.

4.3. Subsistema 3: Alertas e Notificações (Padrões Strategy e Observer)

- **Responsabilidade:** É o "vigia" do sistema. Sua responsabilidade é monitorar violações de regras, mas delega a responsabilidade de como notificar para estratégias intercambiáveis.
- **Padrão de Projeto Aplicado:**
 - **Strategy** (para Notificação) e **Observer** (para Notificação de Eventos).
- **O que faz: Atua como o Sujeito (Subject) no Padrão Observer.**
 - Quando uma regra é violada, o `AlertaService` aciona o método `notificarObservadores()`.
 - Isso garante que o **Painel**, o **Logger** e a **Estratégia de Notificação** sejam atualizados automaticamente.

Definição das Estratégias (Novas Classes/Interfaces):

- Interface (Contrato): `NotificacaoStrategy`
 - Método: `enviar(mensagem, destinatario)`
- Estratégia Concreta A: `NotificacaoWindowsPopup` (Implementação original sugerida ⁶).
- Estratégia Concreta B: `NotificacaoConsoleLog` (Para debug e testes rápidos).
- Estratégia Concreta C: `NotificacaoEmail` (Para uso futuro em produção).

Métodos Internos (Atualizado):

Método Interno	Usado por	Descrição

alert_salvarRegra(id_usuario, limite)	fachada.criarRegraDeAlerta()	Define o limite de consumo. ⁷
alert_buscarAlertasAtivos()	fachada.listarAlertasAtivos()	Retorna alertas disparados. ⁸
alert_definirEstrategia(tipo)	Configuração do Sistema	Novo: Permite trocar dinamicamente o comportamento de envio (ex: mudar de 'Popup' para 'Email' em tempo de execução).

5. Infraestrutura e Persistência de Dados

5.1. Estratégia de Armazenamento

Para atender à necessidade de persistência dos dados dos subsistemas definidos anteriormente, o sistema utilizará um Banco de Dados Relacional Embarcado (SQLite).

- **Justificativa:** O SQLite permite manter a simplicidade de um "arquivo único" (facilitando backups e transporte do projeto), mas oferece a robustez da linguagem SQL para consultas complexas de séries temporais e relacionamentos entre usuários e hidrômetros, superando a limitação de arquivos de texto simples.

5.2. Rastreabilidade e Logs (Padrão Singleton)

Foi adicionado um componente transversal de Logging para facilitar o debug e monitorar a saúde da aplicação.

- **Padrão de Projeto Aplicado:** Singleton.
- **Funcionamento:** A classe **LogManager** deve ser implementada como um **Singleton** para garantir que apenas uma instância exista em todo o sistema. Todos os subsistemas e a Fachada acessarão esta única instância para registrar seus logs no arquivo **system.log**.
- **Estrutura do Log:** [DATA_HORA] [NIVEL: INFO/ERRO] [MÉTODO] Mensagem.
 - *Exemplo:* [2025-11-18 10:00:00] [INFO] [criarUsuario] Usuário 'Joao' criado com ID 45.
- **Armazenamento:** Arquivo de texto local (system.log).

⁷ https://www.sqlite.org/limits.html

⁸ https://www.sqlite.org/funcionamento.html

6. Controle de Acesso e Notificações Avançadas

6.1. Perfis de Acesso (Admin vs. Comum)

O Subsistema de Usuários será expandido para suportar perfis, garantindo que apenas administradores possam realizar operações sensíveis.

- **Atributo Novo:** tipo_perfil (ENUM: 'ADMIN', 'LEITOR').
- **Regra na Fachada:** O método `deletarUsuario` agora verificará se o usuário solicitante possui o perfil 'ADMIN' antes de delegar a tarefa ao subsistema.

6.2. Envio de E-mails

O Subsistema de Alertas, além de gerar notificações visuais (Popups), integrará uma biblioteca de envio de e-mails (ex: smtplib em Python).

- **Fluxo:** Quando o limite de 70L/dia for ultrapassado, o sistema disparará um e-mail para o endereço cadastrado do usuário contendo o alerta.

7. Tratamento de Restrições do Hardware (SHA)

7.1. Isolamento do Painel (Padrão Facade)

Em conformidade com a arquitetura proposta, o Painel (Cliente) é estritamente proibido de acessar diretamente os SHAs.

- **Garantia:** O Painel conhece apenas a classe Fachada. Toda a comunicação com o hardware é encapsulada pelo "Subsistema de Monitoramento". O Painel solicita "quants litros foram gastos", e a Fachada busca essa informação já processada no banco de dados, jamais conectando-se diretamente ao dispositivo físico.

7.2. Processamento de Imagens dos SHAs

A leitura do consumo não será enviada como um dado numérico direto pelo hidrômetro. O SHA envia uma imagem do mostrador.

- **Alteração no Subsistema de Monitoramento:**
 1. **Entrada:** O subsistema recebe um arquivo de imagem (ex: `leitura_sha123.jpg`).
 2. **Processamento (OCR):** O subsistema implementará uma rotina de *Reconhecimento Óptico de Caracteres* (OCR) para extrair os números da imagem.
 3. **Persistência:** Apenas o valor numérico extraído e o caminho da imagem original são

salvos no banco de dados.

4. **Consulta:** Quando o método `monitorarConsumoPorHidrometro` for chamado pela Fachada, ele retornará o dado numérico já processado, tornando transparente para o Painel a complexidade de tratar imagens.

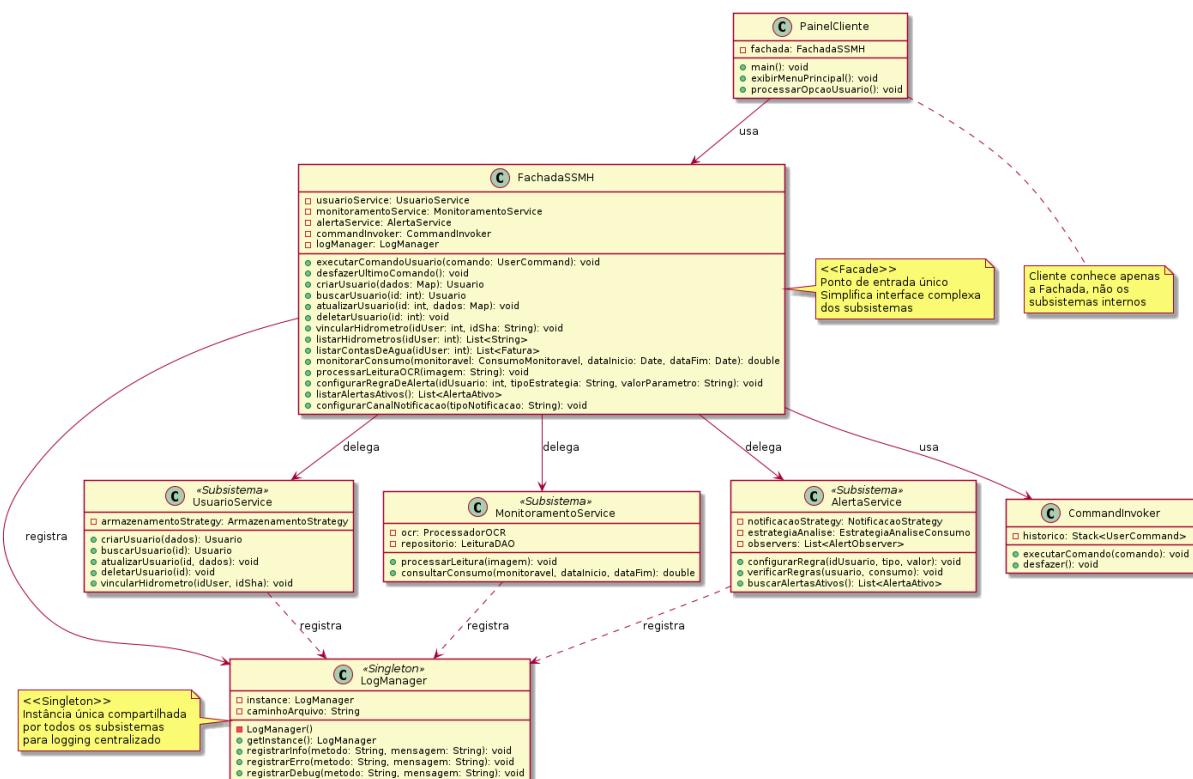
8. Modelagem e Diagramas de Classes

Esta seção apresenta a estrutura de classes para orientar o desenvolvimento, detalhando a aplicação dos Padrões de Projeto.

8.1. Visão Geral (Padrões Façade e Singleton)

Este diagrama ilustra como a classe **FachadaSSMH** (Façade) isola a complexidade dos subsistemas para o cliente (**PainelCliente**). Adicionalmente, demonstra a utilização do padrão **Singleton** (**LogManager**), garantindo que todos os subsistemas compartilhem uma única instância para o registro de eventos.

Diagrama de Classes: Visão Geral (Façade e Singleton)

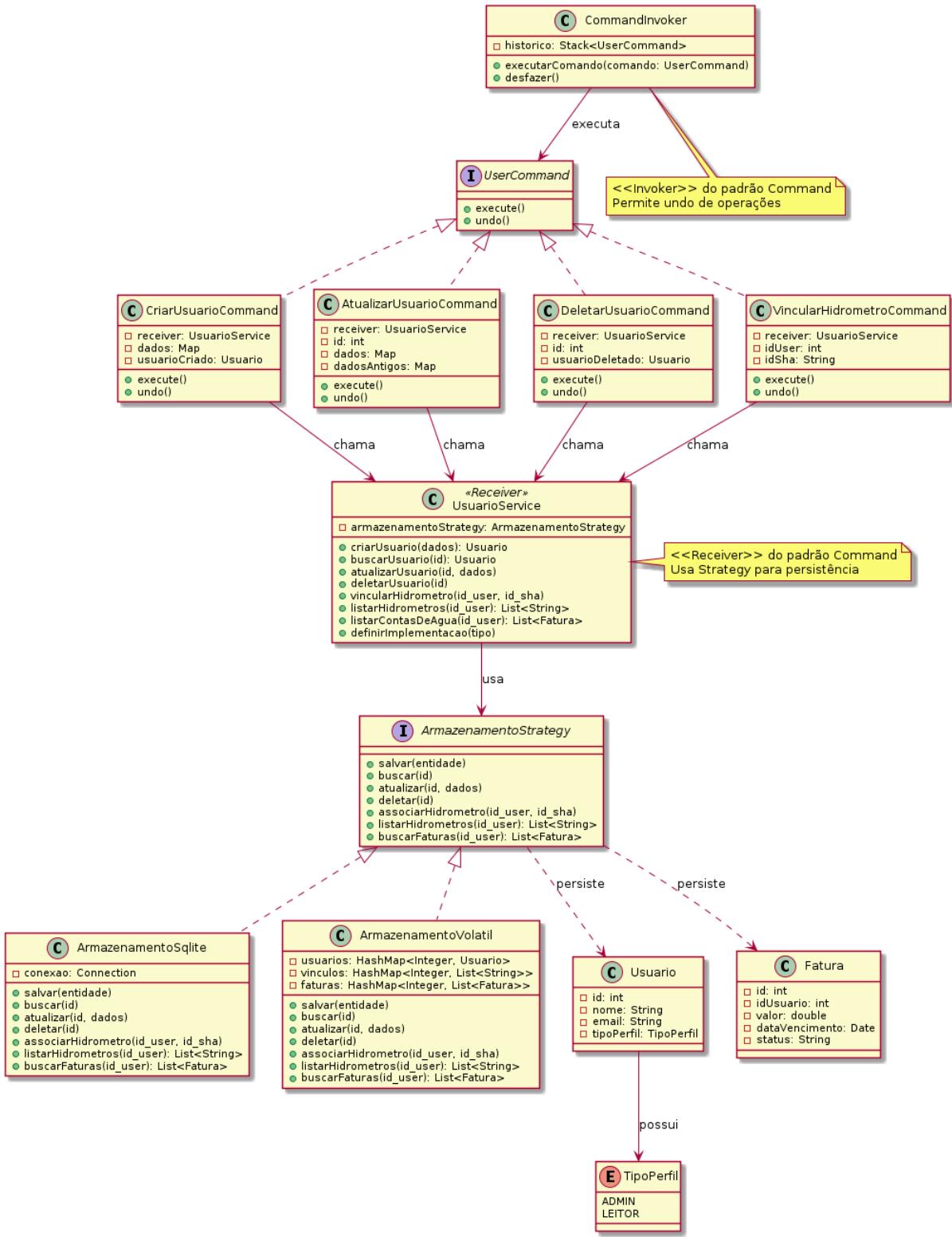


8.2. Detalhamento dos Subsistemas

A. Subsistema de Usuários (Padrões Strategy e Command)

Este subsistema utiliza o padrão **Strategy** para o acesso ao banco de dados (DAO/Repository) e o padrão **Command** para encapsular as requisições de operações de usuário, permitindo o suporte a histórico, *undo* ou filas de operações.

Diagrama de Classes: Subsistema de Usuários (Strategy e Command)

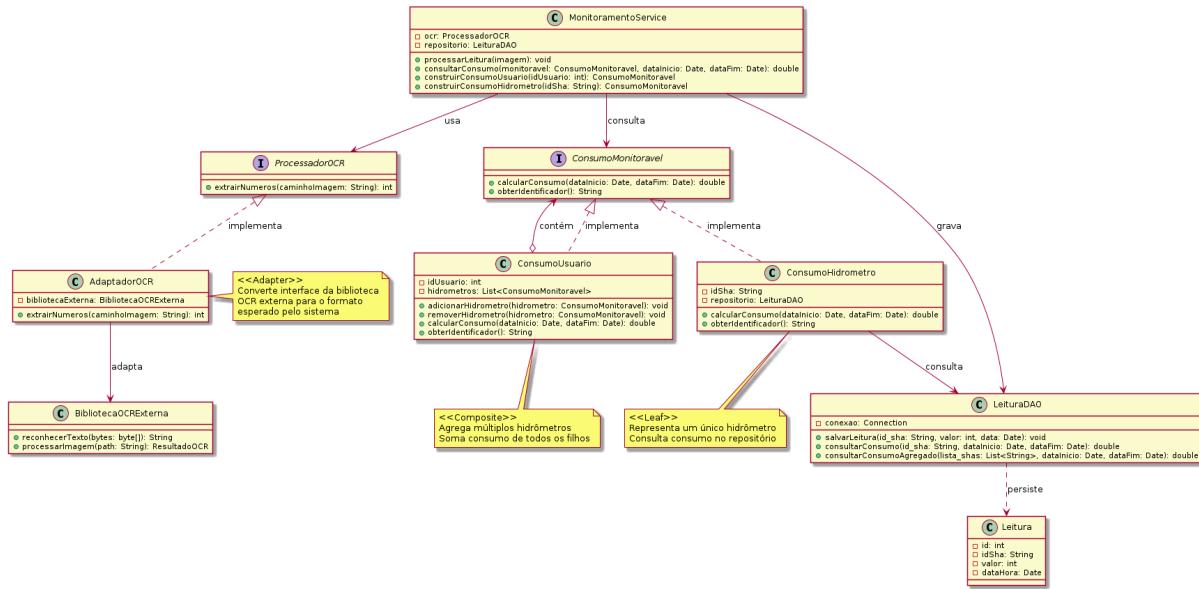


B. Subsistema de Monitoramento (Padrões Adapter e Composite)

O subsistema utiliza o padrão **Adapter** para padronizar a leitura de diferentes fontes (como o OCR da imagem do SHA) em uma interface unificada ([LeituraDigitalizada](#)). O

padrão **Composite** é usado para estruturar o consumo, tratando tanto um hidrômetro individual quanto o consumo total de um usuário (que pode ter múltiplos hidrômetros) de maneira uniforme.

Diagrama de Classes: Subsistema de Monitoramento (Adapter e Composite)



C. Subsistema de Alertas (Padrões Strategy e Observer)

Este subsistema implementa o padrão **Observer** para notificar múltiplos componentes (Interface de Usuário, Logs) sobre a ocorrência de um alerta. O padrão **Strategy** é mantido, mas é integrado ao Observer através do **EstrategiaNotificacaoAdapter**, que media as notificações para diferentes canais (Email, SMS, etc.).

Diagrama de Classes: Subsistema de Alertas (Strategy e Observer)

