**Especificação Técnica do Painel de Monitoramento de Hidrômetros (SMH)**

Este documento define a arquitetura do sistema SMH, detalhando a API pública, os requisitos funcionais e os subsistemas responsáveis pela execução das tarefas. A arquitetura segue o padrão de projeto **Facade** para centralizar a complexidade e desacoplar o painel frontal dos serviços de backend.

## 1. Definição da Interface Pública (Padrão Facade)

A interface abaixo (ISMH_Facade) representa o contrato único que o Painel de Monitoramento utiliza para acessar todas as funcionalidades do sistema. Sua implementação (MonitoramentoFacade) delega as chamadas para os subsistemas especializados.

### 1.1. RF-001: Gestão de Usuários e Contas (CRUD)

**Objetivo:** Permitir que o Painel gerencie usuários e seus hidrômetros (SHAs).

```
import java.util.List;
import java.util.Date;

// Interface principal do Padrão Facade
public interface ISMH_Facade {
    //
==========================================================================
    // GESTÃO DE USUÁRIOS E CONTAS
    //
==========================================================================
    /**
     * Cria um novo usuário (cliente) no sistema.
     * DELEGA PARA: SubsistemaDeUsuarios
     */
    UsuarioDTO criarUsuario(NovoUsuarioRequest request);
    /**
     * Busca os dados de um usuário pelo seu ID.
     * DELEGA PARA: SubsistemaDeUsuarios
     */
    UsuarioDTO getUsuario(String usuarioId);
    /**
     * Atualiza dados cadastrais de um usuário.
     * DELEGA PARA: SubsistemaDeUsuarios
     */
```

```
    UsuarioDTO atualizarUsuario(String usuarioId, AtualizaUsuarioRequest request);
    /**
     * Associa um novo hidrômetro (SHA) a um usuário existente.
     * DELEGA PARA: SubsistemaDeUsuarios (para fazer a associação)
     */
    SHA_DTO vincularShaAoUsuario(String usuarioId, String shaId, String endereco);

    /**
     * Lista todos os SHAs vinculados a um usuário.
     * DELEGA PARA: SubsistemaDeUsuarios
     */
    List<SHA_DTO> listarShasDoUsuario(String usuarioId);

    // Fim da seção RF-001 (Interface continua...)
```

### 1.2. RF-002: Monitoramento de Consumo

**Objetivo:** Fornecer dados de consumo, seja de um único sensor ou de forma agregada por usuário, para o Painel de Monitoramento.

```
    //
================================================================================
    // RF-002: MONITORAMENTO DE CONSUMO
    //
================================================================================

    /**
     * Retorna o histórico de consumo de um único hidrômetro (SHA)
     * em um intervalo de tempo.
     * DELEGA PARA: SubsistemaDeSensores (ou Subsistema de Processamento de Imagens)
     */
    HistoricoConsumoDTO getConsumoPorSha(String shaId, Date inicio, Date fim);

    /**
     * Retorna o consumo AGREGADO (soma) de TODOS os SHAs de um usuário
     * em um intervalo de tempo.
     * * ORQUESTRAÇÃO DA FACHADA:
     * 1. Chama SubsistemaDeUsuarios.listarShasDoUsuario(usuarioId)
```

```
     * 2. Para cada SHA, chama SubsistemaDeSensores.getConsumoPorSha()
     * 3. Agrega (soma) os resultados e os retorna.
     */
    HistoricoConsumoDTO getConsumoAgregadoPorUsuario(String usuarioId, Date inicio, Date
fim);

    /**
     * Retorna a leitura mais recente (tempo real) de um usuário,
     * usado pelo Painel para o alerta >= 70 litros/dia.
     * DELEGA PARA: SubsistemaDeSensores (com otimização)
     */
    ConsumoAtualDTO getConsumoAtualUsuario(String usuarioId);

    // Fim da seção RF-002 (Interface continua...)
```

### 1.3. RF-003: Sistema de Alertas

**Objetivo:** Permitir a configuração de regras de alerta e o disparo de notificações para os sistemas externos ou para o próprio usuário.

```
    //
==========================================================================
    // RF-003: SISTEMA DE ALERTAS
    //
==========================================================================

    /**
     * Cria ou atualiza uma regra de alerta para um usuário.
     * Ex: {usuarioId: "123", limiteLitros: 70, periodoHoras: 24,
     * canaisStrategy: ["EMAIL", "WEBHOOK"], destinatarios: ["admin@cagepa.com"]}
     * DELEGA PARA: SubsistemaDeAlertas (Motor de Regras)
     */
    RegraAlertaDTO configurarRegraAlerta(ConfigAlertaRequest request);
    /**
     * Retorna os alertas ativos no momento (ex: quais usuários estão
     * acima do limite). Usado pelo Painel CAGEPA.
     * DELEGA PARA: SubsistemaDeAlertas
     */
```

```
    List<AlertaAtivoDTO> getAlertasAtivos();

    /**
     * Método a ser chamado quando um alerta é disparado, para enviar a notificação.
     * DELEGA PARA: SubsistemaDeNotificacao
     */
    void dispararNotificacao(Alerta alerta);

} // Fim da interface ISMH_Facade
```

---

**2. Especificação dos Subsistemas Requeridos**

A arquitetura do sistema depende de um conjunto de subsistemas coesos e especializados. O cliente (Painel) não conhecerá esses subsistemas diretamente, interagindo apenas através da Fachada definida acima.

**1. Subsistema de Usuários e Contas**

- **Responsabilidade:** Gerenciar o cadastro de clientes (CRUD de Usuários) e o mapeamento Usuário -> [SHA_1, SHA_2, ...].
- **Interface Interna:** getUsuario(id), criarUsuario(), getShasDoUsuario(id).
- **Implementação no Projeto:** Repositório em memória (HashMap) ou Banco H2.

**2. Subsistema de Sensores / Processamento de Imagens**

- **Responsabilidade:** Monitorar pastas locais para detectar novos arquivos de imagem, processá-los via OCR (Tesseract) e armazenar as leituras. Deve suportar diferentes tipos de entrada (Arquivo Único via Adapter ou Pastas de Histórico).
- **Interface Interna:** getLeiturasPorSha(shaId, periodo), lerImagem(caminho).
- **Implementação no Projeto:** SensorService integrado com biblioteca Tess4J e padrão Adapter.

**3. Subsistema de Alertas (Motor de Regras)**

- **Responsabilidade:** Armazenar as regras de negócio (ex: "limite de 10m³"). Deve observar as novas leituras e comparar o consumo atual com as regras configuradas para identificar violações.
- **Interface Interna:** configurarRegra(), verificarLimites(consumo).
- **Implementação no Projeto:** AlertaService atuando como Observer dos sensores.

**4. Subsistema de Notificação**

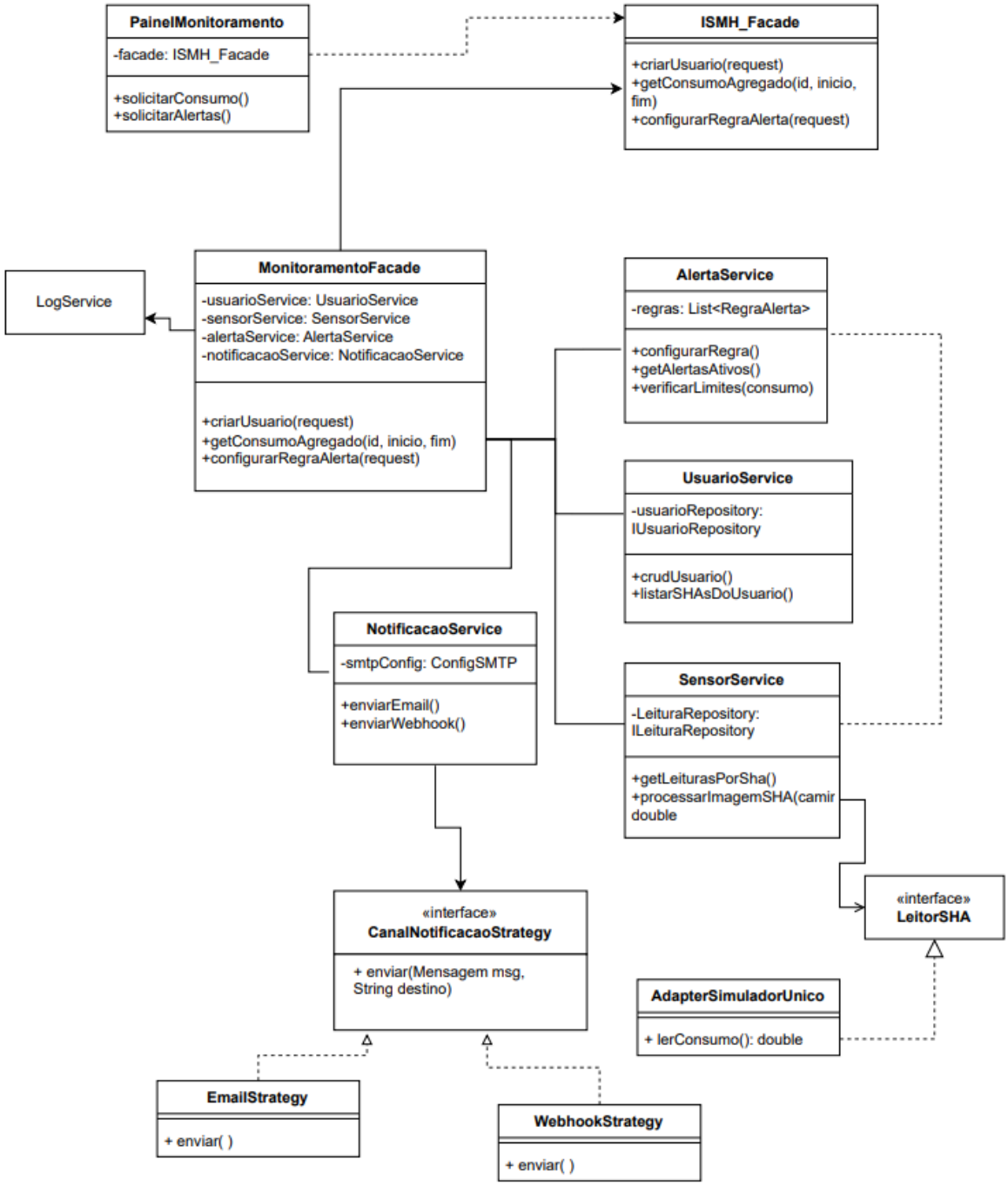
- **Responsabilidade:** Enviar notificações para diferentes canais quando instruído pelo sistema de alertas.
- **Interface Interna:** enviarNotificacao(mensagem, destinatario).
- **Padrão Utilizado: Strategy**. Utiliza a interface CanalNotificacaoStrategy para permitir a troca dinâmica entre envio por E-mail (Simulado) ou Webhook.

**5. Subsistema de Log e Auditoria**

- **Responsabilidade:** Registrar transversalmente todas as operações críticas (alerta disparado, falha de leitura OCR, criação de usuário) para facilitar o debug e auditoria do sistema.
- **Implementação no Projeto:** Mecanismo de logging integrado aos serviços (LogService ou SLF4J).

**Diagrama de Caso de Uso:**


**Diagrama de Classes:**


PainelMonitoramento
-facade: ISMH_Facade
+solicitarConsumo()
+solicitarAlertas()
ISMH_Facade
+criarUsuario(request)
+getConsumoAgregado(id, inicio, fim)
+configurarRegraAlerta(request)
LogService
MonitoramentoFacade
-usuarioService: UsuarioService
-sensorService: SensorService
-alertaService: AlertaService
-notificacaoService: NotificacaoService
+criarUsuario(request)
+getConsumoAgregado(id, inicio, fim)
+configurarRegraAlerta(request)
AlertaService
-regras: List<RegraAlerta>
+configurarRegra()
+getAlertasAtivos()
+verificarLimites(consumo)
UsuarioService
-usuarioRepository: IUsuarioRepository
+crudUsuario()
+listarSHAsDoUsuario()
NotificacaoService
-smtpConfig: ConfigSMTP
+enviarEmail()
+enviarWebhook()
SensorService
-LeituraRepository: ILeituraRepository
+getLeiturasPorSha()
+processarImagemSHA(camir double
«interface»
CanalNotificacaoStrategy
+ enviar(Mensagem msg, String destino)
«interface»
LeitorSHA
AdapterSimuladorUnico
+ lerConsumo(): double
EmailStrategy
+ enviar( )
WebhookStrategy
+ enviar( )