Capitulo 6. Diseño de arquitectura y patrones

1. Estrategia de diseño de software

1.1. Estrategia seleccionada

Para el desarrollo del Sistema de Gestión de Reclamos de Agua Potable de SEDAM, se adopta la estrategia de diseño orientada a objetos. Esta estrategia permite modelar el sistema a partir de entidades del mundo real, representadas como clases con atributos, métodos y relaciones claramente definidas.

1.2. Justificación de la estrategia

La estrategia orientada a objetos es la más adecuada para este proyecto por las siguientes razones:

a) El sistema trabaja con entidades reales del proceso de SEDAM

El dominio del problema contiene objetos naturales como:
Ciudadano, Operador SEDAM, Reclamo, Notificación, Evidencia, Reporte, etc. Cada uno posee atributos propios y métodos asociados a su comportamiento. La POO facilita la representación de estas entidades de forma clara y fiel.

b) Permite la especialización / herencia

La estrategia OO soporta generalización/especialización, por lo que se implementó:

- USUARIO (clase padre)
- CIUDADANO (hijo)
- OPERADOR_SEDAM (hijo)

Esto permite representar adecuadamente a los dos actores principales del sistema, cada uno con atributos y comportamientos propios. Esto cumple directamente profundizar el ER y diferenciar claramente los tipos de usuario.

c) Facilita la extensibilidad del sistema

Las clases pueden evolucionar mediante:

- Herencia (extender funcionalidad)
- Composición (objetos que contienen otros objetos)
- Asociaciones (relaciones directas entre entidades)

Esto permite que el sistema crezca sin romper lo ya construido.

d) Soporta adecuadamente procesos con múltiples relaciones

El sistema requiere manejar relaciones como:

- Un ciudadano registra muchos reclamos
- Un reclamo tiene muchos cambios de estado
- Un operador gestiona múltiples reclamos
- Un reclamo puede generar múltiples notificaciones
- Un reporte está ligado a un formato oficial

La POO modela estas relaciones mediante asociación, agregación o composición según corresponda.

e) Adecuación de la estrategia orientada a objetos al tipo de sistema

La estrategia orientada a objetos se ajusta adecuadamente al sistema de gestión de reclamos debido a que permite representar de manera natural las entidades involucradas como ciudadanos, operadores, reclamos, estados y notificaciones mediante clases que encapsulan atributos y comportamientos. Esto facilita la organización del software en componentes claros y coherentes.

Asimismo, el sistema requiere evolucionar con el tiempo, incorporando nuevos tipos de reclamos, estados del proceso o formatos oficiales. La orientación a objetos proporciona los mecanismos necesarios para esta extensibilidad, como la herencia, la composición y la asociación, permitiendo ampliar funcionalidades sin afectar la estructura existente.

Finalmente, la orientación a objetos favorece la reutilización de componentes por ejemplo, mediante la clase base USUARIO y sus especializaciones, optimizando el desarrollo y manteniendo la consistencia del sistema.

f) Contribución de la estrategia orientada a objetos al cumplimiento de los principios de calidad del diseño

La orientación a objetos permite cumplir de manera efectiva los principios de calidad del diseño. Favorece la modularidad, ya que el sistema se organiza en clases y componentes independientes; mejora la cohesión, asignando responsabilidades claras a cada clase; y reduce el acoplamiento, estableciendo relaciones controladas entre los módulos.

Además, aplica abstracción y encapsulamiento, protegiendo los datos y ocultando detalles internos de implementación. Finalmente, facilita la reutilización del código mediante herencia y composición, permitiendo extender funcionalidades sin reconstruir el sistema.

2. Tipo de arquitectura del sistema

2.1. Arquitectura seleccionada

Para el desarrollo del Sistema de Gestión de Reclamos de Agua Potable para la EPS SEDAM se adopta la Arquitectura en Capas (N-Tier). Este patrón organiza el sistema en niveles con responsabilidades claramente definidas, permitiendo una mayor mantenibilidad, escalabilidad y separación lógica entre la presentación, la lógica de negocio y el acceso a datos.

2.2. Justificación de la elección de arquitectura

a) **Escalabilidad**

La separación por capas permite distribuir la carga en distintos servidores según la necesidad, facilitando el crecimiento gradual del sistema ante un aumento de reclamos ciudadanos.

b) **Mantenibilidad**

Cada capa puede modificarse sin afectar a las demás. Esto facilita:

- corregir errores
- añadir nuevas funcionalidades
- actualizar las reglas de negocio institucionales
- adaptarse a cambios normativos de SEDAM

c) **Seguridad**

Aislar presentación, lógica y datos permite aplicar controles como:

- autenticación y autorización
- protección de datos sensibles
- filtros contra inyecciones
- validaciones internas estructuradas

d) **Cohesión y bajo acoplamiento**

El diseño por capas mantiene un código más organizado y coherente con la estrategia orientada a objetos que estructura las entidades del dominio y sus interacciones.

e) Adecuación al contexto institucional

SEDAM es una única EPS con un sistema centralizado. Por ello, una arquitectura más compleja (microservicios, hexagonal o event-driven) sería innecesaria y costosa. La arquitectura en capas proporciona el balance adecuado entre simplicidad, orden, robustez y claridad técnica, alineándose con los objetivos del sistema y del ODS 6.

La Arquitectura en Capas (N-Tier) ofrece una estructura sólida y escalable para el Sistema de Gestión de Reclamos de Agua Potable de la EPS SEDAM. Se adapta adecuadamente al contexto institucional, garantiza

mantenibilidad, soporte a largo plazo y seguridad, y es plenamente compatible con el diseño orientado a objetos adoptado en el proyecto.

2.3. Descripción de la Arquitectura

I. Capa de Presentación

Responsable de la interacción con los usuarios, tanto ciudadanos como operadores SEDAM. Incluye:

- Interfaces web responsivas.
- Formularios de registro de reclamos.
- Paneles de consulta y seguimiento.
- Pantallas de gestión para operadores.

Esta capa puede implementar el patrón MVC para separar vista, controlador y modelo, evitando mezclar lógica visual con reglas de negocio.

II. Capa de Lógica de Negocio

Contiene las reglas que rigen el funcionamiento interno del sistema:

- Validación y registro de reclamos.
- Generación del código de seguimiento.
- Gestión de estados y flujos internos.
- Registro del historial y trazabilidad.
- Emisión de notificaciones.
- Gestión de formatos oficiales de SEDAM.
- Generación de reportes institucionales.

Es la capa más importante, ya que implementa los procesos definidos en el diseño orientado a objetos (clases, relaciones y responsabilidades del dominio).

III. Capa de Acceso a Datos

Encargada de la comunicación con la base de datos relacional. Incluye:

- Repositorios / DAOs.
- Sentencias SQL optimizadas.
- Manejo de transacciones.
- Consultas y persistencia de entidades.

Esta capa abstrae el acceso físico a la información, protegiendo la lógica de negocio de modificaciones en la infraestructura de datos.

IV. Base de Datos

Modelo relacional normalizado (3FN) que almacena:

- Usuarios (ciudadanos y operadores)
- Reclamos
- Estados e historial
- Evidencias
- Notificaciones
- Reportes institucionales
- Formatos oficiales de SEDAM

Su diseño responde a la profundización realizada previamente.

2.3. Diagrama General de la Arquitectura

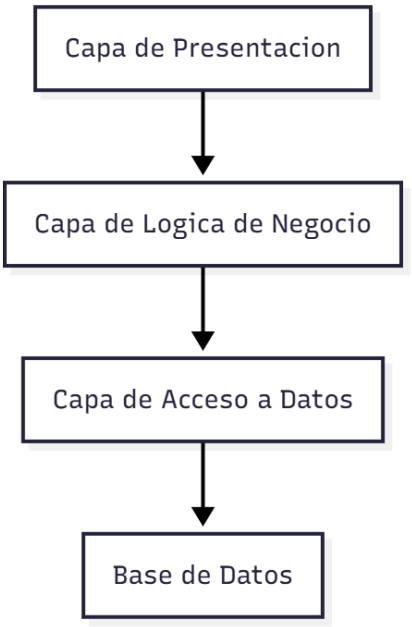

Figura 1. *En el diagrama se observa de manera general como estaría armado la arquitectura por n-capas*

3. Patrones de diseño aplicados

En el desarrollo del Sistema de Gestión de Reclamos para SEDAM, se aplicaron diversos patrones de diseño con el objetivo de mejorar la mantenibilidad, escalabilidad y claridad del sistema. A continuación, se describen los patrones seleccionados, su propósito y el aporte que brindan a la solución.

a) Patrón Observer (Observador)

Propósito
Permite que un objeto (Sujeto) notifique automáticamente a otros objetos interesados (Observadores) cuando ocurre un cambio de estado.

Aplicación en el sistema
En el módulo de reclamos, cada vez que el estado de un reclamo cambia (por ejemplo: *Registrado → En evaluación → Atendido*), el sistema debe enviar notificaciones automáticas al ciudadano o al operador correspondiente. El reclamo actúa como Sujeto, y los módulos de notificaciones como Observadores.

Aporte al sistema

- Garantiza una comunicación automática entre módulos.
- Reduce el acoplamiento entre “Reclamo” y “Notificación”.
- Facilita agregar nuevos medios de notificación (correo, web, SMS) sin modificar el código central.

b) Patrón DAO (Data Access Object)

Propósito
Encapsular la lógica de acceso a datos en clases especializadas que permiten separar la lógica de persistencia de la lógica de negocio.

Aplicación en el sistema
Las entidades principales del sistema (Usuario, Reclamo, Evidencia, Notificación, Historial, etc.) interactúan con la base de datos mediante objetos DAO. Por ejemplo, el objeto ReclamoDAO se encarga de registrar, actualizar, obtener y cerrar reclamos sin exponer detalles del motor de base de datos.

Aporte al sistema

- Facilita el mantenimiento y la reutilización del código.
- Permite cambiar o mejorar la base de datos sin afectar la lógica de negocio.
- Reduce el acoplamiento entre capas y fortalece la arquitectura en capas (N-Tier).

c) Patrón Fachada (Facade)

Propósito
Proporcionar una interfaz única y simplificada para un conjunto complejo de operaciones internas.

Aplicación en el sistema
El proceso de registrar un reclamo implica varios pasos:

1. Registrar los datos del reclamo.
2. Guardar evidencias.
3. Crear el historial inicial.
4. Notificar al ciudadano.

En lugar de que la interfaz gestione cada paso, se implementa una Fachada de Gestión de Reclamos que centraliza todas estas operaciones en un único punto de acceso.

Aporte al sistema

- Simplifica la interacción con procesos complejos.
- Reduce errores al estandarizar el flujo interno.
- Mejora la mantenibilidad al concentrar la lógica de coordinación en una sola clase.

4. Diseño estructural

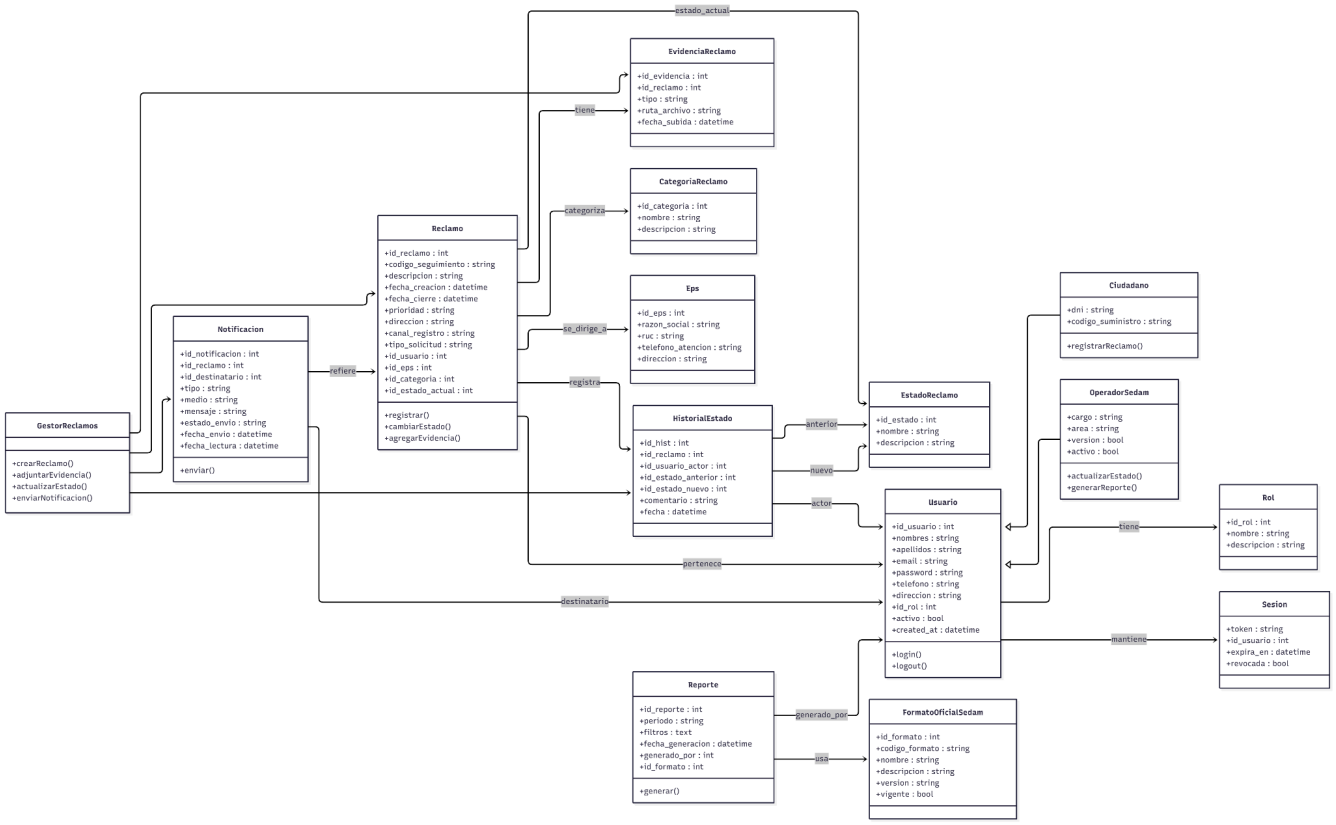

***Figura 2**. La imagen muestra el diseño estructural del proyecto*

https://www.mermaidchart.com/app/projects/989bc838-7a69-462b-aec0-859a513dd7c0/diagrams/12adfcaf-ea14-49e8-be74-327438341677/share/invite/eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJkb2N1bWVudElEIjoiMTJhZGZjYWYtZWExNC00OWU4LWJlNzQtMzI3NDM4MzQxNjc3IiwiYWNjZXNzIjoiVmlldyIsImlhdCI6MTc2MzMyOTUwOH0.KmMLJMtX9_weKGWsqBBsUUjjwtEB5HYat1t9hvEQclo