Sistema de Gestión Hospitalaria

Una aplicación orientada a objetos que modela la gestión integral de un hospital, implementando las operaciones fundamentales para administrar pacientes, personal médico, instalaciones y citas médicas con arquitectura robusta y escalable.

Agenda de la Presentación

01

Arquitectura del Sistema

Estructura jerárquica, herencia y gestión de relaciones

02

Funcionalidades Principales

Gestión de pacientes, citas médicas y validaciones

03

Persistencia y Recuperación

Serialización CSV y gestión de estado

04

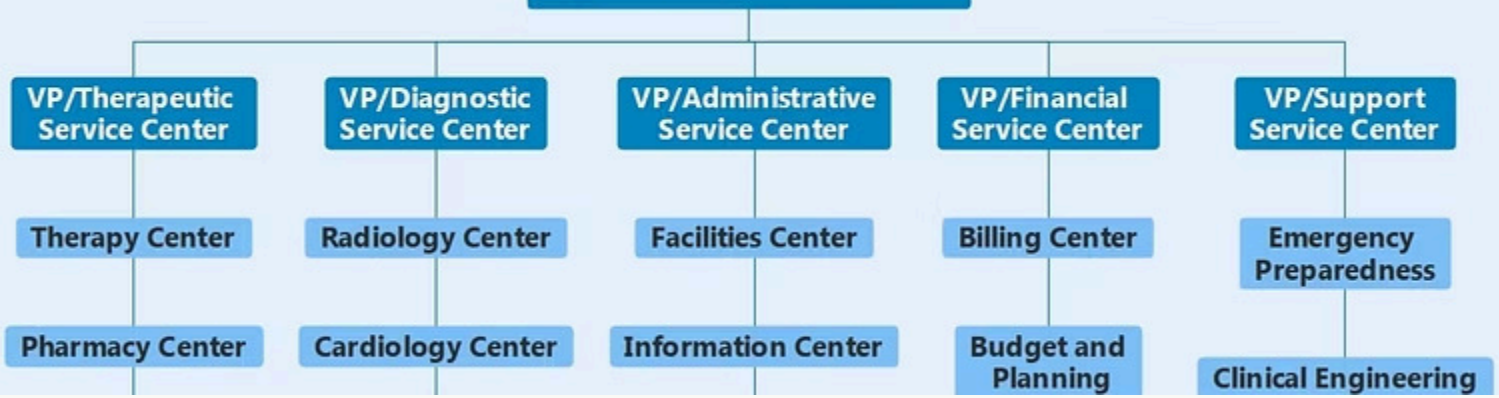
Aspectos de Diseño

Principios SOLID, patrones implementados y manejo de errores

05

Escalabilidad

Extensibilidad y análisis técnico detallado



Arquitectura del Sistema

Estructura Jerárquica del Sistema

Entidad Central

El sistema está organizado alrededor de una entidad central **Hospital** que contiene múltiples **Departamentos** especializados. Esta arquitectura permite una gestión centralizada pero flexible de todos los recursos hospitalarios.

Cada departamento agrupa médicos de la misma especialidad y gestiona sus propias salas médicas, manteniendo la coherencia funcional y operativa.

Herencia y Polimorfismo

1

Clase Abstracta Persona

Sirve como base para **Paciente** y **Medico**, compartiendo atributos comunes como datos personales, tipo de sangre y validaciones fundamentales.

2

Extensibilidad

Esta arquitectura permite extensibilidad para agregar otros tipos de personal hospitalario sin modificar el código base existente.

```
public abstract class Persona {
 // Algoritmo común de validación
 protected void validarString(String valor, String mensajeError)
 protected void validarDni(String dni)
 // Las subclases extienden con validaciones específicas
}
```

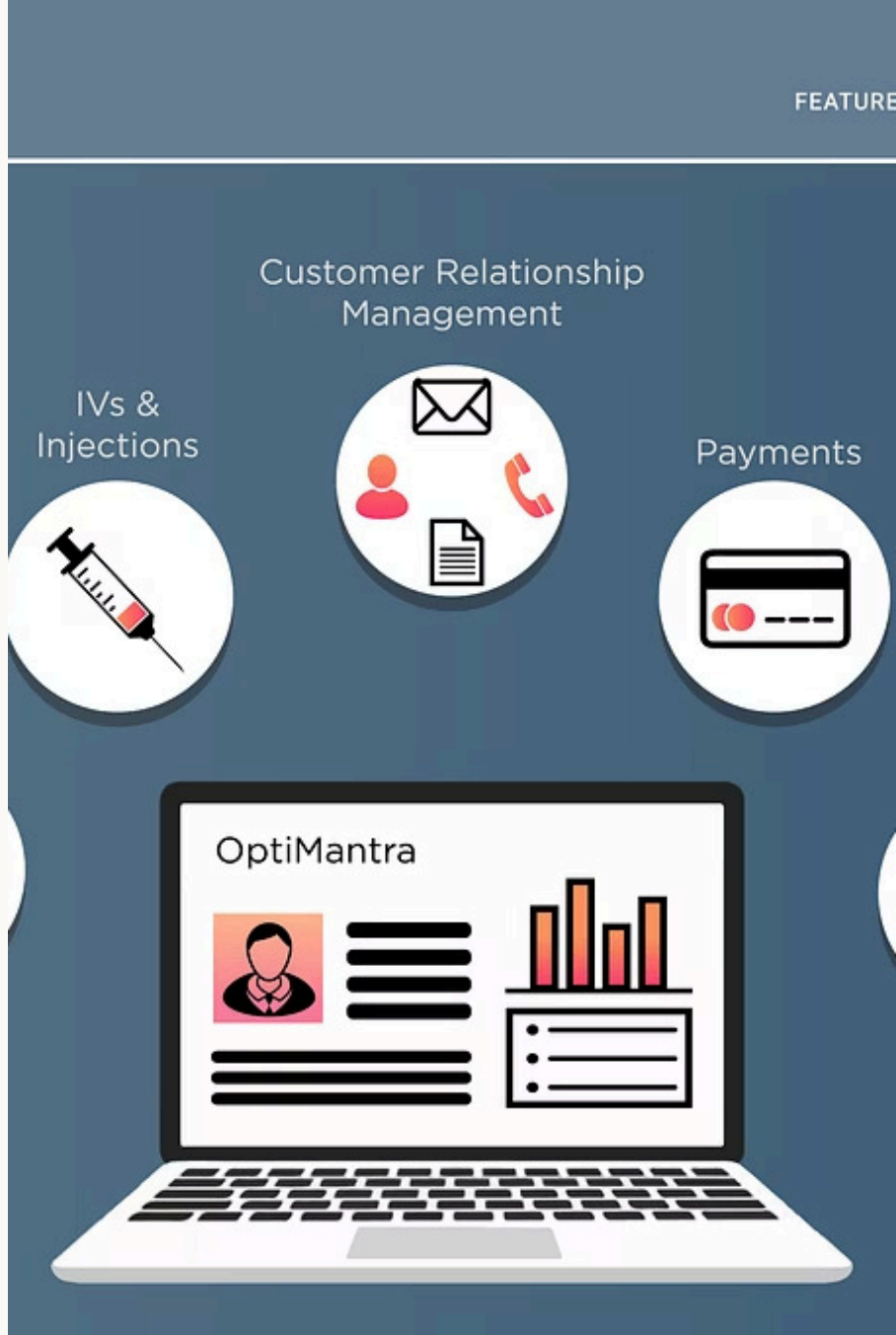
Gestión de Relaciones Bidireccionales

El sistema maneja relaciones bidireccionales sincronizadas entre entidades de forma automática. Cuando se asigna un médico a un departamento, ambos objetos actualizan sus referencias mutuas automáticamente, manteniendo la consistencia de datos en todo momento.

Esta sincronización automática evita inconsistencias de datos y reduce significativamente los errores de programación relacionados con la gestión manual de referencias.

```
public void setHospital(Hospital hospital) {
 if (this.hospital != hospital) {
 // Desregistro del hospital anterior
 // Registro en el nuevo hospital
 // Actualización bidireccional
 }
}
```

Funcionalidades Principales



Gestión de Pacientes

Historia Clínica Única

Cada paciente recibe automáticamente una historia clínica única con identificador generado siguiendo el formato **HC-DNI-AÑO**. Este sistema garantiza la trazabilidad completa de cada paciente.

La historia permite registrar diagnósticos, tratamientos y alergias de forma incremental, manteniendo un historial médico completo y actualizado que es fundamental para la continuidad del cuidado médico.

```
private String generarNumeroHistoria() {
    return "HC-" + paciente.getDni() + "-" + fechaCreacion.getYear();
    // Formato: HC-12345678-2025
    // Garantiza unicidad temporal por paciente
}
```

Sistema de Citas Médicas

La clase **Cita** actúa como entidad asociativa compleja, resolviendo la relación muchos-a-muchos entre pacientes, médicos y salas de forma elegante y eficiente.

Relaciones Múltiples

- Paciente ↔️ Medico (N:M)
- Medico ↔️ Sala (N:M)
- Paciente ↔️ Sala (N:M)

Atributos Propios

- Fecha y hora
- Costo del servicio
- Observaciones médicas

Validaciones de Negocio

- Verificación de disponibilidad
- Compatibilidad de especialidades
- Estados de cita válidos

Validaciones de Integridad Multi-Nivel

El sistema implementa múltiples capas de validación para garantizar la integridad y consistencia de los datos en todos los niveles de la aplicación.

Nivel Sintáctico

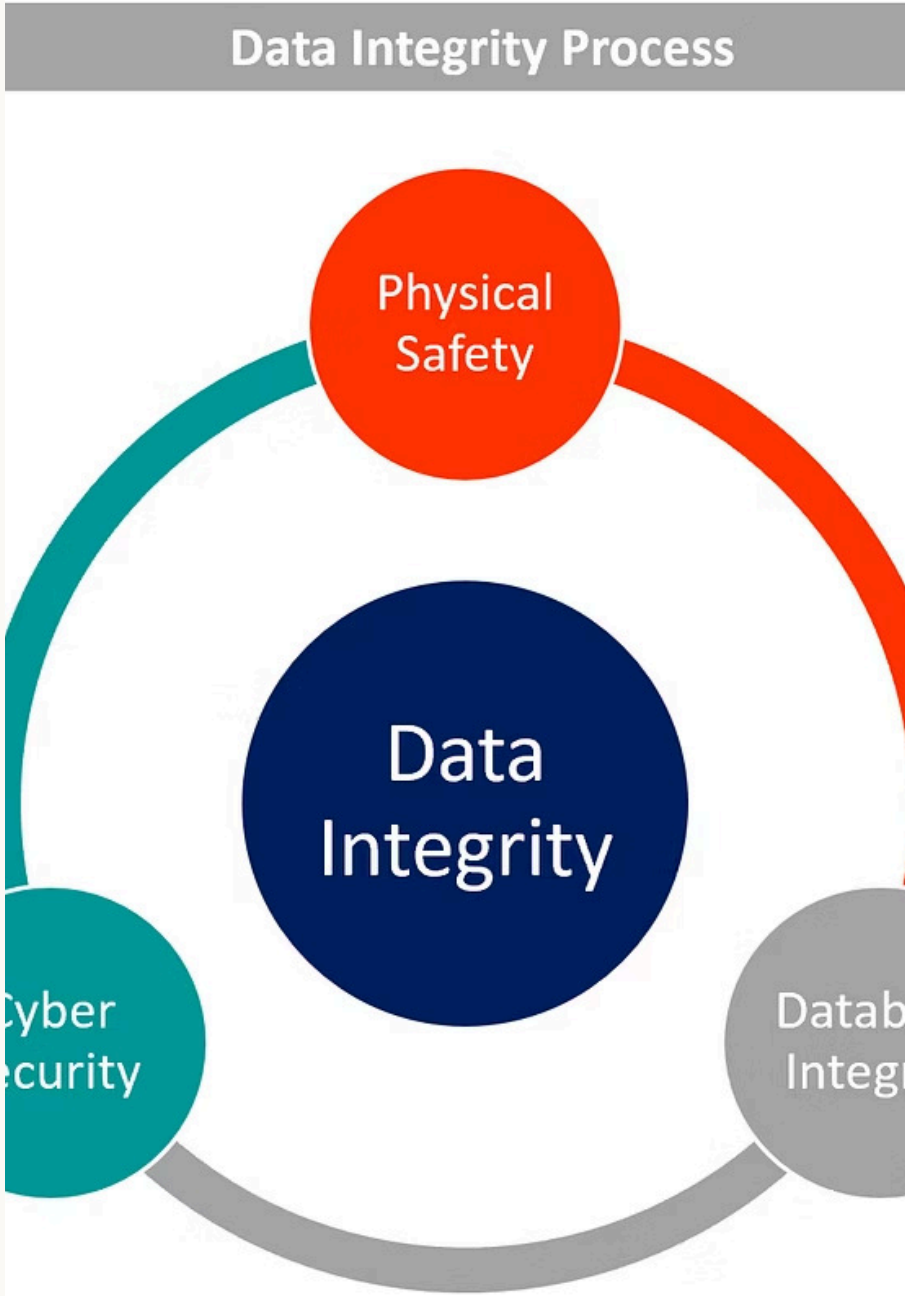
Validación de formato de datos como DNI, matrícula médica y números de identificación con expresiones regulares específicas.

Nivel Semántico

Aplicación de reglas de negocio como fechas futuras para citas, costos positivos y rangos de valores válidos.

Nivel de Consistencia

Verificación de integridad referencial como compatibilidad entre especialidad del médico, departamento y sala asignada.



Ejemplos de Validación Implementada

Validación Sintáctica de DNI

```
private String validarDni(String dni) {
    if (!dni.matches("\\d{7,8}")) {
        throw new IllegalArgumentException("El DNI debe tener 7 u 8 dígitos");
    }
}
```

Validación de Reglas de Negocio

```
private void validarCita(LocalDateTime fechaHora, BigDecimal costo) {
    if (fechaHora.isBefore(LocalDateTime.now())) {
        throw new CitaException("No se puede programar una cita en el pasado");
    }
    // Múltiples validaciones de coherencia
}
```

Validación de Consistencia Referencial

```
if (!medico.getEspecialidad().equals(sala.getDepartamento().getEspecialidad())) {
 throw new CitaException("Especialidad incompatible");
}
```

Persistencia y Recuperación

Serialización CSV Personalizada

El sistema implementa un sistema de persistencia personalizado que convierte objetos complejos a formato CSV, manejando caracteres especiales y referencias entre entidades de forma robusta.

La carga reversa reconstruye el grafo de objetos manteniendo todas las relaciones bidireccionales y la integridad referencial del sistema completo.

Serialización de Referencias

```
public String toCsvString() {
 return String.format("%s,%s,%s,%s,%s,%s,%s",
 paciente.getDni(), // Referencia por clave natural
 medico.getDni(), // Evita dependencias circulares
 sala.getNumero(), // Identificador único
 fechaHora.toString(), // Formato ISO estándar
 costo.toString(), // Precisión decimal preservada
 estado.name(), // Enum serialization
 observaciones.replaceAll(",", ";")); // Escape de caracteres
}
```

Gestión de Estado Optimizada

El **CitaManager** mantiene índices optimizados utilizando **ConcurrentHashMap** para búsquedas eficientes por paciente, médico o sala, evitando recorridos lineales en operaciones frecuentes.

Búsquedas O(1)

Acceso directo a citas por paciente, médico o sala sin recorridos lineales de toda la colección.

Thread-Safety

Utilización de ConcurrentHashMap para operaciones concurrentes seguras en entornos multi-hilo.

Consistencia Automática

Actualización sincronizada de todos los índices mediante métodos privados de gestión.

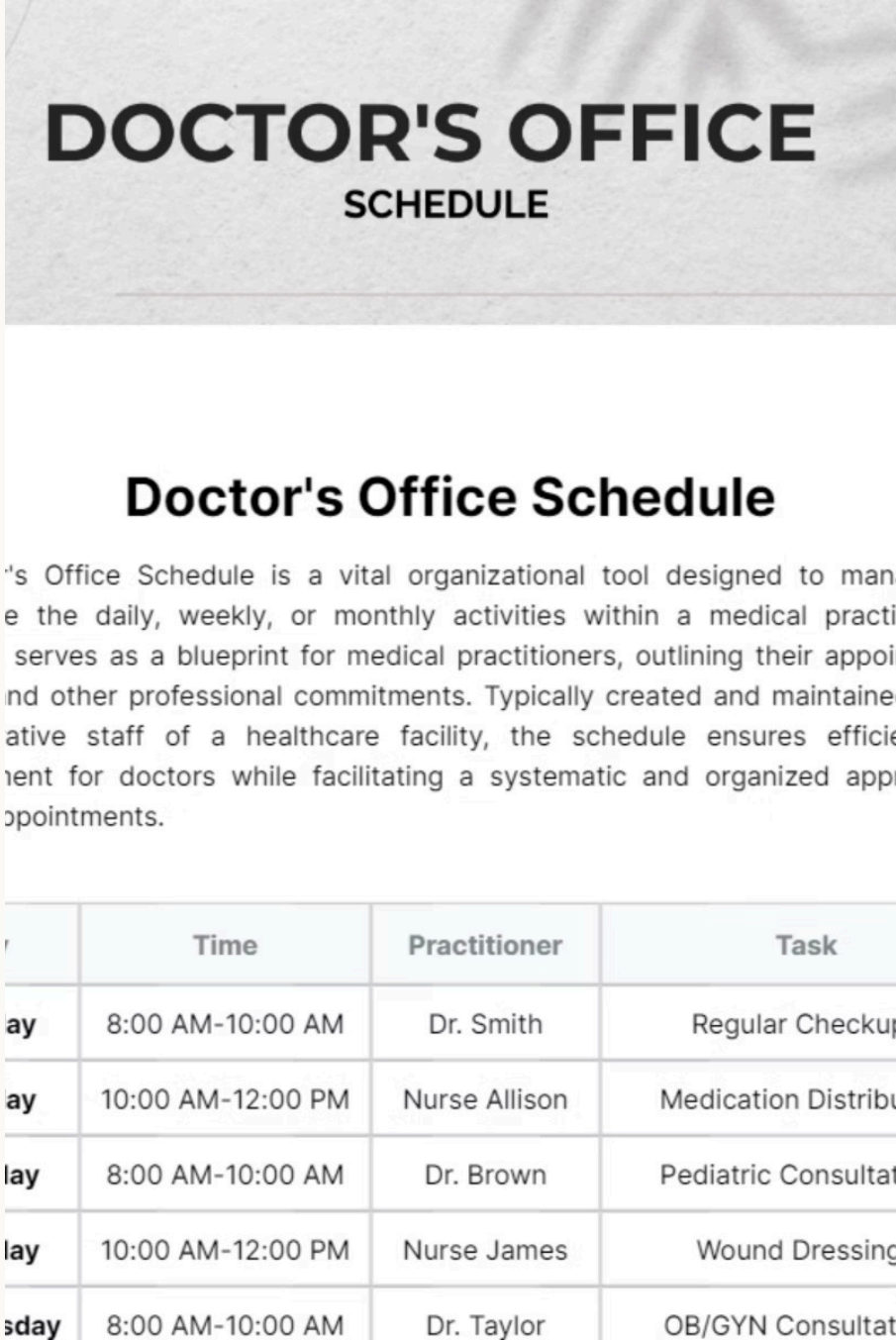
```
private final Map<Paciente, List> citasPorPaciente = new ConcurrentHashMap<>();
private final Map<Medico, List> citasPorMedico = new ConcurrentHashMap<>();
private final Map<Sala, List> citasPorSala = new ConcurrentHashMap<>();
```

Gestión de Disponibilidad Médica

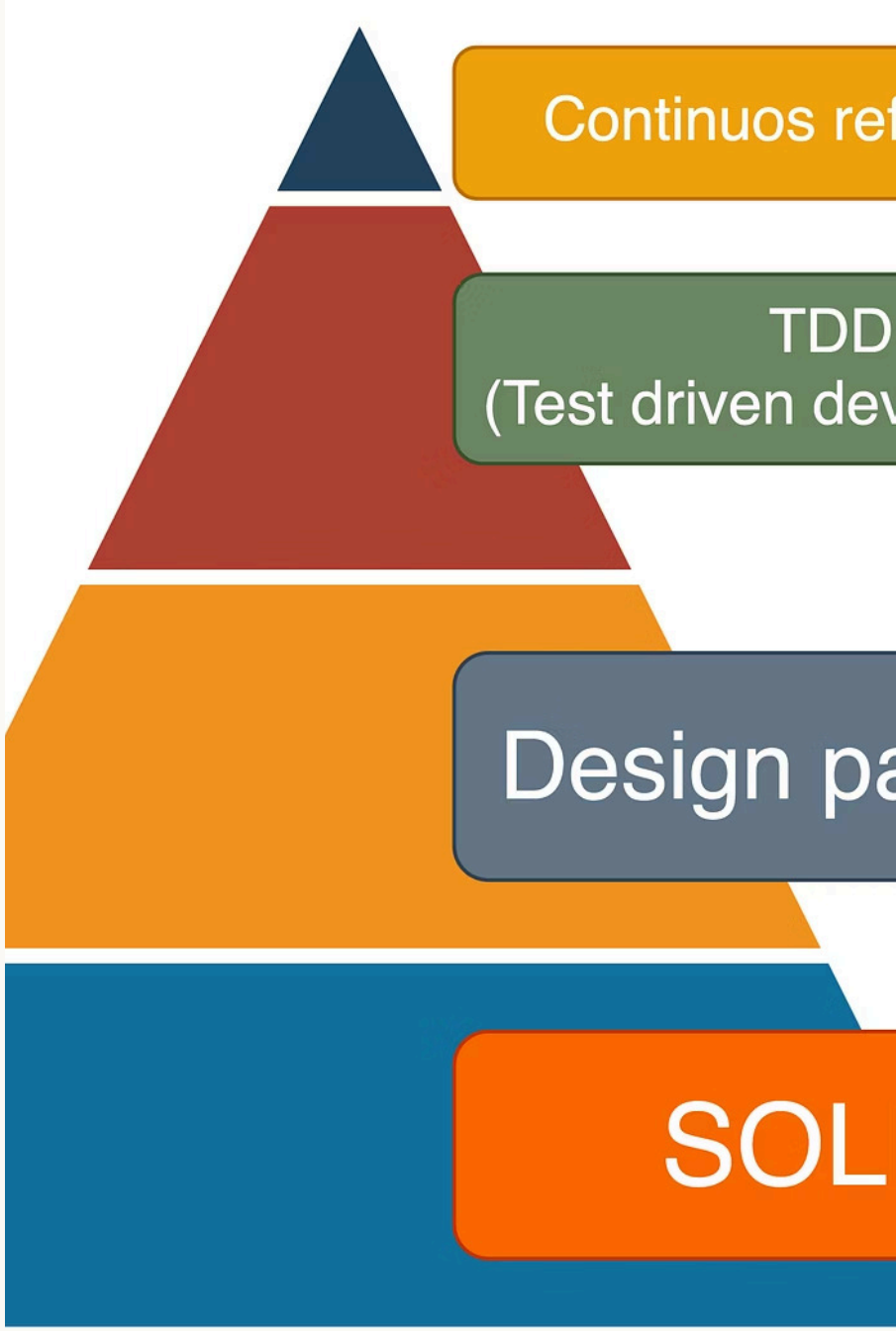
El sistema implementa un algoritmo sofisticado de detección de conflictos temporales que verifica la disponibilidad de médicos considerando ventanas de tiempo y márgenes de seguridad.

```
private boolean esMedicoDisponible(Medico medico, LocalDateTime fechaHora)
{
   // Verifica superposición temporal con margen de 2 horas
   // Implementa lógica de ventana de disponibilidad
}
```

Este algoritmo considera no solo la ocupación directa del médico, sino también tiempos de preparación, descanso y transición entre pacientes para optimizar la experiencia tanto del profesional como del paciente.



Aspectos de Diseño



Principios SOLID Aplicados

1

Responsabilidad Única

Cada clase tiene un propósito específico: HistoriaClinica gestiona información médica, CitaManager coordina operaciones, Matricula valida credenciales.

2

Abierto/Cerrado

Las enumeraciones permiten agregar especialidades sin modificar código existente. Sistema extensible mediante herencia.

3

Sustitución de Liskov

Los subtipos de Persona (Paciente, Medico) son intercambiables en contextos apropiados manteniendo el comportamiento esperado.

4

Segregación de Interfaces

CitaService define un contrato específico sin métodos innecesarios, permitiendo implementaciones focalizadas.

5

Inversión de Dependencias

CitaManager depende de abstracción (CitaService), facilitando testing y implementaciones alternativas.

Patrones de Diseño Implementados

Template Method

Implementado en la clase abstracta Persona para definir algoritmos comunes de validación que las subclases pueden especializar según sus necesidades específicas.

Service Layer

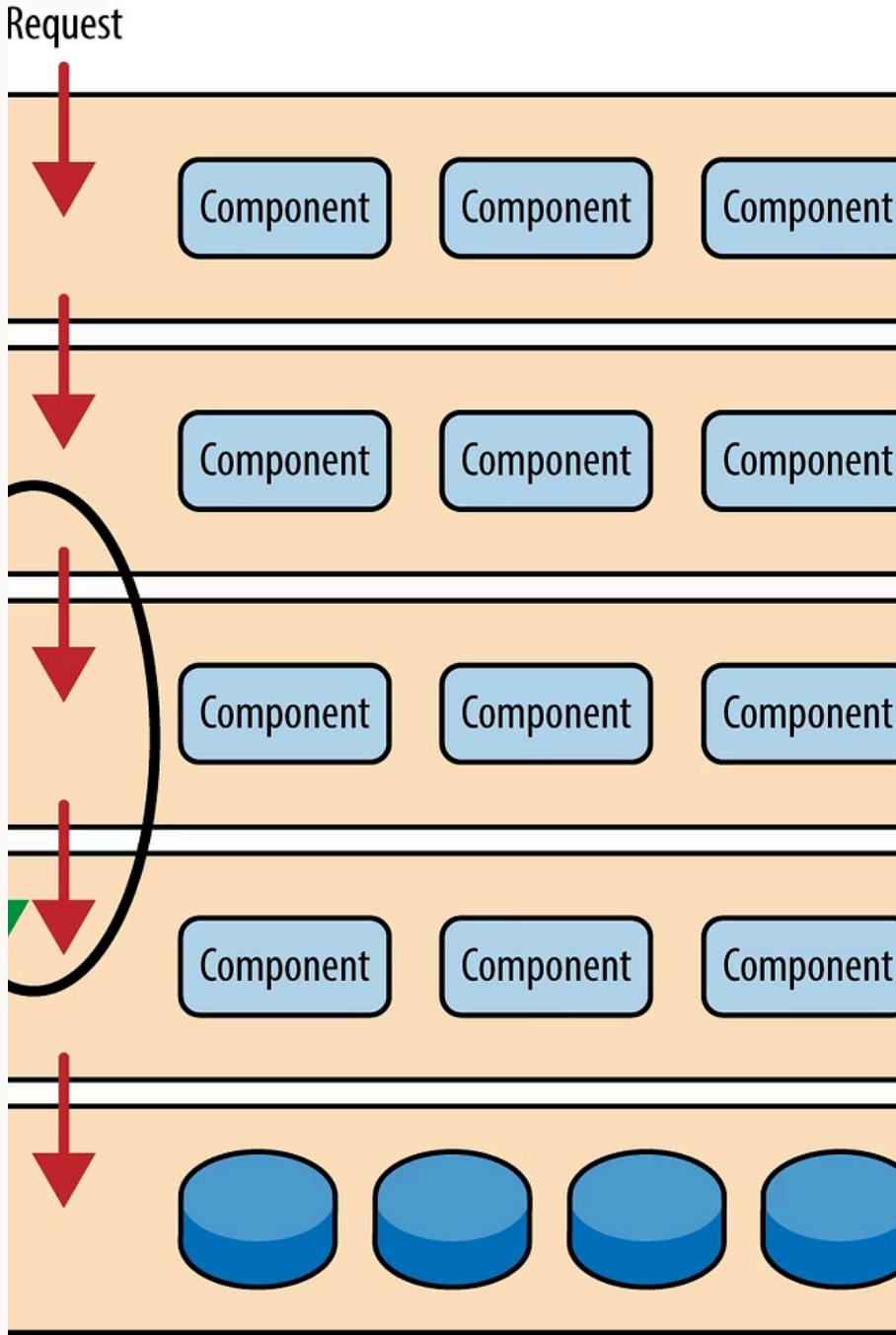
Separación clara entre lógica de negocio (CitaManager) y modelo de datos, facilitando el mantenimiento y testing independiente.

Strategy Pattern

Las enumeraciones encapsulan comportamientos específicos por dominio, permitiendo extensibilidad sin modificación de código base.

Observer Pattern

Implementación implícita en relaciones bidireccionales que mantienen sincronización automática entre entidades relacionadas.



Manejo de Errores Granular

El sistema define excepciones específicas como **CitaException** que proporcionan contexto detallado sobre fallos de validación, permitiendo diagnóstico preciso y recuperación inteligente.

Contexto Específico

Las excepciones proporcionan información detallada sobre el tipo de error, facilitando el debugging y la resolución de problemas.

Manejo Granular

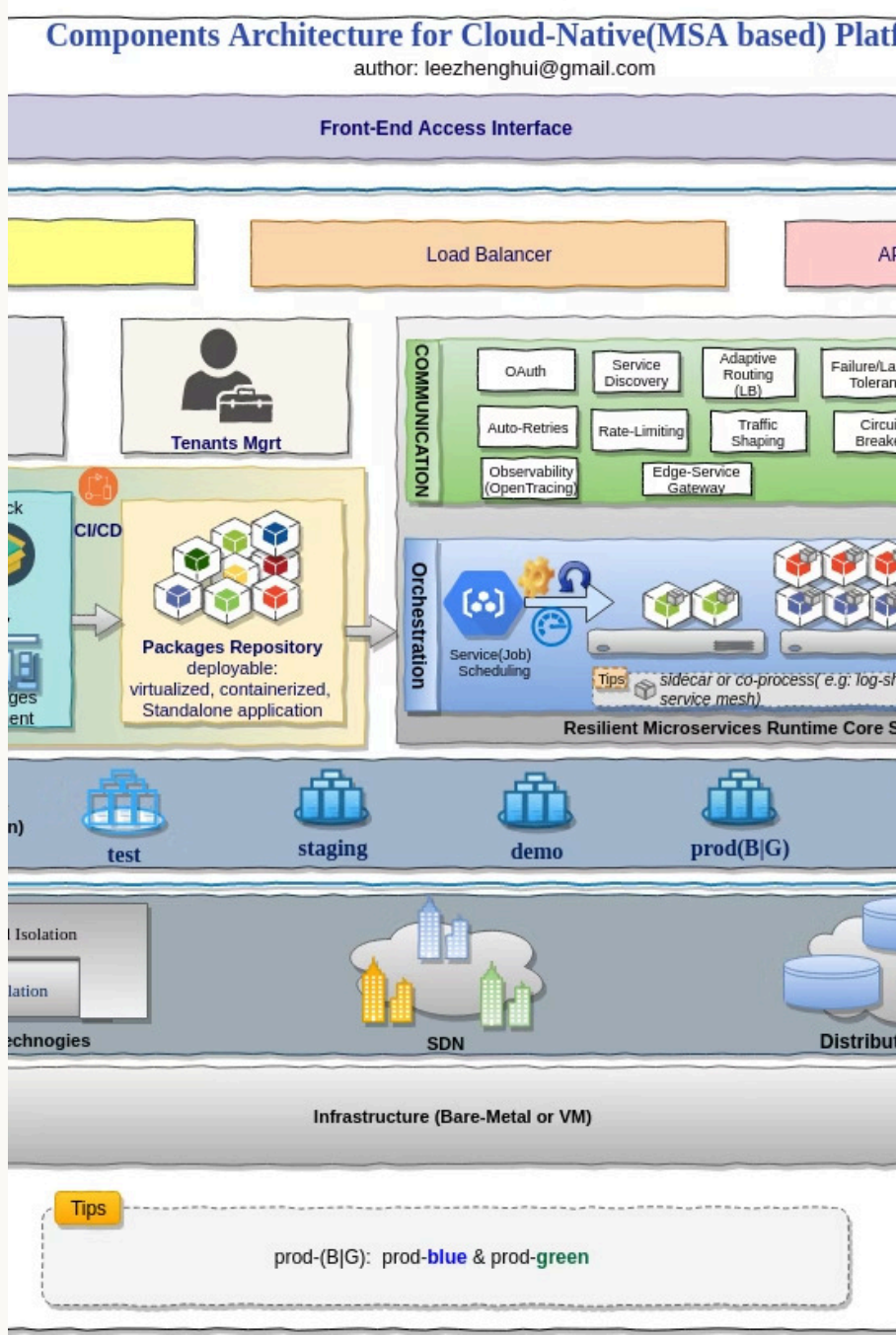
Los errores se manejan de forma específica, permitiendo recuperación parcial en operaciones batch sin afectar el proceso completo.

Fail-Fast Strategy

Las validaciones ocurren en construcción del objeto, detectando problemas inmediatamente y evitando estados inconsistentes.

```
public class CitaException extends Exception {
 // Proporciona contexto específico del dominio médico
 // Permite manejo granular de diferentes tipos de error
 // Facilita logging y debugging
}
```

Escalabilidad y Extensibilidad

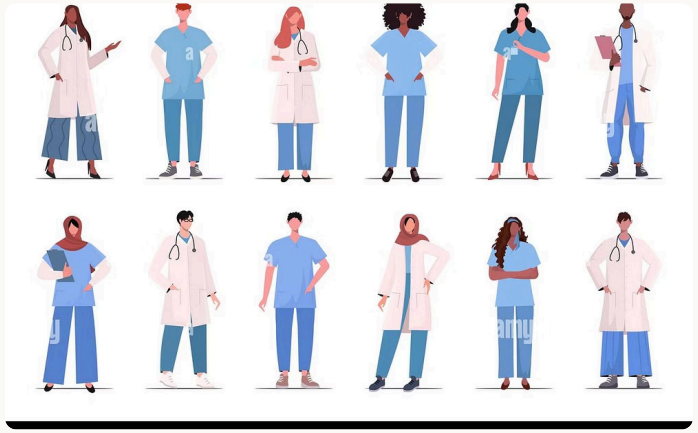


Arquitectura Extensible

La arquitectura permite expansiones naturales sin modificar el código base existente, garantizando la evolución continua del sistema según las necesidades cambiantes del entorno hospitalario.

Nuevas Especialidades Médicas

Mediante enumeraciones extensibles que no requieren modificación de lógica existente.



Tipos Adicionales de Personal

Extendiendo la clase Persona para enfermeros, técnicos y otros roles hospitalarios.

Nuevos Estados de Cita

Sin impacto en lógica existente, permitiendo workflows más complejos.

Ejemplos de Extensibilidad

Nuevas Especialidades Médicas

```
public enum EspecialidadMedica {
    // Agregar nuevas sin modificar código existente
    NUEVA_ESPECIALIDAD("Descripción")
}
```

Nuevos Tipos de Personal

```
public class Enfermero extends Persona {
    // Herencia natural de comportamientos base
    // Especialización específica del rol
}
```

Integración con Bases de Datos

La capa de persistencia CSV puede ser reemplazada por implementaciones de base de datos relacionales o NoSQL sin afectar la lógica de negocio, gracias a la separación de responsabilidades implementada.

Modelo de Datos Completo

El sistema modela 26 elementos principales organizados en una arquitectura coherente y bien estructurada:

3

Enumeraciones

TipoSangre, EspecialidadMedica, EstadoCita

10

Clases Principales

Hospital, Departamento, Persona, Paciente, Medico, etc.

2

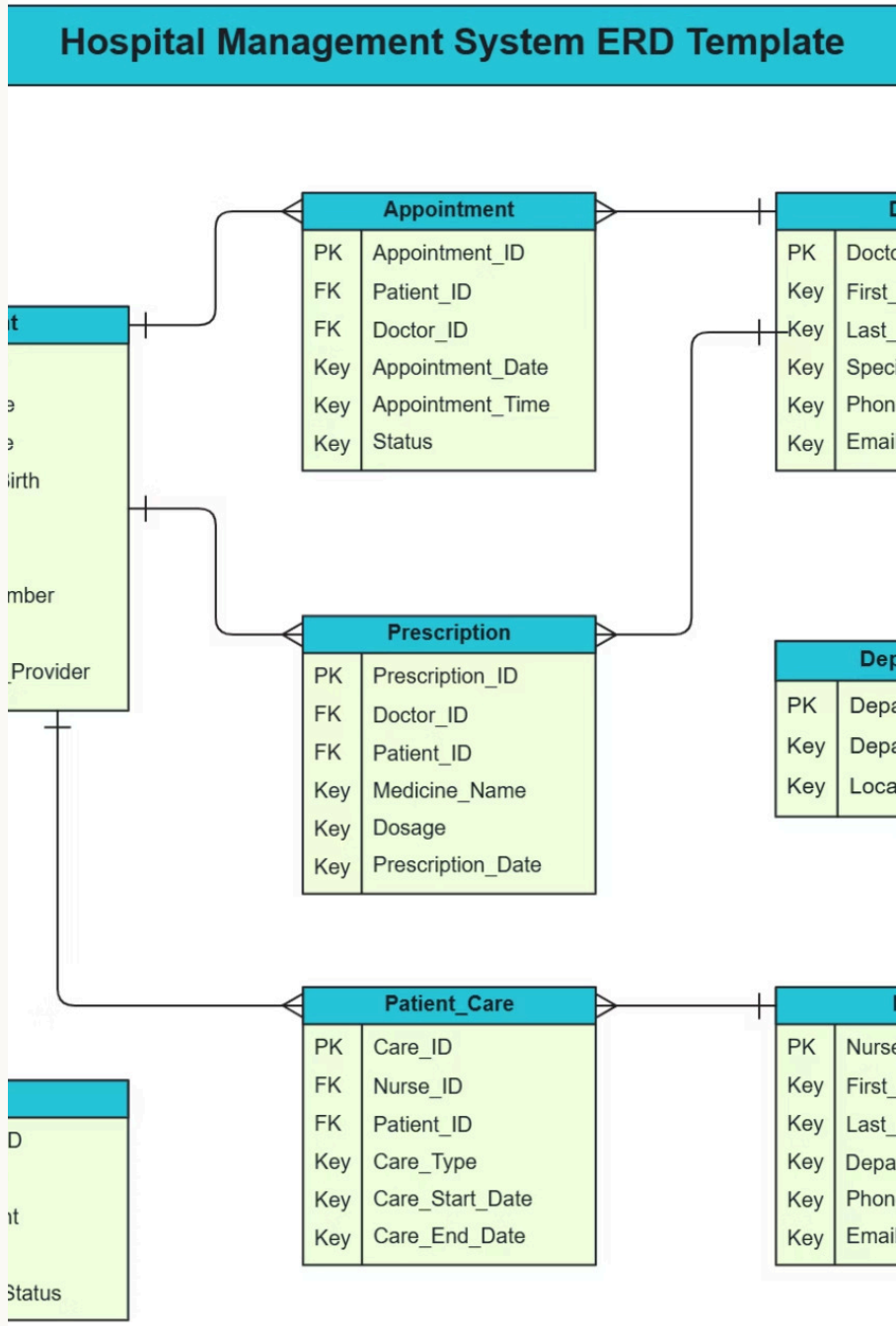
Servicios

CitaService (interface) y CitaManager (implementación)

8

Tipos de Relaciones

Herencia, composición, agregación y asociaciones múltiples



Tipos Sanguíneos y Especialidades

TipoSangre (8 tipos)

- A+, A-, B+, B-
- AB+, AB-, O+, O-

Esencial para emergencias médicas y compatibilidad en transfusiones.

EspecialidadMedica (12 especialidades)

- Cardiología, Neurología
- Pediatría, Ginecología
- Traumatología, Dermatología
- Y 6 especialidades adicionales

Estos enums proporcionan la base para la organización departamental y la compatibilidad médico-paciente en el sistema.

Análisis de Complejidad y Rendimiento

El sistema está optimizado para diferentes tipos de operaciones con complejidades algorítmicas específicas:

Búsquedas por Índice

getCitasPorPaciente(), acceso directo via HashMap

Búsquedas con Filtros

Criterios complejos requieren recorrido completo

Creación con Relaciones

Donde k = número de relaciones bidireccionales

Generación Automática de Identificadores

El sistema implementa algoritmos inteligentes para la generación automática de identificadores únicos que garantizan la trazabilidad completa:

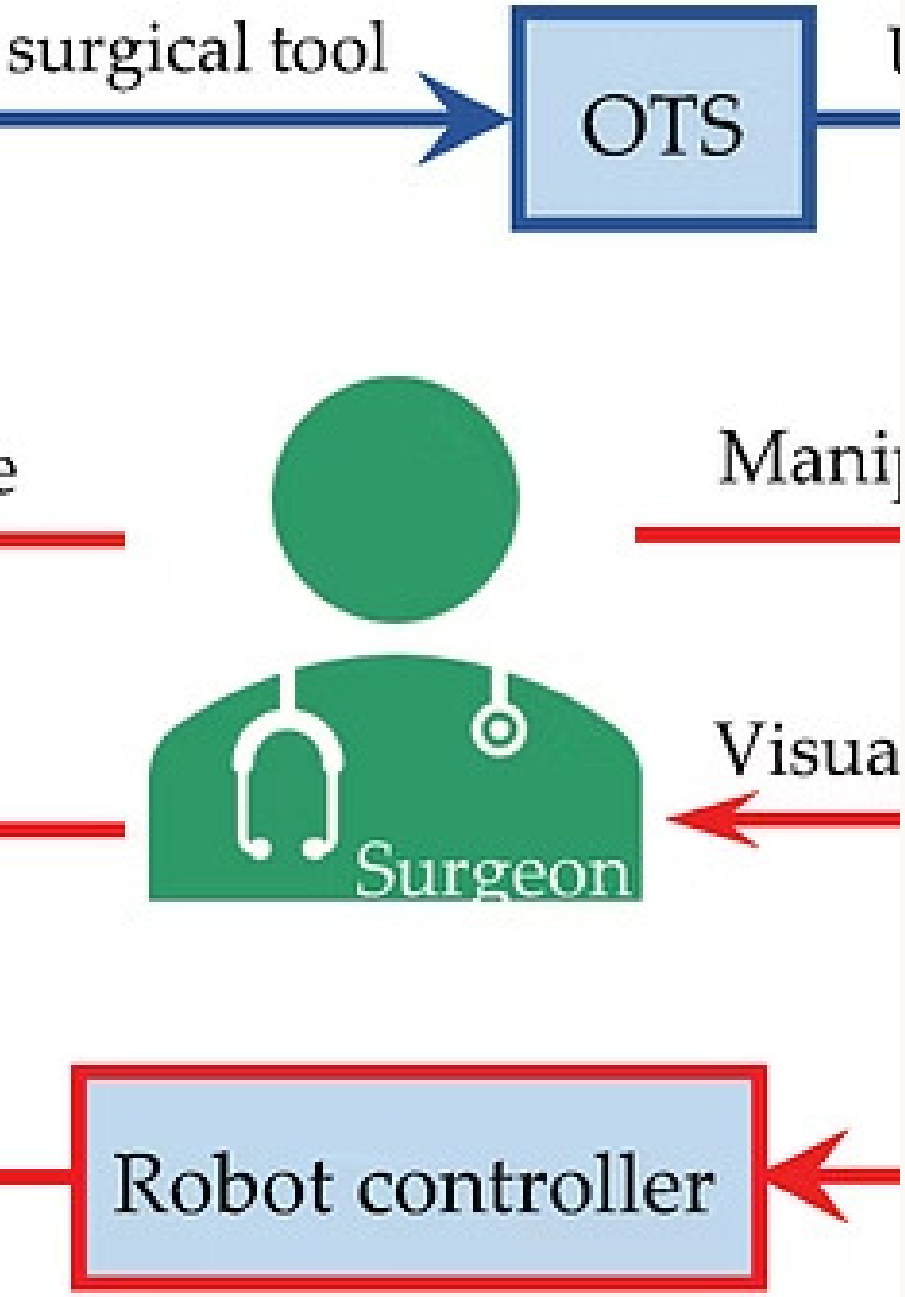
Historia Clínica

```
private String generarNumeroHistoria() {
    return "HC-" + paciente.getDni() + "-" + fechaCreacion.getYear();
    // Formato: HC-12345678-2025
    // Garantiza unicidad temporal por paciente
}
```

Cálculos Dinámicos

```
public int getEdad() {
 return LocalDate.now().getYear() - fechaNacimiento.getYear();
 // Cálculo dinámico, siempre actualizado
}

public long getDiasEstadia() {
 return ChronoUnit.DAYS.between(checkIn, checkOut);
 // Métrica de negocio directamente disponible
}
```



Sincronización Bidireccional

Una de las características más avanzadas del sistema es la sincronización bidireccional automática que mantiene la consistencia entre todas las entidades relacionadas:

1

Agregar Paciente al Hospital

Actualiza automáticamente las referencias en ambas direcciones, manteniendo la integridad referencial.

2

Asignar Médico a Departamento

Sincroniza instantáneamente las listas de médicos en el departamento y la referencia al departamento en el médico.

3

Crear Cita Médica

Actualiza automáticamente los índices en paciente, médico y sala, garantizando consultas eficientes.

Diseño por Capas Implementado

Capa de Presentación

Interfaces de usuario y APIs REST

Capa de Servicios

CitaService, lógica de coordinación

Capa de Negocio

Validaciones, reglas de dominio

Capa de Entidades

13 clases del modelo hospitalario

Capa de Persistencia

Serialización CSV, futuras BD

Conclusiones y Beneficios

Diseño Robusto

Arquitectura que balancea simplicidad de uso con flexibilidad técnica, proporcionando una base sólida para operaciones hospitalarias reales.

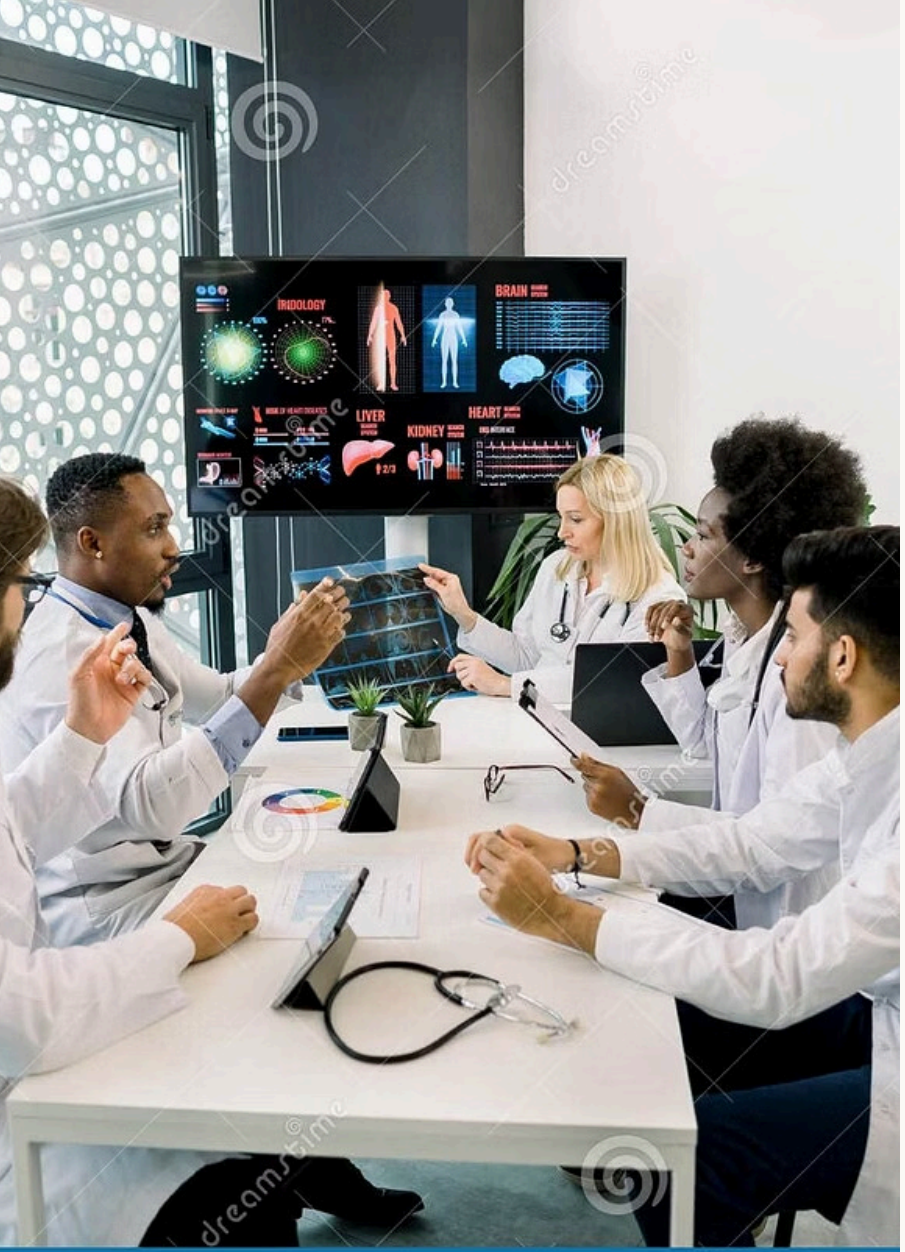
Escalabilidad Probada

Sistema preparado para crecimiento tanto en funcionalidad como en volumen de datos, con patrones establecidos para extensión.

Mantenibilidad Alta

Código bien estructurado con responsabilidades claras, facilitando el mantenimiento y la evolución continua del sistema.

El sistema demuestra un diseño maduro que equilibra complejidad de implementación con facilidad de uso, estableciendo las bases para un sistema de gestión hospitalaria de nivel empresarial.



¿Preguntas?

Gracias por su atención. Este sistema de gestión hospitalaria representa un enfoque moderno y escalable para la administración integral de instituciones sanitarias.

> "Un diseño robusto que balancea simplicidad de uso con flexibilidad técnica, proporcionando una base sólida para operaciones hospitalarias reales."