**Pensamiento Computacional Hospital UQ**

**Alejandro García Cadavid**

**Código: 1095551008  
Luisa Fernanda Salazar**

**Código: 1094888121**

****

**Profesor: Jhan Carlos Martínez Ceballos**

**Universidad del Quindío**

**Facultad de Ingeniería**

**Ingeniería de Sistemas y Computación**

**Programación 1**

**Armenia, Quindío, 2025**

**Pensamiento Computacional - Sistema de Gestión Hospital UQ**

**1. Abstracción**

***¿Qué se solicita finalmente?***

Desarrollar un sistema de gestión para el Hospital UQ que permita administrar pacientes, citas médicas y personal médico de manera eficiente, con una interfaz segura e intuitiva.

***¿Qué información es relevante dado el problema anterior?***

- Datos personales de pacientes y médicos.

- Historiales médicos.

- Horarios y disponibilidad de médicos.

- Registro de citas médicas.

- Información administrativa del hospital.

***¿Cómo se agrupa la información relevante*?**

Administrador

Registro de médicos y pacientes.

Gestión de salas y horarios de atención.

Generación de reportes sobre citas médicas y ocupación del hospital.

Farmacia

String id

String nombre

String direccion

String telefono

Paciente

String direccion

String telefono

Historial médico.

Solicitud y cancelación de citas médicas.

Usuario

String id

String nombre

String correo

Medico

String especialidad

Acceso a historiales médicos de pacientes.

Registro de diagnósticos y tratamientos.

Administración de horarios de consulta.

Cita

String id

Paciente paciente

Medico medico

LocalDateTime fechaHora

String salaId

String estado

ArchivoMedico

String id

String pacienteId

String nombreArchivo

String tipoArchivo

byte[] contenido

LocalDateTime fechaSubida

.

Reporte

String id

String descripcion

String tipo

HistorialMedico

String pacienteId

List<String> diagnosticos

Sala

String id

String nombre

int capacidad

Medicamento

String id

String nombre

String descripcion

int cantidadDisponible

***¿Qué funcionalidades se solicitan?***

- Registro y actualización de datos personales.

- Solicitud, modificación y cancelación de citas médicas.

- Consulta y actualización de historiales médicos.

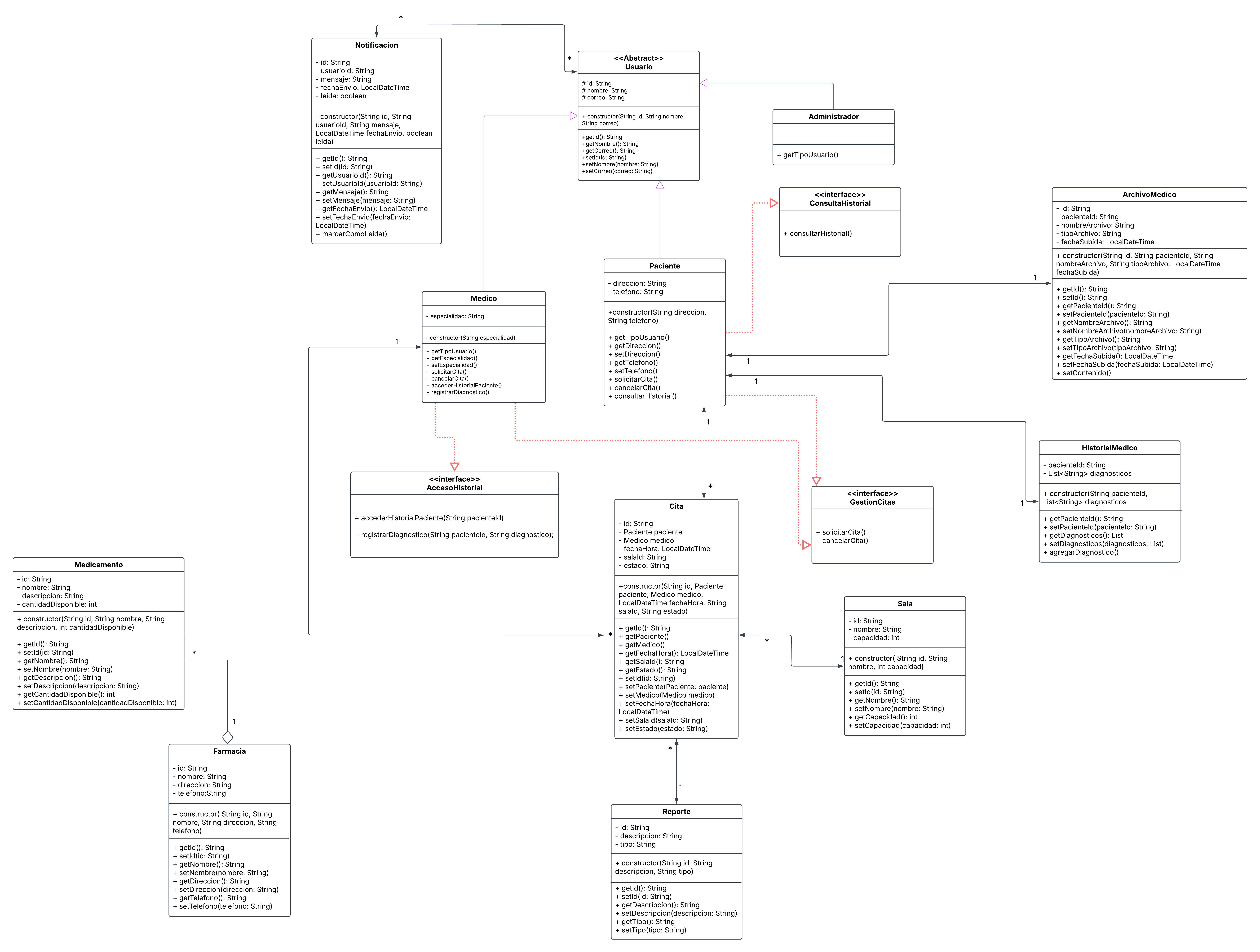
- Administración de personal, salas y horarios.

- Generación de reportes médicos y ocupación del hospital.

- Notificaciones sobre cambios en citas y tratamientos.

**2. Descomposición**

***¿Cómo se distribuyen las funcionalidades?***



Link diagrama Lucidchard:

[*https://lucid.app/lucidchart/fac1444e-f819-456c-8333-458613c9a615/edit?viewport\_loc=755%2C-987%2C2304%2C1028%2C0\_0&invitationId=inv\_412e006d-fc85-440e-b98a-f73b687645bf*](https://lucid.app/lucidchart/fac1444e-f819-456c-8333-458613c9a615/edit?viewport_loc=755%2C-987%2C2304%2C1028%2C0_0&invitationId=inv_412e006d-fc85-440e-b98a-f73b687645bf)

- Los pacientes gestionan sus datos personales, citas y consultas de historial médico.

- Los médicos gestionan diagnósticos, tratamientos y horarios.

- Los administradores gestionan registros de usuarios, salas y reportes.

***¿Qué debo hacer para probar las funcionalidades?***

- Crear objetos de cada clase (`Paciente`, `Medico`, `Administrador`).

- Asignar citas y relaciones entre entidades (por ejemplo, un paciente con su médico).

- Ejecutar métodos que simulan acciones del sistema (registrar cita, actualizar historial, generar reporte).

**3. Reconocimiento de Patrones**

***¿Qué puedo reutilizar de la solución de otros problemas?***

- Uso de herencia (`Persona` → `Paciente`, `Medico`, `Administrador`).

- Interfaces para definir comportamientos comunes (por ejemplo, gestionadores de citas).

- Encapsulamiento para proteger datos personales.

- Polimorfismo para permitir que distintos usuarios ejecuten métodos similares de manera distinta.

**4. Codificación**

***¿Cómo pruebo la solución en Java?***

- Usando pruebas unitarias con JUnit.

- Probando la interfaz gráfica (JavaFX) y simulando las acciones de los usuarios.

- Comprobando la persistencia en memoria y la correcta asignación de relaciones entre clases.

***¿Cómo escribo la solución en Java?***

- Crear clases con atributos y métodos adecuados.

- Aplicar herencia, polimorfismo, encapsulamiento e interfaces.

- Diseñar una interfaz gráfica intuitiva.