SISTEMA DE CITAS MEDICAS

ELABORADO POR: SERGIO NICOLAS JUYO PAMPLONA

INGENIERIA DE SOFTWARE

PROFESOR

WILLIAM ALEXANDER MATALLANA PORRAS

UNIVERSIDAD DE CUNDINAMARCA EXTENSION CHIA

PROGRAMA DE INGENIERIA DE SISTEMAS Y COMPUTACION

11 DE FEBRERO DEL 2026

Ingenieria de software I

1. Información General del Proyecto

Nombre del proyecto	Sistema de Citas Medicas
Integrantes	Sergio Nicolas Juyo Pamplona
Programa académico	Ingenieria de Sistemas y Computación
Fecha	18/02/2026
Lenguaje de programación	Java
Tipo de aplicación	Consola

2. Descripción General

el sistema permite gestionar las citas medicas de un consultorio, permitiendo registrar, listar, cambiar estado y eliminar citas.

Cada cita tiene un estado que determina su disponibilidad (programada, cancelada o atendida). El sistema valida reglas de negocio como que una cita cancelada no puede marcarse como atendida.

La aplicación esta desarrollada en java aplicando principios de programación orientada a objetos con arquitectura de capas (Controller, Model, Service, View).

3. Requerimientos Funcionales (RF)

ID	Nombre	Descripción	Entradas	Procesos	Salidas
RF-01	Registrar cita	Permite registrar una nueva cita medica	Nombre del paciente, fecha	Genera ID automático y asigna estado PROGRAMADA	Cita registrada con ID único
RF-02	Listar citas	Muesra todas las citas registradas en el sistema		Recorre la lista de citas	Lista completa de citas
RF-03	Cambiar estado de cita	Permite modificar el estado de una cita existente	ID de la cita, nuevo estado	Valida reglas de negocio y actualiza estado	Cita con estado actualizado
RF-04	Eliminar cita	Permite eliminar una cita del sistema	ID de la cita	Busca y elimina la cita de la lista	Confirmación de eliminación
RF-05	Buscar cita por ID	Permite consultar una cita especifica por su identificador	ID de la cita	Busca en la lista de citas	Datos de la cita encontrada

4. Requerimientos No Funcionales (RNF)

ID	Tipo	Descripción
RNF-01	Validación	El sistema no debe permitir registrar citas con nombre de paciente vacío
RNF-02	Validación	El sistema no debe permitir registrar citas con fecha vacía
RNF-03	Regla de negocio	Una cita cancelada no puede marcarse como atendida
RNF-04	Regla de negocio	Una cita atendida no puede cambiar de estado
RNF-05	Identificador	El ID de la cita debe generarse automáticamente
RNF-06	Estructura	El sistema debe estar organizado por paquetes: controller, model, service, view
RNF-07	Calidad	El sistema debe aplicar encapsulamiento y manejo de excepciones

5. Relación Requerimiento-POO

ID Requerimiento	Clase	Método	Tipo
RF-01	CitaServiceImpl	registrarCita()	Funcional
RF-02	CitaServiceImpl	LiastarCita	Funcional
RF-03	CitaServiceImpl	cambiarEstado()	Funcional
RF-04	CitaServiceImpl	eliminarCita()	Funcional
RF-05	CitaServiceImpl	BuscarCitaPorId()	Funcional
RNF-01	Cita	establecerFecha()	No Funcional
RNF-02	Cita	establecerFecha()	No Funcional
RNF-03	CitaServiceImpl	cambiarEstado()	No Funcional
RNF-04	CitaServiceImpl	cambiarEstado()	No Funcional
RNF-05	CitaServiceImpl	registrarCita()	No Funcional

6. Análisis del Sistema

Una vez teniendo claro los requerimientos, analizamos como debería funcionar el sistema, identificamos un único usuario del sistema y definimos los casos de uso:

- **Registrar cita**: cuando un paciente llama para pedir cita
- **Listar citas**: para ver la agenda del día
- **Cambiar estado**: cuando el paciente se atiende (ATENDIDA) o cancela (CANCELADA)
- **Eliminar cita**: si se registró por error

- **Buscar cita**: cuando un paciente pregunta por su cita

Regla de negocio identificada:

Una cita nueva siempre inicia como PROGRAMADA

Una cita CANCELADA no puede ser ATENDIDA

Una cita ATENDIDA no puede cambiar de estado

El ID debe 8. generarse automáticamente

7. Diseño

Diseñamos la arquitectura del sistema siguiendo el patrón del ejemplo del profesor. Organizamos el código en 4 capas:

- Model: contiene las clases que representan los datos (Cita y EstadoCita)
- Service: Contiene la lógica de negocio (Interfaz CitaService y la implementación de CitaServiceImpl
- Controller: Coordina las peticiones entre la vista y el servicio
- View: Se encarga de mostrar la información al usuario

Main →Controller →Service →Model

→View (muestra los resultados)

7.1 Diagrama de Caso de Uso

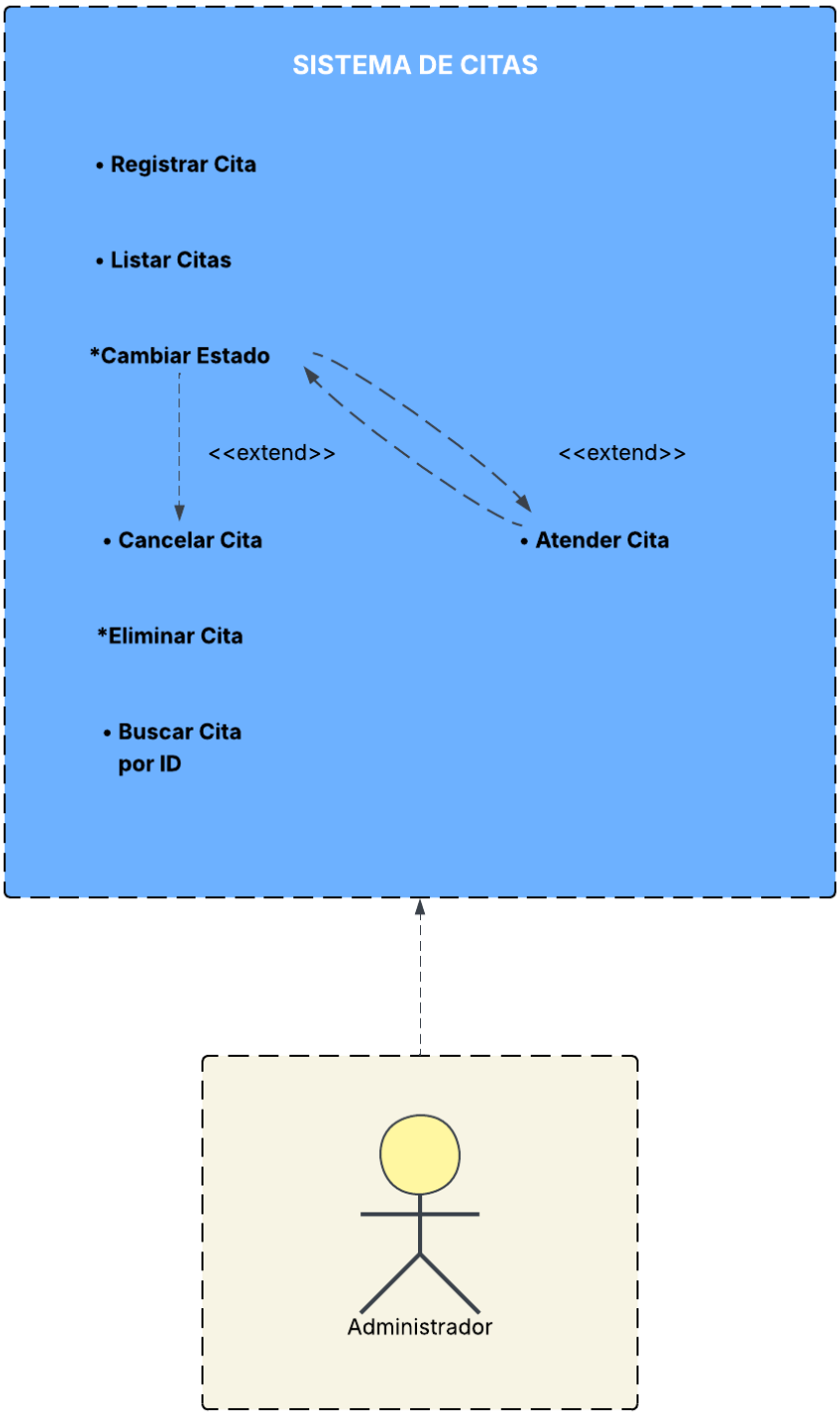

Ilustración 1 Sistema de Citas

7.2 Diagrama de Clases

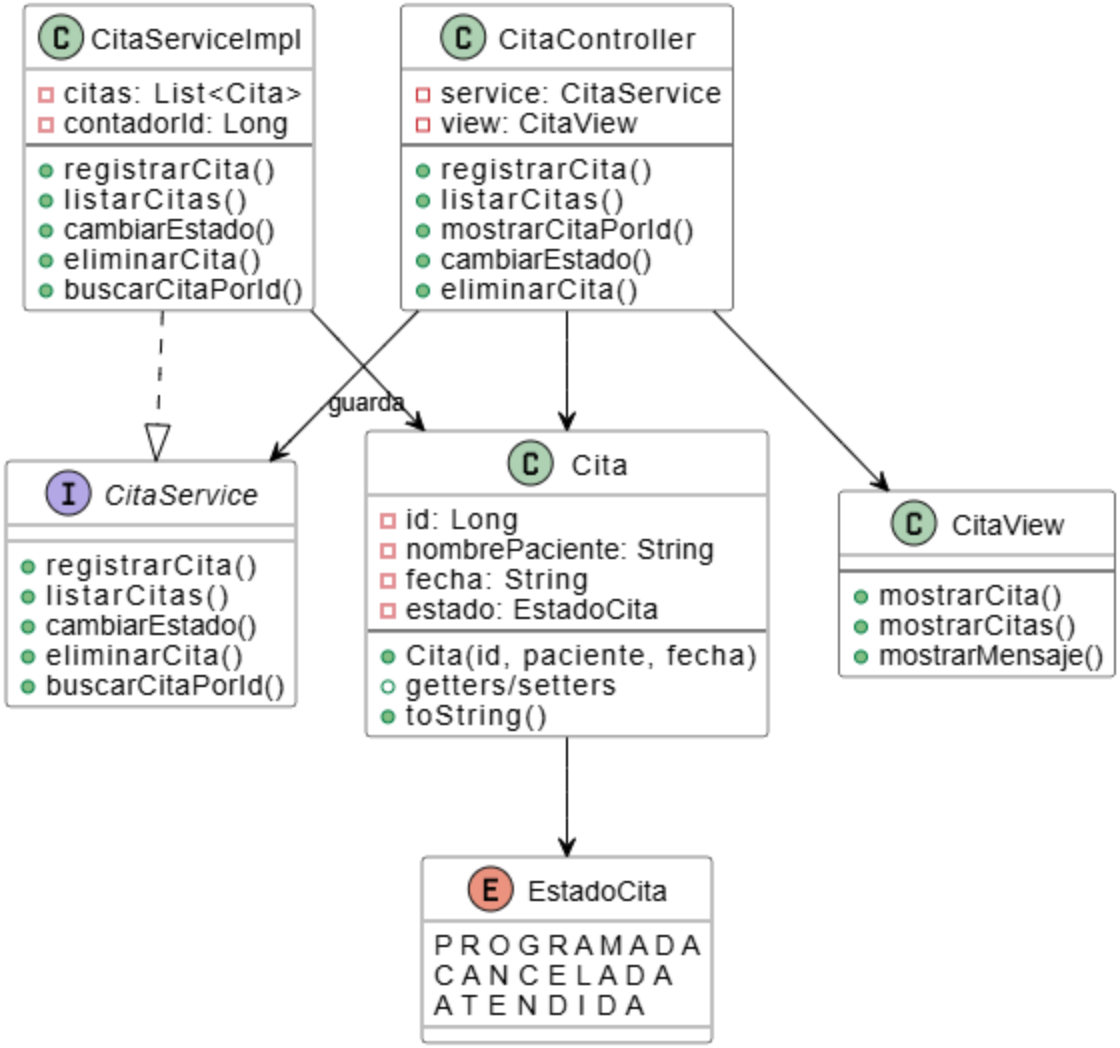

Ilustración 2Diagrama de Clases

8. Implementación

Escribimos el código siguiendo el diseño establecido

1. Crear la estructura de paquete: controller, model, service, view

2. Crear el enumerado EstadoCita: con PROGRAMADA, CANCELADA, ATENDIDA
3. Crear la clase Cita: con sus atributos privados, constructores, getters y setters
4. Crear la interfaz CitaService: definiendo los métodos registrar, listar, cambiarEstado, eliminar y buscarPorId
5. Implementar CitaServiceImpl: con una lista ArrayList para guardar las citas y un contador para IDS automáticos
6. Crear CitaController: que conecte el servicio con las vistas
7. Crear CitaView: con métodos para mostrar la información
8. Crear el Main: para probar todas las funcionalidades

9. URL DEL REPOSITORIO:

https://github.com/serditron69/Ejercicio-4-Consultorio-Medico

CONCLUSION

El desarrollo del Sistema de Citas Médicas permitió aplicar las fases de requerimientos, análisis, diseño e implementación del SDLC, siguiendo la estructura de cuatro capas (controller, model, service, view) del ejemplo del profesor. El sistema cumple con los requerimientos funcionales: registrar citas con ID automático, listarlas, cambiar su estado (programada, cancelada,

atendida), eliminarlas y buscarlas por ID, incluyendo la validación de que una cita cancelada no pueda ser atendida.

Referencias

Oracle Corporation. (2025). *Oracle Corporation*. Obtenido de The Java Tutorials: https://docs.oracle.com/javase/tutorial/

Porras., p. W. (18 de 02 de 2026). *Ejemplo "Sistema de Gestión de Pagos"*. Chia.