Problema #5

Pedro Jose Garcia Correa

Universidad de Cundinamarca

Ingeniería de Software

William Alexander Matallana Porras

17 de febrero del 2026

1. Información general del proyecto

Nombre del proyecto	Sistema de Control de Asistencia
Integrantes	Pedro Jose Garcia Correa
Programa académico	Ingeniería de Sistemas y Computación
Fecha	17/02/2026
Lenguaje de programación	Java
Tipo de aplicación	Consola

2. Descripción General del Sistema

El sistema permite gestionar la asistencia de estudiantes por un registro diario, indicando si cada estudiante asistió o no. Las principales funcionalidades son:

- **Registro de estudiantes**: Se pueden añadir estudiantes con un identificador único y nombre completo, evitando duplicados.
- **Marcado de asistencia**: Para cada estudiante y fecha, se registra el estado (asistió / no asistió), con validación para evitar duplicados el mismo día.
- **Consulta de historial**: Se puede visualizar la lista completa de asistencias de un estudiante específico.
- **Eliminación de registros**: Permite borrar un registro de asistencia erróneo o desactualizado.

3. Levantamiento de Requerimientos

Requerimientos funcionales

ID	Nombre	Descripción	Entrada(s)	Proceso	Salida
RF-01	Registrar Estudiante	Permite registrar un nuevo estudiante con ID único y nombre	ID, Nombre	Verifica que el ID no exista, crea el objeto Estudiante y lo almacena	Estudiante registrado, mensaje de éxito o error
RF-02	Marcar Asistencia	Registra la asistencia de un estudiante en una fecha específica	Estudiante, Fecha, Estado	Busca el estudiante, verifica que no exista registro duplicado para esa fecha y guarda el registro	Registro de asistencia creado, mensaje de éxito o error
RF-03	Consultar Asistencia	Muestra el historial de asistencias de un estudiante	Estudiante	Busca el estudiante, obtiene todos sus registros y los muestra	Lista de registros de asistencia
RF-04	Eliminar Registro Asistencia	Elimina un registro específico de asistencia de un estudiante en una fecha	Estudiante, Fecha	Busca el estudiante, localiza el registro y lo elimina	Mensaje de éxito o error

Requerimientos no Funcionales

ID	Tipo	Descripción
RNF-01	Validación	El sistema no debe permitir registrar estudiantes con ID o nombre vacíos o nulos.
RNF-02	Estructura	El sistema debe estar organizado por paquetes: model, service, controller y view.
RNF-03	Calidad	El sistema debe aplicar encapsulamiento y principios de diseño como inyección de dependencias por constructor.

RNF-04	Mantenibilidad	El código debe seguir principios SOLID, especialmente responsabilidad única e inversión de dependencias.
RNF-05	Persistencia	Los datos se almacenarán en memoria utilizando colecciones de Java (ArrayList).
RNF-06	Interfaz	La interacción con el usuario debe ser a través de consola, con mensajes claros y formato legible.

Relacion Requerimiento Poo

ID Requerimientos	Clase	Método	Tipo
RF-01	EstudianteServiceImp	registrarEstudiante()	Funcional
RF-02	AsistenciaServiceImp	marcarAsistencia()	Funcional
RF-03	AsistenciaServiceImp	consultarAsisEstudiante()	Funcional
RF-04	AsistenciaServiceImp	eliminarRegistro()	Funcional
RNF-01	EstudianteServiceImp	registrarEstudiante()	No funcional
RNF-05	EstudianteServiceImp / AsistenciaServiceImp	atributos estudiantes y registros (ArrayList)	No funcional
RNF-06	AsistenciaView	mostrarMensaje(), mostrarEstudiante()	No funcional

2.Análisis

Actores

- **Docente**: Es el usuario del sistema que realiza las operaciones de registro, marcado, consulta y eliminación.

Casos de uso Identificados

Código	Caso de Uso	Descripción
UC1	Registrar estudiante	El docente ingresa ID y nombre para crear un nuevo estudiante.

UC2	Marcar asistencia	El docente selecciona un estudiante, una fecha y el estado.
UC3	Consultar asistencia	El docente proporciona el ID de un estudiante y Ve todos sus registros de asistencia.
UC4	Eliminar registro	El docente indica el estudiante y la fecha para eliminar un registro específico.

Diagrama de Casos de Uso

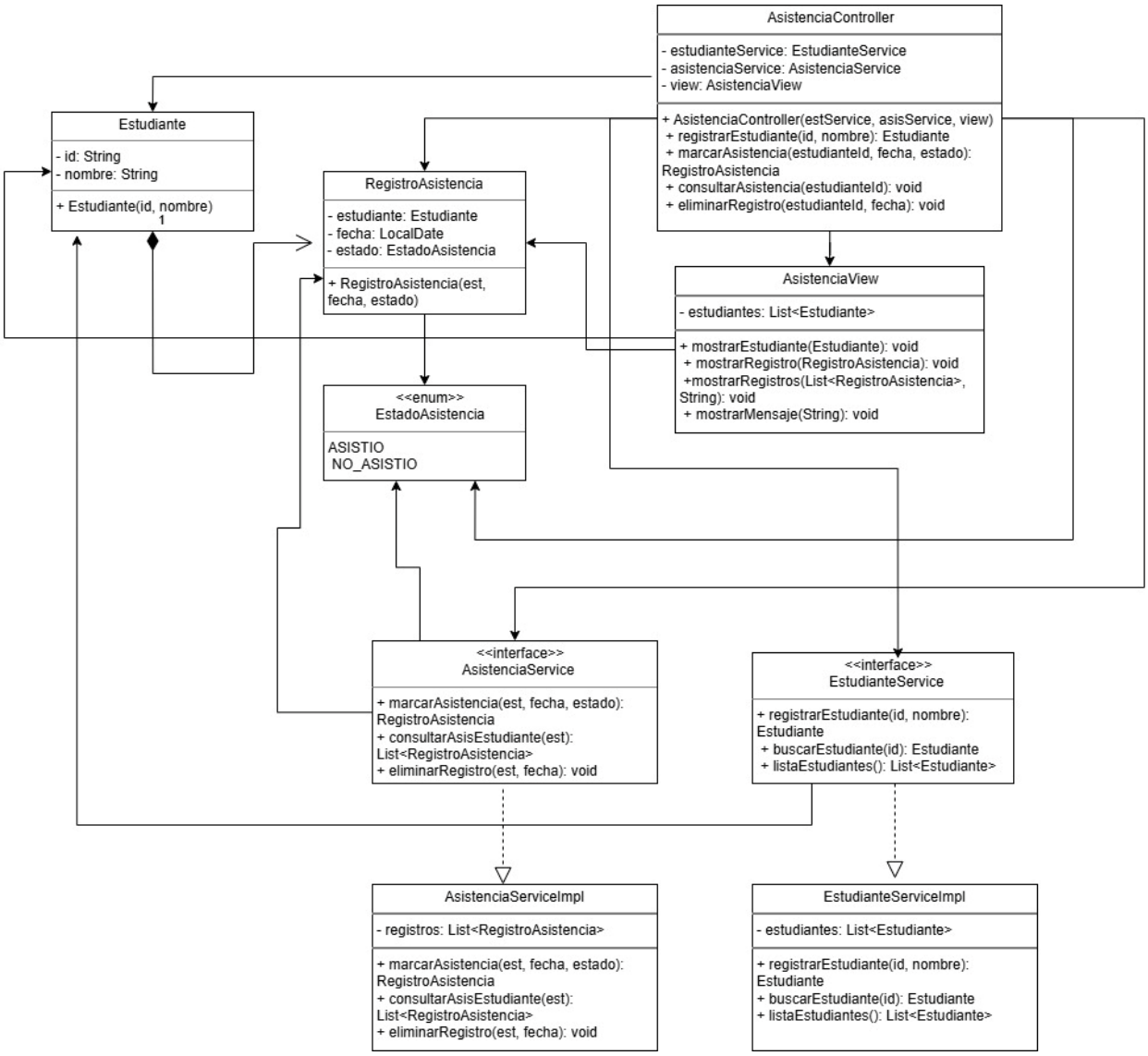
3.Diseño

Arquitectura del Sistema

Se utiliza una arquitectura por capas:

- **Capa de presentación (View)**: Maneja la interacción con el usuario a través de la consola.
- **Capa de control (Controller)**: Recibe las solicitudes del usuario, coordina las operaciones con la capa de servicio y envía respuestas a la vista.
- **Capa de lógica de negocio (Service)**: Contiene las reglas de negocio (validaciones, cálculos) y utiliza las interfaces para abstraer la persistencia.
- **Capa de modelo (Model)**: Representa las entidades del dominio (Estudiante, RegistroAsistencia, EstadoAsistencia).

Diagrama de Clases



Codificación

El sistema de control de asistencia, desarrollado en Java, utiliza una arquitectura por capas que distingue claramente entre la lógica, los datos y la presentación. La capa modelo establece las entidades RegistroAsistencia (que vincula un estudiante, una fecha y un estado, Estudiante (con ID y nombre) y el enumerado EstadoAsistencia (ASISTIO, NO_ASISTIO). La lógica de negocio se encuentra en la capa de servicios, que emplea colecciones en memoria

(ArrayList) para guardar los datos y usa interfaces e implementaciones. La capa de vista tiene la responsabilidad de presentar los mensajes por consola de manera clara y formateada, en tanto que la capa controladora se ocupa de coordinar las operaciones inyectando las dependencias de servicios y vista.

La lógica de negocio abarca validaciones como impedir que un estudiante tenga registros duplicados y que asista a la misma fecha más de una vez. El sistema posibilita la consulta, la eliminación y el registro de asistencias, empleando mensajes descriptivos para manejar las excepciones. La aplicación emplea principios SOLID (como la inversión de dependencias y la responsabilidad única) así como el encapsulamiento, lo cual simplifica que el código sea mantenido y ampliado a futuro.

Link Codigo: https://github.com/Pedro-s19/Sistema-Asistencia