Gestión de Estudiantes

Creador de oportunidades: Juan José Gracia Gómez

Gestor del Conocimiento: William Alexander Matallana Porras

Ingeniería de Software I

501 T

Ingeniería de Sistemas Y computación

Universidad de Cundinamarca

Extensión Chía

2025

FORMATO DE REQUERIMIENTOS DEL SISTEMA
Ingeniería de Software I

1. Información general del proyecto

Nombre del proyecto	Gestión de Estudiantes
Integrantes	Juan José Gracia Gómez
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 información de estudiantes de una institución educativa.
- Se pueden registrar, listar, buscar, modificar y eliminar estudiantes.
- Cada estudiante cuenta con un identificador único, nombre, correo electrónico y programa académico.
- El sistema valida que los datos ingresados sean correctos, incluyendo el formato del correo y que no haya campos vacíos.
- La aplicación está desarrollada en Java, aplicando principios de Programación Orientada a Objetos y siguiendo una arquitectura en capas (Model, View, Controller, Service).

Ciclo de Vida del Desarrollo de Software:

Levantamiento de Requerimientos:

3. Requerimientos Funcionales (RF)

ID	Nombre	Descripción	Entrada(s)	Proceso	Salida
RF-01	Registrar Estudiante	Crear un nuevo estudiante	Nombre, correo, programa	Validar datos, generar Id automático, agregar estudiante a la lista	Estudiante con Id único
RF-02	Listar estudiantes	Mostrar todos los estudiantes	Ninguna	Obtener la lista de estudiantes	Lista completa de estudiantes

		registrados			
RF-03	Buscar estudiante por Id	Encontrar un estudiante específico	Id del estudiante	Recorrer la lista y comparar Id	Estudiante encontrado o mensaje de error
RF-04	Modificar estudiante	Actualizar los datos de un estudiante	Id, nombre, correo, programa	Validar datos, buscar estudiante por Id y actualizar atributos	Mensaje de confirmación de modificación
RF-05	Eliminar estudiante	Borrar un estudiante del sistema	Id del estudiante	Buscar estudiante por Id y eliminar de la lista	Mensaje de confirmación de eliminación
RF-06	Validar datos	Asegurar que los campos no estén vacíos y el correo sea válido	Nombre, correo, programa	Verificar que no sean vacíos y que el correo tenga formato correcto	Mensaje de error si los datos son inválidos

4. Requerimientos No Funcionales (RNF)

ID	Tipo	Descripción
RNF-01	Validación	El sistema no debe permitir campos vacíos ni correos con formato inválido.
RNF-02	Estructura	La aplicación debe estar organizada en paquetes: model, service, controller y view.
RNF-03	Calidad	El sistema debe aplicar encapsulamiento
RNF-04	Rendimiento	El sistema debe procesar registros, búsquedas y modificaciones de manera eficiente.

5. Relación Requerimiento – POO

ID Requerimiento	Clase	Método	Tipo
RF-01	EstudianteServiceImpl	registrarEstudiante()	Funcional
RF-02	EstudianteServiceImpl	listarEstudiantes()	Funcional
RF-03	EstudianteServiceImpl	buscarPorId()	Funcional

RF-04	EstudianteServiceImpl	EstudianteServiceImpl	Funcional
RNF-05	EstudianteServiceImpl	eliminarEstudiante()	No Funcional
RNF-06	EstudianteServiceImpl	validarCampos(), validarCorreo()	Funcinal

Análisis:

Entidades identificadas

- Estudiante (id, nombre, correo, programa)
- ServicioEstudiante (listaEstudiantes, contadorId) – representa la lógica y manejo de estudiantes.
- ControladorEstudiante (service, view) – coordina las acciones entre la vista y el servicio.
- VistaEstudiante (scanner) – maneja la interacción con el usuario.

Relaciones entre entidades

- Un ControladorEstudiante utiliza un ServicioEstudiante para procesar operaciones sobre estudiantes.
- Un ControladorEstudiante utiliza una VistaEstudiante para mostrar información y solicitar datos.
- Un ServicioEstudiante contiene muchos Estudiantes en su lista interna.
- Una VistaEstudiante recibe objetos Estudiante desde el Controller para mostrarlos al usuario.

Diseño:

Arquitectura en capas

Capa de presentación (Controllers):

- EstudianteController – coordina la interacción entre la vista y la lógica.

Capa de negocio (Services):

- EstudianteService – define los métodos para registrar, listar, buscar, modificar y eliminar estudiantes.
- EstudianteServiceImpl – implementación de la lógica, manejo de listas y validaciones.

Capa de presentación de datos (View):

- EstudianteView – muestra menús, solicita datos al usuario y despliega información en consola**.**

Tecnologías utilizadas

- Lenguaje de programación: Java
- IDE sugerido: IntelliJ IDEA
- Gestión de dependencias: Maven
- Entrada/Salida: Consola
- Arquitectura: MVC (Modelo-Vista-Controlador) y capa de servicio (Service)

Codificación:

En esta etapa se desarrolló el sistema de gestión de estudiantes siguiendo la arquitectura en capas definida durante el análisis y diseño, aplicando principios de Programación Orientada a Objetos. Se implementó la clase Estudiante como modelo de datos, con atributos privados y métodos get y set para mantener el encapsulamiento. La lógica del código se desarrolló en EstudianteService y su implementación EstudianteServiceImpl, donde se manejan operaciones como registrar, listar, buscar, modificar y eliminar estudiantes, incluyendo la generación automática de IDs y la validación de campos y correos electrónicos mediante expresiones regulares. La comunicación con el usuario se gestionó a través de EstudianteView, que muestra menús, solicita datos y despliega información, mientras que EstudianteController coordina la interacción entre la View y el Service, asegurando que las operaciones se realicen correctamente sin que la vista o el servicio manejen responsabilidades ajenas. Además, se implementó manejo de errores con try-catch para capturar excepciones y mostrar mensajes claros al usuario, evitando que el programa se detenga abruptamente. El Main inicializa los objetos necesarios y controla el flujo principal del programa mediante un menú en consola, garantizando que cada acción se ejecute de manera ordenada y con datos validados.

Link del Repositorio: https://github.com/JuanJGr/GestionEstudiantes.git

Diagrama de Clases:

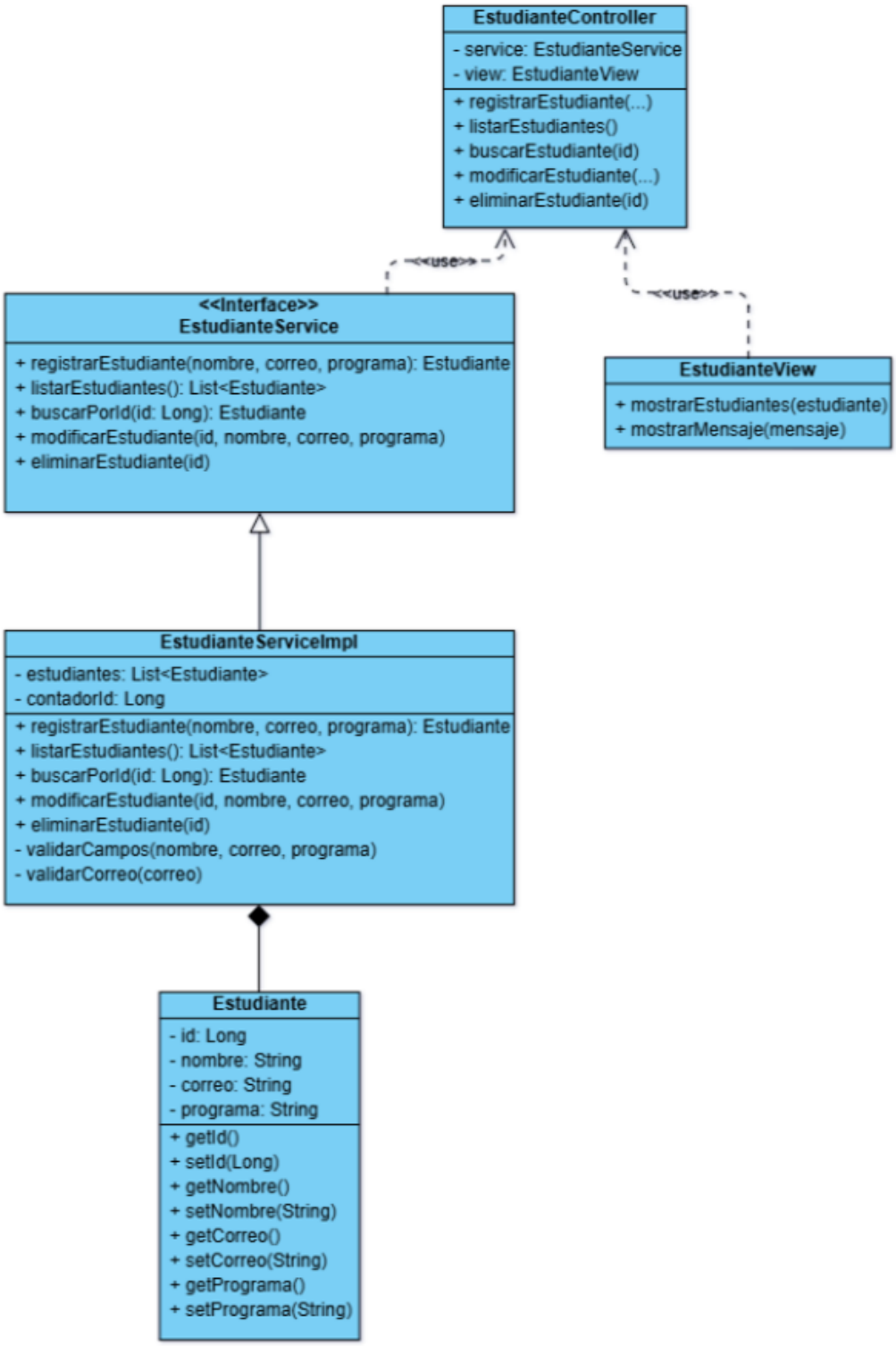

Diagrama de Casos de Uso: