Taller: Creación de un CRUD (ID, Nombres, Edad) aplicando Arquitectura de 3 Capas y Patrón Modelo–Vista–Controlador (MVC)

1. Objetivo del Taller

Construir una aplicación CRUD de Estudiante (ID, nombres, edad) utilizando la arquitectura de 3 capas y el patrón de diseño Modelo–Vista–Controlador (MVC) con una interfaz gráfica en Java Swing.

2. Estructura del Proyecto

La aplicación deberá organizarse en la siguiente estructura de paquetes:

```
src/main/java/ec/edu/espe/
├── datos/
│   ├── model/
│   │   └── Estudiante.java
│   └── repository/
│       └── EstudianteRepository.java
├── logica_negocio/
│   └── EstudianteService.java
└── presentacion/
    ├── EstudianteUI.java
    └── Main.java
```

3. Descripción de las Capas

3.1 Capa Modelo (Model)

Contiene la clase Estudiante.java, que representa el dominio con sus atributos: ID, nombres y edad. No contiene lógica de negocio ni acceso a datos.

3.2 Capa de Acceso a Datos (Repository)

EstudianteRepository.java administra las operaciones CRUD utilizando una colección interna (ArrayList) o almacenamiento simple en archivo. Opcionalmente puede implementarse como Singleton.

3.3 Capa Lógica de Negocio (Service)

EstudianteService.java aplica reglas de negocio como validaciones (ej., edad > 0, ID no repetido) y delega las operaciones CRUD al repositorio.

3.4 Capa Presentación – Swing (Vista y Controlador)

EstudianteUI.java representa la interfaz gráfica del usuario. Incluye:
- Campos de texto para ID, nombres y edad.
- Botones para las acciones CRUD.

- Una JTable que actúa como 'grid' para mostrar los estudiantes.
La UI envía las solicitudes al Service y actualiza la vista según resultados.

4. Flujo de Interacción MVC + 3 Capas

1. El usuario interactúa con EstudianteUI.
2. La UI llama a EstudianteService.
3. El Service valida datos y llama a EstudianteRepository.
4. El Repository ejecuta las operaciones CRUD y devuelve resultados.
5. La UI actualiza el formulario y el grid.

5. Actividades del Taller

Actividad 1: Crear el paquete datos/model y la clase Estudiante.java con:

- Atributos: id, nombres, edad
- Constructor, getters y setters.

Actividad 2: Implementar datos/repository con EstudianteRepository.java:

- Crear una lista interna de estudiantes.
- Implementar: agregar(), editar(), eliminar(), listar().

Actividad 3: Crear logica_negocio/EstudianteService.java:

- Validar reglas del negocio.
- Delegar todas las operaciones al Repository.

Actividad 4: Crear presentacion/EstudianteUI.java en Swing:

- Construir el formulario (ID, nombres, edad).
- Implementar botón Guardar → llama al Service.
- Actualizar la JTable con los datos.

6. Rúbrica de Evaluación (20 puntos)

Criterio	Descripción	Puntaje
Estructura del proyecto	Implementa correctamente la arquitectura de 3 capas.	0–5
Modelo, Servicio y Repositorio	Clases correctamente implementadas con separación de responsabilidades.	0–5
Interfaz gráfica Swing	Formulario funcional y tabla desplegando datos.	0–5

Aplicación del patrón MVC	La vista no contiene lógica de negocio ni acceso a datos.	0–5