**DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN**

**EXAMEN DE: ANALISIS Y DISEÑO DE SOFTWARE**

**FORMA A TEORIA Y PRACTICA**



**PERÍODO:** abril 2025 – agosto 2025 **PARCIAL:** 2do Parcial\_\_\_\_\_

**NOMBRE:** Diego Casignia **CURSO (NRC):**  22426

**CARRERA:** Ingenieria en Software **FECHA:** 03 /07/ 2025\_\_\_\_\_

| **INDICACIONES GENERALES:**   * La evaluación es personal y no se permite intercambiar información con sus compañeros, si es llamado la atención por una vez, se procederá a ponerle CERO y no recibirle su evaluación. * La parte práctica debe subirle En la carpeta de GitHub y formato indicado por el Sr. Docente (Apellidos Nombres\_U2\_Examen), * **Dispone de 2h.** |
| --- |

1. **Parte Práctica (20 Puntos)**
2. **Realizar en un IDE de Java lo siguiente.**

***Plantilla CRUD - Estudiantes (Arquitectura 3 Capas + MVC)***

Esta plantilla provee una estructura básica para implementar un CRUD de la entidad Estudiante, utilizando arquitectura en 3 capas y modelo MVC.

Está pensada para estudiantes con conocimientos en programación orientada a objetos en Java.

***1. Estructura de carpetas recomendada***

/model  
 Contiene la clase Estudiante con sus atributos y métodos.  
  
/dao  
 Contiene la clase EstudianteDAO responsable del almacenamiento y recuperación de datos.  
  
/controller  
 Contiene la clase EstudianteController que coordina la lógica.  
  
/ui  
 Contiene la clase Main que actúa como la interfaz de usuario (vista).

***2. Modelo (Estudiante)***

package model;  
  
public class Estudiante {  
 private int id;  
 private String apellidos;  
 private String nombres;  
 private int edad;  
  
 public Estudiante(int id, String apellidos, String nombres, int edad) {  
 this.id = id;  
 this.apellidos = apellidos;  
 this.nombres = nombres;  
 this.edad = edad;  
 }  
  
 // Getters y Setters  
 public int getId() { return id; }  
 public void setId(int id) { this.id = id; }  
 public String getApellidos() { return apellidos; }  
 public void setApellidos(String apellidos) { this.apellidos = apellidos; }  
 public String getNombres() { return nombres; }  
 public void setNombres(String nombres) { this.nombres = nombres; }  
 public int getEdad() { return edad; }  
 public void setEdad(int edad) { this.edad = edad; }  
}

***3. DAO (EstudianteDAO)***

package dao;  
  
import java.util.ArrayList;  
import java.util.List;  
import model.Estudiante;  
  
public class EstudianteDAO {  
 private List<Estudiante> estudiantes = new ArrayList<>();  
  
 public void agregar(Estudiante e) {  
 estudiantes.add(e);  
 }  
  
 public List<Estudiante> listar() {  
 return estudiantes;  
 }  
  
 public Estudiante buscarPorId(int id) {  
 for (Estudiante e : estudiantes) {  
 if (e.getId() == id) {  
 return e;  
 }  
 }  
 return null;  
 }  
}

***4. Controlador (EstudianteController)***

package controller;  
  
import dao.EstudianteDAO;  
import model.Estudiante;  
import java.util.List;  
  
public class EstudianteController {  
 private EstudianteDAO dao = new EstudianteDAO();  
  
 public void crearEstudiante(int id, String apellidos, String nombres, int edad) {  
 Estudiante e = new Estudiante(id, apellidos, nombres, edad);  
 dao.agregar(e);  
 }  
  
 public List<Estudiante> obtenerTodos() {  
 return dao.listar();  
 }  
  
 public Estudiante buscarEstudiante(int id) {  
 return dao.buscarPorId(id);  
 }  
}

***5. Vista (Main)***

package ui;  
  
import controller.EstudianteController;  
import model.Estudiante;  
  
public class Main {  
 public static void main(String[] args) {  
 EstudianteController controller = new EstudianteController();  
   
 controller.crearEstudiante(1, "Pérez", "Ana", 20);  
 controller.crearEstudiante(2, "García", "Luis", 22);  
   
 for (Estudiante e : controller.obtenerTodos()) {  
 System.out.println(e.getId() + " - " + e.getApellidos() + " " + e.getNombres() + " - Edad: " + e.getEdad());  
 }  
 }  
}

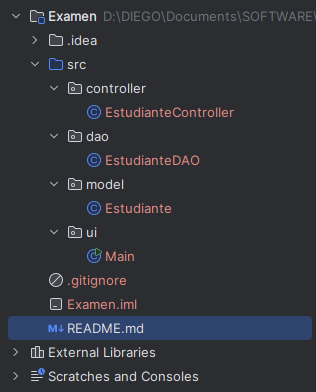
***6. Recomendaciones de implementación***

- Mantén cada clase en su paquete correspondiente.  
- Usa nombres de métodos claros y consistentes.  
- Puedes ampliar el DAO para soportar edición y eliminación.  
- Recuerda que esta plantilla es la base.

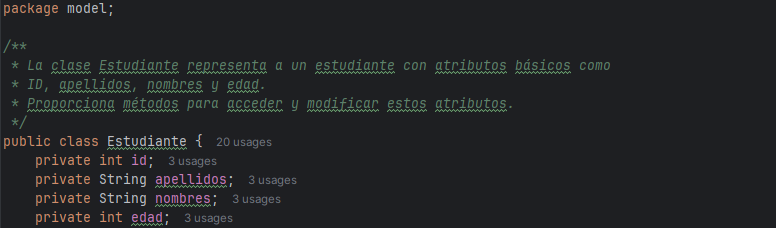
***Rúbrica de Evaluación de la Práctica - Arquitectura 3 Capas y MVC***

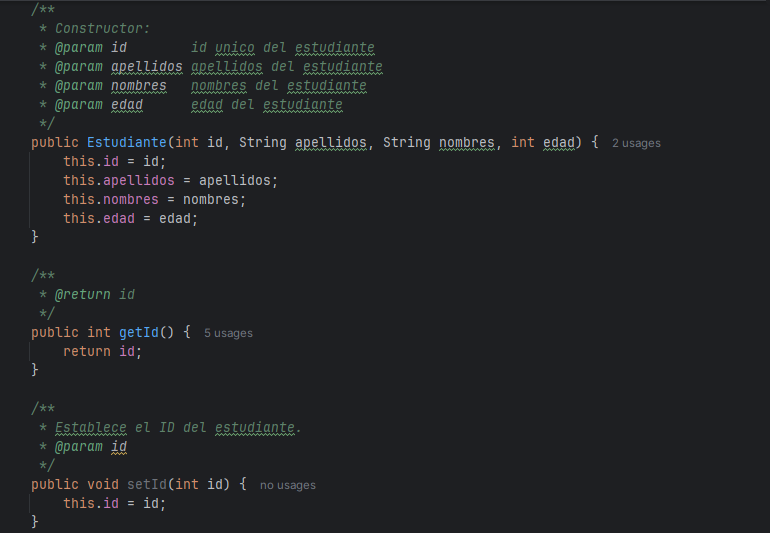
| Criterio de Evaluación | Puntaje Máximo |
| --- | --- |
| 1. Presenta correctamente la estructura de carpetas según la arquitectura 3 capas | 4 puntos |
| 1. Implementa de forma correcta el Modelo (Estudiante) con atributos y métodos | 4 puntos |
| 1. Aplica el patrón MVC separando claramente la lógica en Controller y DAO | 4 puntos |
| 1. La Vista muestra los datos de forma clara y ordenada | 4 puntos |
| 1. El código está bien documentado y se ejecuta sin errores | 4 puntos |
| TOTAL SOBRE 20 |  |

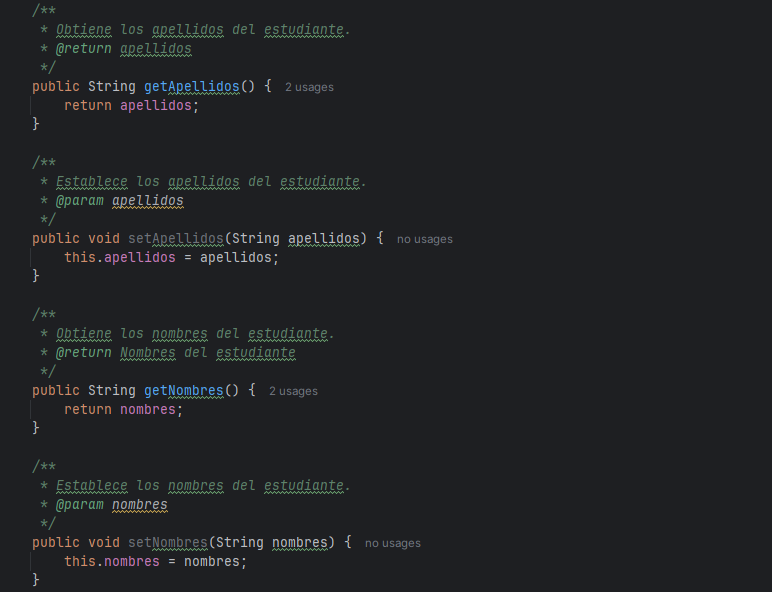
1. Inerte una captura de pantalla sobre: la estructura de carpetas según la arquitectura 3 capas

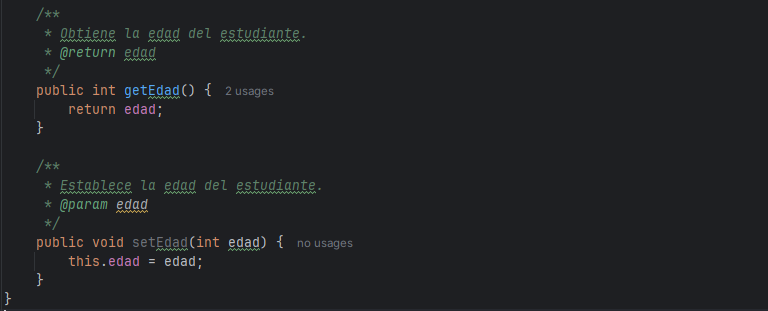


1. Implementa de forma correcta el Modelo (Estudiante) con atributos y métodos



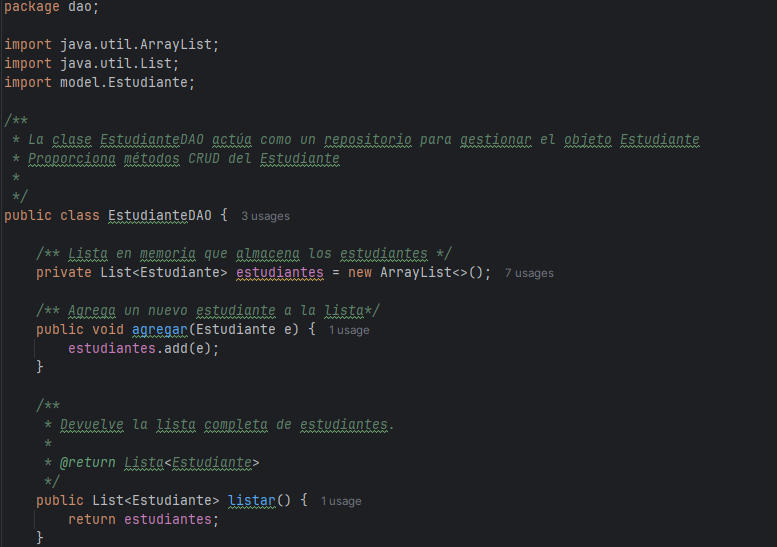


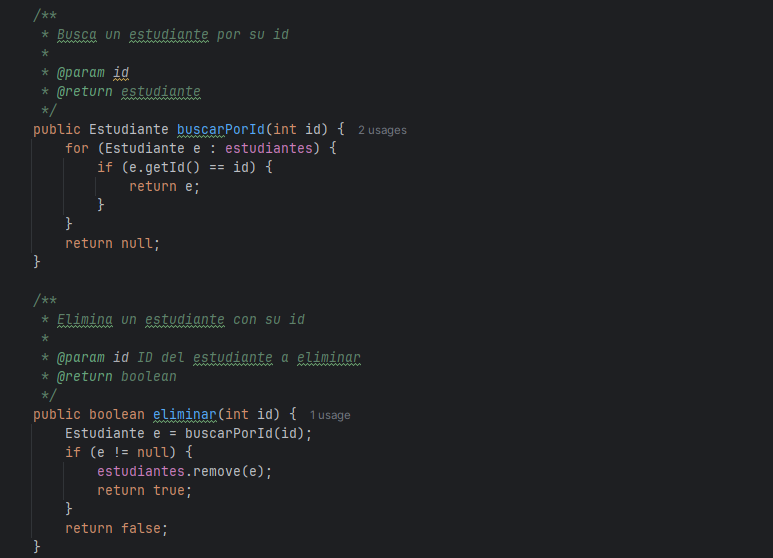


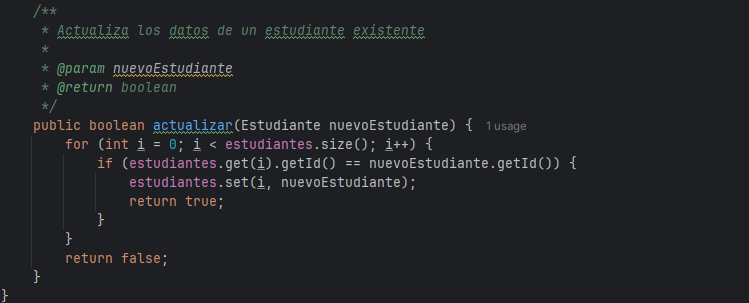


1. Aplica el patrón MVC separando claramente la lógica en Controller y DAO

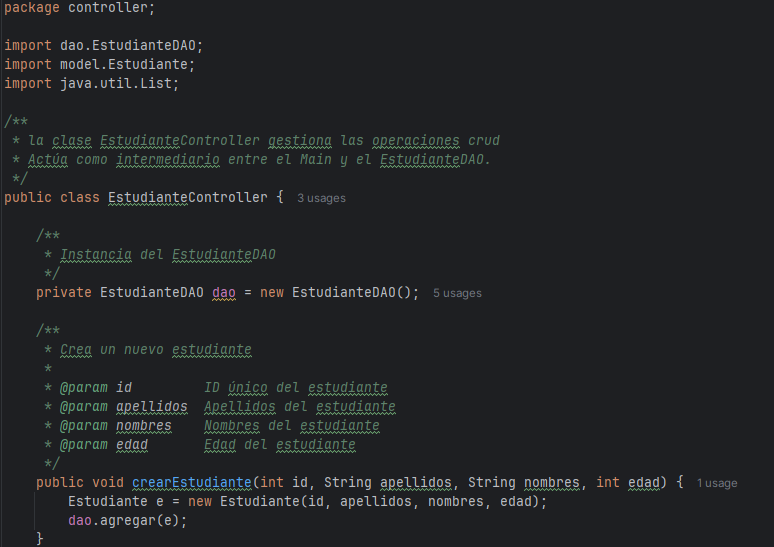
EstudianteDAO:

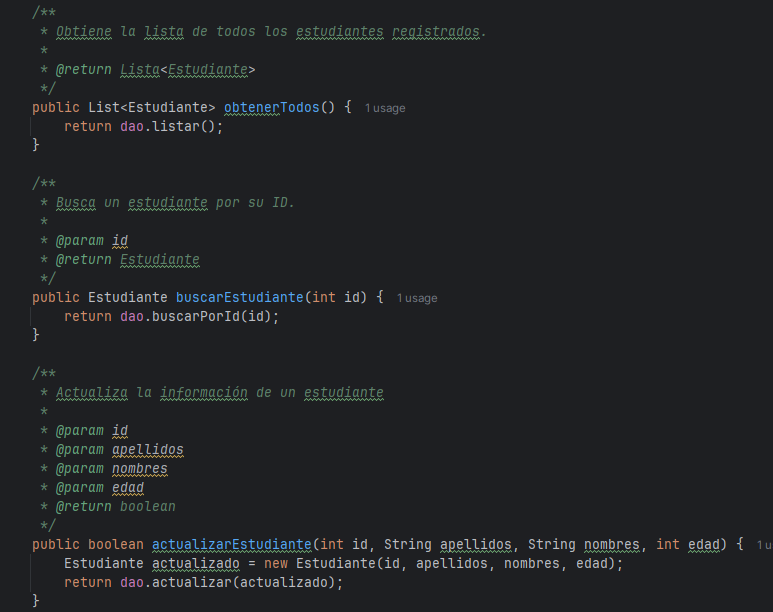


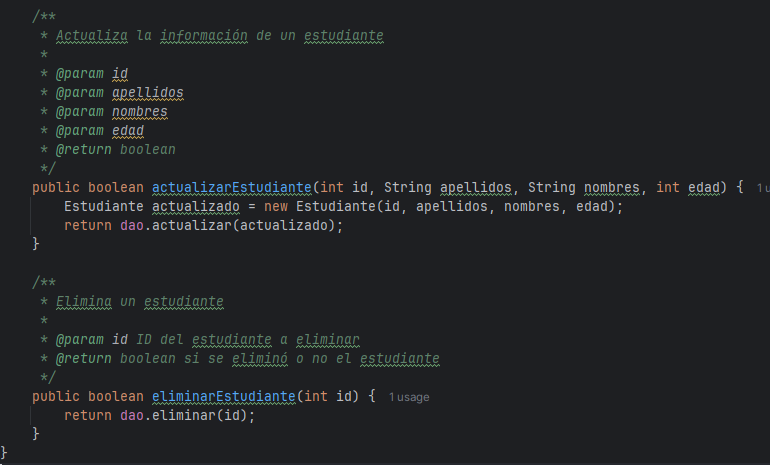




EstudianteController:







1. La Vista muestra los datos de forma clara y ordenada

