Modelado Orientado a Objetos con UML: Análisis de un Sistema Académico

Trabajo presentado como requisito de la actividad #1: Objetos, Atributos y Métodos

Dilan Stiven Muñoz Sanchez
Diego Miranda

Universidad Sergio Arboleda
Facultad de Ingeniería
Fundamentos de Diseño de Software
19 de febrero de 2026

1 Contexto del Problema

1.1 Selección del Sistema

Hemos elegido trabajar con un Sistema de Gestión Académica (como el que usa la universidad para ver notas).

2 Parte 1: Análisis del Problema

2.1 1. Propósito del Sistema

El objetivo principal de este sistema es poner orden en la universidad. Sirve para que la información de los estudiantes, las clases y las notas esté en un solo lugar y no regada en papeles. Básicamente, busca que todo el proceso de estudiar y calificar sea más rápido y fácil para todos.

2.2 2. ¿Cómo funciona el sistema? (Paso a paso)

El sistema funciona siguiendo estos pasos sencillos:

1. **Inicio:** La universidad abre los cursos y define qué profesores van a dictar clase.
2. **Inscripción:** Los estudiantes entran al sistema, ven qué materias hay y las inscriben.
3. **Clases:** Durante el semestre, el profesor revisa en el sistema quiénes son sus alumnos.
4. **Calificación:** Cuando se acaba un corte, el profesor sube la nota al sistema.
5. **Resultado:** El estudiante entra a ver su nota y el sistema le dice si pasó o perdió la materia.

2.3 3. ¿Quiénes y qué datos operan en el sistema?

Para que esto funcione, necesitamos estas partes:
Personas (Usuarios):

- **Estudiantes:** Son los que inscriben materias y revisan sus notas.
- **Docentes:** Son los que dictan la clase y ponen las notas.

Datos importantes:

- **Asignaturas:** La información de cada materia (nombre, créditos).
- **Notas:** Los números que dicen cuánto sacó el estudiante.

2.4 4. ¿Qué problema del mundo real resuelve?

Este sistema soluciona el problema del desorden y la lentitud.

- Evita que se pierdan las notas en papeles físicos.
- Evita que los estudiantes tengan que ir hasta una oficina solo para preguntar cuánto sacaron.
- Ayuda a que los profesores no tengan que sumar notas a mano (y equivocarse).

3 Parte 2: Identificación de Clases y Atributos

3.1 3. Definición de Clases

Pensando en los objetos que mencionamos arriba, definimos estas clases:

- Clase `Estudiante`
- Clase `Docente`
- Clase `Asignatura`
- Clase `Nota`

3.2 4. Atributos de las Clases

Aquí describimos qué características tiene cada uno:
Clase: Estudiante

- `nombre`: String (Cómo se llama)
- `id`: int (Su número de identificación)
- `carrera`: String (Qué está estudiando)
- `semestreActual`: int (En qué semestre va)
- `correoInstitucional`: String (Su email para contacto)

Clase: Docente

- `nombre`: String
- `id`: int
- `departamento`: String (A qué área pertenece)
- `especialidad`: String (Qué sabe enseñar)

Clase: Asignatura

- `nombre`: String
- `codigo`: String (El código único de la materia)
- `creditos`: int (Cuánto vale la materia)
- `contenidoTematico`: String (Qué temas se ven)
- `prerrequisito`: String (Qué materia debió ver antes)

Clase: Nota

- `valor`: double (La nota, ej: 4.5)
- `corteAcademico`: int (Si es primer, segundo o tercer corte)
- `fechaRegistro`: Date
- `estadoAprobacion`: boolean (True si pasó, False si perdió)

4 Parte 3: Identificación de Comportamientos (Métodos)

4.1 5. Métodos de las Clases

Estas son las acciones que puede hacer cada uno:

Clase: Estudiante

- `inscribirMateria()`: boolean (Meter materias al horario)
- `cancelarMateria()`: boolean (Sacar una materia)
- `consultarHorario()`: String (Ver cuándo tiene clase)
- `descargarCertificado()`: String (Bajar un papel de notas)

Clase: Docente

- `asignarNota()`: void (Ponerle nota a un alumno)
- `registrarAsistencia()`: boolean (Llamar a lista)
- `consultarListadoEstudiantes()`: String (Ver quiénes están en su clase)
- `actualizarContenidoProgramatico()`: void (Cambiar temas de la clase)

Clase: Asignatura

- `obtenerSilabo()`: String (Ver el plan de la materia)

- `verificarCuposDisponibles()`: int (Ver si cabe más gente)
- `listarHorarios()`: String (Ver a qué hora se dicta)
- `validarPrerrequisitos()`: boolean (Chequear si se puede ver la materia)

Clase: Nota

- `calcularPromedio()`: double (Sacar la definitiva)
- `modificarValor()`: boolean (Cambiar la nota si hubo error)
- `publicar()`: void (Hacer la nota visible)
- `obtenerEstado()`: String (Saber si aprobó o no)

4.2 6. Semantica de relaciones

1. **Relación de asociación en las clases de Estudiante – Asignatura**
   Existe una relación ya que un estudiante puede incribir muchas asignaturas y una asignatura puede contener multiples estudiantes.
   Ya que ambos pueden existir, la asignatura sin estudiante como el estudiante sin asignatura.

2. **Relación de composición en las clases de Estudiante – Nota**
   La Nota es aquella que representa la calidad académica del estudiante.
   Ya solo exista mientras el estudiante exista

3. **Relación de asociación en las clases de Docente – Asignatura**
   El docente puede dictar varias asignaturas y una misma asignatura puede ser dictada por varios profesores.

4. **Relación de dependencia en las clases de Docente – Nota**
   El docente usa a Nota para el registro y visualización de las calificaciones.

5. **Relación de dependencia en las clases de Asignatura – Nota**
   Ya que toda Nota se encuentra dentro de una asignatura.