



Programación Orientada a Objetos
Matemáticas Aplicadas y Computación
Profesor: Martínez Castro Pablo



UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
FACULTAD DE ESTUDIOS SUPERIORES ACATLÁN

Proyecto Final.

Manual de usuario para el Sistema de Gestión de Cursos.

Profesor: Martínez Castro Pablo.

Grupo: 1303.

Carrera: Matemáticas Aplicadas y Computación

Integrantes del equipo:

- Gil de Gaona Jazmín
- Hernández Peña Angel Adrian
- Moreno Viguera Tadeo Arturo.
- Ruiz Garcia Emiliano.

Equipo: **Plaga**

Fecha de entrega: 04 de diciembre del 2024.

Índice

Introducción.....	3
Objetivo.....	4
Requisitos del sistema.....	4
Estructura del programa.....	5
Funcionalidades del sistema.....	6
Extensión y Mantenimiento del Sistema.....	7
Conocimientos adquiridos.....	8
Conclusiones.....	11

Introducción

El presente proyecto, titulado *Sistema de Gestión de Cursos*, fue desarrollado como parte del curso de Programación Orientada a Objetos (POO) en el semestre 2025-I, clase impartida por el profesor Martínez Castro Pablo para la carrera de Matemáticas Aplicadas y Computación de la Universidad Nacional Autónoma de México. Este sistema busca integrar los conceptos teóricos y prácticos de la programación orientada a objetos, ofreciendo una solución funcional y escalable que a un sistema que simula un entorno educativo digital.

Propósito del sistema

Vivimos en un mundo donde la educación digital está en constante crecimiento, este sistema pretende ofrecer un marco básico para la gestión de usuarios, cursos, evaluaciones y asesorías. Inspirado en plataformas educativas modernas tales como Google Classroom, Zoom, Moodle, Khan Academy entre muchas otras, este proyecto tiene como objetivo ser una herramienta interactiva y práctica para estudiantes, tutores y administradores.

Objetivo

El sistema de gestión de cursos y evaluaciones está diseñado para facilitar el aprendizaje en un entorno digital, permitiendo a los usuarios:

- Administrar cursos con materias, profesores y evaluaciones.
- Realizar evaluaciones personalizadas y predeterminadas.
- Continuar el progreso en actividades a través de un historial detallado.
- Mejorar suscripciones para acceder a servicios adicionales.
- Tomar asesorías en áreas específicas con tutores asignados.

Este programa está orientado tanto a estudiantes como a profesores y administradores, creando un ecosistema educativo digital eficiente y accesible.

Requisitos del sistema

Software necesario:

- Kit de desarrollo Java (JDK): Versión 8 o superior.
- NetBeans IDE: Versión 17.0 o superior.*
- Sistema Operativo: Windows, macOS o Linux.

Hardware mínimo:

- Procesador: Intel i3 de 9na generación o equivalente.
- Memoria RAM: 4 GB.
- Espacio en disco: 100 MB libres.

Como nota aclaratoria, el lenguaje de programación Java sigue en constante actualización igual que el IDE NetBeans, es probable que en unos años el sistema pueda quedar descontinuado por funciones que quedarán obsoletas al cabo de años, en ese caso se recomienda contactar con la persona que le ha proporcionado este manual para solicitar la versión más actualizada.

Acerca de POO

La programación orientada a objetos es un paradigma de programación que organiza cualquier sistema en torno a los **objetos**, en lugar de funciones o procedimientos. Cada uno de estos objetos cumplen el propósito principal de la materia el cual es abstraer entidades del mundo real en un software computacional. Adicionalmente, POO facilita el diseño de software para volverlo más comprensible, modular, reutilizable y mantenible.

Aunado a esto, el proporcionar una interfaz gráfica de usuario o GUI por sus siglas en inglés permiten que el usuario pueda manipular y usar el programa de una manera más efectiva y sin tantas complicaciones.

Temas abordados

El proyecto está diseñado para consolidar los conocimientos adquiridos en POO, incluyendo conceptos como:

- **Clases y Objetos:** Cada clase creada en este sistema es un molde o plantilla que cuenta con propiedades, también llamados atributos, y comportamientos, también conocidos como métodos, todo ello para permitir que cualquier objeto funcione correctamente y cumpla el propósito general del programa.
- **Herencia:** Reutilización de propiedades y comportamientos, esto se logra al compartir o heredar los atributos y métodos entre clases base e hijas. Dentro del lenguaje este concepto se ve claro con la palabra reservada “extends”.
- **Encapsulamiento:** Este concepto se basa en proteger los datos y control del acceso mediante métodos específicos, esto mejora la seguridad y reduce la complejidad del algoritmo. Nos podemos dar cuenta de este concepto gracias a las palabras reservadas “private”, “public” y “protected”.
- **Polimorfismo:** Flexibilidad en el uso de clases y métodos, promoviendo la reutilización del código, ya que una misma acción puede comportarse de forma diferente dependiendo de la firma que se le comparte. Dentro del programa lo podremos identificar con la palabra “Override”.
- **Abstracción:** Simplificación del diseño del sistema al enfocarse en los elementos esenciales, gracias a estos podemos enfocarlos en los conceptos específicos por clase. Esto se puede observar dentro del programa gracias a la palabra clave “Interface”.
- **Uso de genéricos:** Los genéricos permiten definir clases y métodos que funcionan con cualquier tipo de dato, proporcionando flexibilidad y reutilización.
 - Aplicación en el programa:
 - La clase Progreso<T> utiliza genéricos para manejar diferentes tipos de actividades (cursos, evaluaciones, etc).

- Esto permite reutilizar la misma clase para rastrear el progreso en diferentes tipos de actividades.
- **Modularidad:** Consiste en dividir el programa en componentes independientes que trabajan juntos para cumplir el objetivo general.
 - Aplicación en el programa:
 - Cada clase se enfoca en una responsabilidad específica.
 - Por ejemplo, la clase Curso administra la información relacionada con un curso, mientras que Evaluación gestiona las evaluaciones asociadas.
- **Manejo de colecciones:** Uso de estructuras como listas para almacenar y manipular conjuntos de datos. Dentro del programa se utilizan ArrayList y Listas para almacenar listas dinámicas de usuarios, cursos y evaluaciones.

Adicional a estos conceptos “generales” de la POO, a lo largo del curso se tocaron varios temas que fueron de utilidad para el desarrollo del programa, tales como los diccionarios, en nuestro caso utilizamos una base de datos simple que almacena la información del usuario en formato JavaScript Object Notation (JSON), esto con el fin de facilitar la búsqueda y acceso a los datos.

Igualmente, podemos notar temas como sobrecarga de operadores al estar comparando los datos en la base de datos con el fin de iniciar sesión al sistema o bien, el tema de paso por valor y referencia para trabajar con la manipulación de objetos y variables que podemos notar en cada método.

El desarrollo de este programa no solo permite practicar estos conceptos, sino también entender su relevancia en proyectos reales y futuros, además de comprender la dificultad de los programas actuales.

Este proyecto representa un punto de partida para el desarrollo de sistemas educativos más complejos. Al explorar este sistema, tendrá la oportunidad de comprender cómo los conceptos de POO se aplican en la creación de aplicaciones útiles y funcionales.

Instalación

Paso 1: Descargar NetBeans.

- Descargue e instale NetBeans IDE. Esto se puede hacer desde el sitio oficial: <https://netbeans.apache.org/front/main/index.html>

En teoría, el programa está construido para que funcione con cualquier compilador de Java con gráficos tal como Eclipse o Visual Studio Code, pero todo el sistema se ha desarrollado en esta aplicación y por ende se ha comprobado su funcionamiento.

- Adicional a ello, se le solicita instalar y tener actualizado Java JDK si es que no viene por defecto en su sistema, esto lo puede encontrar en el sitio oficial de Java: <https://www.oracle.com/java/technologies/downloads/>

Paso 2: Configurar el proyecto.

1. Abra NetBeans y seleccione Archivo > Nuevo proyecto.
2. Elige Maven > Aplicación Java y haz clic en Siguiente. {
3. Asigna un nombre al proyecto (por ejemplo, SistemaCursos) y seleccione la ubicación donde guardar el proyecto.
4. Asegúrese de que Create Main Class esté desmarcado y haga clic en Finish.

Paso 3: Importar código.

1. Abra el enlace donde se encuentra este documento, es decir: <https://github.com/SinR0str0/Proyecto-Plaga>.
2. Descargue el archivo dando clic en Code>Download ZIP.
3. Extraiga los archivos en la carpeta creada en el paso 2.
4. En NetBeans, haga clic derecho en el proyecto y seleccione Clean and Build para compilar el código.

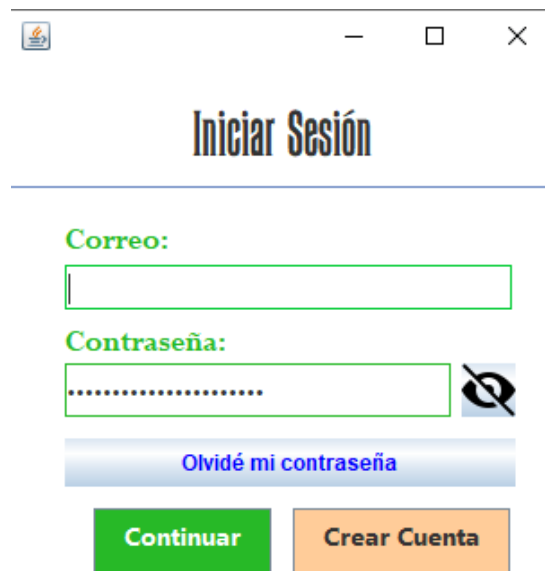
Paso 4: Ejecutar el proyecto.

1. Haga clic derecho en la clase Main y seleccione Ejecutar archivo, el cual es el botón de play de color verde.
2. La aplicación se ejecutará en la consola integrada.

Conoce tu entorno

Login


Una vez ejecutado el programa saldrá la siguiente pantalla de Login o acceso al sistema.



Iniciar Sesión

Correo:

Contraseña:



[Olvidé mi contraseña](#)

Continuar **Crear Cuenta**

Si ya ha usado el sistema anteriormente es probable que ya cuente con un usuario creado, en ese caso solo deberá ingresar con su correo electrónico y contraseña y presionar el botón “Continuar”. Por otra parte, si no recuerda su contraseña, puede consultarla al escribir su correo electrónico usado y presionando el botón “Olvidé mi contraseña”, hacer esto hará pública su contraseña en la parte inferior de la ventana para que pueda copiarla en la casilla contraseña.

Sign-Up

Por último, si es la primera vez usando el programa, deberá crear una cuenta en el botón “Crear Cuenta”. Una vez presionado saldrá la siguiente ventana:

Registro

Nombre:

Contraseña:

Correo:

Fecha de Nacimiento:

noviembre	▼	2006	▼
d...	lun	m...	mié jue vie s...
44			1 2 3 4
45	5	6 7	8 9 10 11
46	12	13 14	15 16 17 18
47	19	20 21	22 23 24 25
48	26	27 28	29 30

Sexo:

Grado:

En ella, se le solicitará algunos datos básicos de registro tal como nombre, contraseña, correo electrónico, fecha de nacimiento, sexo y el tipo de usuario que se registra (Profesor o alumno), todos los datos ingresados deben ser correctos, como una fecha de nacimiento aceptable y un nombre completo.

En caso de que se registre como profesor (para agregar, editar y eliminar sus cursos) también se le pedirá que seleccione el área de estudios, para ello nos basamos en las 4 áreas de conocimientos de la UNAM, consulte aquí para más información: <https://www.orienta.unam.mx/UNAMORIENTA/pages/carreras.html>.

Una vez registrado, sus datos se guardarán en nuestra base de datos para futuras consultas.

Cabe aclarar que por el tipo de programa (y dado que no es relevante para la materia de POO), hemos reducido este sistema para evitar conectarse a internet, por lo cual la base de datos solo estará disponible por computadora.

Cuando el sistema valide sus credenciales de acceso (Login) mostrará una pantalla como la siguiente, donde podrá “tomar” las clases que los profesores han preparado para los estudiantes.

Menú Principal



En el centro de esta se encontrarán varios botones (dependerá de los temas cargados) que serán los temas individuales de cada área.

Del lado izquierdo podrá ver un breve menú de opciones donde tendrá la opción de seleccionar las diversas áreas del conocimiento, al presionar cada área la lista de botones central de los temas se actualizará.

Configuración

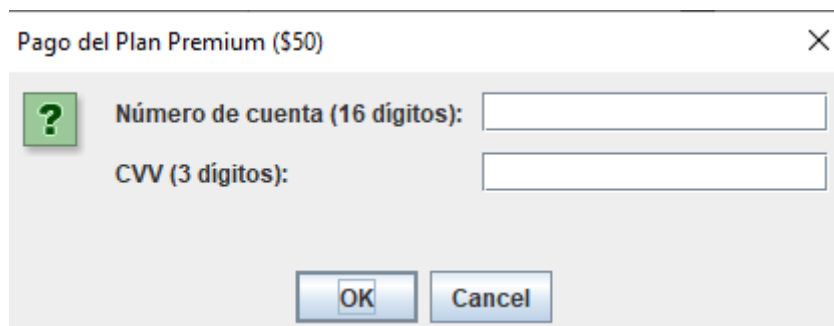
Adicionalmente, tendrá el botón de configuración, en donde podrá corregir alguno de sus datos de registro como correo electrónico, contraseña o mejorar su suscripción con la siguiente interfaz:



The screenshot shows a window titled "Configuración" with the "Volver" logo. It contains the following fields and controls:

- Nombre:** Text field with "yo perez".
- Correo electrónico:** Text field with "YO@GMAIL.COM" and an edit icon.
- Contraseña:** Text field with ".." and an edit icon.
- Rango:** Text field with "Tutor".
- Suscripción:** Text field with "Tutor" and an edit icon.
- Guardar Cambios:** A yellow button at the bottom.

Cada que se edite un parámetro será necesario presionar el botón "Guardar Cambios" para que la base de datos se actualice correctamente. En caso de que se desee actualizar la suscripción, se deberá seleccionar el plan (Básico, Medio o Premium), si se eligen los planes con costo deberá ingresar un número de cuenta de 16 dígitos y un CVV válido para proceder con la compra y actualización del servicio, si la información es válida, el nivel será actualizado.

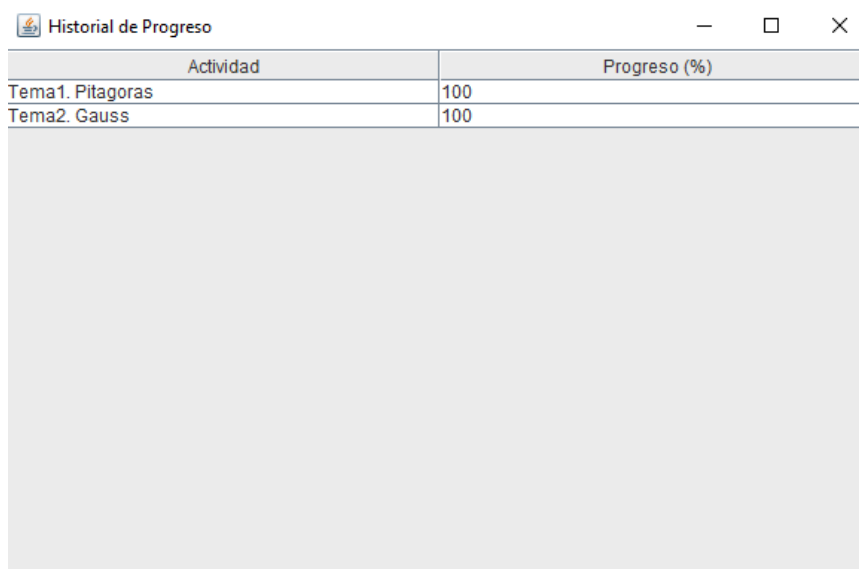


The screenshot shows a dialog box titled "Pago del Plan Premium (\$50)". It contains the following fields and controls:

- Número de cuenta (16 dígitos):** Text field with a green question mark icon.
- CVV (3 dígitos):** Text field.
- OK:** Button.
- Cancel:** Button.

Historial

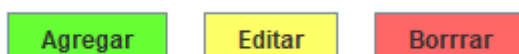
Volviendo al menú principal, tendremos el botón “Historial”, donde se abrirá una ventana que contiene todos los cursos iniciados por el usuario y su nivel de progreso.



Actividad	Progreso (%)
Tema1. Pitagoras	100
Tema2. Gauss	100

ABC de temas

Además, los profesores y administradores tendrán 3 botones extras con los cuales podrán editar, agregar o eliminar temas según su área del conocimiento (Para administradores es indistinta el área).



Si se elige **Agregar**, se le pedirá al profesor que seleccione un archivo HTML para mostrar, una vez seleccionado se agrega el tema a la lista de botones, dentro del sistema la base de datos con el área seleccionada se actualiza con una nueva carpeta que contiene los datos principales del cursos tales como el correo del creador, el título del curso, una breve descripción y su índice.

El botón **Editar** le permite agregar más archivos complementarios a su curso.

El botón **Borrar** le da la oportunidad de borrar los cursos creados.

Es importante mencionar que los profesores solo pueden editar y borrar los cursos que ellos hayan agregado, no los de otros profesores, esto por temas de seguridad.

Estructura del programa

La siguiente lista de clases muestra únicamente las clases principales del programa, junto a sus métodos y atributos con los cuales sin ellos el programa no podría funcionar, ya sea con la GUI o desde la consola.

Clases principales

1. Progreso.

- Maneja el avance de los usuarios en actividades como cursos y evaluaciones.
- **Atributos:** Título, progreso, actividad asociada, fecha de la última actividad
- **Métodos:**
 - actualizarProgreso(int progresoAdicional) : Actualiza el progreso de la actividad.
 - consultarProgreso() : Muestra el progreso actual.

2. Curso.

- Representa un curso con su descripción, profesores, etc.
- **Atributos:** Título, descripción, temario, profesores, materias, evaluaciones.
- **Métodos:**
 - agregarEvaluacion(Evaluacion evaluacion) : Agregue una evaluación al curso.
 - agregarProfesor(String profesor) : Agregue un profesor al curso.

3. Evaluación.

- Administra evaluaciones personalizadas.
- **Atributos:** Título, preguntas, opciones, respuestas correctas.
- **Métodos:** Gestión de preguntas.

4. Usuario.

- Clase base para estudiantes, tutores y administradores.
- **Atributos:** Nombre, apellido, correo, contraseña, tipo de usuario, suscripción.

- **Métodos:**

- crearCuenta(): Crea una nueva cuenta de usuario.
- iniciarSesion(String nombre, String contraseña): Permite al usuario acceder al sistema.

5. **Tutor** (Extensión de Usuario).

- Diseñado para usuarios que actúan como profesores.
- Métodos:
 - crearEvaluacion(Scanner scanner):Crea evaluaciones personalizadas.

6. **Administrador** (Extensión de Usuario).

- Maneja estadísticas del sistema y tareas administrativas.

7. **Principal.**

- Implemente el menú interactivo y las opciones principales del sistema.

Otras funciones

Adicional a todo lo que se ha comentado, el programa cuenta con la clase evaluación donde los estudiantes pueden realizar evaluaciones personalizadas creadas por los profesores.

Asesorías, es una clase dentro del programa donde los profesores pueden atender a sus alumnos y programar clases particulares con ellos para tratar temas relevantes en su trayecto académico.

Desafortunadamente, estas dos clases no se lograron implementar dentro del sistema pues se requiere de un servicio web (hoster) para poder trabajar con dos o más dispositivos a la vez, así como el conocimiento de varios temas relacionados a seguridad computacional y desarrollo web sin olvidar de una atención mayor en el desarrollo de bases de datos con lenguajes avanzados como SQL, temas los cuales no se exploraron durante las clases y que no son relevantes para este curso.

Casos de uso.

Esta página muestra un breve resumen de los diversos casos de uso principales que tiene el programa, sin embargo, se recomienda revisar el siguiente documento el cuál muestra de forma detallada todos los casos de uso principales:

<https://github.com/SinR0str0/Proyecto-Plaga/blob/ed017da701756acba026907f5561f7925cf44754/Diagramas/Casos%20de%20usos.pdf>.

Caso 1: Crear una cuenta.

1. El usuario selecciona la opción Crear Cuenta.
2. Ingresa sus datos personales y tipo de usuario (Estudiante o Profesor).
3. Si es Administrador, debe hacer una petición especial a otro administrador y realizar pruebas externas para que se agregue su rol manualmente.
4. El usuario creado es guardado en la base de datos.

Caso 2: Realizar una evaluación.

1. El usuario inicia sesión como Estudiante.
2. Selecciona un curso y una evaluación disponible.
3. Responde las preguntas y recibe calificación al final.
4. La evaluación queda guardada en el perfil del estudiante.

Caso 3: Crear una evaluación (Tutor).

1. El tutor inicia sesión.
2. Selecciona la opción para crear evaluaciones.
3. Defina preguntas, opciones y respuestas correctas.
4. Asigna la evaluación a un curso.

Si el enlace anteriormente descrito falla, puede encontrar los casos de uso dentro del repositorio donde se encuentra este documento (GitHub) en la carpeta Diagramas>Casos de Uso.

Extensión y Mantenimiento del Sistema

Extensión

1. Para agregar nuevas funcionalidades:
 - Cree nuevas clases o métodos según la funcionalidad requerida.
 - Integrar estas funciones en el menú principal (clase Main).
 - Igual puede agregar clases nuevas con más funcionalidades en cada una de las pantallas.
2. Ejemplo: Agregar un sistema.
 - Crear una clase Notificación.
 - Implementar métodos Usuario para gestionar notificaciones,

Mantenimiento

- Validación de datos. Verificar siempre la entrada del usuario.
- Actualizaciones: Probar el sistema después de modificarlo.
- Optimización: Revisión periódica del rendimiento.

Apéndice

Glosario

Actor: Una entidad (persona, sistema o dispositivo) que interactúa con el sistema.

ArrayList / Lista: Una estructura de datos en Java que permite almacenar elementos en una lista dinámica.

Atributos: Variables o parámetros que tiene cada clase para que funcione correctamente.

Base de Datos: Herramienta que recopila y organiza información de forma electrónica, ya sea en un ordenador, en un servidor o en la nube.

Botón: Componente de una interfaz gráfica que permite al usuario realizar acciones específicas al presionarlo, suele ser los detonadores en nuestro sistema (trigger).

Constructor: Método especial de una clase que se utiliza para inicializar objetos.

Consulta: Solicitud de información o modificación de datos en una base de datos.

Curso: Conjunto estructurado de actividades y contenidos que buscan enseñar un tema específico.

Proceso Dinámico: Interfaces y métodos que se ajustan automáticamente al contenido o acciones del usuario.

Enum: Tipo de dato especial en Java que define un conjunto de valores constantes predefinidos.

Event Listener: Componente que "escucha" eventos como clics de ratón o pulsaciones de teclado en una interfaz gráfica.

Frame (Ventana): Ventana o página principal en una aplicación de escritorio Java donde se muestran los componentes gráficos.

Historial: Registro de acciones o progreso de un usuario dentro de un sistema.

Hoster: Servicio web, por lo general en la nube, para almacenar y procesar información y paquetes que permiten conectar a distancia dos o más dispositivos.

HTML: Por sus siglas en inglés, lenguaje de marcado de hipertexto,

hace referencia al lenguaje de marcado utilizado en la creación de páginas web.

Interfaz Gráfica (GUI): Medio visual a través del cual un usuario interactúa con un sistema.

JSON: Por sus siglas, JavaScript Object Notation, el cual es un formato ligero para almacenar y transportar datos de manera estructurada.

Métodos: También llamadas funciones, son los procesos que una clase u objeto podrá realizar a lo largo de todo el programa.

POO: Siglas de Programación Orientada a Objetos.

Tutor: Usuario que crea y gestiona cursos dentro de un sistema educativo.

Usuario: Persona que interactúa con el sistema, como estudiante, puede tomar clase, ver videos, realizar evaluaciones, etc.

Validación: Proceso de verificar que los datos ingresados por el usuario cumplen con ciertos criterios.

Información de contacto

Para obtener más información sobre este sistema, actualizaciones o contactar directamente con los desarrolladores del software puede comunicarse al correo electrónico: `sin.r0str0rbx@gmail.com`, o bien, a través del medio por el cuál se le fue compartido este documento.

Versiones

1.0: La última versión de este proyecto salió el 03 de diciembre de 2024, el cuál es la presentación oficial del software del equipo Plaga, cuenta con:

- Gestión de Usuario.
- Gestión de Cursos.
- Progresos.
- Interfaz gráfica.
- Base de Datos Simple.
- Funciones Administrativas.
- Documentación.

Otros Datos

De forma complementaria se adjuntan los Diagramas de Actividades, de Estados, de Casos de Uso y UML usados para la planeación, diseño y elaboración de este sistema, lo pueden encontrar en este enlace

<https://github.com/SinR0str0/Proyecto-Plaga/tree/ed017da701756acba026907f5561f7925cf44754/Diagramas> o bien, en la ubicación de este repositorio (GitHub) en la carpeta Diagramas.

Conclusiones

El desarrollo del Sistema de Gestión de Cursos ha sido un proyecto enriquecedor que nos ha permitido reforzar los conocimientos adquiridos en la materia de Programación Orientada a Objetos. A través de este proyecto, se ha demostrado cómo los conceptos fundamentales de POO —como herencia, polimorfismo, encapsulamiento y abstracción— pueden aplicarse para diseñar y construir un sistema funcional, modular y escalable.

Este proyecto no solo cumple con el objetivo académico de integrar teoría y práctica, sino que también simula un escenario real de desarrollo de software. Se logró crear un sistema que gestiona usuarios, cursos, evaluaciones y asesorías de manera eficiente, con una estructura de código clara y fácilmente ampliable.

Además, trabajar en este sistema ha reforzado habilidades críticas como el análisis de problemas, la organización del código y el diseño de soluciones técnicas, no olvidemos que este proyecto nos permitió desarrollarnos autónomamente al revisar conceptos nuevos aplicados en la materia, y comprender mejor el uso de la Inteligencia Artificial para problemas reales. Estas habilidades son fundamentales para proyectos futuros, tanto académicos como profesionales.

En general, el proyecto refleja el potencial de la POO para resolver problemas complejos, promoviendo el diseño de software bien estructurado y preparado para adaptarse a nuevas necesidades. Este sistema es un claro ejemplo de cómo los conceptos teóricos pueden transformarse en soluciones prácticas y útiles.