# **Sistema Integral De Gestión Educativa**

Anderson Bornachera, Farid Bustos, Sergio Chiquillo, Diego de la Hoz y Sharay Machacón

Universidad del Magdalena

Facultad de Ingeniería, Ingeniería de Sistemas

Programación Orientada a Objetos

Ingeniero: Carlos Nelson Henríquez Miranda

Santa Marta, Magdalena

2024

**Problema:**

Gestión eficiente de recursos educativos y apoyo académico.

**Descripción del problema:**

Las instituciones educativas enfrentan un desafío constante en la gestión eficiente de sus recursos, especialmente en lo que respecta a la coordinación entre maestros, estudiantes, asignaturas y monitorias. A medida que crece la complejidad de las estructuras académicas, se vuelve cada vez más difícil asegurar que los estudiantes reciban el apoyo adecuado y que los recursos se utilicen de manera óptima.

**Descripción de la solución:**

El sistema integral de gestión educativa es una plataforma diseñada para centralizar y optimizar la gestión de información relacionada con maestros, estudiantes, asignaturas y monitorias dentro de una institución educativa.

**Historias de usuario**

|  |
| --- |
| **Historia de Usuario 1** |
| Como administrador de Ayre quiero ingresar, eliminar y buscar a un estudiante en una materia con un profesor para gestionar su inscripción y cambios de manera eficiente. |
| **Descripción** |
| El administrador de Ayre debe estar en la capacidad de añadir nuevos estudiantes a una materia, eliminarlos y cambiarlos de materia si es necesario, se deben incluir detalles clave como nombre, código estudiantil, semestre, materia y nota final. |
| **Criterios de aceptación** |
| * Debe ser posible poder ingresar todos los datos del estudiante, tales como nombre, código estudiantil, semestre y la materia que cursará. |

|  |
| --- |
| **Historia de Usuario 2** |
| Como administrador de Ayre quiero ingresar, eliminar y buscar una materia para mantener actualizado el catálogo académico. |
| **Descripción** |
| El administrador debe estar en la capacidad de añadir nuevas materias y eliminarlas si es necesario, se deben incluir detalles como el nombre de la materia, el código de esa materia, el profesor que la dará, los cupos y el semestre en la cual se debe ver esa materia. |
| **Criterios de aceptación** |
| * Debe ser posible ingresar todos los datos de las materias, tales como nombre, código de la materia, nombre del profesor, cupos y semestre. * El semestre no debe ser un requisito previo, es solo un indicativo de cuando debería verse la materia. |

|  |
| --- |
| **Historia de Usuario 3** |
| Como administrador de Ayre quiero asignarle o eliminarle un curso a un profesor de una materia, para tener control de quienes imparten las materias en la universidad. |
| **Descripción** |
| El administrador debe tener la capacidad de agregar y eliminar un curso a un profesor de las materias, con el fin de controlar quienes dictarán las materias en la universidad. |
| **Criterios de aceptación** |
| * Debe ser posible ingresar todos los datos de los profesores, tales como nombre y código. |

|  |
| --- |
| **Historia de Usuario 4** |
| Como estudiante quiero poder añadir una reseña de un profesor para compartir mi experiencia con otros estudiantes. |
| **Descripción** |
| El estudiante debe estar en la capacidad de añadir reseñas de un profesor, siempre y cuando ya haya culminado el semestre y visto la materia con ese profesor. |
| **Criterios de aceptación** |
| * Debe haber terminado el semestre. * El estudiante debió haber cursado la materia con ese profesor. * Todos pueden ver esas reseñas. |

|  |
| --- |
| **Historia de Usuario 5** |
| Como administrador de Ayre quiero poder añadir un monitor a una materia para proporcionar apoyo adicional a los estudiantes. |
| **Descripción** |
| El administrador debe tener la capacidad de agregar un nuevo monitor a la materia si se necesita, se debe incluir información como el código del monitor y el nombre de la materia a la que se va a agregar. |
| **Criterios de aceptación** |
| * Se debe poder añadir todos los datos necesarios para el anexo del monitor a la materia. |

|  |
| --- |
| **Historia de Usuario 6** |
| Como profesor quiero ver la tasa de mortalidad de mi materia para identificar problemas en el rendimiento académico de los estudiantes. |
| **Descripción** |
| El sistema debe poder calcularle al profesor la tasa de mortalidad de su materia, esto con la finalidad de poder determinar qué aspectos son claves en el aprendizaje del estudiante. |
| **Criterios de aceptación** |
| * Se debe calcular con base al semestre y cantidad de estudiantes matriculados en esa materia. |

|  |
| --- |
| **Historia de Usuario 7** |
| Como profesor quiero ver la tasa de éxito de mi materia para evaluar la efectividad de mis métodos de enseñanza. |
| **Descripción** |
| El sistema debe poder calcularle al profesor la tasa de éxito de su materia, esto con la finalidad de poder determinar qué estrategias poder emplear para mejorar esos números. |
| **Criterios de aceptación** |
| * Se debe calcular con base al semestre y cantidad de estudiantes matriculados en esa materia. |

|  |
| --- |
| **Historia de Usuario 8** |
| Como estudiante quiero poder ver la tasa de retiro de los demás estudiantes con un profesor en una materia para saber la metodología del profesor que la imparte. |
| **Descripción** |
| El estudiante debe poder ver la tasa de retiros de estudiantes con un profesor en específico de la materia a su elección. |
| **Criterios de aceptación** |
| * El sistema debe calcular la tasa de retiro con base al semestre y la cantidad de los estudiantes matriculados en dicha materia. |

|  |
| --- |
| **Historia de Usuario 9** |
| Como administrador de Ayre quiero poder ver todos los estudiantes que están inscritos en una materia para poder gestionar mejor la contratación de mas o menos profesores. |
| **Descripción** |
| El administrador debe tener la capacidad de ver la lista de los estudiantes inscritos en una materia, para una mejor gestión de contratación de profesores. |
| **Criterios de aceptación** |
| * El sistema debe mostrar una lista de los estudiantes de dicha materia. |

|  |
| --- |
| **Historia de Usuario 10** |
| Como universidad quiero poder ver el porcentaje de alumnos que están viendo por segunda vez una materia para ofrecerles una ayuda extra en la materia. |
| **Descripción** |
| Esta funcionalidad permitirá identificar a los estudiantes que están repitiendo una materia, para proporcionarles recursos adicionales o intervenciones académicas. |
| **Criterios de aceptación** |
| * El sistema debe mostrar el porcentaje de estudiantes que repiten cada materia. * Los datos deben poder ser filtrados por materia y semestre. |

|  |
| --- |
| **Historia de Usuario 11** |
| Como profesor quiero saber cuántas personas asisten a las monitorías de mi materia para saber si es necesaria dicha monitoría. |
| **Descripción** |
| Esta funcionalidad permitirá a los profesores evaluar la asistencia a las monitorías y decidir sobre su continuidad o ajuste. |
| **Criterios de aceptación** |
| * El sistema debe registrar la asistencia de los estudiantes a cada sesión de monitoría. |

|  |
| --- |
| **Historia de Usuario 12** |
| Como profesor, quiero saber si a un estudiante le sirvió ir a la monitoría para evaluar su impacto en su aprendizaje y mejorar el apoyo brindado. |
| **Descripción** |
| Esta funcionalidad permitirá a los profesores recopilar feedback de los estudiantes sobre la utilidad de las monitorías para mejorar su diseño y ejecución. |
| **Criterios de aceptación** |
| * El sistema debe poder ver si un estudiante repitente fue a una monitoria y gano la materia. |

|  |
| --- |
| **Historia de Usuario 13** |
| Como profesor, quiero saber si a un estudiante no le sirvió ir a la monitoría para identificar áreas de mejora y ajustar las estrategias de apoyo. |
| **Descripción** |
| Esta funcionalidad permitirá identificar qué aspectos de las monitorías no están funcionando bien para realizar mejoras específicas. |
| **Criterios de aceptación** |
| * El sistema debe poder ver si un estudiante repitente fue a una monitoria y no gano la materia. |

|  |
| --- |
| **Historia de Usuario 14** |
| Como administrador de Ayre quiero poder ver a todos los profesores que tengo en nomina para administrar mejor el presupuesto de la universidad |
| **Descripción** |
| El administrador debe tener la capacidad de ver los profesores que haya inscritos en la universidad. |
| **Criterios de aceptación** |
| * El sistema debe poder mostrar una lista de profesores que trabajan en la universidad. |

|  |
| --- |
| **Historia de Usuario 15** |
| Como profesor, quiero poder ver un resumen de los estudiantes con bajo rendimiento en una materia para ofrecerles ayuda adicional antes del final del semestre. |
| **Descripción** |
| El profesor debe tener la capacidad de ver un resumen sobre los estudiantes con bajo rendimiento. |
| **Criterios de aceptación** |
| * El sistema debe mostrar una lista de estudiantes con bajo rendimiento a el profesor. |

|  |
| --- |
| **Historia de Usuario 16** |
| Como estudiante, quiero poder acceder a las notas finales de una materia para saber cómo me desempeño en comparación con otros estudiantes. |
| **Descripción** |
| Los estudiantes deben poder ver no solo sus notas finales, sino también las estadísticas generales de la materia, como el promedio de la clase, el número de aprobados, y su posición relativa dentro de la clase. |
| **Criterios de aceptación** |
| * Los estudiantes deben poder ver su calificación final junto con el promedio de la clase. * El sistema debe mostrar un desglose de las estadísticas generales de rendimiento de la clase. |

|  |
| --- |
| **Historia de Usuario 17** |
| Como administrador quiero poder ver los cursos que un profesor posea. |
| **Descripción** |
| El administrador debe tener la capacidad de ver los cursos en los que un profesor esté inscrito. |
| **Criterios de aceptación** |
| * El sistema debe mostrar una lista de los cursos en los que dicho profesor esté inscrito. |

|  |
| --- |
| **Historia de Usuario 18** |
| Como estudiante quiero poder ver las reseñas de otros estudiantes a un profesor para saber si recomiendan matricular con dicho profesor. |
| **Descripción** |
| El estudiante tendrá la capacidad de ver las reseñas hachas por otros estudiantes acerca de un profesor. |
| **Criterios de aceptación** |
| * El sistema mostrará todas las reseñas hechas por los estudiantes acerca de el profesor que imparte la materia. |

|  |
| --- |
| **Historia de Usuario 19** |
| Como monitor, quiero poder gestionar sesiones de monitorias para tener un control de las monitorias que realizo. |
| **Descripción** |
| El monitor debe tener la capacidad de gestionar una monitoria para mayor control de estas. |
| **Criterios de aceptación** |
| * Debe ser posible ingresar todos los datos necesarios para agregar la monitoria, tales como código, fecha, hora de inicio, hora de finalización y tema. |

|  |
| --- |
| **Historia de Usuario 20** |
| Como monitor quiero poder gestionar asistencias a las monitorias que realizo para llevar un control de cuantas personas acuden a mis monitorias |
| **Descripción** |
| El monitor debe tener la capacidad de gestionar las asistencias a la monitoria que realizó para el control de personal que acuden a estas. |
| **Criterios de aceptación** |
| * Debe ser posible ingresar todos los datos necesarios para agregar la asistencia, tales como el código, facultad, programa y estado del estudiante que acudió a la monitoria. |

**Documentación Técnica del Proyecto de Gestión Universitaria**

**Introducción**

El proyecto de Gestión Universitaria es un sistema diseñado para administrar eficientemente los procesos académicos de una universidad. Proporciona funcionalidades como la gestión de usuarios (estudiantes, profesores y administradores), asignación de materias, control de monitorias, y evaluación de rendimiento académico.

**Arquitectura del Sistema**

El sistema está implementado en **Java** y utiliza una arquitectura orientada a objetos. Todas las clases están organizadas en el paquete modelos, con interfaces adicionales para estandarizar ciertas operaciones.

**Principales Componentes**

**Clases del Sistema:**

**Universidad**

* **Descripción**: Clase principal que gestiona los elementos clave del sistema como usuarios, materias y roles.
* **Atributos**:
  + ArrayList<Usuario> usuarios: Lista de usuarios registrados en el sistema.
  + ArrayList<Materia> materias: Lista de materias disponibles.
  + Usuario usuarioActual: Usuario autenticado actualmente.
* **Métodos principales**:
  + ingresarUsuario(Usuario x): Agrega un usuario al sistema.
  + eliminarUsuario(String username): Elimina un usuario según su nombre de usuario.
  + ingresarMateria(Materia x): Agrega una materia al sistema.
  + eliminarMateria(int codigo): Elimina una materia según su código.
  + tasaDeMortalidad(int codigoMateria): Calcula el porcentaje de estudiantes que pierden una materia.
  + tasaDeExito(int codigoMateria): Calcula el porcentaje de estudiantes que pasan una materia.

**Curso**

* **Descripción**: Representa un curso específico asociado a una materia.
* **Atributos**:
  + Materia materia: Materia a la que pertenece el curso.
  + int grupo: Grupo específico del curso.
  + ArrayList<Estudiante\_Curso> estudiantes: Lista de estudiantes inscritos.
* **Métodos principales**:
  + add(Estudiante\_Curso x): Agrega un estudiante al curso.
  + delete(int codigoEstudiante): Elimina un estudiante del curso.
  + buscar(int codigoEstudiante): Busca un estudiante por su código.

**Estudiante**

* **Descripción**: Clase que extiende a Usuario y representa a un estudiante.
* **Atributos**:
  + int codigo: Código único del estudiante.
  + String facultad: Facultad a la que pertenece el estudiante.
  + String programa: Programa académico del estudiante.
  + String estado: Estado del estudiante (activo, retirado, etc.).
* **Métodos principales**:
  + Métodos getter y setter para todos los atributos.

**Estudiante\_Curso**

* **Descripción**: Clase que representa la inscripción de un estudiante en un curso específico.
* **Atributos**:
  + Estudiante estudiante: Estudiante inscrito.
  + float nota1, nota2, nota3: Notas del estudiante en el curso.
  + String estado: Estado del estudiante en el curso.
* **Métodos principales**:
  + calcularNotaFinal(): Calcula el promedio de las notas del estudiante.
  + ganoMateria(): Determina si el estudiante pasó el curso.

**Profesor**

* **Descripción**: Clase que extiende a Usuario y representa a un profesor.
* **Atributos**:
  + int codigo: Código único del profesor.
  + ArrayList<Curso> cursos: Lista de cursos asignados.
  + ArrayList<Resena> resenas: Lista de reseñas recibidas.
* **Métodos principales**:
  + agregarCurso(Curso x): Agrega un curso al profesor.
  + eliminarCurso(int codigoMateria, int grupo): Elimina un curso específico.
  + tasaDeRetiroDeProfesor(int codigoMateria): Calcula la tasa de retiro de los estudiantes en las materias impartidas por el profesor.

**Materia**

* **Descripción**: Representa una materia del plan académico.
* **Atributos**:
  + String nombre: Nombre de la materia.
  + int codigo: Código único de la materia.
  + Estudiante monitor: Estudiante asignado como monitor.
  + ArrayList<Monitoria> monitorias: Lista de monitorías asociadas.
* **Métodos principales**:
  + add(Monitoria x): Agrega una monitoría a la materia.
  + delete(int codigo): Elimina una monitoría según su código.

**Monitoria**

* **Descripción**: Representa una monitoría realizada para una materia.
* **Atributos**:
  + int codigo: Código único de la monitoría.
  + Fecha fecha: Fecha de realización de la monitoría.
  + Hora horaInicio, horaFin: Horas de inicio y fin de la monitoría.
  + ArrayList<Estudiante> asistencias: Lista de estudiantes que asistieron.
* **Métodos principales**:
  + add(Estudiante x): Registra la asistencia de un estudiante.
  + delete(int codigo): Elimina a un estudiante de la lista de asistencias.

**Fecha y Hora**

* **Descripción**: Clases auxiliares para gestionar información de tiempo.
* **Atributos principales**:
  + Fecha: int dia, int mes, int anio.
  + Hora: int hora, int minuto.

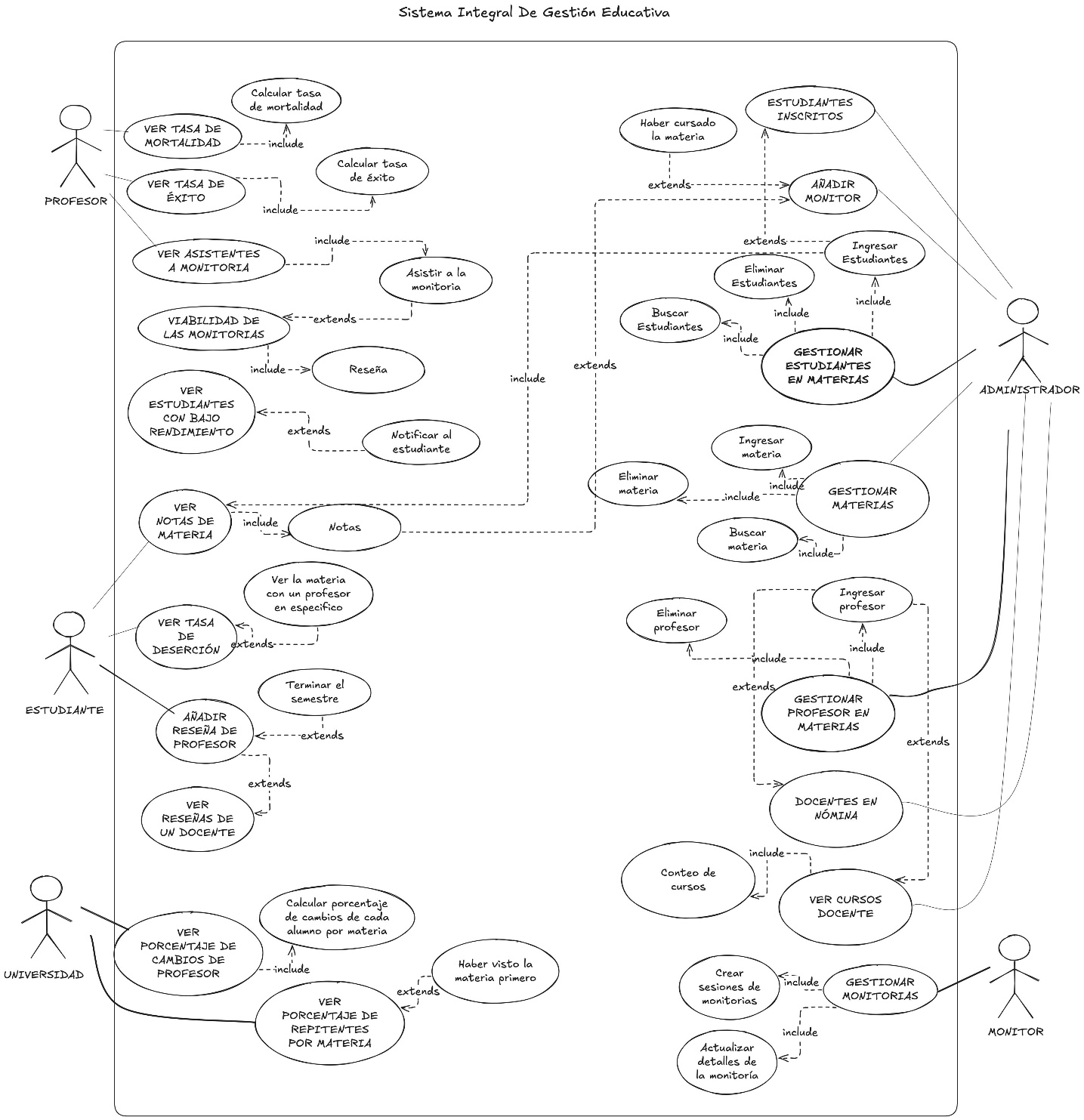
**Resena**

* **Descripción**: Clase para gestionar las reseñas que un estudiante puede dar a un profesor.
* **Atributos**:
  + String descripcion: Contenido de la reseña.
  + float puntaje: Calificación dada.
  + Estudiante estudiante: Autor de la reseña.
  + int codigo: Código único de la reseña.
* **Métodos principales**:
  + Métodos getter y setter para todos los atributos.

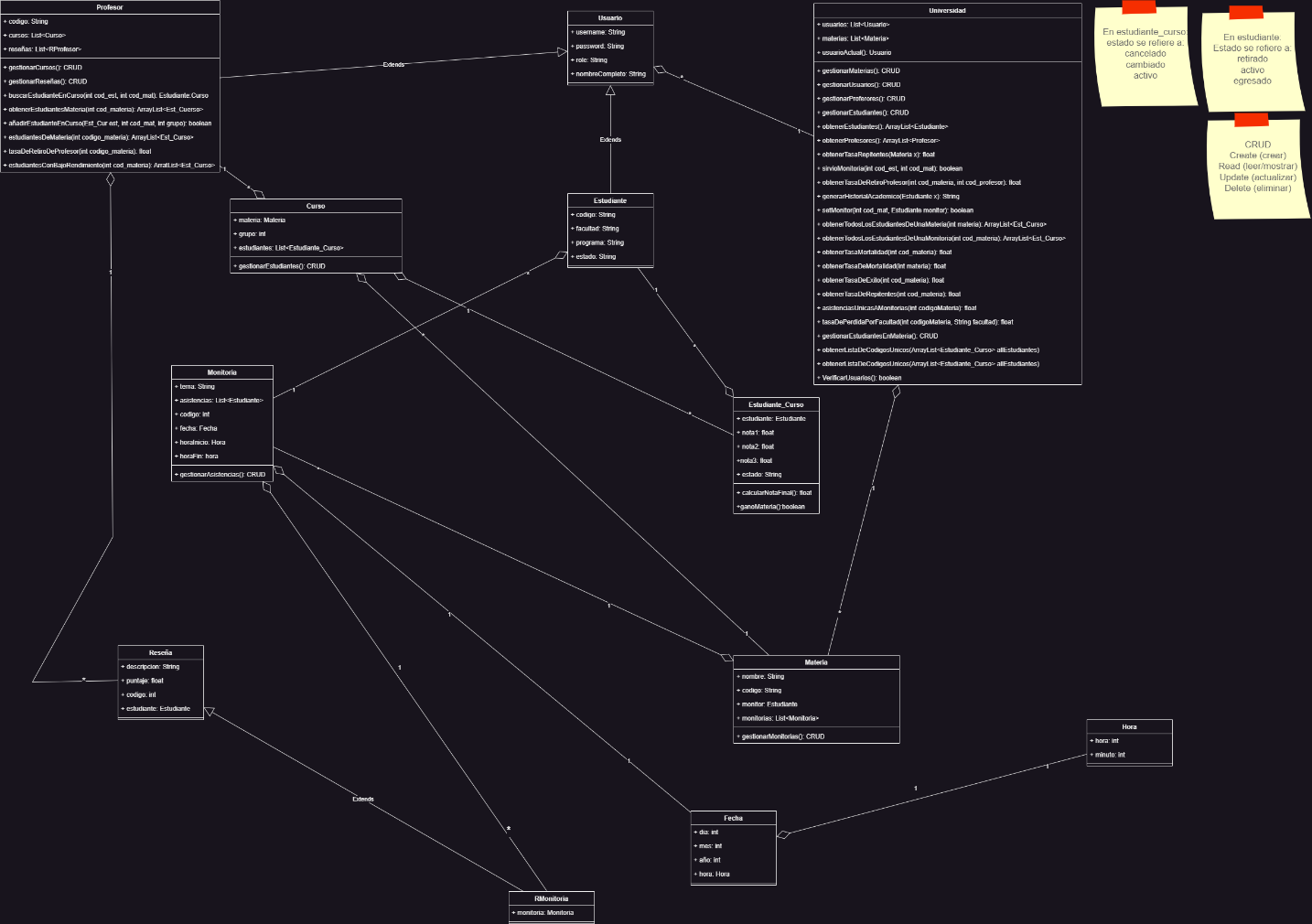
**Usuario**

* **Descripción**: Clase base para los usuarios del sistema.
* **Atributos**:
  + String username: Nombre de usuario.
  + String password: Contraseña.
  + String role: Rol del usuario (estudiante, profesor, admin).
  + String fullName: Nombre completo del usuario.

**Casos de uso**

****

**Diagrama UML**

****

**Wireframe**

* **https://www.figma.com/design/JgvXzI5v71DavkrEdXIWr7/WireframePoo?node-id=0-1&t=a0DDv2zd77JLTWKH-1**