



# **Sistema de gestión académico: Academix**

Autor: Ricardo Cañada Gracia.  
Tutor: Edgar Bernáldez Mangas.  
Desarrollo de Aplicaciones Multiplataforma.  
Junio 2023.

## **RESUMEN**

Hoy en día, las tareas de índole administrativas o de gestión representan una carga significativa para los docentes. Los educadores, inmersos en un universo de informes, registros y trámites, se ven obligados a lidiar con la complejidad de la gestión académica. De esta manera, estas exigencias adicionales, consumen un tiempo valioso que podría emplearse en labores puramente educativas. Es crucial reconocer esta disyuntiva y buscar soluciones que liberen a los docentes de estas responsabilidades, permitiéndoles centrarse plenamente en su misión educativa.

La aplicación Academix se ha desarrollado como una solución al desafío que enfrentan los docentes en su labor diaria; así, Academix intenta optimizar diversos procesos académicos repetitivos. Por ello, surge esta aplicación de escritorio, desarrollada en entorno Java, y haciendo uso de otras tecnologías como MySQL como sistema gestor de base de datos para almacenar la información, Hibernate para facilitar la gestión de la persistencia de datos, o, JavaFX para el desarrollo de interfaces gráficas.

En resumen, Academix se presenta como una solución tecnológica que aborda la carga administrativa que aqueja a los docentes. A través de su implementación, esta aplicación busca optimizar los procesos educativos al liberar a los profesores de tareas burocráticas, permitiéndoles dedicarse plenamente a su labor pedagógica.

## **ABSTRACT**

In today's educational landscape, administrative and management tasks have become a significant burden for teachers. Educators find themselves immersed in a realm of reports, records, and procedures, which detracts valuable time that could be devoted to purely educational endeavors. It is crucial to recognize this dilemma and seek solutions that alleviate teachers from these responsibilities, enabling them to fully focus on their educational mission.

Academix, the developed application, emerges as a solution to address the challenges faced by teachers in their daily work. With the aim of freeing teachers from these burdensome tasks, Academix seeks to optimize repetitive processes, allowing educators to direct their efforts towards their core responsibilities. This desktop application is developed in the Java environment, utilizing technologies such as MySQL as the database for data storage, Hibernate for efficient data persistence management, and JavaFX for the development of graphical interfaces.

In conclusion, Academix presents itself as a technological solution that tackles the administrative burden experienced by teachers. Through its implementation, this application aims to streamline educational processes by relieving teachers from bureaucratic tasks, thus enabling them to fully dedicate themselves to their pedagogical duties.

## ÍNDICE.

<b>1. Datos descriptivos.</b>	<b>4</b>
1.1. Descripción general. Marco teórico. Principios tecnológicos y normas técnicas.	4
1.2. Análisis de la realidad. Entorno de la empresa colaboradora.	4
1.3. Justificación.	5
1.4. Marco legal.	5
<b>2. Desarrollo.</b>	<b>6</b>
2.1. Acuerdo.	6
2.1.1. Generales.	6
2.1.2. Requisitos funcionales.	7
2.1.3. Requisitos no funcionales.	8
2.1.4. Especificaciones técnicas del proyecto.	8
2.2. Actividades y cronología.	9
2.2.1. Diagrama de Gantt.	9
2.2.2. Análisis y diseño.	10
- Diagrama de bases de datos.	10
- Diagrama de clases.	11
- Diagrama de casos de uso.	13
- Diagrama de secuencias.	16
2.2.3. Pruebas del sistema.	17
2.2.4. Documentación.	18
2.3. Metodología.	33
2.4. Recursos.	34
2.4.1. Materiales.	34
2.4.2. Infraestructura y equipamiento.	35
2.4.3. Humanos.	36
<b>3. Presupuesto.</b>	<b>37</b>
3.1. Gastos.	37
3.2. Ingresos.	38
3.3. Beneficios.	38
<b>4. Conclusiones.</b>	<b>39</b>
<b>5. Bibliografía.</b>	<b>40</b>
<b>6. Anexos.</b>	<b>44</b>

## 1. Datos descriptivos.

### 1.1. Descripción general. Marco teórico. Principios tecnológicos y normas técnicas.

Academix es una aplicación de gestión académica, diseñada para simplificar el proceso de seguimiento y evaluación del rendimiento académico de los estudiantes. La aplicación está dirigida a profesores y permite la creación grupos, asignaturas, perfiles de administradores, profesores y estudiantes, la gestión de notas y del rendimiento diario del alumnado, así como la visualización de informes sobre dicho rendimiento.

Por consiguiente, Academix es una solución completa y versátil para la gestión académica. Ofrece a los profesores las herramientas necesarias para simplificar y mejorar la evaluación del rendimiento de los estudiantes, fomentando así un entorno educativo más efectivo y centrado en el éxito académico. Con su enfoque en la creación de perfiles, gestión de notas, seguimiento diario y generación de informes, Academix se posiciona como una plataforma integral para optimizar el proceso educativo.

### 1.2. Análisis de la realidad. Entorno de la empresa colaboradora.

El grupo educativo “Círculo Aurora” es una institución educativa de prestigio nacional, fundada en 1985 por un grupo de educadores comprometidos con la excelencia académica y la formación integral de sus estudiantes. Formada por una red de 20 centros, la institución cuenta con una amplia oferta educativa desde la educación infantil hasta bachillerato, con una plantilla de profesores altamente cualificados y un equipo de apoyo administrativo dedicado al bienestar y desarrollo de los estudiantes.

El grupo educativo tiene una amplia trayectoria en la formación de estudiantes comprometidos con la sociedad y capaces de enfrentar los desafíos del mundo globalizado. Su enfoque educativo se basa en la excelencia académica, el desarrollo de habilidades sociales y emocionales, y la formación en valores que les permitan desempeñarse en cualquier ámbito profesional y personal.

Así, dicha institución se encuentra en un sector altamente competitivo, con una gran oferta de centros educativos privados y públicos, que buscan atraer a los estudiantes. Por ello, el centro educativo ha solicitado la aplicación Academix, con el objetivo de simplificar sus tareas administrativas y ahorrar tiempo en la gestión de

las notas de los estudiantes, ya que el centro cuenta con un gran número de alumnos, lo que hace que la gestión de notas sea una tarea muy laboriosa y que requiere mucho tiempo.

En este sentido, el centro es consciente de que la educación es cada vez más competitiva y que la calidad de la enseñanza y la eficiencia son factores clave para atraer y retener a los estudiantes y sus familias. Asimismo, los profesores también han expresado su deseo de tener una herramienta de estas características, con el objetivo de reducir el tiempo que emplean en tareas repetitivas y manuales, y así poder dedicar más atención a su labor educativa.

### 1.3. Justificación.

La gestión de notas de alumnos es una tarea fundamental en el ámbito educativo. Los profesores deben realizar un seguimiento constante del rendimiento de los estudiantes para poder evaluar su progreso, de forma objetiva, a lo largo del curso académico.

En este sentido, la tecnología puede desempeñar un papel fundamental, ya que puede simplificar la gestión de las notas de los estudiantes y el acceso a información detallada sobre su rendimiento académico. Por consiguiente, esto puede, por un lado, ayudar en la toma de decisiones, ya que los profesores pueden analizar fácilmente los datos de notas de los estudiantes, lo que les permitirá detectar patrones y tendencias en su rendimiento. Esto les ayudará a identificar rápidamente a los estudiantes que necesitan una atención adicional o que requieren un enfoque de enseñanza diferente; y, por otro, suponer un ahorro significativo de tiempo.

### 1.4. Marco legal.

El marco legal se refiere al conjunto de normas, leyes, reglamentos, acuerdos y disposiciones que regulan las actividades de una empresa o proyecto. Es esencial para el buen funcionamiento de cualquier proyecto, ya que permite que la empresa opere de manera legal y segura, protegiendo los derechos de todas las partes involucradas.

En el caso específico de Academix, existen diversas leyes y normativas que deben ser consideradas. En primer lugar, se encuentra la Ley de Protección de Datos Personales, la cual regula el uso de información personal de los usuarios y su almacenamiento en bases de datos. Asimismo, se debe considerar la normativa

relacionada con la protección de la propiedad intelectual, debido a que la aplicación contiene información y contenidos protegidos por derechos de autor.

Otra ley importante es la Ley de Servicios de la Sociedad de la Información y del Comercio Electrónico, que establece las obligaciones y responsabilidades de los proveedores de servicios de la sociedad de la información. Esta ley regula aspectos como la publicidad en línea, la protección de datos personales y la contratación de servicios a través de internet.

Por último, es importante considerar las normativas relacionadas con la seguridad informática, las cuales establecen medidas de seguridad que deben ser adoptadas para garantizar la protección de los datos personales y de la información sensible almacenada en la aplicación. En este sentido, se deben implementar medidas de seguridad como la encriptación de datos y la utilización de contraseñas seguras, así como garantizar el cumplimiento de las normas y regulaciones en cuanto a la privacidad y protección de datos de los usuarios.

En conclusión, el marco legal es esencial para garantizar el correcto funcionamiento de cualquier proyecto o empresa, y en el caso de Academix es importante considerar las leyes y normativas relacionadas con la protección de datos personales, la propiedad intelectual, los servicios de la sociedad de la información y la seguridad informática.

## 2. Desarrollo.

### 2.1. Acuerdo.

#### 2.1.1. Generales.

Academix surge como una aplicación para abordar la necesidad de una gestión académica eficiente y simplificada en entornos educativos. En un mundo cada vez más digital, las instituciones educativas necesitan adaptarse a las nuevas tecnologías y aprovechar las ventajas que estas ofrecen, para agilizar procesos y poder brindar una mejor calidad educativa.

Así, debido a que la complejidad y el volumen de datos requieren una solución tecnológica que automatice y agilice estos procesos, Academix proporciona una plataforma con el objetivo de satisfacer esta necesidad.

En resumen, Academix surge para satisfacer la creciente necesidad de una gestión académica eficiente y precisa en entornos educativos. Proporciona una solución tecnológica integral que simplifica la administración de notas, mejora la

comunicación y facilita la generación de informes, brindando a profesores, estudiantes y administradores una herramienta poderosa para optimizar el proceso educativo.

#### 2.1.2. Requisitos funcionales.

1. Registro de usuarios: permitir a los usuarios registrarse en el sistema para tener una cuenta de acceso a la aplicación.
2. Gestión de profesores: permitir a los administradores agregar, editar y eliminar información sobre los profesores, como su nombre, apellidos, correo electrónico, asignaturas impartidas y el grupo al que pertenecen.
3. Gestión de alumnos: permite a los administradores agregar, editar y eliminar información de los alumnos, como su nombre, apellidos, grupo al que pertenecen o su tutor.
4. Gestión de estudiantes: permitir a los administradores agregar, editar y eliminar información sobre los estudiantes, como su nombre, apellidos, dirección, notas y asistencia. Permite a los profesores agregar, editar y eliminar información referente a calificaciones.
5. Gestión de grupos: permite a los administradores agregar, editar y eliminar información sobre los grupos, como el curso, la letra y el tutor del grupo.
6. Gestión de asignaturas: permitir a los administradores agregar, editar y eliminar información sobre las asignaturas, como el nombre de la asignatura y el profesor que la imparte.
7. Gestión de calificaciones: permitir a los profesores registrar y actualizar las calificaciones de los estudiantes para cada curso, asignatura y/o examen o tarea evaluable.
8. Gestión de tareas evaluables: permite a los profesores generar y eliminar información sobre tareas evaluables.
9. Gestión de exámenes: permite a los profesores generar y eliminar información sobre exámenes.
10. Generación de informes: permitir a los profesores generar informes sobre el rendimiento de los estudiantes y el progreso de los cursos.
11. Gestión de seguimiento del alumnado: permite a los profesores generar, editar y eliminar información sobre el rendimiento diario del alumnado.

### 2.1.3. Requisitos no funcionales.

1. Usabilidad: la aplicación debe ser fácil de usar para usuarios con diferentes niveles de experiencia y habilidades.
2. Rendimiento: la aplicación debe ser rápida y responder de manera eficiente a las solicitudes de los usuarios.
3. Seguridad: la aplicación debe proteger los datos de los usuarios y evitar accesos no autorizados.
4. Escalabilidad: la aplicación debe ser capaz de manejar un gran número de usuarios y datos sin afectar su rendimiento.
5. Disponibilidad: la aplicación debe estar disponible en todo momento, con un tiempo de inactividad mínimo programado para el mantenimiento y actualización del sistema.
6. Mantenibilidad: la aplicación debe ser fácil de mantener y actualizar, con código limpio y bien estructurado y documentación clara.

### 2.1.4. Especificaciones técnicas del proyecto.

Academix es una aplicación desarrollada en lenguaje Java, utilizando el framework JavaFX para la creación de interfaces gráficas de usuario. Además, se utiliza la herramienta de gestión de proyectos Maven para manejar las dependencias y automatizar el proceso de construcción del proyecto.

En cuanto al diseño visual de la aplicación, se utilizan hojas de estilo en cascada (CSS) integradas en JavaFx para la personalización de los componentes de la interfaz gráfica de forma separada del código Java. El CSS integrado en JavaFX se llama "JavaFX CSS" o "Estilo de Hoja de Estilo en Cascada de JavaFX". JavaFX CSS es un conjunto de estilos y reglas que se utilizan para personalizar la apariencia de los componentes gráficos de una aplicación JavaFX.

Para la persistencia de los datos, se utiliza la tecnología de mapeo objeto-relacional (ORM) Hibernate, junto con la API de persistencia de Java (JPA), lo que permite una gestión sencilla y eficiente de la base de datos. Asimismo, se utiliza MySQL como sistema gestor de base de datos para crear, gestionar y administrar una base de datos relacional.

El entorno de desarrollo integrado (IDE) utilizado para la programación es NetBeans, y el sistema de control de versiones utilizado es Git, lo que permite



mantener un registro histórico de los cambios realizados en el código y colaborar en el desarrollo del proyecto.

Además, se pueden utilizar otras herramientas y tecnologías complementarias según las necesidades específicas del proyecto; por ejemplo librerías<sup>1</sup> de terceros, como FontAwesomeFX, para implementar iconos en la aplicación, entre otras,

## 2.2. Actividades y cronología.

### 2.2.1. Diagrama de Gantt.

Un diagrama de Gantt es una herramienta de gestión de proyectos que permite representar gráficamente las tareas, plazos y dependencias de un proyecto en forma de barras horizontales en un eje de tiempo. El diagrama de Gantt se utiliza en la planificación y seguimiento de proyectos, brindando una visión clara y estructurada de la cronología del proyecto.

Por consiguiente, el objetivo principal de este tipo de diagramas es proporcionar una representación visual de las actividades del proyecto, su duración, la secuencia en la que deben realizarse y el plazo establecido.

De esta manera, a través del diagrama de Gantt, se proporciona una visión general del proyecto y sus tareas, lo que permite a los equipos de proyecto comprender fácilmente el progreso, la planificación y la asignación de recursos. Asimismo, facilita la identificación de posibles problemas o retrasos, permitiendo la toma de decisiones oportunas para minimizar los riesgos y garantizar la finalización exitosa del proyecto dentro de los plazos establecidos.

En resumen, el diagrama de Gantt es una herramienta valiosa para la planificación, seguimiento y comunicación de proyectos, ya que proporciona una representación gráfica clara y estructurada de las actividades del proyecto a lo largo del tiempo. Su utilidad radica en su capacidad para visualizar las tareas, los plazos y las dependencias, lo que ayuda a los equipos de proyecto a gestionar eficientemente los recursos, identificar problemas potenciales y mantener el proyecto en curso hacia su finalización exitosa.

Debido a la importancia de este tipo de diagramas en la estructuración y elaboración de proyectos, se ha diseñado un diagrama de Gantt<sup>2</sup> específico para representar

---

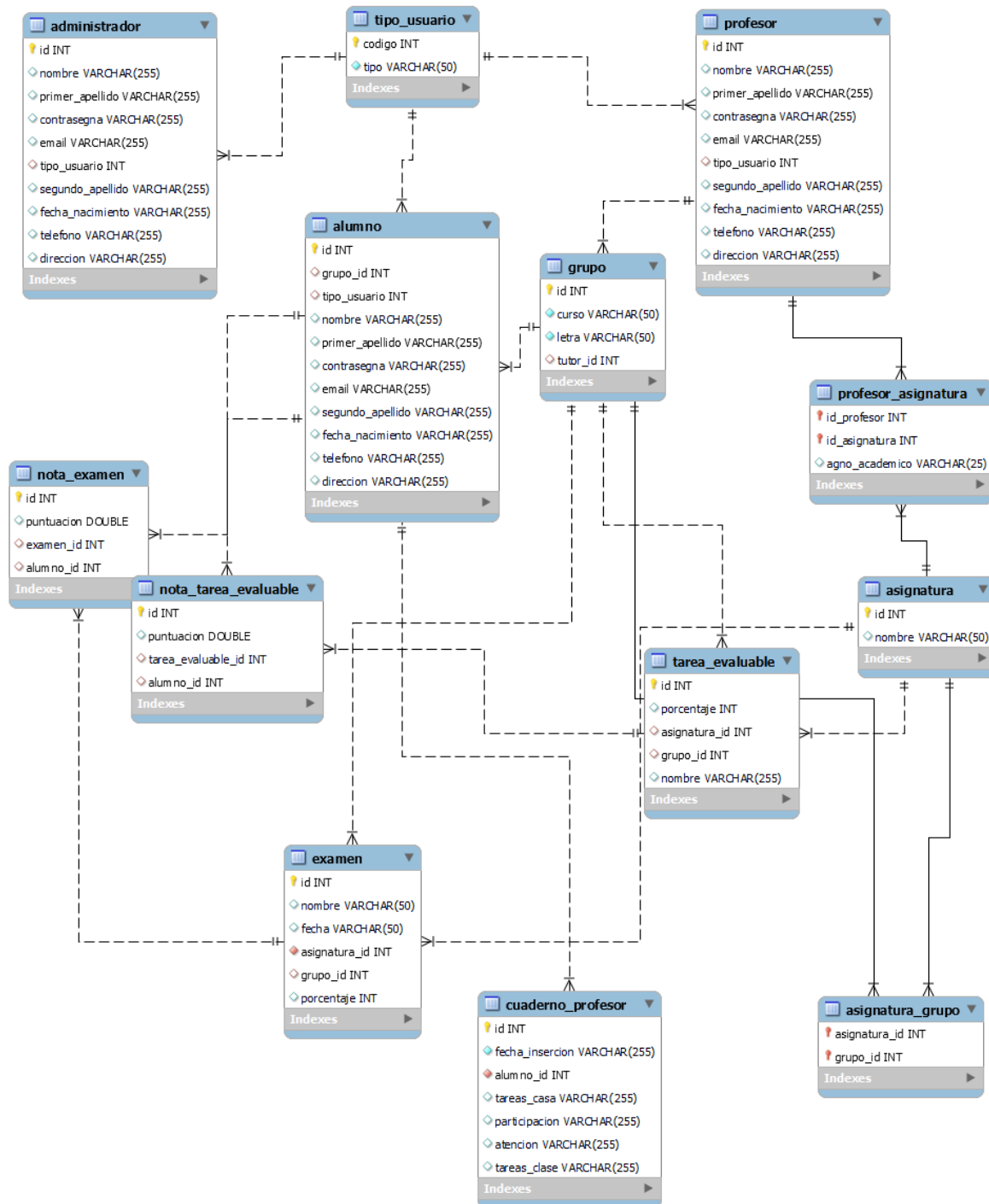
<sup>1</sup> Véase: [Anexos: librerías utilizadas.](#)

<sup>2</sup> Véase: [Anexos: diagrama de Gantt.](#)

gráficamente la planificación de este proyecto académico. Este diagrama permitirá visualizar de manera clara y ordenada las diferentes etapas del proyecto, así como las tareas y actividades necesarias para su implementación exitosa.

### 2.2.2. Análisis y diseño.

- Diagrama de bases de datos.



- Diagrama de clases.

El diagrama de clases es una representación visual estática que muestra la estructura y las relaciones entre las clases en un sistema o aplicación. Es una herramienta fundamental en la fase de diseño del desarrollo de software, ya que proporciona una vista panorámica de la arquitectura del sistema y de cómo las clases se relacionan entre sí.

Por otro lado, en lo referente al patrón utilizado en el proyecto, se ha utilizado el Modelo-Vista-Controlador (en adelante, MVC) en combinación con el patrón Data Access Object (en adelante, DAO). El patrón MVC es ampliamente utilizado en el desarrollo de aplicaciones de software para organizar el código en tres componentes principales: el modelo, la vista y el controlador. Mientras que el patrón DAO (Data Access Object) se utiliza para abstraer el acceso a la capa de persistencia de datos. De esta manera:

- El modelo representa la estructura de datos y la lógica de negocio de la aplicación. Contiene las clases que encapsulan los datos y los métodos para manipularlos. También puede incluir clases de servicio o utilidades que proporcionan funciones adicionales relacionadas con la lógica de negocio.
- La vista es la interfaz de usuario de la aplicación. Es responsable de la presentación de los datos y la interacción con el usuario. Así, las vistas son las diferentes pantallas o ventanas que muestran la información al usuario y capturan su entrada.
- El controlador actúa como intermediario entre el modelo y la vista. Responde a las acciones del usuario en la interfaz de usuario y realiza las operaciones necesarias en el modelo. También actualiza la vista con los cambios en el modelo. Es decir, son los responsables de manejar las interacciones del usuario.
- Los objetos DAO proporcionan métodos para interactuar con los datos en la base de datos. Estos objetos ocultan los detalles de la implementación de la persistencia y permiten que el modelo se comuniqué con la base de datos de manera independiente.

Siguiendo el enfoque del modelo MVC y el patrón DAO en el proyecto, se procederá a incluir los diagramas de clases de cada parte del patrón. A continuación, se describe qué representan los diagramas de clases<sup>3</sup> incluidos en la memoria presente:

---

<sup>3</sup> Véase: [Anexos: diagramas de clase](#).

- Diagrama de clases del modelo: en este diagrama se representan las clases que conforman el modelo de datos. Se mostrarán las propiedades y métodos asociados a cada clase, así como las relaciones entre ellas, como asociaciones, composiciones o herencias.
- Diagrama de clases del controlador: estos diagramas representan las clases de los controladores. En el diagrama, se muestran las clases de controladores correspondientes a cada vista, así como los métodos y eventos asociados a cada una. Además, se representarán las relaciones y dependencias entre los controladores y las otras capas del sistema.
- Diagrama de clases de los objetos DAO: representa las clases que se encargan de la comunicación con la capa de persistencia de datos en mi aplicación.
- Diagrama de casos de uso.

Un diagrama de casos de uso es una representación gráfica que describe las interacciones entre los actores externos y un sistema de software. Es una herramienta esencial utilizada en el campo de la ingeniería de software para analizar, comprender y comunicar los requisitos funcionales de un sistema de manera clara y precisa.

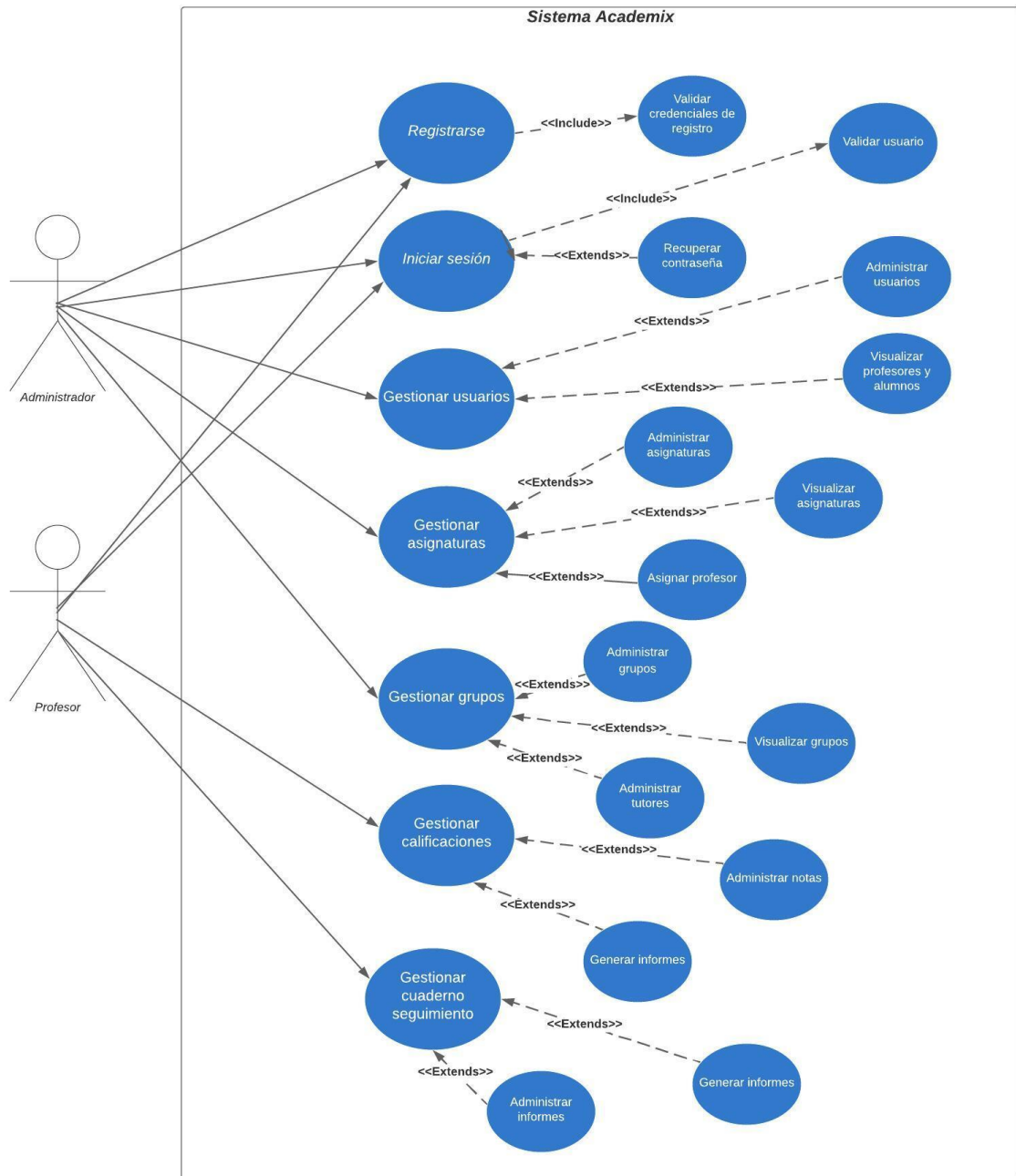
En esencia, un diagrama de casos de uso representa visualmente las situaciones en las que los usuarios interactúan con el sistema y especifica las funcionalidades que el sistema proporciona para satisfacer las necesidades de los usuarios. De esta manera, permite describir de manera visual el comportamiento esperado del sistema, centrándose en los objetivos que los actores desean lograr.

La importancia del diagrama de casos de uso radica en su capacidad para identificar y describir los requisitos funcionales del sistema de una manera intuitiva y accesible. Proporciona una base sólida para la comunicación efectiva entre los diferentes miembros del equipo implicados en el desarrollo del sistema. Además, el diagrama de casos de uso ayuda a visualizar y comprender cómo se relacionan los diferentes actores y funcionalidades del sistema, lo que facilita la detección temprana de problemas y la toma de decisiones informadas durante las etapas de diseño y desarrollo del sistema.

En resumen, el diagrama de casos de uso es una herramienta utilizada en el campo de la ingeniería de software, que permite analizar, comprender y comunicar los requisitos funcionales de un sistema. Su uso adecuado y su comprensión profunda

son fundamentales para el éxito de un proyecto, ya que ayuda a garantizar que el sistema desarrollado cumpla con las necesidades y expectativas de los usuarios finales, al tiempo que facilita una colaboración efectiva entre los diferentes stakeholders involucrados en el proceso de desarrollo de software.

A continuación, se muestra el diagrama de casos de uso del software Academix.



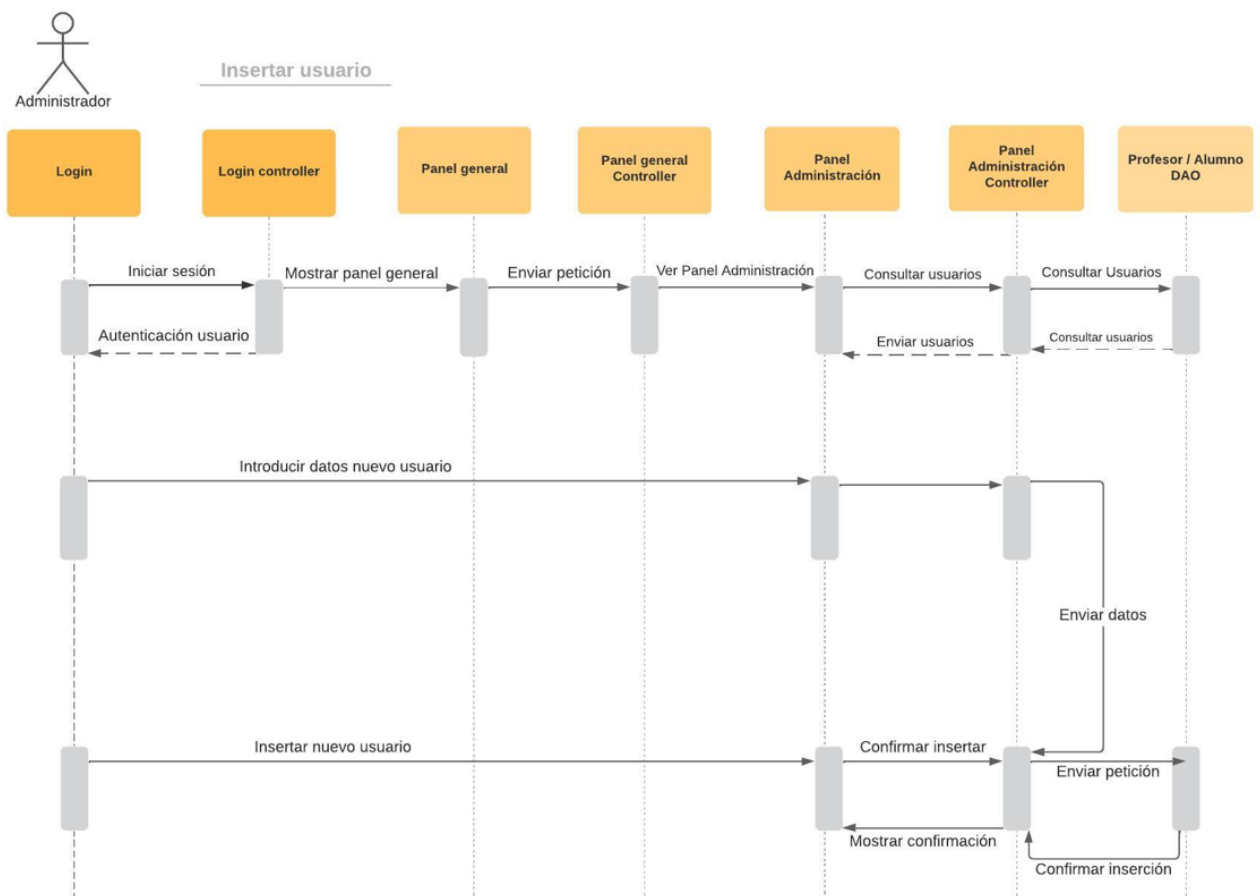
- Diagrama de secuencias.

Un diagrama de secuencias UML es una representación visual que describe la interacción secuencial entre objetos a lo largo del tiempo. Es una herramienta visual utilizada en el campo de la ingeniería de software para modelar y analizar el comportamiento dinámico de un sistema. Este tipo de diagrama permite capturar y comunicar de manera efectiva las interacciones entre los distintos componentes de un sistema, mostrando el orden en el que se producen y cómo se transmiten los mensajes entre ellos.

Este tipo de diagrama es muy eficaz en el proceso de diseño y desarrollo de software, ya que permite comprender y visualizar la lógica de interacción entre los diferentes componentes de un sistema. Proporciona una vista clara y estructurada de cómo se ejecutan las operaciones y cómo se comunican los objetos, lo que facilita la identificación de posibles problemas y la toma de decisiones informadas para optimizar el rendimiento y la eficiencia del sistema.

A continuación, se muestra un diagrama de secuencias sobre la inserción de usuarios por parte de un administrador.





### 2.2.3. Pruebas del sistema.

En el desarrollo de software, las pruebas son una parte fundamental para garantizar su correcto funcionamiento y evitar posibles errores en el futuro. A continuación, se describen algunas pruebas que podrían realizarse en el proyecto que nos atañe:

- Pruebas unitarias: se trata de pruebas automatizadas que se enfocan en verificar el correcto funcionamiento de los diferentes componentes o módulos de la aplicación de manera individual, sin interactuar con el resto del sistema.
- Pruebas de integración: se realizan para verificar que los diferentes módulos de la aplicación interactúan entre sí de manera correcta, asegurándose de que la información fluye de forma adecuada entre ellos.
- Pruebas de aceptación: son pruebas que se llevan a cabo para validar que el sistema cumple con los requisitos establecidos por el usuario. Estas pruebas pueden ser realizadas tanto por el cliente como por el equipo de desarrollo.

- Pruebas de carga: consisten en someter al sistema a cargas de trabajo extremas para evaluar su capacidad de respuesta y detectar posibles cuellos de botella en el rendimiento.
- Pruebas de seguridad: se realizan para detectar posibles vulnerabilidades en la aplicación y garantizar que los datos de los estudiantes estén protegidos de accesos no autorizados.

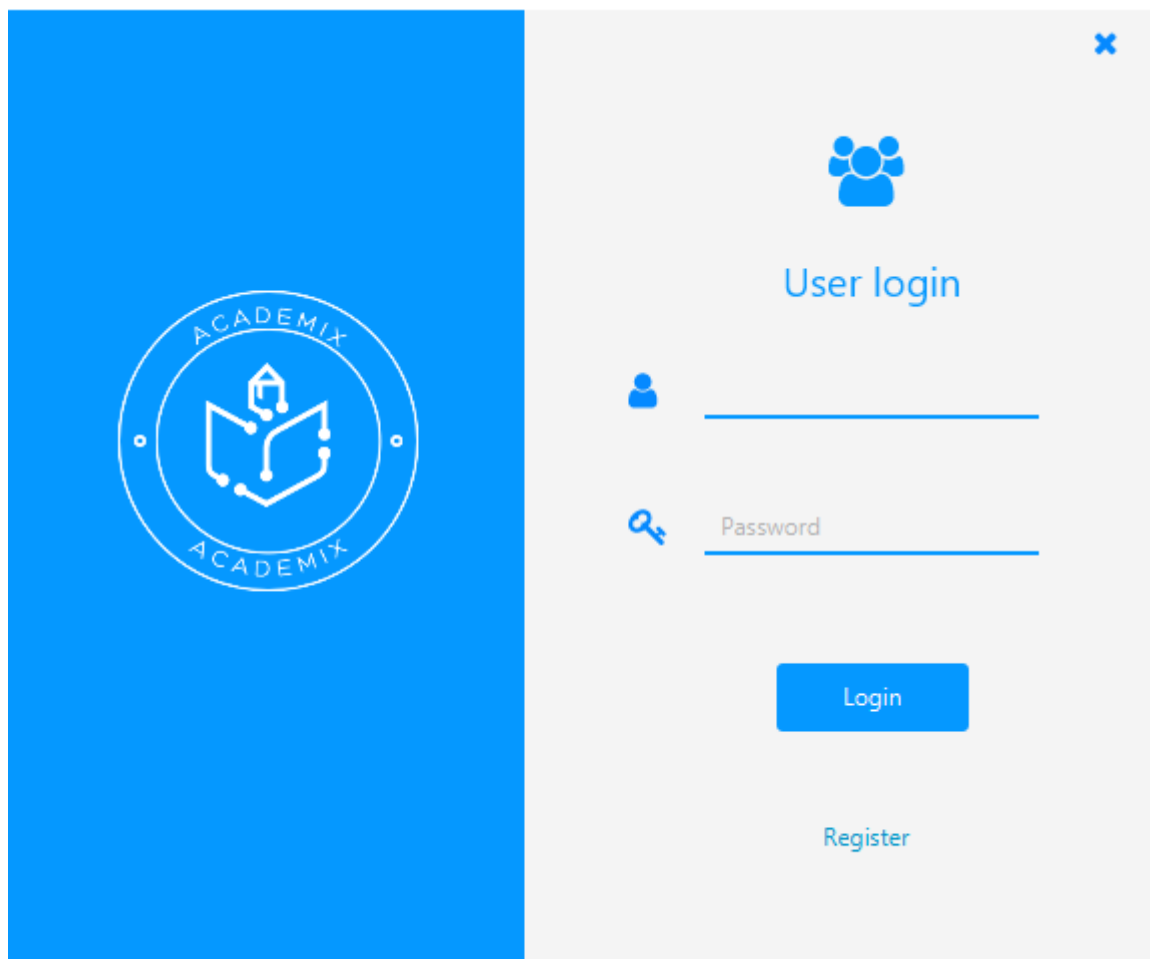
Estos son algunos ejemplos de pruebas que podrían realizarse en Academix para garantizar la calidad del software y su correcto funcionamiento. Es importante tener en cuenta que las pruebas deben ser realizadas de forma continua durante todo el proceso de desarrollo, y no solo al final del mismo, para detectar y solucionar posibles errores de manera temprana.

#### 2.2.4. Documentación.

- Login y registro.

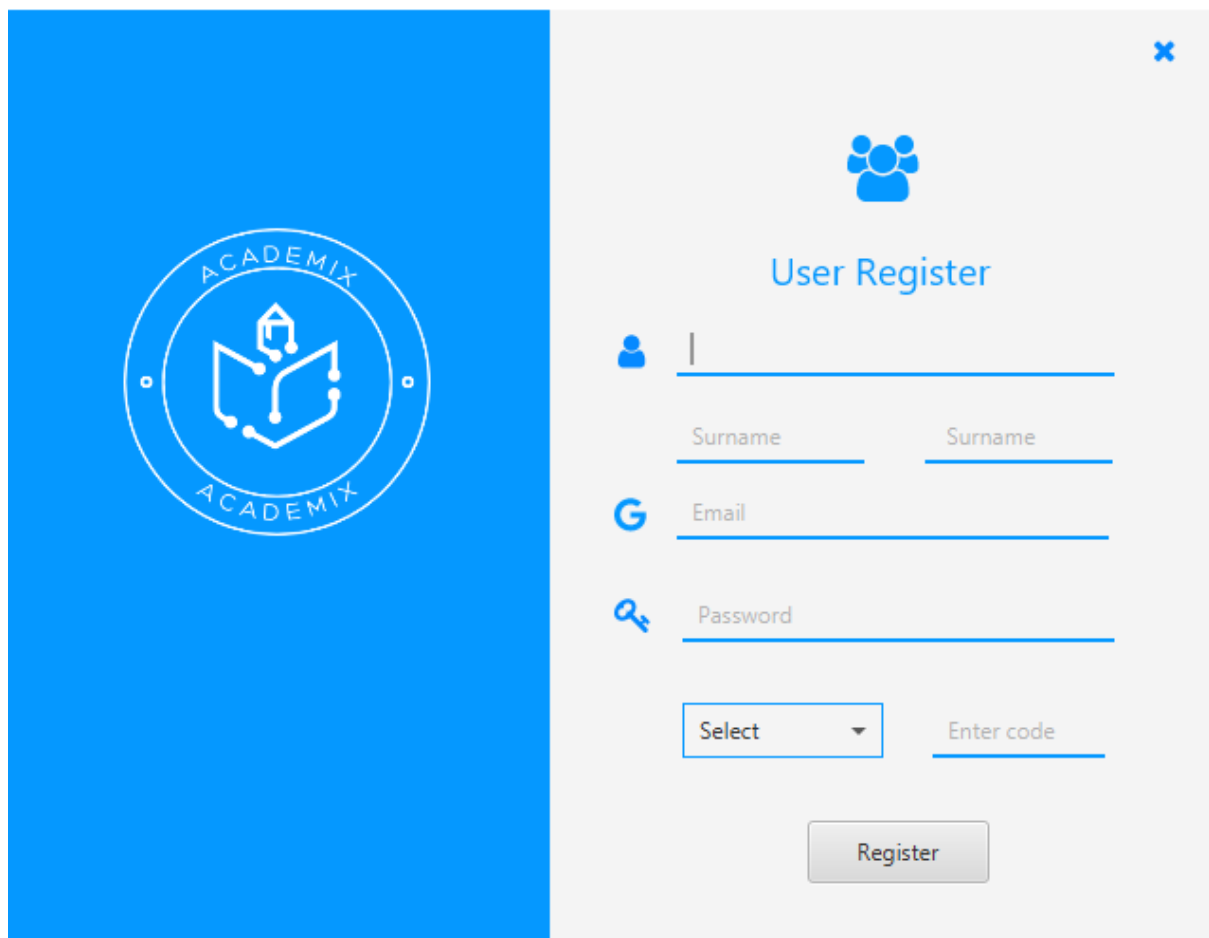
La interfaz de login de usuario es la primera pantalla que se inicia al abrir la aplicación.

En esta interfaz, el usuario debe ingresar sus datos de inicio de sesión, previamente registrados, para acceder a la aplicación. En caso de no ser un usuario registrado todavía en el sistema, se encuentra la opción de registrar, debajo del botón de login.



La interfaz de registro de usuario de Academix permite a los nuevos usuarios registrarse en la aplicación. Esta interfaz requiere que el usuario complete una serie de campos, incluyendo su nombre, apellidos, contraseña, correo electrónico y el rol que desempeña en la organización (administrador, profesor o alumno).

Además, se ha implementado un sistema de seguridad que exige a los usuarios introducir un código de acceso especial para poder registrarse en la aplicación. Esto es necesario para garantizar que solo aquellos usuarios con las credenciales apropiadas puedan darse de alta y acceder a la información sensible.



The image shows a 'User Register' form. On the left is a blue vertical bar with the ACADEMIX logo, which features a stylized book and circuitry inside a circle. The form itself is light gray and contains the following elements: a close button (X) in the top right corner; a group of three people icon; the title 'User Register'; a first name input field with a person icon; two 'Surname' input fields; an 'Email' input field with a 'G' icon; a 'Password' input field with a key icon; a dropdown menu labeled 'Select'; an 'Enter code' input field; and a 'Register' button at the bottom.

Una vez que el usuario ha completado los campos requeridos y ha introducido el código de acceso correcto, puede enviar la información de registro a través de un botón "Registrar". Si el proceso se completa con éxito, se redirige automáticamente al usuario a la interfaz de inicio de sesión. El usuario podrá acceder a la aplicación con un rol específico: administrador, o profesor.

A continuación, se detalla la guía de usuario para los roles indicados:

- **Administrador.**
  - Panel general.

Cuando el usuario inicia sesión como administrador y accede a la aplicación, se muestra el panel general de administración. En este panel, el administrador tiene disponible un menú de opciones que incluye: perfil, panel de administración, profesores, alumnos, calendario y política de privacidad. Según la opción que elija, el administrador accede a distintas opciones que le brinda la aplicación.



ACADEMIX



Perfil



Panel administración



Profesores



Alumnos



Calendario



Política de privacidad

- Perfil.



## Ajustes de perfil

### Detalles de usuario

Nombre

Primer apellido

Segundo apellido

Fecha de nacimiento

### Información de contacto

Número

Email


Ciudad

En la opción de perfil, el administrador puede ver y editar su información personal: nombre, apellidos, fecha de nacimiento, número de teléfono, correo electrónico y ciudad. Asimismo, el administrador puede cambiar su contraseña desde esta pantalla en la opción: contraseña, y, volver al panel general en la opción: inicio.

- Panel de administración.

En el panel de administración, el administrador puede gestionar la información de la organización educativa. De esta manera, puede crear usuarios, asignaturas y grupos de alumnos, asignando su tutor correspondiente.

ADMINISTRADOR: Ricardo Canada Gracia



Agregar usuario

Categoría

Nombre

Apellido

Apellido

Contraseña

Email

Agregar

Agregar grupo

Curso

Letra

Tutor

Agregar

Curso	Letra	Tutor
1º ESO	A	-
1º ESO	B	Juan Gómez Pérez
1º ESO	C	

Agregar asignatura

Nueva asignatura

Seleccionar

Profesor

Agregar

Asignatura	Profesor
Matemáticas	Juan Pablo Asier Ramos
Historia de Aragón	Juan Gómez Pérez
Ética	Juan Gómez Pérez

Modificar / Eliminar


Seleccionar

Seleccionar


Aceptar

Por otro lado, desde esta pantalla el usuario puede acceder a la gestión de la información de la información de usuarios, grupos y asignaturas, a través de las opciones disponibles en: Modificar / eliminar.

ADMINISTRADOR: Ricardo Canada Gracia



Ricardo Canada Gracia



PANEL DE ADMINISTRACIÓN

Agregar usuario

Categoría

Nombre

Apellido

Apellido

Contraseña

Email

Agregar

Agregar grupo

Curso

Letra

Tutor

Agregar

Curso	Letra	Tutor
1º ESO	A	-
1º ESO	B	Juan Gómez Pérez
1º ESO	C	-

Agregar asignatura

Nueva asignatura

Seleccionar

Profesor

Agregar

Asignatura	Profesor
Matemáticas	Juan Pablo Asier Ramos
Historia de Aragón	Juan Gómez Pérez
Ética	Juan Gómez Pérez

Modificar / Eliminar

Seleccionar

Seleccionar

Aceptar

Seleccionar

Usuario

Grupo


Asignatura

- Profesores.

En la sección de profesores, el administrador puede visualizar los profesores del centro académico, así como los siguientes datos: nombre, apellidos, correo electrónico, si es tutor, y el grupo del que es tutor.







ADMINISTRADOR: Ricardo Canada Gracia

PANEL GESTIÓN ALUMNOS

Grupo

Filtrar

Nombre	Primer apellido	Segundo apellido	Correo	Tutor	Grupo
Ana	Fernández	Fernández	anafd@gmail...	Juan Gómez Pérez	1º ESO B
Carlos	Martínez	Ramírez	carlos@gmail...	Laura Rodríguez F...	2º ESO A
Laura	Hernández	Jiménez	laura@gmail.c...	Laura Rodríguez F...	2º ESO A
David	Pérez	Sánchez	david@gmail...	Juan Gómez Pérez	1º ESO B
Carmen	Ruiz	López	carmen@gma...	Juan Gómez Pérez	1º ESO B
Sergio	González	Martínez	sergio@gmail...	Juan Gómez Pérez	1º ESO B
Isabel	Torres	Hernández	isabel@gmail...	-	-
Pablo	Soto	García	pablo@gmail...	-	-
Elena	Vargas	González	elena@gmail...	-	-
Luis	Morales	Sánchez	luis@gmail.com	-	-
Marina	Navarro	Molina	marina@gmai...	-	-
Hugo	Díaz	Pérez	hugo@gmail...	-	-
Marta	Fernández	González	marta@gmail...	-	-
Carlos	Ramírez	Martínez	carlos@gmail...	-	-
Laura	Gutiérrez	López	laura@gmail.c...	-	-

Modificar

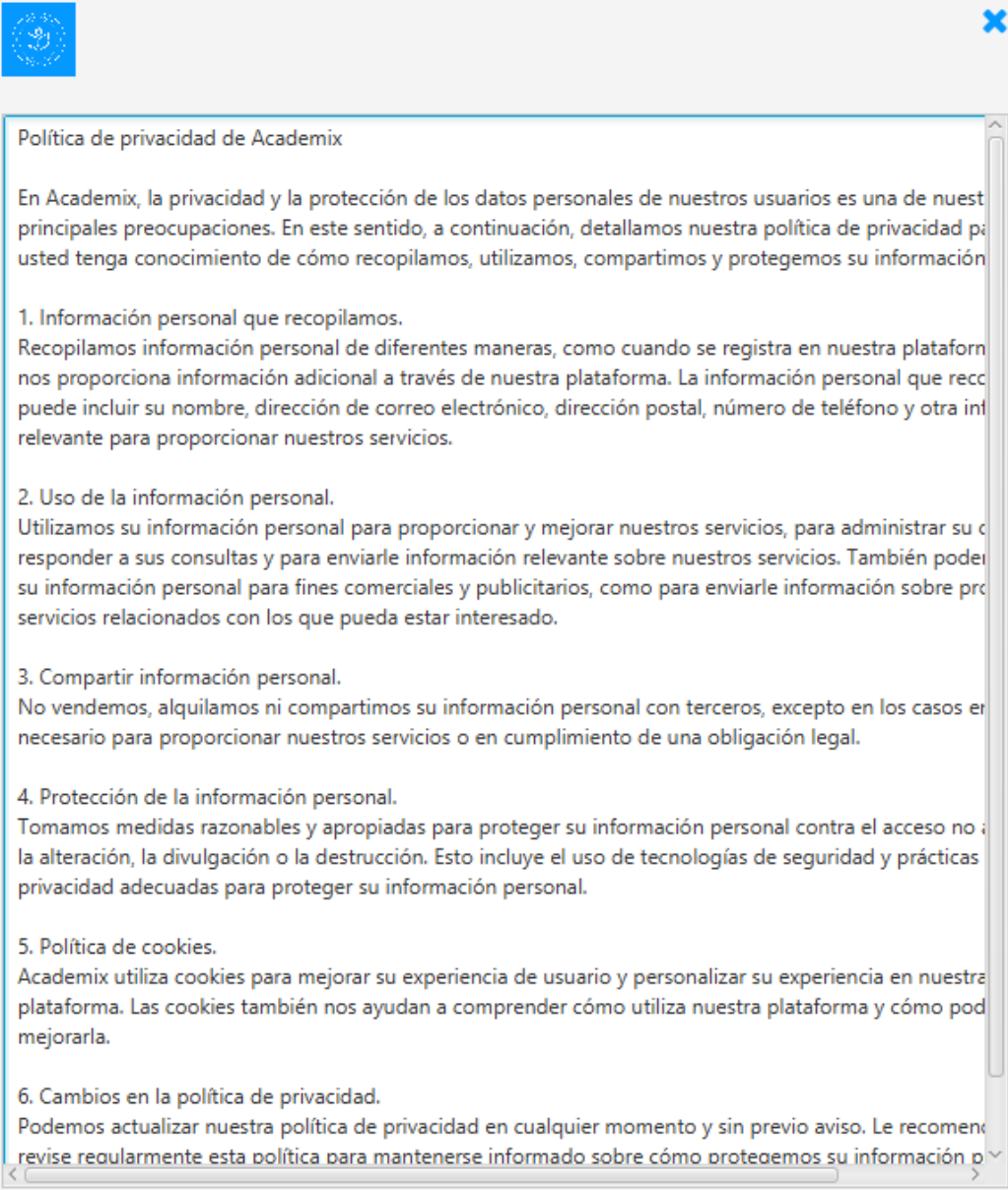
Eliminar

#### - Calendario

En la sección de calendario, el administrador puede gestionar las fechas importantes para la organización, como los periodos de matriculación, los exámenes, las vacaciones, etc.

- Política de privacidad.

Por último, en la opción de política de privacidad, el administrador puede ver la política de privacidad de la aplicación.



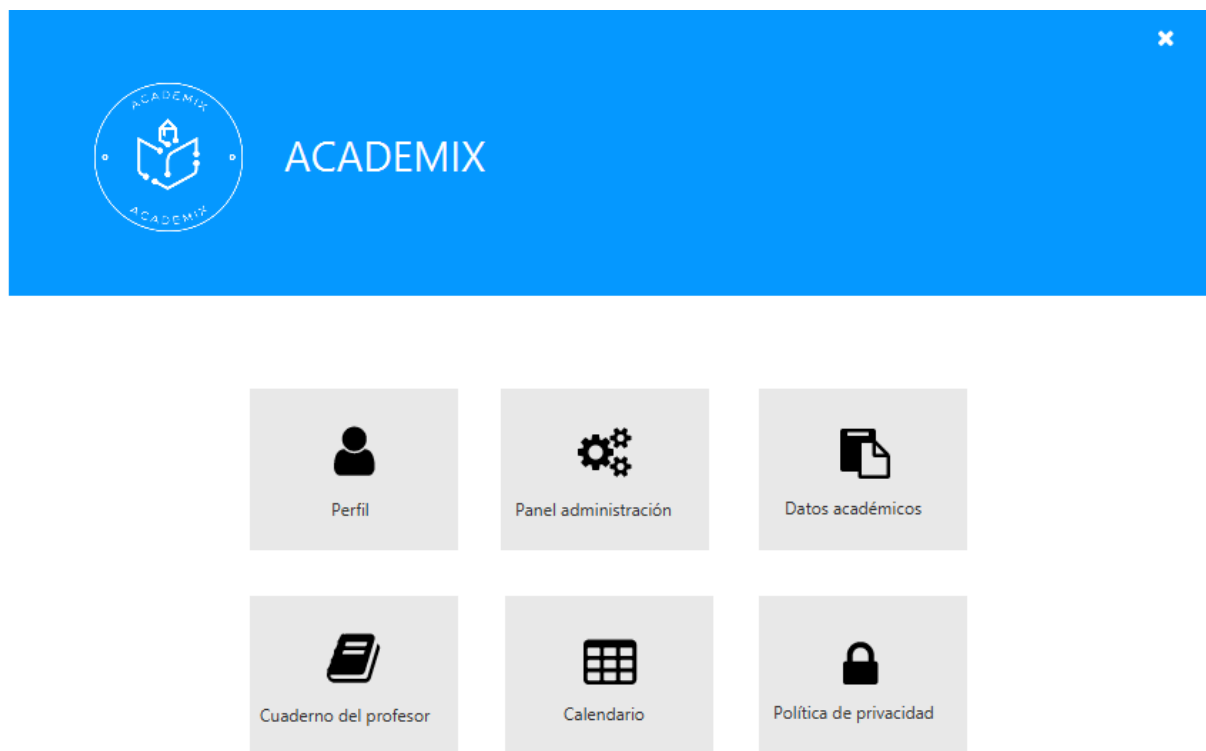
Política de privacidad de Academix

En Academix, la privacidad y la protección de los datos personales de nuestros usuarios es una de nuestras principales preocupaciones. En este sentido, a continuación, detallamos nuestra política de privacidad para que usted tenga conocimiento de cómo recopilamos, utilizamos, compartimos y protegemos su información.

- 1. Información personal que recopilamos.**  
Recopilamos información personal de diferentes maneras, como cuando se registra en nuestra plataforma. También nos proporciona información adicional a través de nuestra plataforma. La información personal que recopilamos puede incluir su nombre, dirección de correo electrónico, dirección postal, número de teléfono y otra información relevante para proporcionar nuestros servicios.
- 2. Uso de la información personal.**  
Utilizamos su información personal para proporcionar y mejorar nuestros servicios, para administrar su cuenta, responder a sus consultas y para enviarle información relevante sobre nuestros servicios. También podemos utilizar su información personal para fines comerciales y publicitarios, como para enviarle información sobre productos y servicios relacionados con los que pueda estar interesado.
- 3. Compartir información personal.**  
No vendemos, alquilamos ni compartimos su información personal con terceros, excepto en los casos en los que es necesario para proporcionar nuestros servicios o en cumplimiento de una obligación legal.
- 4. Protección de la información personal.**  
Tomamos medidas razonables y apropiadas para proteger su información personal contra el acceso no autorizado, la alteración, la divulgación o la destrucción. Esto incluye el uso de tecnologías de seguridad y prácticas de privacidad adecuadas para proteger su información personal.
- 5. Política de cookies.**  
Academix utiliza cookies para mejorar su experiencia de usuario y personalizar su experiencia en nuestra plataforma. Las cookies también nos ayudan a comprender cómo utiliza nuestra plataforma y cómo podemos mejorarla.
- 6. Cambios en la política de privacidad.**  
Podemos actualizar nuestra política de privacidad en cualquier momento y sin previo aviso. Le recomendamos que revise regularmente esta política para mantenerse informado sobre cómo protegemos su información personal.

- Profesor.

La interfaz del profesor también cuenta con un panel general desde donde el usuario puede acceder a las opciones disponibles según su rol. En este caso, el profesor tiene las siguientes opciones: perfil, panel de administración, datos académicos, cuaderno del profesor, calendario y política de privacidad.




- Perfil.

En la opción de perfil, el profesor puede ver y editar su información personal: nombre, apellidos, fecha de nacimiento, número de teléfono, correo electrónico y ciudad. Asimismo, el administrador puede cambiar su contraseña desde esta pantalla en la opción: contraseña, y, volver al panel general en la opción: inicio.

- Panel de administración.

En el panel de administración, el profesor puede crear y eliminar actividades y exámenes, vinculadas a un grupo y una asignatura. Asimismo, puede visualizar las tareas y exámenes creados por grupo y asignatura.

Por otro lado, desde esta pantalla también puede acceder a los datos académicos, desde el botón: Gestión datos académicos.



**PROFESOR:** Juan Pablo Asier Ramos

Nueva actividad

Sintaxis

Examen

Lengua Castell...

1º ESO A

25

Porcentaje

Aceptar

Todos

Todas

Tareas evaluables

Buscar


TAREAS

Actividad	Categoría	Porcentaje	Grupo	Asignatura
La Colmena	Tarea evaluable	55	-	Lengua Castellana y Literatura
Lectura obligatoria	Tarea evaluable	10	-	Lengua Castellana y Literatura

Gestión datos académicos

Eliminar

Juan Pablo Asier Ramos



- En la sección de datos académicos, el profesor puede visualizar, editar y eliminar las notas vinculadas a un grupo, asignatura, tarea y /o examen. Así, puede realizar el seguimiento de los mismos, pudiendo ver su progreso en las diferentes asignaturas.

[illegible]

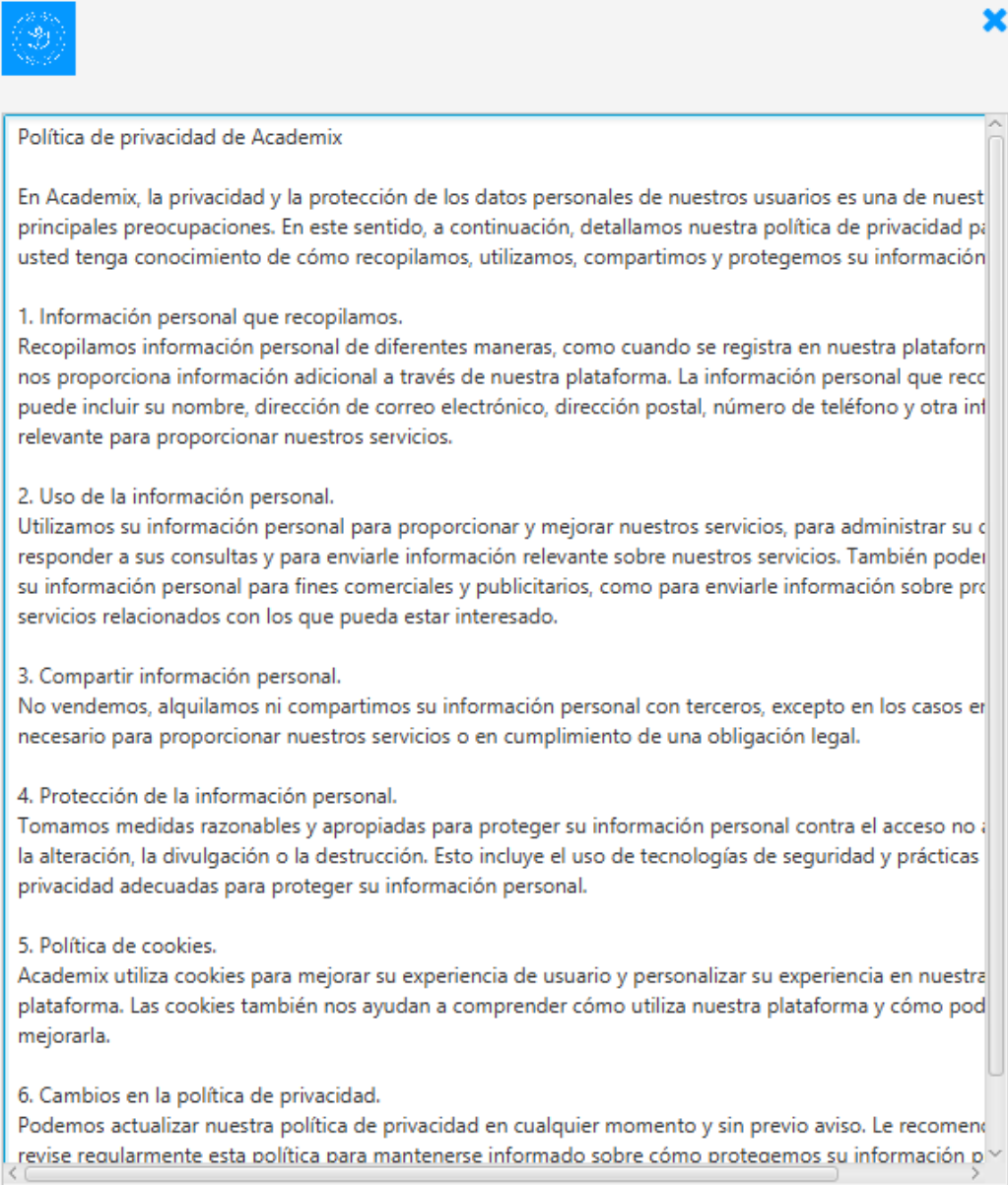
En el cuaderno del profesor, el usuario puede detallar el registro del seguimiento de los alumnos., indicando, para cada día, si ha realizado las tareas de casa, las tareas de clase, su participación y atención. Asimismo, puede modificar, eliminar y exportar los registros que haya creado sobre el seguimiento de cada alumno.

Juan Pablo Asier Ramos

La sección de calendario/horario está diseñada para que el profesor pueda organizar sus clases y horarios de manera efectiva y eficiente. Desde aquí, el profesor puede visualizar su horario de clases, así como asignar y modificar las clases asignadas a su grupo de alumnos.

- Política de privacidad.

Finalmente, del mismo modo que en la interfaz de administrador, también está disponible la política de privacidad, donde se detallan las normas y políticas de privacidad de la aplicación.



**Política de privacidad de Academix**

En Academix, la privacidad y la protección de los datos personales de nuestros usuarios es una de nuestras principales preocupaciones. En este sentido, a continuación, detallamos nuestra política de privacidad para que usted tenga conocimiento de cómo recopilamos, utilizamos, compartimos y protegemos su información.

- 1. Información personal que recopilamos.**  
Recopilamos información personal de diferentes maneras, como cuando se registra en nuestra plataforma. También nos proporciona información adicional a través de nuestra plataforma. La información personal que recopila puede incluir su nombre, dirección de correo electrónico, dirección postal, número de teléfono y otra información relevante para proporcionar nuestros servicios.
- 2. Uso de la información personal.**  
Utilizamos su información personal para proporcionar y mejorar nuestros servicios, para administrar su cuenta, responder a sus consultas y para enviarle información relevante sobre nuestros servicios. También podemos utilizar su información personal para fines comerciales y publicitarios, como para enviarle información sobre productos y servicios relacionados con los que pueda estar interesado.
- 3. Compartir información personal.**  
No vendemos, alquilamos ni compartimos su información personal con terceros, excepto en los casos en los que es necesario para proporcionar nuestros servicios o en cumplimiento de una obligación legal.
- 4. Protección de la información personal.**  
Tomamos medidas razonables y apropiadas para proteger su información personal contra el acceso no autorizado, la alteración, la divulgación o la destrucción. Esto incluye el uso de tecnologías de seguridad y prácticas de privacidad adecuadas para proteger su información personal.
- 5. Política de cookies.**  
Academix utiliza cookies para mejorar su experiencia de usuario y personalizar su experiencia en nuestra plataforma. Las cookies también nos ayudan a comprender cómo utiliza nuestra plataforma y cómo podemos mejorarla.
- 6. Cambios en la política de privacidad.**  
Podemos actualizar nuestra política de privacidad en cualquier momento y sin previo aviso. Le recomendamos que revise regularmente esta política para mantenerse informado sobre cómo protegemos su información personal.



### 2.3. Metodología.

En el contexto del desarrollo de software, la metodología se refiere al conjunto de prácticas, técnicas y herramientas que se utilizan para planificar, diseñar, implementar, probar y mantener un sistema de software. La elección de una metodología adecuada es fundamental para el éxito del proyecto, ya que puede influir en la calidad del software, el tiempo y los recursos necesarios para completar el proyecto, así como en la satisfacción del cliente y de los usuarios finales.

Existen muchas metodologías de desarrollo de software, cada una con sus propias características y enfoques. En el caso de la aplicación que nos atañe, se ha seguido una metodología tradicional en cascada.

La metodología en cascada es un enfoque de desarrollo de software secuencial y lineal, en el cual las fases del proyecto se llevan a cabo en un orden secuencial fijo. El proceso comienza con la definición de requisitos y diseño, seguido de la implementación, pruebas y finalmente la entrega del software al cliente.

La elección de la metodología en cascada para el desarrollo de la aplicación Academix se debe a que la metodología en cascada es especialmente adecuada para proyectos en los que los requisitos están claramente definidos desde el principio y no se esperan cambios significativos a lo largo del desarrollo. En el caso de Academix, se tiene una visión definida de la naturaleza, requisitos del proyecto, y funcionalidad de la aplicación. Asimismo, se requiere que se entreguen resultados en etapas específicas, asegurando la calidad del software en cada una de ellas.

En consiguiente, esto permite una gestión óptima del presupuesto y los plazos, ya que cada fase del desarrollo se lleva a cabo de forma secuencial y se espera que se complete antes de avanzar a la siguiente. Esto garantiza que se cumpla con la calidad del software y que se alcancen los objetivos del proyecto, minimizando los riesgos, ya que se pueden identificar y tratar posibles problemas en etapas tempranas del desarrollo.

En resumen, la elección de la metodología en cascada para el desarrollo de la aplicación Academix se debe a que ofrece un enfoque estructurado y secuencial en la gestión del desarrollo de software, y a su adecuación a proyectos con requisitos claramente definidos y estructurados; ya que permite estipular claramente el alcance del proyecto, identificar los requerimientos y especificaciones, y planificar las tareas y entregables necesarios para alcanzar los objetivos del proyecto, y gestionar los recursos y riesgos asociados al desarrollo.

## 2.4. Recursos.

### 2.4.1. Materiales.

Para el desarrollo de la aplicación Academix, se necesitan varios recursos materiales, entre los cuales se pueden destacar:

- Hardware: se necesitará una serie de ordenadores de alto rendimiento y capacidad que permita ejecutar de forma fluida el software de desarrollo y las herramientas necesarias para el proyecto; así como otros dispositivos (monitores, teclados, ratones, etc.) para el equipo de desarrollo.
- Software: para el desarrollo de la aplicación, se necesitará un software de desarrollo, por lo que es necesario disponer de una licencia o una versión gratuita del mismo.
- Servidores: se requerirán servidores para alojar y gestionar la base de datos y los archivos de la aplicación, así como para implementar la aplicación en la nube o en servidores dedicados. En este caso, los servidores serán externalizados, por lo que será un coste a asumir.
- Equipo de pruebas: se necesitará contar con varios dispositivos con distintos sistemas operativos y versiones de navegadores para realizar pruebas exhaustivas.

### 2.4.2. Infraestructura y equipamiento.

En virtud de las necesidades del proyecto, en el cual se ha decidido contratar a un desarrollador freelance para llevar a cabo la implementación de la aplicación y posteriormente encargarse del soporte, se hace necesario revisar y ajustar las necesidades de equipamiento e infraestructura a estas circunstancias.

En lo referente al equipamiento y la infraestructura requeridos para el trabajador freelance, es menester considerar un enfoque distinto debido a su condición de teletrabajo. En lugar de un espacio físico asignado en un local comercial u oficina, el desarrollador establecerá su propio entorno laboral en el lugar que considere adecuado. A pesar de ello, es fundamental garantizar que se cumplan ciertos requisitos esenciales para asegurar su productividad y eficacia en el desarrollo de la aplicación.

En primer lugar, el desarrollador debe contar con un espacio de trabajo adecuado para poder hacer uso de los recursos materiales necesarios: un ordenador, periféricos esenciales, como teclado, ratón y auriculares, pantallas adicionales, etc.

En cuanto a la conectividad, se debe asegurar una conexión a internet estable y de alta velocidad. Esto permitirá al desarrollador acceder de forma ágil a los recursos en línea necesarios para su trabajo, así como establecer comunicación fluida con el equipo y los demás actores involucrados en el proyecto.

Con respecto a la seguridad de los datos, si bien los servidores estarán externalizados, es fundamental establecer medidas de protección adecuadas. Se debe hacer uso de una configuración de red segura, que incluya el uso de VPN (Red Privada Virtual) para garantizar una conexión cifrada y proteger la confidencialidad de la información sensible. Además, se deben implementar políticas de seguridad rigurosas y seguir las mejores prácticas en cuanto al manejo y resguardo de los datos.

Una vez desarrollada la aplicación, dado que el desarrollador estará trabajando de manera independiente, se requiere un sistema de soporte técnico remoto que esté disponible para brindar asistencia en caso de que surjan problemas o dificultades durante el desarrollo de la aplicación. Esta asistencia técnica debe ser eficiente y rápida, para minimizar cualquier impacto negativo en la experiencia del cliente

En conclusión, las necesidades de equipamiento e infraestructura para el trabajador freelance se centran en proporcionar un entorno de trabajo óptimo desde su propio espacio laboral. Esto implica garantizar un dispositivo informático adecuado, una conexión a internet estable y segura, medidas de seguridad de datos robustas y un sistema de soporte técnico remoto eficiente. Al cumplir con estas necesidades, se promueve la eficacia y productividad del desarrollador, a la vez que se maximizan los recursos disponibles y se reducen los costos asociados a la infraestructura física.

#### 2.4.3. Humanos.

Para el desarrollo de esta aplicación, se ha requerido la participación de un programador freelance con conocimientos en Java y JavaFX, y en el desarrollo de bases de datos con MySQL. Además, es necesario que el programador utilice herramientas de pruebas y depuración de código, para garantizar la calidad del producto final.

También es importante que el programador tenga habilidades de comunicación y trabajo en equipo, ya que se deberá coordinar con el cliente para establecer los requisitos y mantener una comunicación constante durante el proceso de desarrollo.

Por último, se valorará positivamente la capacidad de innovación y creatividad del programador para proponer soluciones efectivas a los desafíos que puedan surgir durante el desarrollo del proyecto.

Asimismo, una vez entregada la aplicación al cliente, el mismo desarrollador será el encargado del soporte de la aplicación, en caso de que fuera requerido por el cliente.

### 3. Presupuesto.

#### 3.1. Gastos.

En el presente informe, se realiza una estimación de los gastos<sup>4</sup> asociados al desarrollo de la aplicación durante un período de dos meses. Se considerarán los siguientes aspectos: el sueldo del desarrollador freelance, externalización de servidores y posibles gastos asociados. A continuación, se proporciona información sobre los gastos estimados.

En lo referente al sueldo del desarrollador, éste constituirá una partida importante en el presupuesto del proyecto. Se estiman unos gastos de 2.000 euros mensuales. Por tanto, se debe asignar una cantidad específica de 4.000 para cubrir los servicios del desarrollador durante el período de desarrollo de dos meses.

Por otro lado, dada la decisión de externalizar los servidores, se optará por hacer uso de los servicios de una empresa especializada en servicios de alojamiento. Los costos asociados a esta externalización se estiman en una tarifa mensual que podría situarse en torno a 200 euros mensuales, lo que supondría 400 euros para los dos meses de desarrollo. A partir de la entrega de la aplicación al cliente, el coste de la externalización de los servidores sería asumido por el cliente.

Además de los gastos mencionados, es necesario considerar otros aspectos adicionales que pueden generar costos. Estos incluyen la adquisición de licencias de software especializado, herramientas de desarrollo y control de versiones, así como los servicios de pruebas y calidad para asegurar el correcto funcionamiento de la aplicación. Asimismo, se deben considerar los costos asociados a medidas de seguridad y protección de datos, registro de dominio y diseño de la interfaz gráfica, entre otros. Estos gastos pueden variar significativamente según las necesidades surgidas del proyecto; por ello, se asigna un presupuesto aproximado de 1000 euros para cubrir necesidades adicionales.

En resumen, la estimación de gastos para el desarrollo de la aplicación durante dos meses asciende a 5.400 euros.

---

<sup>4</sup> Véase: [Anexos: tablas de gastos](#).

### 3.2. Ingresos.

En el ámbito de la comercialización de la aplicación desarrollada para el grupo educativo Círculo Aurora, se vislumbra un amplio alcance debido a la extensa red de centros educativos que conforman este grupo educativo. Esta circunstancia genera una demanda sustancial de la aplicación, ya que su implantación en todos los centros del grupo resulta imperativa para una gestión educativa eficiente y unificada.

Conscientes de la relevancia y el valor estratégico de esta aplicación en el ámbito educativo, se ha establecido un precio de venta estimado en 25.000 euros. Dicha cifra ha sido rigurosamente determinada tras considerar diversos factores, como la amplitud y complejidad de la solución proporcionada, la cantidad de usuarios que se beneficiarán de la aplicación en cada centro y los beneficios tangibles e intangibles que conlleva su implementación en la red de colegios del grupo.

### 3.3. Beneficios.

Una vez estimados los gastos e ingresos derivados del desarrollo de la aplicación, se considera que se podría obtener un beneficio de 20.100 euros, resultantes de descontar los gastos (4.900) de los ingresos (25.000).

## 4. Conclusiones.

El desarrollo de Academix como trabajo de fin de grado ha sido una experiencia enriquecedora que ha contribuido significativamente a mi formación como programador. A través de este proyecto, he tenido la oportunidad de aplicar y consolidar mis conocimientos en la implementación de un modelo MVC (Modelo-Vista-Controlador) junto con la utilización de patrones de diseño, como DAOs (Objetos de Acceso a Datos). Esto me ha permitido comprender en profundidad la estructura y organización de una aplicación robusta y escalable.

Además, el desarrollo de Academix me ha brindado la oportunidad de aprender a manejar herramientas y tecnologías relevantes en el campo de la programación, como JavaFX para la creación de interfaces gráficas de usuario y MySQL para la gestión de la base de datos.

Asimismo, la experiencia de trabajar en este proyecto, ha contribuido a mejorar mi capacidad de análisis, diseño y resolución de problemas, ya que he tenido que enfrentar diferentes retos y tomar decisiones informadas en cuanto a la arquitectura y funcionalidades de la aplicación.

En definitiva, el desarrollo de Academix como parte de mi trabajo de fin de grado ha sido una oportunidad valiosa que ha contribuido significativamente a mi formación como desarrollador, permitiéndome aplicar conocimientos teóricos y adquirir nuevas habilidades y competencias necesarias en el campo de la programación y la gestión de proyectos informáticos.

## 5. Bibliografía.

(n.d.). PMOInformatica.com, La Oficina de Proyectos de Informática. Retrieved 2023, from <http://www.pmoinformatica.com/2017/02/requerimientos-funcionales-ejemplos.html>

. (2019, May 7). - YouTube. Retrieved May, 2023, from <http://universitatcarlemany.com/actualidad/blog/metodologias-de-desarrollo-de-software/>

*Diagrama de casos de uso.* (n.d.). Wikipedia. Retrieved May 05, 2023, from [https://es.wikipedia.org/wiki/Diagrama\\_de\\_casos\\_de\\_uso](https://es.wikipedia.org/wiki/Diagrama_de_casos_de_uso)

*Diagrama de casos de uso: estructura y función.* (2020, July 05). IONOS. Retrieved May 22, 2023, from <https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/diagrama-de-casos-de-uso/>

*Diagrama de clases.* (n.d.). Wikipedia. Retrieved May 05, 2023, from [https://es.wikipedia.org/wiki/Diagrama\\_de\\_clases](https://es.wikipedia.org/wiki/Diagrama_de_clases)

▷ *Diagrama de clases. Teoría y ejemplos.* (n.d.). diagramas UML. Retrieved May 05, 2023, from <https://diagramasuml.com/diagrama-de-clases/>

*Diagrama de Gantt: qué es y cómo crear uno con ejemplos [2022] • Asana.* (2022, September 12). Asana. Retrieved May 02, 2023, from <https://asana.com/es/resources/gantt-chart-basics>

*Diagramas de secuencia: cómo crear diagramas con UML.* (2019, March 4). IONOS. Retrieved May 18, 2023, from <https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/diagramas-de-secuencia/>

*Diferencia entre requisitos funcionales y no funcionales en el desarrollo de software – CJava Perú.* (n.d.). CJava Perú. Retrieved 2023, from



<https://cjavaperu.com/2021/09/diferencia-entre-requisitos-funcionales-y-no-funcional-es-en-el-desarrollo-de-software/>

*El Mejor Tutorial de Diagramas de Clase Para Ayudar a Modelar sus Sistemas Fácilmente.*

(2022, September 05). Creately. Retrieved May 22, 2023, from

<https://creately.com/blog/es/diagramas/tutorial-de-diagrama-de-clases/>

*Hibernate | Marco de Desarrollo de la Junta de Andalucía.* (n.d.). Junta de Andalucía.

Retrieved May 02, 2023, from

<https://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/97>

*Las 5 Mejores Metodologías de Desarrollo de Software.* (2022, October 27). GooApps.

Retrieved May, 2023, from

<https://gooapps.es/2022/10/27/las-5-mejores-metodologias-de-desarrollo-de-software/>

*Metodología de desarrollo de software.* (n.d.). Wikipedia. Retrieved May, 2023, from

[https://es.wikipedia.org/wiki/Metodolog%C3%ADa\\_de\\_desarrollo\\_de\\_software](https://es.wikipedia.org/wiki/Metodolog%C3%ADa_de_desarrollo_de_software)

*Metodologías de desarrollo software | Blog Becas Santander.* (2020, December 21). Becas

Santander. Retrieved May, 2023, from

<https://www.becas-santander.com/es/blog/metodologias-desarrollo-software.html>

Pittet, S. (n.d.). *Los distintos tipos de pruebas en software.* Atlassian. Retrieved May 15,

2023, from

<https://www.atlassian.com/es/continuous-delivery/software-testing/types-of-software-testing>

*Pruebas del Sistema.* (n.d.). Manuel Cillero. Retrieved May 16, 2023, from

<https://manuel.cillero.es/doc/metodologia/metrica-3/tecnicas/pruebas/sistema/>

*¿Qué es la prueba de software y cómo funciona?* (n.d.). IBM. Retrieved May 15, 2023, from

<https://www.ibm.com/es-es/topics/software-testing>

*¿Qué es y para qué sirve un diagrama de Gantt?* (2021, August 18). Teamleader. Retrieved May 06, 2023, from <https://www.teamleader.es/blog/diagrama-de-gantt>

*Qué son los Requisitos Funcionales: Ejemplos, Definición, Guía Completa.* (n.d.). Visure Solutions. Retrieved 2023, from <https://visuresolutions.com/es/blog/functional-requirements/>

*Requerimientos Funcionales y No Funcionales, ejemplos y tips | by Requeridos Blog.* (2018, April 20). Medium. Retrieved 2023, from <https://medium.com/@requeridosblog/requerimientos-funcionales-y-no-funcionales-ejemplos-y-tips-aa31cb59b22a>

Solera, S. (2022, April 27). *Las mejores metodologías para un correcto desarrollo de software.* Occam Agencia Digital. Retrieved May, 2023, from <https://www.occamagenciadigital.com/blog/las-mejores-metodologias-para-un-correcto-desarrollo-de-software>

3. *Técnicas para Identificar Requisitos Funcionales y No Funcionales - Metodología Gestión de Requerimientos.* (n.d.). Google Sites. Retrieved 2023, from <https://sites.google.com/site/metodologiareq/capitulo-ii/tecnicas-para-identificar-requisitos-funcionales-y-no-funcionales>

*Tutorial de diagrama de clases UML.* (n.d.). Lucidchart. Retrieved May 05, 2023, from <https://www.lucidchart.com/pages/es/tutorial-de-diagrama-de-clases-uml>

*Tutorial de diagrama de secuencia UML.* (n.d.). Lucidchart. Retrieved May 18, 2023, from <https://www.lucidchart.com/pages/es/diagrama-de-secuencia>

*Tutorial de diagramas de casos de uso ( Guía con ejemplos ).* (2022, September 29). Creately. Retrieved May 05, 2023, from <https://creately.com/blog/es/diagramas/tutorial-diagrama-caso-de-uso/>

*Tutorial del diagrama de secuencia: Guía completa con ejemplos.* (2022, October 21).

Creately. Retrieved May 18, 2023, from

<https://creately.com/blog/es/diagramas/tutorial-del-diagrama-de-secuencia/>

## 6. Anexos.

### 6.1. Librerías utilizadas.

- JavaFX: se utilizan las dependencias de `javafx-controls`, `javafx-fxml` y `javafx-graphics` para proporcionar una interfaz gráfica interactiva y visualmente atractiva para la aplicación.
- ControlsFX: la dependencia `controlsfx` proporciona una amplia gama de controles y componentes adicionales para JavaFX, lo que permite mejorar la usabilidad y la experiencia del usuario en la aplicación.
- JFoenix: la dependencia `jfoenix` proporciona una biblioteca de componentes de interfaz de usuario de material design para JavaFX. Estos componentes siguen los principios de diseño de Material Design de Google, lo que agrega un aspecto moderno y elegante a la interfaz de la aplicación.
- MySQL Connector/J: la dependencia `mysql-connector-java` es un controlador JDBC que permite la conexión y comunicación con una base de datos MySQL. Se utiliza para realizar operaciones de persistencia y manipulación de datos en la base de datos de la aplicación Academix.
- Hibernate: `hibernate-core` proporciona el framework de persistencia Hibernate, que permite mapear objetos Java a tablas de base de datos y realizar operaciones de persistencia de manera eficiente. Hibernate simplifica la interacción con la base de datos y proporciona características avanzadas como caché, consultas optimizadas y gestión de transacciones.
- Jakarta Persistence API: `jakarta.persistence-api` proporciona la API de persistencia de Jakarta EE, que es un estándar para la persistencia de objetos en aplicaciones Java. Esta API define las interfaces y anotaciones para el mapeo objeto-relacional y la realización de operaciones de persistencia.
- Eclipse Persistence: `javax.persistence` proporciona las clases e interfaces de la especificación JPA (Java Persistence API), que define un estándar para el mapeo objeto-relacional en aplicaciones Java.
- FontAwesomeFX: `fontawesomefx-commons`, `fontawesomefx-fontawesome` y `fontawesomefx-controls` agregan soporte para el uso de iconos de FontAwesome en la interfaz de usuario de la aplicación. Estos iconos son ampliamente utilizados para mejorar la usabilidad y la estética de la interfaz.
- Charm Down: las dependencias `charm-down-plugin-display` y `charm` son proporcionadas por GluonHQ y permiten la integración de funcionalidades específicas de dispositivos móviles en la aplicación, como el acceso a los sensores y la capacidad de pantalla táctil.

## 6.2. Script de base de datos.

```
CREATE TABLE `tipo_usuario` (  
  `codigo` int NOT NULL AUTO_INCREMENT,  
  `tipo` varchar(50) NOT NULL,  
  PRIMARY KEY (`codigo`)  
) ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT CHARSET=utf8mb4  
COLLATE=utf8mb4_0900_ai_ci;
```

```
CREATE TABLE `administrador` (  
  `id` int NOT NULL AUTO_INCREMENT,  
  `nombre` varchar(255) DEFAULT NULL,  
  `primer_apellido` varchar(255) DEFAULT NULL,  
  `contrasegna` varchar(255) DEFAULT NULL,  
  `email` varchar(255) DEFAULT NULL,  
  `tipo_usuario` int DEFAULT NULL,  
  `segundo_apellido` varchar(255) DEFAULT NULL,  
  `fecha_nacimiento` varchar(255) DEFAULT NULL,  
  `telefono` varchar(255) DEFAULT NULL,  
  `direccion` varchar(255) DEFAULT NULL,  
  PRIMARY KEY (`id`),  
  KEY `fk_administrador_tipo_usuario` (`tipo_usuario`),  
  CONSTRAINT `fk_administrador_tipo_usuario` FOREIGN KEY (`tipo_usuario`)  
REFERENCES `tipo_usuario` (`codigo`)  
) ENGINE=InnoDB AUTO_INCREMENT=5 DEFAULT CHARSET=utf8mb4  
COLLATE=utf8mb4_0900_ai_ci;
```

```
CREATE TABLE `alumno` (  
  `id` int NOT NULL AUTO_INCREMENT,  
  `grupo_id` int DEFAULT NULL,  
  `tipo_usuario` int DEFAULT NULL,  
  `nombre` varchar(255) DEFAULT NULL,  
  `primer_apellido` varchar(255) DEFAULT NULL,  
  `contrasegna` varchar(255) DEFAULT NULL,  
  `email` varchar(255) DEFAULT NULL,  
  `segundo_apellido` varchar(255) DEFAULT NULL,  
  `fecha_nacimiento` varchar(255) DEFAULT NULL,  
  `telefono` varchar(255) DEFAULT NULL,  
  `direccion` varchar(255) DEFAULT NULL,  
  PRIMARY KEY (`id`),  
  KEY `grupo_id` (`grupo_id`),  
  KEY `fk_alumno_tipo_usuario` (`tipo_usuario`),  
  CONSTRAINT `alumno_ibfk_1` FOREIGN KEY (`grupo_id`) REFERENCES `grupo` (`id`),
```

```

        CONSTRAINT `fk_alumno_tipo_usuario` FOREIGN KEY (`tipo_usuario`) REFERENCES
`tipo_usuario` (`codigo`)
)    ENGINE=InnoDB    AUTO_INCREMENT=36    DEFAULT    CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;

```

```

CREATE TABLE `profesor` (
    `id` int NOT NULL AUTO_INCREMENT,
    `nombre` varchar(255) DEFAULT NULL,
    `primer_apellido` varchar(255) DEFAULT NULL,
    `contrasegna` varchar(255) DEFAULT NULL,
    `email` varchar(255) DEFAULT NULL,
    `tipo_usuario` int DEFAULT NULL,
    `segundo_apellido` varchar(255) DEFAULT NULL,
    `fecha_nacimiento` varchar(255) DEFAULT NULL,
    `telefono` varchar(255) DEFAULT NULL,
    `direccion` varchar(255) DEFAULT NULL,
    PRIMARY KEY (`id`),
    KEY `fk_profesor_tipo_usuario` (`tipo_usuario`),
    CONSTRAINT `fk_profesor_tipo_usuario` FOREIGN KEY (`tipo_usuario`) REFERENCES
`tipo_usuario` (`codigo`)
)    ENGINE=InnoDB    AUTO_INCREMENT=14    DEFAULT    CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;

```

```

CREATE TABLE `asignatura` (
    `id` int NOT NULL AUTO_INCREMENT,
    `nombre` varchar(50) DEFAULT NULL,
    PRIMARY KEY (`id`)
)    ENGINE=InnoDB    AUTO_INCREMENT=27    DEFAULT    CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;

```

```

CREATE TABLE `grupo` (
    `id` int NOT NULL AUTO_INCREMENT,
    `curso` varchar(50) NOT NULL,
    `letra` varchar(50) NOT NULL,
    `tutor_id` int DEFAULT NULL,
    PRIMARY KEY (`id`),
    KEY `grupo_ibfk_1` (`tutor_id`),
    CONSTRAINT `grupo_ibfk_1` FOREIGN KEY (`tutor_id`) REFERENCES `profesor` (`id`)
)    ENGINE=InnoDB    AUTO_INCREMENT=11    DEFAULT    CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;

```

```

CREATE TABLE `examen` (
    `id` int NOT NULL AUTO_INCREMENT,
    `nombre` varchar(50) DEFAULT NULL,

```

```

`fecha` varchar(50) DEFAULT NULL,
`asignatura_id` int NOT NULL,
`grupo_id` int DEFAULT NULL,
`porcentaje` int DEFAULT NULL,
PRIMARY KEY (`id`),
KEY `asignatura_id` (`asignatura_id`),
KEY `grupo_id` (`grupo_id`),
CONSTRAINT `examen_ibfk_1` FOREIGN KEY (`asignatura_id`) REFERENCES
`asignatura` (`id`),
CONSTRAINT `examen_ibfk_3` FOREIGN KEY (`grupo_id`) REFERENCES `grupo` (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=8 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;

```

```

CREATE TABLE `tarea_evaluable` (
`id` int NOT NULL AUTO_INCREMENT,
`porcentaje` int DEFAULT NULL,
`asignatura_id` int DEFAULT NULL,
`grupo_id` int DEFAULT NULL,
`nombre` varchar(255) DEFAULT NULL,
PRIMARY KEY (`id`),
KEY `asignatura_id` (`asignatura_id`),
KEY `grupo_id` (`grupo_id`),
CONSTRAINT `tarea_evaluable_ibfk_1` FOREIGN KEY (`asignatura_id`) REFERENCES
`asignatura` (`id`),
CONSTRAINT `tarea_evaluable_ibfk_2` FOREIGN KEY (`grupo_id`) REFERENCES
`grupo` (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=12 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;

```

```

CREATE TABLE `profesor_asignatura` (
`id_profesor` int NOT NULL,
`id_asignatura` int NOT NULL,
`agno_academico` varchar(25) DEFAULT NULL,
PRIMARY KEY (`id_profesor`,`id_asignatura`),
KEY `id_asignatura` (`id_asignatura`),
CONSTRAINT `profesor_asignatura_ibfk_1` FOREIGN KEY (`id_profesor`) REFERENCES
`profesor` (`id`),
CONSTRAINT `profesor_asignatura_ibfk_2` FOREIGN KEY (`id_asignatura`)
REFERENCES `asignatura` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;

```

```

CREATE TABLE `asignatura_grupo` (
`asignatura_id` int NOT NULL,
`grupo_id` int NOT NULL,

```

```

PRIMARY KEY (`asignatura_id`,`grupo_id`),
KEY `grupo_id` (`grupo_id`),
CONSTRAINT `asignatura_grupo_ibfk_1` FOREIGN KEY (`asignatura_id`) REFERENCES
`asignatura` (`id`),
CONSTRAINT `asignatura_grupo_ibfk_2` FOREIGN KEY (`grupo_id`) REFERENCES
`grupo` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;

```

```

CREATE TABLE `cuaderno_profesor` (
  `id` int NOT NULL AUTO_INCREMENT,
  `fecha_insercion` varchar(255) NOT NULL,
  `alumno_id` int NOT NULL,
  `tareas_casa` varchar(255) DEFAULT NULL,
  `participacion` varchar(255) DEFAULT NULL,
  `atencion` varchar(255) DEFAULT NULL,
  `tareas_clase` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `alumno_id` (`alumno_id`),
  CONSTRAINT `cuaderno_profesor_ibfk_1` FOREIGN KEY (`alumno_id`) REFERENCES
`alumno` (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=10 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;

```

```

CREATE TABLE `nota_tarea_evaluable` (
  `id` int NOT NULL AUTO_INCREMENT,
  `puntuacion` double DEFAULT NULL,
  `tarea_evaluable_id` int DEFAULT NULL,
  `alumno_id` int DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `tarea_evaluable_id` (`tarea_evaluable_id`),
  KEY `alumno_id` (`alumno_id`),
  CONSTRAINT `nota_tarea_evaluable_alumno_fk` FOREIGN KEY (`alumno_id`)
REFERENCES `alumno` (`id`),
CONSTRAINT `nota_tarea_evaluable_tarea_evaluable_fk` FOREIGN KEY
(`tarea_evaluable_id`) REFERENCES `tarea_evaluable` (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=7 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;

```

```

CREATE TABLE `nota_examen` (
  `id` int NOT NULL AUTO_INCREMENT,
  `puntuacion` double DEFAULT NULL,
  `examen_id` int DEFAULT NULL,
  `alumno_id` int DEFAULT NULL,
  PRIMARY KEY (`id`),

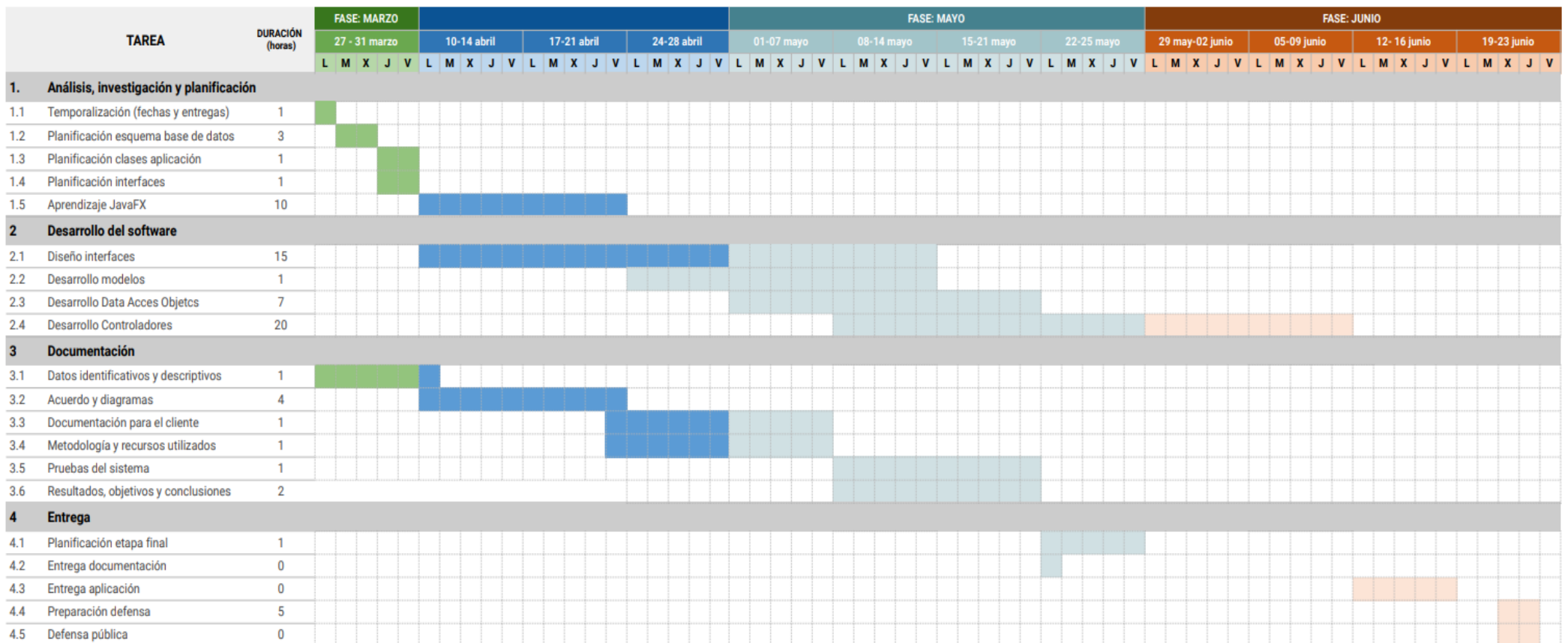
```



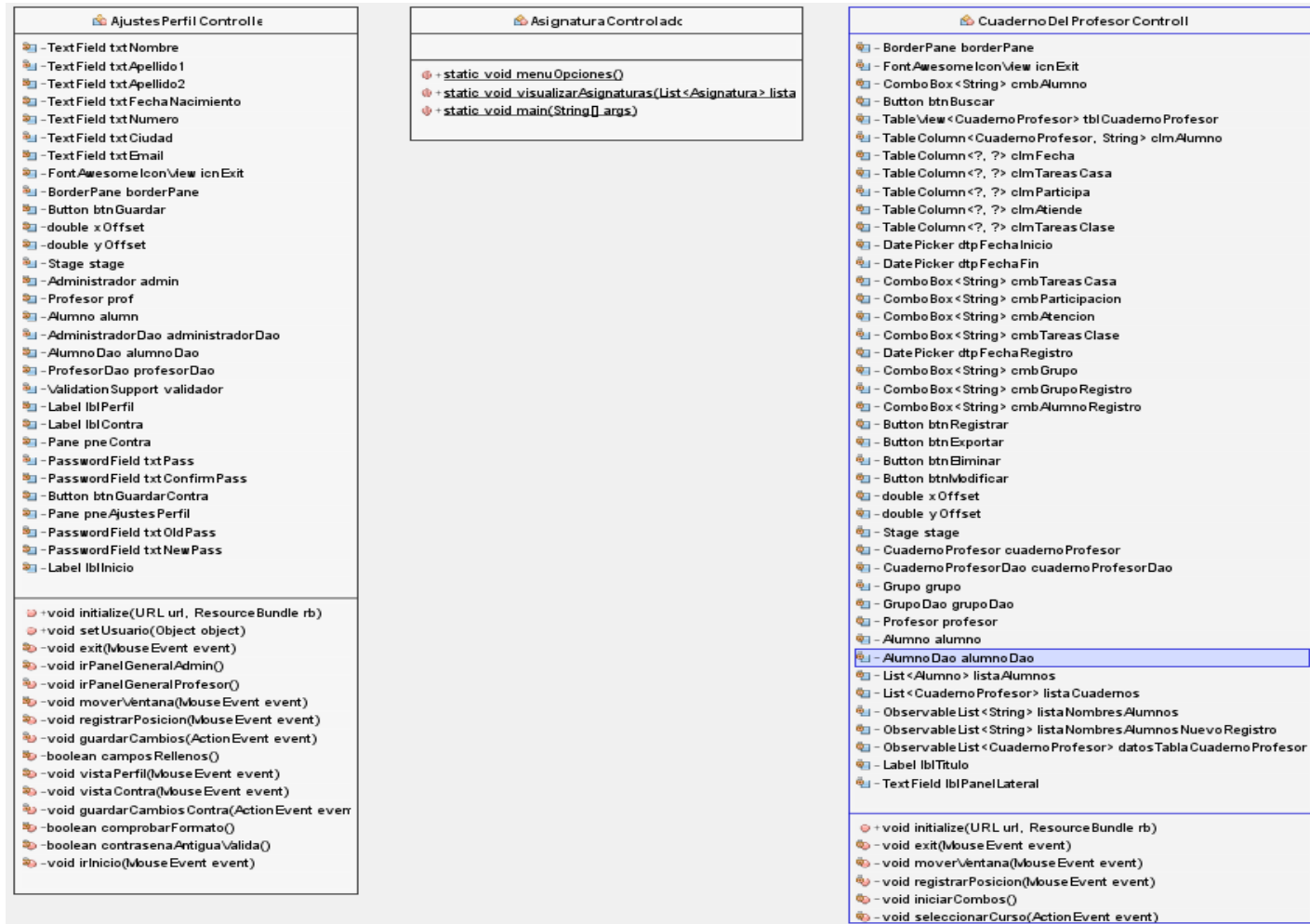
```
KEY `examen_id` (`examen_id`),
KEY `alumno_id` (`alumno_id`),
  CONSTRAINT `nota_examen_alumno_fk` FOREIGN KEY (`alumno_id`) REFERENCES
`alumno` (`id`),
  CONSTRAINT `nota_examen_examen_fk` FOREIGN KEY (`examen_id`) REFERENCES
`examen` (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=13 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;
```

### 6.3. Diagrama de Gantt.

## DIAGRAMA DE GANTT



## 6.4. Diagramas de clase.



### DatosAcademicosControlle

- ScrollPane scrollPane
- FontAwesomeIconView iconExit
- ComboBox<String> cmbGrupo
- Button btnBuscar
- ComboBox<String> cmbAsignatura
- Label lblDescripcion
- Label lblDescripcion1
- TableView<NotaTareaEvaluable> tblTareas
- TableView<NotaExamen> tblExámenes
- TableColumn<NotaTareaEvaluable, String> clmAlumnoTarea
- TableColumn<NotaTareaEvaluable, String> clmActividadTarea
- TableColumn<NotaTareaEvaluable, String> clmNotaTarea
- TableColumn<NotaExamen, String> clmAlumnoExamen
- TableColumn<NotaExamen, String> clmExamen
- TableColumn<NotaExamen, String> clmNotaExamen
- Button btnEliminar
- Button btnModificar
- double xOffset
- double yOffset
- Stage stage
- Alumno alumno
- AlumnoDao alumnoDao
- Asignatura asignatura
- Grupo grupo
- NotasAlumnosDao notasDao
- NotaExamen notaExamen
- NotaExamenDao notaExamenDao
- NotaTareaEvaluable notaTareaEvaluable
- NotaTareaDao notaTareaDao
- Profesor profesor
- List<Alumno> listaAlumnos
- List<NotaAlumnoDTO> listadoNotas
- List<NotaExamen> listadoNotasExámenes
- List<NotaTareaEvaluable> listadoNotasTareas
- ObservableList<NotaExamen> datosNotasExamen
- ObservableList<NotaTareaEvaluable> datosNotasTareas
- Label lblTitulo
- TextField lblPanelLateral

- +void initialize(URL url, ResourceBundle rb)
- +void setProfesor(Profesor profesor)
- void moverVentana(MouseEvent event)
- void RegistrarPosicion(MouseEvent event)
- void exit(MouseEvent event)
- void buscar(ActionEvent event)
- void eliminar(ActionEvent event)
- void modificar(ActionEvent event)
- +void initAttributes(Profesor profesor)
- void iniciarCombos()
- List<String> gruposCursoYLetra(List<Grupo> grupos)
- List<String> nombreAsignaturas(List<Asignatura> asignaturas)
- void mensajeAlertaCamposIncompletos()
- Grupo obtenerGrupo(String grupoSeleccionado)
- void iniciarTablaTareas(Grupo grupo, Asignatura asignatura)
- void iniciarTablaExámenes(Grupo grupo, Asignatura asignatura)
- void eliminarNotaExamen(NotaExamen notaExamen)
- void eliminarNotaTarea(NotaTareaEvaluable notaTareaEvaluable)

### GestionAlumnosControlle

- BorderPane borderPane
- FontAwesomeIconView iconExit
- TableView<Alumno> tblAlumnos
- TableColumn<Alumno, String> clmNombre
- TableColumn<Alumno, String> clmPrimerApellido
- TableColumn<Alumno, String> clmSegundoApellido
- TableColumn<Alumno, String> clmCorreo
- TableColumn<Alumno, String> clmTutor
- TableColumn<Alumno, String> clmGrupo
- ComboBox<String> cmbGrupo
- Button btnEliminar
- Button btnModificar
- Button btnFiltrar
- double xOffset
- double yOffset
- Stage stage
- Grupo grupo
- Profesor profesor
- Administrador administrador
- Alumno alumno
- AlumnoDao alumnoDao
- List<Alumno> listaAlumnos
- ObservableList<Alumno> datosTablaAlumnos
- Label lblTitulo
- TextField lblPanelLateral

- +void initialize(URL url, ResourceBundle rb)
- void exit(MouseEvent event)
- void moverVentana(MouseEvent event)
- void RegistrarPosicion(MouseEvent event)
- void iniciarTabla()
- void iniciarComboBox()
- List<String> gruposCursoYLetra(List<Grupo> grupos)
- void modificarAlumno(ActionEvent event)
- void eliminarAlumno(ActionEvent event)
- void eliminarAlumnoYDatos(Alumno alumno)
- void filtrar(ActionEvent event)
- Grupo obtenerGrupo(String grupoSeleccionado)
- +void setAdministrador(Administrador administrador)



























### GestionAsignaturasControlle

- BorderPane borderPane
- FontAwesomeIconView btnExit
- TableView<ProfesorAsignatura> tblAsignaturas
- TableColumn<ProfesorAsignatura, String> clmAsignatura
- TableColumn<ProfesorAsignatura, String> clmProfesor
- TableColumn<ProfesorAsignatura, String> clmAño
- Button btnModificar
- Button btnEliminar
- Pane modificarAsignatura
- double xOffset
- double yOffset
- Stage stage
- Profesor profesor
- Administrador administrador
- Asignatura asignatura
- ProfesorAsignatura profesorAsignatura
- ProfesorAsignaturaDao profesorAsignaturaDao
- List<ProfesorAsignatura> profesoresAsignaturas
- ObservableList<ProfesorAsignatura> datosTabla
- Label lblTitulo
- TextField lblPanelLateral

- +void initialize(URL url, ResourceBundle rb)
- void moverVentana(MouseEvent event)
- void registrarPosicion(MouseEvent event)
- void exit(MouseEvent event)
- void eliminarAsignatura(ActionEvent event)
- void modificarAsignatura(ActionEvent event)
- void iniciarTabla()
- +void setAdministrador(Administrador administrador)

### GestionElementosControll

-  - TableView<Profesor> tblProfesores
-  - TextField txtNombre
-  - RadioButton rdbTutor
-  - TextField txtEmail
-  - TextField txtSegundoApellido
-  - TextField txtPrimerApellido
-  - TableColumn<Profesor, String> clmNombreProfesor
-  - TableColumn<Profesor, String> clmApellido1Profesor
-  - TableColumn<Profesor, String> clmApellido2Profesor
-  - TableColumn<Profesor, String> clmCorreoProfesor
-  - TableColumn<Profesor, String> clmProfesorTutor
-  - TableColumn<Profesor, String> clmGrupoProfesor
-  - BorderPane borderPane
-  - Button btnModificar
-  - Button btnEliminar
-  - FontAwesomeIconView icnExit
-  - int tipoElemento
-  - double xOffset
-  - double yOffset
-  - Stage stage
-  - ObservableList<Profesor> datosTablaProfesores
-  - ObservableList<Profesor> filtroProfesores
-  - Administrador administrador
-  - List<Profesor> profesores
-  - Label lblTitulo
-  - TextField lblPanelLateral






-  + GestionElementosController(int tipoElemento)
-  + GestionElementosController()
-  + void initialize(URL url, ResourceBundle rb)
-  + void iniciarTablas()
-  + void setElemento(int tipoElemento)
-  - void filtrarPorNombre(KeyEvent event)
-  - void filtrarPorPrimerApellido(KeyEvent event)
-  - void filtrarPorSegundoApellido(KeyEvent event)
-  - void filtrarPorCorreo(KeyEvent event)
-  - void filtrarPorTutor(ActionEvent event)
-  - void moverVentana(MouseEvent event)
-  - void registrarPosicion(MouseEvent event)
-  - void seleccionar(MouseEvent event)
-  - void modificar(ActionEvent event)
-  - void eliminar(ActionEvent event)
-  - void eliminarProfesoryDatos(Profesor profesor)
-  - void exit(MouseEvent event)
-  + void setAdministrador(Administrador usuario)

### GestionGruposControll

-  - BorderPane borderPane
-  - FontAwesomeIconView icnExit
-  - TableView<Grupo> tblGrupo
-  - TableColumn<Grupo, String> clmCurso
-  - TableColumn<Grupo, String> clmLetra
-  - TableColumn<Grupo, String> clmTutor
-  - TableColumn<Grupo, Long> clmNumeroAlumnos
-  - Button btnEliminar
-  - double xOffset
-  - double yOffset
-  - Stage stage
-  - ObservableList<Grupo> datosTablaGrupos
-  - Administrador administrador
-  - Profesor profesor
-  - Grupo grupo
-  - GrupoDao grupoDao
-  - List<Grupo> grupos
-  - Label lblTitulo
-  - TextField lblPanelLateral

-  + void initialize(URL url, ResourceBundle rb)
-  - void exit(MouseEvent event)
-  - void modificarGrupo(ActionEvent event)
-  - void eliminarGrupo(ActionEvent event)
-  - void moverVentana(MouseEvent event)
-  - void registrarPosicion(MouseEvent event)
-  - void iniciarTabla()
-  - void eliminarGrupoYDatos(Gruo grupo)
-  - Long obtenerAlumnosPorGrupo(Gruo grupo)
-  + void setAdministrador(Administrador administrador)

### GrupoControlad

-  +static void menuOpciones()
-  +static void visualizarGrupos(List<Grupo> lista)
-  - static Grupo setDatos()
-  - static int pedirTutor()
-  +static void main(String[] args)

### LoginController

- TextField txtUsuario
- PasswordField txtPassword
- Button btnLogin
- FontAwesomeIconView icnUser
- FontAwesomeIconView icnKey
- FontAwesomeIconView icnExit
- BorderPane borderPane
- Pane pneExit
- Hyperlink hypRegistrar
- double xOffset
- double yOffset
- Stage stage
- ValidationSupport validador

- +void initialize(URL url, ResourceBundle rb)
- void exit(MouseEvent event)
- void registrarPosicion(MouseEvent event)
- void moverVentana(MouseEvent event)
- void irRegistrar(ActionEvent event)
- void login(ActionEvent event)
- boolean camposRellenos()
- Object validarUsuario()
- Administrador validarAdministrador()
- Profesor validarProfesor()
- Alumno validarAlumno()
- void iniciarSesion(Object object)
- void irPanelGeneralAdministrador(Administrador administrador)
- void irPanelGeneralProfesor(Profesor profesor)
- void irPanelGeneralAlumno(Alumno alumno)

### ModificarDatosAlumnoController

- SplitPane splitPane
- FontAwesomeIconView icnExit
- TextField txtNombre
- TextField txtPrimerApellido
- TextField txtSegundoApellido
- TextField txtCorreo
- ComboBox<String> cmbGrupo
- Button btnAceptar
- Button btnCancelar
- Stage stage
- double xOffset
- double yOffset
- SplitPane splitaPane
- Alumno alumno
- AlumnoDao alumnoDao
- Grupo grupo
- ValidationSupport validador

- +void initialize(URL url, ResourceBundle rb)
- +void initAttributes(Alumno alumno)
- void exit(MouseEvent event)
- void moverVentana(MouseEvent event)
- void registrarPosicion(MouseEvent event)
- List<String> gruposCursoYLetra(List<Grupo> grupos)
- void aceptarCambios(ActionEvent event)
- void cancelar(ActionEvent event)
- boolean camposRellenos()
- Grupo obtenerGrupo(String grupoSeleccionado)
- +Alumno getAlumno()

### ModificarDatosAsignaturaController

- SplitPane splitPane
- ComboBox<String> cmbAsignatura
- ComboBox<String> cmbProfesor
- Button btnAceptar
- Button btnCancelar
- Stage stage
- double xOffset
- double yOffset
- SplitPane splitaPane
- Profesor profesor
- Asignatura asignatura
- ProfesorAsignatura profesorAsignaturaAntiguo
- ProfesorAsignaturaDao profesorAsignaturaDao
- FontAwesomeIconView btnExit

- +void initialize(URL url, ResourceBundle rb)
- +void initAttributes(ProfesorAsignatura profesorAsignatura)
- void moverVentana(MouseEvent event)
- void registrarPosicion(MouseEvent event)
- void exit(MouseEvent event)
- void aceptarCambios(ActionEvent event)
- void cancelar(ActionEvent event)
- List<String> nombreAsignaturas(List<Asignatura> asignaturas)
- List<String> nombreCompleto(List<Profesor> profesores)
- Profesor obtenerProfesor(String nombreProfesor)
- List<String> obtenerNombreyApellidos(String nombreProfesor)

### ModificarDatosExamenControl

- SplitPane splitPane
- Button btnAceptar
- Button btnCancelar
- TextField txtNombre
- TextField txtPorcentaje
- ComboBox<String> cmbAsignatura
- ComboBox<String> cmbGrupo
- FontAwesomeIconView iconExit
- Stage stage
- double xOffset
- double yOffset
- Asignatura asignatura
- AsignaturaDao asignaturaDao
- Grupo grupo
- Profesor profesor
- Examen examen
- ExamenDao examenDao

- +void initialize(URL url, ResourceBundle rb)
- +void initAttributes(Profesor profesor, Examen examen)
- void aceptar(ActionEvent event)
- void cancelar(ActionEvent event)
- void exit(MouseEvent event)
- void moverVentana(MouseEvent event)
- void registrarPosicion(MouseEvent event)
- void mensajeAlertaFormato()
- Grupo obtenerGrupo(String grupoSeleccionado)

### ModificarDatosGrupoControl

- +void initialize(URL url, ResourceBundle rb)
- void initAttributes(Grupo grupo)

### ModificarDatosProfesorControl

- SplitPane splitPane
- TextField txtNombre
- TextField txtPrimerApellido
- TextField txtSegundoApellido
- TextField txtEmail
- ComboBox<String> cmbGrupos
- Button btnAceptar
- Button btnCancelar
- Stage stage
- double xOffset
- double yOffset
- SplitPane splitaPane
- Object object
- Profesor profesor
- Administrador administrador
- Alumno alumno
- Grupo grupo
- ValidationSupport validador
- String curso

- +void initialize(URL url, ResourceBundle rb)
- +void initAttributes(Profesor profesor)
- void exit(MouseEvent event)
- void aceptar(ActionEvent event)
- void asignarTutorAGrupo(Grupo grupo, Profesor profesor, GrupoDao grupoDao)
- void cancelar(ActionEvent event)
- void moverVentana(MouseEvent event)
- void registrarPosicion(MouseEvent event)
- boolean camposRellenos()
- List<String> gruposCursoYLetra(List<Grupo> grupos)
- +Profesor getProfesorActualizado()
- +void setAdministrador(Administrador administrador)



Modificar Datos Registro Controlle
<ul style="list-style-type: none"> <li>- SplitPane splitPane</li> <li>- FontAwesomeIconView exit</li> <li>- DatePicker dtpFechaRegistro</li> <li>- ComboBox&lt;String&gt; cmbTareasCasa</li> <li>- ComboBox&lt;String&gt; cmbAtencion</li> <li>- ComboBox&lt;String&gt; cmbParticipacion</li> <li>- ComboBox&lt;String&gt; cmbTareasClase</li> <li>- Button btnCancelar</li> <li>- Button btnAceptar</li> <li>- TextField txtAlumno</li> <li>- Stage stage</li> <li>- double xOffset</li> <li>- double yOffset</li> <li>- Alumno alumno</li> <li>- AlumnoDao alumnoDao</li> <li>- CuadernoProfesor cuadernoProfesor</li> <li>- CuadernoProfesorDao cuadernoProfesorDao</li> </ul>
<ul style="list-style-type: none"> <li>+void initialize(URL url, ResourceBundle rb)</li> <li>+void initAttributes(CuadernoProfesor cuadernoProfesor)</li> <li>-void moverVentana(MouseEvent event)</li> <li>-void registrarPosicion(MouseEvent event)</li> <li>-void exit(MouseEvent event)</li> <li>-void cancelar(ActionEvent event)</li> <li>-void aceptar(ActionEvent event)</li> <li>-void iniciarCombos()</li> <li>-Alumno obtenerAlumnoDelTxt(String alumnoString)</li> <li>-Alumno consultarPorNombreYAPELLIDOS(List&lt;String&gt; nombre</li> </ul>

Modificar Datos Tarea Evaluable Controlle
<ul style="list-style-type: none"> <li>- SplitPane splitPane</li> <li>- Button btnAceptar</li> <li>- Button btnCancelar</li> <li>- TextField txtNombre</li> <li>- TextField txtPorcentaje</li> <li>- ComboBox&lt;String&gt; cmbAsignatura</li> <li>- ComboBox&lt;String&gt; cmbGrupo</li> <li>- FontAwesomeIconView exit</li> <li>- Stage stage</li> <li>- double xOffset</li> <li>- double yOffset</li> <li>- Asignatura asignatura</li> <li>- AsignaturaDao asignaturaDao</li> <li>- Grupo grupo</li> <li>- Profesor profesor</li> <li>- TareaEvaluable tareaEvaluable</li> <li>- TareaEvaluableDao tareaEvaluableDao</li> </ul>
<ul style="list-style-type: none"> <li>+void initialize(URL url, ResourceBundle rb)</li> <li>-void aceptar(ActionEvent event)</li> <li>-void cancelar(ActionEvent event)</li> <li>-void moverVentana(MouseEvent event)</li> <li>-void registrarPosicion(MouseEvent event)</li> <li>-void exit(MouseEvent event)</li> <li>+void initAttributes(Profesor profesor, TareaEvaluable tareaEvaluat</li> <li>- List&lt;String&gt; gruposCursoYLetra(List&lt;Grupo&gt; grupos)</li> <li>- List&lt;String&gt; nombreAsignaturas(List&lt;Asignatura&gt; asignaturas)</li> <li>-void mensajeAlertaFormato()</li> <li>- Grupo obtenerGrupo(String grupoSeleccionado)</li> </ul>

Panel Administracion Administrador Controlle
<ul style="list-style-type: none"> <li>- BorderPane borderPane</li> <li>- ComboBox&lt;String&gt; cmbTipoUsuario</li> <li>- TextField txtNombreUsuario</li> <li>- TextField txtApellidoUsuario</li> <li>- PasswordField txtContraseñaUsuario</li> <li>- TextField txtEmailUsuario</li> <li>- Button btnAgregarUsuario</li> <li>- ComboBox&lt;String&gt; cmbCurso</li> <li>- ComboBox&lt;String&gt; cmbLetra</li> <li>- ComboBox&lt;String&gt; cmbTutor</li> <li>- ComboBox&lt;String&gt; cmbAsignatura</li> <li>- Button btnAgregarGrupo</li> <li>- Button btnAgregarAsignatura</li> <li>- TextField txtNombreAsignatura</li> <li>- ComboBox&lt;String&gt; cmbProfesor</li> <li>- TableView&lt;Grupo&gt; tblGrupo</li> <li>- TableColumn&lt;Grupo, String&gt; clmCurso</li> <li>- TableColumn&lt;Grupo, String&gt; clmLetra</li> <li>- TableColumn&lt;Grupo, String&gt; clmTutor</li> <li>- TableView&lt;ProfesorAsignatura&gt; tblAsignaturas</li> <li>- TableColumn&lt;ProfesorAsignatura, String&gt; clmAsignatura</li> <li>- TableColumn&lt;ProfesorAsignatura, String&gt; clmProfesor</li> <li>- TextField txtSegundoApellidoUsuario</li> <li>- TableView&lt;?&gt; tblModificar</li> <li>- Button btnModificar</li> <li>- FontAwesomeIconView iconExit</li> <li>- ComboBox&lt;String&gt; cmbOpcionModificar</li> <li>- ComboBox&lt;String&gt; cmbOpcionModificar2</li> <li>- double xOffset</li> <li>- double yOffset</li> <li>- Stage stage</li> <li>- ValidationSupport validadorUsuario</li> <li>- String nombre</li> <li>- int tipoElemento</li> <li>- Administrador administrador</li> <li>- Label lblTitulo</li> <li>- TextField lblPanelLateral</li> </ul>
<ul style="list-style-type: none"> <li>+void initialize(URL url, ResourceBundle rb)</li> <li>-void iniciarCombos()</li> <li>- List&lt;String&gt; nombreCompleto(List&lt;Profesor&gt; profesores)</li> <li>- List&lt;String&gt; nombreAsignaturas(List&lt;Asignatura&gt; asignaturas)</li> <li>-void iniciarTablas()</li> <li>-void moverVentana(MouseEvent event)</li> <li>-void registrarPosicion(MouseEvent event)</li> <li>-void agregarUsuario(ActionEvent event)</li> <li>-boolean camposUsuarioRellenos()</li> <li>-void insertar()</li> <li>-Administrador crearAdministrador()</li> </ul>



Panel AdministracionProfesor Controller
<ul style="list-style-type: none"> <li>- BorderPane borderPane</li> <li>- FontAwesomeIconView iconExit</li> <li>- ComboBox&lt;String&gt; cmbCategoriaActividad</li> <li>- TextField txtPorcentaje</li> <li>- TextField txtNombreActividad</li> <li>- ComboBox&lt;String&gt; cmbAsignaturaCrear</li> <li>- TableView&lt;TareaEvaluable&gt; tblActividades</li> <li>- TableColumn&lt;?, ?&gt; clmActividad</li> <li>- TableColumn&lt;TareaEvaluable, String&gt; clmCategoria</li> <li>- TableColumn&lt;?, ?&gt; clmPorcentaje</li> <li>- TableColumn&lt;TareaEvaluable, String&gt; clmGrupo</li> <li>- TableColumn&lt;TareaEvaluable, String&gt; clmAsignatura</li> <li>- ComboBox&lt;String&gt; cmbGrupoBuscar</li> <li>- Button btnBuscar</li> <li>- ComboBox&lt;String&gt; cmbAsignaturaBuscar</li> <li>- Button btnEliminar</li> <li>- ComboBox&lt;String&gt; cmbGrupoCrear</li> <li>- Button btnAceptar</li> <li>- ComboBox&lt;String&gt; cmbTareaOExamen</li> <li>- TableView&lt;Examen&gt; tblExamenes</li> <li>- TableColumn&lt;Examen, String&gt; clmActividadExamen</li> <li>- TableColumn&lt;Examen, String&gt; clmCategoriaExamen</li> <li>- TableColumn&lt;Examen, String&gt; clmPorcentajeExamen</li> <li>- TableColumn&lt;Examen, String&gt; clmGrupoExamen</li> <li>- TableColumn&lt;Examen, String&gt; clmAsignaturaExamen</li> <li>- double xOffset</li> <li>- double yOffset</li> <li>- Stage stage</li> <li>- Asignatura asignatura</li> <li>- AsignaturaDao asignaturaDao</li> <li>- AsignaturaGrupoDao asignaturaGrupoDao</li> <li>- Examen examen</li> <li>- ExamenDao examenDao</li> <li>- Grupo grupo</li> <li>- GrupoDao grupoDao</li> <li>- Profesor profesor</li> <li>- TareaEvaluable tareaEvaluable</li> <li>- TareaEvaluableDao tareaEvaluableDao</li> <li>- List&lt;TareaEvaluable&gt; listaTareasEvaluables</li> <li>- List&lt;Examen&gt; listaExamenes</li> <li>- ObservableList&lt;String&gt; listaTareasEvaluablesString</li> <li>- ObservableList&lt;String&gt; listaExamenesString</li> <li>- ObservableList&lt;TareaEvaluable&gt; datosTablaTareasEvaluable</li> <li>- ObservableList&lt;Examen&gt; datosTablaExamenes</li> <li>- Label lblDescripcion</li> <li>- Button btnGestionDatosAcademicos</li> <li>- Label lblTitulo</li> <li>- TextField lblPanelLateral</li> </ul>
<ul style="list-style-type: none"> <li>+ PanelAdministracionProfesorController()</li> <li>+ void initialize(URL url, ResourceBundle rb)</li> <li>- void exit(MouseEvent event)</li> <li>- void moverVentana(MouseEvent event)</li> <li>- void registrarPosicion(MouseEvent event)</li> </ul>

Panel General Administrador Controller
<ul style="list-style-type: none"> <li>- FontAwesomeIconView iconExit</li> <li>- BorderPane borderPane</li> <li>- Pane pnePerfilUsuario</li> <li>- Pane pnePanelAdministracion</li> <li>- Pane pnePanelGestionProfesores</li> <li>- Pane pneCalendario</li> <li>- double xOffset</li> <li>- double yOffset</li> <li>- Stage stage</li> <li>- Administrador usuario</li> <li>- Pane pneGestionAlumnos</li> </ul>
<ul style="list-style-type: none"> <li>+ void initialize(URL url, ResourceBundle rb)</li> <li>+ void setUsuario(Administrador administrador)</li> <li>- void exit(MouseEvent event)</li> <li>- void registrarPosicion(MouseEvent event)</li> <li>- void moverVentana(MouseEvent event)</li> <li>- void irPerfilUsuario(MouseEvent event)</li> <li>- void irPoliticaPrivacidad(MouseEvent event)</li> <li>- void irPanelAdministracion(MouseEvent event)</li> <li>- void irGestionProfesores(MouseEvent event)</li> <li>- void irGestionAlumnos(MouseEvent event)</li> </ul>

PoliticaPrivacidadController
<ul style="list-style-type: none"> <li>- Pane pnePoliticaPrivacidad</li> <li>- FontAwesomeIconView iconExit</li> <li>- double xOffset</li> <li>- double yOffset</li> <li>- Stage stage</li> <li>- Object usuario</li> <li>- Administrador administrador</li> <li>- Profesor profesor</li> </ul>
<ul style="list-style-type: none"> <li>+ void initialize(URL url, ResourceBundle rb)</li> <li>- void registrarPosicion(MouseEvent event)</li> <li>- void moverVentana(MouseEvent event)</li> <li>- void exit(MouseEvent event)</li> <li>- void irPanelGeneralAdmin()</li> <li>- void irPanelGeneralProfesor()</li> <li>+ Object getUsuario()</li> <li>+ void setUsuario(Object usuario)</li> <li>+ Administrador getAdministrador()</li> <li>+ void setAdministrador(Administrador administrado)</li> <li>+ Profesor getProfesor()</li> <li>+ void setProfesor(Profesor profesor)</li> </ul>

### ProfesorAsignaturaControlad

```
+static void menuOpciones()  
-static void visualizarLista(List<ProfesorAsignatura> listaProfesorAsignatura)  
+static void main(String[] args)
```

### ProfesorControlad

```
+static void menuOpciones()  
+static void visualizarProfesores(List<Profesor> lista)  
-static Profesor setDatos()  
+static void main(String[] args)
```

### RegisterControlad

```
- BorderPane borderPane  
- TextField txtNombre  
- TextField txtPrimerApellido  
- TextField txtSegundoApellido  
- TextField txtEmail  
- PasswordField txtContra  
- ComboBox<String> cmbCategoria  
- TextField txtCodigo  
- Button btnRegister  
- FontAwesomeIconView icnUser  
- FontAwesomeIconView icnKey  
- Pane pneExit  
- FontAwesomeIconView icnExit  
- double xOffset  
- double yOffset  
- Stage stage  
- ValidationSupport validador  
- String nombre
```

```
+void initialize(URL url, ResourceBundle rb)  
-void exit(MouseEvent event)  
-void moverVentana(MouseEvent event)  
-void registrarPosicion(MouseEvent event)  
-void registrar(ActionEvent event)  
-boolean camposRellenos()  
-boolean comprobarFormatos()  
-boolean codigoValido()  
-void registrarUsuario()  
-Administrador crearAdministrador()  
-Profesor crearProfesor()  
-Alumno crearAlumno()
```

### PanelGeneralProfesorControll

```
- BorderPane borderPane  
- FontAwesomeIconView icnExit  
- Pane pneCalendario  
- double xOffset  
- double yOffset  
- Stage stage  
- Profesor profesor  
- Pane pneCuadernoProfesor
```

```
+void initialize(URL url, ResourceBundle rb)  
+void setUsuario(Profesor profesor)  
-void moverVentana(MouseEvent event)  
-void RegistrarPosicion(MouseEvent event)  
-void exit(MouseEvent event)  
-void irPanelAdministracion(MouseEvent event)  
-void irDatosAcademicos(MouseEvent event)  
-void irPoliticaPrivacidad(MouseEvent event)  
-void irPerfilUsuario(MouseEvent event)  
-void irCuadernoProfesor(MouseEvent event)  
+void setProfesor(Profesor profesor)
```

<<interface>>  
∞ ∞ **AdministradorDao**

~List<Administrador> consultarTodos()  
~Administrador consultar(int id)  
~Administrador consultarPorEmailYContrasena(String email, String contrasena)  
~void insertar(Administrador administrador)  
~boolean actualizar(Administrador administrador)  
~boolean eliminar(int id)  
~Administrador consultarContrasena(Administrador admin, PasswordField txtOldPass)  
+boolean actualizarContrasena(Administrador admin, PasswordField txtNewPass)

<<interface>>  
∞ ∞ **AlumnoDac**

~List<Alumno> consultarTodos()  
~Alumno consultarPorEmailYContrasena(String email, String contraseña)  
~void insertar(Alumno alumno)  
~boolean actualizar(Alumno alumno)  
~boolean eliminar(Alumno alumno)  
~List<Alumno> consultarPorGrupo(Grupos grupo)  
~int eliminarGrupo(Grupos grupo)  
~Alumno consultarPorNombreYApellidos(List<String> nombres)

<<interface>>  
∞ ∞ **AsignaturaDao**

~List<Asignatura> consultarTodos()  
~Asignatura consultar(int id)  
~void insertar(Asignatura asignatura)  
~void actualizar(int id)  
~boolean eliminar(int id)  
~List<Asignatura> consultarAsignaturaEnExamen(int codigo)  
~List<Asignatura> consultarAsignaturaProfesor(int codigo)  
+Asignatura consultarPorNombre(String nombreAsignatura)

<<interface>>  
∞ ∞ **AsignaturaGrupoDao**

~List<AsignaturaGrupo> consultarAsignaturasYGrupo()

<<interface>>  
∞ ∞ **CuadernoProfesorDa**

🔧 ~void insertar(CuadernoProfesor cuadernoProfesor)  
🔧 ~List<CuadernoProfesor> consultarTodos()  
🔧 ~List<CuadernoProfesor> consultarRegistrosPorFechas(String fechaInicio, String fechaFin)  
🔧 ~List<CuadernoProfesor> consultarRegistroPorAlumnoYFechas(Alumno alumno, String fechaInicioStr, String fechaFinStr)  
🔧 ~CuadernoProfesor consultarRegistroAlumnoExistente(Alumno alumno, String fechaFormateada)  
🔧 ~void eliminarRegistroPorAlumnoYFecha(CuadernoProfesor cuadernoProfesor)  
🔧 ~boolean actualizar(CuadernoProfesor cuadernoProfesor)

<<interface>>  
∞ ∞ **ExamenDao**

🔧 ~Examen consultarPorNombreAsignaturaYGrupo(String nombre, Asignatura asignatura, Grupo grupo)  
🔧 ~List<Examen> consultarPorAsignaturaYGrupo(Asignatura asignatura, Grupo grupo)  
🔧 ~void insertar(Examen examen)  
🔧 ~List<Examen> consultarTodos()  
🔧 ~boolean eliminar(Examen examen)  
🔧 ~boolean actualizar(Examen examenActualizado)

<<interface>>  
∞ ∞ **GrupoDao**

🔧 ~List<Grupo> consultarTodos()  
🔧 ~Grupo consultar(int id)  
🔧 ~Grupo consultarTutorGrupo(int tutorId)  
🔧 ~void insertar(Grupo grupo)  
🔧 ~void actualizar(Grupo grupo)  
🔧 ~boolean eliminar(int codigo)  
🔧 ~List<Grupo> consultarGrupoConAlumnos()  
🔧 ~List<String> consultarCursos()  
🔧 ~List<String> consultarLetrasGrupos()  
🔧 ~int eliminarTutor(Integer id)  
🔧 ~Grupo consultarGrupoPorCursoYLetra(String curso, String letra)  
🔧 ~void asignarTutor(int grupold, int proflid)  
🔧 ~Long consultarAlumnosPorGrupo(Grupo grupo)

<<interface>>  
∞ ∞ **NotaDao**

🔧 ~void eliminar(int idNota)  
🔧 ~void eliminarPorAlumno(int idAlumno)

<<interface>>  
∞ ∞ **NotaExamenDao**

🔧 ~void crearCasillasNotasPorExamenYGrupo(Examen examenCreado, List<Alumno> listaAlumnos)  
🔧 ~List<NotaExamen> consultarPorGrupoYAsignatura(Grupo grupo, Asignatura asignatura)  
🔧 ~void eliminarNota(NotaExamen notaExamen)  
🔧 ~void insertarNota(NotaExamen notaExamen)  
🔧 ~void eliminarNotasPorExamen(Examen examen)

<<interface>>

∞∞ **NotaTareaDao**

- ~ List<NotaTareaEvaluable> consultarPorGrupoYAsignatura(Grupo grupo, Asignatura asignatura)
- ~ void crearCasillasNotasPorTareasYGrupo(TareaEvaluable tareaEvaluableCreada, List<Alumno> listaAlumnos)
- ~ void eliminarNota(NotaTareaEvaluable notaTareaEvaluable)
- ~ void insertarNota(NotaTareaEvaluable notaTareaEvaluable)
- ~ void eliminarNotasPorTarea(TareaEvaluable tareaEvaluable)

<<interface>>

∞∞ **ProfesorAsignaturaDao**

- ~ List<ProfesorAsignatura> consultarTodos()
- ~ ProfesorAsignatura consultar(int id)
- ~ void insertar(ProfesorAsignatura profesorAsignatura)
- ~ boolean actualizar(ProfesorAsignatura profesorAsignaturaNuevo, ProfesorAsignatura profesorAsignaturaAntiguo)
- ~ boolean eliminar(int codigo)
- + int eliminarProfesoryAsignaturasAsociadas(Integer id)
- ~ ProfesorAsignatura consultarPorProfesorYAsignatura(ProfesorAsignaturaPK nuevo)

<<interface>>

∞∞ **TareaEvaluableDao**

- ~ TareaEvaluable consultarPorNombreAsignaturaYGrupo(String actividad, Asignatura asignatura, Grupo grupo)
- ~ List<TareaEvaluable> consultarPorAsignaturaYGrupo(Asignatura asignatura, Grupo grupo)
- ~ void insertar(TareaEvaluable tareaEvaluable)
- ~ List<TareaEvaluable> consultarTodos()
- ~ boolean eliminar(TareaEvaluable tareaEvaluable)
- + boolean actualizar(TareaEvaluable tareaEvaluable1 Actualizada)

<<interface>>

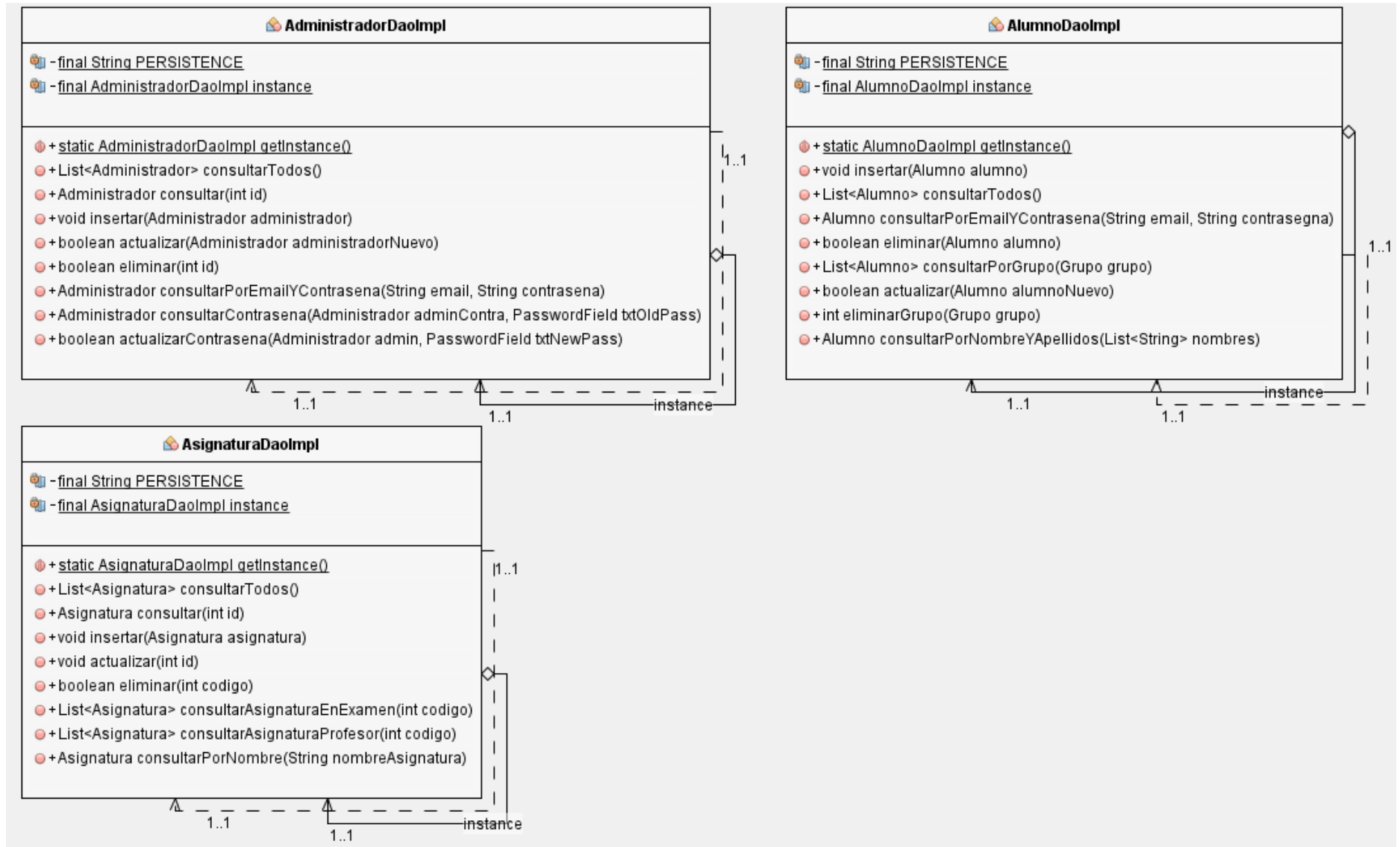
∞∞ **NotasAlumnosDao**

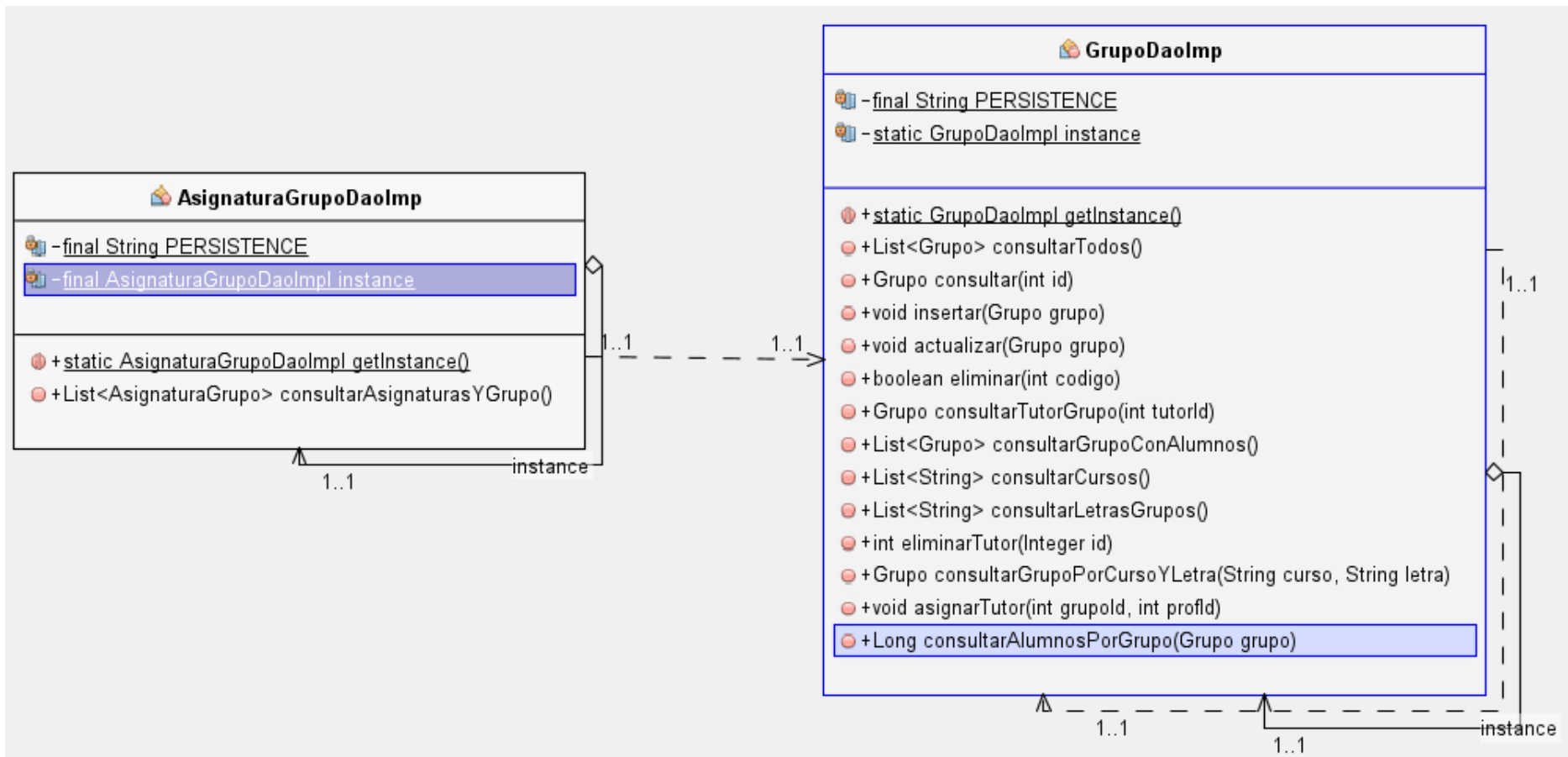
- ~ List<NotaAlumnoDTO> consultarNotasPorGrupoYAsignatura(Grupo grupo, Asignatura asignatura)

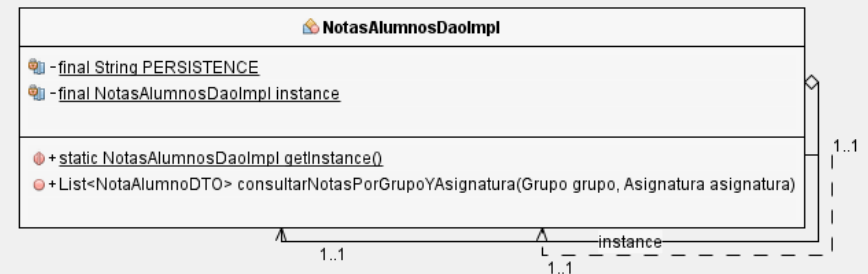
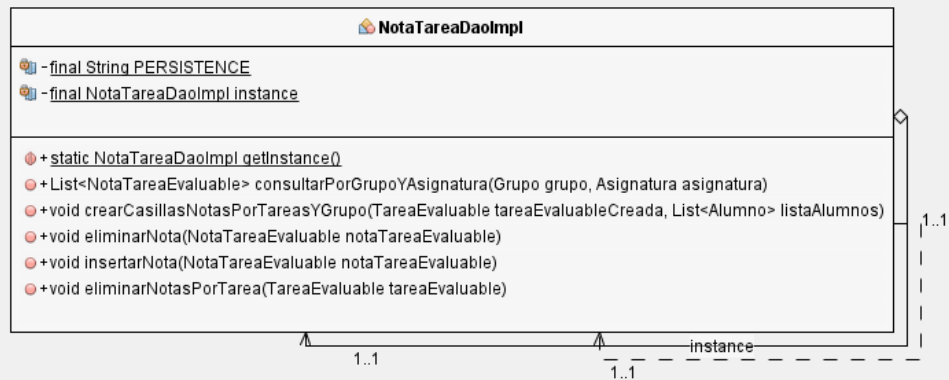
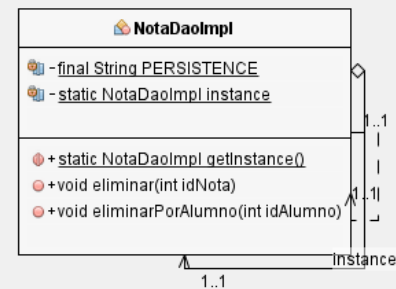
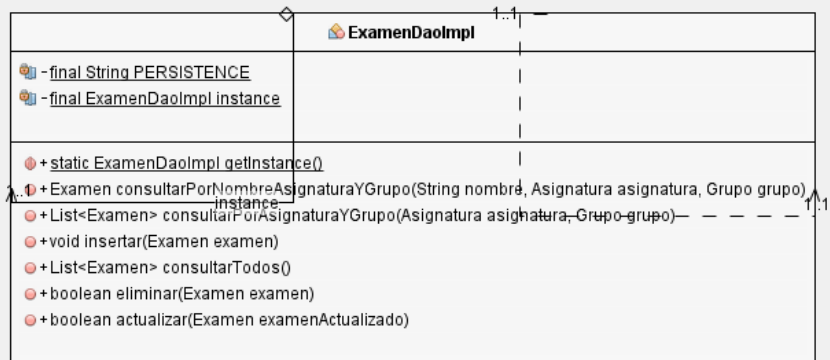
<<interface>>

∞∞ **ProfesorDao**

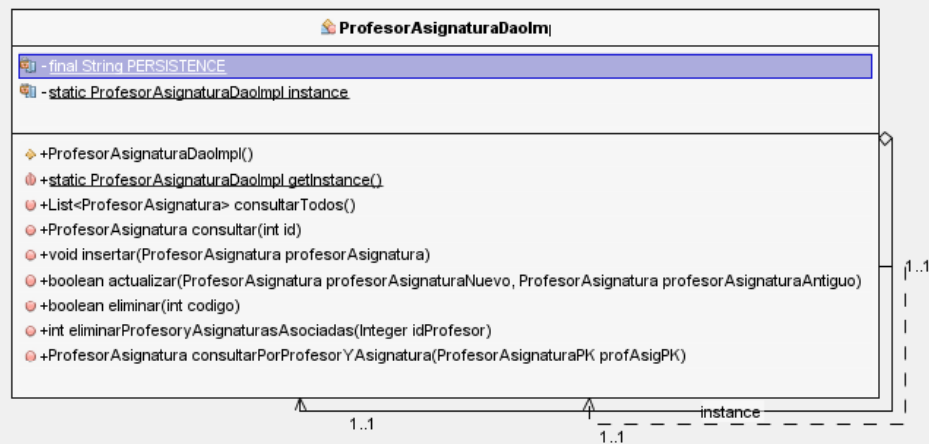
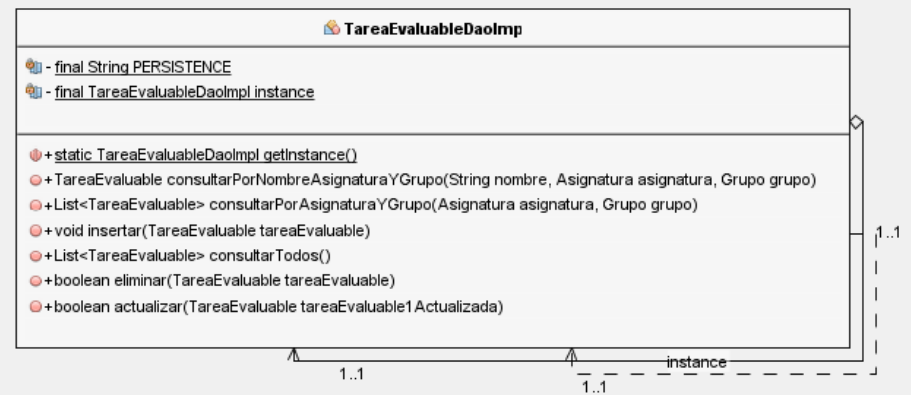
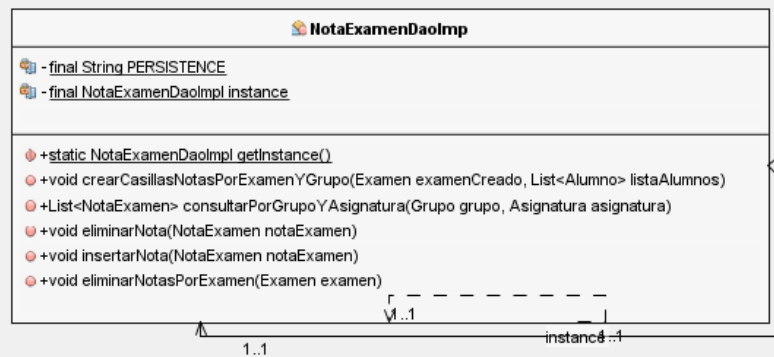
- ~ List<Profesor> consultarTodos()
- ~ Profesor consultar(int id)
- ~ Profesor consultarPorEmailYContrasena(String email, String contraseña)
- ~ void insertar(Profesor profesor)
- ~ boolean actualizar(Profesor profesor)
- ~ boolean actualizarDatosUsuario(Profesor profesor)
- ~ boolean eliminar(int codigo)
- ~ List<Profesor> consultarProfesoresSonTutores()
- ~ Profesor consultarProfesorAsignatura(int codigo)
- ~ int consultarIdPorNombreYApellidos(List<String> nombres)
- ~ Profesor consultarTutor(Integer id)
- ~ Profesor consultarPorContrasena(Profesor prof, PasswordField txtOldPass)
- ~ boolean actualizarContrasena(Profesor prof, PasswordField txtNewPass)

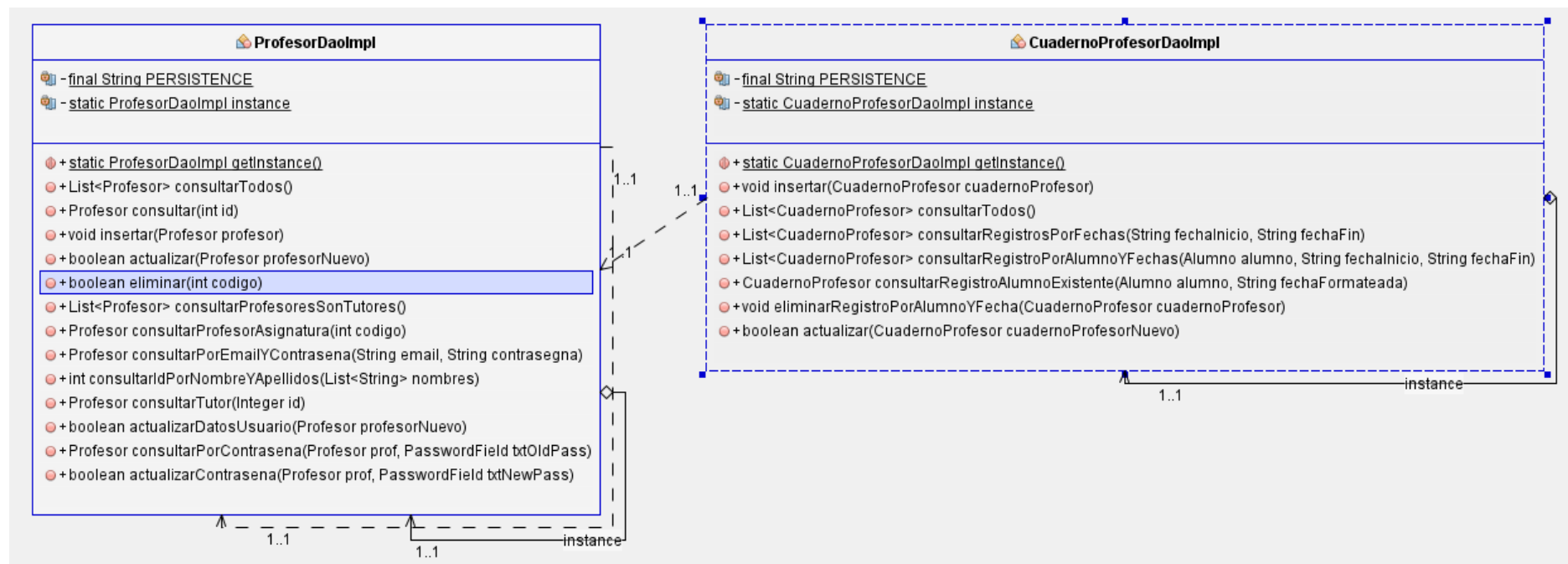


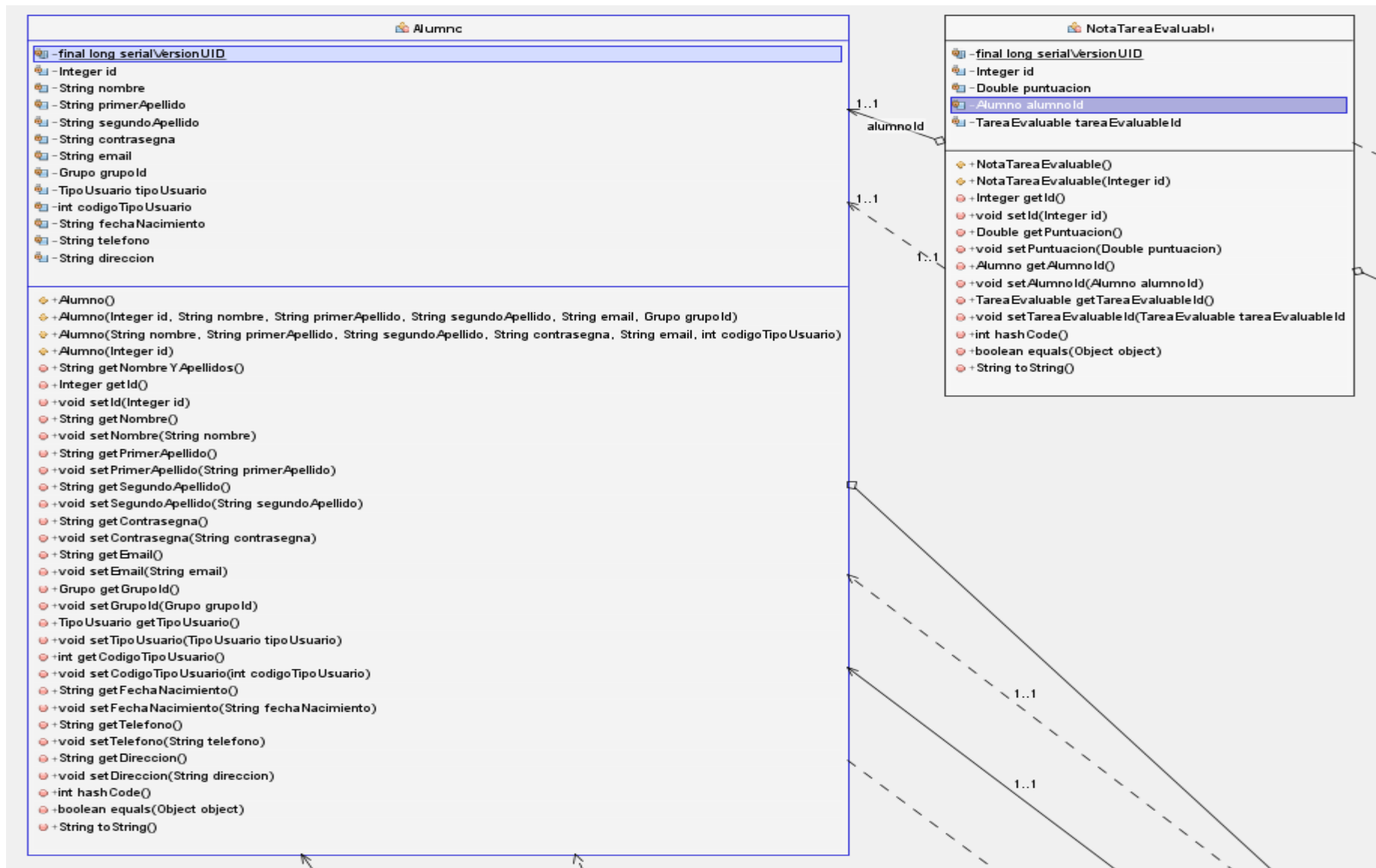


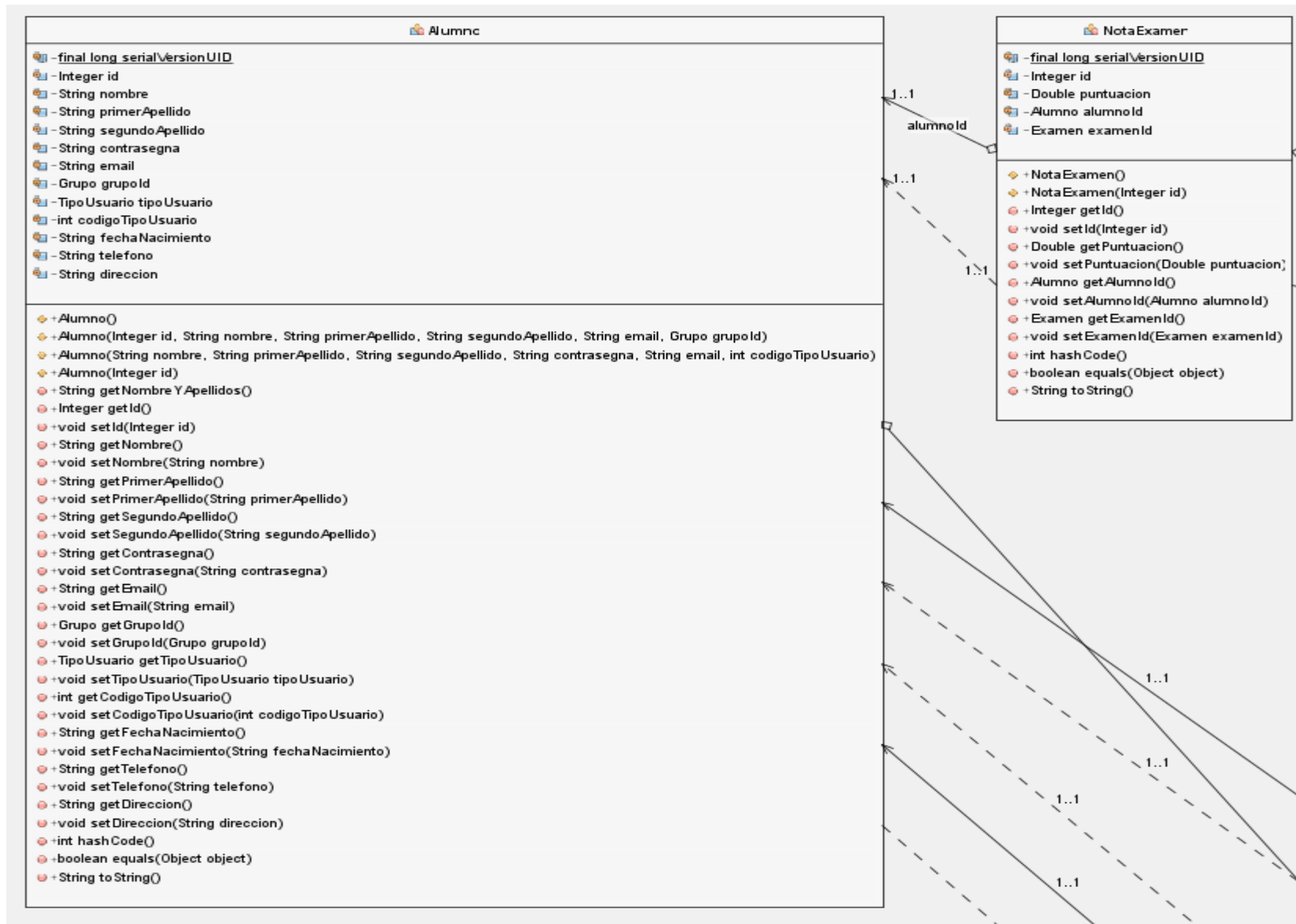


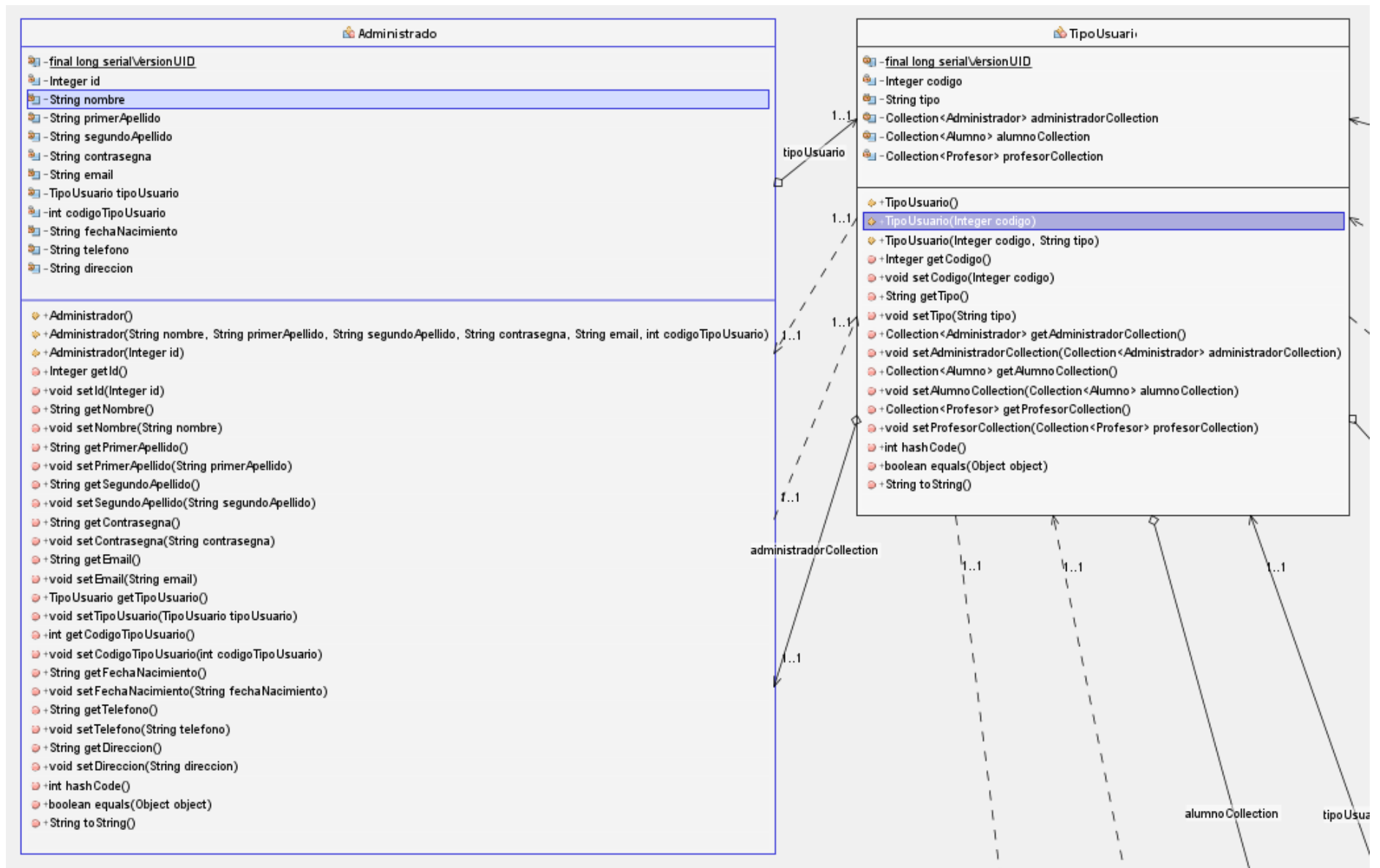




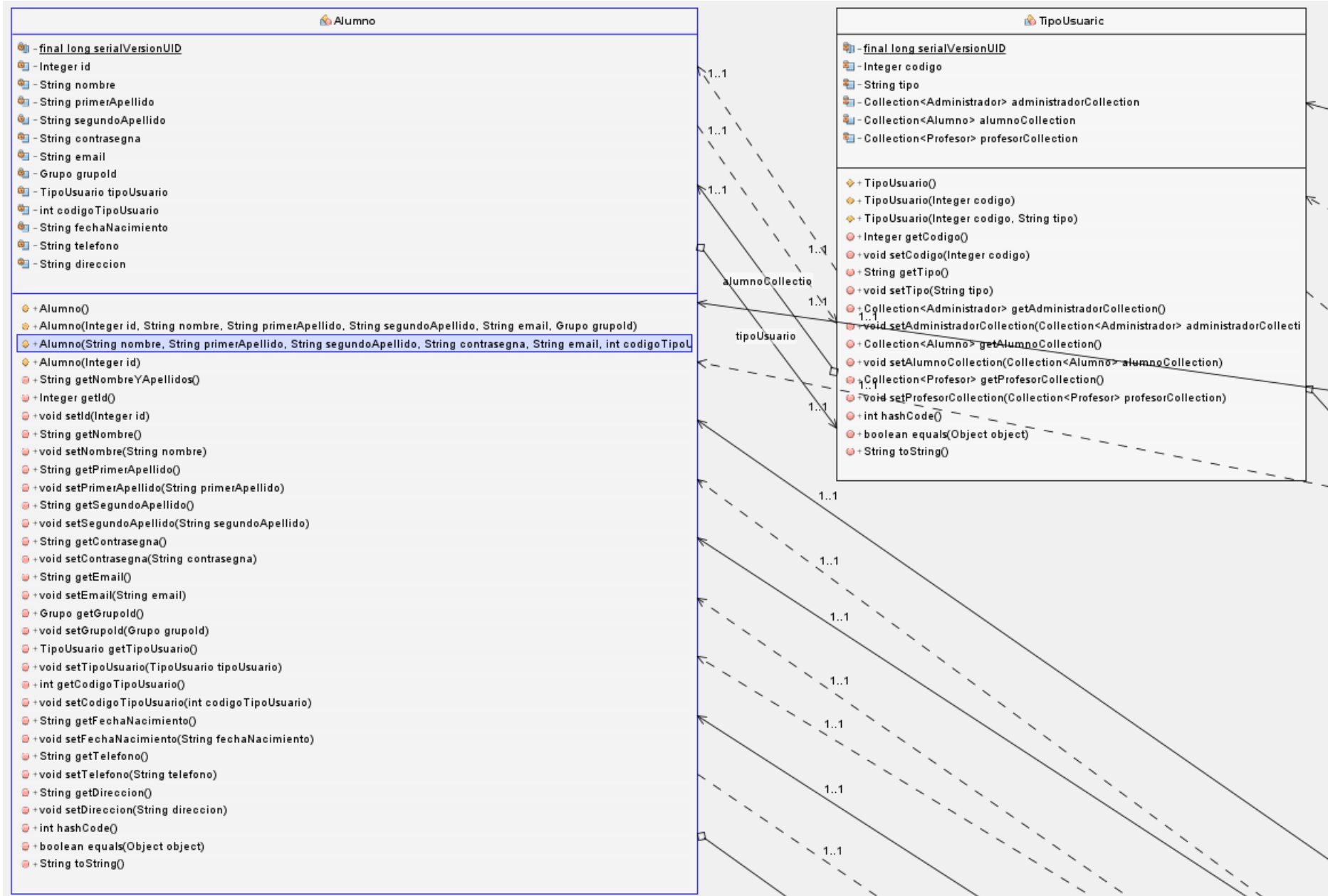


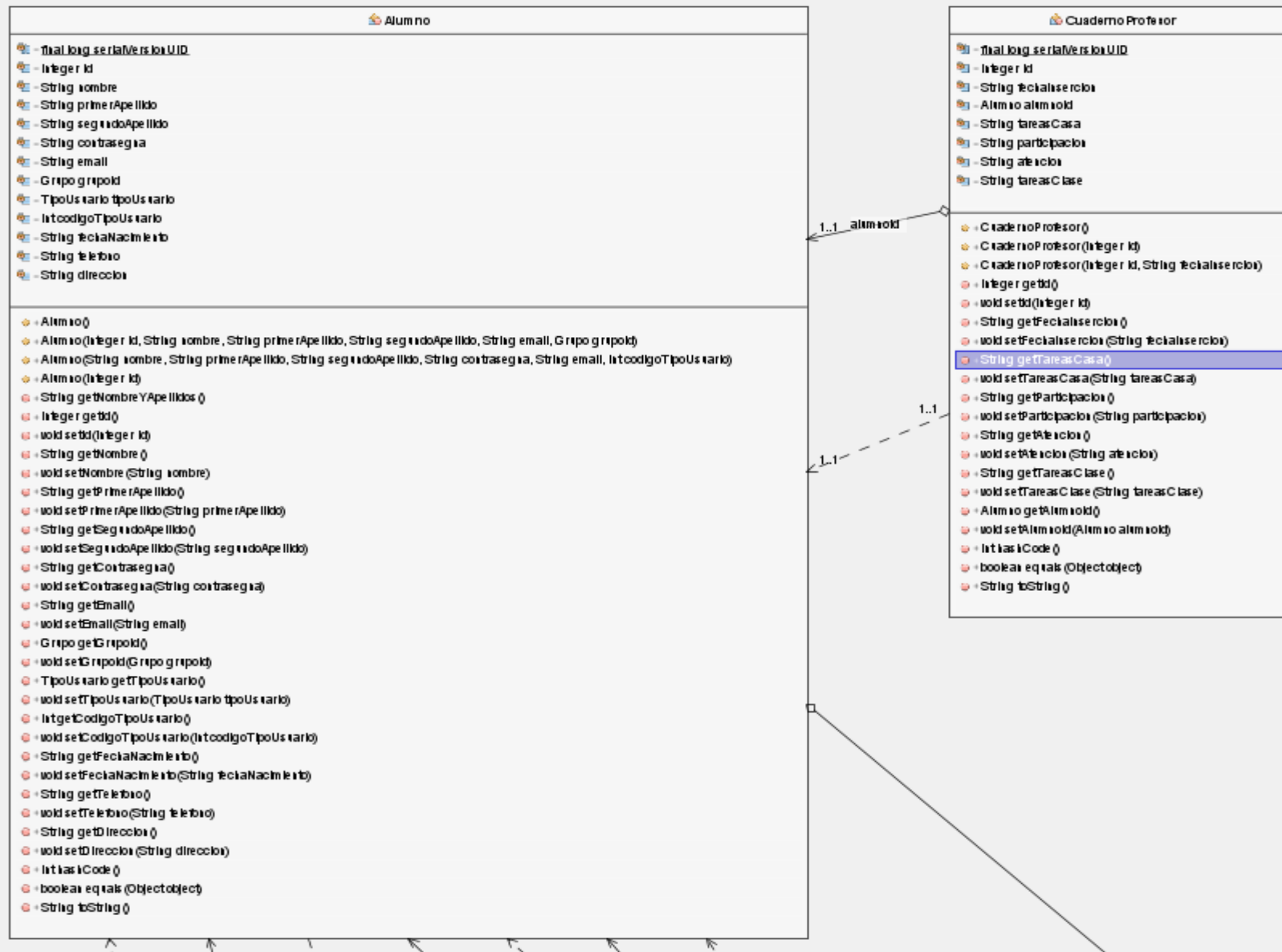




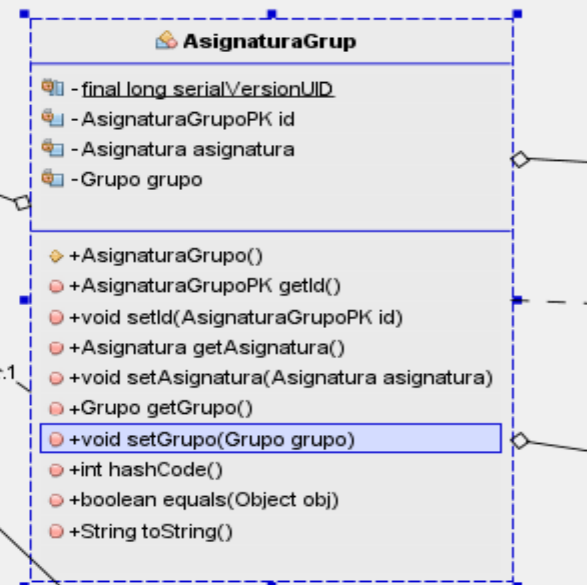












1..1  
grupo

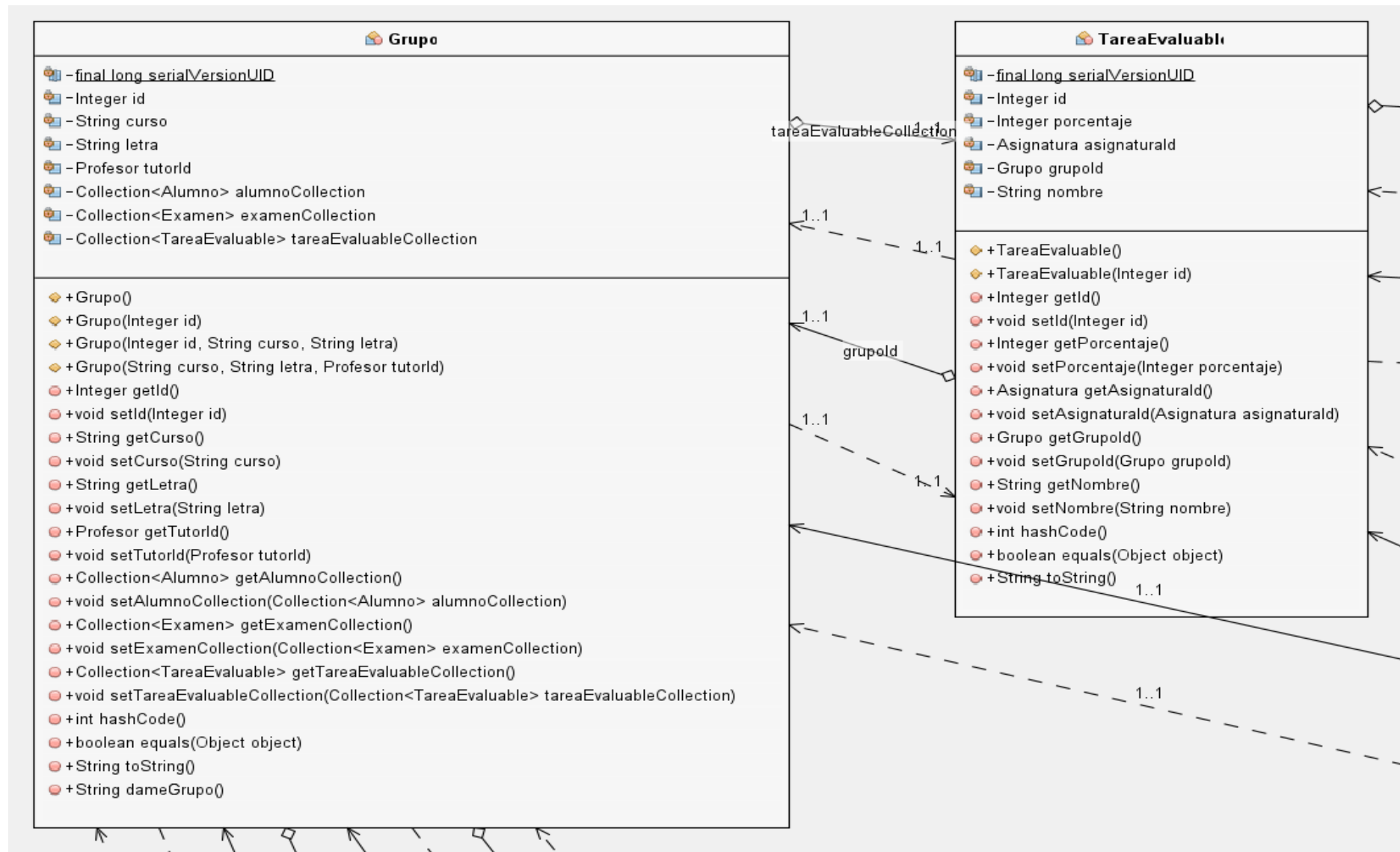
1..1  
1..1

1..1

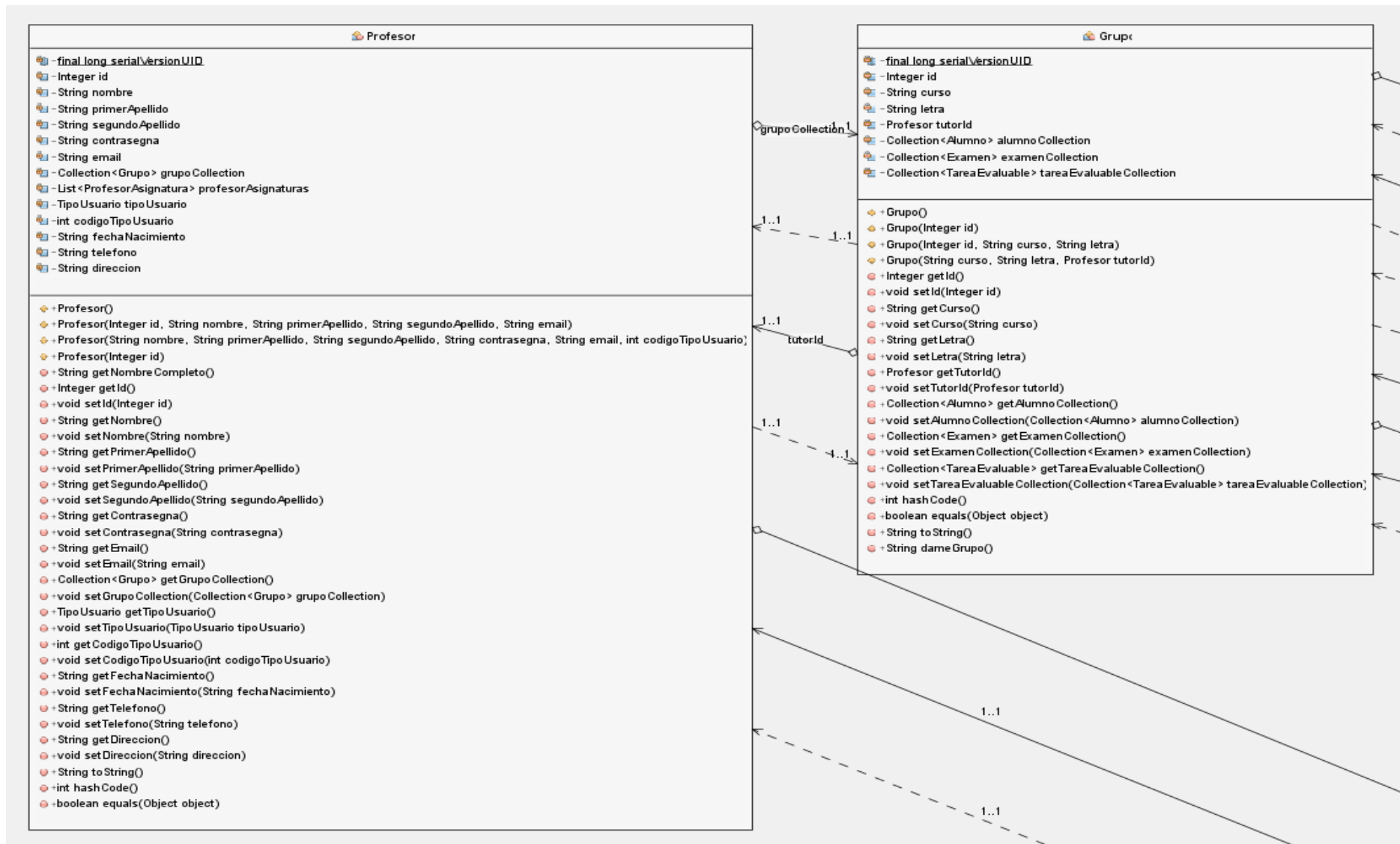
1..1

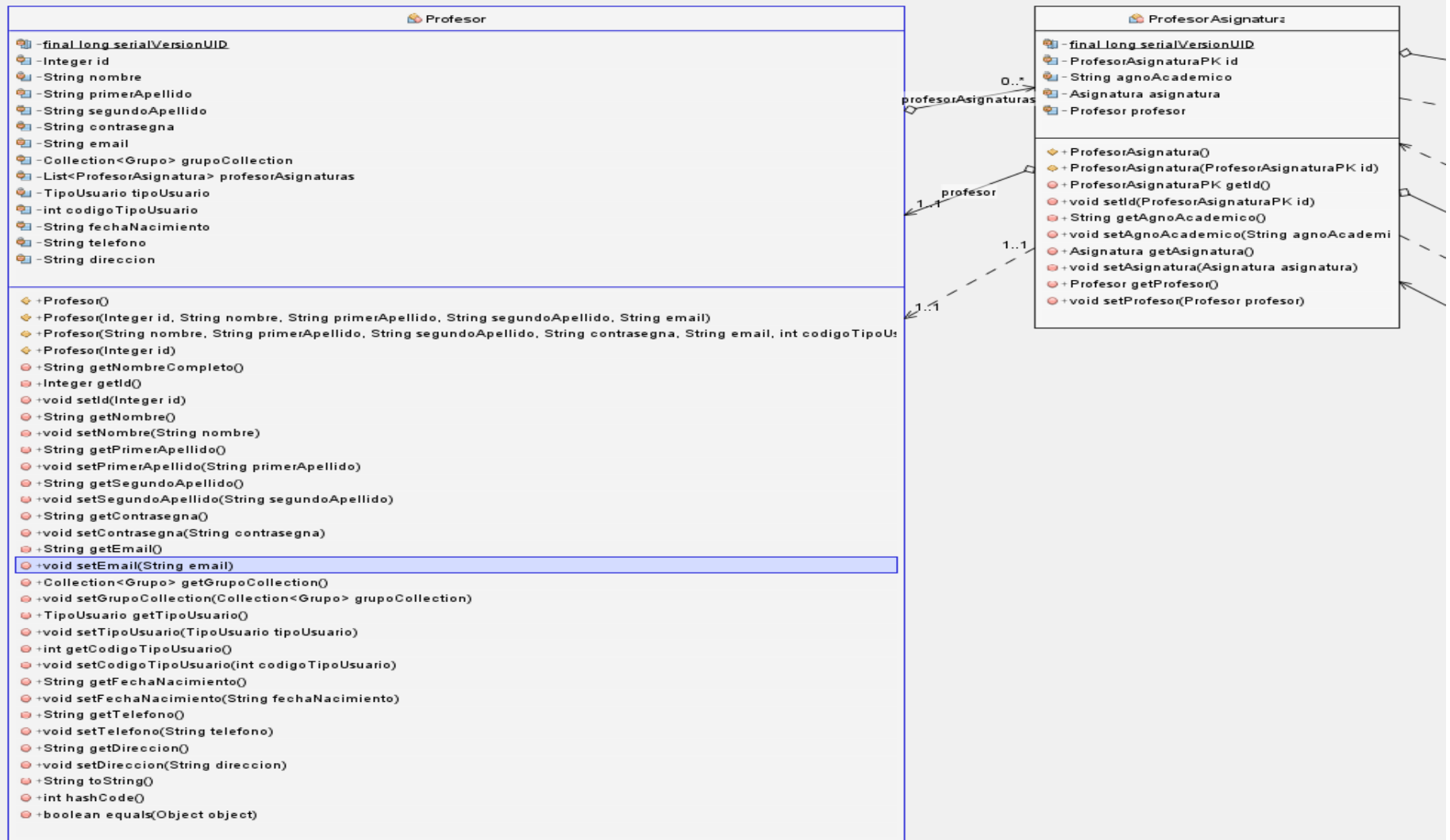
1..1

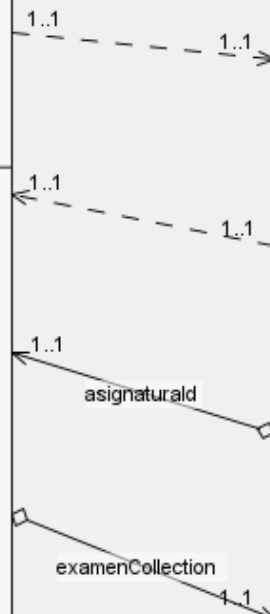
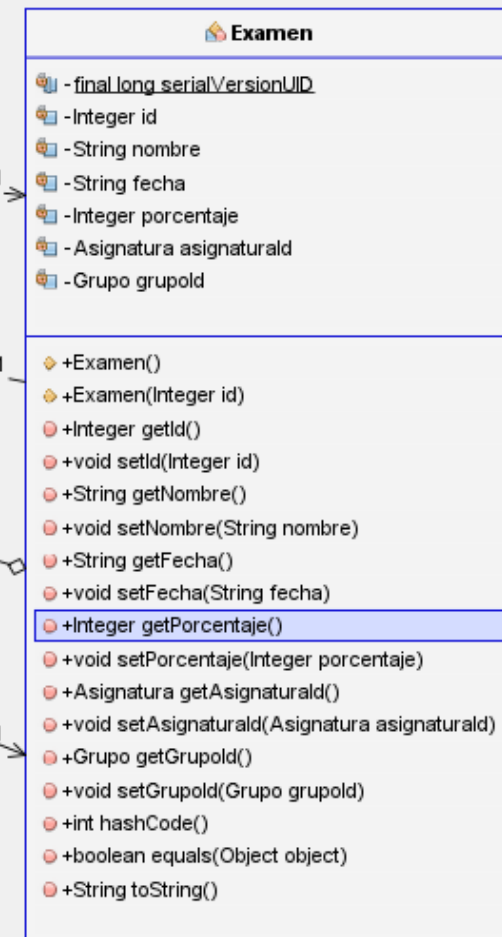
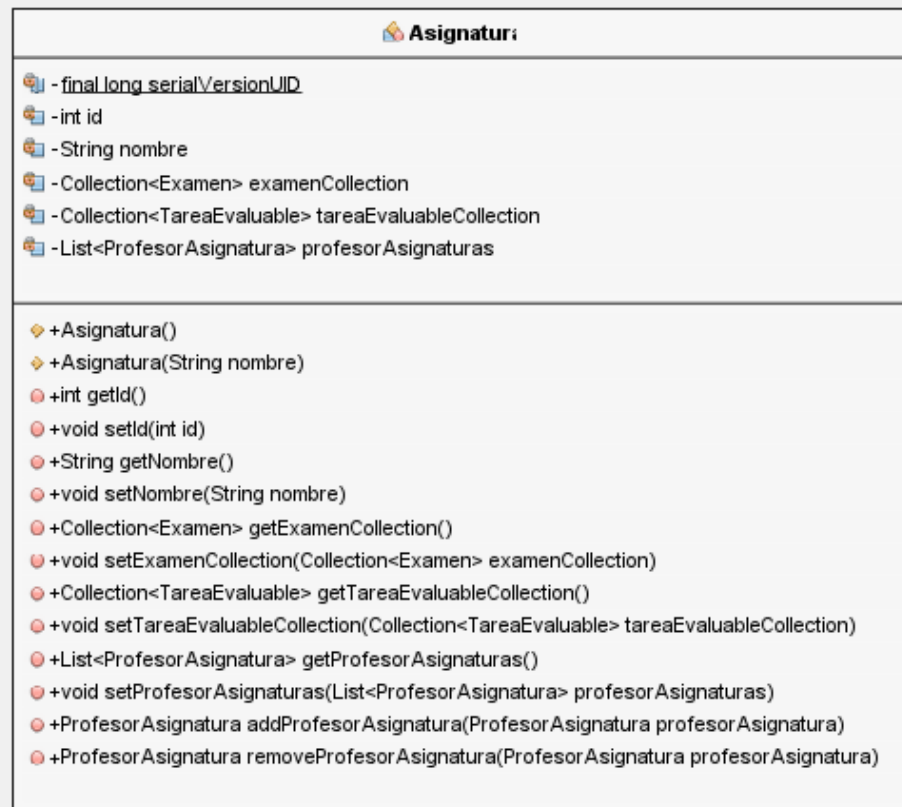
tareaEvaluableCollection

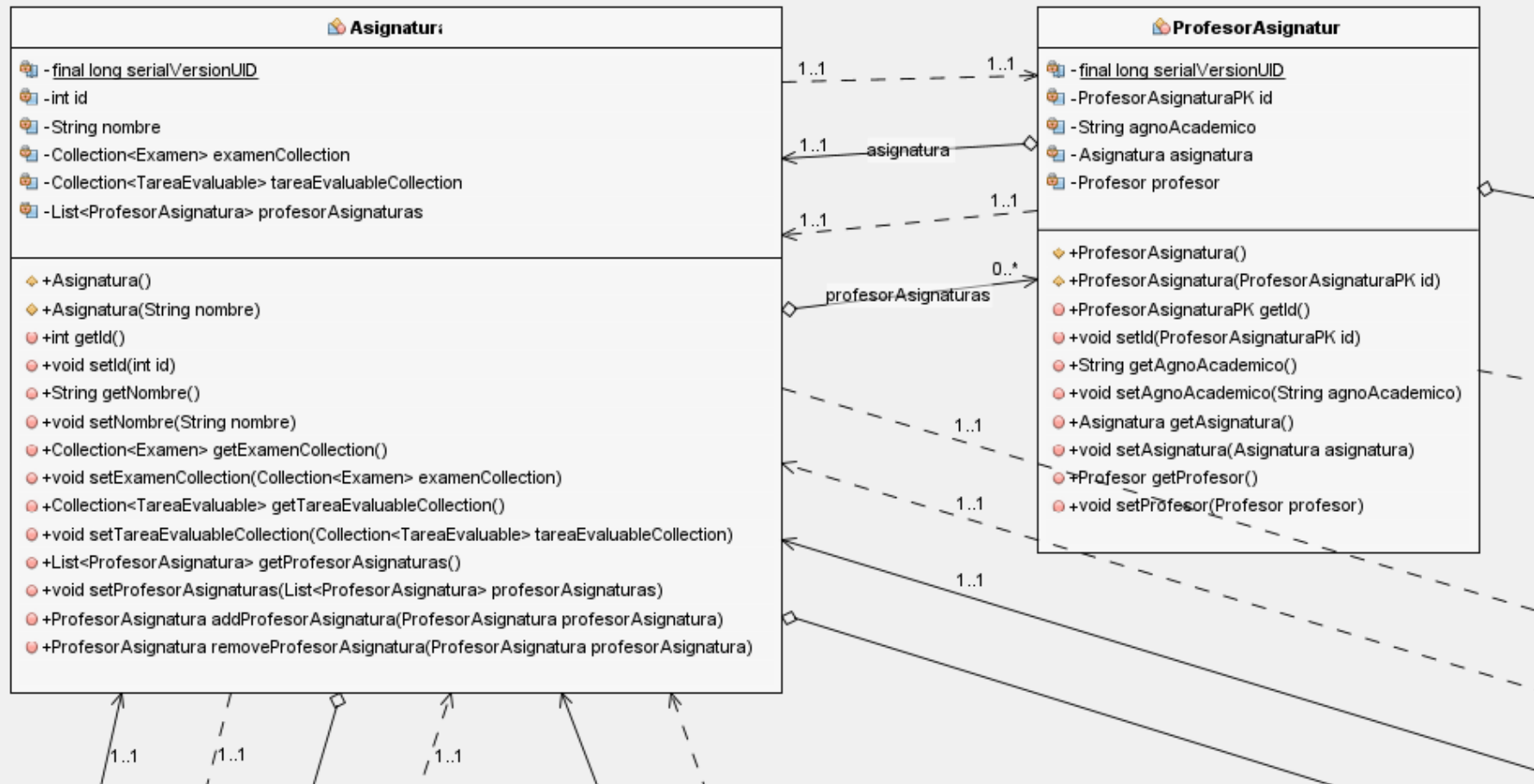


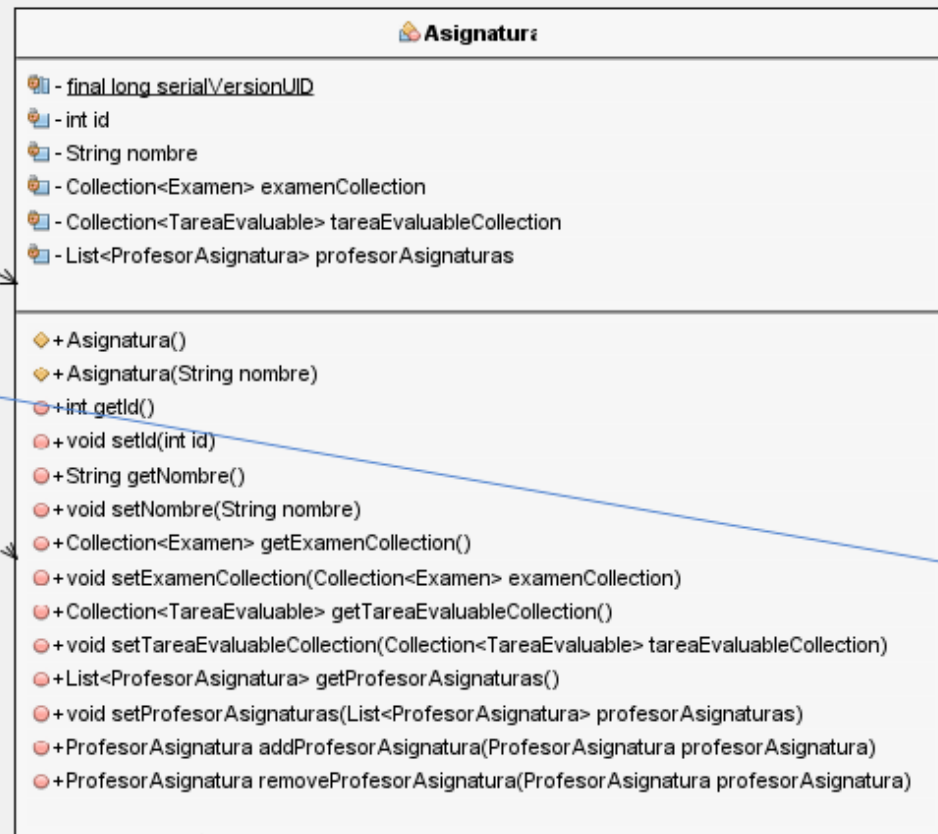
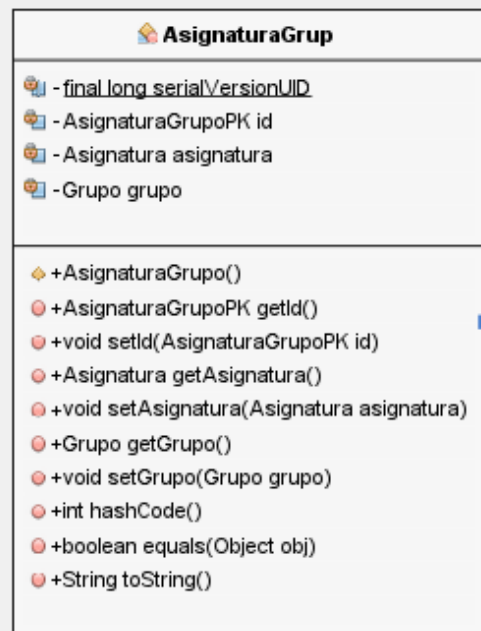










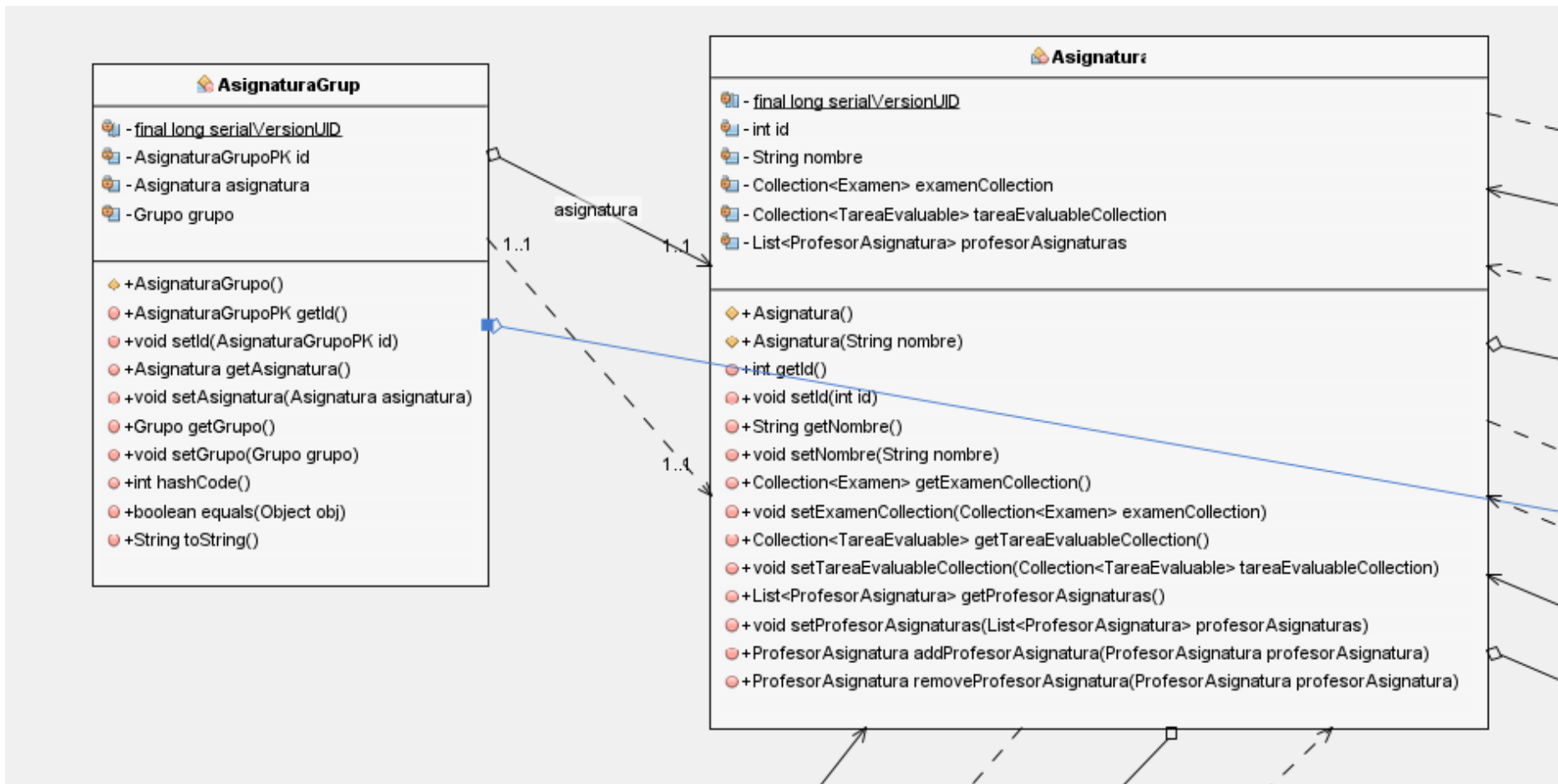


asignatura

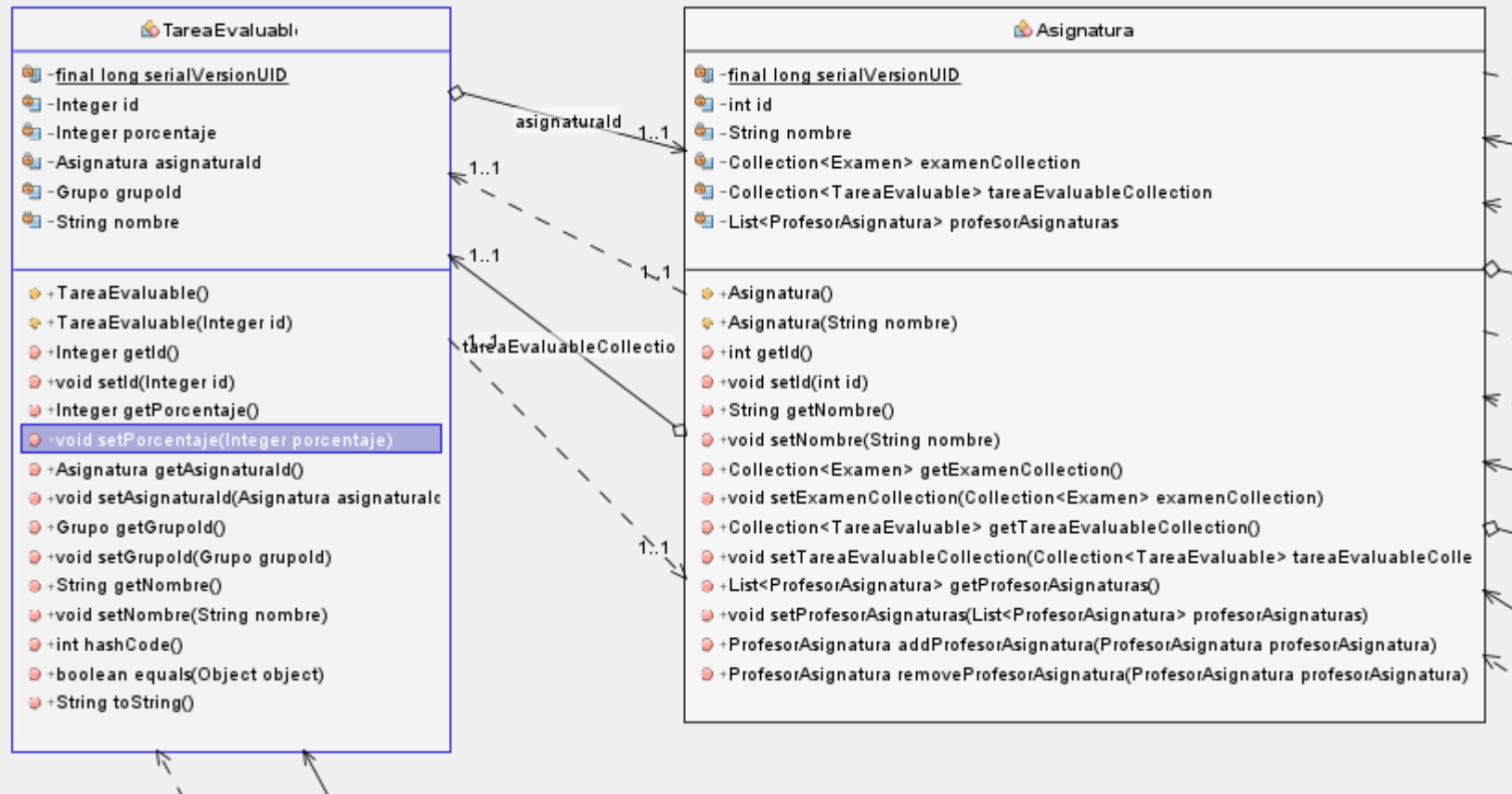
1..1

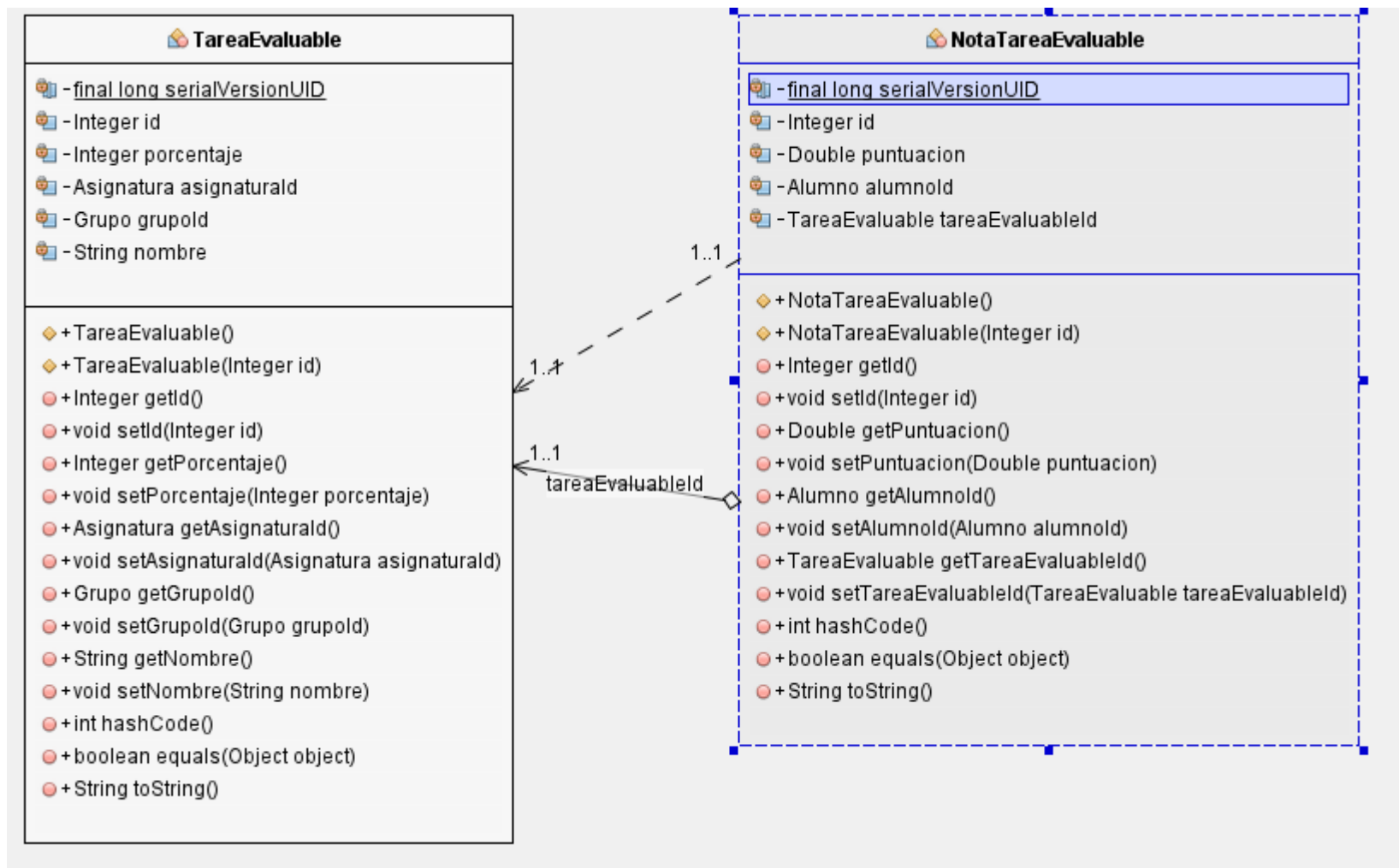
1..1

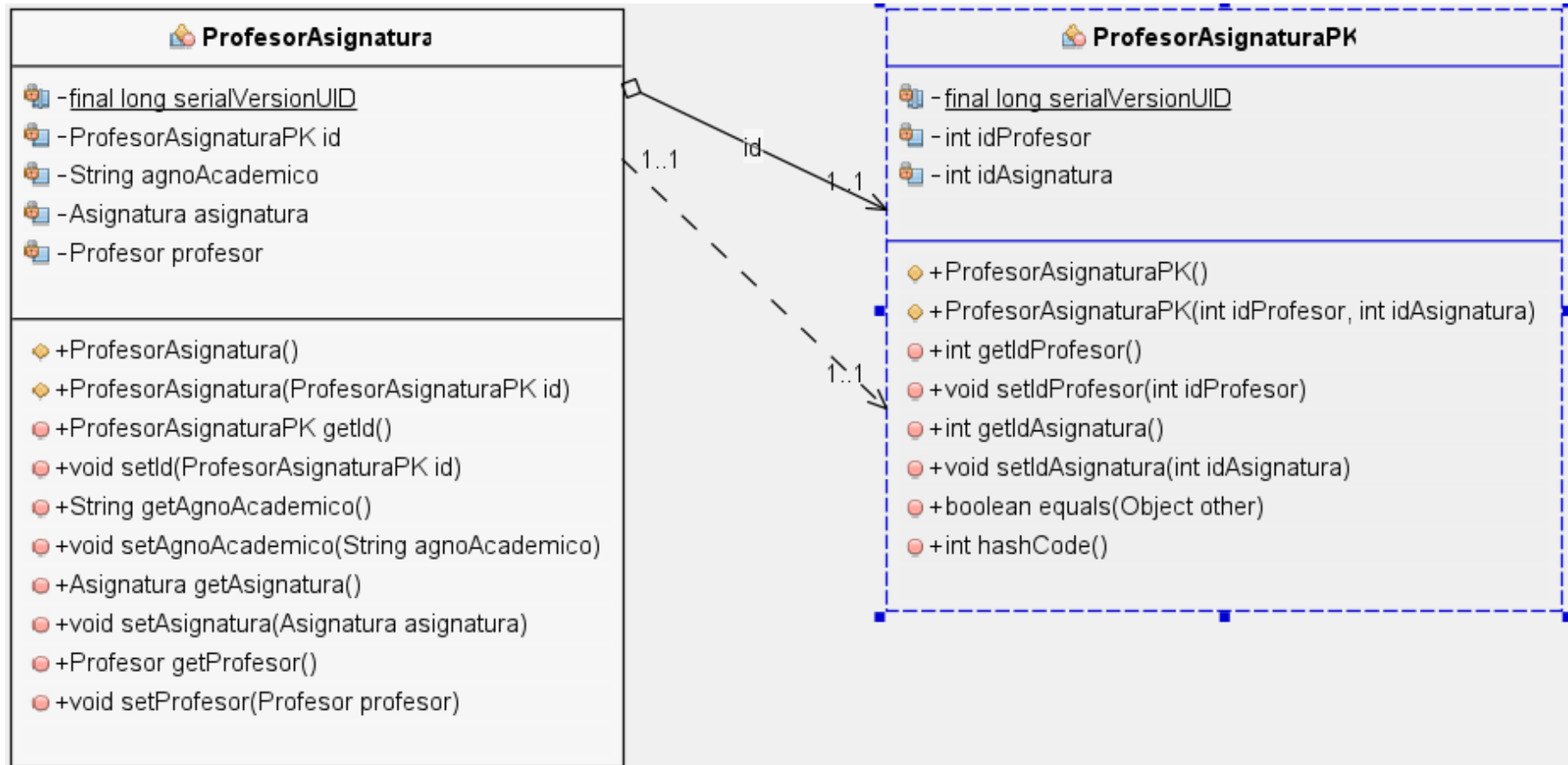
1..4

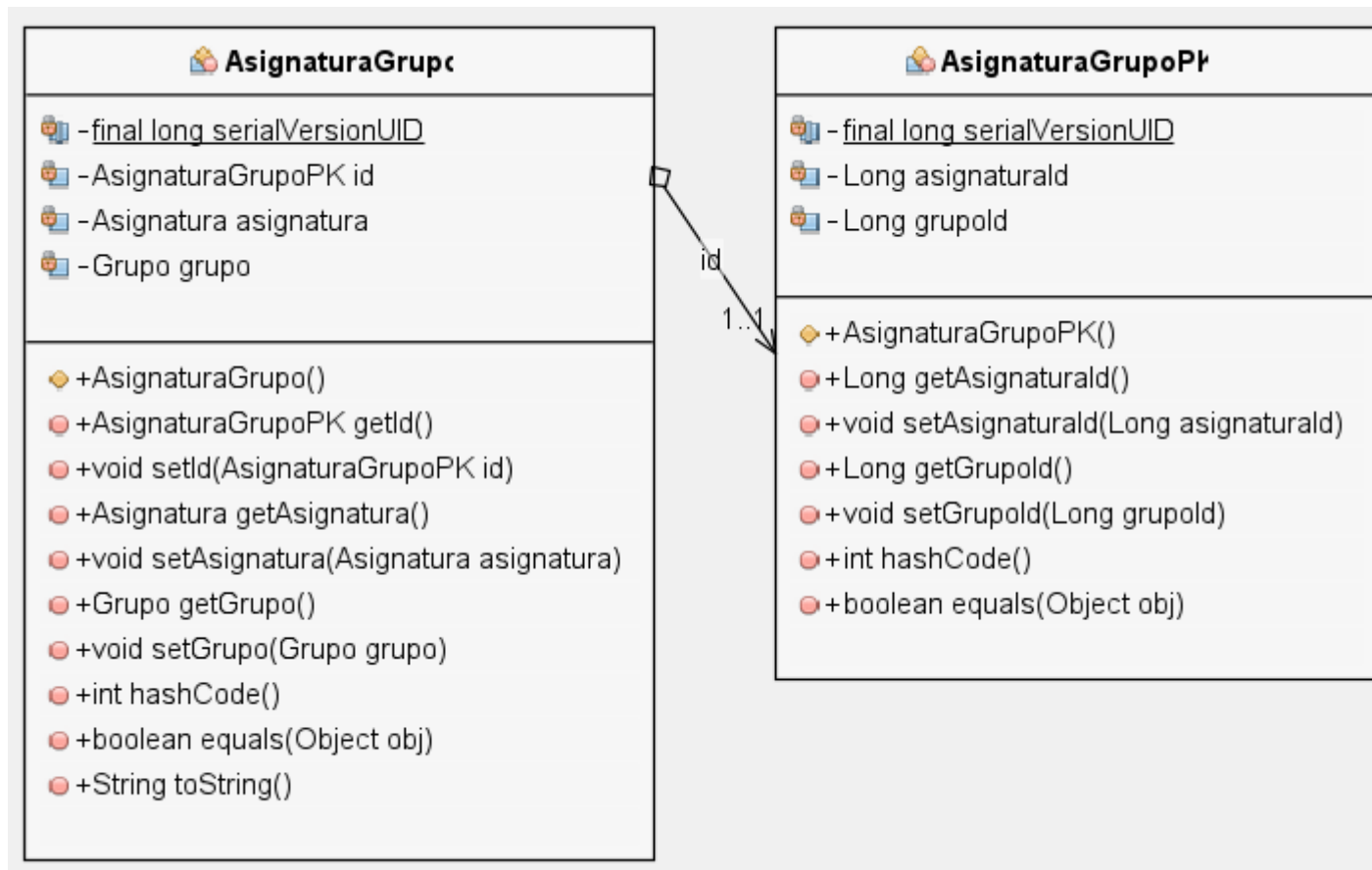


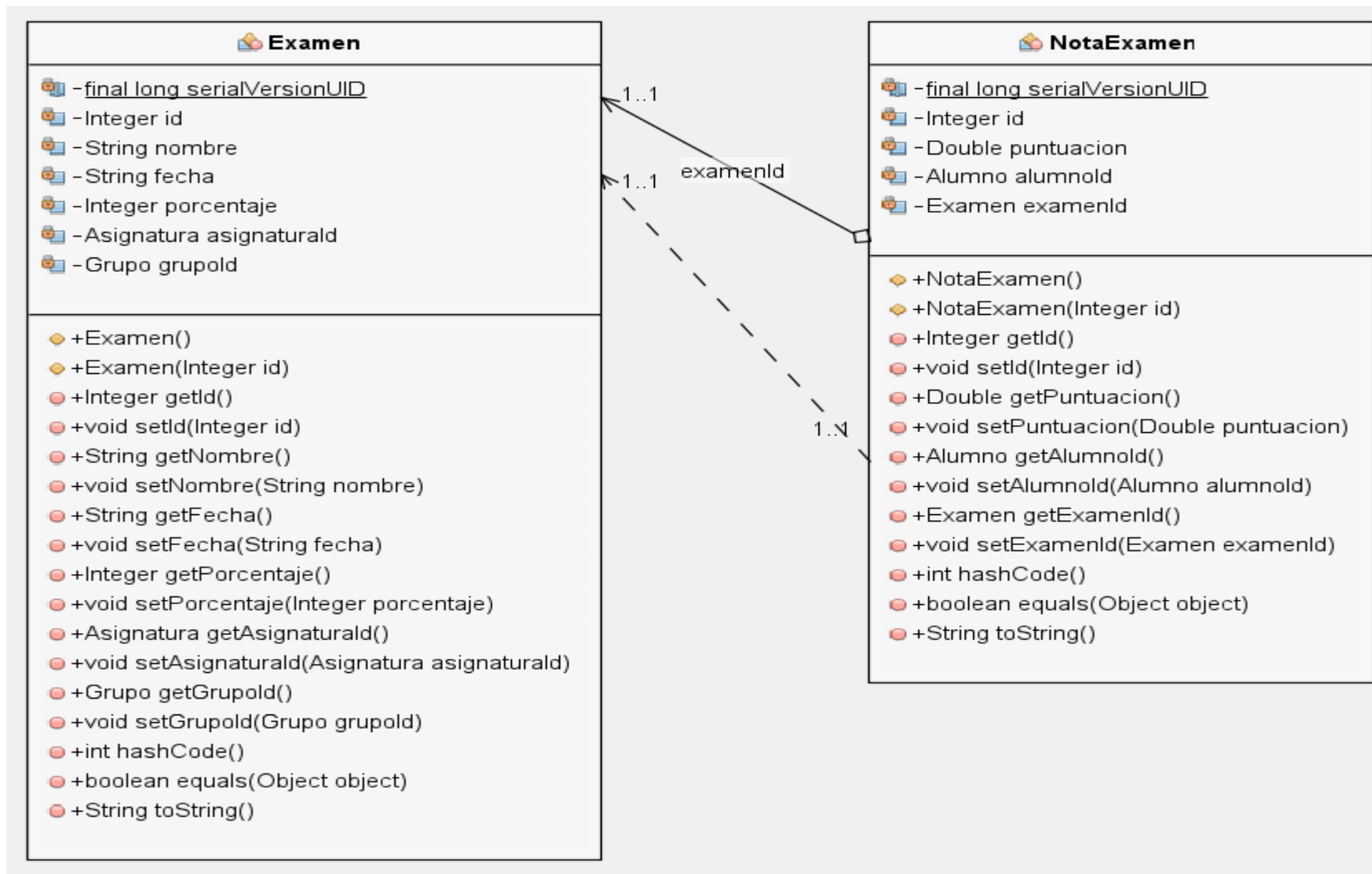












6.5. Tabla de gastos.

Concepto	Costo Unitario	Cantidad	Total
Desarrollador	2.000 e.	2	4.000 e.
Servidores	200 e.	2	400 e.
Reserva gastos imprevistos	-	-	1000 e.
Total			5.400 e.