Juan Andrés Moreno – 202321829

Juan Pablo Caicedo - 202320113

Karina Muñoz - 202321085

**Entrega 2 – Grupo 3**

Los diagramas de los puntos a, b, c y e se pueden ver más claros en la carpeta “diagramas” adjunta en la entrega.

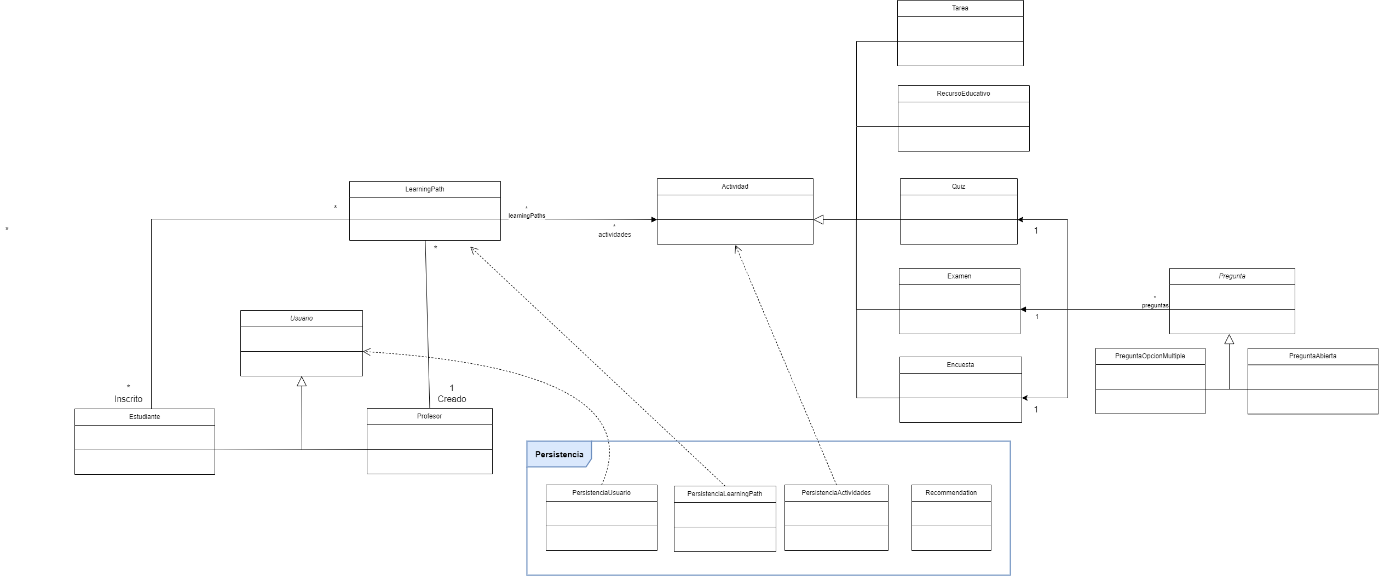
1. **Diagrama de clases del diseño final:**

**Diagrama, Esquemático

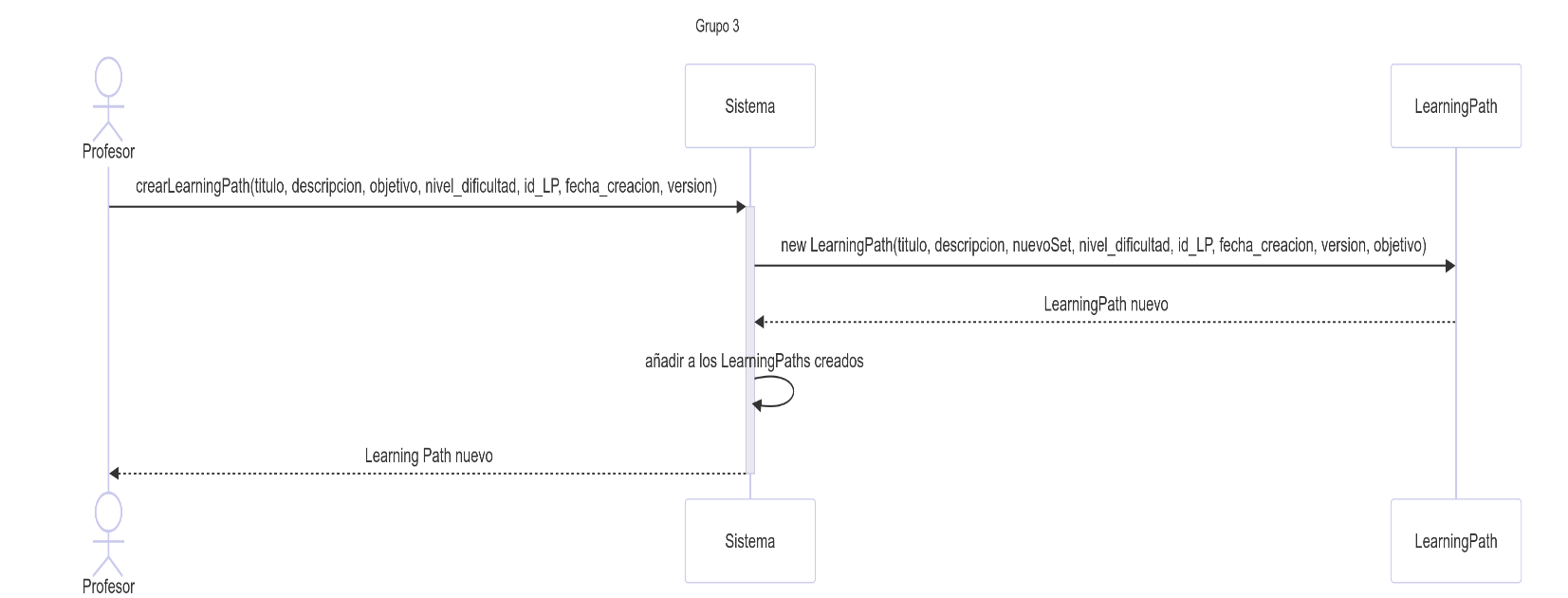
Descripción generada automáticamente**<https://drive.google.com/file/d/1OifF2qdR8H44JejoEk4KVqLHCaCbCYql/view?usp=sharing>

1. **Diagrama de clases de alto nivel:**

<https://drive.google.com/file/d/1YCBVLF7xoK3jeTJ96jEyM9z7XCQXJg68/view?usp=sharing>

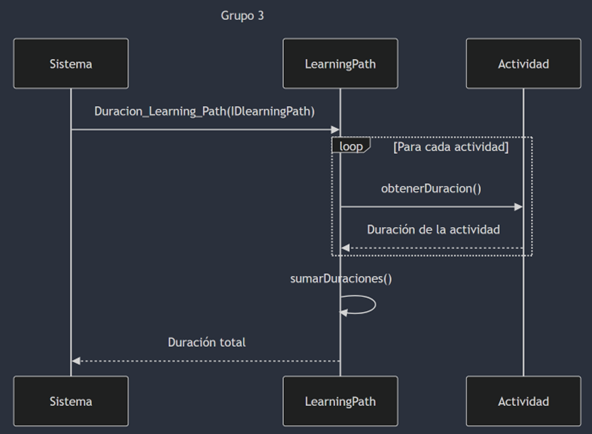


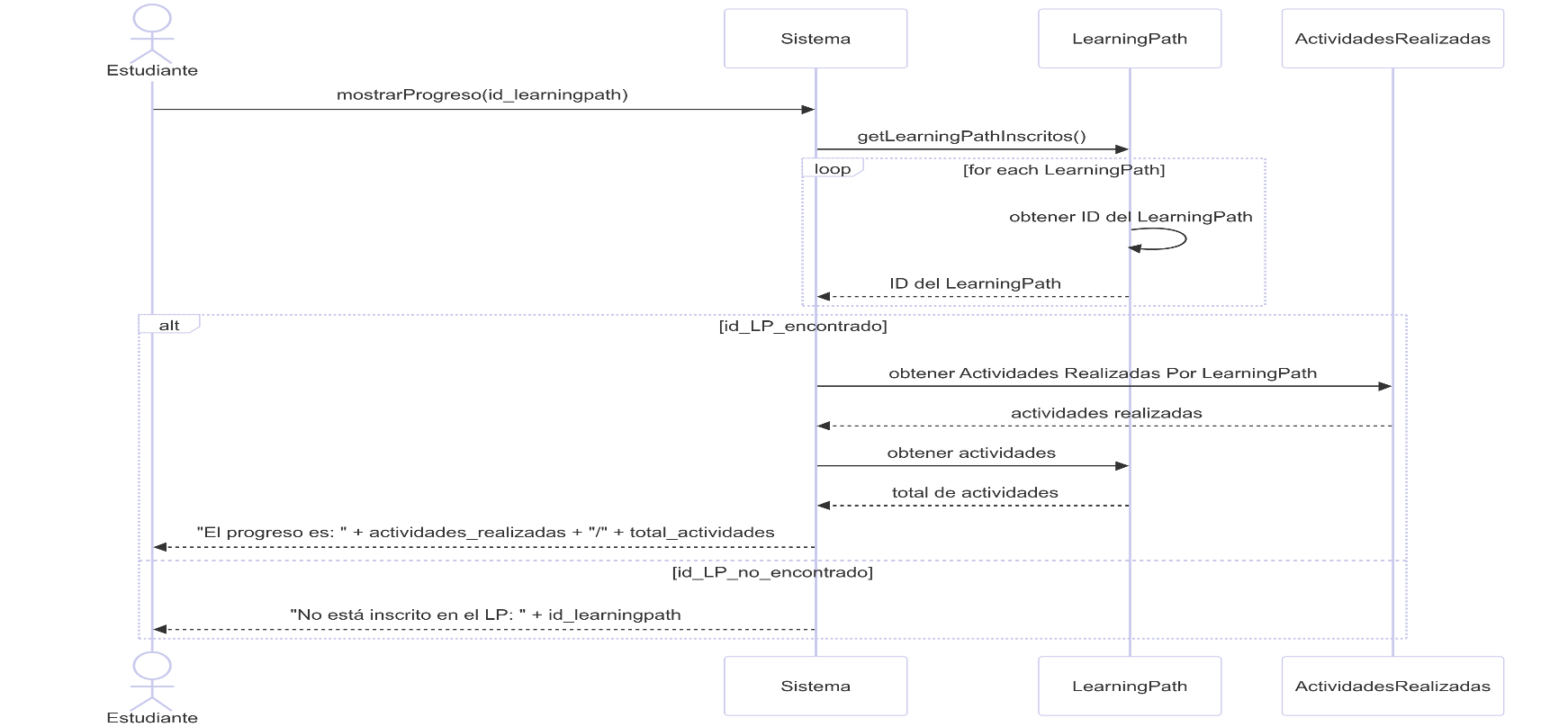
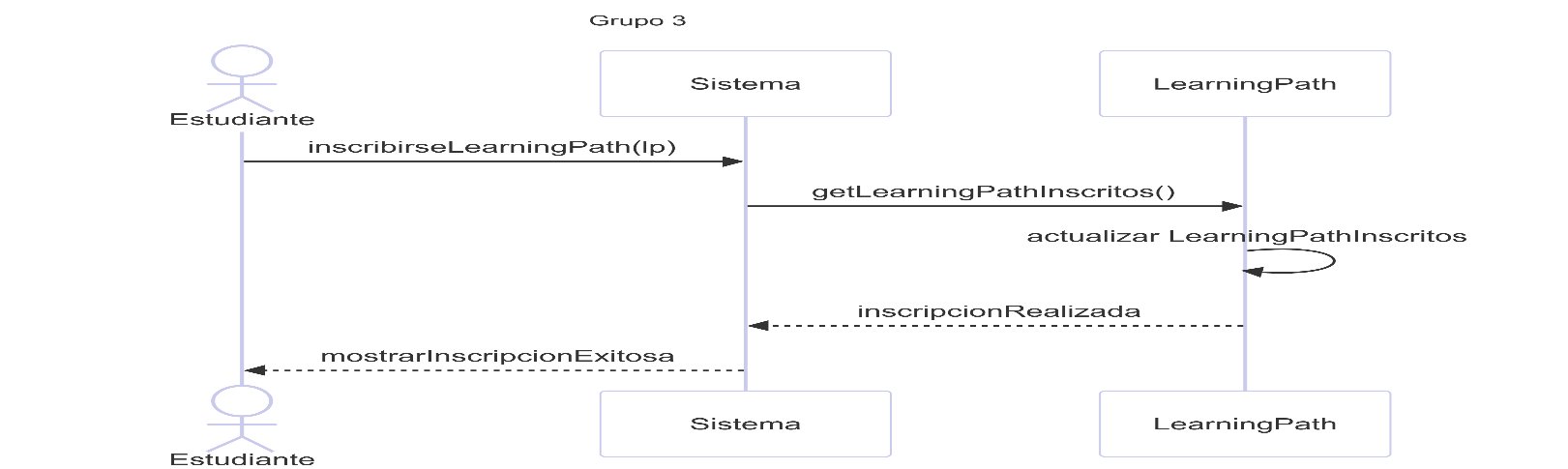
1. **Diagramas de secuencia:**

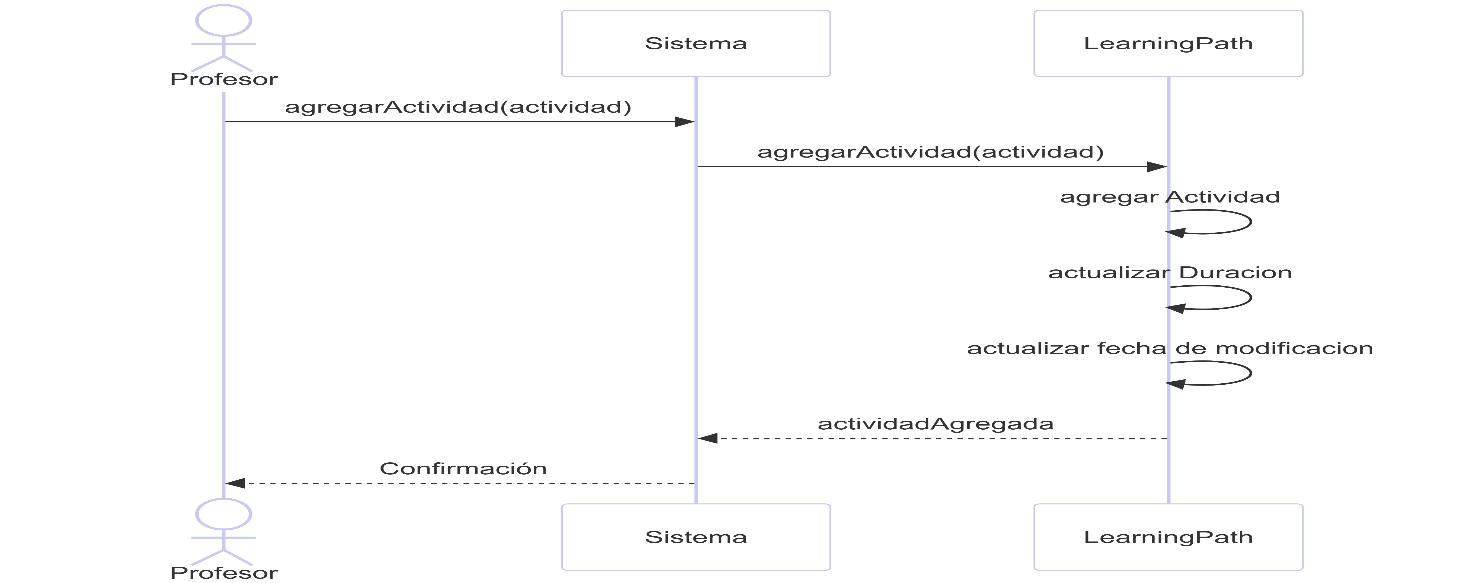
****

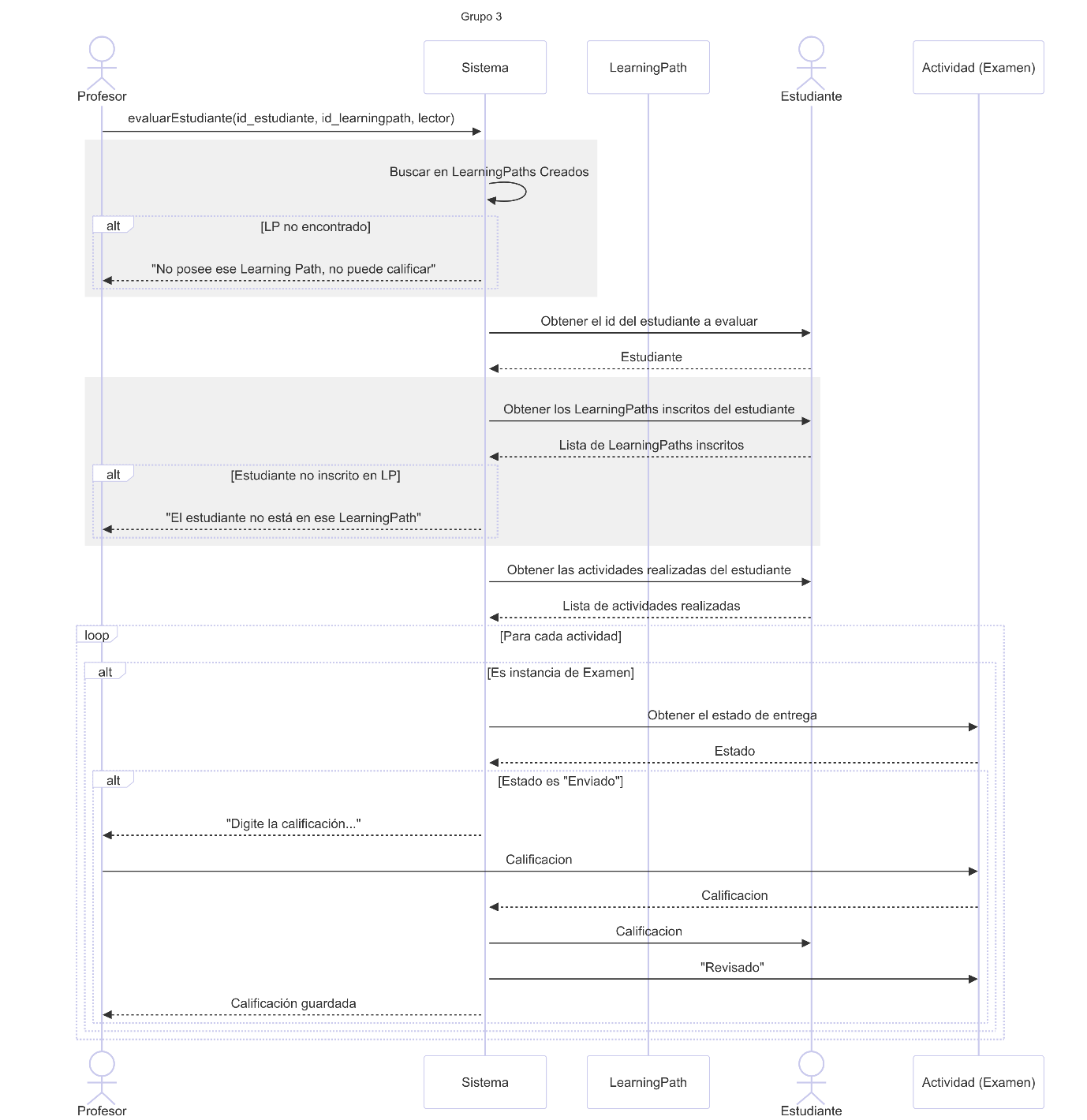
**Diagrama

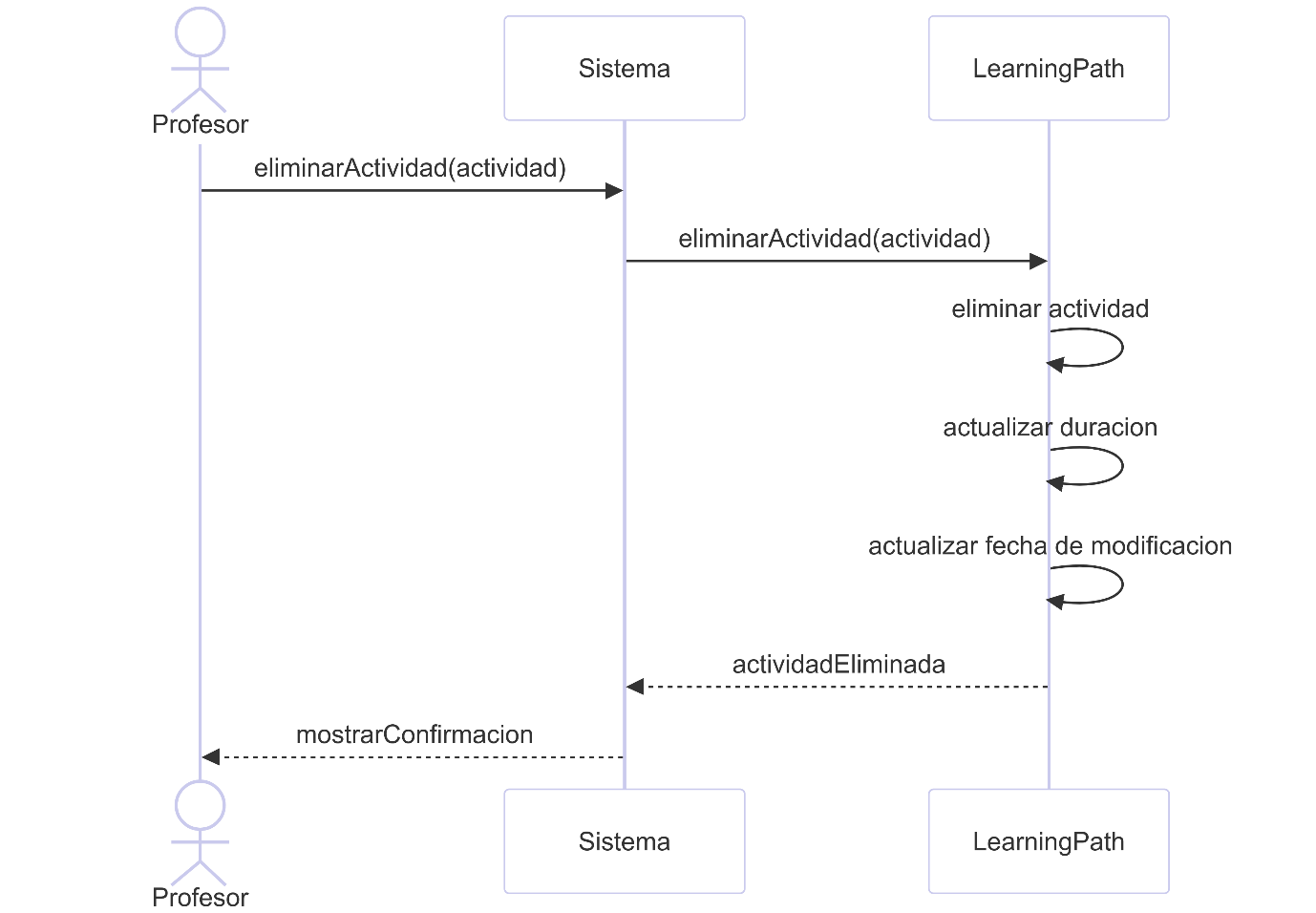
Descripción generada automáticamente**

****

****

****

****

****

**Diagrama

Descripción generada automáticamente**

1. **Explicación/Justificación de las decisiones del diseño:**

* **El diseño está estructurado en torno a tres clases o entidades principales: Usuario, LearningPath y Actividad**.
* En estas clases, se puede reflejar la jerarquía del sistema de los LearningPaths. Elegimos esta estructura debido a que consideramos que representa de manera clara y lógica las relaciones entre todas las clases y como se conectan entre sí.
* Los **usuarios** son quienes interactúan con todo el sistema de los Learning Paths y sus actividades, lo cual facilita la gestión del contenido y el seguimiento del progreso. El UML muestra una relación de herencia entre Usuario, Estudiante y Profesor. Por lo cual, gracias a esa estructura se permite crear una clara conexión entre ellos, compartir atributos comunes y definir comportamientos específicos dependiendo del tipo de usuario. Adicionalmente, facilita mucho la implementación de control de acceso a cada clase o sistema dependiendo del tipo de usuario que quiera acceder.
* Por otro lado, cada **LearningPath** tiene una relación de composición con **Actividad**. Para esto, elegimos composición en vez de agregación ya que, las actividades prácticamente son parte integral y fundamental de un LearningPath, es decir la vida de una actividad está ligada la del LearningPath que la contiene. Por último, esta relación facilita la gestión y organización del contenido tanto del LearningPath como de las Actividades.
* El diagrama tambien demuestra que existen diferentes tipos de **actividades** como Tarea, RecursoEducativo, Quiz, Examen y Encuesta, los cuales heredan de la clase Actividad. Decidimos realizar esta relación de herencia ya que, como dicho anteriormente, esto nos permite definir atributos y comportamientos comunes entre sí, implementar características específicas para cada tipo de actividad y facilita la extensión del sistema en caso de que lleguen a existir nuevos tipos de actividades no implementados.
* **Relaciones entre Clases:**
* **Profesor – LearningPath:** Esta relación refleja que solo los profesores pueden crear, eliminar y acceder a Learning Paths, lo que ayuda a mantener la calidad y consistencia del contenido.
* **Estudiante – LearningPath:** Esta relación permite a los estudiantes inscribirse e interactuar en múltiples Learning Paths y a los Learning Paths tener múltiples estudiantes, reflejando la naturaleza de muchos a muchos de esta relación.
* **Actividad – Actividad:** La auto relación consigo misma permite establecer prerrequisitos entre las mismas actividades para facilitar la creación de Learning Paths estructuradas y progresivas.
* Cada una de la clase incluye tanto atributos relevantes como métodos clave para el funcionamiento de esta. Cada atributo fue seleccionado para capturar la información esencial de cada entidad, y por esto mismo, los métodos reflejan las operaciones principales que cada entidad puede realizar. Esta estructura proporciona una base sólida para la implementación, la cual nos permite asegurarnos que cada clase contiene la información y las capacidades necesarias para cumplir su rol dentro de todo el sistema.
* **Persistencia:**
* Decidimos implementar persistencia ya que nos proporciona una solucion firme y flexible para el almacenamiento de datos en el sistema, permitiendo un manejo eficiente y claro de las diferentes entidades y así mismo, facilitando posibles futuras expansiones y el correcto mantenimiento del sistema.
* Decidimos usar Json por su flexibilidad, legibilidad y fácil integración con Java.
* Los métodos de carga y guardado nos permiten realizar una interfaz clara para todas las operaciones necesarias.
* **PersistenciaUsuario:** Decidimos implementar una clase de persistencia para Usuario, ya que permite gestionar de manera enfocada la información de cada usuario, como sus datos personales, roles y permisos en el sistema. Esta estructura facilita el mantenimiento y la recuperación eficiente de datos, además de asegurar que la información esté organizada y sea accesible cuando en el futuro se implementen posibles nuevas funcionalidades o requerimientos en el sistema. Permitiendo así una gestión más segura y controlada del Usuario.
* **PersistenciaLearningPath:**  Decidimos crear una única clase de persistencia para LearningPath ya que centraliza la lógica de almacenamiento y recuperación de estas. Además, facilita la gestión de relaciones complejas entre LearningPaths y Actividad. Permitiendo así una fácil extensión para incluir posibles nuevas características.
* **PersistenciaActividades**: Para actividades, también se decidió crear una única clase que implementara persistencia. Esto debido a que maneja eficientemente los diferentes tipos de actividades como tarea, quiz, examen, encuesta o recurso. Adicionalmente, facilita la implementación de operaciones específicas para cada actividad. Nuevamente permitiendo una gestión flexible entre la relación de Actividad y LearningPaths.
* El **parámetro rutaArchivo** ofrece flexibilidad en la ubicación de los archivos de datos, permitiendo tener una implementación más rápida.

1. **Aspectos Adicionales:**

* Decidimos añadir un mapa mental con las Exceptions que tendrá nuestro sistema. Lo cual, además de añadirle una explicación más a cómo va a funcionar nuestro diseño, proporciona una estructura jerárquica de excepciones que puede ser útil en el momento de tener que manejar diferentes tipos de errores en el sistema. Adicionalmente, permite un manejo de excepciones más específico, lo cual puede mejorar la estabilidad y la capacidad de mantenimiento que hemos creado.

<https://lucid.app/lucidchart/a8c49ba1-fb6b-42a0-8c08-9aa395af8269/edit?viewport_loc=-254%2C139%2C2327%2C1016%2C0_0&invitationId=inv_8de81820-7c3c-4a23-8b85-de4f2d714eee>

