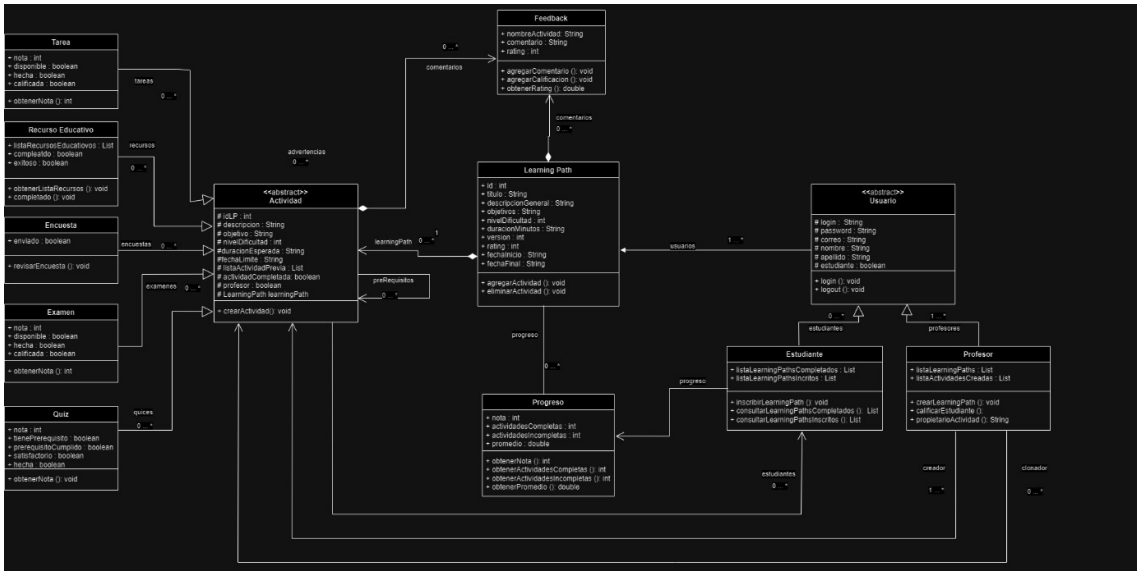
**Análisis modelo de dominio**

Sofia Alcalá - 202321623
María José Rodríguez López - 202226884
Santiago Álvarez Ramírez - 202321575

Mientras implementábamos el sistema diseñado en la primera entrega del proyecto fuimos modificando ciertos aspectos que ya teníamos previamente pensados tales como algunos atributos y clases. Seguidamente se explicará por que se decidió asignar los métodos de cada clase y se explicará la relación de los atributos con cada método.




Actividad: el único método asignado en esta clase es crear actividad, donde se verifica que el usuario que vaya a crear la nueva actividad sea profesor, se compara el ID del Learning Path de destino y se agrega a este. En este método se hace uso de los atributos profesor, un booleano que indica si el usuario es o no profesor, el ID del Learning Path, un entero que representa el ID único de cada Learning Path y el Learning Path, el cual es el destino de la actividad. De esta clase heredan todos los tipos de posibles actividades que se pueden presentar dentro de un Learning Path específico. Estos son: Tarea, Recurso Educativo, Quiz, Examen y Encuesta.

Tarea: en tarea existe únicamente un método, obtener nota. Este se encarga de indicarle al usuario el estado y/o nota de la tarea a realizar. Los atributos usados en este método son disponibles, un booleano que indica si la tarea se encuentra disponible o no, enviada, otro booleano que se encarga de indicar si la tarea ya fue envida o no, nota, un entero en el que se almacena la nota del estudiante y calificada, un último booleano que indica si la tarea ya fue calificada por el profesor o no.

Recurso educativo: en recurso educativo existen dos métodos: obtener nota y obtener lista recursos. Obtener nota funciona con los atributos completado y exitoso, ambos booleanos que indican si la actividad fue completada y exitosa. Este método indica si la actividad fue completada y exitosa o si se debe completar. El método obtener lista recursos emplea los atributos listaRecursosEducativos, una lista con todos los recursos educativos, y recurso educativo, el recurso educativo que se añadirá a la

lista. Como ya fue mencionado, este método añadirá cada recurso educativo a la lista que los compila a todos.

Quiz: esta clase posee un solo método, obtener nota. Este cumple misma función que en el resto de las clases que heredan de actividad: se encarga de indicarle al usuario el estado y/o nota de la tarea a realizar. En este caso, dicho método tambien le indica al usuario si la actividad posee prerrequisitos que deberían ser revisados anteriormente. Los atributos empleados en este caso son disponibles, un booleano que indica si el quiz se encuentra disponible o no, hecha, otro booleano que se encarga de indicar si la tarea ya fue realizada o no, nota, un entero en el que se almacena la nota del estudiante y tienePrerequisto y prerequisitoCumplido, ambos booleanos que están relacionados con los prerrequisitos asociados al quiz.

Examen: esta clase es muy similar a la clase tarea en cuanto a métodos y atributos. De hecho, ambas clases son iguales.

Encuesta: el único método usado en esta clase es obtener nota. Este se encarga, como en las otras clases, de indicarle al usuario el estado y/o nota de la tarea a realizar. En este caso se emplea únicamente el atributo enviado, un booleano que indica si la encuesta fue enviada o no.

Learning Path: en esta clase únicamente encontramos dos métodos: añadir y eliminar actividad. Ambos, claramente, se encargan de añadir y eliminar actividades de la lista que se encargan de reunirlas. Los atributos que se usan en estos métodos son actividades y la lista a la cual se le añadirán o eliminarán elementos.

Usuario: en usuario tenemos, nuevamente, dos métodos. Estos son login y logout. Ambos se encargan de loggear al usuario y desloggear al usuario. Para esto se hace uso de los atributos login, un string que representa el nombre del usuario de, valga la redundancia, el usuario y password, otro Sting que almacena la contraseña asociada con el nombre de usuario.

Estudiante: en la clase estudiante tenemos solo dos métodos. En ambos, el único atributo que se usa es Learning Path. Estos métodos se encargan de añadir a listas los Learning Paths inscritos y los completados por el estudiante.

Profesor: en esta tenemos dos métodos: añadirLP y añadir actividad creada. En ambos solo se utiliza un atributo, en añadirLP se hace uso de Learning Path y en añadir actividad solo usamos actividad. Estos métodos se encargan de añadir a listas los Learning Paths creados por el profesor y las actividades creadas por este mismo. En esta clase existe un tercer método que decidimos no implementar por el momento debido a que no consideramos que fuera completamente relevante.

Feedback y proceso: Ambas clases tienen métodos y atributos asignados. A pesar de esto, decidimos desarrollar el código para estas clases más adelante debido a que no consideramos que sean totalmente relevantes por los momentos.

A continuación, se mencionarán algunas de las restricciones que permiten que el proyecto funcione eficientemente (e1):

Almacenamiento de la información:

La información debe almacenarse en dentro de una carpeta y se puede suponer que sólo la aplicación va a escribir y leer de esa carpeta. Es decir, esta restricción permite que los usuarios no modifiquen los archivos directamente, haciendo que toda interacción con la información almacenada debe hacerse

exclusivamente a través de la aplicación, evitando así la posible edición maliciosa de archivos por parte de usuarios.

Login y password:

Todos los usuarios inscritos en el sistema deben tener un login (correo, usuario y contraseña). Esto permite que se limiten las acciones disponibles para cada usuario, asegurando que solo los profesores puedan crear o modificar Learning Paths y actividades, y que los estudiantes solo puedan inscribirse y completar actividades.

Roles:

Al definir roles se limitan las funciones y/o permisos a los que cada tipo de usuario puede acceder, esto con el fin de tener un control específico sobre las acciones que se pueden realizar. Esta restricción evita conflictos en el uso del sistema, por ejemplo, dado a que el profesor es el único que poder crear y editar Learning Paths y actividades, se busca evitar que un estudiante pueda modificar alguna de estas dos actividades para beneficiarse. Definir estos roles desde un principio evita el uso obligatorio de controles o validaciones más adelante para poder conocer y fijar los roles y las acciones permitidas por cada usuario.

Advertencias sin limitaciones:

El proyecto permite que los estudiantes sigan su propio ritmo mientras realizan las actividades pautadas en los Learning Paths. Normalmente, en un sistema de aprendizaje, no es permitido que los estudiantes avancen a la siguiente actividad si no han terminado las anteriores, esto porque podría afectar la comprensión del contenido. Sin embargo, en este proyecto, el sistema debe darles la opción de hacerlo de todos modos. En lugar de bloquear el acceso a las siguientes actividades, solo se mostrará una advertencia que les explica que no es recomendable saltarse actividades. Al ser esta restricción flexible con los estudiantes, no se crean restricciones adicionales en el proyecto que podrían complicar el acceso a demás actividades.

Para poder demostrar que nuestro diagrama de clases sirve como la base funcional del proyecto, debemos poder modelar correctamente ejemplos. Para nuestro primer caso de prueba, el profesor quiere crear un nuevo Learning Path que se basa en enseñar los principios de derivadas. Para esto, él plantea una serie de actividades para poder asignar. Al empezar a crearlas, se da cuenta que algunas de las actividades ya existen, pero estas fueron creadas e implementadas en Learning Paths diferentes por otros profesores. Si nuestro profesor quiere crear este Learning Path, deberá pedir permiso de uso de estas actividades. Con nuestra UML, se vuelve bastante fácil el proceso para el profesor: las actividades que se quieren crear él las puede crear y añadir a la lista de actividades dentro del learning path directamente, y las actividades que ya existen son simplemente clonadas y luego añadidas al learning path, como se muestra en el gráfico. Digamos que después de un año, el profesor desea Volver a observar su learning path y desea ver qué mejoras se le pueden hacer: para esto, el desea modificar las actividades que son consideradas las peores y simplemente dejar las mejores. Para hacer esto, él debe ser capaz de acceder al feedback dado por los estudiantes en las actividades, y luego de hacer un análisis de este, acceder a las actividades que él desea cambiar. Con nuestra propuesta, el profesor fácilmente puede acceder a esta información. Al ser el creador del learning path y de las actividades (ya sean creadas o clonadas), el profesor puede acceder directamente al feedback dado en cada

actividad. Gracias a que cada actividad tiene un rating, el profesor puede observar y elegir las N actividades que tienen el menor rating. Al ser elegidas, el profesor edita las actividades como él desea, y después de finalizar sus cambios, se guarda la fecha de cambio del learning path y se actualiza la versión de este learning path (este dato es uno que elige el profesor, ya que puede incrementar por decimal la versión o hacer un salto a una versión con un numero entero nuevo). De esta manera, se puede actualizar constantemente los learning paths y asegurar la mayor calidad posible. Además, si existen errores pequeños dentro de los learning paths, el profesor puede fácilmente, y de manera directa, acceder a la actividad con el error y cambiar esta información.

(e1) Ahora, pongámonos en los zapatos de un estudiante. Digamos que un estudiante quiere inscribir un nuevo learning path, pero no sabe qué hacer, entonces decide empezar a investigar. Llega a un learning path sobre biología que le interesa mucho, y por lo tanto decide inscribirlo. Esto reflejado en nuestro modelo se ve como la posibilidad de la clase de usuario (de la cual hereda estudiante) tener la capacidad de acceder directamente a un learning path. Adicionalmente, la clase de estudiante guarda en una lista los learning paths que en el momento el estudiante tiene inscrito. Unos cuantos meses han pasado, y el estudiante ya ha tenido dos quices y su primer examen se aproxima. En este momento, el estudiante desea ver su progreso en la clase y las notas de quices que ha tenido para poder prepararse para su examen apropiadamente. Si el estudiante quiere acceder a esta información, él simplemente debe acceder a su learning path, y acceder al progreso de este. Acá, él podrá ver qué tantas actividades del learning path ha completado gracias a la lista de las actividades completadas. Luego, el podrá acceder (gracias a la lista) a la información de los dos quices que ha hecho, y mirar su nota en estos. Luego de prepararse y aprobar su examen, el estudiante sigue con el progreso de su curso, pero algo terrible pasa: este se enferma y tiene que poner pausa al progreso de su aprendizaje. Ya después de recuperarse, se habilita el tercer quiz del curso, y el chico no se siente preparado para este ya que no ha revisado todo el material y hecho las actividades. Entonces, para casos como este, en nuestra propuesta basta con acceder a la actividad deseada, que en este caso es su tercer quiz, y entrar a la lista de prerrequisitos que él debe cumplir para tomar la actividad. Resulta que él se da cuenta que no ha hecho una tarea, y decide empezar a hacerla inmediatamente. Pero ocurre otro problema, esta tarea requería hacer 2 lecturas previamente, las cuales el chico no hizo. Estas lecturas son opcionales, y hacerlas es bastante recomendado porque facilita la realización de la tarea. Gracias a nuestro diseño, cuando el chico decida empezar a hacer la tarea sin haber revisado los recursos previamente, el sistema le arrojará una advertencia, la cual describe que aún tiene actividades no completadas que se recomiendan completar. Nuestro diseño maneja cada advertencia como una única para cada actividad, ya que las actividades tienen prerrequisitos únicos para cada una. En este momento, el estudiante tiene la posibilidad de elegir que hacer: realizar o no realizar las actividades faltantes.

Ya pasados unos 2 años, el estudiante acaba de terminar su último Learning Path, y desea revisar su nota final, y la fecha en la se registró este. Para esto, nuestro estudiante simplemente debe acceder a la clase de progreso y revisar la nota que aparece en esta, y para revisar la fecha solo accede al Learning Path y ahí podrá obtener esta fecha de inicio. Finalmente, el estudiante desea escribir un último Learning Path, pero ya ha visto muchos, y pues no quiere Volver a inscribir un Learning Path que ya finalizó: para esto, en la clase de estudiante él puede acceder directamente a la lista de los Learning

Paths que ya ha completado, y así, cuando esté mirando qué último Learning Path quiere ver, solo revisa que no esté en la lista. Como se puede apreciar en estos casos de ejemplo, nuestra propuesta tiene soluciones directas para las acciones que los usuarios deseen hacer.

**Diagramas**

**Diagrama de clases completo:**

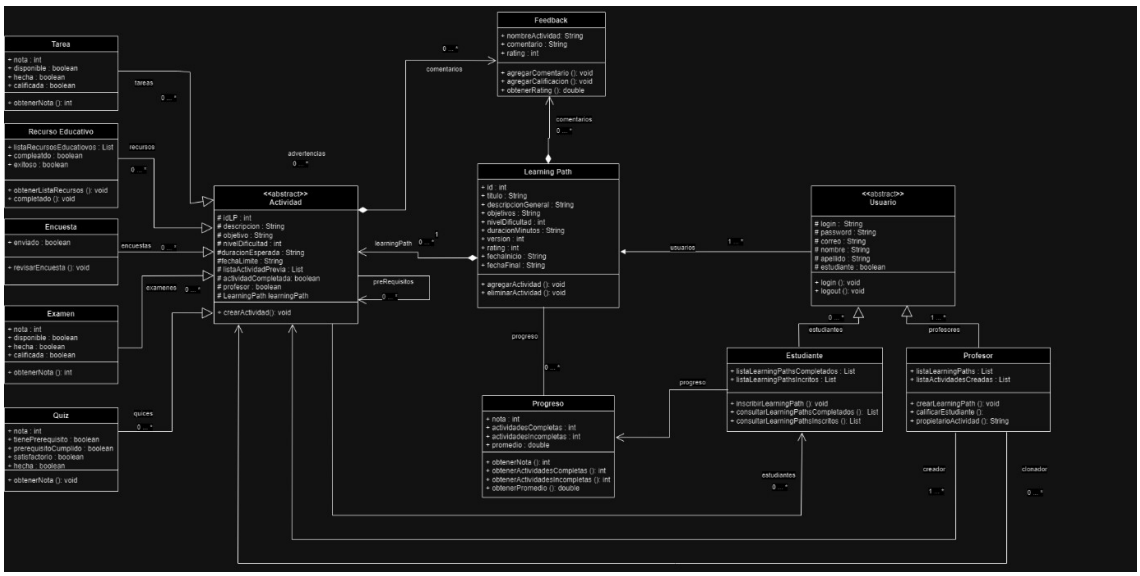
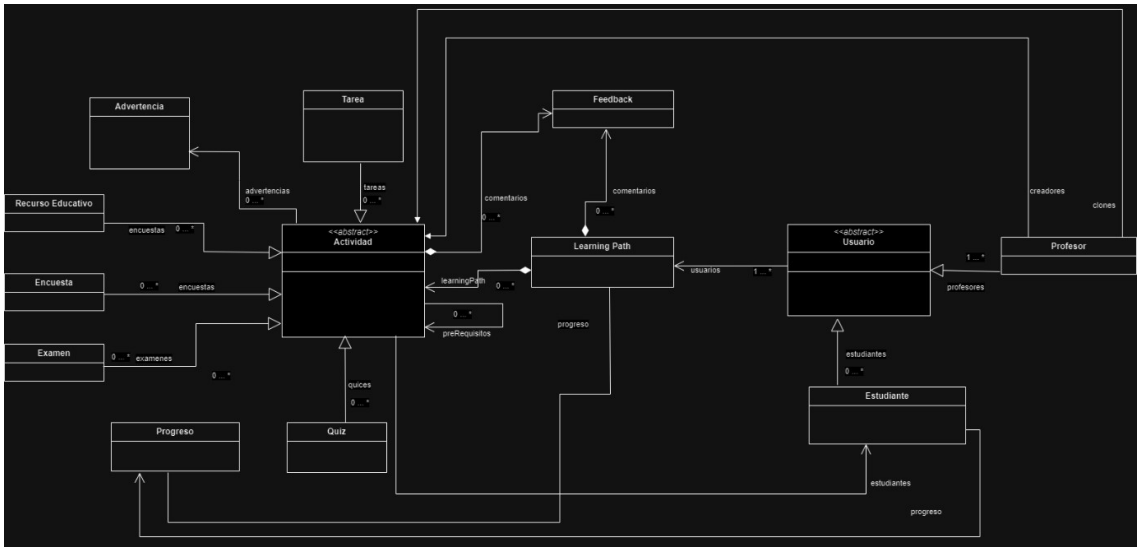
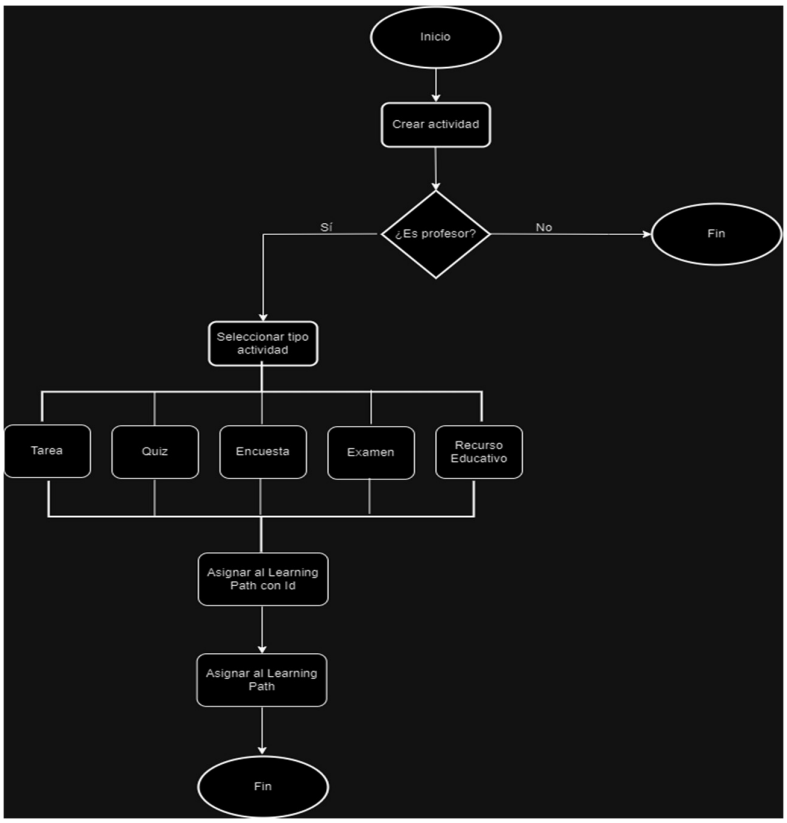



**Diagrama de clases de alto nivel:**



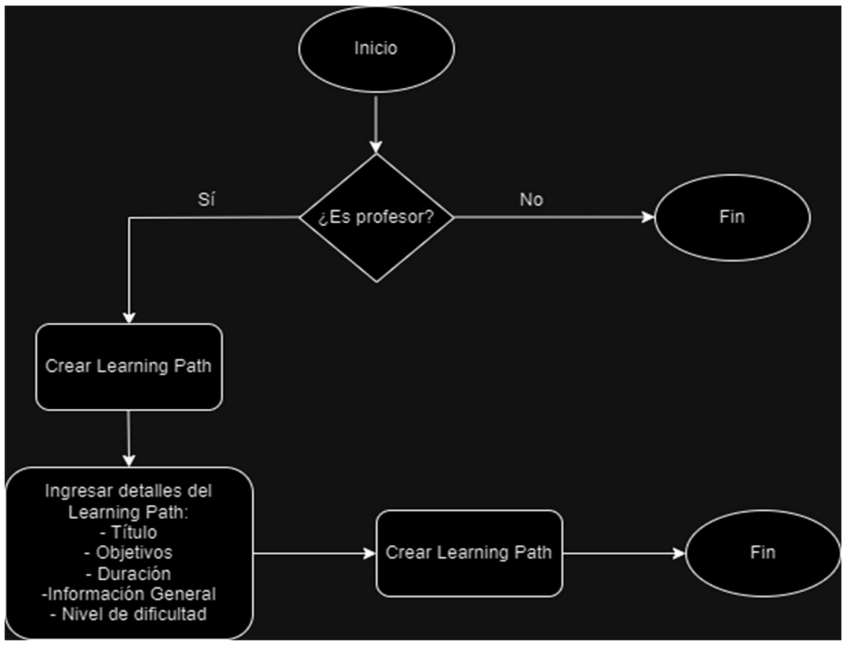

**Diagramas de secuencia:**

- **Diagrama de flujo proceso de creación de una actividad:**



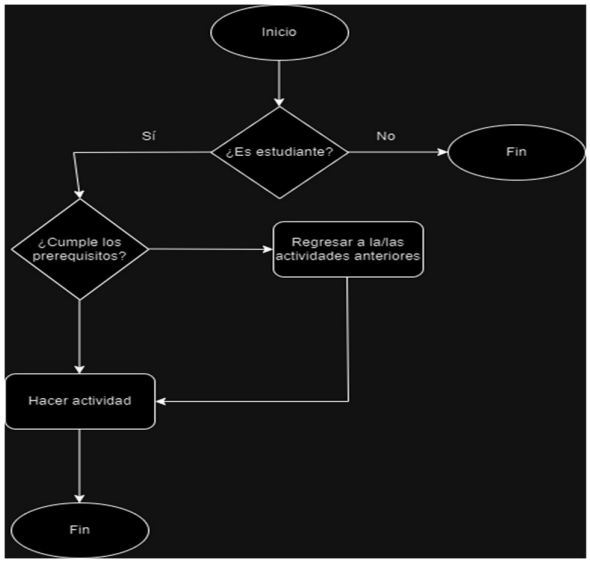

Este diagrama tiene como principal objetivo explicar el proceso de creación de una actividad. La primera decisión que se toma es la de preguntarle al usuario si es profesor o estudiante, en caso de que sea estudiante, se finaliza el proceso debido a que contamos con la restricción de que solo un profesor puede crear una actividad. En caso de que el usuario cumpla con ser un profesor, se sigue el proceso preguntando qué tipo de actividad desea crear, para posteriormente preguntar a qué Learning Path asignará la actividad y finalmente agregarla al Learning Path especificado.

- **Diagrama de flujo proceso de creación de un Learning Path:**



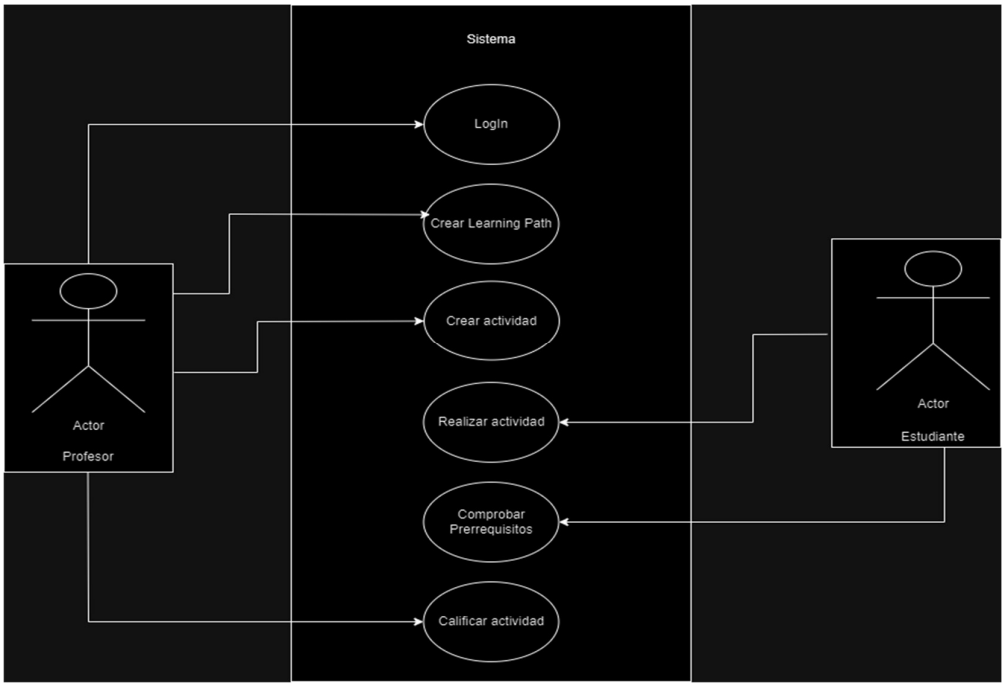

En este diagrama podemos ver de una manera más clara cómo se puede llegar a crear un Learning Path. Nuevamente se inicia el proceso preguntando al usuario si es un estudiante o un profesor. En caso de ser estudiante se finaliza el proceso. En cambio, en caso de que el usuario sea un profesor, se puede continuar con el proceso tomando la decisión de crear el Learning Path, ingresando los detalles necesarios para que el LP tenga toda la información necesaria para que en otro proceso el estudiante decida a qué LP inscribirse basado en esa información.

- **Diagrama de flujo proceso de realización de una actividad:**



En este diagrama se expresa el proceso de realización de una actividad. Al igual que en los dos diagramas anteriores, se inicializa el proceso verificando el tipo de usuario que quiere realizar una actividad. En este caso si el usuario es profesor, se finaliza el proceso. Pero si el usuario es un estudiante, se procede a verificar si cumple con los prerrequisitos necesarios para realizar la actividad, en caso de que no cumpla con estos, se le pedirá que realice estas actividades previas. Para finalizar con la realización de la actividad.

**Diagrama de uso:**




En este diagrama se especifican las funcionalidades principales que tiene el programa y los actores que intervienen en cada una de estas.

**Diagrama de paquetes:**