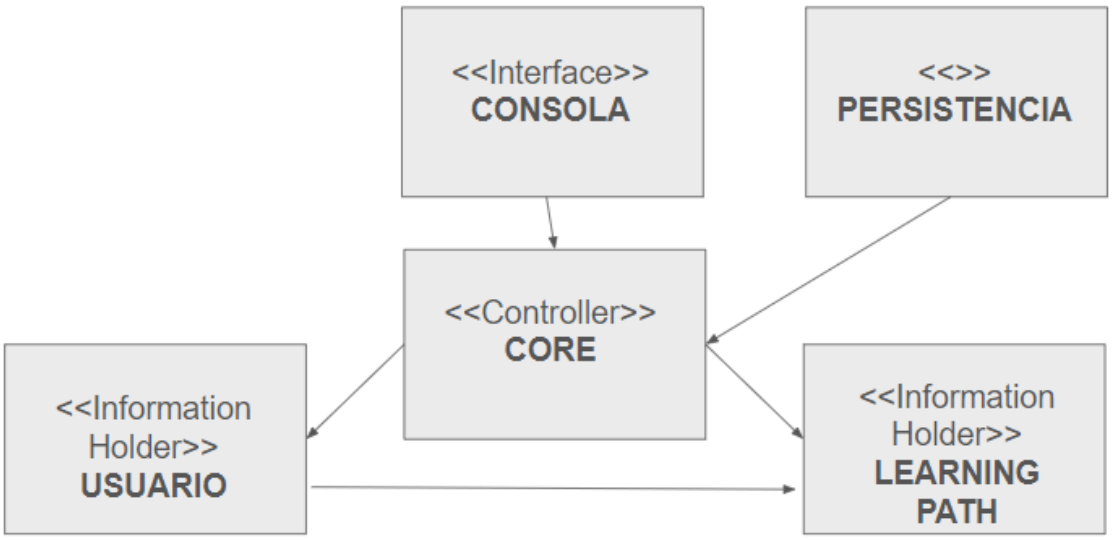
Para poder atacar el enunciado correctamente decidimos primero que todo dividir nuestro programa con los siguientes paquetes o divisiones.

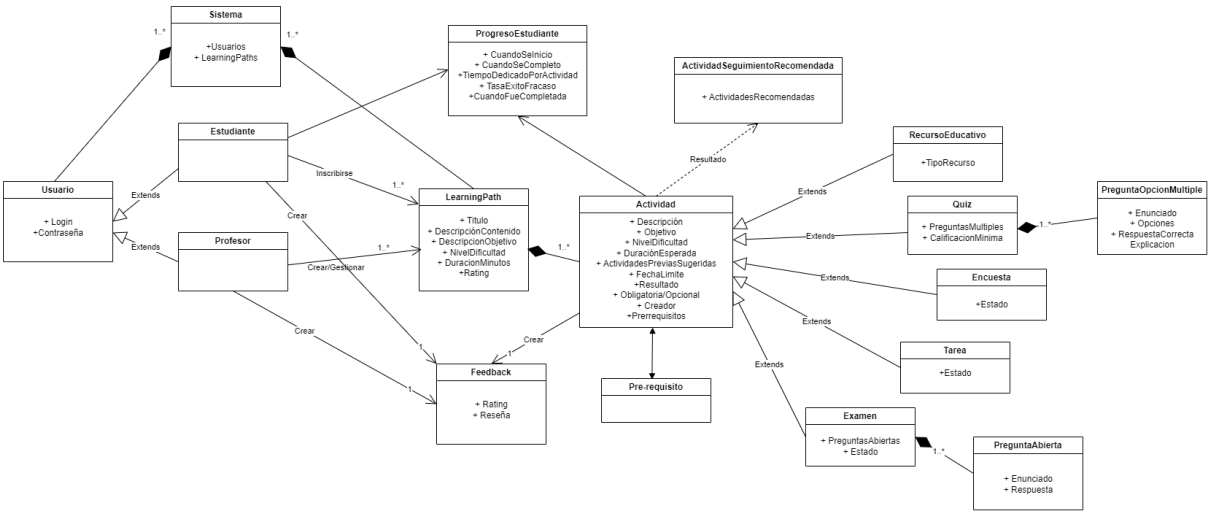


Definimos que dos componentes importantes que permiten que un Learning Path Recommendation System funcione y exista son los usuarios y los propios learning paths. Sin estos la interacción que se busca no sería posible. Un learning path necesita un usuario tanto que lo cree como uno que lo complete, y un usuario necesita un learning path para aprender o enseñar. Es por esto que estos dos paquetes siguen el estereotipo de Information Holder, ya que serán los que almacenen todos los datos, todas las interacciones, creaciones, etc.

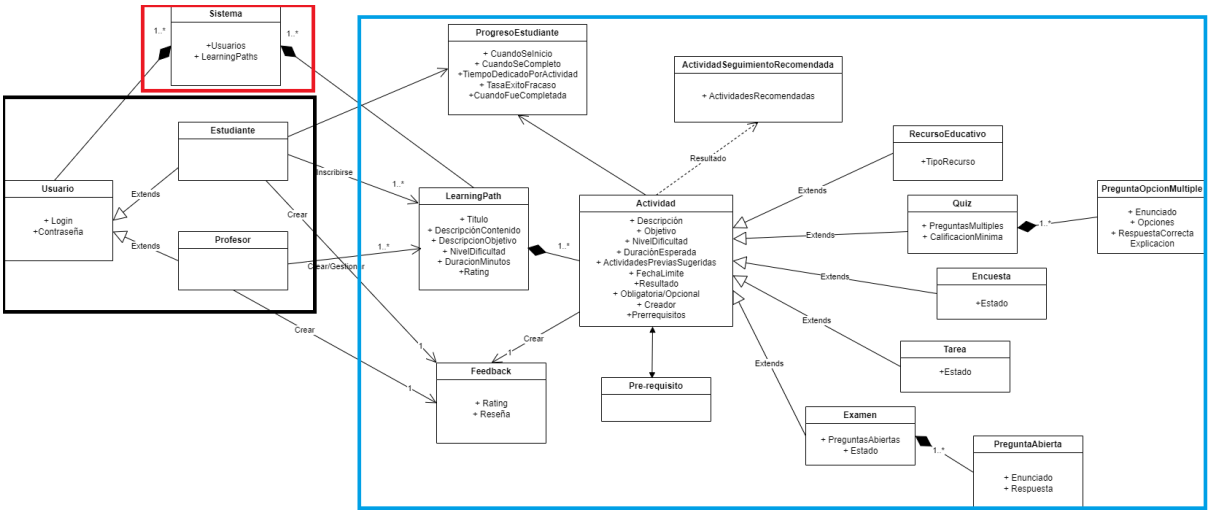
Es por esto que estos dos componen el core, el cual es un controlador, que toma decisiones importantes y controla las acciones de otros componentes, ayudando a llamar a acción los métodos dentro de los paquetes, para por ejemplo buscar los learning paths, ayudar a completar actividades, calificarlas, etc.

Además se incluye una consola, que servirá como interfaz entre el usuario y los learning paths, por ejemplo recolectando las respuestas del estudiante al completar una actividad, y en resumen para transformar información y peticiones entre diferentes partes de un sistema. Y por último está la persistencia que busca que nuestro programa no colapse ante un posible fallo, debido a algo interno, como una actividad que no existe, o externo, como un input no valido.

Al inicio habíamos definido esta estructura básica, ya definiendo las clases dentro de cada paquete, sus relaciones y algunos atributos.



Este es el planteamiento inicial, y el siguiente sería ese mismo pero con la definición de que pertenece a cada elemento de una manera más clara:



Con esto definimos que las relaciones que predominan entre las clases serán de composición y herencia. La clase de usuarios partiría de una clase padre denominada “Usuario” y que tendría dos clases hijas, “Estudiante” y “Profesor”, ya que estos son los únicos dos usuarios que son descritos en el planteamiento de problema y que interactúan con la plataforma.

El paquete de Learning Path parte de que este se compone de muchas actividades, y que estas actividades pueden tener varios tipos. Es por esto que “Actividad” actúa como una clase padre para que “RecursoEducativo”, “Quiz”, “Encuesta”, “Tarea”, “Examen”, que son los tipos de actividades que existen, hereden esos atributos. Al mismo tiempo un “Examen” se compone de preguntas abiertas es por esto que también creamos la clase “PreguntaAbierta”. Y por la misma razón pero para un “Quiz” creamos la clase “PreguntaOpcionMultiple”. Más adelante nos dimos cuenta de que la clase “Encuesta” al igual que “Examen” se compone de preguntas abiertas, es por esto que también debería haber una relación de composición entre esta y la clase.

También inicialmente planteamos que una actividad conforma, o es conformada por pre-requisitos y actividades de seguimiento recomendadas. Pero a medida de que el

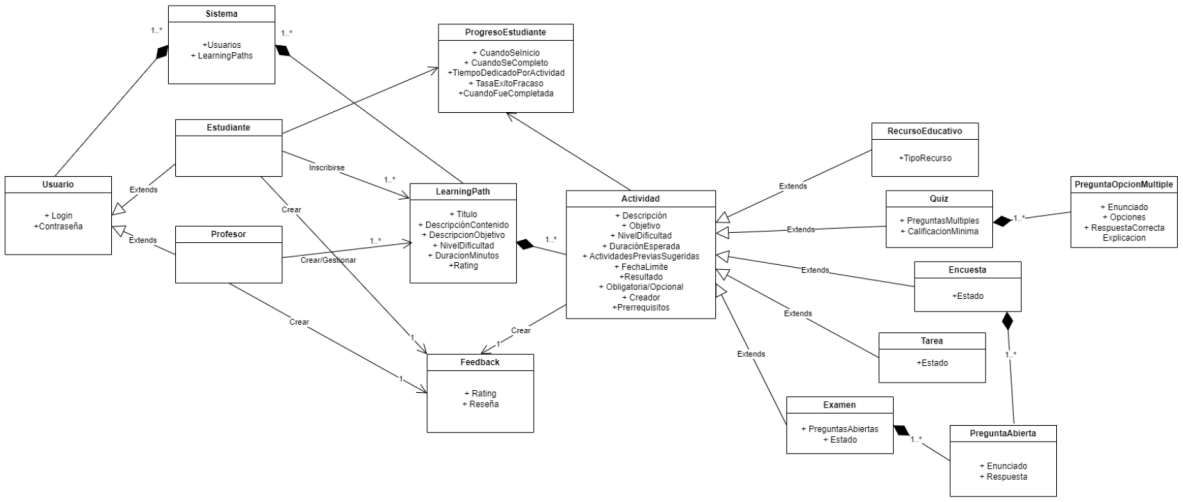
programa se fue desarrollando el programa nos dimos cuenta de que era más sencillo plantearlo como una estructura o atributo propio de la actividad, por lo tanto este cambio será reflejado y explicado más adelante.

Una actividad también tendría que tener feedback, y por lo tanto creamos una clase que contenga todos los atributos que se requieren para crear una, y que después esta se almacenen en la propia clase de actividad a la cual corresponde.

Y por último para el paquete de Learning Path también se creó otra clase llamada “Progreso Estudiante” para almacenar todas las respuestas y básicamente lo que va haciendo y completando el estudiante. Y es por esto que se puede ver que esta se relaciona con una actividad y un estudiante.

El último paquete es el sistema que como dijimos se compone de todos los learning Paths y usuarios existentes.

Teniendo en cuenta estos pequeños ajustes, este sería el diagrama de dominio corregido:



Con esto en cuenta se comenzó a definir bien los atributos de cada clase y las estructuras que se usarán para almacenar los datos.

Estos son todos los atributos de las clases que componen el paquete de Usuario:

- Usuario:
    - **login** (`String`): Cadena de texto que representa el nombre de usuario o el identificador que utiliza el usuario para iniciar sesión en el sistema.
    - **contraseña** (`String`): Cadena de texto que almacena la contraseña del usuario, utilizada para la autenticación.
    - **id** (`int`): Identificador único para cada usuario en el sistema, lo que permite distinguir entre diferentes usuarios.

Las siguientes son clases hijas y por lo tanto heredan los atributos de la clase padre “Usuario”:

- Estudiante:

- **actividadesCompletadas** (`List<Actividad>`): Lista que contiene las actividades que el estudiante ha completado. Permite llevar un registro del progreso del estudiante.
- **learningPathsCompletados** (`List<LearningPath>`): Lista que almacena los caminos de aprendizaje que el estudiante ha finalizado. Esto ayuda a evaluar el recorrido educativo del estudiante.
- **learningPathsEnCurso** (`List<LearningPath>`): Lista que contiene los caminos de aprendizaje que el estudiante está actualmente siguiendo. Permite monitorear la participación activa del estudiante en su educación.
- **actividadesEnCurso** (`List<Actividad>`): Lista que almacena las actividades en las que el estudiante está actualmente trabajando. Facilita el seguimiento de las tareas y recursos educativos pendientes.

- Profesor:
  - **learningPaths** (`List<LearningPath>`): Esta lista almacena los caminos de aprendizaje que el profesor ha creado o gestionado. Permite al profesor organizar y supervisar las actividades educativas que ofrece a los estudiantes.

Estos son todos los atributos de las clases que componen el paquete de LearningPath:

- LearningPath:
  - **id** (`int`): Identificador único del LearningPath. Esto es necesario ya que se puede dar el caso de que dos tengan el mismo nombre y por lo tanto es necesario diferenciarlos el uno del otro.
  - **titulo** (`String`): Título de la actividad o recurso.
  - **descripcionContenido** (`String`): Descripción del contenido del recurso o actividad.
  - **descripcionObjetivo** (`String`): Descripción del objetivo o propósito de la actividad.
  - **nivelDificultad** (`String`): Nivel de dificultad asociado con la actividad (por ejemplo, "Fácil", "Intermedio", "Difícil").
  - **duracionMinutos** (`String`): Tiempo estimado en minutos para completar la actividad.
  - **listaActividades** (`List<Actividad>`): Lista de las actividades que componen ese LearningPath. Y se decidio que fuera una lista, ya que es la estructura más simple para almacenarlas y no es necesario relacionar cada actividad con otro atributo, como para usar un HashMap.
  - **progresoEstudiante** (`HashMap<Estudiante, ProgresoEstudiante>`): Mapa que asocia a cada estudiante con su progreso de ese learningPath. En este caso si se decidio usar un HashMap ya que el progreso, que se compone en las respuestas por cada actividad, si depende del estudiante que la resuelve, y es necesario que este vinculado a este, para que despues por ejemplo un profesor pueda ver esa avance. Y el atributo “ProgresoEstudiante” proviene directamente de la clase “ProgresoEstudiante” de la cual se hablo anteriormente.
- Actividad:

- **id** (`int`): Identificador único de la actividad.
- **descripcion** (`String`): Descripción general de la actividad.
- **objetivo** (`String`): Propósito o meta que se espera alcanzar con la actividad.
- **nivelDificultad** (`String`): Nivel de dificultad asignado a la actividad (por ejemplo, "Fácil", "Medio", "Difícil").
- **duracionEsperada** (`String`): Tiempo estimado para completar la actividad.
- **actividadesPreviasSugeridas** (`List<Actividad>`): Lista de actividades recomendadas para realizar antes de esta.
- **actividadesSeguimientoRecomendadas** (`List<Actividad>`): Lista de actividades recomendadas para realizar después de esta.
- **fechaLimite** (`String`): Fecha límite para completar la actividad.
- **resultado** (`String`): Estado o resultado de la actividad, puede ser "exitosa", "fallida", etc.
- **obligatoria** (`boolean`): Indica si la actividad es obligatoria o no.
- **idCreador** (`int`): Identificador del creador de la actividad, el cual corresponde a un profesor. Y esto es necesario para mantener un registro y despues poder definir si un profesor puede editar dircetamente una actividad, que tiene que ser de su propiedad, o si tiene que hacerle un copia para despues editarla.
- **prerequisitos** (`List<Actividad>`): Lista de actividades que deben completarse antes de realizar esta. Con la misma logica con la que se usa una lista de actividades que componen un learningPath, se usa una lista para guardar las actividades prerrequisitos de esta.
- **tituloActividad** (`String`): Título o nombre de la actividad.
- **feedbacks** (`List<Feedback>`): Lista de comentarios o retroalimentación proporcionada por los estudiantes o profesores sobre la actividad. Ya que una actividad puede tener varios feedbacks o reseñas estas se tienen que guardar en una lista, asociandolas propiamente a la actividad a las que hacen referencia. Y este atributo de Feedback hace referencia a la clase feedback antes descrita.

Las siguientes son clases hijas y por lo tanto heredan todos los atributos de una actividad pero también tiene unos atributos únicos.

- RecursoEducativo:
  - **tipoRecurso** (String): Tipo de recurso educativo (por ejemplo, video, libro, página web, etc.).
  - **recurso** (String): Descripción o enlace al recurso específico que se está proporcionando.
  - **estado** (String): Estado del recurso educativo, que puede ser "no terminada", "completada", u otros estados que representen el progreso o resultado de la actividad.
- Tarea:
  - **estado** (String): Estado de la tarea, que puede indicar si la tarea está "pendiente", "enviada", "calificada", etc.

    - **medioEntrega** (String): Medio de entrega de la tarea, como "correo", "LMS", u otro método específico de entrega.
- Examen:
    - **preguntasAbiertas** (List<PreguntaAbierta>): Una lista de preguntas abiertas que forman parte del examen.
    - **estado** (String): Estado del examen, que puede indicar si el examen está "en progreso", "finalizado", "calificado", etc.
    - **respuestasEstudiante** (HashMap<PreguntaAbierta, String>): Un mapa que asocia cada pregunta abierta con la respuesta proporcionada por el estudiante. Esto es necesario ya que se necesita tener un registro de lo que un estudiante puede llegar a responder, para completar una actividad.

      Tanto en preguntasAbiertas como en respuestasEstudiantes se usa el atributo PreguntaAbierta el cual viene de la clase “PreguntaAbierta” y tiene los siguientes atributos:
        - **enunciado** (String): el enunciado de la pregunta abierta, que presenta el problema o cuestión que el estudiante debe responder.
        - **respuesta** (String): La respuesta esperada o correcta para la pregunta abierta, que puede ser utilizada para la evaluación o referencia.

- Encuesta: Esta sigue la misma logica que examen, con respecto a que se compone de preguntas abiertas.
    - **preguntasAbiertas** (List<PreguntaAbierta>): Lista de preguntas abiertas que forman parte del examen o actividad.
    - **estado** (String): Indica el estado actual de la actividad, en este caso, inicializado como "no enviada".
    - **respuestasEstudiante** (HashMap<PreguntaAbierta, String>): Mapa que asocia cada pregunta abierta con la respuesta dada por el estudiante.
- Quiz:
    - **preguntasMultiples** (List<PreguntaOpcionMultiple>): Lista de preguntas de opción múltiple que forman parte del quiz.
    - **calificacionMinima** (int): Calificación mínima requerida para aprobar el quiz.
    - **pregunta** (PreguntaOpcionMultiple): Una instancia de una pregunta de opción múltiple (aunque parece que debería ser más relevante como una lista, dependiendo del contexto).
    - **respuestasEstudiante** (HashMap<PreguntaOpcionMultiple, String>): Mapa que asocia cada pregunta de opción múltiple con la respuesta dada por el estudiante.

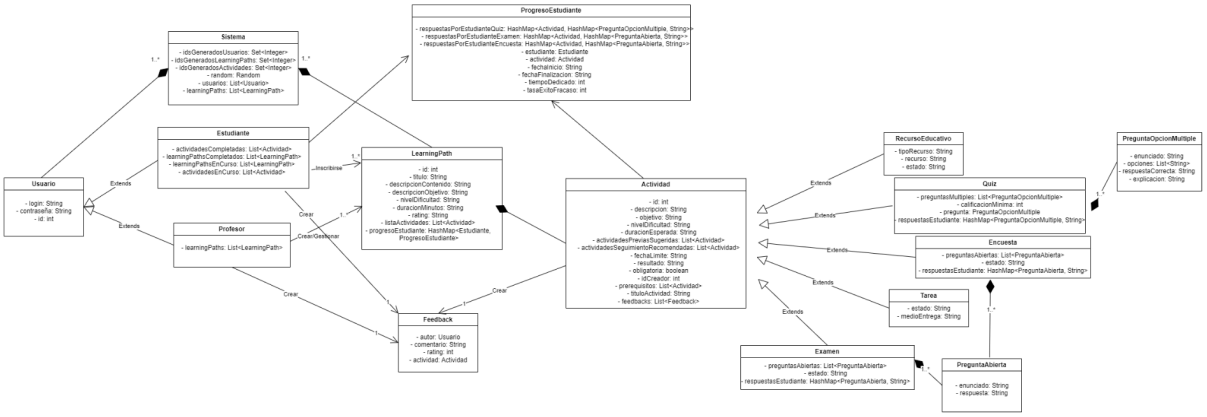
      Tanto en preguntasMultiple como en respuestasEstudiantes se usa el atributo PreguntaOpcionMultiple el cual viene de la clase “PreguntaOpcionMultiple” y tiene los siguientes atributos:
        - **enunciado** (String): el texto que presenta la pregunta al estudiante.
        - **opciones** (List<String>): Lista de las opciones disponibles para que el estudiante elija su respuesta.
        - **respuestaCorrecta** (String): La opción que es correcta entre las opciones proporcionadas.

        - **explicacion** (String): Una explicación adicional que puede ofrecer contexto o justificación sobre la respuesta correcta.
- ProgresoEstudiante:
    - **respuestasPorEstudianteQuiz** (HashMap<Actividad, HashMap<PreguntaOpcionMultiple, String>>): Mapa que almacena las respuestas proporcionadas por el estudiante para los quizes, organizadas por actividad. Y esta al igual que las siguientes dos se asociacion a una actividad porque es necesario que se sepa bien a que corresponden esas respuestas.
    - **respuestasPorEstudianteExamen** (HashMap<Actividad, HashMap<PreguntaAbierta, String>>): Mapa que almacena las respuestas proporcionadas por el estudiante para los exámenes, organizadas por actividad.
    - **respuestasPorEstudianteEncuesta** (HashMap<Actividad, HashMap<PreguntaAbierta, String>>): Mapa que almacena las respuestas proporcionadas por el estudiante para las encuestas, organizadas por actividad.
    - **estudiante** (Estudiante): Objeto que representa al estudiante cuyo progreso se está registrando.
    - **actividad** (Actividad): Objeto que representa la actividad en la que el estudiante está participando.
    - **fechaInicio** (String): La fecha en la que el estudiante comenzó la actividad.
    - **fechaFinalizacion** (String): La fecha en la que el estudiante finalizó la actividad.
    - **tiempoDedicado** (int): La cantidad de tiempo (en minutos o segundos, dependiendo de la implementación) que el estudiante dedicó a completar la actividad.
    - **tasaExitoFracaso** (int): Un valor que representa la tasa de éxito o fracaso del estudiante en la actividad, que puede ser calculada como un porcentaje.
- Feedback:
    - **autor** (Usuario): Objeto que representa al usuario que creó el feedback. Este usuario podría ser un estudiante, un profesor, o cualquier otro tipo de usuario del sistema.
    - **comentario** (String): Cadena de texto que contiene el comentario del autor sobre la actividad.
    - **rating** (int): Valor entero que representa la calificación o puntuación otorgada al feedback, que podría variar en un rango específico (por ejemplo, de 1 a 5).
    - **actividad** (Actividad): Objeto que representa la actividad a la que se refiere el feedback. Esto permite asociar el comentario y la calificación a una actividad específica.

- Core:
    - **idsGeneradosUsuarios** (Set<Integer>): Conjunto que almacena los identificadores generados para los usuarios del sistema, garantizando que no haya duplicados.
    - **idsGeneradosLearningPaths** (Set<Integer>): Conjunto que almacena los identificadores generados para los caminos de aprendizaje, asegurando que cada uno sea único.

- **idsGeneradosActividades** (Set<Integer>): Conjunto que almacena los identificadores generados para las actividades en el sistema, evitando duplicados.
- **usuarios** (List<Usuario>): Lista que contiene todos los usuarios registrados en el sistema, lo que permite gestionar y acceder a la información de cada uno de ellos.
- **learningPaths** (List<LearningPath>): Lista que almacena todos los caminos de aprendizaje disponibles en el sistema, facilitando la gestión de los mismos.

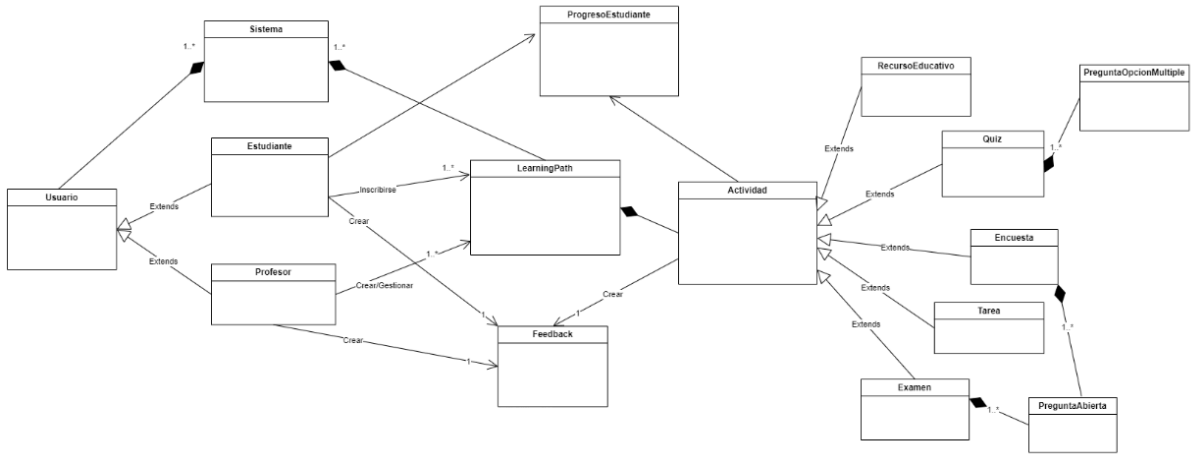
Este sería el diagrama con sus respectivos atributos:



También al establecer estos atributos clave para cada clase, se definió las principales funcionalidades de nuestro programa, que componen las funcionalidades críticas:

1. Crear Learning Path
2. Editar Learning Path
3. Crear Nueva Actividad
4. Editar Actividad
5. Calificar Actividad
6. Crear Reseña
7. Añadir Prerrequisitos a una Actividad
8. Añadir Actividades de Seguimiento a una Actividad
9. Añadir Actividades Previas a una Actividad
10. Ver Reseñas de una Actividad
11. Ver Información de una Actividad
12. Mostrar Información de un Learning Path
13. Mostrar Respuestas de un Estudiante
14. Inscribirse a un Learning Path
15. Iniciar una Actividad
16. Ver Progreso Learning Path

Y ahora con respecto a estos vamos a diagramar la secuencia de estos para poder identificar los métodos necesarios para cada uno. Y para este proceso vamos a usar este diagrama UML que solo se compone de las clases, para que sea más facil seguir los “movimientos” del sistema.



Secuencia Crear Learning Path:

1. crearLearningPath(Scanner scanner)
2. new LearningPath(id, titulo, descripcionContenido, descripcionObjetivo, nivelDificultad, duracionMinutos, rating)
3. sistema.agregarLearningPath(nuevoLearningPath)
4. agregarLearningPath(LearningPath learningPath)

Secuencia editar learningPath:

1. editarLearningPath(Scanner scanner)
2. Profesor.editarLearningPath(learningPathEncontrado, nuevoTitulo, nuevaDescripcionContenido, nuevaDescripcionObjetivo, nuevoNivelDificultad, nuevaDuracionMinutos, nuevoRating)

Secuencia crear actividad:

1. crearActividad(Scanner scanner)
2. Profesor.nuevaActividad(tipoActividad, id, tituloActividad, descripcion, objetivo, duracionEsperada, obligatoria, profesorEncontrado.getId(), nivelDificultad, actividadesPreviasSugeridas, fechaLimite, prerequisitos, actividadesSeguimientoRecomendadas, scanner)
3. sistema.buscarLearningPath(Integer.parseInt(idLP)
4. learningPathEncontrado.agregarActividad(nuevaActividad)

Secuencia editar Actividad:

1. editarActividad(Scanner scanner)
2. sistema.buscarActividadPorId(Integer.parseInt(idActividadParaEditar))
3. Profesor.editarActividad(actividadEncontrada, Integer.parseInt(id), nuevaDescripcion, nuevoObjetivo, nuevoNivelDificultad, nuevaDuracionEsperada, actividadesPreviasSugeridas, nuevaFechaLimite, nuevaObligatoria, prerequisitos)

Calificar actividad:

1. editarResultadoActividad(Scanner scanner)
2. sistema.buscarActividadPorId(Integer.parseInt(id))

3. sistema.buscarLearningPathPorActividad(actividadEncontrada)
4. learningPath.obtenerProgresoDeEstudiante(estudianteEncontrado)
5. progreso.cambiarResultadoActividad(actividadEncontrada, nuevoResultado)

Secuencia ver respuesta estudiante:

1. verResultadoEstudiante(Scanner scanner)
2. sistema.buscarActividadPorId(Integer.parseInt(id))
3. sistema.stringAEstudiante((Integer.parseInt(idUsuario)))
4. sistema.buscarLearningPathPorActividad(actividadEncontrada)
5. estudianteEncontrado.mostrarRespuestasEstudiantes(actividadEncontrada, learningPath)

Secuencia Inscribirse LearningPath:

1. inscribirselearningPath(Scanner scanner)
2. sistema.buscarLearningPath(Integer.parseInt(id))
3. estudianteEncontrado.registrarselearningPath(learningPathEncontrado)

Secuencia iniciar actividad:

1. iniciarActividad(Scanner scanner)
2. sistema.buscarActividadPorId(Integer.parseInt(id))
3. estudianteEncontrado.iniciarActividad(actividadEncontrada)
4. sistema.realizarActividadEstudiante(actividadEncontrada, estudianteEncontrado, scanner)

Secuencia ver progreso estudiante:

1. verProgresoEstudiante(Scanner scanner)
2. sistema.buscarLearningPath(Integer.parseInt(id))
3. estudianteEncontrado.establecerProgresoEstudiante(learningPathEncontrado)

Secuencia crear feedback(reseña):

1. crearFeedbackEstudiante(Scanner scanner)
2. sistema.buscarActividadPorId(Integer.parseInt(id))
3. Usuario.crearFeedbackEstudiante(Usuario,actividadEncontrada, Integer.parseInt(rating), comentarioFeedback)

Secuencia añadir Actividades Seguimiento Recomendadas:

1. añadirActividadesSeguimientoRecomendadas(Scanner scanner)
2. sistema.buscarActividadPorId(Integer.parseInt(idActividadParaAñadir))
3. actividadEncontrada.agregarActividadesSeguimientoRecomendadas(actividadEncontradaParaAñadir)

Secuencia añadir Actividades Previas:

1. añadirActividadPrevia(Scanner scanner)
2. sistema.buscarActividadPorId(Integer.parseInt(idActividadParaAñadir))
3. actividadEncontrada.agregarActividadPrevia(actividadEncontradaParaAñadir)

Secuencia añadir Actividades Seguimiento Recomendadas:

1. añadirPrerequisito(Scanner scanner)
2. sistema.buscarActividadPorId(Integer.parseInt(idActividadParaAñadir))
3. actividadEncontrada.agregarPrerequisito(actividadEncontradaParaAñadir)

Secuencia ver reseñas actividades:

1. verReseñasActividades(Scanner scanner)
2. sistema.buscarActividadPorId(Integer.parseInt(id))
3. actividadEncontrada.mostrarFeedbacks()

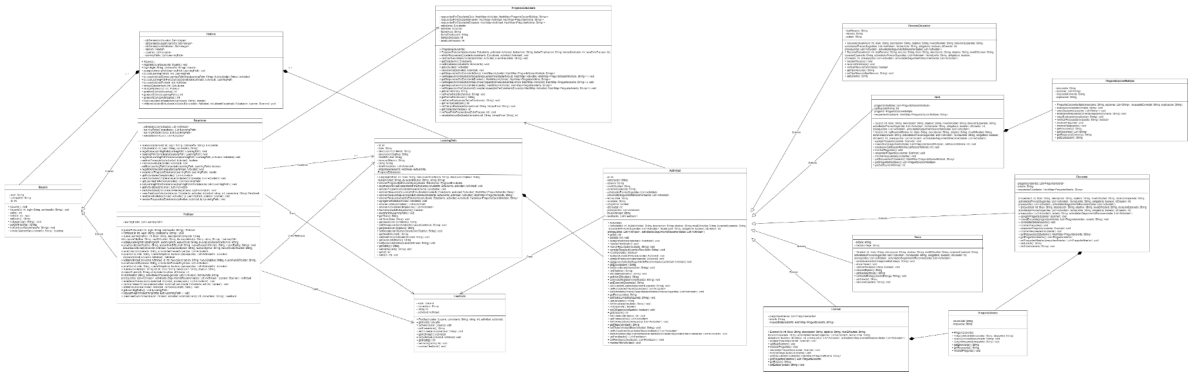
Secuencia ver información Actividad:

1. mostrarInfoActividad(Scanner scanner)
2. sistema.buscarActividadPorId(Integer.parseInt(id))
3. actividadEncontrada.mostrarInfoActividad()

Secuencia ver informacion LearningPath:

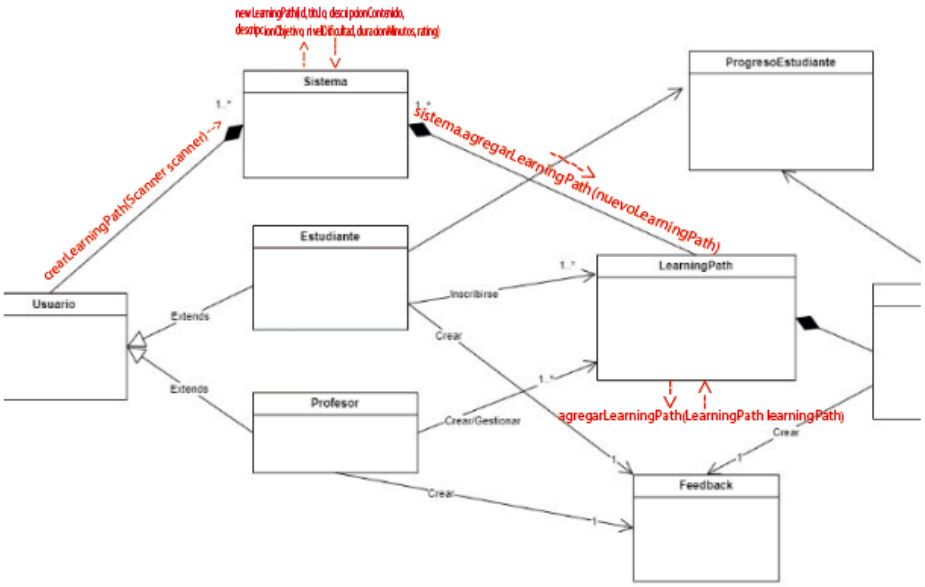
1. mostrarInfoLearningPath(Scanner scanner)
2. sistema.buscarLearningPath(Integer.parseInt(id))
3. LPEncontrado.mostrarInfoLearningPath()

De esta forma logramos ir complementando el diagrama con todos sus métodos, atributos y relaciones:



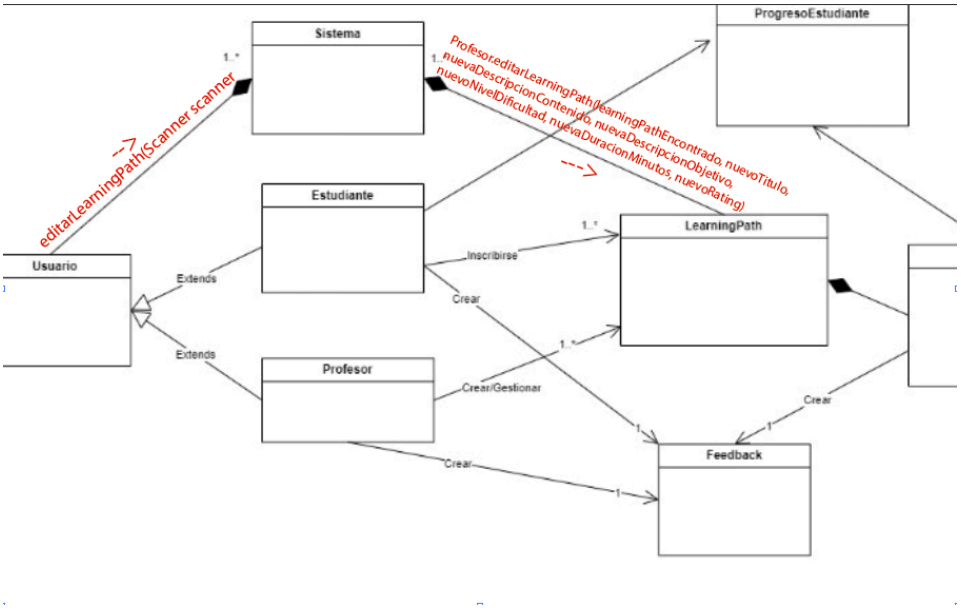
(La imagen completa y con buena resolución está en la carpeta de Entrega2, igual que el resto de diagramas UML).

Diagrama Crear Learning Path:



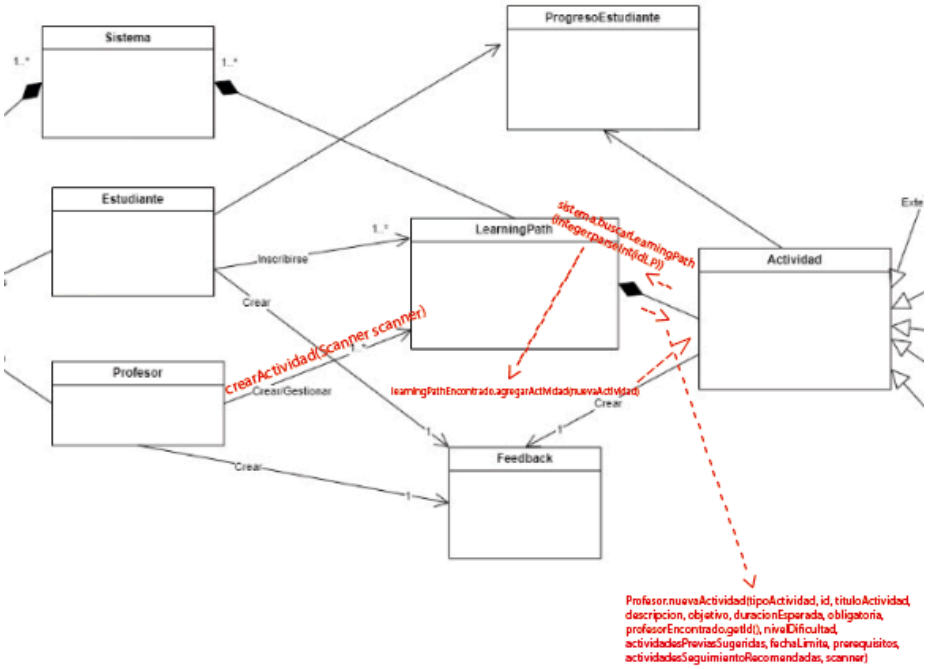
El usuario crea un nuevo LearningPath, e ingresa todos los datos necesarios. En sistema se utilizan estos datos para crear el nuevo LearningPath. Sistema envía el Learning Path creado, para que LearningPath lo agregue (Information Holder)

Diagrama Editar LearningPath:



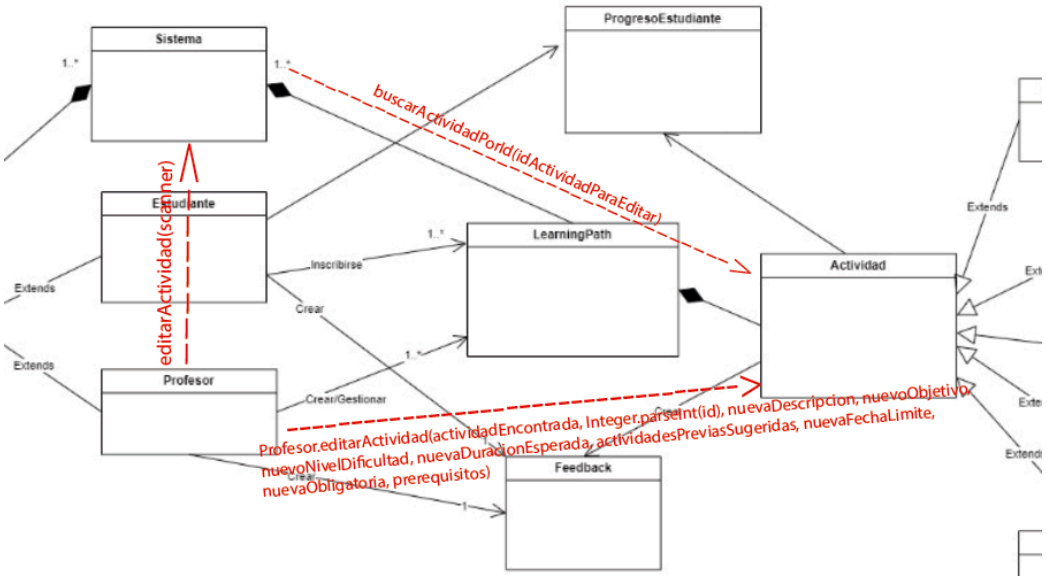
Inicia desde que el usuario desea editar algún LearningPath. Sistema guarda la información proporcionada por el usuario y la envía al info holder (LearningPath). Ahí, se actualizan los datos del LearningPath.

Diagrama Crear Actividad:



Se inicia cuando se crea una actividad por parte del profesor. Llega al Information Holder (LearningPath), para posteriormente pasar a Actividad (profesor.nuevaActividad….). Luego, regresa al Information Holder (LearningPath), quien guarda la actividad con todos sus contenidos, y finalmente vuelve a Actividad, para añadirla en su lista, bajo su correspondiente LearningPath.

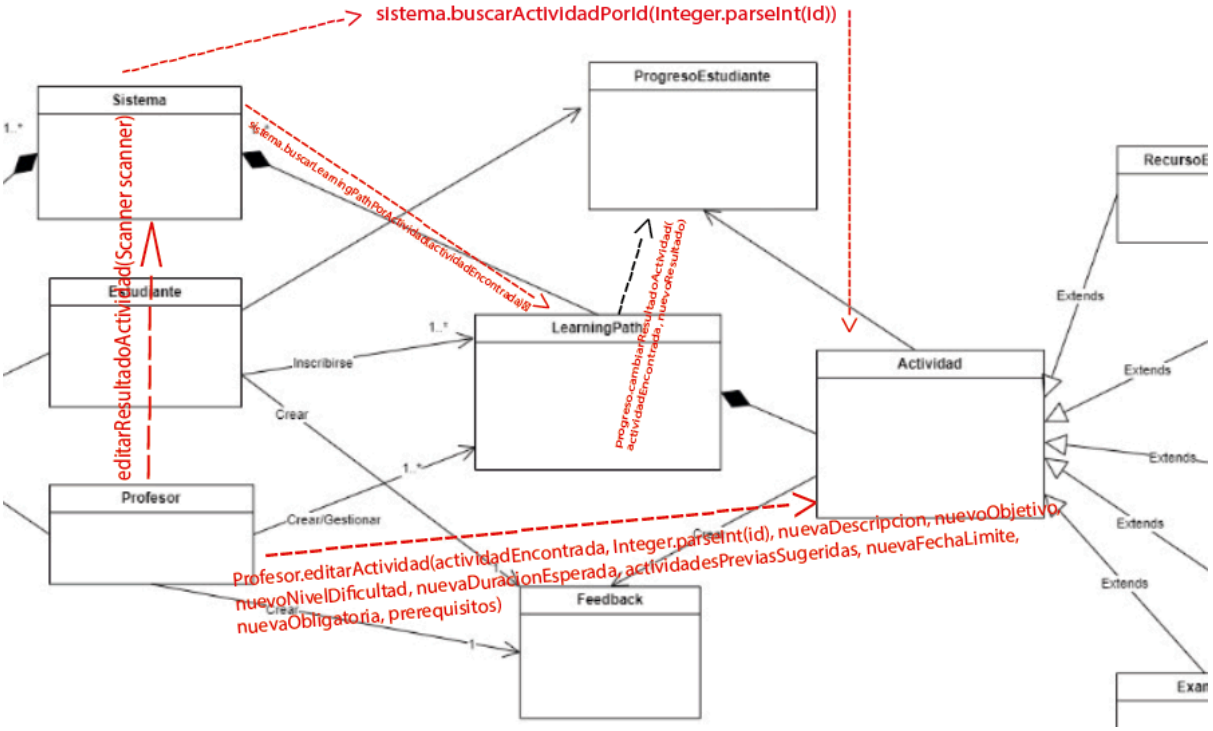
Editar Actividad Diagrama:



El profesor, al editar una actividad, llama al sistema, para que este encuentre la actividad con el correspondiente ID en la lista de actividades. Posteriormente, cuando la actividad ha

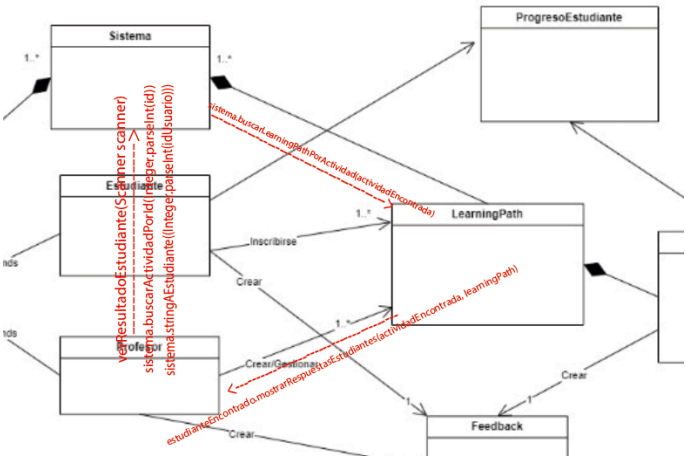
sido encontrada, le agrega toda la información proporcionada por el profesor, actualizando sus datos.

Calificar Actividad Diagrama:



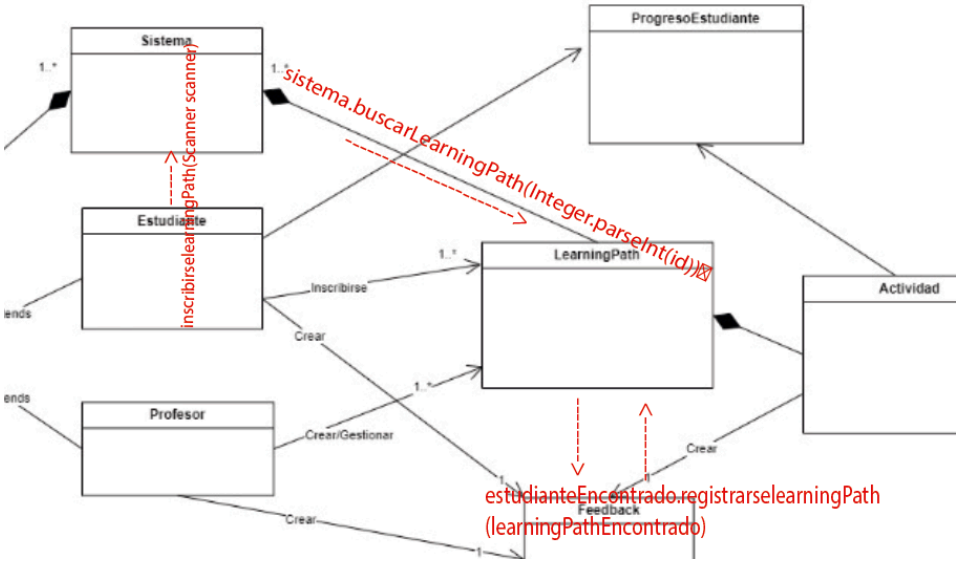
El profesor inicia el proceso de calificar la actividad por buscarla. Para ellos, el sistema (structurer) busca el correspondiente LearningPath de la actividad. Luego, busca la actividad por su ID y devuelve la actividad encontrada. Luego, actualiza el progreso del estudiante añadiendo la nota de la actividad. .

Diagrama Ver Respuesta Estudiante:



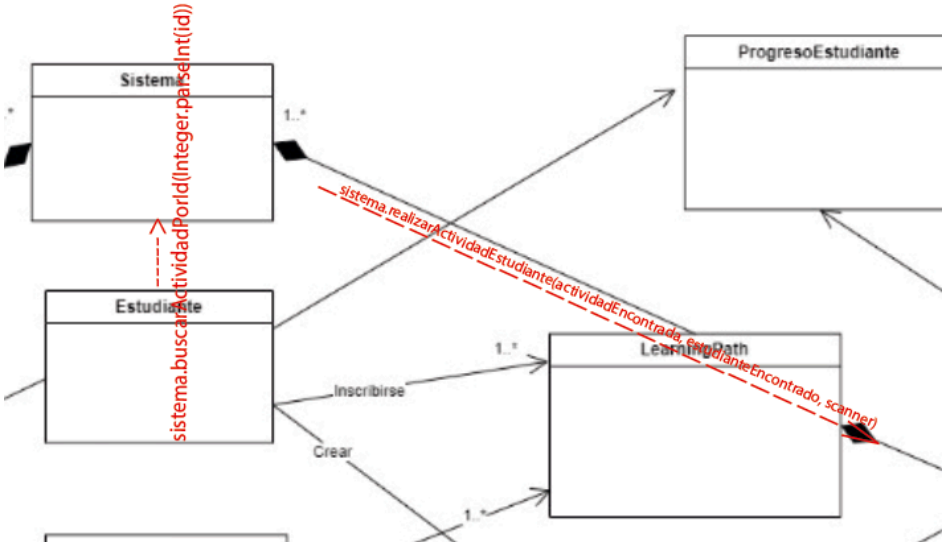
El profesor busca tanto el ID del estudiante como de la actividad. Para ello va a sistema (structurer) y este la busca en el learning path. Una vez se encuentra el estudiante y la actividad se vuelve a el profesor con la respuesta del estudiante.

Diagrama Inscribirse a LearningPath:



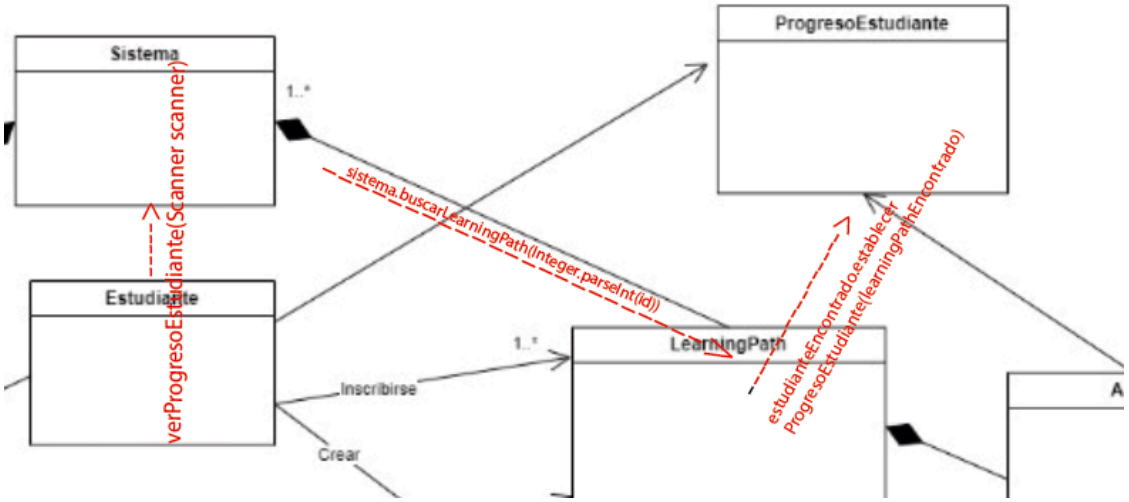
El estudiante le indica al sistema (structurer) que quiere encontrar learning path. El sistema busca y devuelve el learning path encontrado por el ID. Con el Learning path encontrado, se inscribe al estudiante.

Diagrama Iniciar Actividad:



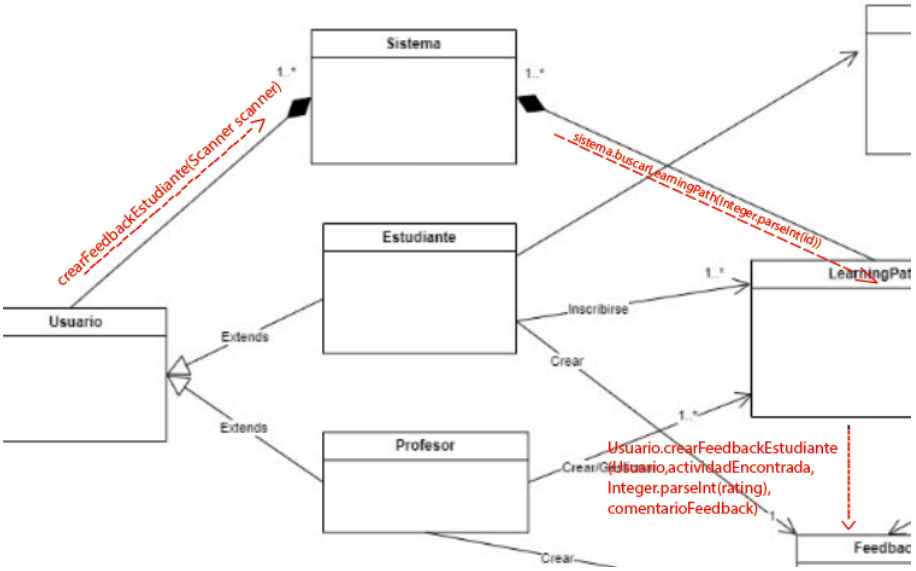
El estudiante busca la actividad que desea iniciar mediante el ID. Para ello el sistema (structurer) va a la lista de actividades y retorna la actividad correspondiente.

Ver Progreso Estudiante Diagrama:



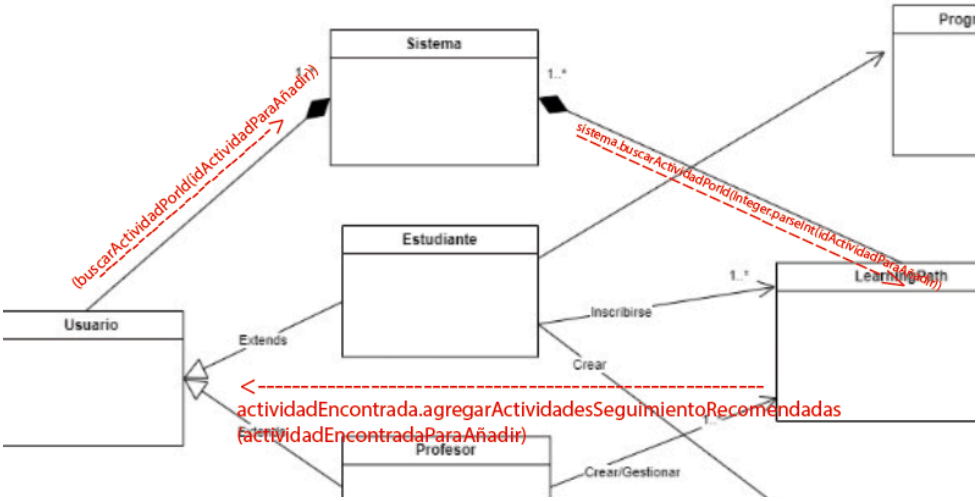
El estudiante o el profesor buscan un Learning path y un estudiante mediante su ID para luego almacenar su progreso. Todo esto se realiza a través del sistema (structurer).

Diagrama Crear Feedback:

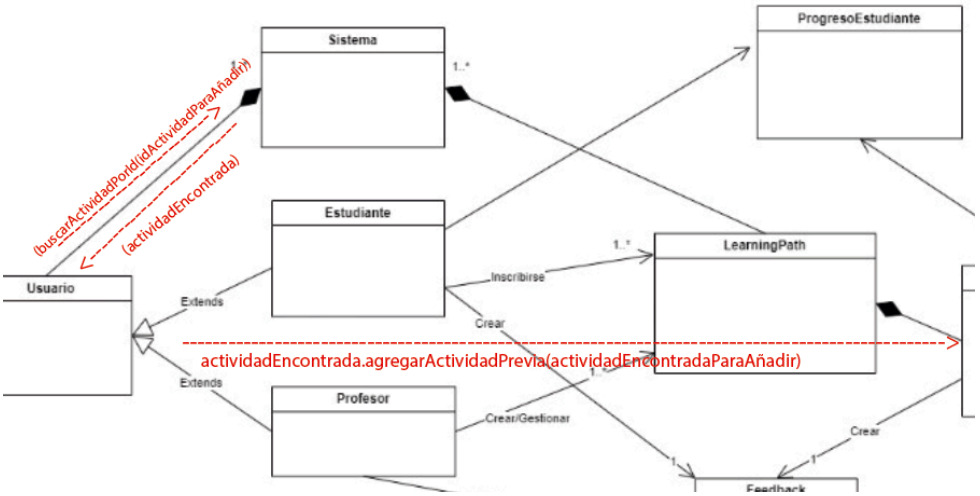


El estudiante busca una actividad por ID mediante el sistema. Una vez el sistema (structurer) encuentra la actividad busca el ID del estudiante y agrega la reseña en la actividad correspondiente.

Diagrama Añadir Actividades Seguimiento Recomendadas:

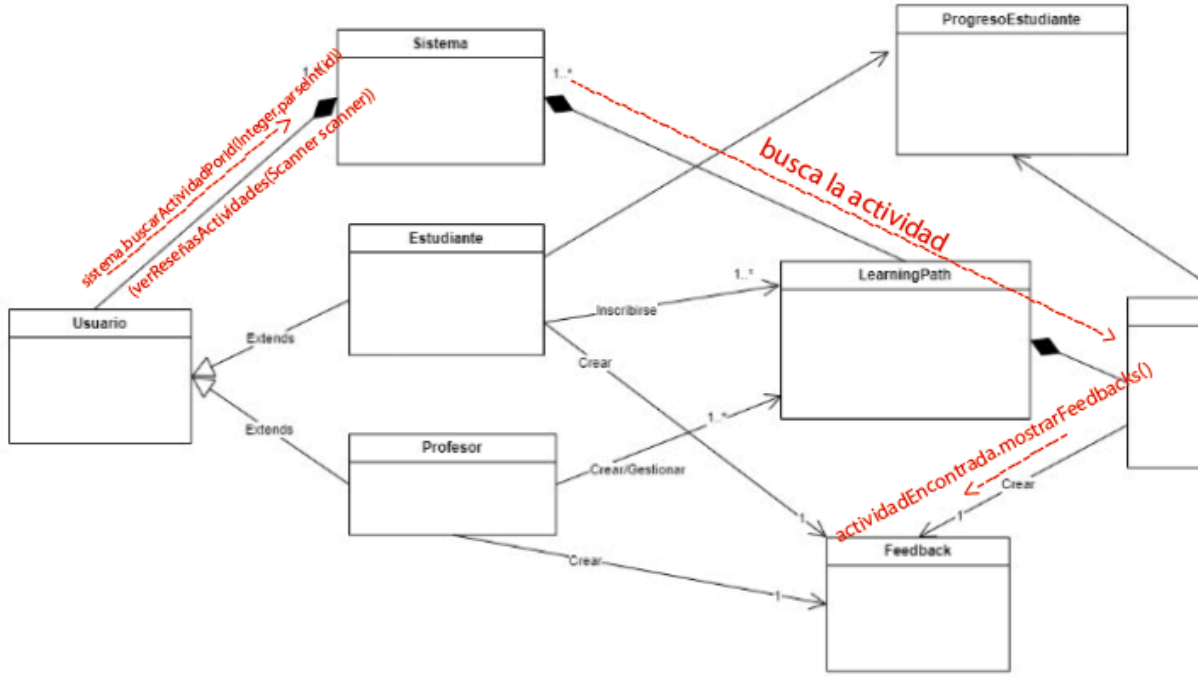


El usuario busca una actividad mediante su ID a través del sistema (structurer), luego retorna la actividad encontrada y le regresa al usuario una actividad recomendada.
Diagrama Añadir Actividades Previas:



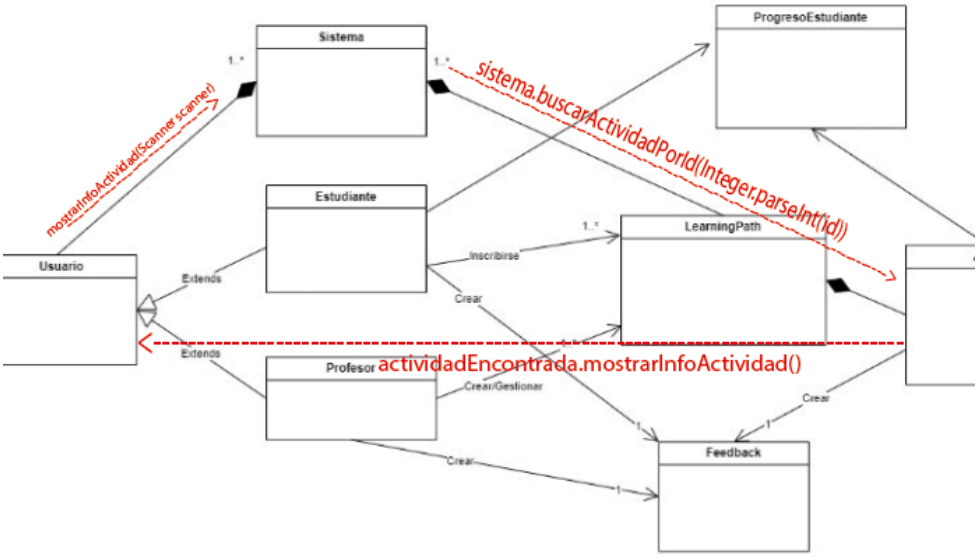
El usuario busca el ID de la actividad principal a través del sistema (structurer), luego el sistema busca la actividad solicitada y la regresa para el usuario añadiendo la actividad previa.

Diagrama Ver Reseñas Actividades:



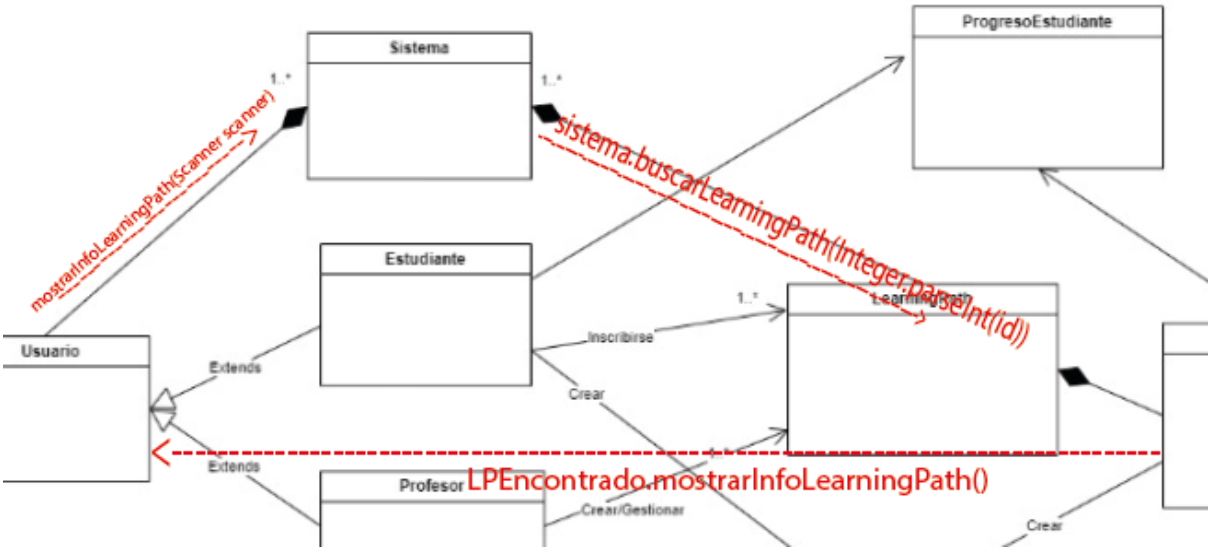
El usuario, a través de sistema (structurer), busca una actividad por ID en la lista de actividades. Una vez que la actividad está encontrada se acude a feedbacks para mostrar las reseñas de la actividad.

Diagrama ver información actividad:



El usuario inicia buscando una actividad por ID, a través del sistema (structurer). El sistema la busca en la lista de actividades por su ID. Una vez la encuentra, la retorna al usuario junto con toda su información adjunta.

Diagrama ver Información Learning Path :



El usuario busca un Learning path por su ID mediante el sistema, una vez encontrado se lo retorna al usuario con toda su información adjunta.