Juan José Cortés Villamil – 202325148

Alejandro Franco - 202225577

Amelia Serrano Arango- 202221556

Análisis de Diseño - Entrega 2 Proyecto 1

1. Introducción

El objetivo de este documento es presentar el diseño detallado del sistema Learning Path Recommendation System. Este sistema permite a los profesores crear y gestionar rutas de aprendizaje personalizadas (Learning Paths) para que los estudiantes puedan explorar sus intereses y desarrollar habilidades de manera estructurada. El software funciona de manera distinta para estudiantes y profesores. Por un lado, los estudiantes pueden inscribirse a learning paths y desarrollarlos uno a uno, ellos pueden luego realizar reseñas sobre el learning path y las actividades que realizaron. Los profesores, por su parte, están encargados de crear los learning paths para que los estudiantes los puedan realizar, añadiendo actividades como quices o exámenes, luego estos califican el trabajo y le permiten al estudiante continuar su aprendizaje.

2. Diagrama de Clases de Diseño (Hay una imagen de mejor calidad en el repositorio de Github)

Descripción del UML:

Clase: LearningPath

Atributos:

- título: String → El título que identifica al Learning Path.
- creador: String → El nombre del profesor que creó el Learning Path.
- descripcionGeneralContenidoYObjectos: String → Una descripción general sobre el contenido y los objetivos del Learning Path.
- nivelDificultad: String → Una palabra que indica la dificultad (por ejemplo: básico, intermedio, avanzado).
- Rating: int -> La cantidad de estrellas que tiene el LearningPath dentro de la comunidad.
- duracion: int → La duración estimada en minutos para completar todo el Learning Path.
- fechaCreacion: string → La fecha en la que se creó el Learning Path.
- fechaModificacion: string → La fecha de la última modificación realizada al Learning Path.
- actividades: List<Actividad> → Lista de actividades que componen el Learning Path
- estudiantes: HashMap<Estudiante, Progreso> → Mapa que relaciona estudiantes con su progreso en el Learning Path.

Métodos:

- Tiene todos los getters correspondientes a sus atributos, que nos permiten obtener información de forma eficiente.

Esta clase permite definir un camino de aprendizaje para los estudiantes que deseen utilizar este programa, esta clase se relaciona con la clase actividad mediante la agregación, una relacion de dependencia con profesor y estudiante ya que los profesores son quienes los crean y estudiantes quienes hace uso de ellos por lo que es importante que se relacionen y por ultimo tiene una relación con sistema que es la clase que lograra unir todo el programa.

Clase: Actividad (abstracta)

Atributos:

- id: int → Identificador único de la actividad.
- descripcion: String → Descripción de lo que trata la actividad.
- nivelDificultad: String→ Nivel de dificultad de la actividad.
- duracion: int → Tiempo estimado en minutos para completar la actividad.
- obligatoria: boolean → Define si la actividad es obligatoria o no dentro del Learning Path.
- creador: String → El nombre del profesor que creó la actividad.
- fechaLimite: String → Fecha límite para completar la actividad.
- estado: HashMap<Estudiante, String> → Un mapa que asocia a cada estudiante con el resultado de la actividad (por ejemplo, "completado", "pendiente", "fallido").
- Empezado: boolean → Define si el estudiante ya empezo a realizar esta actividad.

- **Métodos**:
    - Esta actividad tiene los getters para todos sus atributos ademas de un setter que determina el estado de la actividad con respecto a un estudiante.

Esta clase permite tener la información necesaria para llevar a cabo un learning path y completar el programa. De esta clase se heredan 5 subclases: Actividad recurso, encuesta, examen, quiz y tarea. Estas serán todos los tipos de actividades que se pueden llevar a cabo en un learning path. Así, promovemos la reutilización de código y la extensibilidad.

Clase: Tarea (extiende de Actividad)

- **Atributos**:
    - comentario: String → Comentario del profesor acerca de la tarea entregada por el estudiante.
- **Metodos:**
    - Esta clase solo tiene el get de su único atributo comentario.

Esta clase es necesaria ya que será el único tipo de actividad que llevará un comentario del profesor. Esta lase permite un mejor manejo de las calificaciones del profesor.

Clase: Quiz (extiende de Actividad)

- **Atributos**:
    - preguntas: List<PreguntaOpcionMultiple> → Lista de preguntas de opción múltiple que conforman el quiz.
    - calificacionMinima: int → Calificación mínima necesaria para aprobar el quiz.
    - respuestas: HashMap<Estudiante, List<Opcion>> → Un mapa que asocia a cada estudiante con sus respuestas seleccionadas en el quiz.
- **Métodos**:
    - Esta clase tiene getters para todos sus atributos y setters para las preguntas y respuestas

    Esta clase tiene relación con las preguntas de opción multiple, es importante ya que es uno de los tipos de actividad que hay en un learning path.

Clase: PreguntaOpciónMultiple (extiende de pregunta)

- **Atributos**:
    - opciones: ArrayList<Opcion> → Lista de opciones disponibles para la pregunta.
    - respuestaCorrecta: Opcion → La opción que es correcta.

- **Métodos**:

- Solo tiene get para la lista de opciones.

Esta clase hereda de la clase pregunta en donde se encontrará como atributo el enunciado y tiene una relacion de composición con la clase de opción.

Clase: Opción

- **Atributos**:
    - explicacion: String → Explicación de por qué la opción es correcta o incorrecta.
    - Correct: boolean → Define si la opción es la respuesta correcta a la pregunta
    - Enunciado: String → Es el enunciado de la pregunta a la que pertenece la opción.
    - Id: int → Es el identificador de la opcion
- **Métodos:**

    Tiene getters para todos sus atributos.

La clase Opcion facilita la gestión de opciones en preguntas de elección, asociando una opción a una Pregunta Opcion Multiple. Incluye atributos como Correct para identificar la opción correcta y explicacion para brindar retroalimentación a los estudiantes, que hacen que el programa sea mas eficiente.

Clase: ActividadRecurso (extiende de Actividad)

- **Atributos**:
    - documentPath: String → Ruta del archivo o recurso asociado a la actividad (por ejemplo, un PDF, video, o sitio web).
- **Métodos**:
    - showDocument(): void → Muestra o abre el documento asociado al recurso para que el estudiante lo pueda visualizar.

Esta actividad es la que permite que los estudiantes aprendan ya que no cuenta con ningun aspecto evaluativo, simplemente busca proporcionar el material necesario para cumplir con el learning path.

Clase: Examen (extiende de Actividad)

- **Atributos**:
    - preguntas: ArrayList<PreguntaAbierta> → Lista de preguntas abiertas que conforman el examen.

    - respuestas: HashMap<Integer, HashMap<String, Object>> → Un mapa que asocia a cada estudiante con su respuesta en el examen.
- **Métodos:**
    - SetCalificación(estudiante estudiante, int calificacion): void → Permite que un profesor califique a un estudiante que esta realizando el learning Path.
    - Ademas tiene un setter de las preguntas con las que cuenta el examen.

Esta clase permite tener una mejor organizacion de los posibles tipos de actividades que se pueden encontrar en un learning path.

Clase: Encuesta (extiende de Actividad)

- **Atributos**:
    - preguntas: ArrayList<Pregunta> → Lista de preguntas abiertas que forman la encuesta.
    - respuestas: HashMap<Integer, HashMap<String, Object>> → Un mapa que asocia a cada estudiante con sus respuestas a las preguntas de la encuesta.
- **Métodos**:
    - Cuenta con un getter y un setter para preguntas

Esta clase permite tener una mejor organizacion de los posibles tipos de actividades que se pueden encontrar en un learning path.

Clase: Sistema

- **Atributos**:
    - actividades: HashMap<Integer, Actividad>→ Mapa que relaciona el ID que la identifica con la información total de las actividades correspondientes que han sido creadas en el sistema.
    - learningPathsCreados :HashMap<String, LearningPath> → Mapa que relaciona los Learning Paths con el titulo con el que serán reconocidos que se están creando o han sido creados en el sistema.
    - estudiantes: HashMap<String, Estudiante> → Mapa de estudiantes que relaciona su información general con su login.
    - profesores: HashMap <String, Profesor> → Mapa de profesores que relaciona su información general con su login.
    - Connection connetion
    - Usuario Session

Clase: Usuario (abstracta)

- **Atributos**:

    - login: String → Nombre de usuario para autenticarse en el sistema.
    - password: String → Contraseña asociada al nombre de usuario
    - Correo: String → Es el correo con el que se inscribe el usuario
- **Métodos:**
    - Tiene un getter para el login

Esta clase es la que permite reconocer al usuario para saber si puede o no acceder al programa, ademas es la clase base para las subclases de estudiante y profesor.

Clase: Estudiante (extiende de Usuario)

- **Atributos**:
    - learningPathsInscritos: ArrayList<LearningPath> → Lista de Learning Paths en los que el estudiante está inscrito.

- **Métodos**:
    - Tiene un getter y un setter para los learning paths en los que esta inscrito.

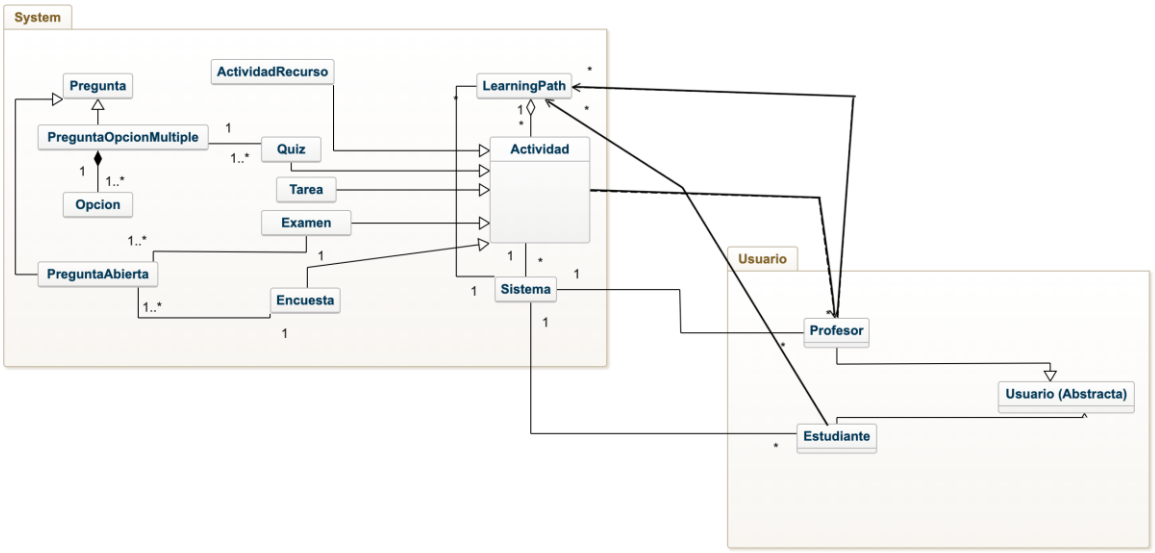
Clase: Profesor (extiende de Usuario)

- **Atributos**:
    - **learningPathsCreados: List<LearningPath>** → Lista de Learning Paths creados por el profesor.
    - **ActividadesCreadas: ArrayList<Actividad>()** → Lista de actividades creados por el profesor.

- **Métodos**:
    - **añadirLearningPath()** → Permite que el profesor cree un nuevo Learning Path.
    - **crearActividad()** →Permite que el profesor cree una nueva actividad
    - Ademas de getters y setters para los learning paths y actividades creadas por este profesor

  La subclase Profesor extiende Usuario y es importante ya que añade métodos específicos para crear Learning Paths y actividades.

3. **Diagrama de Clases de Alto Nivel**



4. Justificación del Diseño

Algunas decisiones importantes:

1. Hacer una clase por cada tipo de actividad:

Crear una clase separada para cada tipo de actividad permite una representación clara y organizar cada tipo de actividad dentro del sistema. Cada actividad tiene diferentes características y comportamientos específicos que pueden ser implementados en sus respectivas clases. Esta organización permite que:

- Cada tipo de actividad se pueda definir su propio comportamiento, atributos y métodos.
- Al tener una clase para cada actividad, es fácil añadir nuevos tipos de actividades en el futuro sin cambiar la estructura básica.
- Cómo cada tipo de actividad puede heredar de una clase abstracta Actividad, se reduce el código repetido.

2. Hacer una super clase de usuarios de la cual hereden estudiante y profesor:

Crear una superclase Usuario de la cual heredan Estudiante y Profesor da claridad y eficiencia al diseño del sistema. Esto ofrece varios beneficios:

- Cómo usuario contiene atributos y métodos que todos los usuarios necesitan, como nombre, login, password, y métodos para autenticación. Esto evita la duplicación de código y simplifica las modificaciones.
- Diferentes roles pueden acceder a diversas funcionalidades. Profesor puede tener permisos para crear y editar actividades, mientras que Estudiante solo interactúa con las actividades. Esta estructura permite implementar estas diferencias de forma sencilla.

3. Hacer clases de pregunta abierta, seleccion multiple y opción:

Diseñar clases específicas para los tipos de preguntas y opciones en actividades de evaluación hace que el sistema sea más flexible. Es necesario ya que cada tipo de pregunta (PreguntaAbierta y de PreguntaOpcionMultiple) tienen diferentes estructuras y comportamientos. Una PreguntaAbierta permite respuestas textuales, mientras que una PreguntaOpcionMultiple requiere opciones predefinidas y una respuesta correcta. La especialización de estas clases permite manejar estos requisitos específicos. La Clase Opcion es importante ya que PreguntaOpcionMultiple puede asociarse con múltiples instancias de Opcion, lo cual estructura claramente cada respuesta posible.

5. **Funcionamiento de la base de datos:**

Utilizamos una base de datos Derby de la maquina virtual de Java en su modo Embeded que se puede adjuntar al proyecto. En esta base de datos se encuentran tablas que tienen toda la información necesaria para el funcionamiento del programa. Estas cuentan con unas primary keys que son la llaves principales de una tabla y son las que verifican que los datos no se puedan repetir, esto es muy util, por ejemplo, en el caso de los usuarios ya que de esta forma podemos verificar que todos los logins sean únicos. Además las tablas se pueden relacionar entre ellas, donde las columnas pueden referenciar información de otras tablas para así hacer conexiones entre la información que esta en el programa. En esta base de datos encontramos las tablas de: Usuarios, Learning Paths, Activities y otras que ayudan a asociar el estado de las actividades y de los learning paths.