

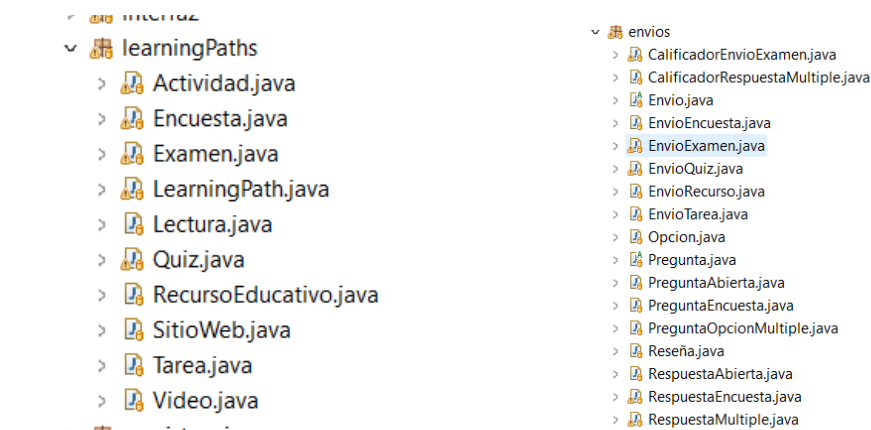
Juan Manuel Rojas 202321306
Julián Restrepo 202320177
Julián Pinto 202321207

Justificaciones de diseño de nuestra entrega:



Nuestra entrega final representa una síntesis de las mejoras y aprendizajes adquiridos a lo largo del año, integrando recomendaciones de las entregas anteriores y aplicando los principios y patrones de diseño estudiados en clase. Nuestro objetivo fue desarrollar una aplicación que se caracterice por su eficacia, facilidad de uso e implementación, sin comprometer la escalabilidad ni el mantenimiento.

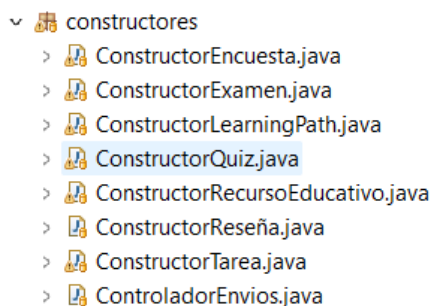
Nuestra mayor virtud fue la **alta cohesión** y el **bajo acoplamiento**. Dividimos los requerimientos en clases independientes para que encontrar errores fuera lo más simple posible, afectando únicamente una o dos clases como máximo. La estructura promueve la cohesión al organizar funcionalidades relacionadas dentro de clases específicas, asegurando que cada clase cumpla una única responsabilidad (Single Responsibility Principle). El bajo acoplamiento entre módulos, como usuarios y actividades, facilita la escalabilidad y reduce la complejidad de mantenimiento, ya que los cambios en un componente no afectan significativamente a los demás.



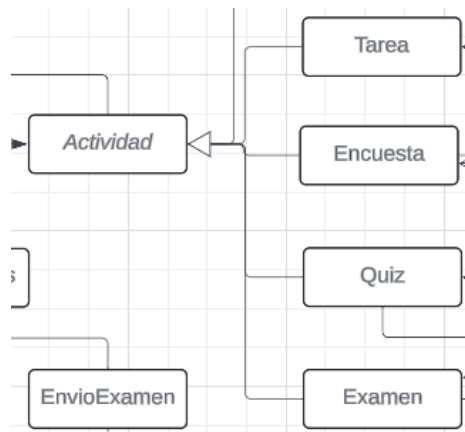
Otra gran decisión tomada por parte de nosotros fue el patrón de diseño **Singleton** que usamos a lo largo de nuestra implementación. Esto garantiza que una clase tenga una única instancia y proporciona un punto global de acceso a ella. Esta la usamos para guardar los datos de los usuarios creados en el sistema. Esto permitió reducir significativamente los problemas relacionados con la implementación de este patrón y optimizar el tiempo destinado a resolver cuestiones de lógica más complejas.

```
public static ControladorUsuarios getInstancia() {
    if (instancia == null) {
        instancia = new ControladorUsuarios();
    }
    return instancia;
}
```

Además, adoptamos el patrón de diseño **Builder** para crear actividades personalizadas como tareas, encuestas y exámenes. Este patrón nos permitió construir objetos complejos combinando diferentes atributos de manera flexible y modular, reduciendo significativamente el acoplamiento. Este enfoque también se alineó con el Principio Abierto/Cerrado (OCP), permitiendo extender las funcionalidades del sistema (como añadir un nuevo tipo de actividad) sin modificar las clases existentes.



Otra justificación clave para nuestro diseño fue la implementación de clases abstractas para usar el **patrón de herencia**. Un ejemplo claro de esto es la actividad de la cual heredan los diferentes tipos de actividades. Esta decisión se fundamenta en que todas las actividades comparten atributos y comportamientos comunes, como descripciones, duración, nivel de dificultad, entre otros. Utilizar herencia redujo la duplicación de código, facilitó la organización del sistema y permitió que cada subclase añadida o sobrescriba comportamientos específicos de manera clara y modular.



Por último una gran justificación fue separarlos calificadoros. La decisión de tener calificadoros especializados para diferentes tipos de actividades asegura que el comportamiento de calificación sea preciso y ajustado a las necesidades específicas de cada tipo de pregunta. Esto permite manejar eficientemente las diferencias entre un examen de preguntas abiertas y un quiz de opción múltiple. Incluso cuando nos pidieron más adelante el proyecto 2 cambiar el tipo de quiz a verdadero y falso, fue realmente fácil ya que fue solo cambiar el calificador de respuestas múltiples y abiertas.



- >  CalificadorEnvioExamen.java
- >  CalificadorRespuestaMultiple.java

Diagrama de clases: (adjunto en el repositorio)

Diagrama de alto nivel: (adjunto en el repositorio)

Diagrama de alto nivel interfaz:(adjunto en el repositorio)

En resumen, nuestra entrega final no solo integra las recomendaciones y aprendizajes adquiridos a lo largo del año, sino que también refleja un diseño sólido basado en principios y patrones de desarrollo. Nuestro sistema combina simplicidad, funcionalidad y buenas prácticas de ingeniería de software. Creemos que el enfoque adoptado garantiza una experiencia eficiente para los usuarios y sienta una gran base para futuras mejoras o adaptaciones de nuestro mismo sistema.

Requerimientos funcionales

1. Gestión de Usuarios:

- Los usuarios deben poder registrarse y autenticarse en el sistema.
- Cada usuario debe poder editar su perfil, actualizar cierta información personal y cambiar su contraseña.

2. Creación y Gestión de Learning Paths:

- Los profesores pueden crear y editar Learning Paths
 - Los profesores pueden clonar Learning Paths de otros profesores y hacer modificaciones en la copia.
- 3. Gestión de Actividades:**
- Los profesores pueden crear y gestionar diferentes tipos de actividades
 - Los profesores deben poder marcar actividades como obligatorias u opcionales dentro de un Learning Path.
 - Los estudiantes deben poder reseñar las actividades y darles un rating.
- 4. Progreso de los Estudiantes:**
- El sistema debe rastrear el progreso de los estudiantes en los Learning Paths en los que estén inscritos, calculado como un porcentaje de actividades obligatorias completadas exitosamente.
 - Un estudiante solo puede iniciar una actividad a la vez.
- 5. Sistema de Recomendación:**
- El sistema debe recomendar Learning Paths a los estudiantes basándose en sus intereses y progreso anterior.
 - El sistema debe permitir al estudiante seleccionar qué actividades realizar, con advertencias si no se han completado actividades previas recomendadas.
- 6. Feedback y Rating:**
- Profesores y estudiantes pueden dejar reseñas y ratings sobre actividades y Learning Paths.
 - El feedback debe ser visible para los profesores al editar o crear Learning Paths, para seleccionar las actividades más apropiadas.
- 7. Manejo de Quizzes y Evaluaciones:**
- El sistema debe permitir crear quizzes con preguntas de opción múltiple, con una sola respuesta correcta.
 - Las preguntas deben tener explicaciones que se muestren al estudiante después de realizar el quiz.
 - Las tareas y exámenes deben tener un estado de “enviado” hasta que el profesor las califique manualmente.
- 8. Seguridad y Autenticación:**
- Todos los usuarios deben autenticarse con login y contraseña.

Restricciones del Proyecto

- Los estudiantes no pueden iniciar más de una actividad a la vez.
- Solo se puede realizar la encuesta si ya se hizo el quiz o el examen
- Los profesores pueden clonar actividades, pero no editarlas si no son los creadores.
- El sistema debe soportar varios tipos de actividades (Clase, Tarea, Quiz, Examen, Encuesta).
- El progreso se calcula solo en función de las actividades obligatorias completadas.
- Una pregunta de opción múltiple debe tener 4 opciones y solo puede tener una respuesta correcta

- Un estudiante solo puede mandar una reseña por actividad
- La actividad estará en el estado “enviada” hasta que el profesor responsable del Learning Path la marque como exitosa o no exitosa
- Los Títulos de los Learning Paths son únicos