# **Formulario: Vista Lógica del Proyecto de Software**

**Proyecto:** KineCap  
**Grupo:** 3  
**Fecha:** 18/10/2024

**1. Descripción General del Sistema**

Proporciona una descripción general del sistema y los principales componentes lógicos que lo forman.

**Descripción del sistema:**

KineCap es una aplicación web desarrollada con React en el frontend y Flask en el backend, diseñada para la gestión de cursos en el área de capacitación médica. Incluye módulos para la autenticación de usuarios, manejo de cursos, y evaluación del progreso educativo. Los usuarios pueden inscribirse en cursos relacionados con técnicas de kinesiología y tratamientos médicos, completar módulos y lecciones, y finalmente recibir certificaciones. La aplicación también integra una base de datos PostgreSQL para la gestión y almacenamiento de la información.

**Componentes lógicos principales:**

* Autenticación de usuarios.
* Gestión de cursos, módulos y lecciones.
* Evaluación del progreso y obtención de certificaciones.
* Almacenamiento seguro de la información.
* Sistema de Pagos. (incorporación futura)

**2. Principales Clases y Entidades del Sistema**

Describe las clases o entidades principales del sistema, sus atributos y métodos principales, y su propósito dentro del sistema.

**Clase User:   
Atributos:** id, nombre, apellidos, telefono, email, pais, password, created\_at. **Métodos:** register(), login(), reset\_password(), update\_password(), get\_users(). **Propósito:** Manejar la información del usuario, incluyendo el registro, autenticación y gestión de credenciales para acceder a la plataforma.

**Clase 2 Course:   
Atributos:** id, nombre, descripcion, created\_at.  
**Métodos:** add\_course(), get\_course(). **Propósito:** Representar los cursos disponibles en la plataforma, gestionando la información de cada curso.

**Clase 3 Module:   
Atributos:** id, nombre, course\_id, created\_at. **Métodos:** add\_module(), get\_modules(). **Propósito:** Agrupar las lecciones dentro de un curso y proporcionar una organización adecuada del contenido.

**Clase 4 Lesson:   
Atributos:** id, nombre, contenido, module\_id, created\_at **Métodos:** add\_lesson(), get\_lessons(), mark\_complete(). **Propósito:** Proporcionar contenido específico dentro de un módulo, como videos, cuestionarios, o documentos de referencia para facilitar el aprendizaje del usuario.

*(Agrega más clases si es necesario)*

**3. Relaciones entre clases**

Especifica cómo están relacionadas las clases entre sí, indicando relaciones de herencia, composición, agregación o dependencias.

**Relaciones entre clases:**

**User:**

* **Asociación con Course:** Un usuario puede inscribirse en varios cursos y un curso puede tener varios usuarios inscritos. (Relación muchos a muchos).
* **Asociación con Certificado:** Un usuario puede recibir varios certificados, pero cada certificado pertenece a un solo usuario. (Relación uno a muchos).

**Course:**

* **Composición con Module:** Un curso está compuesto por varios módulos. Si se elimina el curso, también se eliminan los módulos correspondientes.
* **Asociación con User:** Un curso puede tener varios usuarios inscritos.

**Module:**

* **Composición con Lesson:** Un módulo está compuesto por varias lecciones. Si se elimina el módulo, también se eliminan sus lecciones.

**Lesson:**

* **Dependencia con Module:** Las lecciones están vinculadas a un módulo específico.

**4. Interfaces y dependencias**

Indica si alguna de las clases o entidades tiene interfaces y cómo interactúan con otros módulos del sistema a través de ellas.

**Descripción de interfaces y dependencias:**

**Interfaces:**

**REST API:**

El backend implementa una API REST para manejar la comunicación entre el frontend y la base de datos.

Las principales operaciones que se realizan mediante esta API REST incluyen:

**GET:**

* Obtener información sobre cursos disponibles, módulos y lecciones.
* Consultar detalles de un usuario específico (por ejemplo, perfil o progreso).

**POST:**

* Registrar un nuevo usuario.
* Iniciar sesión para obtener un token de autenticación.
* Agregar un nuevo curso, módulo o lección.

**PUT:**

* Actualizar información del perfil del usuario.
* Actualizar el progreso del usuario en los módulos y lecciones.

**DELETE:**

* Eliminar usuarios, cursos, módulos o lecciones en caso de ser necesario.

Los endpoints de la API están diseñados para ser fácilmente utilizables desde el frontend, proporcionando respuestas en formato JSON, lo que permite al frontend manejar los datos de forma dinámica y amigable para el usuario.

La autenticación se maneja utilizando JSON Web Tokens (JWT). Cada usuario que inicia sesión recibe un token que debe incluirse en los headers de las solicitudes a los endpoints protegidos. Esto asegura que solo los usuarios autenticados puedan acceder a ciertos recursos, como ver los módulos y lecciones del curso.

**Dependencias:**

Los usuarios utilizan formularios para registrarse e iniciar sesión. Estas acciones envían peticiones al backend mediante la API REST.

**El frontend (React) interactúa con los endpoints del backend para:**

* Obtener información sobre cursos, módulos y lecciones.
* Actualizar el progreso de los usuarios en las lecciones.

**Tecnologías utilizadas:**

* Flask para el backend, PostgreSQL para la base de datos y Flask-JWT-Extended para la autenticación.
* Flask-CORS permite la comunicación entre frontend y backend, facilitando el desarrollo y la ejecución en dominios diferentes.

**5. Diagrama de Clases del Sistema**

Adjunta un diagrama de clases que representa la estructura lógica del sistema, las clases principales y las relaciones entre ellas.

**Diagrama de clases:***(Adjunta el diagrama en esta sección)*

**Diagrama

Descripción generada automáticamente**

**6. Diagrama de componentes (opcional)**

Si su sistema lo requiere, adjunte un diagrama de componentes que represente cómo los componentes principales del sistema están organizados y cómo interactúan.

**Diagrama de componentes:***(Adjunta el diagrama en esta sección)*

**7. Comentarios adicionales**

Incluye cualquier comentario adicional o aclaración relevante sobre la vista lógica de tu sistema.

**Comentarios:**

1. Se planea integrar más tipos de lecciones, incluyendo cuestionarios con puntajes de aprobación, así como la posibilidad de subir archivos para completar tareas.
2. Los módulos y lecciones tendrán lógica de dependencia para asegurar que los usuarios completen las lecciones en el orden correcto antes de avanzar a la siguiente.
3. La arquitectura está diseñada para facilitar la escalabilidad, permitiendo agregar más funcionalidades como foros de discusión, seguimiento de progresos y evaluaciones detalladas.