# Informe Final Integrado

## Resumen Ejecutivo

Este informe resume y documenta el desarrollo del MVP de EduTech IA. Esta aplicación tiene como objetivo ser una plataforma de aprendizaje interactiva potenciada por IA, la cual recomendara cursos a sus alumnos en base a sus gustos y preferencias.

A lo largo de este proyecto, se han abordado aspectos clave a tener en cuenta a la hora de desarrollar cualquier aplicación, como la selección de un lenguaje y paradigma de programación adecuados para la naturaleza del proyecto, configuración del entorno de desarrollo, y desarrollo de un pipeline de integración y entrega continua, entre otros. Se han aplicado buenas prácticas de desarrollo, enfocándose principalmente en escalabilidad, seguridad, y capacidad para el desarrollo cooperativo.

El objetivo principal ha sido crear la base para el desarrollo de un producto mínimo viable, sobre el cual se pueden añadir nuevas funcionalidades que se adapten a las necesidades de los usuarios, las cuales cambiaran según factores externos. Con todo este sistema establecido, incluyendo todas las herramientas a utilizar a lo largo del desarrollo de la aplicación, solo queda delegar responsabilidades a cada uno de los desarrolladores y desarrollar la aplicación en código.

## Justificación de Elecciones Técnicas y Metodológicas

Selección del Lenguaje y Paradigma de Programación:

El MVP se ha desarrollara usando Java como lenguaje de programación. He tomado esta decisión debido a la robustez, comunidad amplia, y compatibilidad con entornos empresariales del lenguaje. Es un lenguaje que va a facilitar la cooperación, ya que es comúnmente usado por desarrolladores de aplicaciones como esta. Además, se ajusta bien al paradigma de programación que he escogido.

Este paradigma es el orientado a objetos. Hay varias razones para esta decisión. Nos permite modular el código en distintas clases, las cuales se pueden desarrollar cada una por separado por distintos programadores. Este formato también facilita la escalabilidad tan necesaria para este tipo de aplicaciones. Se pueden crear nuevas clases que hereden funcionalidades de clases existentes. Por ejemplo, se pueden crear nuevos cursos que hereden métodos de la clase Curso, o alterar el modelo de IA sin tener que modificar ningún otro aspecto del programa.

Esto sin embargo no significa que el único lenguaje que se vaya a utilizar a lo largo del programa sea Java o el paradigma orientado a objetos. Para desarrollar el front-end y la interfaz gráfica de la aplicación será necesario el uso de lenguajes como HTML, CSS, y JavaScript, o incluso se podrá usar Python para el desarrollo del modelo de inteligencia artificial, debido a su mayor robustez y abundancia de librerías en este aspecto. Esta combinación permite optimizar cada aspecto del programa según sus necesidades específicas, y así logramos un equilibrio entre facilidad de desarrollo y rendimiento.

Diseño de la Arquitectura del software:

La arquitectura del MVP se basa en un modelo muy básico y escalabale basado en 4 clases principales. Mas bases se podrán ir incorporando a lo largo del desarrollo, en forma de nuevos cursos o funcionalidades. Este es un modelo extremadamente simplificado comparado a lo que va a ser la estructura del programa final, ya que no se están teniendo en cuenta elementos de la interfaz grafica ni de interacciones con APIs o bases de datos, y solo se centra en la función básica de la plataforma.

La clase “Campus” sirve de Hub principal para el manejo de los cursos y alumnos. Contiene estructuras de datos de todos los cursos disponibles a los alumnos, y de todos los usuarios del campus. Además, contiene métodos para añadir y borrar usuarios y cursos, además de mostrarlos todos. La función de mostrar cursos será usada en la interfaz gráfica para mostrarle todos los cursos disponibles a los usuarios, pero la función de mostrar usuarios deberá ser disponible solo para los administradores de la plataforma.

La clase Usuario contiene todos los atributos de los usuarios, y sus instancias representan cuentas adscritas en la plataforma. Los atributos de user y password deberán ser privados y solo visibles para los administradores. La clase también contiene una lista de todos los cursos suscritos del usuario, y de sus intereses, los cuales serán usados para recomendarle nuevos cursos.

Las clases Usuario y Campus implementan una interfaz llamada GestorCursos, la cual contiene métodos comunes tanto para los usuarios como para el campus, como por ejemplo las habilidades de añadir o borrar cursos. En el contexto del campus, esto añade y borra cursos de la plataforma, mientras que en el usuario se usan para subscribirse o darse de baja de cursos. Este es el principal beneficio de las interfaces, que sirven como una promesa de que se van a implementar ciertos métodos en una clase, lo que es beneficioso para la claridad en el desarrollo colaborativo, pero estos métodos pueden tener distintas implementaciones según la clase en la que se usan.

La clase Curso sirve como marco para los cursos de la plataforma, representados por sus instancias. Tiene un atributo para el nombre del curso y su tipo. Los tipos de cursos están en un enumerativo TiposCurso, el cual contiene zonas de conocimiento generales comunes como humanidades, matemáticas, o informática. Este enumerativo podrá ser expandido en un futuro. También contiene una lista con todos los usuarios adscritos al curso, lo que facilitara el trabajo de los administradores de la plataforma.

Selección de Herramientas y Configuración del Entorno: