**Sistema de Detección de Plagio Académico PlagiTracker**

**Visión**

**Versión 2.0**

**Visión**

1. **Posicionamiento**
   1. **El Problema**

| El problema | En el desarrollo de software, especialmente en entornos académicos, es común que los estudiantes, tanto principiantes como avanzados, reutilicen fragmentos de código y de diferentes fuentes sin dar el debido crédito o sin la autorización correspondiente. Esto no solo plantea problemas éticos y legales, sino que también obstaculiza el aprendizaje y la creatividad. |
| --- | --- |
| Afecta | **Estudiantes de programación:** Reutilizar o compartir código en exceso entre estudiantes de programación impide el desarrollo de habilidades propias. Esta dependencia los convierte en programadores ineficientes, incapaces de resolver problemas por sí mismos, lo que afecta su rendimiento en situaciones reales.  **Docentes:** Los encargados de evaluar proyectos de programación pueden enfrentar dificultades para detectar plagio, lo que complica la evaluación justa y el desarrollo de habilidades auténticas en los estudiantes. |
| El impacto | La ausencia de herramientas adecuadas para detectar similitudes en el código puede aumentar el plagio, comprometiendo la calidad del software y dificultando evaluaciones justas. Desarrollar una aplicación que identifique plagio de código ayudaría a los docentes a detectar casos de manera más eficiente, mejorando la integridad académica y promoviendo un entorno de aprendizaje más justo y riguroso. |

* 1. **Posición del Producto**

| Producto | Un sistema de detección de plagio especializado para código. |
| --- | --- |
| Diferenciación | La principal diferenciación es que contará con un registro de estudiantes similar al modelo de Google Classroom. Los estudiantes podrán inscribirse en cursos de diferentes profesores, quienes tendrán acceso a la creación de clases y dentro de ellas podrán generar tareas. En estas tareas se realizará la detección automática de plagio, facilitando el control y la transparencia en las evaluaciones. Además, habrá una cuenta de administrador para gestionar todo el sistema. |

**Stakeholders y Descripciones de Usuarios**

* 1. **Stakeholders**

| **CLIENTE** | **DESCRIPCIÓN** |
| --- | --- |
| Docentes | Utilizan la aplicación para detectar el plagio , crear cursos y tareas. |
| Alumnos | Se inscribe en los cursos que crea el docente. |

| **ROL** | **DESCRIPCIÓN** | **RESPONSABLES** |
| --- | --- | --- |
| **Líder de Proyecto** | Coordina y supervisa el desarrollo del sistema, asegurando que se cumplan los requisitos y plazos. Es el principal punto de contacto entre el equipo de desarrollo y los stakeholders. | Isabel Karina Ttito Campos |
| **Revisor** | Garantiza que el software cumpla con los estándares de calidad y las especificaciones requeridas. | Karlo Pacha Curimayhua |
| **Analista de Requisitos** | Trabaja con los stakeholders para identificar y documentar las necesidades y requisitos del sistema, asegurando que el desarrollo del proyecto esté alineado con las expectativas de los usuarios finales. | Jorge Olivera Ticona  Olger Quispe Vilca  LuisFelipe Mamani Arosquipa  Karlo Pacha Curimayhua  Elmerson Portugal Carpio  Aron Hurtado Cruz  Luis Luque Nieto  Isabel Karina Ttito Campos |
| **Arquitecto de Software** | Diseña la estructura y la arquitectura general del sistema, definiendo los componentes principales y asegurando que el sistema sea escalable, mantenible y eficiente. | Jorge Olivera Ticona  Olger Quispe Vilca |
| **Desarrollador Frontend** | Se encarga de implementar la interfaz de usuario del sistema, asegurando que sea intuitiva, accesible y fácil de usar para los docentes y otros usuarios finales. | LuisFelipe Mamani Arosquipa  Jorge Olivera Ticona  Olger Quispe Vilca |
| **Desarrollador backend** | Implementa la lógica del servidor, la base de datos y las APIs necesarias para el funcionamiento del sistema, incluyendo la implementación de los algoritmos de detección de plagio. | Karlo Pacha Curimayhua  Elmerson Portugal Carpio  Aron Hurtado Cruz  Luis Luque Nieto |
| **Tester / QA** | Realiza pruebas de calidad en el sistema para identificar y corregir errores, asegurando que el sistema funcione correctamente bajo diferentes escenarios de uso. | Aron Hurtado Cruz  Karlo Pacha Curimayhua |
| **Diseñador**  **UI/UX** | Diseña interfaces visuales y experiencias de usuario intuitivas, optimizando la funcionalidad y estética del producto para satisfacer tanto las necesidades del usuario como los objetivos del negocio. | LuisFelipe Mamani Arosquipa |

* 1. **Resumen de necesidades de Stakeholders**

| **Necesidad** | **Solución propuesta** | **Prioridad** |
| --- | --- | --- |
| Interfaz intuitiva | Usar componentes gráficos minimalistas. | Media |
| Extracción del código fuente desde las URLs proporcionadas por los usuarios. | Utilizar una herramienta automatizada para extraer el ejercicio de programación del estudiante desde las URLs proporcionadas por los usuarios, identificando y capturando las secciones relevantes del contenido. | Alta |
| Manejar un gran número de URLs. | Usar un ingreso de datos dinámicos. El rango básico de URLs podría estar entre 2 a más URLs por análisis. | Alta |
| Tener una vista ágil de similitudes existentes | El informe generado muestra las coincidencias entre las distintas URLs ingresadas. | Alta |
| Identificar correctamente a los involucrados | El estudiante deberá incluir su nombre o identificador en un comentario en el ejercicio de programación, así cuando se extraiga y se analice el ejercicio por la herramienta se podrá saber su autor.  El sistema asigna un identificador único y permanente al estudiante después que se haya logueado. | Alta |
| Cuantificar el porcentaje de similitud existente | Implementar un algoritmo de comparación que analice y cuantifique las similitudes entre los ejercicios de programación extraídos. | Alta |

1. **Requisitos Especiales**

Requisitos especiales de software

* **Usabilidad:** El sistema debe ser intuitivo para el usuario, en el caso del docente incluyendo las configuraciones que pueda elegir para el cambio en el sistema.
* **Rendimiento:** El sistema es capaz de analizar y comparar los ejercicios de programación con las URLs obtenidas, los archivos pequeños/medianos serán procesados en corto tiempo. Para archivos más grandes, el tiempo de procesamiento debe ser escalable de manera eficiente, manteniendo un rendimiento aceptable para nuestro sistema.
* **Seguridad:** El sistema debe garantizar la protección y privacidad de los datos de los usuarios mediante el cifrado seguro de contraseñas, implementación de controles de acceso adecuados, protección de datos personales y académicos.
* **Escalabilidad:** Una arquitectura bien diseñada permitirá que el sistema maneje el versionamiento y el mantenimiento de manera eficiente, facilitando la incorporación de nuevas funciones o la modificación de estas mismas sin que el proceso se vuelva complejo.
* **Capacidad de Prueba:** La arquitectura bien diseñada permitirá la facilidad de hacer diferentes tipos de prueba al sistema.
* **Compatibilidad:** El sistema será compatible con todos los navegadores y todas las resoluciones de equipos móviles y de escritorio.