UNIVERSIDAD LA SALLE

CodeGuardGT
Visión

Versión 2.2.3

Visión

1. Posicionamiento

El sistema CodeGuardGT se posiciona como una herramienta integral para la detección de plagio en entornos de programación SQL, garantizando la integridad académica y la calidad educativa.

1.1 El Problema

El problema	En el entorno académico, es común que los estudiantes recurren al plagio para completar sus tareas de programación, copiando código de compañeros o de fuentes en línea.
Afecta	La integridad académica y la evaluación justa de los conocimientos y habilidades de los estudiantes.
El impacto	El plagio compromete la integridad académica y la evaluación justa de los conocimientos y habilidades de los estudiantes, desvirtuando el proceso educativo, desmotivando a los estudiantes honestos y afectando la calidad de la formación académica.
Solución satisfactoria	Implementar un sistema de detección de plagio especializado en código SQL que combine múltiples técnicas para garantizar una evaluación justa y precisa del código presentado por los estudiantes.

1.2 Posición del Producto

Para	Instituciones educativas que buscan asegurar la integridad académica en la entrega de código SQL por parte de los estudiantes.
Quién	Docentes y administradores académicos que necesitan herramientas efectivas para detectar y gestionar el plagio en proyectos de programación SQL.
Producto	Un sistema de detección de plagio en un compilador SQL online.
Que	Proporciona una evaluación exhaustiva del código SQL, identificando plagio tanto de fuentes internas (entre compañeros) y detectando patrones sospechosos basados en la ejecución y el historial del código.
A diferencia de	Otros sistemas que se enfocan en técnicas específicas o que no contemplan todas las posibles fuentes de plagio.
Nuestro producto	Ofrece una solución híbrida que combina detección textual, comparación con bases de datos en línea, comparación entre entregas de estudiantes, análisis de resultados de ejecución y detección basada en el historial de código para una protección completa contra el plagio en SQL.

2. Stakeholders y Descripciones de Usuarios:

2.1 Stakeholders

ROL	DESCRIPCIÓN
Coordinador Académico	Supervisan la implementación del sistema y su alineación con las políticas académicas de la institución.
Docente	Utilizan el sistema para evaluar la originalidad del código SQL presentado por los estudiantes y proporcionar retroalimentación.
Estudiante	Usuarios finales que envían sus proyectos de programación SQL y deben cumplir con las políticas de integridad académica.

2.2 Resumen de necesidades de Stakeholders

Necesidad	Solución propuesta	Prioridad
Detección precisa de plagio en código SQL.	Implementar un sistema de detección de plagio con múltiples técnicas	**Alta**, para asegurar la integridad académica y la evaluación justa.
Minimizar interrupciones durante el desarrollo del código SQL.	Realizar la detección de plagio en la etapa final para evitar interferencias.	**Alta**, para mantener un flujo de trabajo ininterrumpido.
Cobertura amplia para detectar plagio tanto interno como externo.	Integrar herramientas que comparen el código SQL con recursos en línea y entre entregas de estudiantes.	**Alta**, para asegurar una detección exhaustiva.

3. Requisitos Especiales:

- **Estándares y Requisitos de Plataforma:**
  - **Compatibilidad con SQLFiddle**: El sistema debe ser completamente compatible con SQLFiddle (https://sqlfiddle.com/), permitiendo la integración directa para la evaluación de código SQL ingresado en esta plataforma. No se considerará soporte para otros compiladores SQL o entornos de desarrollo integrados (IDEs) de escritorio.
  - **Interoperabilidad**: El sistema solo interactúa con SQLFiddle para importar y analizar el código SQL, sin proporcionar soporte o documentación para el uso de SQLFiddle.
- **Requisitos de Rendimiento**
  - **Tiempo de Análisis**: El sistema debe realizar análisis de plagio en menos de 3 minutos para conjuntos de datos de hasta 5,000 líneas de código, optimizado específicamente la interacción con SQLFiddle.
  - **Escalabilidad**: Debe manejar múltiples consultas SQL simultáneas en SQLFiddle sin afectar el rendimiento de la plataforma, asegurando que los usuarios puedan ejecutar sus códigos sin demoras significativas.
  - **Consumo de Recursos**: El sistema debe funcionar eficientemente en un entorno basado en la web, utilizando una cantidad moderada de CPU y memoria del servidor, con un máximo del 70% de uso de CPU y 50% de uso de RAM bajo carga máxima, para no sobrecargar SQLFiddle.
- **Requisitos Ambientales:**
  - **Seguridad y Protección de Datos**: Implementar en un entorno de servidor seguro con cifrado SSL para todas las comunicaciones con SQLFiddle, asegurando que los datos del código SQL sean tratados con confidencialidad y cumpliendo con normativas de privacidad de datos como el GDPR.
  - **Disponibilidad**: El sistema debe estar disponible 99.9% del tiempo, con soporte para recuperación automática en caso de fallos del servidor, garantizando que las evaluaciones de plagio se puedan realizar sin interrupciones mientras se utiliza SQLFiddle.
  - **Entorno de Ejecución**: El sistema será desplegado en un entorno cloud, optimizado para la ejecución en servidores que soportan la infraestructura de SQLFiddle, asegurando tiempos de respuesta rápidos y una alta disponibilidad.

4. Técnicas:

4.1 Detección Final:

- **Análisis Integral:** Realiza una evaluación exhaustiva del código al término del período de entrega, asegurando una revisión completa y justa de todas las presentaciones. Esto permite identificar cualquier forma de plagio sin interrumpir el flujo de trabajo del estudiante.
- **Preservación de la Productividad:** Evita la interrupción durante el proceso de desarrollo del estudiante, permitiendo que su creatividad y enfoque se mantengan sin distracciones por advertencias prematuras de plagio.
- **Implementación:**
  Realiza un análisis profundo y minucioso del código al final del período de entrega utilizando herramientas avanzadas de detección de similitudes y patrones sospechosos. Esta revisión exhaustiva garantiza la identificación precisa de cualquier plagio que haya podido ocurrir.

4.2 Comparación entre Entregas de Estudiantes:

- **Detección de plagio interno:** Es fundamental en entornos académicos donde el plagio entre compañeros es una práctica común. Este enfoque permite identificar si un estudiante ha copiado de otro, protegiendo la integridad académica.
- **Relevancia Académica:** Añade una capa adicional de seguridad al detectar plagio que no sería visible mediante la comparación con fuentes externas, proporcionando una evaluación más completa.
- **Implementación:** Desarrolla un sistema que compare sistemáticamente los códigos presentados por todos los estudiantes en el mismo curso o asignatura. Analiza las similitudes y patrones sospechosos entre las entregas para identificar casos de plagio interno de manera efectiva.

4.3 Comparación de Resultados de Ejecución:

- **Análisis Funcional:** En lugar de comparar únicamente el código fuente, este enfoque compara los resultados generados por la ejecución de los programas. Esto incluye la salida del programa, el tiempo de ejecución y el uso de recursos, ofreciendo una evaluación más completa del código presentado.
- **Evidencia Basada en Resultados:** Proporciona pruebas de plagio basadas en el comportamiento y el rendimiento del programa, no solo en la similitud del código fuente, lo que fortalece la detección de plagio.
- **Implementación:** Ejecuta los programas presentados contra un conjunto predefinido de casos de prueba, comparando los resultados de salida, tiempos de ejecución y uso de recursos. Registra estos datos y analiza las coincidencias para detectar posibles plagios funcionales.

4.4 Detección Basada en el Historial de Código:

- **Monitoreo de Actividades de Plagio:** Realiza un seguimiento detallado de las acciones del estudiante durante el desarrollo del código, identificando comportamientos que sugieren plagio, como el pegado de grandes bloques de código en intervalos de tiempo inusualmente cortos. Esto permite detectar la inserción de fragmentos de código copiados de fuentes externas.
- **Implementación:** Utiliza un sistema de control de versiones para registrar cada cambio en el código en tiempo real. Implementa herramientas que analizan estos cambios para identificar eventos sospechosos, como el pegado de más de 150 caracteres en menos de 2 segundos. Estas herramientas generan alertas automáticas que notifican a los docentes sobre posibles incidentes de plagio, permitiendo una revisión temprana y proactiva.

4.5 Rango de Calidad

- **Precisión de Detección**: El sistema debe mantener una precisión mínima del 90% en la detección de similitud de código SQL. Una vez superado este umbral, se enviará automáticamente una alerta de posible plagio, sin realizar análisis adicionales sobre modificaciones leves o refactorizaciones del código.
- **Simplicidad del Sistema**: Diseñar el sistema para una evaluación directa de similitudes, sin considerar complejas variaciones o alteraciones menores en el código, enfocándose en la detección rápida y eficiente de plagio.
- **Tolerancia a fallos**: El sistema debe poder manejar interrupciones menores en el servicio de SQLFiddle y recuperarse automáticamente, generando alertas en caso de errores críticos que requieran intervención manual.

5. Requisitos de Documentación:

5.1 Documentación Técnica: Proveer documentación exhaustiva para administradores del sistema, incluyendo guías de instalación, configuración y mantenimiento del sistema en un entorno cloud.

5.2 Registro de Cambios: Mantener un changelog actualizado que documente todas las actualizaciones y mejoras del sistema, con un enfoque en la optimización para SQLFiddle.

5.3 Roles en el equipo

ROL	ESTUDIANTE	DESCRIPCIÓN
Líder de Proyecto:	Pedro Humberto Rondon Ponce	Coordinar el equipo, definir los objetivos y plazos, supervisar el progreso del proyecto, garantizar la comunicación entre los miembros del equipo y con los stakeholders, y tomar decisiones clave en la gestión de recursos y riesgos.
Arquitecto de Software:	Carla Fernanda Ropa Calizaya Melany Yasmin Lazo Arana	Diseñar la arquitectura del sistema, definir los componentes clave, seleccionar las tecnologías y herramientas apropiadas, asegurar la escalabilidad y robustez del sistema, y guiar a los desarrolladores en la implementación técnica.
Programador Backend:	Roger Euclides Infa Sanchez	Desarrollar la lógica del servidor, implementar las APIs necesarias para la detección de plagio, integrar las bases de datos y servicios externos, y optimizar el rendimiento del sistema.
Programador Frontend:	Roger Euclides Infa Sanchez Carlos Daniel Corrales Yarasca	Desarrollar la interfaz de usuario del compilador SQL online, asegurando una experiencia de usuario intuitiva y eficiente, implementar las funcionalidades de interacción con el sistema de detección de plagio, y garantizar la compatibilidad con diferentes navegadores y dispositivos.
Revisor de Contenidos - pruebas:	Christian Omar Rodriguez H… Jose Carlos Peraltilla Nuñez	Diseñar y ejecutar pruebas para asegurar la calidad del sistema, identificar y documentar defectos, validar que el sistema cumple con los requisitos funcionales y no funcionales, y asegurar que el sistema sea robusto y fiable antes de su implementación.