UNIVERSIDAD PRIVADA DE TACNA

FACULTAD DE INGENIERIA

Escuela Profesional de Ingeniería de Sistemas

Informe Final

Proyecto *DocuCode-AI*

Curso: *PATRONES DE SOFTWARE*

Docente: *PATRICK JOSE CUADROS QUIROGA*

Integrantes:

Farley Rodrigo Eduardo Viveros Blanco - 2020066896
Ronal Daniel Lupaca Mamani - 202006146

Tacna – Perú
2025

CONTROL DE VERSIONES					
Versión	Hecha por	Revisada por	Aprobada por	Fecha	Motivo
1.0	MPV	ELV	ARV	3/04/2025	Versión Original
2.0	MPV	ELV	ARV	30/05/2020	mejora

INDICE GENERAL

1. Antecedentes

   En el contexto académico actual, los estudiantes y docentes de ingeniería de sistemas enfrentan dificultades crecientes para comprender, documentar y evaluar grandes volúmenes de código fuente. La ausencia de herramientas automatizadas que apoyen la interpretación estructurada del código afecta la calidad del aprendizaje, la eficiencia del desarrollo y la comprensión de buenas prácticas en programación. Las revisiones manuales de código suelen ser lentas, subjetivas y poco escalables, especialmente en cursos con numerosos participantes.

   En respuesta a estas necesidades, surge el proyecto DocuCode-AI, una plataforma web que utiliza Inteligencia Artificial para analizar código fuente, generar automáticamente documentación técnica, diagramas UML y evaluaciones de calidad. Este sistema busca apoyar tanto a docentes en sus procesos de revisión como a estudiantes en el entendimiento del diseño y arquitectura del software.

2. Planteamiento del Problema
   a. Problema
      Actualmente, los procesos de revisión y documentación del código fuente en entornos académicos son mayormente manuales, lo que genera demoras, errores de interpretación y dificultades para validar la calidad del software desarrollado por los estudiantes. No existen herramientas integradas que automaticen el análisis estructural del código ni que generen documentación técnica útil para el aprendizaje.
   b. Justificación
      Este proyecto es importante porque responde a una necesidad concreta del entorno educativo: facilitar la comprensión del código fuente mediante el uso de tecnologías emergentes. Al integrar algoritmos de IA y generación de UML, se proporciona una solución moderna y efectiva que optimiza el proceso de enseñanza-aprendizaje, fomenta buenas prácticas de codificación, y reduce el tiempo de evaluación. Su implementación permitirá a los docentes centrarse más en el contenido y menos en tareas repetitivas, mejorando así la calidad educativa.
   c. Alcance
      El sistema DocuCode-AI se implementará como una plataforma web accesible desde navegadores modernos. Permitirá la carga de archivos individuales o comprimidos (RAR/ZIP), su análisis automático, y la generación de reportes con comentarios de código, diagramas UML (clases, casos de uso, secuencia, actividad, componentes, paquetes, etc.), evaluación de calidad y detección de duplicados. El sistema está orientado al uso académico y no contempla funcionalidades de despliegue en producción empresarial.

3. Objetivos

Objetivo General

Diseñar e implementar un sistema web basado en inteligencia artificial que permita analizar código fuente, generar documentación técnica automática y evaluar su calidad, con fines educativos en el contexto universitario.

Objetivos Específicos

- Desarrollar un sistema web con arquitectura MVC que permita subir y procesar archivos de código fuente.
- Implementar una integración con la API de OpenAI para generar comentarios y descripciones automáticas del código.
- Generar diagramas UML automáticos a partir del análisis del código.
- Evaluar la calidad del código fuente utilizando criterios estructurales y sintácticos.
- Detectar posibles duplicaciones o similitudes entre bloques de código.
- Diseñar una interfaz amigable y accesible para usuarios docentes y estudiantes.
- Documentar el sistema, sus funcionalidades y arquitectura, integrando estándares académicos.

4. Marco Teórico

El avance de las tecnologías web e inteligencia artificial ha abierto nuevas posibilidades para la automatización de tareas complejas en la ingeniería de software. En el ámbito académico, uno de los desafíos recurrentes es la documentación estructurada del código fuente. El sistema DocuCode-AI responde a esta necesidad al permitir generar, de forma automática, diagramas UML y resúmenes analíticos que facilitan la comprensión y presentación de proyectos de software en informes técnicos y universitarios.

4.1. Sistemas Web y su Rol Académico

Un sistema web es una plataforma accesible mediante un navegador, desarrollada con tecnologías cliente-servidor, capaz de ofrecer servicios a múltiples usuarios. Sommerville (2020) indica que los sistemas web son ideales para entornos educativos, ya que centralizan el acceso, permiten escalabilidad y fomentan la colaboración. DocuCode-AI opera bajo esta premisa, brindando a estudiantes y docentes una herramienta en línea para analizar, interpretar y documentar proyectos de software de forma estructurada.

4.2. Patrón Arquitectónico MVC

El patrón Modelo-Vista-Controlador (MVC) organiza el código fuente en tres componentes independientes: el modelo (lógica de negocio), la vista (interfaz de usuario) y el controlador (gestión de eventos). Esta arquitectura, propuesta inicialmente por Krasner y Pope (1988), es ampliamente utilizada en aplicaciones educativas, ya que facilita la separación de responsabilidades, simplifica la mantenibilidad y favorece la reutilización. DocuCode-AI adopta

este patrón para organizar sus funcionalidades, especialmente en la gestión modular de los procesos de análisis y generación de diagramas.

4.3. Diagramas UML como Herramienta de Documentación
UML (Unified Modeling Language) es el estándar de modelado más utilizado para representar la estructura y el comportamiento de sistemas de software. Pressman y Maxim (2020) afirman que los diagramas UML permiten describir visualmente el diseño de un sistema, facilitando su entendimiento, validación y documentación. DocuCode-AI automatiza la creación de los principales diagramas utilizados en informes académicos, como los diagramas de clases, casos de uso, secuencia, actividad, componentes y paquetes, ayudando a los estudiantes a representar su arquitectura sin necesidad de crearlos manualmente.

4.4. Inteligencia Artificial Aplicada a la Ingeniería de Software
La inteligencia artificial ha demostrado ser una herramienta poderosa en el apoyo a tareas de ingeniería de software, desde la generación automática de código hasta la revisión de calidad. En el caso de DocuCode-AI, la IA se integra para interpretar archivos fuente y proponer representaciones gráficas que reflejan los patrones, estructuras y funcionalidades del sistema. Según Leiva Suero et al. (2020), este tipo de tecnologías, al ser aplicadas en contextos educativos, permite a los usuarios enfocarse más en la comprensión de conceptos que en la elaboración manual de diagramas.

4.5. Evaluación de Calidad y Código Duplicado
La calidad del código influye directamente en su mantenibilidad, rendimiento y fiabilidad. Martin (2009) argumenta que un código limpio y bien estructurado mejora la productividad y reduce la aparición de errores. DocuCode-AI incluye mecanismos de análisis de calidad que valoran aspectos como legibilidad, modularidad y estándares de estilo. Además, incorpora una función de detección de código duplicado (clones), lo que permite identificar malas prácticas comunes y mejorar la eficiencia en la programación.

5. Desarrollo de la Solución
    a. Análisis de Factibilidad (técnico, económica, operativa, social, legal, ambiental)
        a) Factibilidad Técnica
        El sistema DocuCode-AI se desarrolla utilizando tecnologías ampliamente conocidas y disponibles como PHP, JavaScript, PlantUML, HTML5 y bases de datos MySQL/MariaDB. La arquitectura del sistema se basa en el patrón MVC, facilitando la modularidad y mantenibilidad del proyecto. El análisis y generación

automática de diagramas se logra mediante la integración de bibliotecas IA como OpenAI y herramientas de interpretación de código. Se considera técnicamente viable al contar con los recursos y conocimientos necesarios para su implementación.

b) Factibilidad Económica
El sistema se implementa con herramientas de software libre, sin incurrir en costos de licencias. Los gastos se limitan a los servicios de hosting básico (en caso de despliegue público) y API tokens bajo demanda. El desarrollo académico se justifica económicamente debido al alto valor que representa la automatización en la elaboración de documentación técnica. En la evaluación beneficio/costo, se obtuvo un valor mayor a 1, siendo rentable.

c) Factibilidad Operativa
DocuCode-AI responde directamente a una necesidad de estudiantes y docentes en carreras de ingeniería, al facilitar la documentación de proyectos. Su uso es simple y no requiere conocimientos técnicos avanzados, gracias a su interfaz intuitiva y amigable. La aceptación por parte de los usuarios es alta debido a la reducción significativa en tiempo de elaboración de informes.

d) Factibilidad Social
La implementación del sistema impacta positivamente en el entorno académico, al mejorar la calidad de los proyectos documentados y reducir las barreras de entrada a tecnologías de documentación. Fomenta el aprendizaje asistido por herramientas inteligentes, promoviendo la autonomía y profesionalización del estudiante.

e) Factibilidad Legal
El sistema se desarrolla en cumplimiento con las normativas nacionales sobre software educativo y protección de datos personales. Al no almacenar información sensible del usuario ni publicar datos confidenciales, se mantiene dentro del marco legal vigente. Las herramientas utilizadas (PlantUML, OpenAI, PHP) respetan licencias de uso aceptables para proyectos académicos.

f) Factibilidad Ambiental
El sistema promueve la digitalización total del proceso de documentación de proyectos, eliminando la necesidad de impresión física de diagramas. Esto contribuye con la sostenibilidad y reducción del uso de papel en entornos universitarios.

b. Tecnología de Desarrollo

- **Backend**: PHP 8.x con arquitectura MVC
- **Frontend**: HTML5, CSS3, Bootstrap 5, JavaScript (AJAX)
- **Base de datos**: MySQL / MariaDB
- **Generación de diagramas**: PlantUML integrado mediante archivos .puml
- **Inteligencia Artificial**: OpenAI API para análisis semántico y generación asistida
- **Herramientas complementarias**: GitHub Actions (automatización), GitHub Pages (documentación), Composer, XAMPP

c. Metodología de implementación

Para el desarrollo del sistema DocuCode-AI se empleó la metodología ágil SCRUM, la cual permite una planificación iterativa y entrega incremental de funcionalidades. Se definieron cinco sprints principales, cada uno con objetivos específicos, responsables designados y entregables parciales.

Fase / Sprint	**Actividades Principales**	**Entregables**	**Responsable**
Fase 1	- Levantamiento de requisitos - Diseño del prototipo - Modelo entidad-relación	Boceto de interfaz Modelo de base de datos	Equipo de desarrollo
Fase 2	- Implementación de carga de archivos - Extracción de código fuente	Módulo de carga y lectura de archivos	Backend Developer
Fase 3	- Integración con PlantUML - Generación de diagramas UML - Visualización en UI	Diagramas: clase, secuencia, casos de uso, etc.	Fullstack Developer

Fase 4	- Evaluación de calidad con IA - Detección de duplicados	Módulo de IA y reporte de duplicación	IA Developer / Tester
Fase 5	- Documentación técnica - Despliegue en servidor - Revisión final	Documentos FD01–FD05 Wiki y Readme en GitHub	Líder del proyecto

d. (Documento de VISION, SRS, SAD)

El proyecto se acompaña de los siguientes entregables:

- Documento de Visión (FD02): Describe los objetivos, funcionalidades clave y posicionamiento del producto.
- SRS – Especificación de Requerimientos de Software (FD03): Contiene los requerimientos funcionales, no funcionales, reglas de negocio, perfiles de usuario y modelos UML.
- SAD – Documento de Arquitectura de Software (FD04): Detalla las vistas arquitectónicas (lógica, desarrollo, procesos, despliegue), diagramas técnicos y atributos de calidad.

6. Cronograma

El desarrollo del sistema **DocuCode-AI** se realizó en cinco fases distribuidas a lo largo del semestre 2025-I. Cada fase tuvo entregables específicos y responsables asignados, siguiendo una metodología ágil basada en sprints.

Fase	**Actividades principales**	**Fecha de inicio**	**Fecha de fin**	**Responsable**
Fase 1	Levantamiento de requisitos, diseño del prototipo, modelo entidad-relación	01/04/2025	07/04/2025	Todo el equipo
Fase 2	Implementación de carga de archivos, lectura de código fuente	08/04/2025	15/04/2025	Todo el equipo
Fase 3	Generación automática de diagramas UML con PlantUML	16/04/2025	23/04/2025	Todo el equipo
Fase 4	Integración IA (OpenAI), análisis de calidad y duplicados	24/04/2025	30/04/2025	Todo el equipo
Fase 5	Documentación técnica, despliegue en VPS, entrega final (FD01–FD05)	01/05/2025	06/07/2025	Líder del proyecto

7. Presupuesto

El sistema se desarrolló utilizando tecnologías libres y servicios con bajo costo, logrando un equilibrio entre funcionalidad y viabilidad económica.

Concepto	**Detalle**	**Costo estimado**
Servidor VPS	Elastika (2 vCPU, 4GB RAM, 50GB SSD) – anual	S/ 100.00
Dominio web + SSL	Hostinger	S/ 12.00
Consumo de API OpenAI	GPT-4, uso estimado académico (mensualmente)	S/ 20.00
Herramientas de desarrollo	VS Code, GitHub, Composer (gratuitos)	S/ 0.00
Total estimado anual		**S/ 132.00**

8. Conclusiones
    - El sistema DocuCode-AI cumple con los objetivos planteados al inicio del proyecto, ofreciendo una solución eficaz para la generación automática de documentación técnica y evaluación de código.
    - Se logró implementar funcionalidades clave como análisis semántico, detección de duplicados y generación de diagramas UML sin intervención manual.
    - La arquitectura modular basada en MVC facilitó la organización del sistema y su mantenimiento.
    - La integración de herramientas modernas como OpenAI, PlantUML y Terraform permitió explorar tecnologías de vanguardia en un contexto académico real.
    - El proyecto fue desarrollado exitosamente dentro del tiempo establecido, con roles bien definidos y cumpliendo con los entregables técnicos y documentales.

9. Recomendaciones
    - Incluir soporte para más lenguajes de programación (ej. C++, C#).
    - Optimizar el rendimiento de análisis con procesamiento asíncrono y colas de tareas.
    - Desplegar la plataforma en ambientes institucionales o integrar con LMS como Moodle.
    - Implementar métricas más avanzadas de calidad y estilo de código.
    - Añadir una opción para generación de reportes ejecutivos o académicos en Word y PDF.
    - Continuar utilizando metodologías ágiles para mantener iteraciones de mejora continua.

10. Bibliografía

- Pressman, R. & Maxim, B. (2020). *Ingeniería de Software: Un Enfoque Práctico*. McGraw-Hill.
- Sommerville, I. (2020). *Software Engineering*. Pearson.
- Martin, R. C. (2009). *Clean Code: A Handbook of Agile Software Craftsmanship*. Prentice Hall.
- Krasner, G. E. & Pope, S. T. (1988). *A Description of the Model-View-Controller User Interface Paradigm in the Smalltalk-80 System*.
- Leiva Suero, F. et al. (2020). *Aplicación de inteligencia artificial para el análisis de código fuente*.
- Fowler, M. (2002). *Patterns of Enterprise Application Architecture*.
- Documentación oficial:
    - https://platform.openai.com/docs
    - https://plantuml.com
    - https://www.terraform.io
    - https://marp.app

11. Anexos

Repositorio : https://github.com/UPT-FAING-EPIS/proyecto-si889-2025-i-u3-docucode-ai-1

Anexo 01 Informe de Factiblidad

Anex0 02 Documento de Visión

Anexo 03 Documento SRS

Anexo 04 Documento SAD