Generador Automático de Documentación para Proyectos Java

1. Objetivo

El objetivo del proyecto es desarrollar una aplicación capaz de analizar un proyecto Java y generar documentación técnica automáticamente. El sistema debe extraer la estructura del código, producir diagramas UML mediante PlantUML, generar documentación en Markdown y convertirla a PDF. La aplicación se ejecutará en Docker, incluirá un backend en Node.js, un frontend en React con TailwindCSS y empleará un modelo de IA (como LMStudio) para enriquecer las descripciones técnicas. La interfaz debe permitir seleccionar el proyecto Java, iniciar la generación y visualizar o descargar los resultados.

2. Frontend (Requisitos técnicos)

El frontend debe estar implementado en React con TailwindCSS y debe incluir:

1. Una vista principal que permita seleccionar o subir el proyecto Java.
2. Un botón de generación que dispare una petición al backend.
3. Gestión de estados mediante useState para: proyecto seleccionado, estado del proceso, resultados y errores.
4. Manejo de eventos: onClick, onChange y onSubmit.
5. Comunicación entre componentes mediante props (lifting state up).
6. Uso obligatorio de useEffect para cargar historial o reaccionar a cambios de estado.
7. Renderizado condicional para mostrar estados de carga, error o éxito.
8. Una vista de resultado que renderice el Markdown generado y permita descargar el PDF.
9. Una vista de historial con ejecuciones previas.
10. Estructura limpia de componentes: selección, carga, resultado, historial.

3. Backend (Requisitos técnicos)

El backend debe estar implementado en Node.js y exponer, como mínimo, los siguientes endpoints:

POST /api/analyze: recibe el proyecto Java, lo analiza y genera la documentación.

GET /api/history: devuelve el historial de ejecuciones anteriores.

GET /api/docs/id devuelve los archivos generados (PDF o MD).

El backend debe:

1. Analizar el código Java (parseo básico o modularizado en carpetas analyzers/).

2. Generar archivos PlantUML (.puml) en generators/.

3. Integrarse con un modelo de IA encapsulado en services/.

4. Generar Markdown y convertirlo a PDF.

5. Gestionar un historial de ejecuciones.

4. Tecnologías mínimas

- Node.js (backend)
- React + TailwindCSS (frontend) Puerto 8978 FRONT
- Docker (despliegue obligatorio)
- PowerShell (Setup.ps1 para automatizar el entorno)
- PlantUML (diagramas UML)
- LMStudio u otro modelo local para enriquecer la documentación
- Conversión Markdown → PDF mediante herramienta adecuada (por ejemplo, Pandoc)

5. Mejoras a documentar

Cada grupo o alumno debe documentar mejoras adicionales implementadas, tales como:

1. Estadísticas del proyecto analizado.
2. Filtros por paquete o clase.
3. Comparación de versiones del mismo proyecto.
4. Soporte parcial a otros lenguajes.
5. Mejoras en accesibilidad o estructura visual del frontend.
6. Recomendaciones generadas por IA sobre estructura o diseño del código.

Todo orientado a la documentación en la Rúbrica:

Ejercicio 1	No se realiza el archivo Markdown. **0 puntos**	Se Genera correctamente el archivo Markdown **3 punto**
Ejercicio 2	No se generan diagramads UML utilizando PlantUML. **0 puntos**	Se generan diagramas UML utilizando PlantUML **1.5 punto**
Ejercicio 3	No se genera y descarga el archivo PDF. **0 puntos**	Se genera y descarga del archivo PDF **1 puntos**
Ejercicio 4	No funciona correctamente ell frontend React, incluyendo estados, eventos, efectos y comunicación entre componentes. **0 puntos**	Se funciona el frontend React, incluyendo estados, eventos, efectos y comunicación entre componentes **1.5 puntos**
Ejercicio 5	No se tiene un despliegue funcional mediante Docker y script Setup.ps1. **0 puntos**	Se tiene un despliegue funcional mediante Docker y script Setup.ps1 **2 puntos**
Ejercicio 6	No se integra con IA y claridad del código. **0 puntos**	Se integra con IA y claridad del código **1 puntos**