Generador Automático de Documentación para Proyectos Java

1. Objetivo

El objetivo de este Sprint 7 es implementar, integrar y asegurar la funcionalidad completa de la documentación generada y el sistema de soporte desarrollado en el Sprint 6, enfocándose en la gestión de resultados, la usabilidad del frontend y el despliegue funcional de la aplicación. Esto implica garantizar que el sistema pueda no solo generar la documentación (Markdown, PlantUML, PDF) sino también almacenarla, presentarla al usuario y permitir su descarga

2. Tareas Clave

Este sprint se centra en finalizar la integración de los artefactos generados (Markdown, PDF) en las interfaces de usuario y en la infraestructura de despliegue.

Área	Requisito de Implementación	Base en T1 Sprint 6
Frontend: Gestión de Resultados	Implementación y verificación de la **vista de resultado** que renderice el archivo **Markdown generado**.	El sistema debe generar documentación en Markdown.
Frontend: Descarga	Asegurar la funcionalidad para **descargar el archivo PDF** generado desde la vista de resultados.	El sistema debe convertir Markdown a PDF y permitir la descarga.
Frontend: Historial	Implementación completa de la vista de historial que muestre las **ejecuciones previas**.	El backend debe gestionar y el frontend debe mostrar el historial de ejecuciones.
Backend: APIs de Documentación	Garantizar que el *endpoint* `GET /api/docs/id` devuelva correctamente los archivos generados (PDF o MD).	Se requiere un *endpoint* específico para acceder a los resultados.
Despliegue y Entorno	Puesta a punto y validación del **despliegue funcional mediante Docker**. El *script* `Setup.ps1` debe automatizar correctamente el entorno.	El despliegue funcional es un requisito obligatorio.

3. Requisitos Transversales y de Integración

Se debe confirmar la correcta integración de todos los componentes esenciales definidos en el Sprint 6.

- **Flujo Completo de Generación:** Se debe confirmar que el proceso de análisis de código Java, generación de archivos PlantUML (.puml) en `generators/`, generación de Markdown y conversión a PDF es exitoso y estable.
- **Integración de IA:** Se debe verificar la funcionalidad del modelo de IA (LMStudio u otro modelo local) encapsulado en `services/` para enriquecer las descripciones técnicas.
- **Funcionalidad del Frontend (React):** Confirmar que el frontend funcione correctamente, incluyendo la gestión de estados (`useState`), manejo de eventos, uso de `useEffect` y comunicación entre componentes mediante *props*. El frontend debe operar en el **Puerto 8978 FRONT**.

4. Tecnologías Clave a Validar

Para la correcta implementación de la documentación y el despliegue del sistema, se validarán las siguientes tecnologías mínimas:

- Node.js (Backend)
- React + TailwindCSS (Frontend)
- Docker y PowerShell (para `Setup.ps1`)
- PlantUML (Generación de diagramas UML)
- Herramienta de conversión Markdown → PDF (ej. Pandoc)

5. Criterios de Éxito

Los siguientes puntos representan la culminación exitosa de la implementación del sistema:

Criterio	Puntuación Máxima	Condición de Éxito
Generación y Descarga de PDF	**1 punto**	Se genera y se permite la descarga del archivo PDF.
Frontend Funcional	**1.5 puntos**	El frontend React funciona correctamente, incluyendo estados, eventos, efectos y comunicación entre componentes.
Despliegue Funcional	**2 puntos**	Se tiene un despliegue funcional mediante Docker y script `Setup.ps1`.
Integración con IA y Claridad	**1 punto**	Se integra con IA y el código cumple con la claridad requerida.