SDD - DESARROLLO BASADO EN ESPECIFICACIONES

Análisis comparativo de herramientas para desarrollo de software asistido por IA: especificaciones frente a configuración de contexto

Autor: Mario Tomé Core

Ciclo Formativo de Grado Superior: Desarrollo de Aplicaciones Web (DAW)

Módulo: Inteligencia Artificial

Fecha: Enero de 2026

Índice de contenidos

1. Introducción

La llegada de la Inteligencia Artificial generativa en el desarrollo de software ha supuesto un **gran cambio** en la forma en que se diseñan, implementan y mantienen las aplicaciones.

Los modelos de lenguaje de gran tamaño (LLM) permiten generar código, documentación y pruebas de manera automática, incrementando notablemente la **productividad de los desarrolladores**.

Sin embargo, el uso inicial de estas tecnologías se ha basado mayoritariamente en interacciones informales mediante prompts en lenguaje natural. Aunque este enfoque resulta eficaz en tareas puntuales, presenta **limitaciones** cuando se aplica a proyectos reales que requieren **mantenibilidad**, **coherencia arquitectónica** y **control a largo plazo**.

Como respuesta a estos problemas, han surgido nuevas metodologías que buscan estructurar la colaboración entre humanos y agentes de IA. Entre ellas destacan el **desarrollo basado en especificaciones (Specification-Driven Development, SDD)** y la **configuración explícita del contexto de los agentes**, dos enfoques que buscan mejorar la calidad y fiabilidad del software generado por IA.

En este documento voy a analizar el estado actual de estas metodologías, comparar diferentes herramientas representativas y aplicar de forma práctica uno de estos enfoques en el desarrollo de una **API REST** funcional.

2. Investigación teórica

2.1 Estado del arte del desarrollo asistido por IA

Existencia de estándares o metodologías consolidadas

En la actualidad **no existe un estándar único y consolidado** para trabajar con agentes de IA en el desarrollo de software basado en especificaciones. A diferencia de metodologías tradicionales como UML, TDD o Scrum, el desarrollo asistido por IA se encuentra aún en una **fase de evolución**.

No obstante, están surgiendo **convenciones de facto** impulsadas tanto por grandes empresas tecnológicas como por la comunidad open source. Estas prácticas buscan aportar estructura, reproducibilidad y control a procesos que, de otro modo, serían muy impredecibles.

Enfoques emergentes

Pueden identificarse 3 enfoques principales:

1. Desarrollo dirigido por especificaciones (SDD)
Las especificaciones se convierten en la fuente de verdad del proyecto. El código es un artefacto derivado que debe cumplir estrictamente dichas especificaciones. Este enfoque prioriza la trazabilidad y la reducción de deuda técnica.

2. Programación basada en configuración de contexto
El agente de IA recibe archivos de contexto en lenguaje natural (Markdown) que definen reglas de estilo, arquitectura y restricciones técnicas. Es un enfoque flexible y de rápida adopción, aunque menos riguroso.

3. Sistemas colaborativos multi-agente
Se utilizan varios agentes especializados (arquitecto, desarrollador, tester) que colaboran entre sí. Aunque prometedor, su uso en entornos productivos todavía es limitado y experimental.

Principales empresas y organizaciones impulsoras

Este ecosistema está siendo impulsado por empresas como **GitHub (Microsoft)**, **AWS**, **Google** y **Anthropic**, junto a startups especializadas como **Tessl**, **Cursor** o **Fission AI**.

Paralelamente, la comunidad open source desempeña un papel clave en la experimentación y adopción temprana de estas tecnologías.

2.2 Taxonomía de herramientas

Las herramientas analizadas pueden clasificarse en 2 grandes categorías:

Herramientas de especificación formal

Gestionan el ciclo completo del desarrollo: requisitos → diseño → tareas → código

Buscan aportar estructura, trazabilidad y consistencia. Ejemplos representativos son **Kiro**, **Spec-kit**, **OpenSpec** y **Tessl**.

Herramientas de configuración de contexto

Proporcionan instrucciones y restricciones al agente de IA mediante archivos de texto, sin imponer un flujo completo de desarrollo. Algunos ejemplos son **AGENTS.md**, **CLAUDE.md**, **GEMINI.md** y **.cursorrules**.

Ambas categorías no son excluyentes, sino **complementarias**: las herramientas de especificación definen *qué* se debe construir, mientras que las de contexto definen *cómo* debe construirse.

En términos de adopción, la configuración de contexto es actualmente la más extendida debido a su simplicidad, mientras que el desarrollo basado en especificaciones presenta una mayor madurez conceptual, pero una adopción más limitada.

3. Análisis de herramientas de especificación

3.1 Spec-kit (GitHub)

Es una herramienta de línea de comandos open source desarrollada por GitHub cuyo objetivo es e**standarizar la creación de especificaciones de software**. Promueve una separación clara entre especificación e implementación, fomentando buenas prácticas de diseño.

Permite definir **requisitos**, **generar tareas** y **validar** que el código cumple con la especificación. Es compatible con múltiples agentes de IA y no impone dependencia con un proveedor concreto.

Entre sus ventajas destacan su transparencia, su integración con Git y su enfoque didáctico. Como limitación, requiere una mayor disciplina inicial y puede resultar excesivo para proyectos muy pequeños.

3.2 OpenSpec (Fission AI)

Es una herramienta ligera orientada a proyectos existentes que busca introducir el desarrollo basado en especificaciones sin imponer una estructura rígida ni dependencia de proveedores externos.

No requiere claves de API y su adopción es sencilla, lo que la convierte en una opción adecuada para equipos que desean experimentar con SDD sin alterar radicalmente su flujo de trabajo.

3.3 Tessl

Tessl representa un enfoque radical conocido como **“spec-as-source”**, en el que el código fuente deja de versionarse y se genera automáticamente a partir de las especificaciones durante el proceso de construcción.

Aunque promete una reducción significativa de la deuda técnica y una coherencia total del código, introduce una gran dependencia de la herramienta y una curva de aprendizaje elevada. Su ecosistema aún se encuentra en una fase temprana.

4. Comparativa entre especificaciones y configuración de contexto

El desarrollo basado en especificaciones ofrece una mayor estructura y trazabilidad, especialmente en proyectos complejos o a largo plazo.

Por el contrario, la configuración de contexto destaca por su rapidez y facilidad de adopción.

En la práctica profesional, ambos enfoques deben considerarse **complementarios**, combinando la robustez de las especificaciones con la flexibilidad del contexto.

5. Parte práctica: aplicación de SDD a un mini-proyecto

5.1 Objetivo de la práctica

El objetivo de esta parte práctica es aplicar una herramienta de desarrollo basado en especificaciones (SDD) a un mini-proyecto real, demostrando cómo las especificaciones guían de forma directa el diseño y la implementación del software.

Para ello, he desarrollado una **API REST sencilla** utilizando **Python y Flask**, siguiendo el flujo:

Requisitos → Diseño → Tareas → Implementación

Este enfoque está alineado con las herramientas analizadas en la parte teórica, especialmente **Spec-kit**, que propone un flujo estructurado y agnóstico del agente de IA.

5.2 Descripción del mini-proyecto

El proyecto desarrollado se denomina **SpecTool Comparator API**.

Se trata de una API REST cuyo propósito es gestionar y comparar herramientas de desarrollo basado en especificaciones asistido por IA.

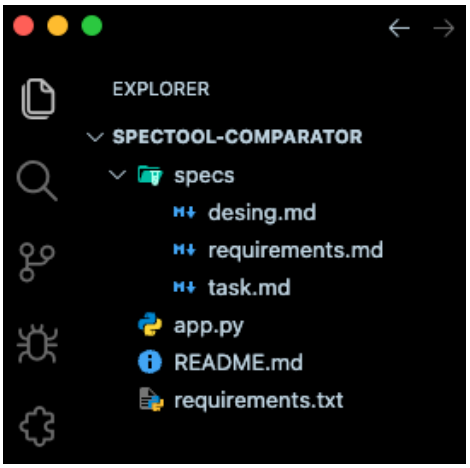
La aplicación permite:

- Listar herramientas disponibles
- Consultar información detallada de una herramienta
- Añadir nuevas herramientas
- Comparar dos herramientas entre sí

No se ha utilizado base de datos persistente, ya que el objetivo principal es demostrar el proceso de especificación y no la infraestructura.

5.3 Estructura del proyecto

La estructura del proyecto es la siguiente:



La carpeta **specs/** contiene los **ficheros de especificación**, que actúan como fuente de verdad del proyecto.

5.4 Especificación de requisitos

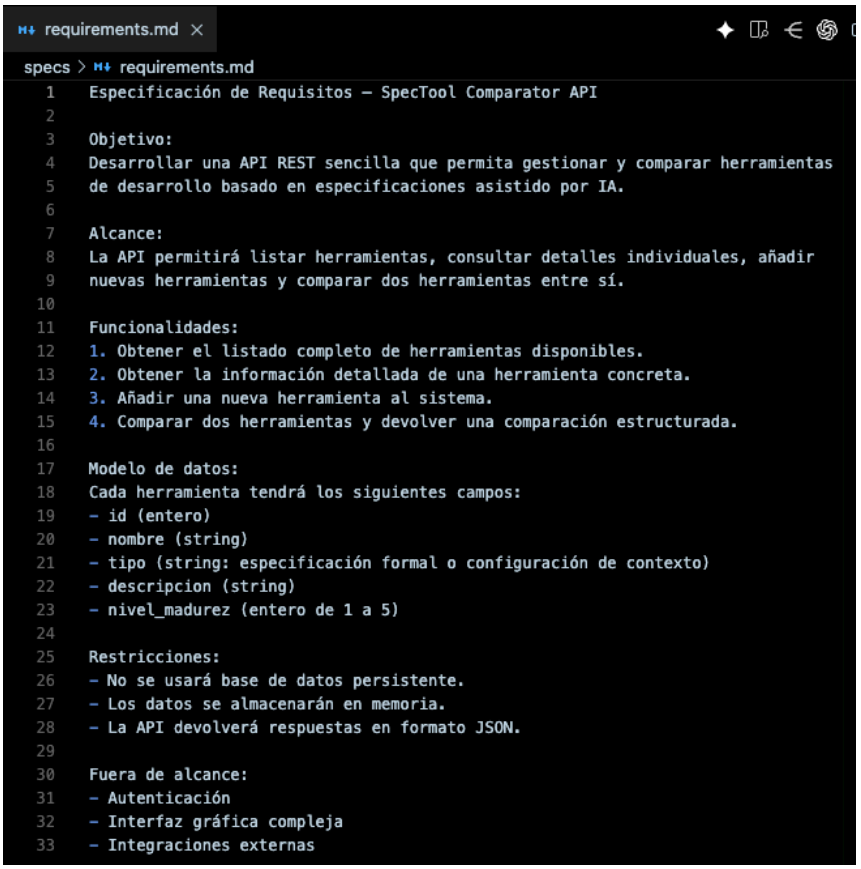
El primer paso ha sido definir los requisitos funcionales del sistema en el archivo:

specs/requirements.md

En este documento se describen:

- El objetivo de la aplicación
- El alcance funcional
- Las funcionalidades principales
- El modelo de datos
- Las restricciones del sistema

Este documento sirve como base para todo el desarrollo posterior.



5.5 Diseño de la solución

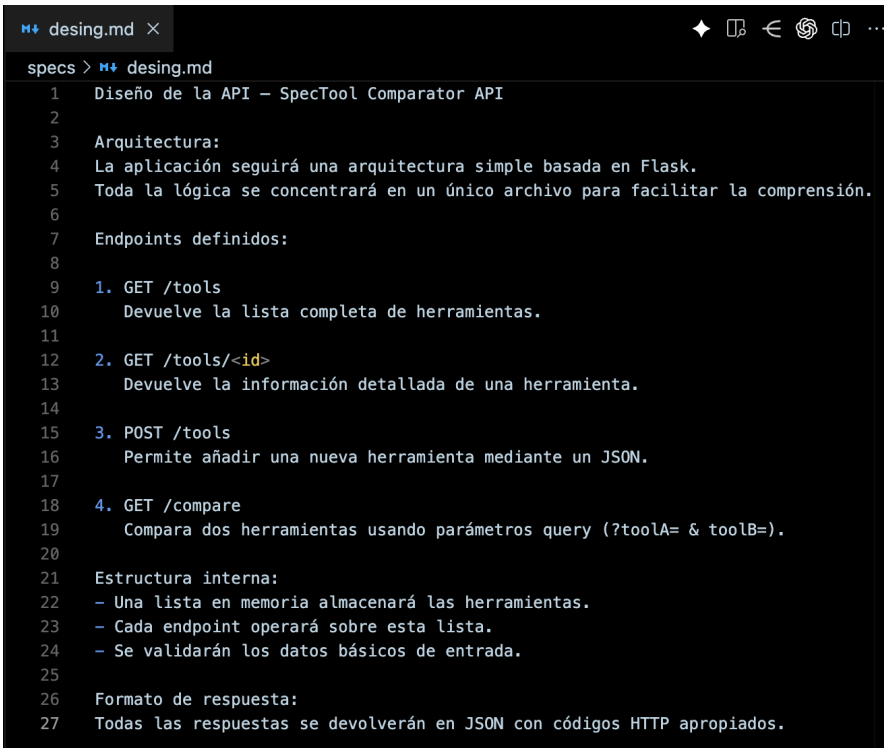
A partir de los requisitos, se ha definido el diseño de la API en el archivo:

specs/design.md

En este documento se especifican:

- La arquitectura general de la aplicación
- Los endpoints de la API
- El tipo de peticiones HTTP
- El formato de las respuestas

Este paso permite traducir los requisitos a una solución técnica concreta antes de programar.



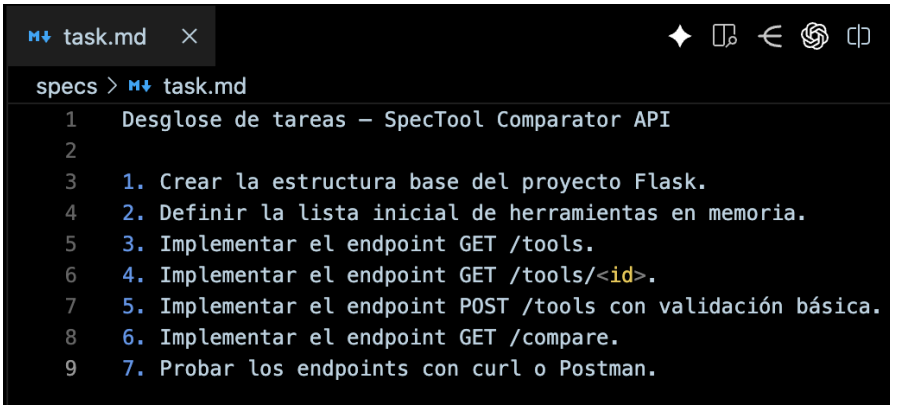
5.6 Desglose de tareas

Una vez definido el diseño, se ha realizado el desglose de tareas en el archivo:

specs/tasks.md

En este documento se enumeran las tareas necesarias para implementar la aplicación, siguiendo un enfoque incremental y ordenado.

Este enfoque refleja el flujo propuesto por herramientas como Spec-kit, donde las tareas se derivan directamente del diseño.



5.7 Implementación de la API REST

La implementación se ha realizado en Python utilizando el framework Flask.

1. Instalación de dependencias

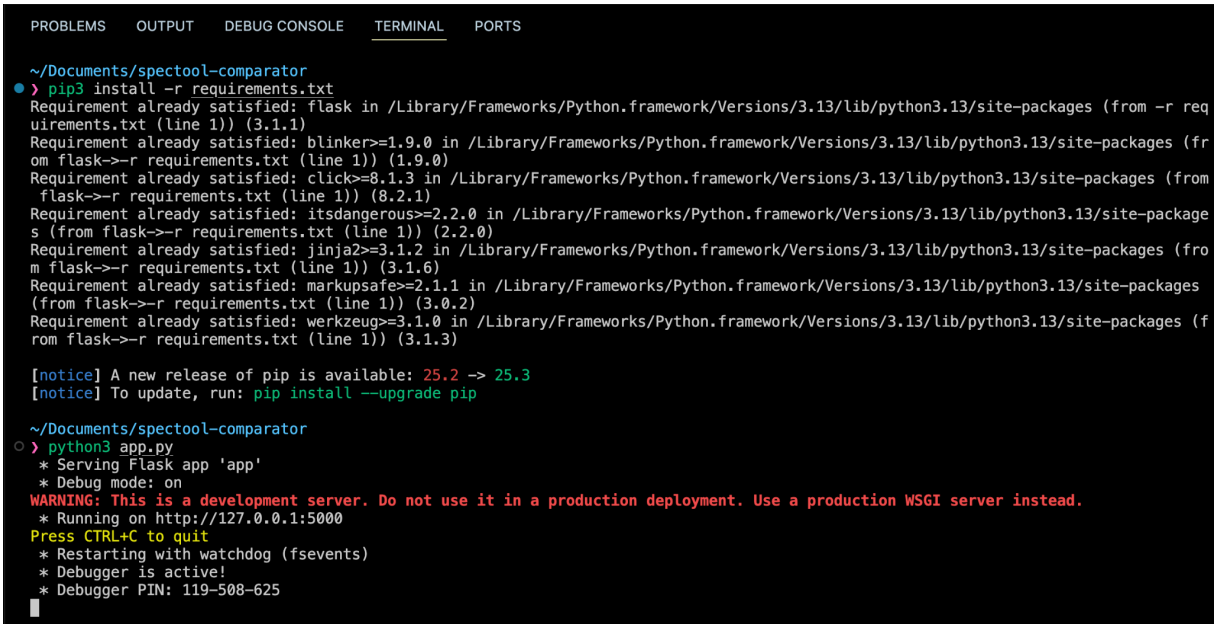
En primer lugar, se deben instalar las dependencias del proyecto:

`pip3 install -r requirements.txt`

2. Ejecución de la aplicación

Para ejecutar la API, se utiliza el siguiente comando: **`python3 app.py`**

Al ejecutarse correctamente, Flask inicia el servidor de desarrollo en la dirección: **`http://127.0.0.1:5000`**

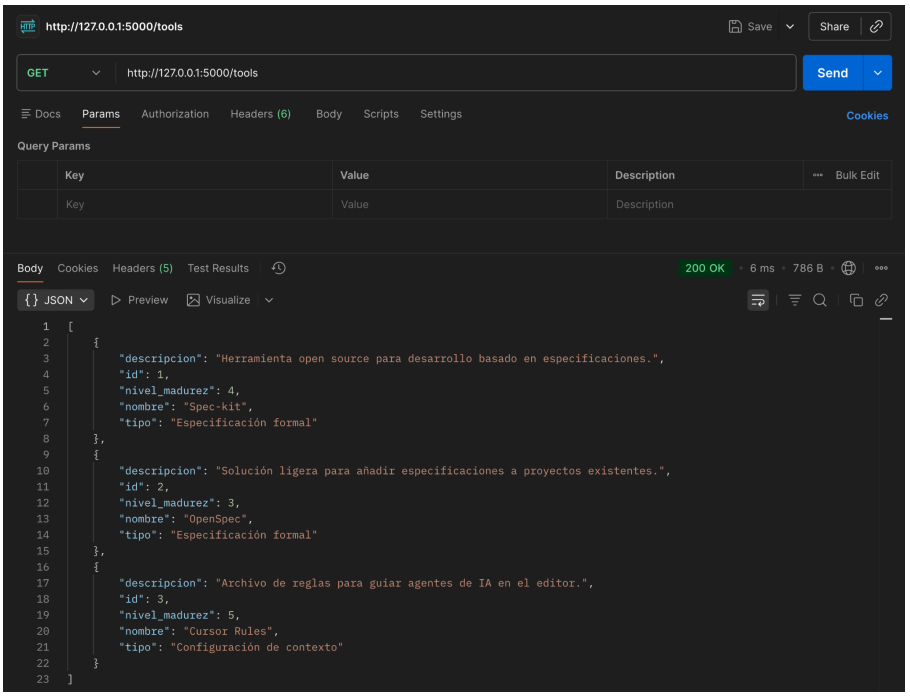


5.8 Pruebas de los endpoints

Las pruebas de la API se han realizado utilizando **Postman**, una herramienta estándar en el desarrollo de APIs REST.

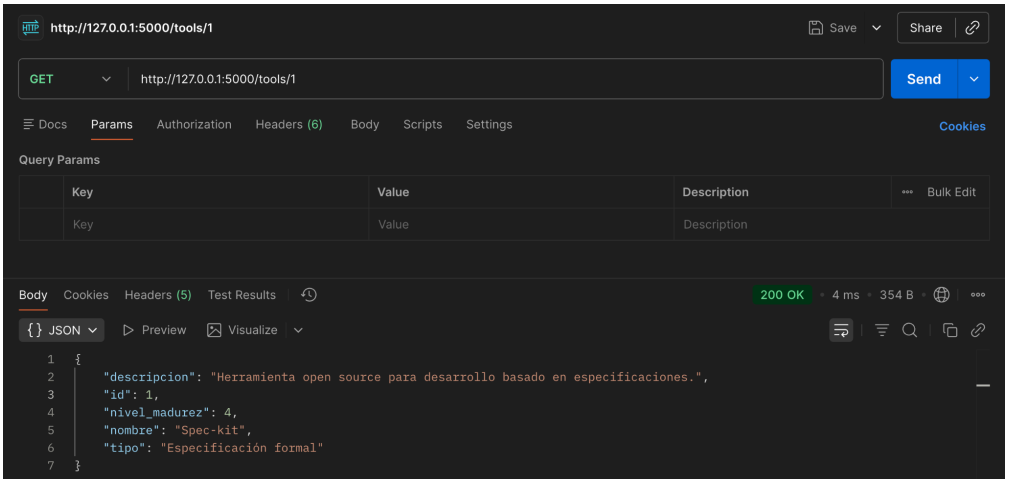
1. Endpoint: Obtener todas las herramientas

- Método: GET
- URL: `http://127.0.0.1:5000/tools`
- Este endpoint devuelve el listado completo de herramientas en formato JSON.



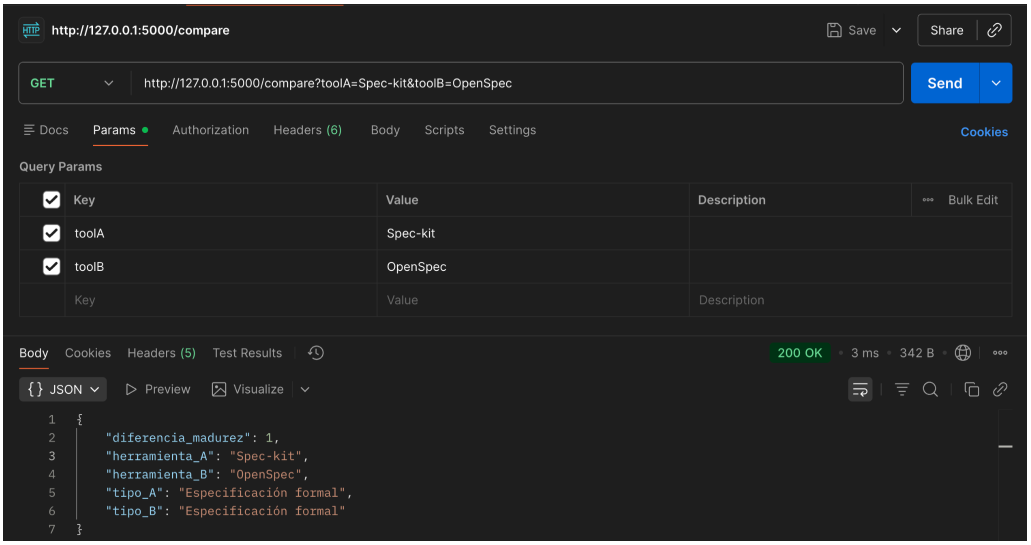
2. Endpoint: Obtener una herramienta por ID

- Método: GET
- URL: `http://127.0.0.1:5000/tools/1`
- Este endpoint devuelve la información detallada de una herramienta concreta.



3. Endpoint: Comparar dos herramientas

- Método: GET
- URL: **http://127.0.0.1:5000/compare?toolA=Spec-kit&toolB=OpenSpec**
- Este endpoint devuelve una comparación estructurada entre dos herramientas.



5.9 Relación con el desarrollo basado en especificaciones

Este mini-proyecto demuestra cómo el desarrollo basado en especificaciones permite:

- Reducir ambigüedades antes de programar
- Guiar el diseño de la API
- Facilitar la implementación
- Mantener coherencia entre requisitos y código

Los artefactos generados (requirements.md, design.md, tasks.md) cumplen el mismo rol que las herramientas modernas de especificación asistida por IA, actuando como intermediarios entre la intención del desarrollador y la implementación final.

5.10 Conclusión de la parte práctica

La aplicación desarrollada cumple con los requisitos del enunciado, demostrando de forma práctica la aplicación de una herramienta de desarrollo basado en especificaciones a un proyecto real.

El uso de Flask y Postman permite centrarse en el proceso SDD, que es el objetivo principal de esta práctica, dejando la interfaz gráfica fuera de alcance.

6. Conclusión

El desarrollo asistido por Inteligencia Artificial está evolucionando rápidamente. Aunque los enfoques basados únicamente en prompts resultan útiles, presentan limitaciones claras en proyectos reales.

El desarrollo basado en especificaciones ofrece una alternativa más estructurada y profesional, capaz de mejorar la calidad y la trazabilidad del software.

A través de la práctica realizada, se demuestra su aplicabilidad real y su potencial en entornos profesionales.