Plan de Proyecto

API de Comparación de Productos - Backend

|  |  |
| --- | --- |
| **Proyecto** | API RESTful para comparación de productos |
| **Tipo** | Proof of Concept (POC) - Backend Only |
| **Enfoque** | Generación asistida con IA + Desarrollo manual |
| **Herramientas** | Claude Code (generación), Cursor IDE (desarrollo) |

# 1. Objetivos del Proyecto

Desarrollar un API backend que demuestre capacidad de diseño arquitectónico y toma de decisiones técnicas fundamentadas, priorizando claridad y extensibilidad sobre complejidad prematura.

## Alcance

* **Backend only:** No frontend, HTML, CSS o JavaScript
* **POC extensible:** Diseño que permita evolución sin reescritura
* **Arquitectura hexagonal:** Separación clara de capas y responsabilidades
* **Documentación técnica:** Justificación de decisiones arquitectónicas

# 2. Estrategia de Desarrollo

## Enfoque Híbrido: IA + Desarrollo Manual

Se utilizó **Claude Code** para generar el scaffolding inicial del proyecto y estructura base, permitiendo una iteración rápida del POC. El desarrollo y refinamiento posterior se realizó en **Cursor IDE**, aprovechando sus capacidades de asistencia con IA para optimizar el código.

**Ventajas de este enfoque:**

* Scaffolding rápido de estructura arquitectónica
* Reducción de boilerplate y código repetitivo
* Focus en decisiones de diseño vs. implementación mecánica
* Iteraciones rápidas para validar arquitectura

## Principios Rectores

* **YAGNI (You Aren't Gonna Need It):** Evitar features y complejidad innecesaria
* **Pragmatismo sobre perfeccionismo:** POC funcional, no sistema completo
* **Documentar el 'por qué':** Justificar decisiones técnicas
* **Claridad > Cleverness:** Código legible sobre optimizaciones prematuras

# 3. Decisiones Arquitectónicas Clave

## UUID como Identificador

Usar UUID v4 para IDs de productos garantiza integración seamless sin colisiones entre sistemas, previene enumeración de catálogo, y facilita escalabilidad distribuida futura.

## Repository Pattern

Abstracción completa de la capa de persistencia permite reemplazar la fuente de datos (JSON → PostgreSQL/MongoDB/API) sin modificar lógica de negocio.

## Compromiso POC vs. Producción

**Incluido:** Manejo de errores, validaciones, logging, type hints. **Pospuesto:** Rate limiting, autenticación, cache distribuido (documentado como próximos pasos con justificación).

# 4. Documentación como Entregable Principal

La documentación no es un afterthought sino un componente crítico del proyecto. El README debe explicar el diseño del API, decisiones arquitectónicas, setup, y estrategia técnica, sirviendo como blueprint para evaluación y futura extensión.

## Contenido Obligatorio del README

* Decisiones de arquitectura con justificaciones
* Stack tecnológico con razones de elección
* Setup completo paso a paso
* Explicación de lo que NO se implementó y por qué
* Endpoints principales con ejemplos de uso
* Próximos pasos y roadmap de producción

# 5. Roadmap y Herramientas

**Flujo de trabajo:** Análisis → Scaffolding (Claude Code) → Desarrollo (Cursor IDE) → Validación → Documentación técnica.

# 6. Criterios de Éxito

* API funcional con arquitectura hexagonal claramente implementada
* Repository pattern abstrae correctamente la persistencia
* Código cumple PEP 8 con type hints completos
* README documenta decisiones arquitectónicas con justificaciones
* Endpoints retornan respuestas JSON correctamente estructuradas
* Manejo de errores robusto con status codes apropiados
* Aplicación de YAGNI - sin over-engineering