# Documento de Arquitectura de Software: Podcastify

## 1. Requisitos Arquitectónicamente Significativos (ASR)

Los ASR son aquellos requisitos que moldean la estructura del sistema:

- RF-1 (Ingestión Masiva): Capacidad para leer y persistir miles de registros CSV.
- RF-7 (Lógica de Negocio Compleja): El cálculo de umbrales por usuario único requiere una gestión eficiente del estado en memoria o base de datos.
- RFN-1 (Interoperabilidad): Obligación de exponer una API OpenAPI.

Restricción Tecnológica: El sistema debe ser desplegable en Docker de forma automática.

## 2. Atributos de Calidad Prioritarios

Mantenibilidad (Alta): El sistema es un piloto "vivo"; debe ser fácil añadir nuevos requisitos semanales.

Interoperabilidad (Alta): Debe comunicarse con clientes externos vía API REST y herramientas de visualización (Superset/Kibana).

Portabilidad: Debe funcionar en cualquier entorno mediante contenedores.

Testabilidad: Prioridad absoluta en DevOps para asegurar que los cambios no rompen la lógica (RF-2 al RF-5).

## 3. Selección de la Arquitectura

Modelo Elegido: Arquitectura Hexagonal (Puertos y Adaptadores).

¿Por qué esta arquitectura?

Permite aislar la lógica de negocio (el cálculo de descargas y recomendaciones) de las tecnologías externas (la base de datos que elijamos o el framework de API). Si decidimos cambiar de SQLite a MongoDB, solo cambiamos un "adaptador".

Alternativas Consideradas

Monolito por capas tradicional: Más simple, pero acopla demasiado la lógica a la base de datos. Se descartó por la naturaleza cambiante del proyecto.

Microservicios: Demasiado complejo para un piloto individual en esta etapa inicial.

Trade-offs (Compromisos)

Complejidad inicial: Requiere más archivos y carpetas (boilterplate) que un script sencillo.

Curva de aprendizaje: Implementar Inversión de Dependencias (DI) es más costoso al principio pero ahorra tiempo en el futuro.

**4. Organización y Dependencias**

El sistema se organiza en:

- Domain (Núcleo): Entidades (Podcast, Episodio) y Reglas de Negocio. No depende de nada.
- Application (Casos de Uso): Procesar CSV, Calcular Ranking. Depende del Dominio.
- Infrastructure (Adaptadores): Implementación de la BD, API REST, Cliente de archivos CSV. Depende de las interfaces de Aplicación.

**5. Implementación y Métricas**

Implementación: Python + FastAPI (para OpenAPI) + SQLAlchemy/Motor NoSQL.

Métricas de Arquitectura: * Acoplamiento: Medido por herramientas como Radon o SonarQube.

Cobertura de test: Objetivo > 80%.

Tiempo de respuesta de la API: Monitorizado en el pipeline.

**6. Documentación**

La arquitectura se documenta mediante:

Este documento en /docs/architecture/.

Diagramas C4 (Contexto y Contenedores).

ADRs (Architecture Decision Records) en /docs/adr/.