

Descripción del Modelo C4 - Sistema Podcastify

El modelo C4 se utiliza para describir la arquitectura de Podcastify en diferentes niveles de abstracción.

Nivel 1: Diagrama de Contexto del Sistema

En este nivel, tratamos a **Podcastify** como una "caja negra" y nos centramos en cómo interactúa con los usuarios y otros sistemas externos.

Elementos del Diagrama:

1. **Persona: Analista de Datos / Usuario Final**
 - **Descripción:** Usuario que consume las métricas de los podcasts, consulta el ranking Top-K y recibe recomendaciones personalizadas.
 - **Interacción:** Utiliza el sistema a través de la interfaz web o el API.
2. **Sistema: Podcastify (Caja Negra)**
 - **Descripción:** Plataforma centralizada para la ingestión, validación, análisis y visualización de registros de podcasts.
 - **Responsabilidades:** Procesar archivos CSV, detectar bots, calcular umbrales de descarga y generar recomendaciones.
3. **Sistema Externo: Proveedor de Logs / CDN**
 - **Descripción:** Fuente de datos externa que genera los registros de peticiones en formato CSV.
 - **Interacción:** Podcastify "lee" o "recibe" estos archivos para alimentar su base de datos.
4. **Sistema Externo: Herramientas de Visualización (Superset/Kibana)**
 - **Descripción:** Plataformas externas de BI que pueden conectarse a la base de datos de Podcastify para mostrar gráficos avanzados.

Nivel 2: Diagrama de Contenedores

En este nivel, abrimos la "caja negra" para mostrar los contenedores de software que componen la arquitectura técnica basada en el **Modelo Hexagonal**.

Elementos del Diagrama:

1. **Contenedor: Web API (FastAPI / Python)**
 - **Tecnología:** Python + FastAPI + OpenAPI.
 - **Responsabilidad:** Punto de entrada para las operaciones CRUD y el motor de recomendaciones. Expone los servicios de forma estándar.
 - **Comunicación:** Utiliza JSON sobre HTTPS.

2. **Contenedor: Motor de Procesamiento de Datos (Python)**
 - **Tecnología:** Scripts de Python con lógica de negocio pura (Dominio).
 - **Responsabilidad:** Ingesta de CSV, validación de reglas (RF-2 a RF-5), detección de bots y cálculo de acumulado de bytes para generar observaciones (RF-7).
 - **Comunicación:** Lectura directa de archivos y llamadas a la base de datos.
3. **Contenedor: Base de Datos (MongoDB / SQLAlchemy)**
 - **Tecnología:** NoSQL (MongoDB) o Relacional (SQLite/PostgreSQL).
 - **Responsabilidad:** Persistencia de registros en crudo, observaciones analizadas y entidades del sistema (Podcast, Episodios).
 - **Comunicación:** Conexión nativa de base de datos desde la API y el Procesador.
4. **Contenedor: Interfaz de Usuario (Frontend / Dashboard)**
 - **Tecnología:** React.js / Angular o Dashboards embebidos.
 - **Responsabilidad:** Visualización de los gráficos de columnas (RF-10) y presentación de las recomendaciones al usuario.
5. **Contenedor: Sistema de Archivos**
 - **Tecnología:** Volumen de Docker.
 - **Responsabilidad:** Almacenamiento físico de los ficheros data.csv cargados por el sistema.

Justificación de Contenedores

Cada contenedor está diseñado para ejecutarse de forma aislada dentro de un entorno **Docker**, permitiendo que la escalabilidad sea independiente (por ejemplo, escalar solo la API si hay muchas consultas de usuarios).