# Informe de Arquitectura y Defensa Técnica

Implementación de Dashboard Multimedia Reactivo (Angular 19+)

**Autor:** Fabian
**Fecha:** 24 de enero de 2026

## 1. Contraste Técnico: Signals vs. Modelo React (VDOM)

La diferencia fundamental entre la implementación realizada con Angular Signals y el modelo tradicional de React radica en la **granularidad de la actualización** y el mecanismo de detección de cambios.

- **React (Pull-based / VDOM):** En React, un cambio de estado dispara la re-ejecución del componente y sus hijos (re-render). React genera un nuevo árbol de Virtual DOM, lo compara con el anterior (Diffing) y reconcilia los cambios en el DOM real. Aunque optimizado, este proceso implica un costo computacional de .ªrriba hacia abajo$\tau$ depende fuertemente de la integridad referencial (‘useMemo‘, ‘useCallback‘) para evitar renderizados innecesarios.
- **Angular Signals (Push-based / Fine-grained):** Angular 19 utiliza un grafo de dependencias reactivo. Cuando una señal (como ‘selectedAssetId‘) cambia, notifica únicamente a sus dependencias directas.
    1. **Sin VDOM Diffing:** Angular no necesita comparar árboles. La señal actualiza directamente el nodo de texto o atributo en el DOM real.
    2. **Sin Zone.js (Opcional):** Al usar Signals, la detección de cambios deja de ser un barrido global del árbol de componentes para convertirse en notificaciones precisas y localizadas.

## 2. Criterio Profesional: Estabilidad del Estado en Alta Demanda

En un entorno de producción de alta demanda (ej. un sistema de gestión de activos con miles de transacciones), elegiría el modelo de Angular 19 sobre los Hooks de React debido a la **previsibilidad y la estabilidad del estado ("Glitch-free execution")**.

**Justificación:**

1. **Ejecución Glitch-Free (Libre de fallos):** En sistemas complejos, el estado derivado es crítico. Angular garantiza que una señal computada (‘computed‘) siempre vea el estado consistente y actualizado de sus dependencias antes de emitir un valor. En React, las cadenas complejas de ‘useEffect‘ pueden generar estados intermedios inconsistentes o bucles de renderizado si las dependencias no se gestionan manualmente con extrema precisión.
2. **Gestión de Efectos Secundarios y Sincronización:** La implementación de ‘linkedSignal‘ demuestra una superioridad arquitectónica sobre ‘useEffect‘ para la sincronización de estado (patrón reset on change"). En React, sincronizar un formulario cuando cambian las props requiere efectos que se ejecutan *después* del renderizado, causando a menudo un "flash"de contenido antiguo o complejidad en la gestión de ‘refs‘. ‘linkedSignal‘ maneja esto de forma síncrona y declarativa dentro del grafo reactivo, eliminando una clase entera de bugs de condiciones de carrera.

3. **Escalabilidad del Rendimiento:** Al eliminar la sobrecarga de la reconciliación del VDOM, el dashboard mantiene 60 FPS incluso si el modelo de datos subyacente crece exponencialmente, ya que el costo de actualización es proporcional a lo que cambia en pantalla, no al tamaño del árbol de componentes.

## 3. Repositorio de Código

La implementación completa, incluyendo el uso de `Resource API`, `LinkedSignal` y optimización de imágenes, se encuentra disponible en el siguiente enlace:

https://github.com/FabianSSJ/multimedia-dashboard