Universidad Autónoma de Occidente

Facultad de Ingeniería y Ciencias Básicas

Programa de Ingeniería de Datos E Inteligencia Artificial

Redes e Infraestructura

Arquitectura Completa del Sistema Redesflix

Plataforma de Streaming y Análisis de Datos Distribuido

Integrantes:

Jose David Mesa Ramirez/2236699
Sara Lucia Rojas/2236943
Esteban Cobo Gomez/2235882
Carlos Andres Orozco Caicedo/2237208

Fecha: 23 de mayo de 2025
Ciudad: Cali, Valle del Cauca, Colombia

RESUMEN

Este proyecto presenta el desarrollo y despliegue de *RedesFlix*, una plataforma web basada en microservicios orientada a la visualización de contenido audiovisual segmentado por tipo de membresía (básica, estándar y premium). La solución se compone de cuatro microservicios (Usuarios, Suscripciones, Catálogo e Historial) desarrollados en Node.js y desplegados mediante Docker en un entorno orquestado con Docker Swarm, lo que garantiza escalabilidad, portabilidad y gestión eficiente de los contenedores.

Durante la segunda etapa del proyecto, se implementó un pipeline completo de procesamiento de datos utilizando PySpark para simular y analizar visualizaciones de películas, generando reportes específicos sobre comportamientos de usuarios y rendimiento de títulos. Los resultados fueron almacenados en una base de datos MySQL y posteriormente consultados y visualizados a través del frontend de la plataforma.

El sistema fue validado a través de pruebas funcionales y de carga, asegurando la correcta comunicación entre servicios y el procesamiento eficiente de grandes volúmenes de datos. La integración de procesamiento distribuido y microservicios en un entorno contenedorizado demuestra la viabilidad de la arquitectura propuesta para aplicaciones escalables y orientadas a datos.

PALABRAS CLAVES

Microservicios, Docker Swarm, PySpark, Análisis de Datos, API REST, Streaming, Arquitectura, Escalabilidad, Contenedores.

INTRODUCCIÓN

En la actualidad, la creciente demanda por plataformas de contenido audiovisual bajo demanda ha generado la necesidad de sistemas informáticos capaces de responder eficientemente a altos volúmenes de usuarios, ofrecer disponibilidad constante y facilitar la adaptación a nuevos requerimientos funcionales. Ante este panorama, las arquitecturas monolíticas tradicionales presentan limitaciones en cuanto a escalabilidad, mantenibilidad y tolerancia a fallos, lo cual ha impulsado la adopción de enfoques más flexibles y desacoplados como la arquitectura de microservicios.

Las arquitecturas basadas en microservicios permiten dividir la lógica del sistema en componentes independientes que pueden ser desarrollados, desplegados y escalados de forma autónoma. Al combinarse con tecnologías de contenedorización como Docker, y mecanismos de orquestación como Docker Swarm, se facilita la automatización del despliegue, la replicación de servicios y el balanceo de carga entre múltiples nodos, garantizando una infraestructura más resiliente y portable.

En respuesta a este escenario, el proyecto *RedesFlix* propone el diseño e implementación de una plataforma académica de transmisión de películas por suscripción, con funcionalidades similares a las ofrecidas por servicios comerciales, pero adaptadas a un entorno experimental. La plataforma permite a los usuarios registrarse, suscribirse a planes diferenciados (básico, estándar o premium) y acceder a un catálogo de películas conforme a su nivel de membresía. La solución se estructura en torno a cuatro microservicios principales: gestión de usuarios, catálogo de contenido, suscripciones y registro de historial.

Adicionalmente, el sistema incorpora capacidades de análisis de datos a gran escala mediante el uso de PySpark, lo cual permite procesar visualizaciones simuladas de los usuarios y generar reportes analíticos segmentados. Esta funcionalidad complementa la experiencia del usuario al ofrecer estadísticas sobre patrones de uso y comportamiento por tipo de membresía.

El presente documento detalla el diseño arquitectónico y funcional de la plataforma, las decisiones tecnológicas adoptadas para su implementación, el pipeline de procesamiento de datos empleado, y los resultados obtenidos a partir de pruebas funcionales y de carga, validando la viabilidad del enfoque distribuido propuesto.

Los principales componentes son:

- Usuarios: Maneja el registro y la gestión de perfiles.
- Catálogo: Administra películas, géneros y metadatos.
- Suscripciones: Controla el acceso según el plan del usuario (Básico, Estándar, Premium).
- Historial: Registra las visualizaciones realizadas por los usuarios.
- Frontend: Interfaz web desarrollada en React.js que consume los servicios REST.
- Base de datos: Sistema MySQL compartido entre servicios, con soporte para replicación.
- Análisis: Módulo en PySpark para el procesamiento masivo de datos simulados.

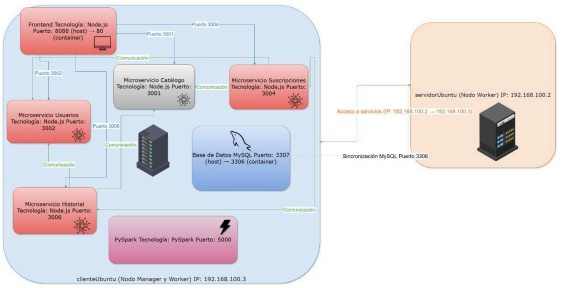
DIAGRAMAS SUGERIDOS

Diagrama de Componentes

El diagrama de componentes representa el flujo lógico del sistema desde la generación de datos hasta su visualización. Inicia con un archivo CSV de visualizaciones simuladas, que es procesado por PySpark para obtener estadísticas segmentadas. Los resultados se almacenan en MySQL y luego se consultan desde los microservicios para ser entregados al frontend. Este pipeline modular refleja la separación de responsabilidades en cada fase del procesamiento de datos, permitiendo mantener independencia entre el análisis, el almacenamiento y la visualización.

Diagrama de Despliegue (Docker Swarm)

El diagrama de despliegue muestra la arquitectura física y lógica del sistema desplegado en un clúster con Docker Swarm. Cada microservicio (usuarios, suscripciones, catálogo, historial y PySpark) corre en contenedores aislados, comunicándose mediante puertos específicos y conectados a una base de datos MySQL también contenida. El entorno incluye un nodo manager y un nodo worker distribuidos entre dos máquinas virtuales Ubuntu, facilitando la orquestación, balanceo y alta disponibilidad. Este diseño garantiza portabilidad, escalabilidad y sincronización eficiente entre servicios y nodos.



IMPLEMENTACIÓN

La implementación de la plataforma se basó en una arquitectura contenedorizada utilizando Docker, permitiendo el aislamiento de cada microservicio en su propio entorno de ejecución. El sistema fue desplegado mediante Docker Swarm, facilitando la distribución de servicios entre nodos, el escalado horizontal y la alta disponibilidad.

Cada microservicio fue desarrollado en Node.js, y configurado para exponer su funcionalidad mediante APIs REST. El frontend, desarrollado en *html*, se comunica directamente con los microservicios para realizar autenticación, navegación del catálogo, gestión de la suscripción y consulta del historial de visualización.

La persistencia de datos se gestiona mediante una base de datos MySQL, compartida entre los servicios mediante volúmenes definidos en el stack. Se implementó replicación y exposición del puerto 3307 en el host para permitir sincronización entre nodos.

La red entre contenedores se definió como una red overlay en el stack de Docker, asegurando la comunicación entre servicios distribuidos, incluso cuando se encuentran en nodos distintos.

Para la aplicación de análisis de datos del proyecto, se diseñó y generó una arquitectura que trabaja con dos datasets simulados: uno con información de usuarios de la plataforma (como edad, género, membresía y dispositivo de visualización), y otro con datos de películas y su comportamiento de visualización por tipo de suscripción.

Posteriormente, se desarrolló un componente central utilizando Apache Spark con PySpark, una tecnología especializada en el procesamiento distribuido de datos a gran escala. Este componente fue implementado dentro de un contenedor Docker y es responsable de realizar distintos análisis agregados y estadísticas clave sobre el comportamiento de visualización en la plataforma.

Entre las tareas que realiza la aplicación se encuentran: identificar las películas más y menos vistas, analizar el comportamiento de los usuarios por tipo de membresía, y generar resúmenes estadísticos útiles para la toma de decisiones. Todo el procesamiento se realiza en memoria usando Spark DataFrames, lo que garantiza eficiencia incluso con volúmenes altos de datos simulados.

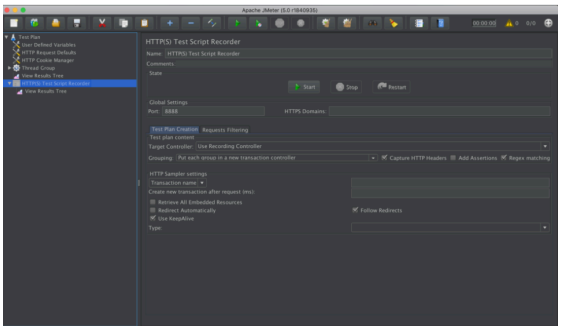
Los resultados del análisis son exportados automáticamente en formato CSV a una carpeta interna (/resultados), accesible desde la máquina virtual, facilitando su posterior consulta o integración con herramientas de visualización. Esta separación entre la lógica de procesamiento, los datasets de entrada y los archivos de salida permite un diseño modular, escalable y preparado para pruebas de carga o futuras integraciones analíticas.

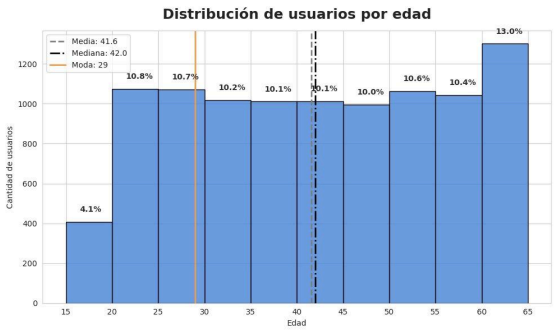
PRUEBAS Y RESULTADOS

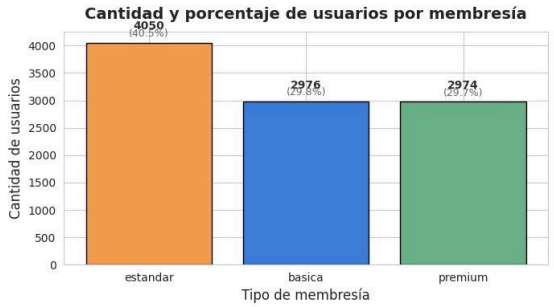
Se llevaron a cabo pruebas de escalabilidad con Apache JMeter sobre la aplicación principal *RedesFlix*, simulando cargas de 10, 50, 100 y hasta 1000 usuarios concurrentes. En todos los casos, se observaron tiempos de respuesta estables, sin pérdidas de conexión ni saturación de los microservicios, lo cual demuestra que el sistema responde adecuadamente bajo condiciones de uso intensivo.

En el componente analítico, se procesó un dataset de aproximadamente 8000 registros simulados que representaban visualizaciones reales dentro de la plataforma. El análisis con PySpark permitió generar reportes segmentados por tipo de membresía, género y frecuencia de uso. Los resultados fueron correctos, consistentes con los datos simulados y compatibles con las consultas estructuradas de la plataforma, validando así la eficiencia del pipeline de procesamiento.

Estas pruebas confirman que la arquitectura implementada ofrece escalabilidad, confiabilidad y capacidad de procesamiento distribuido, tanto en la gestión de servicios como en el análisis de grandes volúmenes de datos.







CONCLUSIONES

El desarrollo de la plataforma permitió demostrar la viabilidad técnica y arquitectónica de construir un sistema de visualización de contenido audiovisual segmentado por membresías, aplicando principios de escalabilidad, modularidad y procesamiento distribuido. La implementación basada en microservicios, contenedores Docker y orquestación mediante Docker Swarm facilitó un despliegue distribuido, replicable y gestionable, adaptado a las necesidades de sistemas modernos con múltiples usuarios concurrentes.

Uno de los aspectos clave del proyecto fue la integración del componente analítico utilizando PySpark, que permitió procesar visualizaciones simuladas de usuarios a gran escala. Esto posibilitó la generación de reportes analíticos clasificados por tipo de membresía, género de contenido y frecuencia de acceso. Estos reportes no solo evidencian el correcto funcionamiento del pipeline de datos, sino que también sientan las bases para la toma de decisiones fundamentadas, orientadas a la mejora continua del catálogo y la experiencia del usuario dentro de la plataforma.

Asimismo, las pruebas de escalabilidad realizadas con Apache JMeter jugaron un papel fundamental en la validación del rendimiento del sistema. Se simularon múltiples escenarios de carga con 10, 50, 100 y hasta 1000 usuarios concurrentes. Estas pruebas permitieron identificar el comportamiento del sistema bajo estrés, verificar la estabilidad de los servicios y garantizar que la arquitectura soporta el crecimiento progresivo de usuarios sin degradar la calidad del servicio. El uso de JMeter no solo evidenció la resiliencia y estabilidad del sistema, sino que también aportó datos cuantitativos clave para su futura optimización.

En conclusión, la página demostró ser una solución funcional, confiable y extensible. Su enfoque técnico, respaldado por un modelo de análisis de datos y una infraestructura escalable, lo convierten en una base sólida para futuras versiones que integren autenticación real, dashboards dinámicos, personalización basada en IA y procesamiento en tiempo real.

REFERENCIAS

[1]GitHub: https://github.com/EstebanCG2006/Redesflix