Arquitectura e Implementación El salami del italiano
Proyecto de curso
Redes e infraestructura

Universidad Autónoma de Occidente
Nicole Ruiz - Samuel Peña - Juan Pablo Coll
2240429 – 2240021 - 2240074

Resumen—Este trabajo describe el diseño, la implementación y la validación de una plataforma de analítica para ventas de pizzas basada en contenedores. La solución integra un proceso de extracción y transformación con Apache Spark en modo distribuido, la exposición de indicadores mediante una API REST en Flask con balanceo en Nginx y un panel interactivo en Streamlit. Se aborda el modelado de datos, la orquestación por *Docker Compose*.

I. Introducción

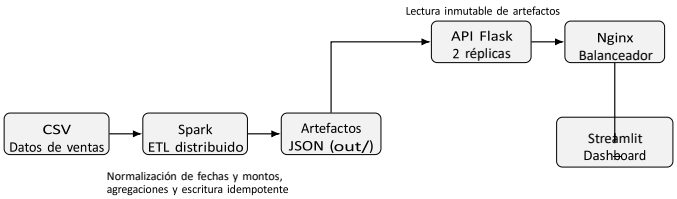
Este trabajo presenta una solucion de referencia para un caso de ventas, en la que se prioriza la separación de responsabilidades: un pipeline de datos con Spark, un borde de servicios REST balanceado y una interfaz de visualización ligera. La contribución principal radica en una implementación mínima y portable que permite ejecutar el flujo extremo a extremo en una única estación de trabajo con recursos limitados.

II. Datos y objetivos

Se parte de un conjunto de datos con registros de pedidos. El objetivo es estandarizar fechas y montos, depurar inconsistencias y producir métricas de negocio como ventas por categoría y tamaño, top de productos, ventas por hora y tickets promedios diarios. Estas métricas deben persistirse como JSON estructurado para consumo por la API y el dashboard

III. Metodología de implementación

La metodología siguió cuatro fases. Primero, limpieza y tipificación en Spark, con detección robusta de separadores de precio y fecha. Segundo, agregaciones con el dataframe para calcular indicadores clave. Tercero, publicación de artefactos como archivos JSON en un volumen compartido. Cuarto, exposición de los resultados por una API Flask detrás de Nginx con dos réplicas, y construcción de un panel Streamlit que consulta los endpoints



o lee directamente los artefactos según el modo de ejecución. La orquestación se realizó con compose en una red interna, exponiendo puertos mínimos al host.

IV. Diseño de la solución

El diseño separa computo, servicio y presentación. El clúster Spark ejecuta el ETL y escribe salidas en un volumen. La API consume los artefactos como fuente y aplica formateos ligeros. El balanceador Nginx unifica el acceso y facilita pruebas de reparto de carga. El panel Streamlit muestra consultas determinísticas.

IV-B. Diagrama de despliegue

La Figura 2 muestra la disposición física y de red en el host, incluyendo contenedores, redes internas y puertos expuestos. Se utiliza una red de cómputo para Spark y otra de borde para servicios web.

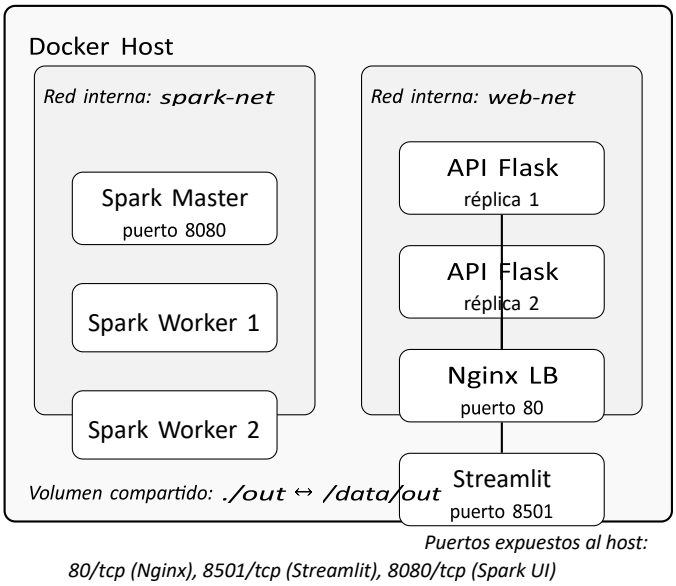


Figura 2. Diagrama de despliegue: contenedores por red interna, volumen compartido para artefactos y puertos publicados al host.

V-A. Alternativas consideradas y justificación

Para empaquetar y orquestar se evaluó el uso de Docker Compose, Docker Swarm y Kubernetes. Compose ofreció un equilibrio adecuado entre simplicidad operativa y expresividad para redes y dependencia. Swarm brindaba escalado nativo y, pero añadía una capa de complejidad innecesaria para un despliegue docente sin requisitos de alta disponibilidad reales. Kubernetes aportaba control fino de recursos. Se eligió Compose por su curva de aprendizaje baja, portabilidad entre sistemas operativos y capacidad suficiente para aislar redes y publicar puertos.

V. RESULTADOS

El proceso ETL normalizó campos, calculó ventas por categoría y tamaño, identificó los productos con mayor contribución y estimó el comportamiento por franja horaria. Los artefactos se almacenaron en un volumen compartido con escrituras idempotentes, y la API los expuso en endpoints estables. El panel visualizó los indicadores con latencia baja y trazabilidad directa hacia los archivos fuente.

VI. CONCLUSIONES

La arquitectura propuesta demuestra que es posible consolidar un flujo de analítica extremo a extremo con componentes abiertos, manteniendo claridad conceptual y portabilidad. La separación entre cómputo, servicio y presentación reduce el acoplamiento y preserva la auditabilidad. Las decisiones adoptadas permiten una evolución gradual hacia escenarios de mayor escala mediante cambios localizados en orquestación y formatos de datos.