

PROYECTO DE CURSO REDES E INFRAESTRUCTURA

Angela Maria Ruiz Tovar

Programa de ingeniería de datos e inteligencia Artificial, Facultad de ingeniería, Universidad Autónoma de Occidente , Cali, Colombia

Selección del dataset objetivo

Para llevar a cabo el análisis, fue necesario contar con un conjunto de datos que nos permitiera obtener información relevante sobre usuarios, productos, órdenes de compra, tendencias de consumo y otros aspectos fundamentales. Para ello, recurrimos a la plataforma **Kaggle.com**, un entorno ampliamente utilizado por la comunidad de ciencia de datos, que ofrece acceso a grandes volúmenes de información proporcionada tanto por usuarios como por la propia plataforma.

El conjunto de datos seleccionado para este proyecto se titula “**Sample - Superstore**”, y contiene información detallada sobre las ventas de una tienda durante el periodo comprendido entre los años **2014 y 2018**. Este dataset incluye columnas con datos como nombres de usuarios, productos, fechas, identificadores de órdenes, valores de venta, descuentos aplicados, entre otros. Gracias a esta variedad de atributos, fue posible analizar el comportamiento del consumidor, identificar patrones de compra y establecer tendencias que pueden ser útiles para optimizar el servicio al cliente y maximizar las ventas, basándose en datos históricos.

El dataset cuenta con **9.994 registros** y **21 columnas**, lo que proporciona un volumen considerable de información y permite realizar un análisis robusto y preciso.

Además, su estructura bien organizada y la calidad de los datos permiten explorar múltiples dimensiones del negocio, facilitando una comprensión integral del funcionamiento de la tienda y sus dinámicas de venta.

Alternativas analizadas

Despliegue y Empaquetamiento de Aplicaciones

Durante el desarrollo del curso , se exploraron distintas alternativas modernas para el empaquetamiento y despliegue de aplicaciones, enfocadas en la eficiencia, escalabilidad y portabilidad en entornos de producción. Entre las principales opciones evaluadas se encuentran:

- **Docker:** es una plataforma ligera que permite crear, empaquetar y ejecutar aplicaciones en contenedores. Estos contenedores encapsulan todo lo necesario para que una aplicación funcione correctamente, garantizando su portabilidad entre distintos entornos sin alterar su comportamiento.
- **Kubernetes:** es un sistema de orquestación de contenedores, diseñado para automatizar la implementación, el escalado y la gestión de aplicaciones basadas en Docker u otros motores de contenedores. Es ideal para entornos donde se requiere alta

disponibilidad, balanceo de carga y tolerancia a fallos.

- **AWS Fargate:** es un servicio de Amazon Web Services que permite ejecutar contenedores sin necesidad de gestionar servidores o clústeres. lo que lo hace una opción atractiva para equipos que buscan simplicidad y escalabilidad en la nube.

- **Apache Mesos:** actúa como un sistema operativo distribuido para centros de datos, gestionando recursos y ejecutando diferentes tipos de aplicaciones, incluidos contenedores y servicios como Spark y Hadoop. Su ventaja es la flexibilidad para soportar múltiples frameworks sobre una misma infraestructura.

Procesamiento Distribuido

Con el fin de optimizar el manejo de grandes volúmenes de datos, también se analizaron distintas herramientas para procesamiento distribuido, fundamentales en el análisis de datos a gran escala. Las principales alternativas fueron:

- **Apache Hadoop:** Es un marco de trabajo que permite el almacenamiento distribuido y el procesamiento masivo de datos mediante el paradigma MapReduce. Su fortaleza radica en su capacidad para gestionar grandes cantidades de datos estructurados y no estructurados en clústeres de bajo costo.
- **Apache Spark:** Es una plataforma de procesamiento distribuido en memoria, significativamente más rápida que Hadoop en ciertas tareas. Su arquitectura permite realizar análisis complejos, como aprendizaje automático y procesamiento en tiempo real, con gran eficiencia y velocidad.

Tecnologías Seleccionadas

Tras una cuidadosa evaluación de distintas alternativas, se seleccionaron Docker y Apache Spark como las tecnologías principales para el empaquetamiento de la aplicación y el procesamiento de datos, respectivamente. La elección se fundamentó en su consolidada trayectoria en la industria y en su capacidad para responder eficazmente a las necesidades técnicas del proyecto

Docker fue elegido como herramienta de empaquetamiento y despliegue por su capacidad para crear entornos aislados y reproducibles, lo que facilita significativamente el desarrollo, la prueba y la puesta en producción de aplicaciones. Al contener todos los componentes necesarios como dependencias, librerías y configuraciones dentro de contenedores livianos, Docker garantiza que la aplicación se ejecute de manera consistente sin importar el entorno. Esta portabilidad, sumada a su bajo consumo de recursos y facilidad de integración con plataformas de nube y orquestadores como Kubernetes, lo convierte en una solución altamente confiable y escalable.

Por otro lado, para el análisis de grandes volúmenes de datos, se optó por *Apache Spark*, una potente plataforma de procesamiento distribuido en memoria. Spark destaca por su velocidad, eficiencia y capacidad de ejecutar tareas complejas como transformaciones masivas, análisis en tiempo real y algoritmos de aprendizaje automático. A diferencia de otras soluciones como Hadoop, Spark permite manejar grandes conjuntos de datos con mayor agilidad, lo cual fue fundamental para procesar y analizar el dataset utilizado en este proyecto con un alto rendimiento.

La combinación de Docker y Apache Spark ofrece una arquitectura moderna, flexible y orientada al desempeño, que permite escalar el sistema fácilmente, mejorar los tiempos de respuesta y asegurar un flujo de trabajo completo desde el despliegue hasta el análisis final.

Definición de la arquitectura completa del sistema

Definición de microservicios

Microservicio de Usuarios: tiene como función principal almacenar y administrar la información personal de cada usuario registrado en la aplicación. Entre los datos que gestiona se encuentran el identificador único, nombre de usuario, correo electrónico, contraseña, ciudad de residencia y dirección de vivienda, entre otros campos relevantes para el perfil.

Este componente permite realizar operaciones fundamentales como el registro de nuevos usuarios, la actualización de datos personales y la gestión del perfil, garantizando así una

experiencia personalizada, segura y eficiente dentro de la plataforma.

Microservicio de Productos: es el encargado de almacenar y administrar toda la información relacionada con los artículos disponibles en la tienda. Entre los datos que gestiona se encuentran el nombre del producto, su descripción, precio, cantidad en stock y posibles descuentos aplicables.

Este componente permite al administrador crear nuevos productos, así como modificar o actualizar los ya existentes, adaptándose a las necesidades operativas y comerciales de la tienda. Su implementación facilita una gestión eficiente del catálogo, garantizando la disponibilidad y precisión de la información en todo momento.

Microservicio de Carrito de Compras : permite a los usuarios crear y gestionar un carrito personalizado donde pueden almacenar temporalmente los productos que desean adquirir. Este componente es responsable de asociar correctamente el carrito con el identificador del usuario, así como de mantener la información de los productos seleccionados.

Una vez que el usuario confirma su compra, el microservicio se encarga de enviar la información para generar la orden correspondiente y luego eliminarse automáticamente, ya que su contenido pasa a formar parte del pedido formal. Este proceso garantiza una transición fluida entre la selección de productos y la creación de la orden final, optimizando la experiencia de compra dentro de la aplicación.

Microservicio de Órdenes: es el responsable de procesar y gestionar las órdenes de compra realizadas por los usuarios. Este componente almacena información clave como los productos adquiridos, la fecha de la orden, la dirección de envío, el modo de envío, el identificador del usuario y el ID del carrito de compras asociado.

Además, este microservicio se encarga de enviar solicitudes al sistema de productos para reducir el stock correspondiente a cada artículo incluido en la orden, asegurando así la coherencia del inventario.

Por motivos de integridad y trazabilidad de los datos, una vez generada, la información de una orden no puede ser modificada ni eliminada, lo que garantiza la fidelidad del historial de transacciones dentro de la plataforma.

Descripción de los componentes

Este proyecto consiste en el desarrollo de una aplicación web compuesta por un frontend interactivo, un backend distribuido basado en microservicios, una API analítica con capacidades de procesamiento de datos, y una infraestructura de despliegue y orquestación mediante Docker Swarm. A continuación, se describen los principales componentes del sistema:

Frontend(Super-Store-web): El frontend es la interfaz gráfica con la que interactúan los usuarios finales. Su objetivo es proporcionar una experiencia de usuario clara, intuitiva y eficiente. Permite a los usuarios navegar por los productos disponibles, gestionar su cuenta, operar

con el carrito de compras y realizar órdenes.

Responsabilidades:

- Consumir servicios RESTful proporcionados por los microservicios y la API de análisis.
- Renderizar dinámicamente la información recibida.
- Manejar la navegación y las acciones del usuario en el navegador.

Tecnologías utilizadas:

HTML, CSS3, JavaScript, PHP

Backend (Microservicios): Su función en la aplicación web es gestionar distintos aspectos del sistema como el manejo de usuarios, productos disponibles, operaciones del carrito de compras y la generación de órdenes. Cada microservicio opera de forma independiente y se comunica con los demás mediante APIs REST.

Descripción general:

Los microservicios son componentes autónomos que realizan funciones específicas dentro de la aplicación. Esta arquitectura permite una mayor escalabilidad, facilidad en el mantenimiento, despliegue independiente de servicios y una separación clara de responsabilidades, optimizando el rendimiento general del sistema.

API de Análisis de Datos (Flask + Apache Spark): Es un servicio adicional que analiza los datos generados por la

aplicación (usuarios, productos, transacciones) utilizando técnicas de procesamiento distribuido. Expuesto a través de una API REST desarrollada con Flask.

Responsabilidades:

- Ejecutar procesos de análisis mediante Apache Spark.
- Extraer estadísticas como productos más vendidos, usuarios más activos, comportamientos de compra, etc.
- Proveer endpoints al frontend para visualizar resultados en tiempo real o por lotes.

Tecnologías utilizadas:

Python (Flask), Apache Spark, Pandas, Numpy

Docker y Orquestación con Docker Swarm: El sistema está empaquetado utilizando Docker, lo que permite un entorno homogéneo, portable y reproducible. La orquestación se realiza con Docker Swarm.

Responsabilidades:

- Empaquetar cada servicio en contenedores independientes.
- Orquestar el despliegue en múltiples nodos del clúster.
- Escalar servicios automáticamente según la carga.

Herramientas utilizadas:

- Docker Engine
- Docker Compose (para entornos de desarrollo)
- Docker Swarm (para despliegue en producción)

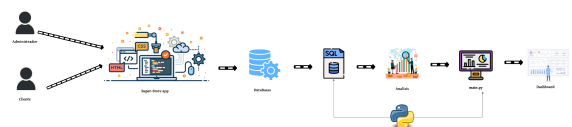
Bases de Datos Relacionales: Cada microservicio dispone de su propia base de datos, siguiendo el patrón **Database-per-service**, lo que garantiza autonomía, independencia en el escalado y menor acoplamiento entre componentes.

Responsabilidades:

- Almacenar datos estructurados y específicos del dominio del microservicio.
- Permitir transacciones seguras y consistentes.
- Facilitar la integración con sistemas de análisis (Spark).

Propuesta de pipeline a utilizar ***pipeline***

fig 1



Administrador: Representa al encargado de la tienda, quien tiene la responsabilidad de publicar los productos que desea vender, así como de establecer sus respectivos precios y gestionar su disponibilidad.

Usuario: Corresponde al cliente que accede a la tienda con el propósito de visualizar o adquirir productos. Para ello, crea un perfil dentro de la aplicación web.

Aplicación - Super-Store: Es la interfaz principal que permite la interacción entre el administrador, los usuarios y los productos. Su objetivo es facilitar el proceso de compra y venta dentro de la plataforma.

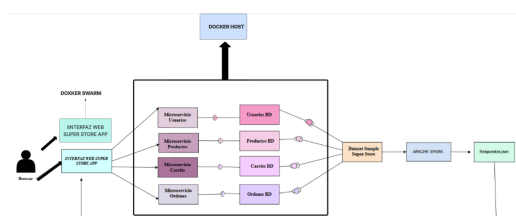
Base de datos (SQL): Es el sistema de almacenamiento donde se conserva toda la información recolectada por la aplicación, incluyendo datos de usuarios, productos, transacciones, entre otros.

Python (main.py): Es el componente encargado de recopilar, procesar y gestionar la información almacenada en la aplicación web. Actúa como intermediario entre la base de datos y la lógica de negocio de la plataforma.

Dashboard: Es una herramienta visual que presenta los datos recolectados de manera gráfica. Su propósito es facilitar el análisis de la información y apoyar la toma de decisiones estratégicas mediante indicadores clave.

Diagrama de componentes

fig 2



Docker Swarm: Es el sistema de orquestación que permite distribuir y balancear la carga de la aplicación

Super-Store entre dos o más nodos. Gracias a esta tecnología, se garantiza la alta disponibilidad y la escalabilidad del servicio web, mejorando su rendimiento y estabilidad.

Browser (Navegador): Representa el punto de entrada del usuario a la aplicación web. A través del navegador, tanto administradores como compradores pueden acceder a la plataforma e interactuar con sus diferentes funcionalidades.

Microservicios: La arquitectura del sistema se basa en microservicios, cada uno de los cuales está encargado de una función específica dentro de la plataforma. Entre los principales microservicios se encuentran:

- **Usuario:** Encargado de la gestión de credenciales, creación de perfiles, autenticación y administración de la información personal del cliente.
- **Productos:** Responsable de la creación y administración de productos, incluyendo atributos como precio, descuentos, descripciones y stock disponible.
- **Órdenes:** Administra las órdenes de compra generadas por los usuarios, así como el historial de transacciones.
- **Carrito:** Gestiona los carritos de compra individuales de los usuarios, permitiendo agregar, modificar o eliminar productos antes de confirmar la compra.

Bases de datos: Cada microservicio cuenta con una base de datos relacional independiente, diseñada para almacenar de manera estructurada la información relevante a su dominio. Esta segmentación facilita la escalabilidad, seguridad y mantenibilidad del sistema.

Dataset Sample Super Store: Se trata de un conjunto de datos utilizado como referencia para el análisis del comportamiento del sistema. Este dataset contiene información recopilada sobre productos, ventas, usuarios y órdenes, y sirve como base para la generación de reportes y visualizaciones.

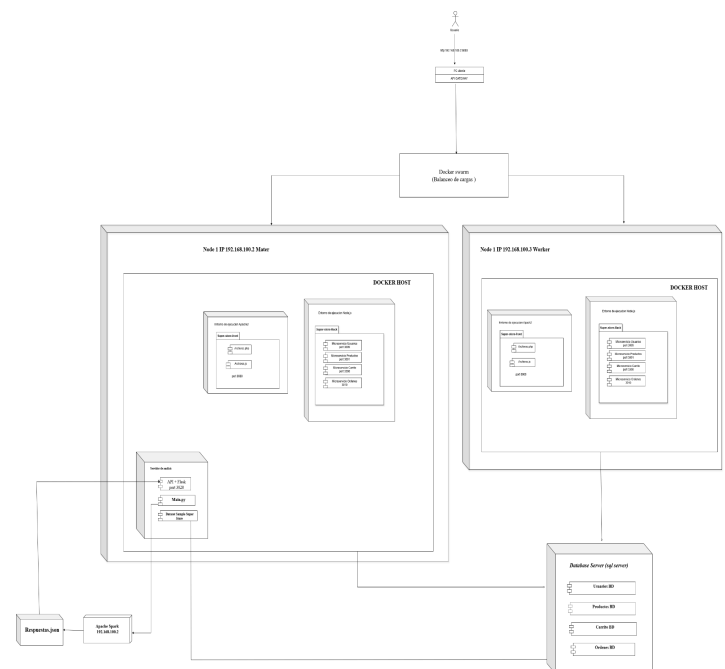
Apache Spark: Es la plataforma utilizada para procesar y transformar el dataset. Gracias a sus capacidades de procesamiento distribuido, permite realizar análisis complejos de grandes volúmenes de datos de forma eficiente.

respuestas.json: Este archivo representa la salida estructurada del análisis de datos realizado con Apache Spark. Contiene la información procesada en formato JSON y es consumido posteriormente por el frontend para la visualización de resultados y toma de decisiones.

Docker Host: Se refiere al entorno de ejecución en el cual se despliegan y ejecutan los contenedores que encapsulan cada uno de los microservicios. A través de Docker, se garantiza la portabilidad, aislamiento y eficiencia en la ejecución de cada componente del sistema.

Diagrama de despliegue

Fig 3



Infraestructura y Despliegue de Servicios

Servidores:

- **Node 1 (192.168.100.2):** Este nodo actúa como nodo maestro dentro del clúster de Docker Swarm. En él se despliegan los servicios principales de la aplicación Super-Store, generando múltiples réplicas para garantizar la disponibilidad y el balanceo de carga. Es el responsable de coordinar y distribuir las tareas a los nodos workers.
- **Node 2 (192.168.100.3):** Funciona como nodo trabajador (worker), encargado de ejecutar los servicios asignados por el nodo maestro. En este nodo se implementa también la interfaz de usuario de la aplicación.

web Super-Store asegurando redundancia y eficiencia en la ejecución de los microservicios.

Los microservicios fueron diseñados inicialmente en Node 1, y posteriormente desplegados también en Node 2 utilizando Docker Swarm. Cada microservicio se ejecuta en un contenedor aislado con un puerto específico asignado, lo que permite modularidad, escalabilidad y facilidad de mantenimiento.

- **Microservicio de Usuarios (Puerto 3009):** Gestiona toda la lógica relacionada con los usuarios de la plataforma. Esto incluye una autenticación, el registro, la administración de credenciales y gestión del perfil personal.
- **Microservicio de Productos (Puerto 3001):** Se encarga de la lógica de negocio asociada a los productos, incluyendo la creación, actualización, eliminación y visualización de artículos disponibles para la venta, así como la gestión de precios, descripciones y descuentos.
- **Microservicio de Órdenes (Puerto 3010):** Administra el flujo de órdenes de compra generadas por los usuarios. Se encarga de registrar, procesar y consultar pedidos realizados dentro de la plataforma.
- **Microservicio de Carrito (Puerto 3308):** Maneja los carritos de compra individuales de cada usuario. Permite agregar, modificar o eliminar productos antes de

completar una orden.

Bases de Datos

Cada microservicio cuenta con su propia base de datos SQL para almacenar información de forma estructurada y eficiente.

Apache Spark

Se utiliza como motor de procesamiento de datos para ejecutar tareas ETL y análisis sobre grandes volúmenes de información.

Dataset Sample Super Store

Conjunto de datos que contiene registros históricos sobre ventas, productos, regiones y usuarios.

Archivo respuestas.json

Archivo en formato JSON que almacena los resultados del análisis de datos, utilizado por el frontend.

Dashboard

Interfaz visual que muestra gráficas e indicadores clave para facilitar la toma de decisiones basada en datos.

Problemática

En la actualidad, muchas tiendas en línea enfrentan grandes retos relacionados con la gestión de productos, usuarios y procesos de compra. La ausencia de plataformas integradas y automatizadas genera ineficiencias operativas, pérdida de tiempo en tareas manuales y dificultades para tomar decisiones basadas en datos. Además, la falta de escalabilidad y de sistemas que permitan el análisis del comportamiento del cliente limita el

crecimiento y la competitividad de estos negocios en el entorno digital.

Solución

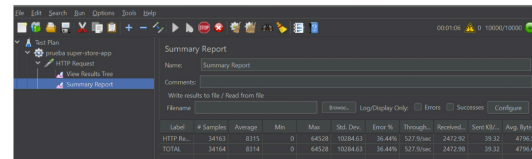
Como respuesta a esta problemática, se desarrolló **Super-Store**, una aplicación web moderna que integra funcionalidades clave para la administración de una tienda virtual. La plataforma permite a los administradores publicar productos, gestionar pedidos y visualizar métricas en tiempo real. Su arquitectura basada en microservicios y desplegada con Docker Swarm garantiza escalabilidad, rendimiento y disponibilidad. A su vez, la integración con Apache Spark permite procesar grandes volúmenes de datos, generando análisis que son representados gráficamente en un dashboard interactivo. Esto facilita la toma de decisiones estratégicas y mejora tanto la experiencia del administrador como la del cliente.

Pruebas de Escalabilidad con JMeter

Apache JMeter es una herramienta de código abierto utilizada para realizar pruebas de rendimiento, carga y estrés en aplicaciones web. Permite simular múltiples usuarios concurrentes interactuando con un sistema para evaluar su comportamiento bajo distintas condiciones de carga.

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Through...	Received...	Sent KB/s	Avg. Bytes
HTTP Re...	1000	77	1	1084	188.78	0.00%	486.6/sec	2876.04	57.03	6052.1
TOTAL	1000	77	1	1084	188.78	0.00%	486.6/sec	2876.04	57.03	6052.1

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Through...	Received...	Sent KB/s	Avg. Bytes
HTTP Re...	101000	736	0	77494	4223.99	0.00%	419.8/sec	2481.16	49.20	6052.0
TOTAL	101000	736	0	77494	4223.99	0.00%	419.8/sec	2481.16	49.20	6052.0



Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Through...	Received...	Sent KB/s	Avg. Bytes
HTTP Re...	34163	8315	0	64528	10384.63	36.44%	527.9/sec	2472.92	78.32	4796.6
TOTAL	34164	8314	0	64528	10384.63	36.44%	527.9/sec	2472.98	78.32	4796.6

En el caso de la aplicación Super-Store, se llevaron a cabo tres pruebas de escalabilidad con el objetivo de evaluar su capacidad de respuesta ante diferentes volúmenes de usuarios:

- Primera prueba: se simularon 100 usuarios con un loop count de 10, lo que generó 1.000 solicitudes. El sistema respondió correctamente, sin presentar errores (0% de error).
- Segunda prueba: se incrementó la carga a 1.000 usuarios con un loop count de 100, lo que resultó en 100.000 solicitudes. La aplicación mantuvo una respuesta estable, nuevamente con 0% de error.
- Tercera prueba: se alcanzó un nivel de estrés significativo con 10.000 usuarios y un loop count de 500, generando un total de 5 millones de solicitudes. En este escenario, el sistema experimentó un 50% de error y finalmente colapsó debido a la alta demanda.

Análisis:

Los resultados indican que la aplicación maneja de forma eficiente cargas pequeñas y medianas, demostrando una arquitectura sólida hasta cierto punto. Sin embargo, bajo una carga masiva, se evidencian limitaciones en la capacidad de procesamiento, disponibilidad de recursos o configuración del entorno de despliegue (por ejemplo, contenedores, base de datos

o red). Esto sugiere la necesidad de optimizar el rendimiento del sistema, incorporar mecanismos de escalado dinámico o balanceo de carga más robusto, y realizar ajustes en la infraestructura para soportar un mayor número de usuarios concurrentes sin afectar la estabilidad del servicio.

Enlaces de interés:

[1] Enlace para ver diapositivas

https://www.canva.com/design/DAGoLans-sk/TYcD9V_EttjowHhGpjzywQ/edit?utm_content=DAGoLans-sk&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton

[2] Enlace del repositorio de GitHub

<https://github.com/20angela26/Super-Store-App-with-Docker.git>

Referencias

[1] “Contenedores de Docker | ¿Qué es Docker? | AWS,” Amazon Web Services, Inc. <https://aws.amazon.com/es/docker/>

[2] “Apache JMeter - Apache JMeter™.” <https://jmeter.apache.org/>

[3] “Apache Spark™ - Unified Engine for large-scale data analytics.” <https://spark.apache.org/>

[4] “Orquestación de contenedores para producción,” Kubernetes. <https://kubernetes.io/es/>

[5] “MySQL.” <https://www.mysql.com/>