AutomarketUao (2025)

Arroyo. Miguel, Correa. Annie, Gomezcaseres. Fabián, y Hoyos. Juan.
Universidad Autónoma de Occidente
Facultad de Ingeniería
Ingeniería de Datos e Inteligencia Artificial

[1]

***Resumen*—** Este proyecto presenta el diseño e implementación de *AutoMarketUAO*, una plataforma web basada en microservicios para la gestión de compra y venta de vehículos en línea. La arquitectura está compuesta por tres microservicios principales (Usuarios, Publicaciones, Trámites), desplegados usando Docker Swarm. En la fase de infraestructura, se integró un clúster de procesamiento de datos usando Spark para analizar dos fuentes principales de datos: publicaciones, trámites. Entre los análisis realizados se incluyen el conteo de trámites finalizados, distribución porcentual de marcas publicadas y promedios de precios por año de fabricación. Los resultados se visualizan en un dashboard para un posterior análisis. Este proyecto demuestra la integración efectiva entre una arquitectura basada en contenedores y herramientas de análisis distribuido, cumpliendo con los requerimientos de escalabilidad, rendimiento y modularidad.

Palabras clave: Microservicios, Docker, Análisis de Datos, Plataforma de Vehículos, Dashboard

INTRODUCCIÓN

La transformación digital ha impulsado la evolución de plataformas en línea que permiten conectar a usuarios para la compra y venta de bienes y servicios. En este contexto, el sector automotriz no ha sido ajeno a dicha transformación, y cada vez son más las personas que utilizan soluciones digitales para adquirir o vender vehículos usados. Sin embargo, muchas de estas plataformas carecen de una estructura tecnológica escalable, lo que limita su rendimiento, flexibilidad y capacidad de análisis de datos.

El presente proyecto, denominado **AutoMarketUAO**, propone el diseño, desarrollo y despliegue de una plataforma web moderna para la compra y venta de vehículos usados, construida bajo una arquitectura de microservicios y contenedores. El sistema está compuesto por distintos servicios independientes como gestión de usuarios, publicaciones, favoritos y trámites, cada uno encapsulado y desplegado en contenedores Docker.

Además del desarrollo funcional de la aplicación, AutoMarketUAO incorpora capacidades analíticas a través del procesamiento distribuido de datos con Apache Spark, permitiendo la extracción de métricas relevantes sobre el comportamiento de los usuarios, estado de los trámites, distribución de publicaciones y más. Estos datos se visualizan mediante un dashboard interactivo diseñado para el administrador de la plataforma.

Este documento describe el análisis, diseño, implementación y validación de la solución. Se exploran diferentes alternativas de empaquetado y despliegue, se define una arquitectura modular y escalable, y se llevan a cabo pruebas de funcionamiento y rendimiento que permiten evaluar la efectividad del sistema propuesto.

I. SELECCIÓN DEL DATASET OBJETIVO

En esta parte del proyecto se han considerado dos datasets como objetivo principal.

Cada uno de los datasets propuestos contiene aproximadamente 10.000 registros. A continuación, se describe la estructura de cada uno basada en los esquemas SQL:

El primer dataset modela información relacionada con trámites, basándose en transacciones de compra-venta, por lo que incluye información de vendedor, comprador y vehículo. Cada trámite contiene su id, id del vendedor como también su usuario, teléfono, email, recolectando la información del vendedor, así como también almacena la información del comprador.
Además, el dataset incluye los datos relacionados con el vehículo involucrado en el trámite como su id, marca, modelo, año, precio lo que permite la identificación y

[1]

reconocer sus características.
La fecha de inicio del trámite y la fecha del fin, la revisión de los documentos, cita, contrato, pago, traspaso, entrega y el estado actual del trámite (ej. "Finalizado", "Activo") son indicadores almacenados en tal dataset que representan el estado de diversas etapas dentro del proceso del trámite lo que es crucial para el seguimiento del progreso y el análisis de la finalización de los trámites.

Este dataset permitiría realizar análisis sobre el flujo y la eficiencia de los procesos de trámite, la información de vendedores/compradores, y las características de los vehículos involucrados en las transacciones.

Por otro lado, el segundo dataset contiene información sobre las publicaciones que se realizan de vehículos puestos a la venta.

Cada publicación cuenta con un Identificador único, la fecha de publicación, el Identificador del usuario que realiza la publicación.
La marca, modelo, año, precio, kilometraje, tipo de combustible, transmisión, tamaño del motor, puertas y último dueño son características detalladas del vehículo publicado. También se incluye la descripción, ubicación y el estado de la publicación (Disponible, Vendido, Reservado).

Es importante notar que, aunque cada dataset consta de 10.000 registros que son suficientes para demostrar la configuración, el despliegue y el funcionamiento de las herramientas y el clúster de procesamiento distribuido, un escenario verdaderamente "a gran escala" implicaría la necesidad de trabajar con volúmenes de datos significativamente mayores.

II. GENERACIÓN Y SELECCIÓN DE ALTERNATIVAS DE SOLUCIÓN

Para la primera parte del proyecto, que consiste en el empaquetado de una aplicación inicial basada en microservicios y API REST y su despliegue en un clúster de contenedores con mecanismos de escalabilidad y balanceo de carga, se evaluaron diversas tecnologías para cumplir con estos requisitos.

A. Empaquetado de la Aplicación en Contenedores

La evaluación comenzó con la selección de una plataforma para crear contenedores de software. Se identificó que los contenedores empaquetan el software en un sistema de archivos completo que contiene todo lo necesario para ejecutarse (código, runtime, librerías, etc.). Esto garantiza que el software se ejecutará siempre de la misma manera, independientemente de su ambiente.

Se consideró Docker como la plataforma principal para el empaquetado. Las ventajas clave de Docker que se evaluaron incluyeron:

- Es liviano, ya que los contenedores comparten el mismo kernel del sistema operativo y usan menos RAM en comparación con las máquinas virtuales.
- Es eficiente en el uso de disco, pues las imágenes comparten archivos comunes en el sistema de archivos.
- Es abierto, basado en estándares, lo que permite que los contenedores se ejecuten en la mayoría de las distribuciones de Linux y Windows.
- Ofrece seguridad por defecto, ya que los contenedores son componentes de software aislados entre sí y de la infraestructura subyacente.

A diferencia de las máquinas virtuales, que incluyen el sistema operativo completo del invitado y pueden ocupar decenas de GBs, los contenedores de Docker comparten el kernel del sistema operativo host y corren como procesos aislados en el "user space". Docker no está ligado a una infraestructura específica, pudiendo correr en cualquier computador, infraestructura o cloud.

Dada la naturaleza basada en microservicios de la aplicación, se consideró una herramienta para definir y ejecutar aplicaciones Docker de contenedores múltiples. Docker Compose fue evaluado como una herramienta que utiliza un archivo YAML para configurar los servicios de una aplicación multi-contenedor, permitiendo crear e iniciar todos los servicios con un solo comando. El uso de Dockerfile para definir el entorno de cada servicio y el archivo docker-compose.yml para orquestarlos se identificó como un flujo de trabajo adecuado para empaquetar la aplicación multi-servicio.

B. Despliegue en un Clúster de Contenedores y Mecanismos Asociados

El proyecto requiere el despliegue de la aplicación empaquetada en un clúster de contenedores. Para gestionar aplicaciones desplegadas en clústeres, se evaluaron herramientas de orquestación de contenedores. Estas herramientas extienden las capacidades de administración del ciclo de vida de contenedores a escenarios complejos, abstraen la infraestructura de los hosts y permiten tratar un clúster completo como si fuera una sola máquina.

Se consideró Docker Swarm como una plataforma de orquestación. Docker Swarm permite a los usuarios crear y administrar un clúster de nodos Docker. En Docker Swarm, los contenedores se lanzan usando servicios, que son grupos de contenedores de la misma imagen, y estos servicios permiten escalar la aplicación. La escalabilidad es un requisito explícito del proyecto, y la capacidad de definir servicios replicados en Swarm aborda esta necesidad.

Otro requisito clave para el despliegue es la configuración y prueba del balanceo de carga. Adicionalmente, se evaluó HAProxy (High Availability Proxy) como una alternativa o complemento. HAProxy es un balanceador y proxy TCP/HTTP de fuente abierta que mejora el desempeño y la confiabilidad distribuyendo cargas de trabajo entre múltiples servidores. Utiliza conceptos como Frontends (que definen cómo las peticiones se reenvían) y Backends (conjuntos de servidores que reciben peticiones), y soporta algoritmos de balanceo como Round Robin y Leastconn. La selección entre la capacidad de balanceo de carga inherente de Docker Swarm o el uso de una herramienta dedicada como HAProxy dependerá de los requisitos específicos de desempeño y la complejidad del balanceo necesario, lo cual se definirá en la fase de diseño.

Basado en la necesidad de gestionar el despliegue de contenedores a escala, la escalabilidad y el balanceo de carga en un entorno de clúster, Docker Swarm fue seleccionado como la plataforma de orquestación principal. La estrategia específica para el balanceo de carga (usando Swarm) se definirá en detalle en la siguiente fase.

III. DEFINICIÓN DE LA ARQUITECTURA COMPLETA DEL SISTEMA

La arquitectura propuesta para el sistema AutoMarketUAO se fundamenta en el paradigma de microservicios, encapsulados en contenedores mediante Docker. Estos servicios se definen y gestionan como una aplicación multicontenedor a través de Docker Compose y se despliegan utilizando Docker Swarm, lo cual permite la orquestación de servicios y su escalabilidad horizontal.

La infraestructura está distribuida entre dos máquinas virtuales: una máquina máster, que aloja el frontend y los servicios lógicos, y una máquina worker, dedicada al almacenamiento de datos y procesamiento distribuido. El archivo docker-stack.yml permite definir y desplegar los servicios del sistema de forma centralizada. Se contempla también la incorporación de mecanismos de balanceo de carga para distribuir eficientemente las solicitudes entrantes hacia los diferentes servicios del clúster.

La arquitectura garantiza escalabilidad, modularidad, aislamiento entre servicios, y una alta disponibilidad, permitiendo además la integración con la aplicación de análisis de datos correspondiente a la segunda fase del proyecto, cuyos resultados se presentan en el dashboard del sistema.

A continuación, se describen los principales componentes de esta arquitectura:

A. Capa de Presentación

La interfaz de usuario está desarrollada utilizando tecnologías web como PHP, HTML, JavaScript y Java. Esto ofrece funcionalidades diferenciadas para los distintos tipos de usuario: compradores, vendedores y administradores. Mientras que los compradores y vendedores interactúan con funcionalidades como la publicación o búsqueda de vehículos, los administradores acceden a un dashboard interactivo donde se visualizan los resultados analíticos obtenidos por el clúster de procesamiento. Es importante destacar que esta visualización no es en tiempo real, sino que se basa en análisis periódicos o generados bajo demanda.

B. Capa de Microservicios (Backend)

El backend del sistema está compuesto por tres microservicios independientes, desplegados como contenedores y orquestados mediante Docker Swarm:

Servicio de Usuarios: Gestiona el registro, autenticación y administración de perfiles de usuario.

Servicio de Publicaciones: Permite la creación, edición, consulta y eliminación de publicaciones de vehículos.

Servicio de Trámites: Administra los procesos de compraventa, incluyendo el seguimiento del estado de cada trámite.

Esta separación de responsabilidades permite realizar despliegues modulares y facilita el mantenimiento evolutivo del sistema.

C. Capa de Persistencia de Datos

Cada microservicio cuenta con su propia instancia de base de datos MySQL, asegurando así independencia y evitando acoplamiento entre servicios. Estas instancias están alojadas en contenedores dentro de la máquina worker, y cada una de ellas mantiene un esquema de datos propio y especializado.

D. Capa de Procesamiento de Datos

Para el análisis de información, se ha implementado un clúster de procesamiento distribuido con Apache Spark, el cual opera sobre los datos exportados desde las bases MySQL. Este clúster ejecuta diversos análisis estadísticos, entre ellos:

Cantidad de trámites por estado (finalizados, cancelados, en proceso).

Distribución porcentual de publicaciones por marca de vehículo.

Promedio de precios por modelo y año de fabricación.

Tiempo promedio entre publicación y finalización del trámite.

Los resultados de estos análisis se exportan en archivos CSV y son visualizados posteriormente por los administradores del sistema en el dashboard.

IV. COMPONENTES DEL SISTEMA

La arquitectura de AutoMarketUAO está diseñada bajo un enfoque de microservicios, asegurando la independencia, escalabilidad y mantenibilidad de cada uno de sus componentes. El sistema está conformado por tres microservicios principales: gestión de usuarios, publicaciones y trámites, los cuales se comunican a través de API REST. Además, se integran componentes de análisis y visualización para procesar los datos y mostrar reportes al administrador.

- **Diagrama de Pipeline**

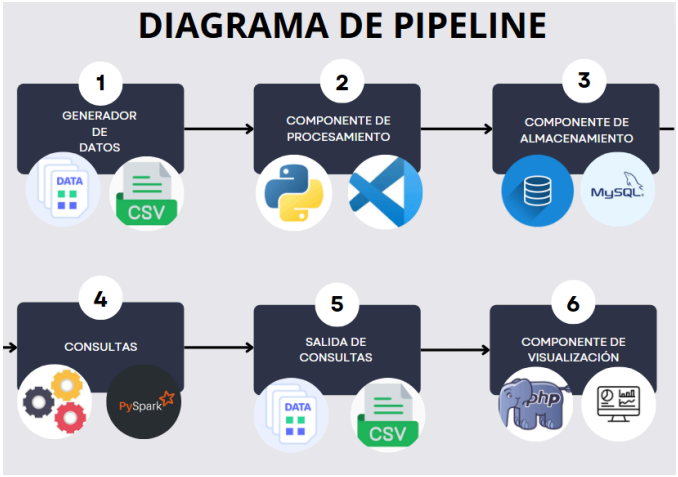

Diagrama de Pipeline

El pipeline del sistema consta de seis etapas, como se muestra en la Figura:

1. **Generador de Datos**: Las fuentes de datos son simuladas y exportadas como archivos CSV, los cuales representan los trámites y publicaciones realizados en la plataforma.

2. **Componente de Procesamiento**: Utiliza Python para procesar los datos y transformarlos en información útil.

3. **Componente de Almacenamiento**: Los datos generados son almacenados en bases de datos MySQL distribuidas, cada una correspondiente a un microservicio.

4. **Consultas**: PySpark realiza consultas específicas sobre los datos almacenados,

facilitando el análisis distribuido.

5. **Salida de Consultas**: Los resultados de las consultas se exportan en formato CSV para ser utilizados en visualizaciones.

6. **Visualización**: Se construyó un dashboard en PHP que carga los archivos CSV procesados y presenta información estadística e indicadores clave.

- **Diagrama de Flujo y Despliegue**

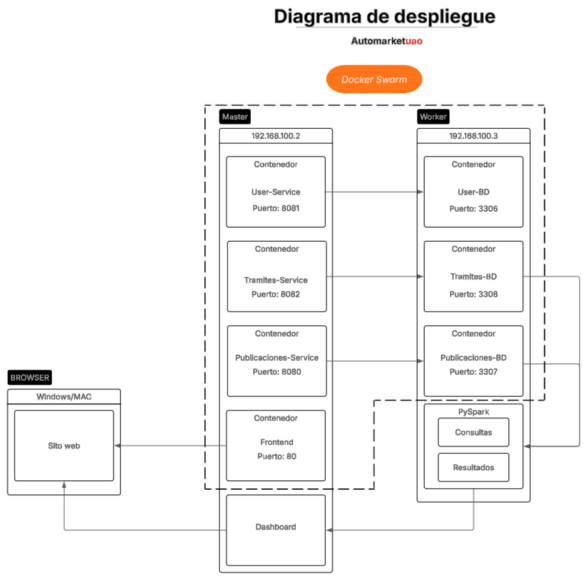

Diagrama de Despliegue

El sistema se encuentra desplegado en un clúster Docker Swarm, como se observa en la Figura. Se utiliza un nodo **Master** (192.168.100.2) que gestiona tres servicios principales: User-Service, Trámites-Service y Publicaciones-Service, cada uno corriendo en un contenedor individual. El frontend también corre en este nodo, atendiendo las solicitudes de los usuarios a través del puerto 80.

En el nodo **Worker** (192.168.100.3), se encuentran los contenedores de las bases de datos (User-BD, Trámites-BD, Publicaciones-BD) conectados a sus respectivos servicios. Además, fuera del entorno Dockerizado se encuentra el componente de análisis PySpark y el dashboard, debido a que no fueron empaquetados como contenedores en esta fase del proyecto.

- **Descripción de los Componentes**

Microservicios REST: Implementados en Node.js (usuarios y trámites) y Java Spring Boot (publicaciones), cada uno se comunica con su propia base de datos y expone endpoints específicos.

Bases de Datos: Distribuidas en el nodo Worker para desacoplar los servicios y optimizar el rendimiento.

PySpark: Ejecuta consultas complejas y procesamiento de datos en paralelo a partir de los datos exportados desde las bases.

Dashboard PHP: Permite la visualización interactiva de indicadores como duración de trámites, publicaciones por marca, estado de trámites, entre otros.

V. IMPLEMENTACIÓN DEL SISTEMA

La solución diseñada fue implementada utilizando las tecnologías y herramientas seleccionadas previamente, tales como Docker para la contenerización, Docker Compose para el empaquetado multicontenedor y Docker Swarm como plataforma de orquestación. Cada microservicio del sistema (Usuarios, Publicaciones, Trámites) fue desarrollado y encapsulado en contenedores independientes, asegurando el aislamiento y la portabilidad entre entornos.

La infraestructura fue desplegada en un entorno distribuido que consta de dos nodos: uno máster que alberga los servicios de frontend y lógica de negocio, y un nodo worker para el almacenamiento de datos y procesamiento distribuido con Apache Spark. La configuración se realizó a través de archivos YAML que permitieron la definición detallada de redes, volúmenes y servicios.

Posterior a la implementación, se realizaron pruebas funcionales de cada uno de los componentes del sistema. Estas pruebas garantizaron que los microservicios respondieran adecuadamente a las solicitudes esperadas, cumpliendo con las especificaciones técnicas definidas en la fase de diseño.

Se utilizó Postman para validar los endpoints de las API REST de cada microservicio. Las pruebas incluyeron casos de uso como registro de usuarios, publicación de vehículos, inicio y seguimiento de trámites, entre otros. Se verificó además la correcta integración entre los servicios, asegurando que la comunicación por medio de redes internas de Docker se realizará de forma eficiente y segura.

Con el fin de validar la capacidad del sistema para operar bajo distintas condiciones de carga, se llevaron a cabo pruebas de escalabilidad y desempeño. Estas incluyeron:

- **Pruebas de carga**: se simularon múltiples usuarios concurrentes accediendo a los servicios, utilizando Apache JMeter. Se midió el tiempo de respuesta y la tasa de errores en distintos escenarios.

- **Pruebas de estrés**: se incrementó progresivamente la carga sobre el sistema hasta observar su comportamiento en situaciones límite. Estas pruebas permitieron identificar puntos críticos de saturación y evaluar la robustez de la infraestructura.

- **Pruebas de rendimiento**: se evaluaron métricas como latencia, rendimiento y utilización de recursos (CPU, RAM, red) durante operaciones normales y bajo carga.

- **Pruebas de balanceo de carga**: se verificó que el sistema pudiera distribuir adecuadamente las peticiones entre réplicas de servicios desplegadas con Docker Swarm. Se utilizó el algoritmo de Swarm, para confirmar que las solicitudes se repartieran equitativamente entre los nodos disponibles.

Los resultados obtenidos permitieron validar que el sistema AutoMarketUAO no solo cumple con los requisitos funcionales, sino que también está preparado para escalar y mantener un desempeño óptimo en un entorno distribuido.

VI. CONCLUSIONES

El desarrollo del proyecto AutoMarketUAO permitió integrar múltiples conceptos clave de infraestructura de software moderna, incluyendo arquitecturas basadas en microservicios, contenedores, despliegue distribuido, y procesamiento de datos a gran escala. A lo largo de este proceso, se logró construir una plataforma funcional para la compraventa de vehículos usados, apoyada en una infraestructura sólida y escalable.

Uno de los principales logros fue el diseño y despliegue de tres microservicios independientes —usuarios, publicaciones y trámites— que se comunican de forma eficiente mediante API REST y fueron empaquetados utilizando Docker. El uso de Docker Swarm permite orquestar los contenedores en un clúster distribuido, lo cual facilitó la escalabilidad y la modularidad del sistema. Esta arquitectura habilitó un desarrollo más organizado, favoreciendo la separación de responsabilidades y la posibilidad de mantener y escalar los servicios de manera independiente.

En la capa de análisis de datos, se implementó un pipeline eficaz utilizando Apache Spark y PySpark para el procesamiento distribuido de la información generada por la aplicación. Esta solución permitió generar indicadores clave como: duración promedio de trámites, porcentaje de trámites por estado, pasos críticos donde más se cancelan procesos, publicaciones por ciudad y marca, entre otros. Estos resultados fueron presentados en un dashboard interactivo que ofrece al administrador una visión clara del funcionamiento del sistema y el comportamiento de los usuarios.

Durante el proceso también se enfrentaron desafíos importantes, especialmente en la integración entre los microservicios y los módulos de análisis. Sin embargo, se superaron mediante estrategias como el uso de archivos intermedios CSV, consultas asincrónicas y separación de entornos. El proyecto evidencia el valor de combinar tecnologías de contenedores con herramientas de Big Data para construir soluciones robustas, reutilizables y enfocadas en la toma de decisiones.

En conclusión, AutoMarketUAO no solo es una solución funcional para la compraventa de vehículos, sino también una plataforma tecnológica que demuestra la aplicabilidad de arquitecturas distribuidas en escenarios reales. El proyecto representa un avance significativo en la formación profesional de los estudiantes involucrados, consolidando conocimientos teóricos mediante una implementación práctica que integra desarrollo web, despliegue en contenedores y análisis de datos a gran escala. *Repositorio de GitHub*

Referencias

S. Ratliff, “Docker: Accelerated Container Application Development,” Docker, May 09, 2025. https://www.docker.com/

“‘Docker compose,’” Docker Documentation, Apr. 28, 2025. https://docs.docker.com/compose/

“‘docker,’” Docker Documentation, Aug. 14, 2024. https://docs.docker.com/engine/reference/commandline

J. Msv, “From containers to container orchestration,” The New Stack, Oct. 23, 2017. [Online]. Available: https://thenewstack.io/containers-container-orchestration