Informe Proyecto Final Infraestructura

Brayan Stiven Tigreros González - brayan.tigreros@uao.edu.co
Andrés Felipe Rivera Diaz – andres_f.rivera_d@uao.edu.co
Jose David Santa Parra - jose.santa@uao.edu.co
Julian David Cuellar Cubillos – julian_david.cuellar@uao.edu.co

***Resumen*—Este documento es un informe que evidencia como se desarrollaron las diferentes etapas del proyecto para sus dos partes, empaquetado y despliegue de una API mediante contendores [2-5] y la creación de un clúster de procesamiento de datos distribuido [6-9].**

***Abstract*—This document is a report that demonstrates how the different stages of the project were developed for its two parts: packaging and deployment of an API using containers [2-5] and the creation of a distributed data processing cluster [6-9].**

I. Introducción

En el presente proyecto se abordan dos componentes fundamentales en el desarrollo de aplicaciones modernas: el empaquetado y despliegue de una API mediante contenedores, y la implementación de un clúster para el procesamiento distribuido de datos.

El objetivo principal es integrar herramientas de infraestructura como Docker y Apache Spark para construir una solución eficiente, escalable y replicable. A través del uso de contenedores, se busca facilitar el despliegue de servicios, mientras que con el procesamiento distribuido se pretende aprovechar al máximo los recursos computacionales disponibles para el análisis de datos a gran escala.

Este proyecto refleja la aplicación práctica de conceptos clave aprendidos durante el curso, combinando teoría y práctica en un entorno controlado que simula un caso real en la industria.

II. Análisis

– *Selección del dataset objetivo*

Este dataset contiene información detallada sobre laptops disponibles para la venta, incluyendo especificaciones técnicas y precios finales [1]. Está diseñado para análisis de productos tecnológicos en el mercado.

Nombre del dataset: laptops_csv_REI

Propósito: Recopilamos este dataset para ser utilizado en una aplicación de análisis de datos distribuidos para realizar análisis varios para resolver problemáticas como:

- ¿Cuántas laptops de determinada marca hay en el dataset?
- ¿Cuáles son las laptops que tiene una pantalla táctil?
- ¿Cuál es el promedio del precio de una marca determinada de laptops?

– *Descripción de las variables*

Información de cada columna:

- **Status**: Estado del producto (Nuevo, Usado o Renovado)
- **Brand**: Nombre del fabricante (Asus, Acer, HP, Lenovo, MSI, Toshiba, etc)
- **Model**: Modelo específico del producto
- **CPU**: Tipo y modelo de procesador
- **RAM**: Cantidad de memoria RAM
- **Storage**: Capacidad de almacenamiento
- **Storage Type**: Tipo de de almacenamiento
- **GPU**: Tarjeta Gráfica de la laptop
- **Screen**: Tamaño de la pantalla en pulgadas
- **Touch**: Indica si la pantalla es táctil (“Yes” or “No”)
- **Final Price**: Precio de la laptop

– *Tamaño y estructura del Dataset*

Número de filas: 12.099
Número de columnas: 11

– *Formato y acceso*

Formato: CSV
Tamaño del archivo: 155 KB (159.481 bytes)
Este dataset fue modificado por uno que fue tomado desde Kaggle, este nuevo dataset modificado fue organizado y se le agregaron 10.000 laptops nuevas mediante python para aumentar la cantidad de datos a analizar

– *Generación y selección de alternativas de solución*

Alternativas para el empaquetado de la aplicación en contenedores

Empaquetado con Docker [2-5]

Pros:

- Aislamiento completo del entorno
- Portable en cualquier sistema que ejecute Docker
- Fácil de escalar con Kubernetes o Swarm

Contras:

- Requiere estar familiarizado con Docker, imágenes y contenedores
- Requiere que Docker este completamente configurado, sino puedo generar problemas

Máquina Virtual con Gestión Manual

Pros:

- Configuras todo según las necesidades específicas
- No requiere herramientas adicionales como Docker
- Fácil de implementar para configuraciones básicas

Contras:

- Difícil replicar el entorno en otras VM
- Requiere actualizaciones manuales
- Escalabilidad reducida

Máquina Virtual con Imágenes Preconfiguradas

Pros:

- La configuración está empaquetada en la imagen.
- Nuevas máquinas pueden iniciarse rápido usando la imagen.
- Garantiza que el entorno es consistente en cada VM.

Contras:

- Cambios en la aplicación requieren volver a generar y redistribuir la imagen.
- Las imágenes pueden ser grandes y difíciles de transferir.

Uso de Ansible, Puppet o Chef

Pros:

- Configuración declarativa asegura que los entornos sean consistentes
- Fácil de usar en múltiples VM simultáneamente
- Reduce el trabajo manual en configuraciones y despliegues

Contras:

- Requiere crear y probar scripts de configuración
- Curva de aprendizaje alta en herramientas con sintaxis y conceptos específicos

Empaquetado como Binario o Aplicación Standalone

Pros:

- No requiere instalar ni configurar dependencias adicionales
- Fácil despliegue
- No hay tiempos de configuración extensos

Contras:

- Difícil de adaptar a entornos dinámicos
- Limitado para servicios con múltiples dependencias
- Problemas al ejecutarse en diferentes sistemas operativos

Empaquetado con Snap o Flatpak

Pros:

- Instalación sencilla en sistemas compatibles.
- Las aplicaciones pueden actualizarse automáticamente.

Contras:

- No todos los sistemas operativos los soportan nativamente.
- Su curva de aprendizaje al crear paquetes requiere aprender nuevas herramientas.
- Puede ser complicado interactuar con otros servicios externos.

– Alternativas para el despliegue en un cluster de procesamiento de datos distribuido

Apache Spark Standalone Cluster [6-9]

Pros:

- Fácil de configurar y usar
- Ligero y no requiere sistemas adicionales
- Excelente para entornos pequeños

Contras:

- Carece de características de elasticidad y orquestación
- No recomendado para clusters de gran escala

Hadoop YARN (Yes Another Resource Negotiator)

Ventajas:

- Bien integrado con HDFS
- Escalabilidad en grandes clusters
- Soporte para empresas con ecosistema Hadoop

Contras:

- Configuración más compleja que apache standalone
- Más pesado en cuanto a recursos

Kubernetes

Pros:

- Escalabilidad y alta disponibilidad

- Ideal para clusters en la nube y entornos modernos
- Facil integracion con contenedores de docker

Contras:

- Curva de aprendizaje más pronunciada
- Requiere contenedores docker y configuración avanzada

– ***Servicios gestionados en la nube***

Amazon EMR: Servicio gestionado por AWS para clústeres de Spark y Hadoop

Pros:

- Configuración rápida de clusters
- Integración nativa con servicios de AWS

Contras:

- Dependencia de AWS
- Puede ser costoso si no se optimizan el uso de recursos

Google Cloud Dataproc: Servicio gestionado de clústeres Spark y Hadoop en Google Cloud

Pros:

- Escalabilidad rápida y sencilla
- Integración con herramientas de google

Contras:

- Dependencia de Google cloud
- Similar a EMR en términos de costos

Azure HDInsight: Servicio gestionado de Spark y Hadoop en Microsoft Azure

Pros:

- Buena integración con el ecosistema Azure
- Soporta múltiples frameworks

Contras:

- Depende de Azure
- Similar a EMR y Dataproc en costos

Databricks: Plataforma basada en Apache Spark que ofrece un entorno optimizado para análisis de datos y machine learning

Pros:

- Interfaz fácil para trabajar con spark
- Características avanzadas para análisis
- Integración con la nube

Contras:

- Requiere licencia
- Puede ser costoso

Definición de la arquitectura completa del sistema

La arquitectura completa que escogimos para este sistema para la parte del empaquetado y despliegue de la aplicación en contenedores fue el empaquetado con Docker ya que según sus pros y contras es la que mejor se adapta a la aplicación, aparte de que Docker es el componente de empaquetado que vimos en el curso y con el que estamos familiarizados, permitiendo escalar la aplicación y utilizar imágenes disponibles en Dockerhub mediante Swarm

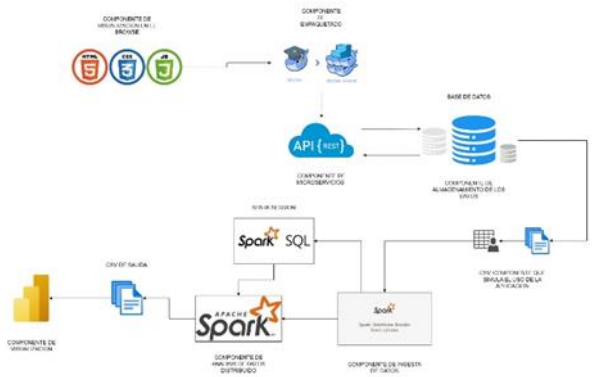
Para nuestra aplicación de procesamiento de datos distribuidos hemos escogido apache Spark Standalone Cluster [6-9] ya que Spark procesa y conserva los datos en la memoria RAM, sin escribir ni leer en el disco, lo que da como resultado velocidades de procesamiento mucho más rápidas, aparte de su de implementación y uso en máquinas virtuales como lo aprendimos es bastante fácil y cómodo.

El número de filas en nuestro dataset es de 12.099 siendo una cantidad fácil de procesar para Spark, si en un futuro decimos agregar una inmensa cantidad de filas a nuestro dataset simulando nuevos productos tampoco será problema para Spark

III. Diseño

– ***Propuesta del pipeline, componentes o algoritmos a utilizar***

– ***Diagrama de los componentes a utilizar***



– ***Relación y flujo de trabajo entre los componentes***

Componente de visualización en el browse: Es el HTML que permite al usuario ver la página de la tienda de la aplicación

Componente de empaquetado: Es el componente que permite el empaquetado de los microservicios junto con sus dependencias los cuales leen la información ingresada por el usuario en el HTML

Componente de almacenamiento de los datos: Permite que los microservicios puedan guardar cada uno su información

CSV (dataset): Es montado en la máquina virtual en un directorio donde se debe descomprimir, una vez extraído estará listo para ser leído por la app de procesamiento

Spark Dataframe Read: Lee el dataset y lo hace entendible para la app, (ejemplo: separa por comas e indica si hay cabecera en el dataset) transformándolo en dataframe

Spark SQL: Crea una Spark Session que se encarga generar las variables de tipo spark y lanza la aplicación

que será ejecutada la cual se puede ver en el dashboard de apache spark

Spark Standalone Cluster: Distribuye el trabajo de la aplicación en los diferentes nodos(worker) realiza los análisis de interés propuesto en el código y los guarda CSVs de salida en carpetas las cuales se guardan en el worker que realizó el trabajo

Csv De Salida: Guardan la información de los análisis realizados por el cluster, se usan los directorios compartidos de vagrant para tenerlos en nuestro sistema principal y así llevarlos a visualización

Componente De Visualización: Recibe los CSV de salida para su visualización en herramientas de creación de dashboards

- ***Descripción de los componentes***

Componente de visualización en el browse: Es el codigo HTML encargado de que el usuario pueda ver la página de la tienda

Componente de empaquetado: Es el componente que permite el empaquetado de la aplicación en contenedores

Componente de almacenamiento de los datos: Son las bases de datos SQL que permite a los microservicios guardar los datos de la aplicacion

CSV (dataset): Representa los datos de las laptops que se encuentran en formato CSV (Comma Separated Value). Es el punto de partido del flujo de procesamiento siendo una fundamental para el análisis en el cluster

Componente De Ingesta De Datos(Spark Dataframe): Se encarga de cargar y leer los datos del CSV usando apache Dataframe Reader para que la aplicación los pueda procesar más fácilmente

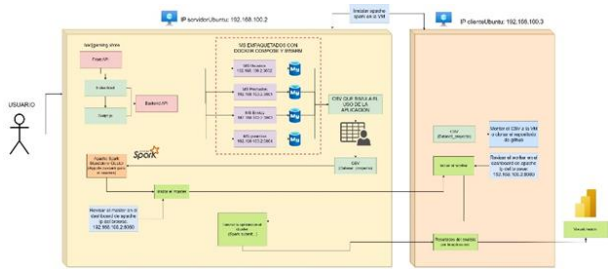
Spark SQL: Se encarga de habilitar las variables de tipo spark para realizar consultas SQL sobre el Dataframe, también permite construir la aplicación la cual se puede ver en browse en el dashboard de apache spark

Componente De Análisis (Cluster Spark Standalone): Es el cluster de Apache Spark configurado en modo solitario o standalone, su función es realizar el procesamiento distribuido de los datos en los nodos correspondientes. Ejecuta los análisis de interés definidos en el código y procesa grandes volúmenes de datos o en este caso nuestro dataset

CSV De Salida: Son los archivos CSVs resultantes del procesamiento y análisis de datos los cuales se exportan de nuevo a formato CSV y se guardan en carpetas distintas en el nodo que realizó la ejecución de la app

Componente De Visualización: Es el encargado de consumir los CSVs de salida resultantes del análisis generando gráficos, reportes u otras representaciones visuales

- ***Diagrama de despliegue***



IV. Implementación

Empaquetado

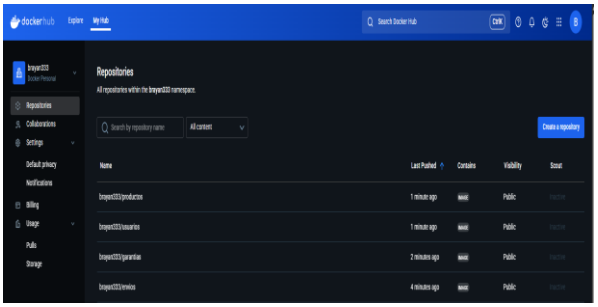
Docker Compose

```
vagrant@servidorUbuntu:~/proyecto-REI$ sudo docker-compose ps
    Name                  Command               State                    Ports
-------------------------------------------------------------------------------------------------
envios_ms       docker-entrypoint.sh node  ...   Up           0.0.0.0:3003->3003/tcp,:::3003->3003/tcp
garantias_ms    docker-entrypoint.sh node  ...   Up           0.0.0.0:3004->3004/tcp,:::3004->3004/tcp
productos_ms    docker-entrypoint.sh node  ...   Up           0.0.0.0:3001->3001/tcp,:::3001->3001/tcp
proyecto_mysql  docker-entrypoint.sh mysqld    Up (healthy)   0.0.0.0:3306->3306/tcp,:::3306->3306/tcp, 33060/tcp
usuarios_ms     docker-entrypoint.sh node  ...   Up           0.0.0.0:3002->3002/tcp,:::3002->3002/tcp
```

Docker Swarm

```
vagrant@servidorUbuntu:~/proyecto-REI$ sudo docker stack services proyecto-rei
ID             NAME                       MODE         REPLICAS   IMAGE                          PORTS
b2qmefy1mkbp   proyecto-rei_envios        replicated   1/1        brayan333/envios:latest        *:3003->3003/tcp
qoiq8qsnge2u   proyecto-rei_garantias      replicated   1/1        brayan333/garantias:latest     *:3004->3004/tcp
l82b7ucd7gll   proyecto-rei_mysql         replicated   1/1        mysql:8.0                      *:3306->3306/tcp
jl46639nmeoqu  proyecto-rei_productos     replicated   1/1        brayan333/productos:latest     *:3001->3001/tcp
gftw4llxgvzx   proyecto-rei_usuarios      replicated   1/1        brayan333/usuarios:latest      *:3002->3002/tcp
vagrant@servidorUbuntu:~/proyecto-REI$
```

Docker hub



Escalar Swarm

```
vagrant@servidorUbuntu:~/proyecto-REI$ sudo docker service scale proyecto-rei_usuarios=3
proyecto-rei_usuarios scaled to 3
overall progress: 3 out of 3 tasks
1/3: running   [==================================================>]
2/3: running   [==================================================>]
3/3: running   [==================================================>]
verify: Service proyecto-rei_usuarios converged
```

Ejemplos

```
Calle 123 #45-67"}]vagrant@servidorUbuntu:~/proyecto-REI$ curl 192.168.100.2:3002/usuarios
[{"nombre":"Ana García","email":"ana.garcia@example.com","usuario":"anag23","password":"12345","telefono":"3101234567","cedula":"1234567890123","direccion":"
Calle 123 #45-67"},{"nombre":"jose","email":"jose@example.com","usuario":"jose23","password":"54321","telefono":"310155555","cedula":"12345","direccion":"Col
ombia"}]vagrant@servidorUbuntu:~/proyecto-REI$ curl 192.168.100.2:3001/productos
```

```
ombia"}]vagrant@servidorUbuntu:~/proyecto-REI$ curl 192.168.100.2:3001/productos
[{"id":1,"nombre":"Laptop Gamer Pro","estado":"Nuevo","marca":"TechBrand","modelo":"Y-5000","cpu":"Intel Core i9-12900H","ram":"32GB DDR5","gb_almacenaient
o":"1000GB","tipo_almacenamiento":"SSD NVMe","gpu":"NVIDIA RTX 3080","pantalla":"15.6\" 4K OLED","es_tactil":"No","precio":"1099.99","garantia":"24 meses","
stock":"50"},{"id":2,"nombre":"MSI RAPTOOR","estado":"Nuevo","marca":"TechBrand","modelo":"Y-5000","cpu":"Intel Core i9-12900H","ram":"32GB DDR5","gb_alnace
namiento":"1000GB","tipo_almacenamiento":"SSD NVMe","gpu":"NVIDIA RTX 3080","pantalla":"15.6\" 4K OLED","es_tactil":"No","precio":"5000.99","garantia":"24 m
eses","stock":"5"}]vagrant@servidorUbuntu:~/proyecto-REI$
```

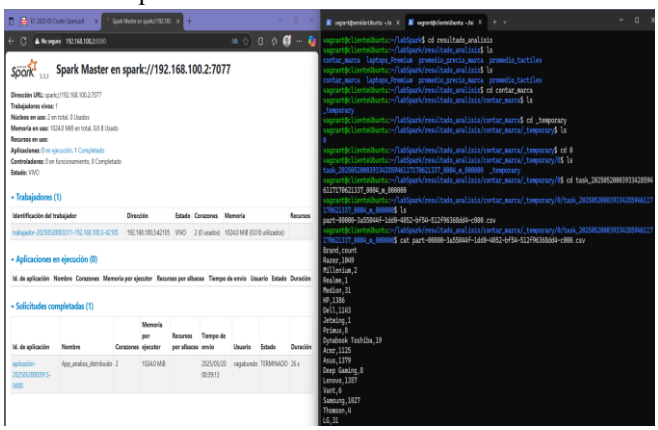
```
vagrant@servidorUbuntu:~/proyecto-REI$ curl 192.168.100.2:3003/envios
[]vagrant@servidorUbuntu:~/proyecto-REI$
```

Segunda parte: Desarrollo de un cluster de procesamiento de análisis de datos distribuido

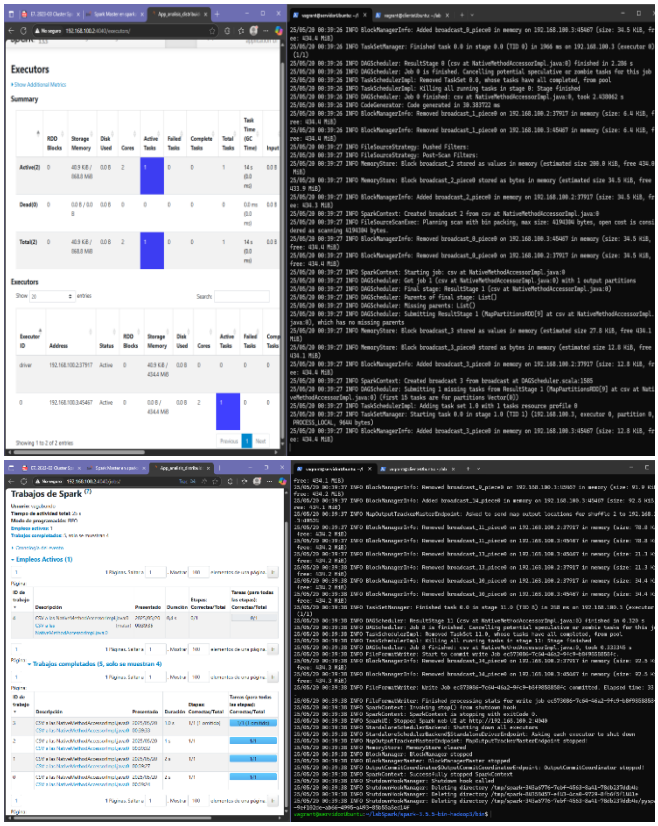
- ***Dataset aplicación de análisis de datos distribuido***

Para mayor comodidad puede mirar, descargar los archivos o clonar el repositorio de GitHub [9-11] en donde se encuentra la parte del análisis distribuido.

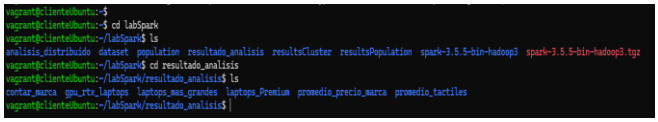
Dashboard apache



Información del proceso de ejecución de la app de análisis (hptt://192.168.100.2:4040)



Resultados

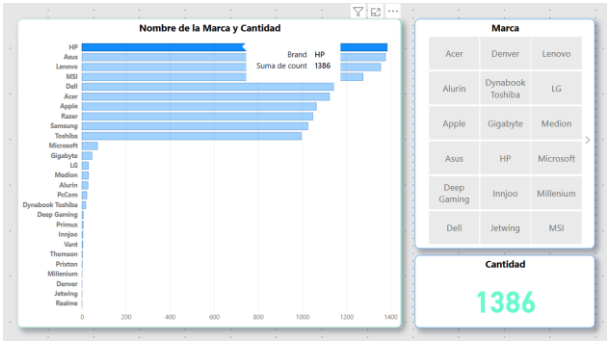


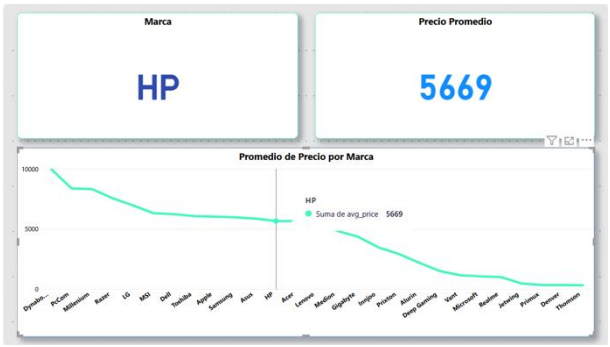
Directorio compartido de windows

Nombre	Fecha de modificación	Tipo	Tamaño
.vagrant	17/02/2025 10:34 a. m.	Carpeta de archivos	
2019-Dec	2/05/2025 7:18 p. m.	zip	75.546 KB
laptops_csv_REI	19/05/2025 7:01 p. m.	zip	119 KB
mynew.box	18/02/2025 10:14 p. m.	Archivo BOX	1.036.004 KB
part-00000-1c5cf02e-22a7-479a-ad00-e7...	19/05/2025 8:05 p. m.	Archivo de valores...	245 KB
part-00000-3e21bf72-72af-4bc8-b4fc-4e2...	19/05/2025 8:05 p. m.	Archivo de valores...	24 KB
part-00000-9a0c009c-a435-4079-aa20-d5...	19/05/2025 8:05 p. m.	Archivo de valores...	55 KB
part-00000-3555db01-da26-4cab-aeed-7...	19/05/2025 8:05 p. m.	Archivo de valores...	1 KB
part-00000-5105e26a-f5c7-49cd-9aae-2b...	19/05/2025 8:05 p. m.	Archivo de valores...	1 KB
part-00000-bb2e493e-9f4e-411f-9f9f-90b...	19/05/2025 8:05 p. m.	Archivo de valores...	1 KB
Vagrantfile	17/02/2025 10:53 a. m.	Archivo	1 KB
Vagrantfile	17/02/2025 10:53 a. m.	Documento de te...	1 KB
world_population	2/05/2025 9:21 p. m.	zip	16 KB

Se copiaron todos los archivos CSV de los análisis realizados por Spark al directorio compartido con Windows para después poderlo llevar a un componente de visualización con PowerBI, Tableau entre otros

– *Visualización de los archivos CSV de los análisis recolectados*





Referencias

[1] J. Merino Bermejo, “Laptops Price Dataset,” Kaggle. [Enlínea]. Disponible: https://www.kaggle.com/datasets/juanmerinobermejo/laptops-price-dataset. [Accedido: 22-may-2025]

Fuentes sobre docker

[2] Openinnova, “¿Qué es Docker en español? Ventajas y desventajas de utilizarlo,” Openinnova, 2023. [En línea]. Disponible: https://www.openinnova.es/que-es-docker-espanol-ventajas-y-desventajas-de-utilizarlo/. [Accedido: 22-may-2025]

[3] Asimov, “Docker: Ventajas y desventajas en el desarrollo de proyectos Angular,” Asimov Cloud. [En línea]. Disponible:https://asimov.cloud/blog/programacion-5/docker-ventajas-y-desventajas-en-el-desarrollo-de-proyectos-angular-196. [Accedido: 22-may-2025]

[4] Datacamp, “Kubernetes vs Docker,” Datacamp. [En línea]. Disponible: https://www.datacamp.com/es/blog/kubernetes-vs-docker. [Accedido: 22-may-2025]

Fuentes de información sobre spark

[5] IBM, “¿Qué es Apache Spark?” IBM. [En línea]. Disponible: https://www.ibm.com/es-es/topics/apache-spark. [Accedido: 22-may-2025]

[6] Datacamp, “Hadoop vs Spark,” Datacamp. [En línea]. Disponible: https://www.datacamp.com/es/blog/hadoop-vs-spark. [Accedido: 22-may-2025]

[7] Amazon Web Services, “Diferencias entre Hadoop y Spark,” AWS. [En línea]. Disponible: https://aws.amazon.com/es/compare/the-difference-between-hadoop-vs-spark/. [Accedido: 22-may-2025]

[8] Education Wiki, “Apache Spark Architecture,” Education Wiki. [En línea]. Disponible: https://es.education-wiki.com/2394836-apache-spark-architecture. [Accedido: 22-may-2025]

Repositorio Docker hub donde están las imágenes del api

[9] B. Tigreros, “Repositorio de imágenes Docker,” Docker Hub. [En línea]. Disponible: https://hub.docker.com/repositories/brayan333. [Accedido: 22-may-2025].

[10] B. Tigreros, “API-backend,” GitHub. [En línea]. Disponible: https://github.com/BrayanTigreros/API-backend. [Accedido: 22-may-2025].

[11] B. Tigreros, “App de análisis distribuido,” GitHub. [En línea]. Disponible: https://github.com/BrayanTigreros/app_analisis. [Accedido: 22-may-2025].