

Trabajo Grupal | Modulo Infraestructura

Proyecto Despliegue Página Web

Reseñas de Carros

Integrantes:

Juan José Grueso
Juan Sebastián Torres
Joseph Marín Parra
Juan David Meza



www.super-cars.co

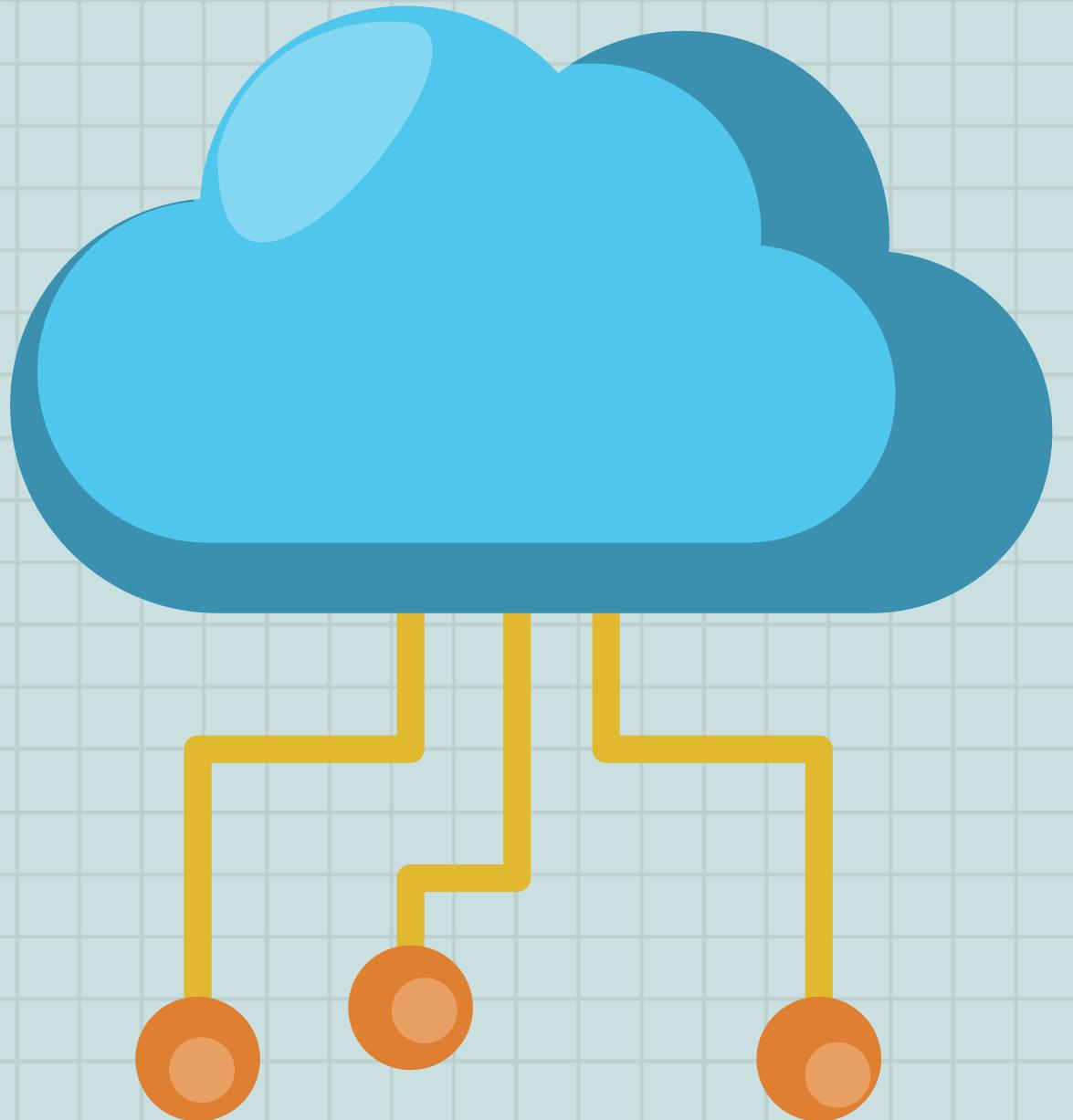


Contenido

- 1. Introducción
- 2. Análisis
- 3. Diseño
- 4. Implementación
- 5. Pruebas
- 6. Conclusiones



Introducción



Proyecto enfocado en el despliegue de una aplicación web previamente diseñada para reseñar automóviles nuevos y usados.

Integra una arquitectura basada en microservicios.

El objetivo principal es ofrecer una plataforma escalable, moderna y analítica, que combine interacción de usuarios con procesamiento de información.



Análisis

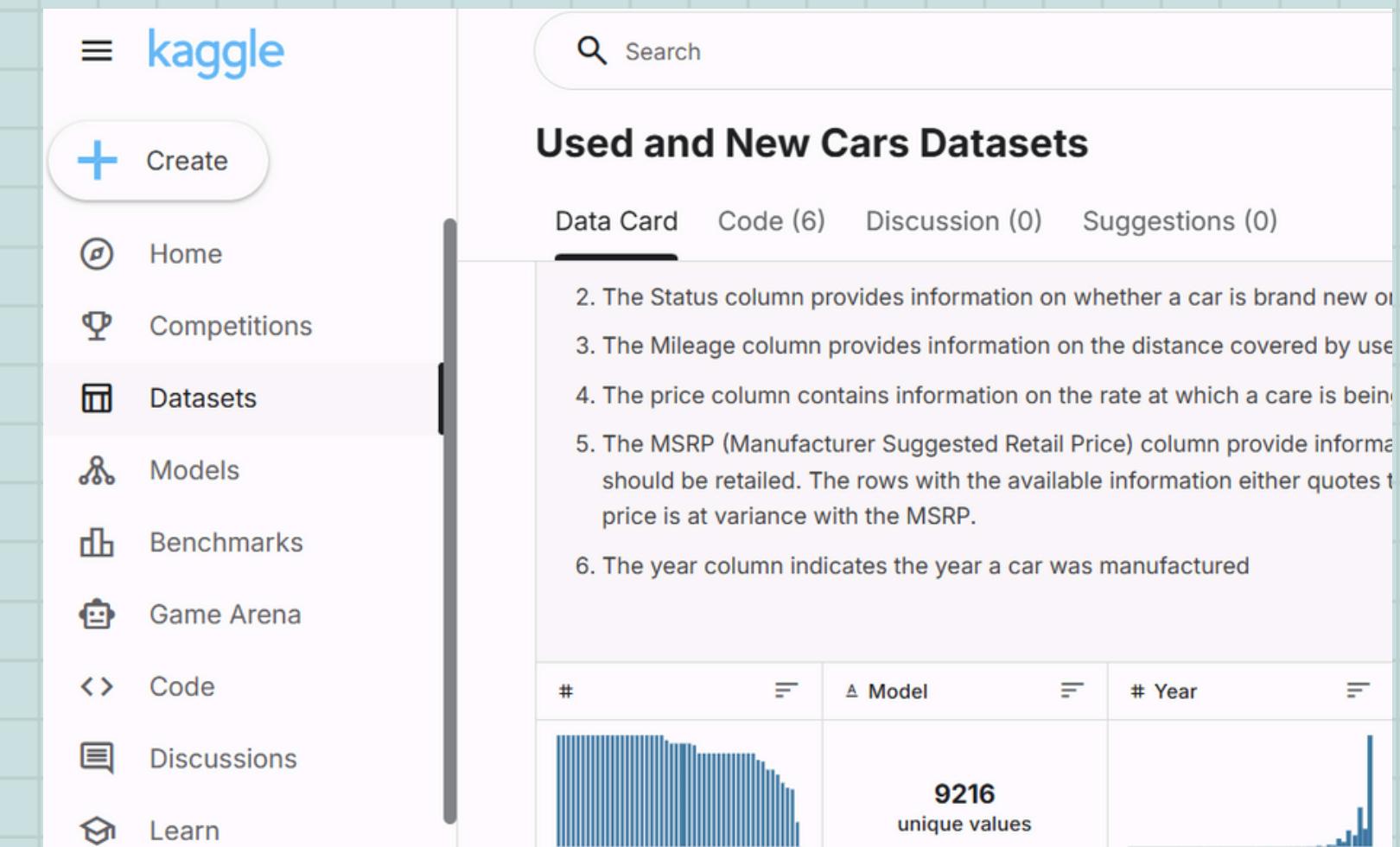


Selección del Dataset Objeto

- Se seleccionó como dataset el Used and New Cars Dataset, disponible públicamente en la plataforma Kaggle.
- Este conjunto de datos contiene información detallada sobre vehículos nuevos y usados, incluyendo variables como marca, modelo, año, precio, kilometraje, MSRP y estado del vehículo.

¿Por qué Kaggle junto con este dataset y no otro?

Al Kaggle ser una web que brinda datos abiertos al público nos permitió elegir el que mas se adecuara a lo que queríamos mostrar en la página.



Selección de Tecnologías

Se evaluaron las tecnologías y arquitecturas más adecuadas para garantizar el rendimiento, la escalabilidad y la integración del sistema.

El proyecto se basa en dos entornos principales:

- Aplicación web con microservicios desplegados en contenedores Docker.
- Clúster de análisis de datos distribuido con Apache Spark.



TABLA COMPARATIVA

Característica	Alternativa	Selección	Puntos a Favor selección
Orquestación	Kubernetes	Docker	Despliegue ligero. Ya usado anteriormente.
Procesar los Datos	Hadoop	Spark	Procesa datos en Memoria
Visualización	Graficos en Excel	Graficos en la Web	Fácil acceso (vía web)

Comparación de tecnologías y justificación de las herramientas seleccionadas.

Definición de la arquitectura

Arquitectura compuesta por dos entornos principales:

1. Aplicación web con microservicios (Usuarios, Vehículos, Reseñas y Calificaciones).
2. Clúster de análisis de datos con Apache Spark (Maestro y Workers).
3. Comunicación mediante API REST.
4. Despliegue en contenedores Docker orquestados con Swarm.
5. Uso de HAProxy como balanceador de carga para mejorar disponibilidad y rendimiento.

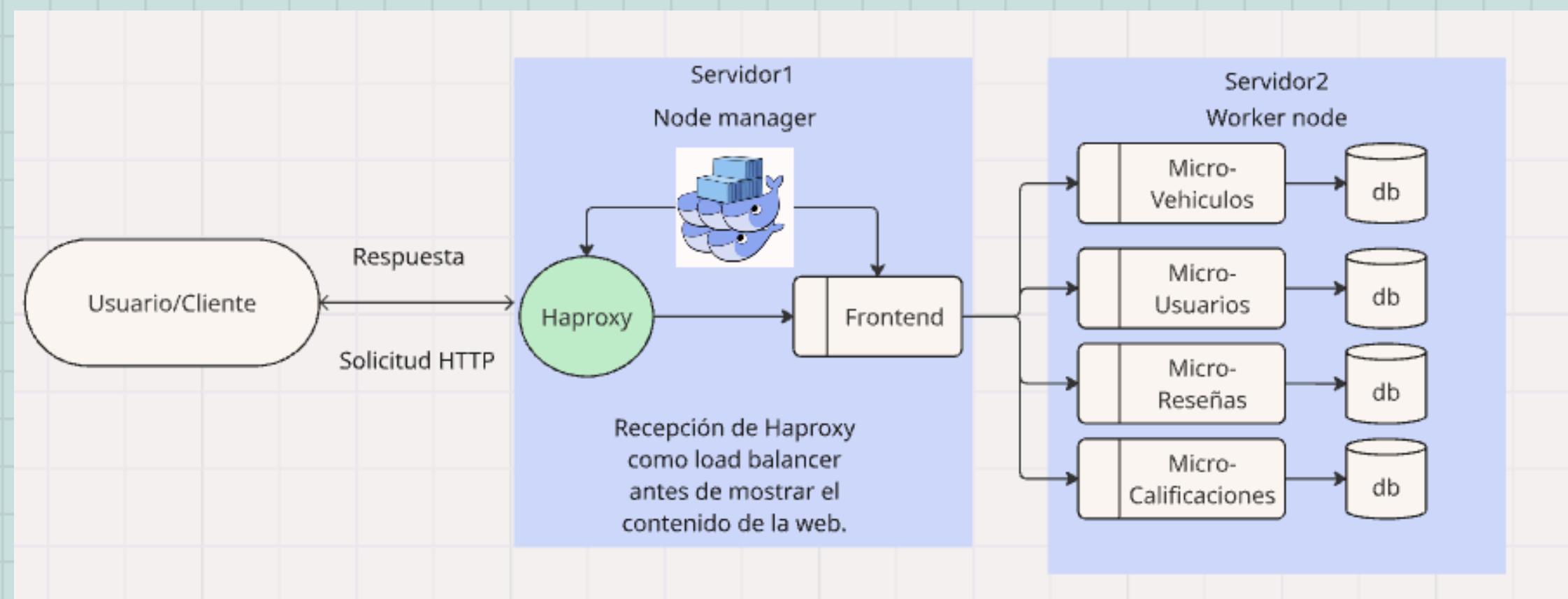
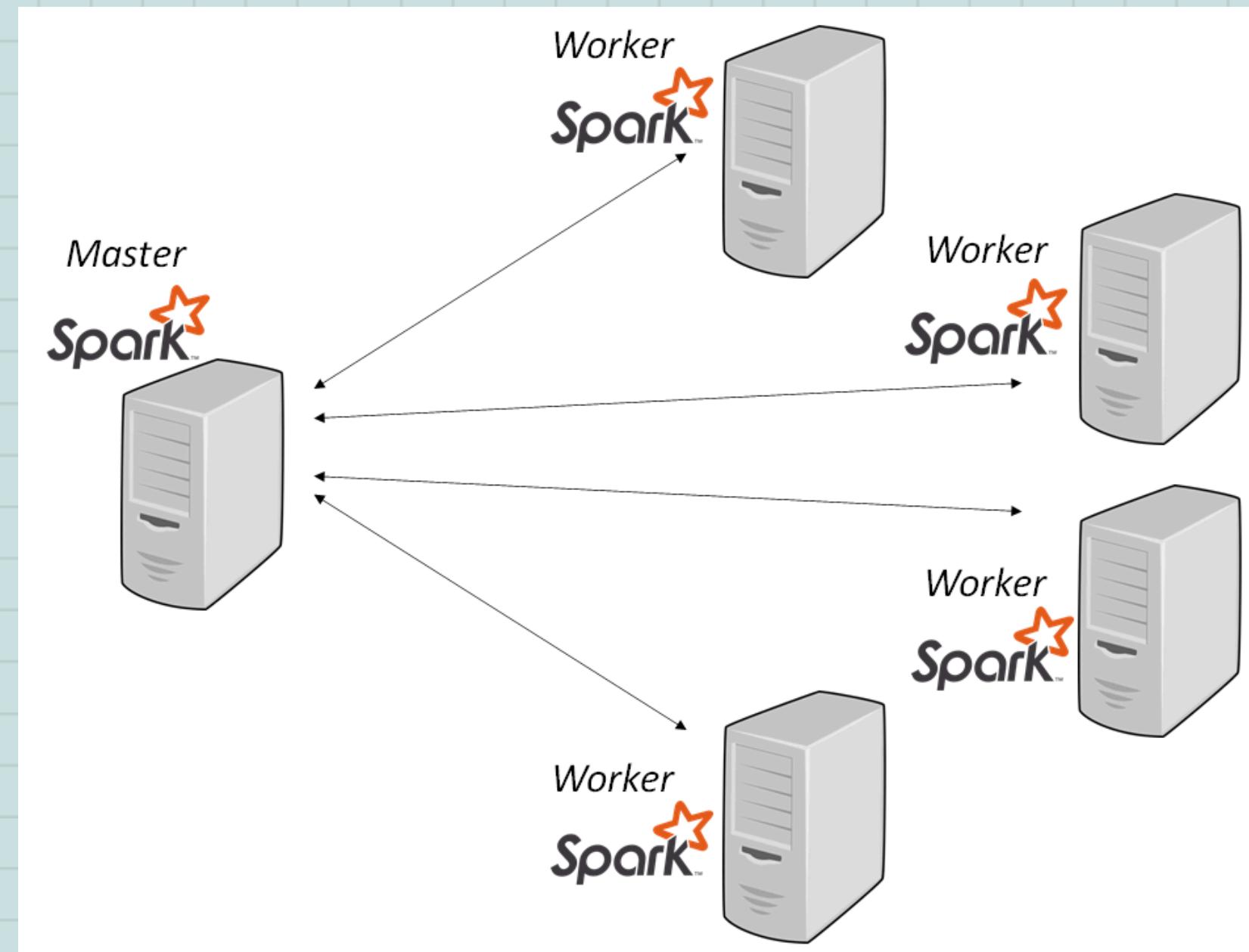


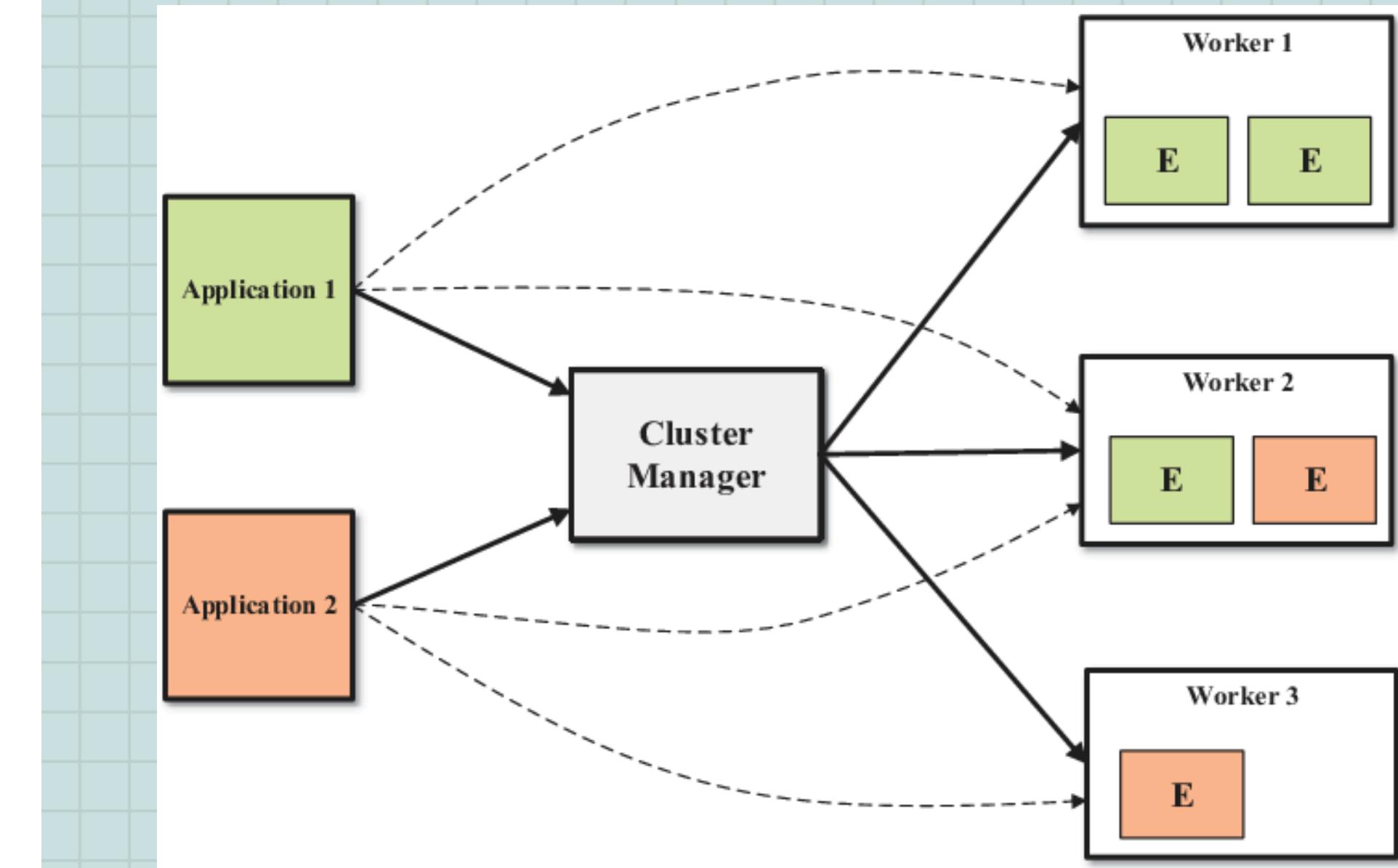
Fig. 1. Representación grafica de la comunicación entre el usuario y los servidores.

Cluster orquestado con Swarm

Cluster Analítica de Datos



*Fig. 2. Representación grafica
del Cluster Spark.*



*Fig. 3. Representación grafica de la
comunicación interna del Cluster.*

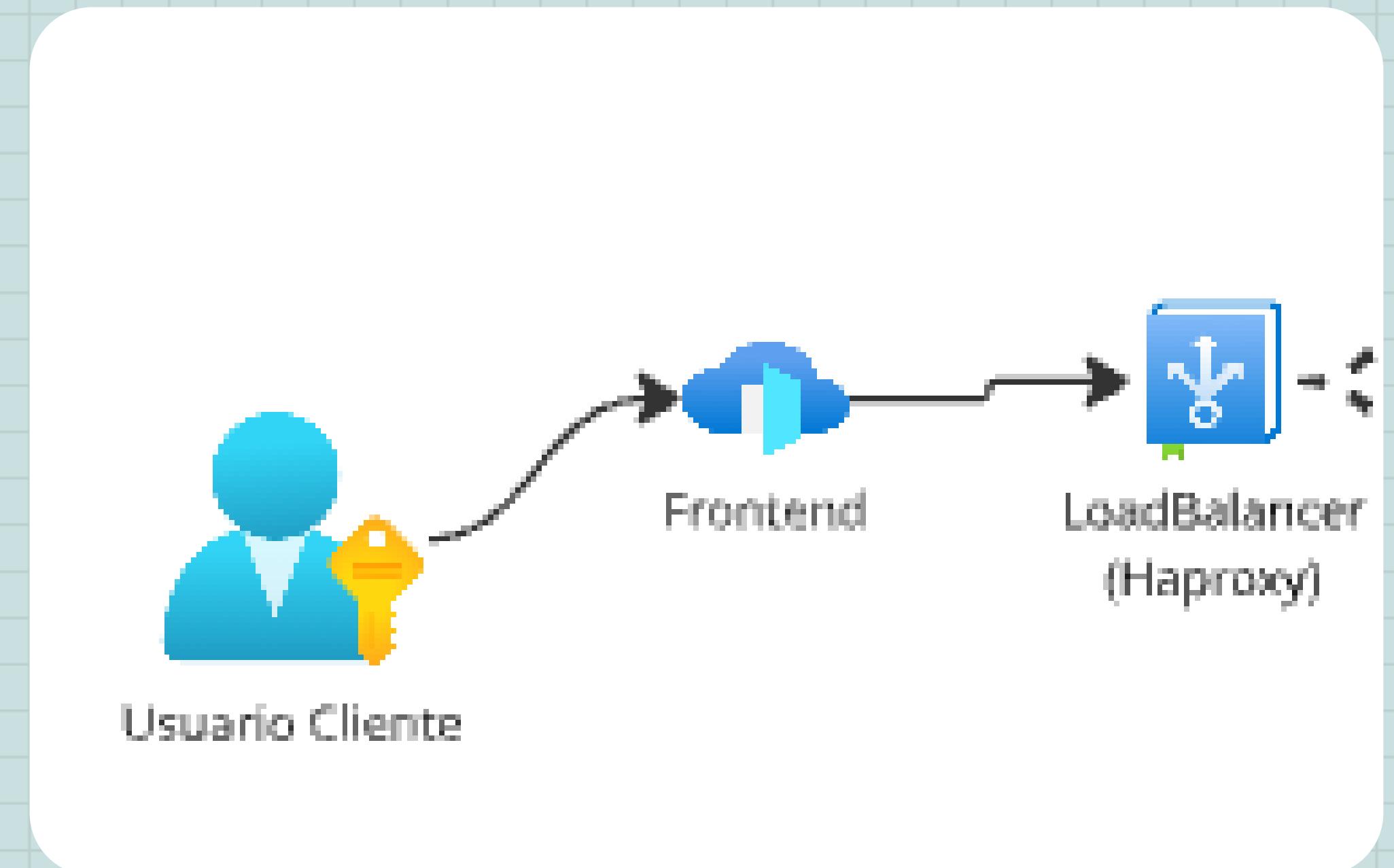
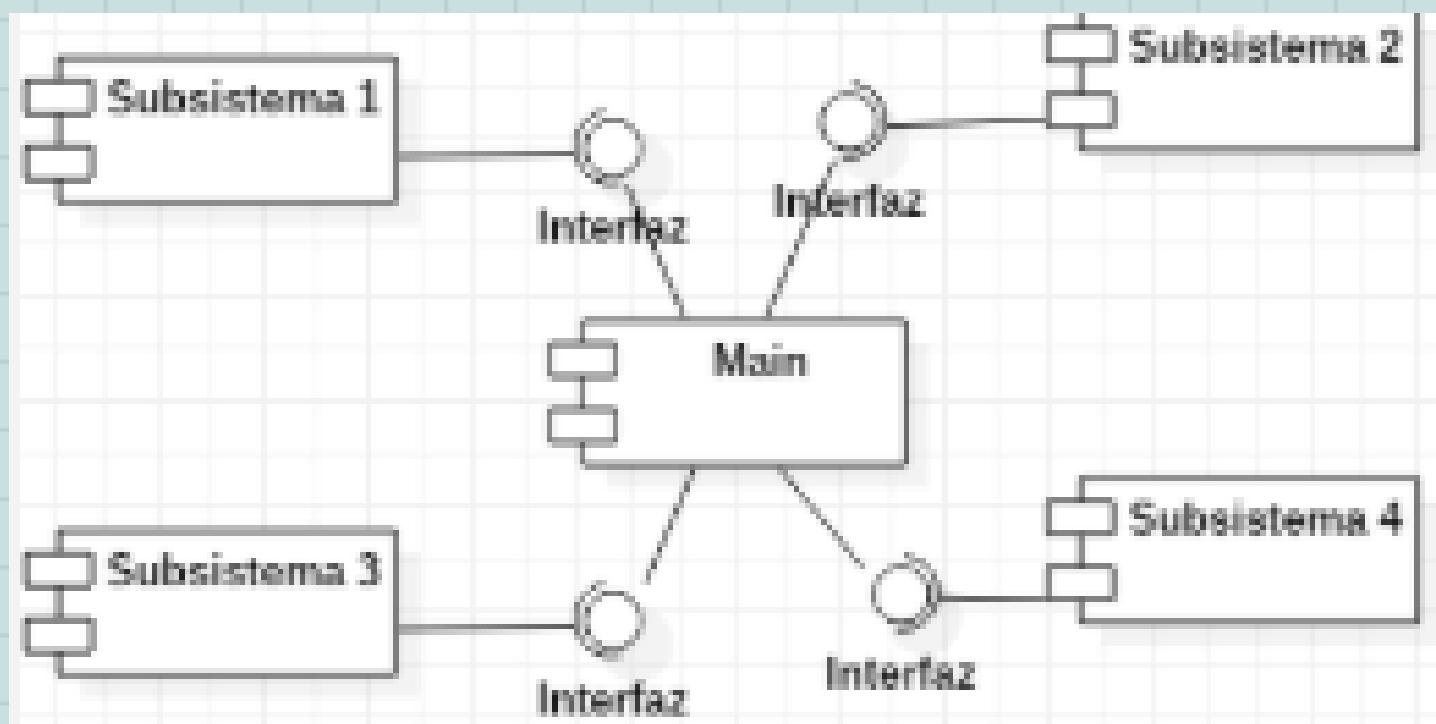


Diseño



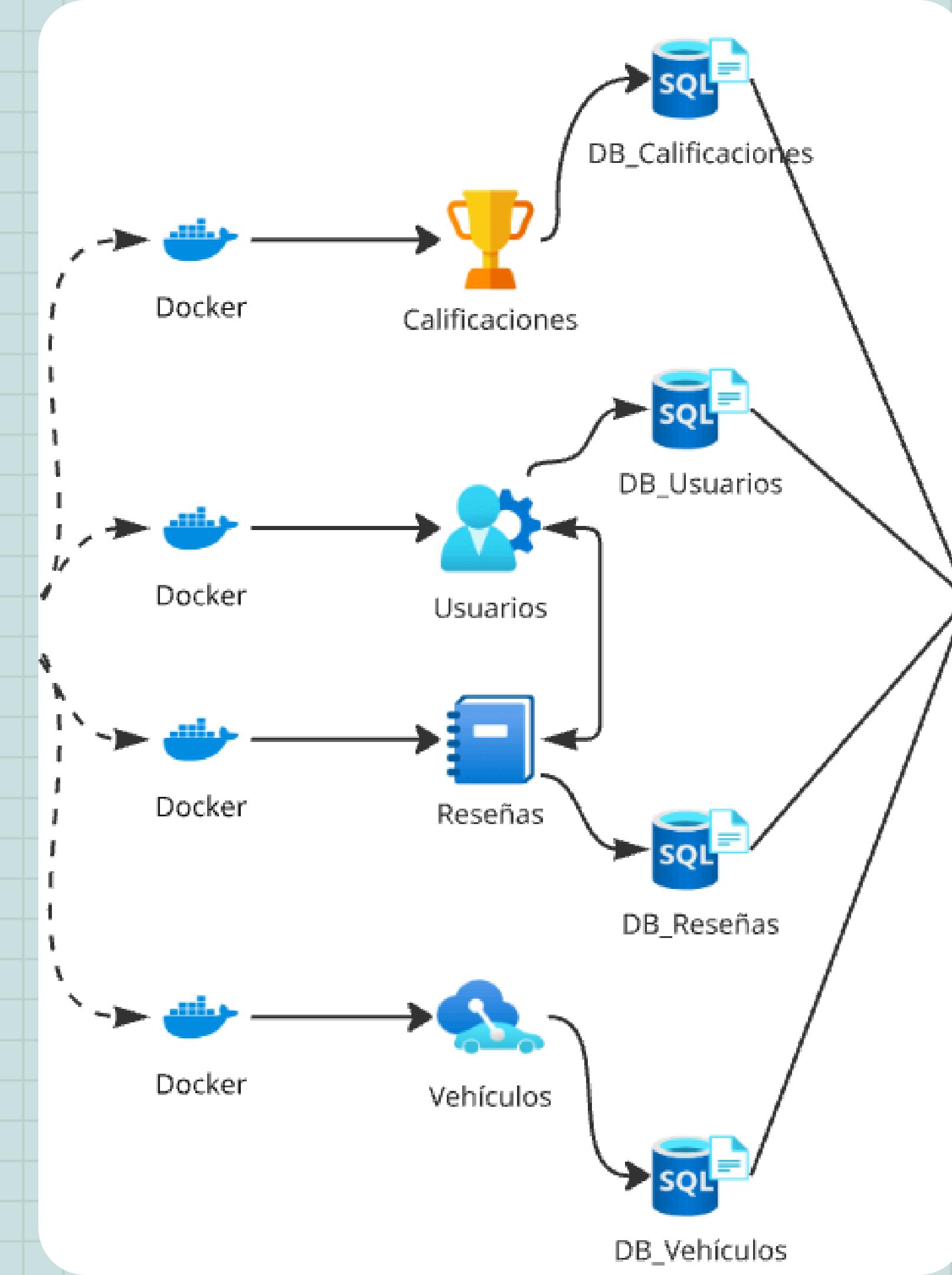
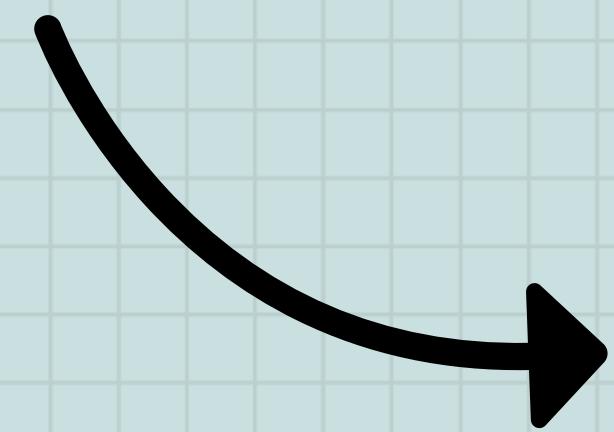
Diagrama de componentes

Sirve para visualizar la estructura de un sistema complejo, entender cómo sus partes se conectan y se organizan, y comunicar el diseño a los interesados.



Visualización más amplia

La 'comunicación'

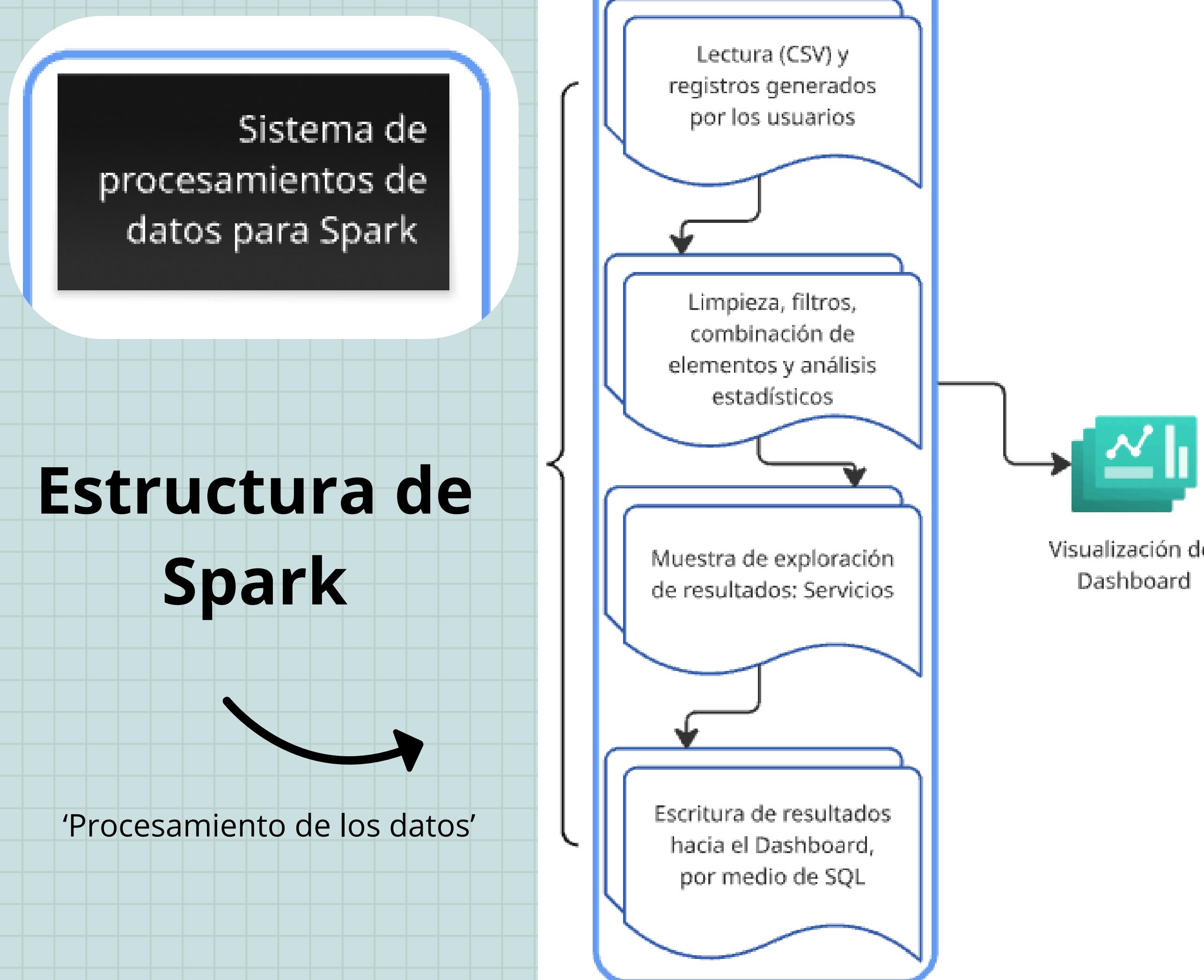


¿Qué hace Spark?

1. Carga los datos del CSV y del dataset de Kaggle.
2. Limpia y combina toda la información.
3. Analiza los datos y genera métricas.
4. Envía los resultados procesados al dashboard.

Estructura de Spark

'Procesamiento de los datos'

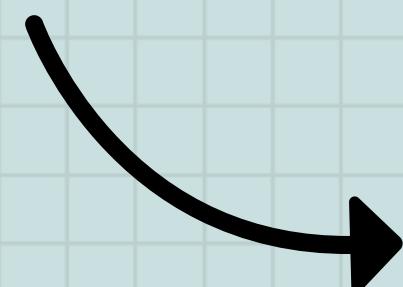


Descripción de los componentes

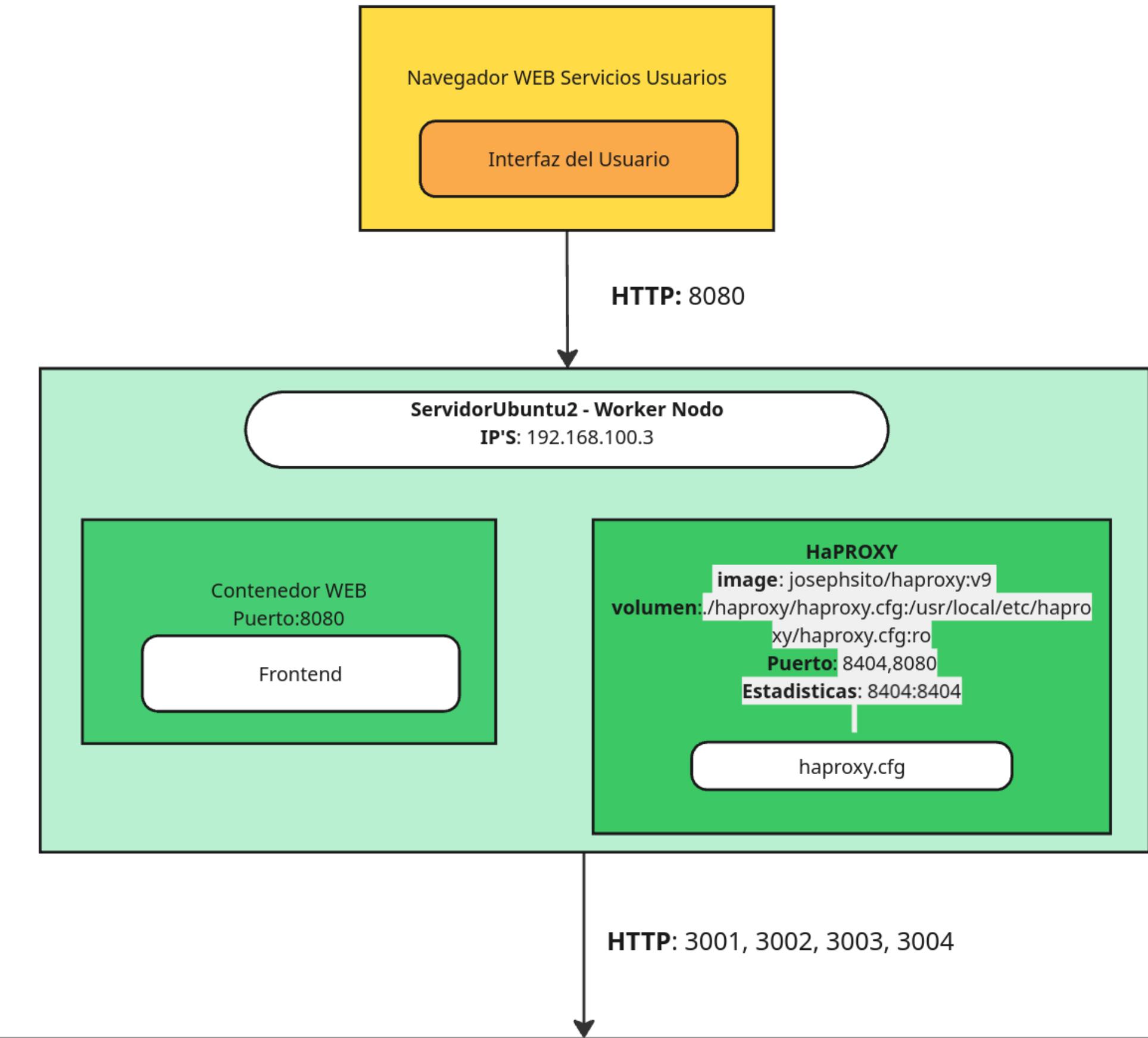
	Componente	Descripción	Datos manejados
1	Frontend Web	Interfaz visual donde los usuarios reseñan y califican vehículos.	Datos de reseñas, calificaciones y consultas.
2	HAProxy	Balancea la carga y enruta /api/* a...	Tráfico HTTP.
3	Microservicio de Vehículos	Catálogo técnico/comercial.	Marca, modelo, año, precio, combustible, km.
4	Microservicio de Usuarios	Autenticación y gestión de usuarios.	Nombre, correo, hash de contraseña, ID.
5	Microservicio de Reseñas	Comentarios de usuarios por...	Microservicio de Calificaciones
6	Microservicio de Calificaciones	Calificación 1..5 y promedio por vehículo.	Calificaciones numéricas y agregados.
7	Spark Master	Coordina trabajos distribuidos.	Metadatos del procesamiento.
8	Spark Workers	Ejecutan transformaciones/joins...	Particiones del dataset Kaggle.
9	Dataset Kaggle	Fuente de datos de vehículos nuevos/usados.	CSV con marca, año, precio, kilometraje.
10	MySQL	Persistencia transaccional.	Tablas: usuarios, vehiculos, resenas, calificaciones, califs_agg.
11	Dashboard / Power BI	Visualiza promedios, conteos, tendencias.	Resultados procesados (tabla califs_agg).

Diagrama de despliegue

El diagrama de despliegue muestra cómo se distribuye el sistema entre los servidores, qué componentes se ejecutan en cada nodo y cómo se comunican entre sí dentro del clúster Docker Swarm.

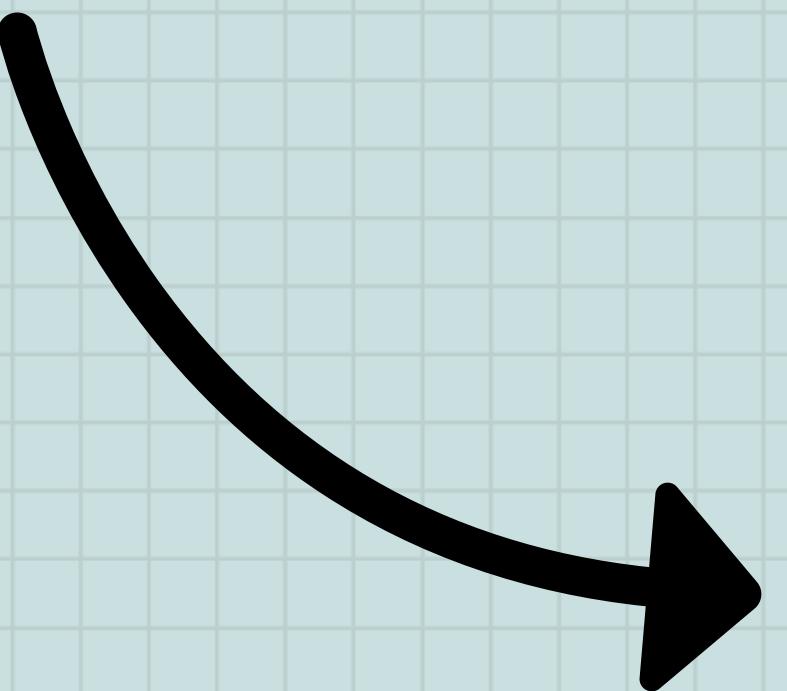


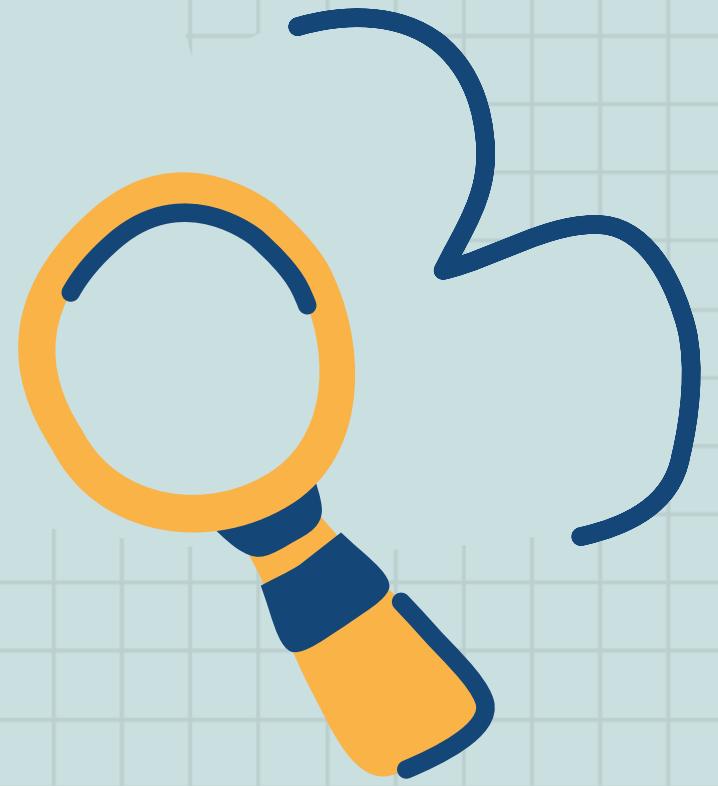
'Distribuidor'





'El corazón del Cluster'

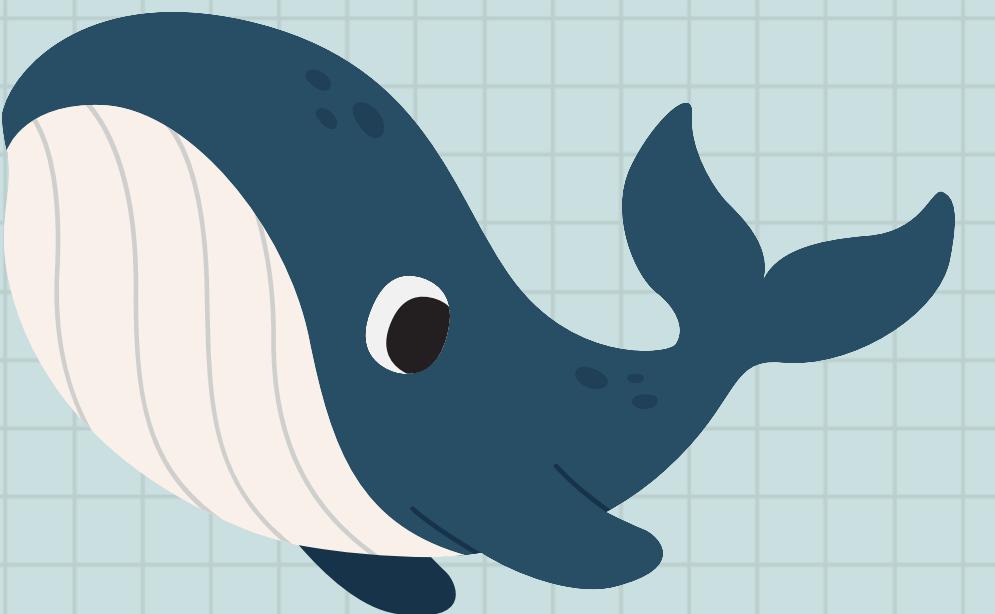




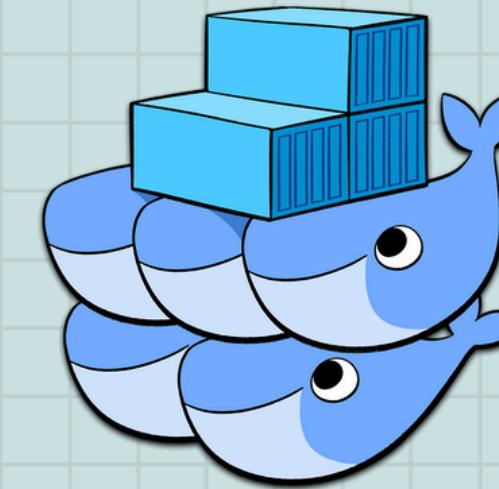
Implementación y pruebas



Despliegue con Docker



docker



docker swarm

Nov 13 18:21

0.00 es ▾

DiplomaMasterwireless1-2 X Joseph46832/proyecto-docker X Presentación Circuitos el... X

github.com/Joseph46832/proyecto-docker/main

Joseph46832 / proyecto-docker

Type [?] to search

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

projeto-docker Public

main 1 Branch 0 Tags

Go to file Add file Code

About

No description, website, or topics provided.

Readme Activity 0 stars 0 watching 0 forks

Commits

Joseph46832 Add Docker installation steps to README 3a8c566 · 1 minute ago 29 Commits

MS-Calificaciones Actualizar proyecto: cambios en MS, bases de datos, fron... 3 days ago

MS-Resenas Actualización de bases de datos y microservicio de Resen... 4 hours ago

MS-Usuarios Update index.js 2 days ago

MS-Vehiculos Update CORS settings and log message in index.js 2 days ago

databases Actualización de bases de datos y microservicio de Resen... 4 hours ago

frontend Update API URLs to new host IP 2 days ago

haproxy Actualizar proyecto: cambios en MS, bases de datos, fron... 3 days ago

.gitattributes Agregar CSV grande con Git LFS 3 days ago

README.md Add Docker installation steps to README 1 minute ago

Vagrantfile Add files via upload 2 days ago

docker-compose.yml Actualización de bases de datos y microservicio de Resen... 4 hours ago

README

Pagina web - Reseñas de Carros

A continuación se encontrará el paso a paso a reproducir para el lanzamiento de nuestra página web usando Docker.

About

No description, website, or topics provided.

Commits

Joseph46832 Add Docker installation steps to README 3a8c566 · 1 minute ago 29 Commits

MS-Calificaciones Actualizar proyecto: cambios en MS, bases de datos, fron... 3 days ago

MS-Resenas Actualización de bases de datos y microservicio de Resen... 4 hours ago

MS-Usuarios Update index.js 2 days ago

MS-Vehiculos Update CORS settings and log message in index.js 2 days ago

databases Actualización de bases de datos y microservicio de Resen... 4 hours ago

frontend Update API URLs to new host IP 2 days ago

haproxy Actualizar proyecto: cambios en MS, bases de datos, fron... 3 days ago

.gitattributes Agregar CSV grande con Git LFS 3 days ago

README.md Add Docker installation steps to README 1 minute ago

Vagrantfile Add files via upload 2 days ago

docker-compose.yml Actualización de bases de datos y microservicio de Resen... 4 hours ago

README

Packages

No packages published Publish your first package

Languages

JavaScript 67.6% CSS 16.2%

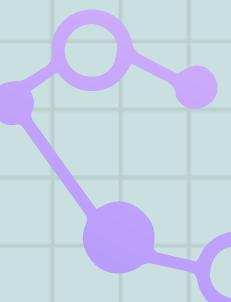
HTML 12.1% Dockerfile 4.1%

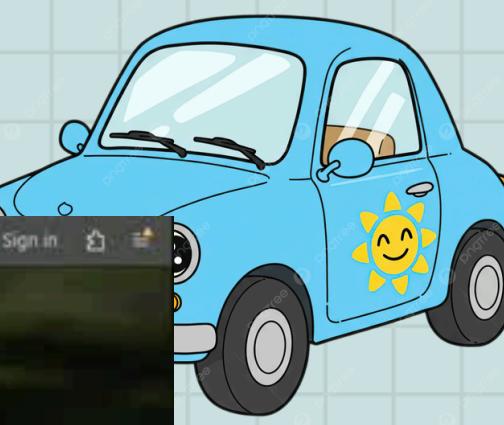
Suggested workflows

Based on your tech stack

Publish Node.js Configure

Package to GitHub





Not Secure http://192.168.100.3:8080/index.html

150% Sign in

a elegir con seguridad, confianza y tranquilidad.

Unirme ahora

Algunos de nuestros carros

Puedes valorar según las especificaciones del carro y el estado en que se encuentran

2022 Acura TLX A-Spec
2022

New • NaN km

\$NaN

[Ver detalles](#)

2023 Acura RDX A-Spec
2023

New • NaN km

\$NaN

[Ver detalles](#)

2023 Acura TLX Type S 2023

New • NaN km

\$NaN

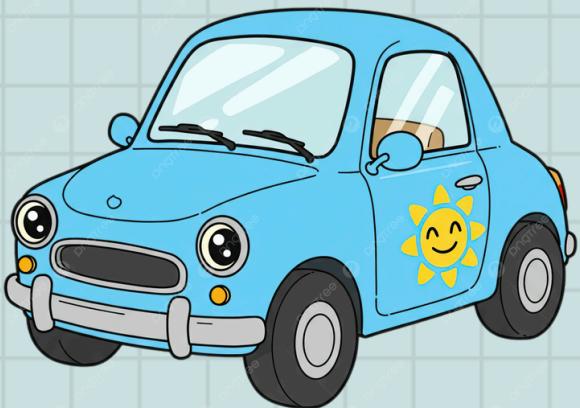
[Ver detalles](#)

2023 Acura TLX Type S 2023

New • NaN km

\$NaN

[Ver detalles](#)



Pruebas

A continuacion vamos a hacerle solicitudes al microservicio de reseñas sin escalar nada (replicas):

- Prueba con 2 réplicas
 - Tiempo total: 16.817 segundos
 - Fallos: 0
 - Requests por segundo: 59.46
 - Tiempo por solicitud: 840 ms (promedio)
- Conclusión:

A dos réplicas, el sistema es estable completamente. No hubo fallos, el tiempo de respuesta fue rapido y el servicio manejó la carga sin problemas.

Vamos a utilizar ApacheBench el cual se trata de lo siguiente
ab -n 1000 -c 50 <http://192.168.100.2:8080/resenas/>

ab: Es el comando

-n: Aqui van el numero de solicitudes que uno le va a pedir al microservicio

-c: Hasta 50 solicitudes al mismo tiempo. Esto simula que 50 usuarios hacen la petición concurrentemente.

Url: Es el lugar a donde apunta para hacer las solicitudes

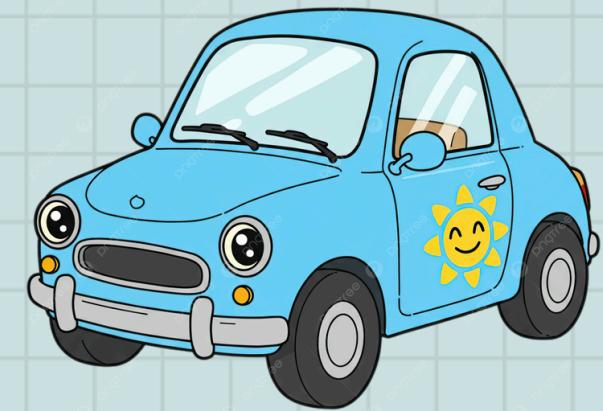
```
No VM guests are running outdated hypervisor (qemu) binaries on this host.  
root@servidorUbuntu1:/vagrant# ab -n 1000 -c 50 http://192.168.100.2:8080/resenas/  
This is ApacheBench, Version 2.3 <$Revision: 1879490 $>
```

```
Concurrency Level: 50  
Time taken for tests: 16.817 seconds  
Complete requests: 1000  
Failed requests: 0  
Total transferred: 327113000 bytes  
HTML transferred: 326862000 bytes  
Requests per second: 59.46 #[/sec] (mean)  
Time per request: 840.865 [ms] (mean)  
Time per request: 16.817 [ms] (mean, across all concurrent requests)  
Transfer rate: 18995.10 [Kbytes/sec] received
```

```
Connection Times (ms)  
min mean[+/- sd] median max
```

Pruebas

Ahora haremos solicitudes a reseñas a tan solo 1 replica:



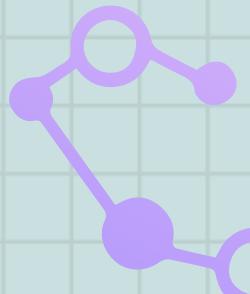
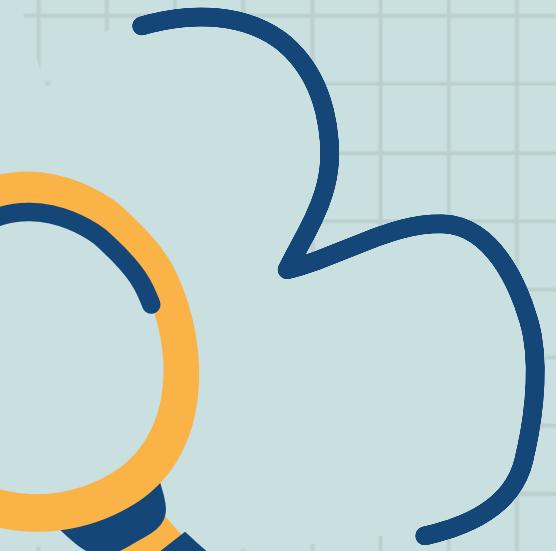
Prueba con 1 réplica

- Tiempo total: 20.081 segundos
- Fallos: 999
- Requests por segundo: 49.80
- Tiempo por solicitud: 1004 ms (promedio)

Conclusión:

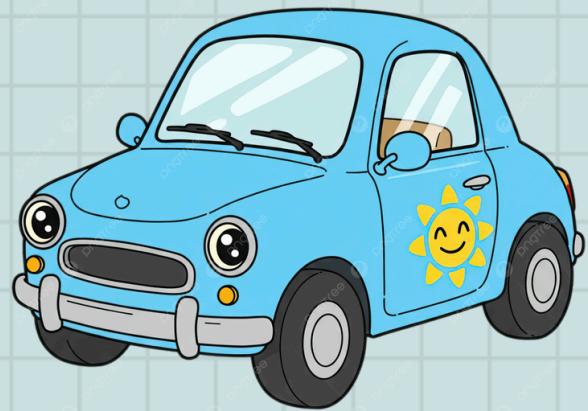
Con una sola réplica el microservicio no soportó la carga: 999 solicitudes fallaron, generando respuestas incompletas y lentitud general.

```
100% 4578 (longest request)
root@servidorUbuntu1:/vagrant# docker service scale proyecto-docker_ms-resenas=1
proyecto-docker_ms-resenas scaled to 1
overall progress: 1 out of 1 tasks
1/1: running  [=====]
verify: Service proyecto-docker_ms-resenas converged
jovincpoggesk@proyecto-docker_ventanas: ~ repeated 1/1 josephs
root@servidorUbuntu1:/vagrant# ab -n 1000 -c 50 http://192.168.100.2:8080/resenas/
This is ApacheBench, Version 2.3 <$Revision: 1879490 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/
Concurrency Level:      50
Time taken for tests:   20.081 seconds
Complete requests:      1000
Failed requests:        999
    (Connect: 0, Receive: 0, Length: 999, Exceptions: 0)
Non-2xx responses:     1
Total transferred:      326786261 bytes
HTML transferred:       326535248 bytes
Requests per second:   49.80 [#/sec] (mean)
Time per request:       1004.027 [ms] (mean)
Time per request:       20.081 [ms] (mean, across all concurrent requests)
Transfer rate:          15892.36 [Kbytes/sec] received
```



Pruebas

Ahora haremos solicitudes a reseñas a 4 replicas:



Con 4 réplicas el sistema alcanza su mejor rendimiento, procesando casi el doble de solicitudes que con 2 réplicas y reduciendo la latencia a la mitad.

Esto demuestra que la aplicación escala de forma eficiente y que el balanceo de carga con HAProxy distribuye el tráfico de manera óptima.

Pero fue levemente mas lento en completar las 1000 solicitudes

```
root@servidorUbuntu1:/vagrant# docker service scale proyecto-docker_ms-resenas=4
proyecto-docker_ms-resenas scaled to 4
overall progress: 4 out of 4 tasks
1/4: running [=====>]
2/4: running [=====>]
3/4: running [=====>]
4/4: running [=====>]
verify: Service proyecto-docker_ms-resenas converged

Concurrency Level:      50
Time taken for tests:   17.059 seconds
Complete requests:      1000
Failed requests:         0
Total transferred:      327113000 bytes
HTML transferred:       326862000 bytes
Requests per second:   58.62 [#/sec] (mean)
Time per request:       852.972 [ms] (mean)
Time per request:       17.059 [ms] (mean, across all concurrent requests)
Transfer rate:          18725.49 [Kbytes/sec] received
```



Análisis de la información



Generar csv con spark

Contexto



¿A qué se debe?

Se extrajo la información de la base de datos de calificaciones y se tomaron los vehículos del dataset para generar un nuevo csv que relacione ambos.

Como calificaciones registra el id del vehículo necesitamos relacionar el id con la marca/modelo al que se refiere para poder analizar.



Video de ejecución

The terminal window shows the user navigating through the Spark configuration directory and executing a command. The browser window shows the Spark Master UI with one application listed.

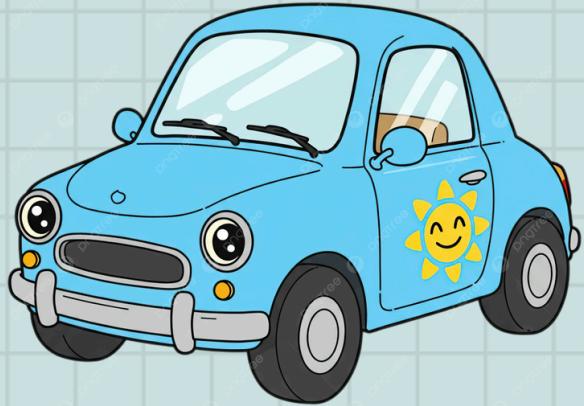
```
vagrant@servidorUbuntu2:~/SparkAutos$ cd ..
vagrant@servidorUbuntu2:~/SparkAutos/spark-4.0.1-bin-hadoop3$ cd ..
vagrant@servidorUbuntu2:~/SparkAutos$ ls
autos_limpios8.csv  calificaciones.csv  relacion_autos.py
URIautos_relacionados.csv  ExecutionCommand.txt  spark-4.0.1-bin-hadoop3
Worvagrant@servidorUbuntu2:~/SparkAutos$ cd ~/SparkAutos/spark-4.0.1-bin-hadoop3/conf
vagrant@servidorUbuntu2:~/SparkAutos/spark-4.0.1-bin-hadoop3/conf$ ls
Corfairscheduler.xml.template  metrics.properties.template  spark-env.sh.template
Metlog4j2-json-layout.properties.template  spark-defaults.conf.template  workers.template
Reslog4j2.properties.template  spark-env.sh
vagrant@servidorUbuntu2:~/SparkAutos/spark-4.0.1-bin-hadoop3/conf$ vim spark-env.sh
Appvagrant@servidorUbuntu2:~/SparkAutos/spark-4.0.1-bin-hadoop3/conf$ cd ..
Drivagrant@servidorUbuntu2:~/SparkAutos/spark-4.0.1-bin-hadoop3$ cd ..
Statvagrant@servidorUbuntu2:~/SparkAutos$ ls
autos_limpios8.csv  calificaciones.csv  relacion_autos.py
autos_relacionados.csv  ExecutionCommand.txt  spark-4.0.1-bin-hadoop3
- Vvagrant@servidorUbuntu2:~/SparkAutos$ vim ExecutionCommand.txt
vagrant@servidorUbuntu2:~/SparkAutos$ cd ~/SparkAutos/spark-4.0.1-bin-hadoop3/conf
Worvagrant@servidorUbuntu2:~/SparkAutos/spark-4.0.1-bin-hadoop3/conf$ ls
fairscheduler.xml.template  metrics.properties.template  spark-env.sh.template
Wolog4j2-json-layout.properties.template  spark-defaults.conf.template  workers.template
log4j2.properties.template  spark-env.sh
vagrant@servidorUbuntu2:~/SparkAutos/spark-4.0.1-bin-hadoop3/conf$ vim spark-env.sh
Rvagrant@servidorUbuntu2:~/SparkAutos/spark-4.0.1-bin-hadoop3/conf$ cd ..
vagrant@servidorUbuntu2:~/SparkAutos$ ls
- Cautos_limpios8.csv  calificaciones.csv  relacion_autos.py
autos_relacionados.csv  ExecutionCommand.txt  spark-4.0.1-bin-hadoop3
Application$ Borraremos el au...
```

State	Cores	Memory
ALIVE	2 (0 Used)	1944.0 MiB (0.0 B Used)

Submitted Time	User	State	Duration



Master + Worker



```
vagrant@servidorUbuntu1:~/labSpark/spark-3.5.1-bin-hadoop3$ ./sbin/start-worker.sh spark://192.168.100.3:7077
org.apache.spark.deploy.worker.Worker running as process 15814. Stop it first.
vagrant@servidorUbuntu1:~/labSpark/spark-3.5.1-bin-hadoop3$ |
```

```
vagrant@servidorUbuntu1:~/labSpark/spark-3.5.1-bin-hadoop3$ jps
23877 Jps
15814 Worker
15657 Master
```

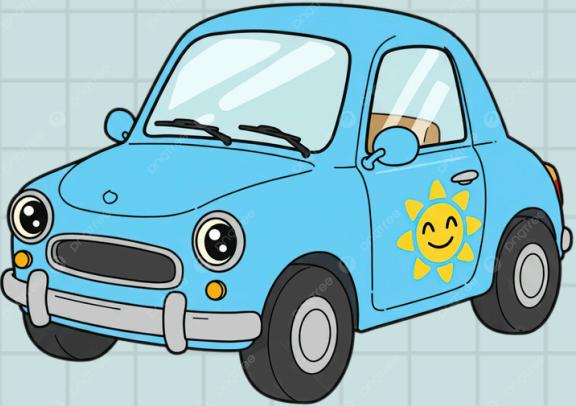
192.168.100.3:7077

Master

<http://192.168.100.3:18080/>

Worker





Análisis en spark

```
vagrant@servidorUbuntu1:~/spark-proyecto$ cd ~/spark-proyecto  
vagrant@servidorUbuntu1:~/spark-proyecto$ spark-submit \  
--master spark://192.168.100.3:7077 \  
analisis_real_spark_def.py
```



Levantar el Dashboard Flask

```
vagrant@servidorUbuntu:~/spark-proyecto/dashboard$ python3 app.py
 * Serving Flask app 'app'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:8081
 * Running on http://10.0.2.15:8081
Press CTRL+C to quit
```

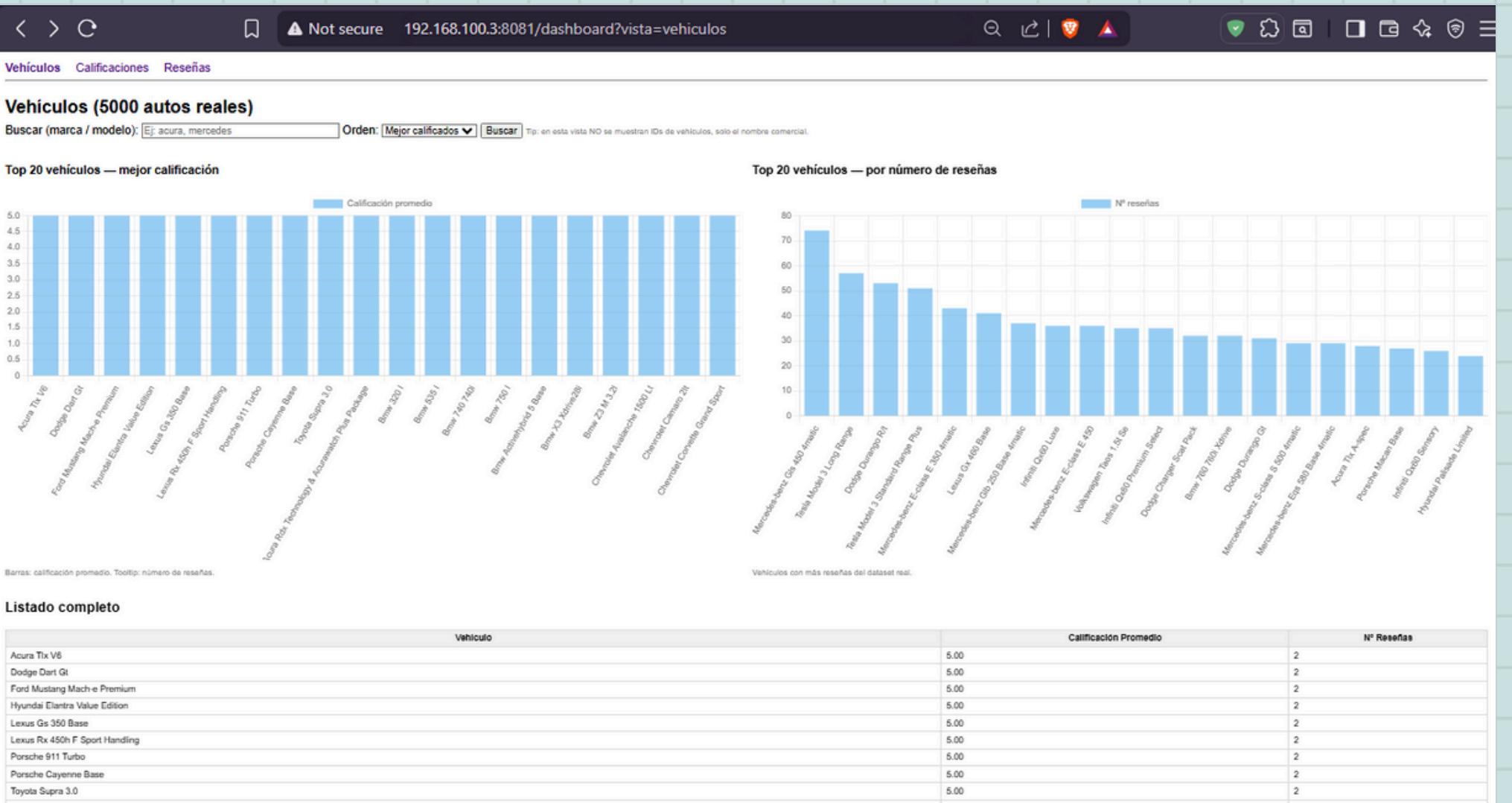
[http://192.168.100.3:8081/dashboard?
vista=vehiculos&q=&orden=mejor](http://192.168.100.3:8081/dashboard?vista=vehiculos&q=&orden=mejor)

**http://192.168.100.3:8081/dashboard?
vista=calificaciones**

**http://192.168.100.3:8081/dashbo
ard?vista=rese%C3%B1as**

Visualizar los Dashboards

***http://192.168.100.3:8081/dashboard?
vista=vehiculos***



*Imagen de
Referencia*



Conclusión

Además de lo técnico, este proyecto nos enseñó algo clave: cómo resolver problemas reales. Cada error, cada contenedor que no levantaba, cada ruta que fallaba, nos hizo entender cómo trabajan los sistemas por dentro.

Al final no solo entregamos una plataforma terminada, sino que también aprendimos a trabajar con tecnologías que se usan en empresas de verdad. Esa experiencia es lo que nos llevamos.



**¡Muchas
gracias!**

