Informe de la solución: aplicación web para la gestión y análisis académico

Valentina Bueno Collazos, Estefanía Hernández Rojas, Valeria Franco Córdoba, Natalia Moreno Montoya

***Resumen*— Este proyecto desarrolla una aplicación basada en microservicios para gestionar el programa académico de Ingeniería de Datos e Inteligencia Artificial en la Universidad Tecnológica. La aplicación utiliza Docker Compose para empaquetar contenedores, Docker Swarm para orquestación, HA Proxy con el algoritmo Least Connection para balanceo de carga y JMeter para pruebas de desempeño. La arquitectura de microservicios incluye gestión de usuarios, asignaturas y cursos, con bases de datos independientes para cada componente. Se realizó una prueba de carga y estrés para evaluar la capacidad de la aplicación frente a 1000 usuarios concurrentes, obteniendo resultados positivos con una tasa de error cercana a 0%. La aplicación cumple con los requisitos de escalabilidad y rendimiento. Para el procesamiento de datos, se utilizó Apache Spark para mostrar información relevante de la aplicación.**

I. Introducción

Las instituciones de educación superior enfrentan retos cada vez más complejos en la gestión eficiente de su información, especialmente debido al creciente volumen de datos generados por estudiantes, profesores y administradores. Esta sobrecarga puede afectar la calidad del servicio educativo y la eficiencia de la administración académica de los programas. Por lo tanto, la gestión adecuada de la información es esencial para tener un rendimiento óptimo.

La Universidad Tecnológica carece actualmente de un sistema centralizado para gestionar el programa académico de Ingeniería de Datos e Inteligencia Artificial. Con esta herramienta, es posible evaluar adecuadamente a los estudiantes y analizar su rendimiento en relación con diversas características propias de cada estudiante.

El objetivo de este proyecto es desarrollar una aplicación basada en microservicios para la gestión eficiente del programa académico. Esta solución permitirá la administración de estudiantes y profesores, la gestión de asignaturas, la creación y el manejo de cursos y notas, y la visualización del rendimiento académico. Utilizando APIs REST para facilitar la interacción entre los componentes del sistema, la aplicación se desplegará en un clúster de procesamiento distribuido, lo que garantiza escalabilidad y un rendimiento óptimo en la gestión del programa académico.

II. Métodos

Para el desarrollo de este proyecto, se utilizó el conjunto de datos “Predict Students’ Dropout and Academic Success” [1], que contiene información sobre estudiantes de educación superior y variables que pueden predecir la deserción académica. Este dataset, obtenido de Kaggle, incluye 4000 registros de estudiantes con datos como su país de origen, necesidades de educación especial, género, estado civil, si tiene un préstamo, beca y si es desplazado.

Se consideraron múltiples herramientas para definir la arquitectura del sistema, asegurando el cumplimiento de los requisitos. Para el empaquetado, se eligió Docker Compose por su capacidad para gestionar múltiples contenedores de manera sencilla, esencial dada la cantidad de contenedores necesarios en el proyecto.

Para la distribución y orquestación, se evaluaron Docker Swarm, Kubernetes y Apache Mesos. Docker Swarm se destaca por su facilidad de configuración y su integración nativa con Docker; Kubernetes, aunque altamente escalable, es más adecuado para entornos complejos; Apache Mesos, por otro lado, es especializado en Big Data y compatible con múltiples framework.

En cuanto al balanceo de carga, se compararon Docker Swarm, HA Proxy y Nginx. Docker Swarm es ideal para entornos Dockerizados, ofreciendo integración nativa y facilidad de configuración, pero con opciones limitadas en comparación con otras soluciones. HA Proxy se destaca por su rendimiento y flexibilidad, permitiendo configuraciones avanzadas para balancear tráfico HTTP y TCP, ideal para entornos de alto tráfico. Nginx, además de balanceo de carga, actúa como proxy inverso y gestiona eficientemente el tráfico web, siendo adecuado para aplicaciones de alto rendimiento, pero con menos personalización que HA Proxy.

En cuanto a algoritmos de balanceo, se pensó en usar Roundrobin, ya que se distribuyen las solicitudes entrantes de manera equitativa, sin embargo, no tenía en cuenta las conexiones activas, algo muy a tener en cuenta sabiendo el contexto de la aplicación web, donde varios estudiantes y profesores podrían estar conectados a la vez. Finalmente, se escogió Least Connection, que sí tiene en cuenta las conexiones activas, y envía las nuevas solicitudes al servidor con menos de estas, de esta manera, resulta más eficiente en el contexto.

Para las pruebas de desempeño, se evaluaron JMeter y Artillery, y se optó finalmente por JMeter.

Las herramientas anteriormente mencionadas se pensaron de la siguiente manera para la implementación:

TABLE I
ALTERNATIVAS DE HERRAMIENTAS

Alternativas	Descripción
Docker Compose + Docker Swarm + HA Proxy (Least Connection) JMeter	Cumple su propósito para proyectos de mediano a grande, tiene las características necesarias para el proyecto.
Docker Compose + Kubernetes + Ngix (Round robin) + JMeter	Kubernetes ofrece mejor manejo para aplicaciones distribuidas, pero es más complejo de gestionar y configurar, además, requiere más recursos de cómputo.
Docker Compose + Apache Mesos + HA Proxy (Round robin) + Artillery	Apache Mesos es muy específico para ambientes de Bigdata, por lo que no iba enfocado al proyecto.

Para la decisión final se tomó la alternativa 1: DockerCompose, Docker Swarm, HA Proxy (Least Connection) y JMeter, ya que se adecuaba más a las necesidades y requerimiento del proyecto.

En cuanto al procesamiento de datos, se evaluaron Apache Hadoop y Apache Spark, seleccionando Spark debido a su rapidez y flexibilidad en el procesamiento mediante el uso de memoria RAM.

Por último, para el análisis de datos se eligió PowerBI por su versatilidad en el análisis estadístico y exploratorio, así como su capacidad de generar visualizaciones desde básicas hasta avanzadas, en comparación con herramientas como Python y GraphX.

A. Componentes

Los microservicios se diseñaron de modo que cada uno gestione un aspecto específico del sistema; en este caso, se definieron cuatro microservicios: usuarios, asignaturas, cursos y web, los cuales interactúan entre sí mediante API REST. Estos microservicios se empaquetaron usando Docker Compose para facilitar la administración y despliegue en múltiples contenedores.

El microservicio de Usuarios permite la gestión de perfiles de estudiantes, profesores y el administrador, lo que incluye el registro, inicio de sesión y edición de la información del usuario. Este microservicio tiene su propia base de datos “usuariosdb”, contenerizada en Docker y la cual cuenta con las siguientes tablas con la siguiente información:

TABLE II
INFORMACIÓN DE LA BASE DE DATOS DE USUARIOS

Tablas de usuariosdb	Campos
Estudiantes	Usuario, contraseña, nombre, correo, país de origen, necesidades especiales de educación, género, estado civil, préstamo, beca, desplazado, total de créditos matriculados.
Profesores	Usuario, contraseña, rol, nombre, correo, último grado de formación

El microservicio de asignaturas tiene la información del pénsum del programa académico, así como los cupos de las asignaturas. Esto permite que los estudiantes consulten la información de asignaturas y que el administrador edite, cree o elimine asignaturas según sea necesario. Este microservicio cuenta con su base de datos “asignaturasdb” contenerizada en Docker y la cual cuenta con la siguiente tabla e información:

TABLE III
INFORMACIÓN DE LA BASE DE DATOS DE ASIGNATURAS

Tablas de asignaturasdb	Campos
Asignaturas	Id, nombre de la asignatura, créditos, cupos, semestre

El microservicio de cursos administra los cursos de los estudiantes y los profesores, así como los grupos de cada asignatura del período. La información se almacena en la base de datos “cursosdb” contenerizada en Docker y la cual cuenta con la siguiente información:

TABLE IV
INFORMACIÓN DE LA BASE DE DATOS DE CURSOS

Tablas de cursosdb	Campos
cursos	Id, nombre del curso, grupo, profesor, correo del profesor, nombre del estudiante, correo del estudiante, nota, periodo

En la tabla IV podemos ver que cierta información de la base de datos de curso es traída de otros microservicios por medio de la API como el nombre del curso, profesor, correo del profesor, nombre del estudiante, correo del estudiante.

El balanceador de carga, en este caso HA Proxy, distribuye las solicitudes de acceso a la página web, y las conexiones entre los microservicios, para garantizar una carga equilibrada y proveer una mejor tolerancia a fallos.

En cuanto a la orquestación de los contenedores Docker, el uso de Docker Compose y Docker Swarm facilitó el trabajo, ya que los contenedores establecían conexión por medio de la red que Swarm crea.

El clúster de procesamiento de datos con Apache Spark utiliza los utiliza los datos de los archivos CSV de estudiantes y cursos, que están relacionados a través de la columna "nombre del estudiante" y sus respectivas calificaciones. Estos archivos se cargan en el clúster para realizar un análisis de cómo ciertas características de los estudiantes puede afectar sus calificaciones. Este procesamiento permite generar resúmenes que incluyen, entre otros, promedios de calificaciones, por país de origen, necesidades especiales, si es becado, si tienen préstamos, entre otros. Además, los datos procesados se transforman en nuevos archivos CSV que luego se exportan y almacenan en BigQuery.

Los archivos resultantes en BigQuery son estructurados para permitir una fácil consulta y visualización de las métricas en Power BI.

Este proceso se puede ver representado en el siguiente pipeline:

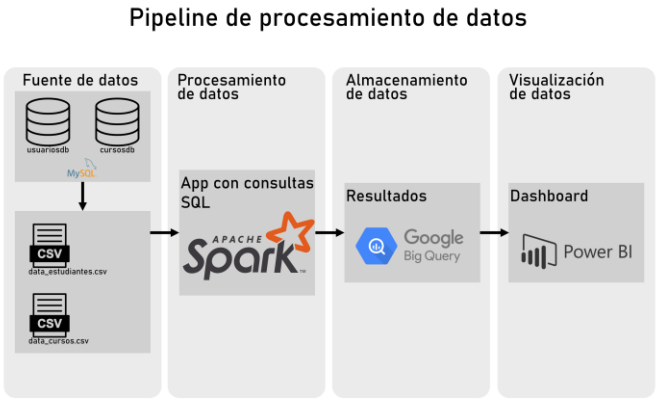

Fig. 1. Pipeline del procesamiento de datos

Todos los componentes anteriormente mencionados, se puede ver representados en la siguiente arquitectura:

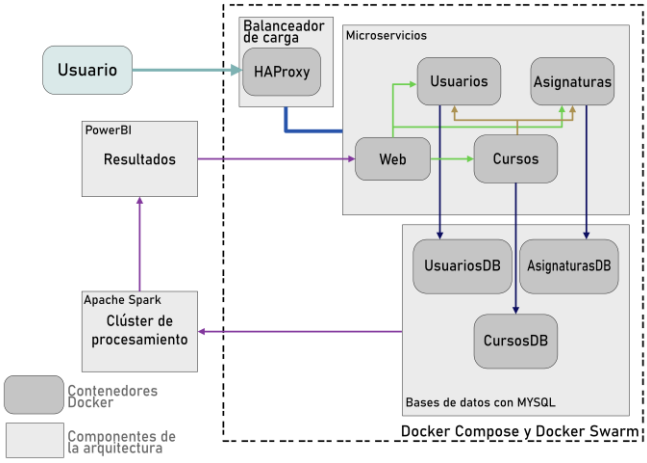

Fig. 2. Arquitectura de los componentes y sus relaciones.

B. Flujo de trabajo

El flujo de trabajo en el sistema comienza cuando un usuario accede a la web por medio de HA Proxy, el cual es el encargado de balancear los microservicios; el usuario puede crear una cuenta o iniciar sesión a través de la interfaz web. Cada tipo de usuario, ya sea estudiante, profesor o administrador, puede iniciar sesión y gestionar sus datos a través del microservicio de Usuarios. Este microservicio administra la creación, validación y edición de la información del usuario. Los administradores, además, tienen permisos para editar y eliminar usuarios según sea necesario.

El microservicio de Cursos gestiona la administración de los cursos, tanto para profesores como para estudiantes. Cuando un estudiante desea matricularse en un curso, Cursos primero consulta al microservicio de Usuarios para verificar que el estudiante tenga créditos suficientes para el período activo y revisa si la asignatura ya fue cursada. Si la asignatura fue cursada previamente y la nota final fue de 2.9 o inferior, el estudiante podrá volver a matricularse; de lo contrario, no se le permitirá hacerlo. En caso de que el curso sea nuevo para la asignatura en ese período, se asignará automáticamente un profesor. Cursos también se comunica con el microservicio de Asignaturas para verificar la disponibilidad de cupos. Si no hay cupos, la matrícula no será permitida; si los hay, se actualizarán los cupos disponibles en Asignaturas. Al quedar matriculado, el estudiante verá en su perfil los detalles del curso, el profesor asignado y la nota una vez que esta se le haya asignado.

Para los profesores, al acceder a su apartado de cursos asignados, Cursos muestra los cursos en los que están involucrados en el período activo, permitiéndoles asignar calificaciones a cada estudiante.

El microservicio de Asignaturas proporciona a los estudiantes una visión completa del pénsum académico, mostrando los créditos, cupos disponibles y otra información relevante de cada asignatura. Los administradores, por su parte, pueden crear, eliminar o editar la información de cupos en las asignaturas, manteniendo los datos actualizados.

Finalmente, el apartado de métricas muestra los resultados del clúster de procesamiento de datos por medio de un dashboard de PowerBI, lo cual le permite ver al administrador métricas relevantes sobre el promedio de notas de acuerdo a características y variables propias de cada persona, como el país de origen, si necesita de educación especial, préstamo estudiantil o si tiene una beca.

Con todo lo anterior definido, se continuó con el despliegue de la aplicación, el cual se ve reflejado en el siguiente diagrama:

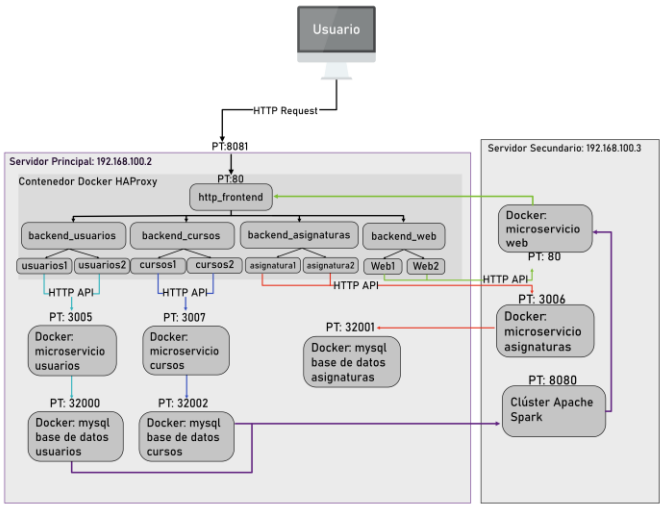

Fig. 3. Diagrama de despliegue

III. Resultados

Con la aplicación funcionando, se realizaron pruebas, para determinar si se cumplieron los requisitos del proyecto y que pudiera escalarse a medida que la carga aumenta. Se realizó como primera prueba una prueba de carga, la cual determina si el sistema maneja eficientemente el volumen de trabajo esperado. Para esta prueba, se tomó en cuenta la cantidad de usuarios que tiene la aplicación actualmente, que son alrededor de 45.000 personas, por lo que es normal que al menos 1.000 estén conectadas al mismo tiempo, es por esto que la configuración para esta prueba fue la siguiente:

- Grupo de hilos: 1000 en 60 segundo en bucles de 10.

- Petición Http de inicio de sesión de los estudiantes.

Los resultados fueron los siguientes:

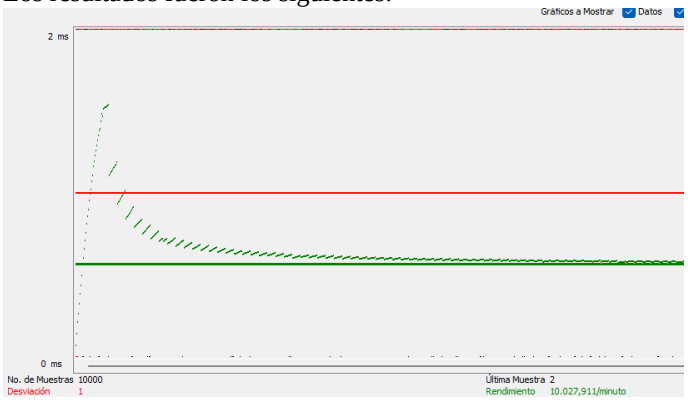

Fig. 4. Gráfica de resultados de prueba de carga

En la Fig.4 se puede ver cómo el rendimiento es bajo y consistente, por lo tanto, maneja bien la carga.

Etiqueta	# Muestras	Media	Máx	Desv. Estándar	% Error	Rendimiento	Kb/sec	Sent KB/sec	Media de Bytes
Petición HTTP	10000	1	47	1,50	0,00%	167,1/sec	707,54	29,05	4335,0
Total	10000	1	47	1,50	0,00%	167,1/sec	707,54	29,05	4335,0

Fig. 5. Reporte de rendimiento de prueba de carga

En la Fig.5 podemos ver la tabla de la prueba, la cual nos muestra que hubo un porcentaje de error del 0%, lo que significa que el servidor no tiene problemas para manejar la carga, y los kilobytes por segundo son altos, por lo que se está enviando buena cantidad de información por parte del servidor.

Se prosiguió con una prueba de estrés en la cual se fue subiendo gradualmente el número de hilo de la prueba.

La prueba se inició con 2000 hilos en 10 segundos y 1 bucle, y los resultados arrojados fueron los siguientes:

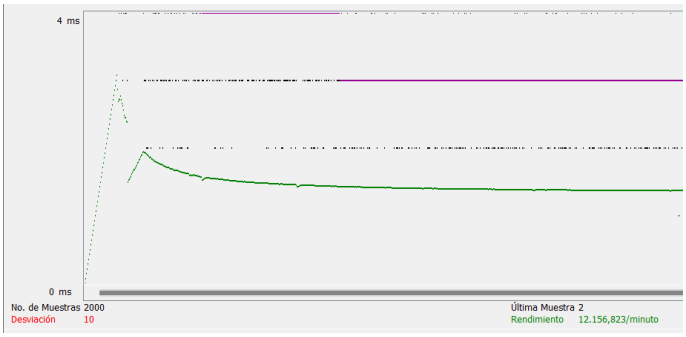

Fig. 6. Gráfica de resultados de prueba de estrés

Etiqueta	# Muestras	Media	Mín	Máx	Desv. Estándar	% Error	Rendimiento	Kb/sec	Sent KB/sec
Petición HTTP	2000	4	1	93	10,82	0,00%	202,6/sec	1186,40	24,34
Total	2000	4	1	93	10,82	0,00%	202,6/sec	1186,40	24,34

Fig. 7. Reporte de rendimiento de prueba de estrés 2

En la Fig.6 y 7 se puede observar el esfuerzo adicional del servidor, sin embargo, continúa manteniendo su tasa de error en 0.

Luego, se realizó la prueba con 2000 hilos en 10 segundos y en 2 bucles, la prueba arrojó el siguiente resultado:

Fig. 8. Gráfica de resultados de prueba de estrés 2

Etiqueta	# Muestras	Media	Mín	Máx	Desv. Estándar	% Error	Rendimiento	Kb/sec	Sent KB/sec
Petición HTTP	4000	3	0	85	7,56	0,03%	404,7/sec	2369,53	48,60
Total	4000	3	0	85	7,56	0,03%	404,7/sec	2369,53	48,60

Fig. 9. Reporte de rendimiento de prueba de estrés 2

En esta prueba, ya se puede ver el esfuerzo que requieren esta cantidad de solicitudes, ya que el índice de error subió ligeramente a 0.03%.

Con las pruebas anteriores, es posible afirmar que la aplicación cumple con la escalabilidad que se buscaba, ya que aún con gran cantidad de usuarios a la misma vez, hay una casi nula tasa de errores.

Además, se puede observar que el algoritmo de balanceo de carga está funcionando correctamente y permite un buen rendimiento en la página. Esto se puede ver reflejado en el dashboard presentado por HA Proxy:

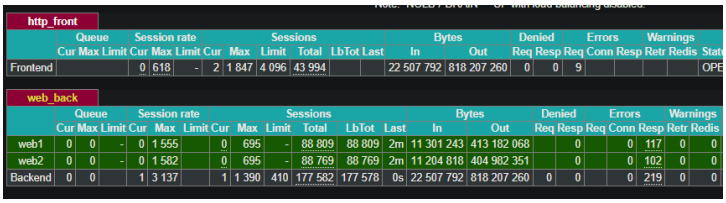

Fig. 10. Dashboard del rendimiento de HA Proxy

IV. Conclusiones

El proyecto aborda la problemática de manera eficiente para la gestión del programa de Ingeniería de Datos e Inteligencia Artificial de la Universidad Tecnológica. Se logró administrar a los estudiantes, profesores, las asignaturas y los cursos ofertados del programa por medio de una arquitectura basada en microservicios y API REST distribuida en contenedores y orquestada en dos servidores, lo que permitió crear una aplicación escalable y con un buen rendimiento con los resultados obtenidos en la prueba de JMeter, lo que muestra la eficacia del balanceo de carga de HA Proxy y la gestión de varios servidores.

El clúster de procesamiento y el análisis de datos también demuestran la efectividad del proyecto al proveer informes detallados y actualizados sobre el rendimiento académico de los estudiantes. Esta arquitectura analítica permite a los usuarios acceder a información clave a través del dashboard de Power BI.

Además, el uso de BigQuery y Spark facilita la gestión y

almacenamiento de grandes volúmenes de datos, lo que asegura la escalabilidad del sistema y respalda futuras expansiones del programa académico en la Universidad Tecnológica.

Este proyecto se puede tomar como referencia para mejorar significativamente la administración educativa, ya que brinda una plataforma robusta, escalable y fácil de usar para todos los involucrados en el proceso académico.

REFERENCIAS

[1] *Predict students' dropout and academic success*. (2023, 3 enero). Kaggle. https://www.kaggle.com/datasets/thedevastator/higher-education-predictors-of-student-retention/data

[2] *«Home»*. (2024, 31 octubre). Docker Documentation. https://docs.docker.com/

[3] *Apache mesos*. (s. f.). Apache Mesos. https://mesos.apache.org/documentation/latest/

[4] *¿Qué es Kubernetes?* (2022, 17 julio). Kubernetes. https://kubernetes.io/es/docs/concepts/overview/what-is-kubernetes/

[5] Artillery. (s. f.). *Artillery · The complete load testing platform*. Artillery. https://www.artillery.io/

[6] *Apache JMeter - Apache JMeterTM*. (s. f.). https://jmeter.apache.org/

[7] Apache Spark™ - Unified Engine for large-scale data analytics. (s. f.). https://spark.apache.org/

[8] *Power BI - Data Visualization | Microsoft Power Platform*. (s. f.). https://www.microsoft.com/en-us/power-platform/products/power-bi

[9] *The world's largest container registry | Docker*. (2024, 21 octubre). Docker. https://www.docker.com/products/docker-hub/

[10] *«Swarm mode»*. (2024, 10 septiembre). Docker Documentation. https://docs.docker.com/engine/swarm/

[11] Fakhroutdinov, K. (s. f.). UML deployment diagrams overview, common types of deployment diagrams - manifestation diagram, specification and instance level deployment diagram. https://www.uml-diagrams.org/deployment-diagrams-overview.html

[12] Sentrio. (2024, 17 abril). Mejora los resultados de pruebas de estrés con JMeter. Sentrio. https://sentrio.io/blog/pruebas-de-estres-jmeter/