Evaluación Comparativa del Sistema de Base de Datos NoSQL y el Gestor de Ficheros Tradicional en Dispositivos Móviles Android

Jorge Andrés García Zapata

Agosto 2025

Evaluación Comparativa del Sistema de Base de Datos NoSQL y el Gestor de Ficheros Tradicional en Dispositivos Móviles Android

Jorge Andrés García Zapata

Tesis de investigación presentada como requisito para optar al título de: Magíster en Ingeniería de Software

Directores

Doctor Jesús Andrés Hincapié Londoño

Doctor Mauricio González Palacio

Universidad de Medellín

Facultad de Ingeniería

Medellín, Colombia

2025

RECONOCIMIENTOS

En primer lugar, expreso mi más profundo agradecimiento a Dios por haberme dado la vida, la oportunidad de estar en la posición de educación y por guiarme por un buen camino; a mis padres, por inculcarme valores y darme su apoyo para alcanzar mis metas; y en especial a mi madre que desde el cielo me acompaña y me ilumina lo necesario para salir adelante.

A la Universidad de Medellín y sus directivos por brindarme la oportunidad de realizar mi maestría, y a los doctores Jesús Andrés Hincapié Londoño y Mauricio González Palacio, profesores de tan prestigiosa Universidad, por su valioso aporte académico como directores de esta tesis de maestría y, en especial por su dedicación desinteresada.

A mis docentes de la maestría por su dedicación, paciencia experiencia y conocimientos trasmitidos, en mi proceso de formación como magíster.

RESUMEN

En este trabajo se propone la evaluación comparativa entre sistemas de bases de datos NoSQL y el gestor tradicional de ficheros en dispositivos móviles Android. Para ello, se desarrolló una aplicación específica en el entorno de desarrollo Android Studio, orientada a medir las características de velocidad de acceso y replicación en el almacenamiento de datos. Las herramientas utilizadas se enfocaron en el manejo de bases de datos en entornos móviles, considerando variables asociadas a los diferentes métodos de almacenamiento. La aplicación se diseñó con el planteamiento de una función objetivo, incorporando variables relacionadas con el tiempo de ingreso de información, búsqueda y replicación de los datos. La información recolectada mediante la aplicación fue analizada utilizando métodos estadísticos, tales como la correlación de Spearman, regresión lineal múltiple, estadística descriptiva y comportamiento gráfico de las variables. Entre los principales hallazgos, se observó que: La base de datos *Realm* fue la mejor en todos los procesos de almacenamiento, las demás se destacan en términos medios con variaciones posicionales entre ellas, mientras que el sistema de ficheros tradicional es el peor de los sistemas. Las versiones más recientes de Android no mejoran significativamente los tiempos de acceso ni los tiempos de replicación. El tamaño de los datos a almacenar influye considerablemente en el incremento de los tiempos de acceso, búsqueda y replicación. A medida que aumenta el número de núcleos del dispositivo, los tiempos de acceso y replicación tienden a disminuir. Finalmente, se determinó que el uso del método FILES incrementa sistemáticamente el tiempo requerido tanto para la inserción y búsqueda como para la replicación de datos, siendo la opción menos eficiente frente a otros métodos evaluados.

Palabras claves: Bases de datos, almacenamiento, dispositivos móviles, Android, *web sockets*.

ABSTRACT

This study proposes a comparative evaluation between NoSQL database systems and the traditional file manager on Android mobile devices. To achieve this, a specific application was developed in the Android Studio development environment, aimed at measuring data storage access speed and replication characteristics. The tools used focused on database management in mobile environments, considering variables associated with different storage methods. The application was designed with an objective function approach, incorporating variables related to data entry time, search, and data replication. The data collected through the application was analyzed using statistical methods, such as Spearman's correlation, multiple linear regression, descriptive statistics, and graphical behavior of the variables. Among the main findings, it was observed that: The Realm database performed the best in all storage processes, while the others showed intermediate performance with positional variations among them, whereas the traditional file system was the worst among the systems. The most recent versions of Android do not significantly improve access or replication times. The size of the data to be stored considerably influences the increase in access, search, and replication times. As the number of device cores increases, access and replication times tend to decrease. Finally, it was determined that using the FILES method systematically increases the time required for both data insertion and search, as well as replication, making it the least efficient option compared to the other methods evaluated.

Keywords: Databases, storage, mobile devices, Android, web sockets.

CONTENIDO

Lista de tablas

Lista de figuras, gráficos e ilustraciones

Abreviaturas

NoSQL: No Solo SQL (*Not Only SQL* por sus siglas en inglés)

SQL: Lenguaje de Consulta Estructurada

OIDs: Sistema de Bases de Datos Orientados

CRUD: Operaciones de Escritura, Lectura, Actualización y Borrado

ACID: Propiedades de las bases de datos: Atomicidad, coherencia, aislamiento y durabilidad.

SGBD SQL: Sistema Gestor de Base de Datos SQL

MySQL: Sistema de Gestión de Bases de Datos Relacionales

NTFS: *New Technology File System* por sus siglas en inglés

FAT: *File Allocation Table* por sus siglas en inglés

EXT: *Extended File system* por sus siglas en inglés

RAM: *Random Access Memory* por sus siglas en inglés

GB: *Gigabyte Memory* por sus siglas en inglés

ROM: *Read Only Memory* por sus siglas en inglés

LCD: Pantalla de Cristal Liquido

HDFS: *Hadoop Distributed File System* por sus siglas en inglés

CAPITULO 1
MOTIVACIÓN

En la actualidad, el crecimiento exponencial de los dispositivos móviles, el acceso a Internet de las distintas plataformas y la creciente demanda de aplicaciones que requieren la gestión y almacenamiento de grandes volúmenes de información han generado diversas formas de estructurar los datos, lo que nos permite evaluar las formas en que se realiza el almacenamiento de datos en los diversos entornos

Los usuarios no solo consumen información, sino que también generan datos constantemente a través de aplicaciones de mensajería, redes sociales, plataformas multimedia y dispositivos del Internet de las Cosas (IoT). Este ecosistema ha impuesto nuevos retos sobre la capacidad de los sistemas de almacenamiento para garantizar eficiencia, escalabilidad y rendimiento, particularmente en dispositivos con recursos limitados como los teléfonos inteligentes y tabletas.

Diversos estudios han abordado la problemática del almacenamiento en dispositivos móviles, aunque de forma fragmentada y con un enfoque limitado en un único sistema de almacenamiento, generalmente orientado a bases de datos relacionales embebidas o al uso directo del sistema de archivos tradicional. Entre las soluciones más implementadas se destacan los motores de bases de datos relacionales como SQLite, los cuales, si bien han demostrado ser efectivos para ciertas cargas de trabajo, presentan limitaciones al enfrentar demandas contemporáneas como el manejo de archivos binarios pesados (videos, imágenes, audios) y la necesidad de replicación de datos sin conectividad persistente a Internet. En contraste, las bases de datos NoSQL han emergido como una alternativa prometedora gracias a su capacidad para gestionar datos no estructurados y su flexibilidad en modelos de almacenamiento orientados a documentos, claves-valor o grafos. Sin embargo, su aplicabilidad directa en dispositivos móviles, sin un esquema cliente-servidor, sigue siendo un área poco explorada.

Uno de los desafíos críticos en este contexto es determinar cuál estructura de almacenamiento - ya sea un sistema de ficheros tradicional como EXT4 o una base de datos NoSQL - ofrece el mejor rendimiento bajo escenarios de uso real en Android. Esto es especialmente relevante en aplicaciones de alto impacto, como aquellas orientadas a comunicación y multimedia, donde el rendimiento del sistema de almacenamiento afecta directamente la experiencia de usuario. Factores como la fragmentación de archivos, la gestión de permisos introducida en versiones recientes de Android, y la capacidad de replicar datos localmente sin dependencia de servidores externos son variables que pueden influir en la viabilidad de estas soluciones.

Actualmente, existe una brecha significativa en la literatura respecto a estudios comparativos entre los sistemas de archivos tradicionales (como EXT4) y bases de datos NoSQL implementadas directamente en dispositivos Android. La mayoría de los análisis disponibles se centran en entornos de servidor o en arquitecturas distribuidas, dejando de lado el estudio del comportamiento de estas tecnologías en escenarios móviles con recursos de hardware heterogéneos. Esta carencia dificulta la toma de decisiones técnicas sobre la adopción de tecnologías de almacenamiento más eficientes y seguras en aplicaciones móviles modernas.

En este sentido, la presente investigación propone el desarrollo de una aplicación experimental orientada a evaluar y comparar la aplicabilidad de bases de datos NoSQL frente a los sistemas de archivos tradicionales en dispositivos móviles Android. Este estudio se centrará en criterios clave como:

- Tiempos de replicación (inserción, búsqueda y replicación) para valorar la eficiencia en la sincronización y recuperación de datos.
- Consumo de espacio, considerando las particularidades de almacenamiento y las restricciones.
- Facilidad de acceso a la información, analizando las implicaciones de los nuevos modelos de permisos en Android.
- Rendimiento bajo condiciones reales, mediante pruebas en emuladores, y simulaciones de comunicación entre dispositivos usando *web sockets*.

El análisis contempla además las implicaciones prácticas en diversos escenarios de uso, como respaldo de información local, sincronización de datos sin Internet, aplicaciones de mensajería peer-to-peer, donde la replicación de datos es crítica. La evaluación en diferentes versiones del sistema operativo Android permitirá entender cómo los cambios recientes impactan en la viabilidad y rendimiento de estas soluciones.

La pregunta de investigación central es: ¿Cuál es el modelo más eficiente en términos de tiempo de procesamiento para el almacenamiento de ficheros tradicionales en un sistema de base de datos NoSQL en dispositivos móviles Android?

Este trabajo aspira a aportar conocimiento original en el área mediante:

1. La identificación de tecnologías de almacenamiento que ofrezcan mayor eficiencia y flexibilidad en dispositivos móviles.
2. El planteamiento de un modelo conceptual que implemente las ventajas de un sistema de ficheros tradicional mejorado empleando de las bases de datos NoSQL y sus características .

3. Un análisis comparativo exhaustivo que mida, en condiciones de uso real, el rendimiento y la escalabilidad de ambas alternativas en dispositivos móviles Android, considerando la diversidad de hardware y restricciones propias del ecosistema móvil.

Así, esta investigación no solo busca cerrar una brecha teórica en la literatura actual, sino también ofrecer recomendaciones prácticas para desarrolladores y arquitectos de sistemas sobre las mejores estrategias de almacenamiento en dispositivos móviles modernos. Sus resultados podrían influir en el diseño de futuras aplicaciones móviles que requieran gestionar grandes volúmenes de datos de forma eficiente y segura.

CAPITULO 2
OBJETIVOS

2.1 Objetivo General

Evaluar las características velocidad de acceso y replicación de los sistemas de base de datos NoSQL de ficheros y gestor de ficheros tradicional en dispositivos móviles Android mediante un análisis comparativo basado en un caso de estudio.

2.2 Objetivos específicos

1. Identificar las tecnologías relacionadas con los sistemas de almacenamiento de datos en los dispositivos móviles.
2. Identificar las variables que pueden impactar en las en la velocidad de acceso y replicación.
3. Desarrollar una aplicación que permita comparar el tiempo de procesamiento de un sistema de ficheros tradicional contra el de base de datos NoSQL bajo diferentes configuraciones de *hardware y software*.
4. Evaluar el sistema de archivos de ficheros tradicional contra el de bases de datos no relacionales en dispositivos móviles Android, en su velocidad de respuesta y velocidad de replicación.

CAPITULO 3
INTRODUCCIÓN

Las bases de datos surgen como respuesta a la necesidad de almacenar información de manera permanente, duradera y accesible. Entre los requisitos de las bases de datos se encuentran las técnicas de preservación de datos en el tiempo y la accesibilidad en la consulta. Las bases de datos se diseñan de manera que se puedan realizar lecturas, actualizaciones, escrituras y eliminaciones de una forma robusta, usable y eficiente. Para su cumplimiento se utilizan sistemas de gestión de bases de datos que automatizan el proceso de almacenamiento ordenado y la recuperación de los datos (Román Pérez, 2020).

Las bases de datos en los dispositivos móviles se volvieron más exigentes por el auge de éstos, el uso de internet y la aparición de redes sociales, produciendo un crecimiento exponencial en el volumen de datos de diversos orígenes y estructuras (estructurados, semiestructurados o no estructurados), donde, actualmente son satisfechas con las tecnologías SQL y NoSQL (Tesone, 2021).

El avance de las tecnologías en bases de datos presenta algunas oportunidades de mejora en el contexto de los dispositivos móviles tales como: 1) la información generada en los dispositivos móviles comúnmente emplea un sistema de gestión de ficheros corrientes y no cuenta con un modelo de replicación de información a una velocidad adecuada y protegiendo la integridad de la información, lo cual deriva en problemas de seguridad en el almacenamiento de información (Tesone, 2021); 2) la información presenta problemas de fragmentación por el aumento en el tamaño de almacenamiento de los ficheros; 3) se puede presentar riesgo de pérdida de datos por la falta de centralización de la información, problema agravado por la ausencia de indexación adecuada que impide búsquedas rápidas y eficientes; y 4) la falta de exploración exhaustiva del modelo relacional, especialmente en lo que respecta a sus limitaciones en la definición de modelos de datos (Tesone, 2021).

La evolución de los sistemas de almacenamiento de datos ha sido influenciada principalmente por el tamaño de la información. Desde mediados de los años 60 hasta la actualidad, se ha observado un progreso gradual en la optimización del manejo de los datos, en paralelo con los adelantos tecnológicos (Gómez, 2022).

Los sistemas de almacenamiento de datos han permitido el desarrollo del Lenguaje de Consulta Estructurada (SQL, por sus siglas en inglés) como el lenguaje estándar para definir, manipular y consultar información. No obstante, con la necesidad de estructuras de datos complejas y operaciones

específicas, se encontraron algunas limitaciones en el modelado relacional de datos y los contratiempos ocasionados al mapear los objetos en las tablas. El principal inconveniente se puede presentar al realizar escrituras o modificaciones en programas, e incluso al realizar consultas de información (Gómez, 2022).

Las bases de datos SQL demandan aplicaciones para las bases de datos con escalabilidad horizontal, alta disponibilidad, tolerancia a fallos, confiabilidad en las transacciones y mantenibilidad del esquema de la base de datos. No obstante, dichas bases de datos podrían no satisfacer estos requisitos de manera eficiente, debido a la rigidez del modelo relacional y las dificultades para escalar (Gómez, 2022). Esto dio pie al desarrollo de sistemas de bases de datos orientadas a objetos (OIDs, por sus siglas en inglés), pero no lograron popularizarse debido a la gran cantidad de inversión realizada en el modelo relacional. No obstante, se plantearon sistemas que extendían del modelo relacional incorporando claves orientadas a objetos, que almacenaban datos como relaciones planas en lugar de objetos reales, lo que facilitaba la creación automática del mapeo y las uniones (Gómez, 2022).

Para darle solución a las dificultades mencionadas, surgió el almacenamiento de datos y la tendencia de las bases de datos "No solo SQL" (NoSQL), que se enfocan en el cumplimiento con los requisitos de escalabilidad y alta disponibilidad, sacrificando características no esenciales como soporte transaccional desde la perspectiva de la aplicación (Gómez, 2022).

Las bases de datos NoSQL surgen como una alternativa ante la gran problemática de volúmenes de información ofreciendo alto rendimiento, flexibilidad de esquema, disponibilidad, replicación de datos y escalabilidad y añadiendo características como variedad y velocidad del *Big Data*. Existen varios tipos de bases de datos NoSQL: clave-valor, orientadas a columnas, documentos y gráficos. La más popular ha sido la base de datos documental por su flexibilidad en la gestión de los esquemas (Carvalho et al., 2023).

A pesar de que las bases de datos relacionales son ampliamente utilizadas, existe una escasez de estudios profundos sobre dichos esquemas. Esta carencia se extiende al ámbito del *benchmarking* diferencial y la evaluación de características como la replicación y el rendimiento en situaciones de alto volumen. Asimismo, hay una notable ausencia de evaluación en cuanto al uso de motores de bases de datos NoSQL, los cuales podrían ofrecer un mejor rendimiento y adaptación a casos de uso específicos en dispositivos móviles (Abramova et al., 2014).

Los estudios previos realizados no evalúan el rendimiento de las tecnologías actuales en cuanto a las operaciones de escritura, lectura, actualización y borrado (CRUD por sus siglas en inglés) y

replicación en diferentes arquitecturas y sistemas operativos, por lo que se desconoce cuál tecnología ofrece mejor desempeño ante los requisitos puntuales de una aplicación bajo una estructura de hardware y software específicas (Venegas Bravo et al., 2022).

CAPITULO 4
MARCO TEÓRICO Y ESTADO DEL ARTE

4.1 Marco teórico

4.1.1 Bases de datos

Las bases de datos son una colección de información y datos almacenados de manera ordenada, perteneciente a un mismo contexto, con una estructura específica dependiendo del motor utilizado, con el fin de que su contenido pueda ser tratado y analizado posteriormente de una forma rápida y sencilla (Osorio Trejos, 2019).

El término “base de datos” es un concepto que se refiere a un conjunto de información almacenada de manera ordenada en un contexto determinado, presentando una estructura específica, muy utilizado hoy en día tanto en las empresas, que requieren de su utilización para guardar información, conservándola de tal manera que no se deteriore con el tiempo, como de las personas con el fin de asegurar los datos para su recuperación posterior (Román Pérez, 2020).

Por lo tanto, las bases de datos están diseñadas de tal modo que el usuario obtenga la información de forma rápida y segura con facilidad. La información se encuentra almacenada siguiendo algunos parámetros formados por tablas con afinidad temática. Esto se obtiene con sistemas de gestión de bases de datos en forma automatizada, estructurada y con facilidad de recuperación (Arias M., 2017).

Para crear una base de datos existen varios modelos, cada uno con sus características tales como seguridad, estructuración de datos, facilidad de aprendizaje, tiempos de respuesta, compatibilidad, entre otros. Las bases de datos se pueden implementar en función de las necesidades, con diseño e implementación de algoritmos y mecanismos lógicos cumpliendo con las especificaciones dadas (Román Pérez, 2020).

En la actualidad existen diferentes tipos de bases de datos. Entre ellas se encuentran las bases de datos jerárquicas, en red, las relacionales (SQL) y las no relacionales (NoSQL). Por los años 70s, en las bases de datos relacionales los usuarios se permitían interactuar con una colección de relaciones que varían en el tiempo y lo único que necesitaban era conocer acerca de la relación entre éstas, su nombre y la relación con sus dominios. Por la facilidad de su manejo estas bases de datos perduraron por mucho tiempo, hasta que, en el año 2005, Google presentó el problema de las bases de datos relacionales para el manejo de grandes volúmenes de datos. En adición, dicho paradigma presentaba

grandes latencias en la velocidad de recuperación de la información. Estos problemas se fueron presentando también en las empresas con el "*Big Data*". Desde 1998, comenzó a utilizarse el término bases de datos no relacionales (NoSQL) para referirse a sistemas de almacenamiento de datos que no emplean el lenguaje (SQL) como interfaz principal. Aunque los sistemas NoSQL se alejan del modelo relacional tradicional, algunos de ellos adoptan ciertos principios de los sistemas de gestión de bases de datos (DBMS) relacionales en cuanto a la organización y manejo de la información. Estas bases de datos surgieron como soluciones viables para afrontar los desafíos de escalabilidad, disponibilidad y rendimiento que enfrentaban compañías como Google, y han adquirido un papel fundamental en el procesamiento de grandes volúmenes de datos, característicos de entornos de *Big Data* (Esquivel & Sevilla, 2021).

De tal manera, en la actualidad las bases de datos están dadas en dos grupos: relacionales y no relacionales (SQL y NoSQL). En la literatura se han comparado las ventajas y desventajas en cuanto al rendimiento de dichas bases de datos; sin embargo, existe una notable deficiencia en el análisis relacionado con la seguridad. Esto hace necesario considerar sus filosofías y principios fundamentales, como ACID para las bases de datos relacionales y BASE para las bases de datos no relacionales (Esquivel & Sevilla, 2021).

4.1.2 Sistemas de gestores de bases de datos

Son modelos de sistemas que determinan las relaciones o vínculos que se presentan entre datos a considerar. Cada relación o vínculo presenta un conjunto de propiedades para cada dato y está dado por una serie de filas o registros (tuplas), así como por los datos que dependen de sus atributos (columnas). Para estos casos los principales sistemas de bases de datos relacionales (SGBD SQL) son: MySQL, MariaDB, SQLite, PostgreSQL, Microsoft SQL Server y Oracle (Peralta de Aparicio & Zurita Barrios, 2020).

En la gestión del manejo de los datos, los motores de bases de datos son los forjadores para el almacenamiento de los datos, su manipulación y luego su recuperación de una manera eficiente de la información. Estos motores son necesarios para el buen funcionamiento de las bases de datos, debido a que estos actúan como la base que impulsa la administración, organización y accesibilidad a la información en todas las aplicaciones y sistemas (Osorio Pérez, 2016).

El papel fundamental de los motores de bases de datos es gestionar la interacción entre las aplicaciones y los datos almacenados, mostrando una sólida infraestructura para el manejo de la información en el entorno. Los motores de bases de datos son la fuerza motriz que permite gestionar

grandes volúmenes de datos de manera eficiente y escalable, para asegurar la integridad y disponibilidad de la información en el momento oportuno para una toma de decisiones (Osorio Trejos, 2019).

Los motores de bases de datos más conocidos son: MySQL (Almacenamiento relacional), *MongoDB* (Almacenamiento documental), *Apache Cassandra* (almacenamiento de familia de Columnas), Neo4j (almacenamiento orientado a grafos) y Redis (Almacenamiento clave-valor) (Osorio Pérez, 2016).

4.1.3 Sistemas de gestores de bases de datos no relacionales

Son sistemas de gestión de bases de datos que no se basan en el modelo relacional tradicional, con código abierto que no empleaban tablas con filas y columnas ni el lenguaje SQL como mecanismo de consulta. Entre las características de las bases de datos no relacionales se encuentra que son distribuidas, de código abierto, escalables horizontalmente, no tienen esquemas, no utilizan SQL, ni uniones, no almacenan información en tablas de manera estructurada y utilizan la memoria principal para operar, lo que permite manejar grandes volúmenes de datos. Los Sistemas de Gestores de Bases de Datos no relacionales (NoSQL) más utilizados actualmente son: *MongoDB, Redis y Cassandra* (Peralta de Aparicio & Zurita Barrios, 2020).

4.1.4 Base de datos móvil

Una base de datos móvil desempeña un papel fundamental en la optimización del almacenamiento y la recuperación de datos en dispositivos pequeños con capacidad de almacenamiento limitada (Jianhong & Xinyue, 2020).

Por definición, una base de datos móvil es aquella que puede ser instalada y operada en un dispositivo de computación móvil, accediendo a través de redes inalámbricas similares a las utilizadas en entornos cliente-servidor. En este contexto, tanto el cliente como el servidor se comunican mediante conexiones inalámbricas, y se emplea una memoria caché para almacenar temporalmente las transacciones. El usuario puede acceder y actualizar la información en los directorios de inicio de un servidor o cliente de registros de una base de datos. Las bases de datos móviles permiten a los usuarios introducir información sobre la marcha. La información puede ser sincronizada con una base de datos de servidor posteriormente (Rivero Hernández et al., 2013).

Los sistemas de base de datos fueron desarrollados a partir de la necesidad de almacenar grandes cantidades de datos, a partir de la aparición de los dispositivos en el siglo XX. El incremento en el

uso de computadoras, portátiles, teléfonos móviles, dispositivos inteligentes y *tablets,* ha generado el manejo de una cantidad de datos en forma exponencial, que deben ser administrados de forma eficiente, para ello se han creado diferentes sistemas de almacenamiento de bases de datos y aplicaciones que se deben instalar y administrar en los diferentes dispositivos, siendo probable que a futuro su incremento sea mayor especialmente en los dispositivos móviles. Es evidente que cada tipo de aplicación requerirá el uso de una base de datos de algún tipo con la capacidad de descargar información y actuar sobre la misma aun cuando se esté desconectado (Rivero Hernández et al., 2013).

4.1.5 Sistemas de ficheros

Un sistema de ficheros o archivos es una estructura lógica, utilizada por un sistema operativo para organizar, gestionar, manejar y acceder a la información almacenada en dispositivos de almacenamiento, como discos duros o de estado sólido. Este sistema define cómo se almacenan, nombran, operan, acceden y protegen los diferentes archivos en el sistema, además de gestionar y organizar el espacio disponible. Su correcto funcionamiento es muy importante durante el proceso de inicio del sistema. Entre los sistemas de archivos más utilizados actualmente se encuentran: NTFS (*New Technology File System*, por sus siglas en inglés), FAT (*File Allocation Table*, por sus siglas en inglés) y EXT (*Extended File System*, por sus siglas en inglés) (Repiso et al., 2017).

4.1.6 Dispositivos móviles

Los dispositivos móviles son dispositivos electrónicos que amplían las interacciones más allá de las llamadas telefónicas. Permiten una amplia gama de funcionalidades, como tomar fotografías, enviar mensajes, jugar, leer y diseñar. Estos dispositivos poseen una potencia comparable a la computadora de escritorio o portátil, con su propio sistema operativo integrado (González Morte, 2019).

El sistema operativo de los sistemas móviles puede almacenarse en memoria NAND o NOR. Su código se ejecuta en memoria RAM. En la actualidad, los dispositivos móviles están equipados con microprocesadores del nivel del sistema, la cantidad de chips es mínima y su capacidad de memoria interna es hasta 64 GB. Las comunicaciones inalámbricas como *Bluetooth, Near Field Communication* o comunicación de campo cercano y WIFI están integradas para intercambiar datos (Baz Alonso et al., 2009).

Los dispositivos móviles se clasifican en dos tipos: Dispositivos con funciones de comunicación de voz y mensajería simple y, dispositivos inteligentes con capacidades y servicios avanzados para

multimedia, similares a un ordenador personal. Tanto los dispositivos con funciones de comunicación de voz y los dispositivos inteligentes son compatibles con los mensajes de texto más un conjunto de aplicaciones básicas de información personal. Los dispositivos inteligentes agregan capacidad de computador para poder ejecutar una gran variedad de aplicaciones (Baz Alonso et al., 2009).

Los dispositivos móviles poseen un tamaño compacto, funcionan con batería interna y son ligeros de peso. Un dispositivo móvil presenta las siguientes características: microprocesador, memoria de solo lectura (ROM), memoria de acceso aleatorio (RAM), módulo de radio, procesador de señal digital, micrófono y altavoz, variedad de teclas e interfaces de hardware y pantalla de cristal líquido (LCD) (Postigo Palacios, 2021).

4.1.7 Coeficientes de correlación

El concepto de asociación entre dos variables numéricas se fundamenta en el análisis del comportamiento conjunto que presentan los pares de valores. Esta relación se cuantifica mediante coeficientes de correlación. El interés principal radica en determinar si existe una tendencia a que los valores se agrupen según su magnitud.

Cuando los valores altos de ambas variables tienden a coincidir, al igual que los valores bajos, se dice que existe una correlación directa. En cambio, si los valores altos de una variable se asocian con los valores bajos de la otra, se trata de una correlación inversa.

Estas correlaciones no implican necesariamente una relación funcional explícita entre las variables, sino que describen un patrón general de asociación.

Entre los diferentes coeficientes, el coeficiente de correlación de Pearson es ampliamente utilizado, y solo en casos extremos - cuando su valor es exactamente 1 o -1 - la relación entre las variables puede representarse perfectamente mediante una línea recta. En situaciones menos ideales, esta suposición no se sostiene, por lo que puede ser más apropiado emplear el coeficiente de correlación de Spearman, que no asume linealidad y evalúa asociaciones basadas en rangos (Ortiz Pinilla et al., 2021).

4.1.7.1 Coeficiente de correlación de Spearman

La correlación de rangos de Spearman examina la relación entre dos variables, siendo estas consideradas no paramétrica de la correlación de Pearson, pero en este caso no se requiere una

distribución normal de los datos. Existe una diferencia importante entre ambos coeficientes de correlación. La correlación de Spearman utiliza los rangos de los datos en lugar de los datos en sí, de ahí el nombre de correlación de rangos (Kuckartz et al., 2013).

Con la ayuda del coeficiente se puede determinar la fuerza de la correlación y en qué dirección va la correlación. Si tenemos un coeficiente entre -1 y 0, existe una correlación negativa, es decir, una relación negativa entre las variables. Si tenemos un coeficiente entre 0 y 1, hay una correlación positiva, es decir, una relación positiva entre las dos variables. Si el resultado es 0, no tenemos correlación. También la fuerza se puede interpretar en intervalos generales así:

Si va de 0.8 a 1.0	Correlación muy fuerte positiva
Si va de 0.6 a 0.8	Correlación fuerte positiva
Si va de 0.4 a 0.6	Correlación moderada positiva
Si va de 0.2 a 0.4	Correlación débil positiva
Si va de 0.0 a 0.2	Correlación muy débil o nula
Si va de -0.2 a 0.0	Correlación muy débil o nula negativa
Si va de -0.4 a -0.2	Correlación débil negativa
Si va de -0.6 a -0.4	Correlación moderada negativa
Si va de -0.8 a -0.6	Correlación fuerte negativa
Si va de -1.0 a -0.8	Correlación muy fuerte negativa

4.2 Antecedentes

El autor Díaz Achina, (2022) en su artículo propone utilizar bases de datos NoSQL, las cuales previamente fueron evaluadas y comparadas para realizar un modelo óptimo de almacenamiento y búsqueda de información para una aplicación que mejore la movilidad pública y privada, en el cual se desarrolla una aplicación móvil usando el modelo MobileD. Sin embargo, el autor no evaluó la posibilidad de manejar bases de datos nativas en los dispositivos móviles con el fin de poder garantizar el óptimo funcionamiento en caso de no tener conexiones a internet,

Tesone (2021), en su trabajo, llevó a cabo un estudio experimental mediante el desarrollo de una aplicación de prueba que implementa funcionalidades equivalentes utilizando SQLite y *CouchbaseLite*. A través de esta implementación, evaluó aspectos como la facilidad de uso, la legibilidad del código y la adaptabilidad al modelo de datos. Los resultados obtenidos señalan que SQLite, particularmente cuando se utiliza en conjunto con la librería *Room*, favorece una mayor

claridad y estructura en la abstracción de datos, mientras que *CouchbaseLite* presenta ventajas en escenarios donde los modelos de datos son dinámicos, poco estructurados o contienen propiedades opcionales. Este análisis resulta relevante para orientar la selección del motor de base de datos más adecuado según las características de la aplicación móvil a desarrollar; el autor no evaluó los factores de replicación en los diferentes dispositivos ni tampoco se tuvo en cuenta la forma en que se comporta la estructuración de ficheros binarios en estos motores de bases de datos.

Pozo Puñal (2019), en su artículo manifestó que debido al incremento de información constante se requiere crear un sistema de ficheros basado en la base de datos NoSQL *apache Cassandra* que cumpla con el estándar POSIX de IEEE con el fin de evaluar el rendimiento y distribución de archivos vs el Sistema Tradicional de Ficheros HDFS (*Hadoop Distributed File System* por su sigla en inglés), En este sistema no se evaluó la posibilidad de generar un sistema similar en dispositivos móviles ya que la infraestructura puede ser compatible en un modelo Mobile para dispositivos inteligentes.

Wałachowski y Kozieł (2020), en su trabajo, llevaron a cabo un análisis comparativo entre sistemas de gestión de bases de datos móviles relacionales y no relacionales en Android, evaluando específicamente *SQLite, SnappyDB, Realm y ObjectBox*. Para ello, implementaron una aplicación que realizaba operaciones básicas como inserción, edición, lectura y eliminación de datos, a fin de medir el rendimiento en condiciones controladas. Los resultados del estudio indicaron que SQLite, a pesar de su antigüedad, ofreció un rendimiento superior en tareas de edición y recuperación de datos, posicionándose como una opción eficiente y confiable. En contraste, las bases de datos NoSQL analizadas presentaron ventajas en cuanto a simplicidad de uso y menor necesidad de configuración, pero con tiempos de ejecución ligeramente mayores. Este análisis permite visualizar las fortalezas y limitaciones de cada tecnología en función del tipo de operación y contexto de uso en aplicaciones móviles. Sin embargo, no tuvieron en cuenta los tiempos de la replicación de información y como usarlo en un modelo desconectado del servidor ya sean de archivos binarios o de información simple.

Pozo Puñal E, (2019), describe un sistema que opera de manera paralela al sistema de archivos normal basado en POSIX. Este sistema personalizado utiliza la base de datos Apache *Cassandra*, lo que le proporciona los beneficios de una base de datos NoSQL para el manejo de información y superando a HDFS en lectura escritura cuando se tiene un clúster con ocho nodos. Sin embargo, es importante mencionar que este sistema fue implementado exclusivamente en entornos de nube y para el procesamiento de *bigdata* con el sistema HDFS. No obstante, este tipo de alternativas no se han explorado para dispositivos móviles.

Chingo Esquivel y López Sevilla (2021) comparan las bases de datos relacionales (SQL) con las no relacionales (NoSQL), señalando que, aunque ambas buscan almacenar y gestionar datos, lo hacen desde filosofías distintas: mientras que las bases relacionales se fundamentan en estructuras tabulares y en los principios ACID (Atomicidad, Consistencia, Aislamiento y Durabilidad), las no relacionales priorizan la flexibilidad, la escalabilidad horizontal y la gestión de datos no estructurados, como documentos o pares clave-valor. Los autores destacan que la mayoría de los estudios existentes se centran únicamente en el rendimiento, por lo que proponen una comparación enfocada en los aspectos de seguridad, empleando para ello una metodología experimental basada en servidores en la nube, con PostgreSQL como representante de SQL y MongoDB como exponente de NoSQL. Concluyen que, a pesar de sus diferencias arquitectónicas, ambos sistemas de gestión de bases de datos (DBMS) ofrecen garantías fundamentales en términos de autenticación, confidencialidad, concurrencia, integridad y disponibilidad. No obstante, el estudio omite otros factores igualmente relevantes en contextos móviles, como la capacidad de replicación entre dispositivos, el comportamiento de los DBMS en condiciones de conectividad intermitente y la evaluación de sistemas nativamente diseñados para funcionar en entornos móviles, lo que abre la necesidad de investigaciones complementarias orientadas a estos escenarios.

Pérez Román (2020), en su trabajo, desarrolló tres aplicaciones que implementan funcionalidades equivalentes utilizando PostgreSQL (relacional), MongoDB (NoSQL) y Kaleido (Blockchain). A través de estas implementaciones, evaluó aspectos como el rendimiento en términos de tiempos de ejecución y la complejidad de desarrollo. Los resultados obtenidos señalan que PostgreSQL ofrece un rendimiento superior y una menor complejidad en su implementación, mientras que Kaleido, aunque proporciona ventajas en términos de seguridad y descentralización inherentes a la tecnología *Blockchain*, presenta mayores desafíos en su desarrollo y tiempos de respuesta más elevados. Este análisis resulta relevante para orientar la selección del sistema de gestión de bases de datos más adecuado según las características y necesidades específicas de la aplicación a desarrollar.

Pozo Puñal (2019), en su trabajo, aborda la creciente necesidad de soluciones de almacenamiento escalables ante el vertiginoso crecimiento de datos en sectores como el Internet de las cosas, la salud y la meteorología. El autor resalta cómo la producción diaria de datos ha alcanzado volúmenes de hasta 2.5 exabytes, lo que ha impulsado el desarrollo de tecnologías como Apache Cassandra. Esta base de datos NoSQL se destaca por su capacidad de operar en entornos distribuidos con alta disponibilidad, tolerancia a fallos y escalabilidad horizontal, permitiendo mantener el rendimiento aun ante la pérdida de nodos. No obstante, aunque su propuesta resulta sólida en escenarios de Big Data y sistemas distribuidos, el estudio no contempla el análisis de estos sistemas en dispositivos

móviles Android ni los compara directamente con gestores de ficheros tradicionales. En contraste con el enfoque de la presente tesis, el trabajo no evalúa la eficiencia, consumo de recursos ni el rendimiento en contextos móviles, donde los desafíos están condicionados por restricciones de energía, conectividad y procesamiento. Tampoco se examinan otras alternativas NoSQL más ligeras como: *Couchbase Lite, Realm o SnappyDB*, que son especialmente diseñadas para su ejecución en dispositivos móviles, ni se estudia la viabilidad del uso de sistemas de ficheros tradicionales frente a estas soluciones, aspecto central del presente estudio.

Conforme pasan los años, va aumentando la necesidad de almacenar grandes cantidades de información a medida que también va aumentando el tamaño de los datos que los usuarios tienen a su disposición. Para ello, han surgido diferentes sistemas de ficheros, bases de datos o sistemas de almacenamiento que permiten a estos usuarios guardar ficheros o información cada vez más grandes.

La cantidad de información con la que se trabaja durante los últimos años ha ido incrementándose con el avance de la tecnología gracias a la aparición de nuevos dispositivos electrónicos y al desarrollo de nuevas tecnologías que permiten crear datos cada vez más grandes, entre ellos los dispositivos móviles. Esto ha provocado que, junto con el aumento de nuevos usuarios que tienen acceso a Internet, haya cada vez una mayor cantidad de datos que se envían entre diferentes puntos de la red a velocidades cada vez mayores. Para poder almacenar toda esta información diaria, fue necesaria la creación de sistemas de ficheros o de bases de datos que no sigan los métodos tradicionales y no tengan limitaciones en cuanto a capacidad ni a velocidades de cómputo y respuesta.

Se ha encontrado una gran cantidad de estudios sobre bases de datos SQL y NoSQL con condiciones muy favorables. Sin embargo, no se han identificado investigaciones que realicen comparaciones directas entre ambas tecnologías, orientadas a determinar posibles soluciones al problema del almacenamiento de datos. La mayoría de los trabajos se han centrado en modelos tradicionales de gestión de la información, sin considerar en profundidad las particularidades de estos gestores de bases de datos a la hora de manejar información binaria o con estructuras dinámicas y cambiantes.

Este contexto motivó la realización de un estudio comparativo con el objetivo de evaluar de forma objetiva el rendimiento del sistema de ficheros tradicional frente al sistema de bases de datos NoSQL nativo para dispositivos móviles.

La elección de las bases de datos NoSQL se fundamenta en dos de sus características principales: la escalabilidad y la flexibilidad para definir tanto los datos como los esquemas. Adicionalmente, el estudio busca analizar las capacidades de replicación, comparando escenarios donde se realiza

mediante transferencia de archivos convencionales frente a aquellos gestionados de forma centralizada por una base de datos NoSQL.

Actualmente, observamos una amplia aceptación de las bases de datos relacionales como SQLite en el ámbito de los dispositivos móviles. No obstante, aún no se han explorado en profundidad los escenarios potenciales para la implementación de bases de datos NoSQL. Estas podrían ofrecer ventajas significativas, como una mayor capacidad para el almacenamiento de datos en modo offline y una gestión más eficiente y centralizada de archivos multimedia, permitiendo un acceso más rápido e integral a la información por parte de los usuarios.

CAPITULO 5
METODOLOGÍA

5.1 Diseño de la investigación

La investigación realizada en este trabajo se clasifica como de tipo experimental, exploratoria y con enfoque cuantitativo, debido a que compara diferentes tipos de bases de datos no relacionales (NoSQL) frente a gestores de ficheros tradicionales. A partir de esta comparación se puede obtener conocimientos clave que se pueden utilizar para la realización de estudios de carácter más profundos sobre problemas relacionados con las bases de datos no relacionales en dispositivos móviles.

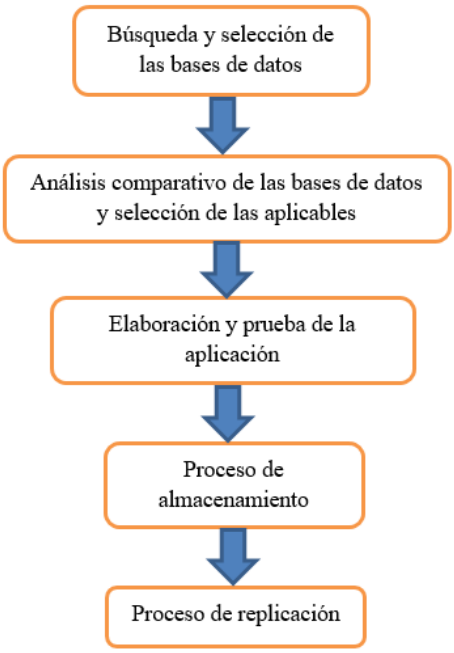
El objetivo principal es la "*Evaluación de las características velocidad de acceso y replicación de los sistemas de base de datos no relacionales (NoSQL) y gestor de ficheros tradicional en dispositivos móviles* Android *mediante un análisis comparativo basado en un caso de estudio*", en busca de ser eficientes en términos del tiempo de inserción, búsqueda y replicación de la información, utilizando las bases de datos NoSQL (*ObjectBox, Realm, SnappyDB, DbO4, Couchbase Lite*), identificadas en el estado del arte como las tecnologías relacionadas con los sistemas de almacenamiento de datos en los dispositivos móviles.

En este estudio se incluye un análisis comparativo de los procesos de inserción, búsqueda y replicación de almacenamiento de datos en los sistemas de ficheros tradicionales y las bases de datos NoSQL en dispositivos móviles Android. Este análisis se centra en calcular el tiempo de procesamiento de cada base de datos NoSQL y del sistema de gestor de ficheros tradicional, en función de las variables seleccionadas para el estudio. Dichos sistemas de almacenamiento se integran posteriormente en una función que permite compararlas mediante el coeficiente de Spearman, a fin de identificar las que tienen mayor peso en cada sistema, considerando su comportamiento monotónico.

La estructura metodológica de investigación en este trabajo de grado se encuentra representada en la **Figura 1**, la cual detalla las fases, procesos y relaciones que guían el desarrollo adoptado para el estudio. Esta figura sirve como referencia para la planeación, ejecución y análisis de los datos obtenidos y permite establecer la base que asegura la validez y la replicación del enfoque metodológico empleado.

Figura 1:

Elementos de la estructura metodológica de la investigación.



Fuente: Elaboración propia

A continuación, se explican las diferentes fases del diseño metodológico mostrado en la **Figura 1.**

Búsqueda y selección de las bases de datos

Se fundamenta en la identificación de sistemas de gestión de bases de datos (SGBD) que respondan a los requerimientos funcionales y no funcionales en la elaboración de este trabajo. Este proceso implica el análisis de múltiples criterios, entre los que se destacan: el modelo de datos no relacional (NoSQL), la escalabilidad, el rendimiento, la disponibilidad, el soporte para transacciones, la facilidad de integración y la comunidad de soporte.

Para llevar a cabo la selección adecuada, se realiza una revisión sistemática de la literatura y de documentación técnica de proveedores reconocidos, así como comparaciones especializadas de revistas científicas, comunidades académicas y tecnológicas. Además, se consideran casos de uso reales y estudios de adopción en la industria.

La elección final de las bases de datos está dada por su capacidad para adaptarse al entorno de implementación en los dispositivos móviles, así como en su compatibilidad con las tecnologías empleadas en el trabajo. También, se considera bases de datos móviles o distribuidas, el soporte para sincronización, tolerancia, fallos, consumo de recursos y facilidad de mantenimiento.

Análisis comparativo de las bases de datos y selección de las aplicables

El análisis comparativo de las bases de datos se realizó con el propósito de identificar aquellas que mejor se ajustan a los requerimientos técnicos y operativos establecidos en este trabajo. Para ello, se evaluaron diversas soluciones, incluyendo tanto el sistema de archivos tradicional como bases de datos NoSQL, considerando parámetros clave. El conjunto de datos consistió en imágenes de tamaño uniforme (en kilobytes), y se realizaron pruebas variando la cantidad de datos de 50 en 50 hasta un máximo de 300. Cada escenario fue ejecutado diez iteraciones, modificando variables como la memoria RAM, el número de procesadores, la versión del sistema operativo y el tipo de sistema de almacenamiento.

La metodología aplicada para esta comparación incorpora el análisis de correlación mediante el coeficiente de Spearman (r_s), con el propósito de evaluar la existencia y magnitud de relaciones monotónicas entre las variables seleccionadas. El proceso inició con la recopilación y normalización de los datos, asegurando su integridad y calidad para el análisis estadístico. Se calcula este coeficiente entre la variable tiempo y cada una de las variables explicativas previamente mencionadas: la memoria RAM, el número de procesadores, la versión del sistema operativo y el tipo de sistema de almacenamiento.

Para este trabajo se formularon las hipótesis para determinar cuál de las variables independientes tiene mayor peso en la inserción, búsqueda y replicación en un modelo de regresión múltiple sobre la variable dependiente. La hipótesis nula (H_o) no existe diferencia significativa entre los sistemas de almacenamiento NoSQL y los sistemas de ficheros tradicionales ($\beta_{NoSQL,i} = \beta_{ficheros,i}$). La hipótesis alternativa (H_1) al menos uno de los sistemas de almacenamiento presenta un comportamiento significativo entre los sistemas de almacenamiento NoSQL y los sistemas de ficheros tradicionales ($\exists\ al\ menos\ un\ \beta_{NoSQL,i} \neq \beta_{ficheros,i}; max\beta_{NoSQL,i} > \beta_{ficheros,i}$). Se establece un nivel de significancia $\alpha = 0.05$.

Para identificar específicamente qué pares de sistemas de almacenamiento de datos muestran diferencias significativas, se implementó el análisis Post-Hoc de Tukey. Donde los resultados obtenidos proporcionan: Comparaciones por pares entre todos los sistemas de almacenamiento; la diferencia media entre los diferentes grupos, los intervalos de confianza a un nivel del 95 % y el valor de la probabilidad “p” ajustado para múltiples comparaciones.

Los criterios de decisión utilizados para la comparación entre pares son: si $p - valor < \alpha(0.05)$ se rechaza la hipótesis nula (H_o) para ese par específico, y si $p - valor \geq \alpha(0.05)$ no hay evidencia para rechazar la hipótesis nula (H_o).

Con la metodología adoptada para el análisis comparativo en la prueba de hipótesis se permite:

1. Determinar si existen diferencias globales mediante (ANOVA).
2. Identificar exactamente qué pares de sistemas difieren (Tukey).
3. Cuantificar la magnitud de esas diferencias.
4. Controlar la tasa de error tipo I en comparaciones múltiples.

Elaboración y prueba de la aplicación

La elaboración de la aplicación se llevó a cabo siguiendo un enfoque sistémico, lo que facilitó el avance ordenado de las fases de diseño, codificación y validación. Durante el desarrollo, se utilizaron herramientas y entornos de desarrollo adecuados para implementar las funcionalidades definidas en los requisitos del sistema, asegurando la escalabilidad y mantenibilidad del código.

Una vez completada la fase de desarrollo, se procedió a la ejecución de pruebas, que incluyeron pruebas de integración y pruebas funcionales. Las pruebas de integración evaluaron la correcta interacción entre los diferentes componentes del sistema, mientras que las pruebas funcionales se orientaron a verificar el cumplimiento de los requisitos establecidos.

Los resultados obtenidos durante las pruebas permitieron identificar y corregir errores, optimizar el rendimiento y mejorar la experiencia de usuario. Este proceso garantizó que la aplicación alcanzara un nivel óptimo de calidad, confiabilidad y usabilidad antes de su implementación final.

Proceso de almacenamiento

La aplicación que se desarrolló mide el tiempo que tarda el proceso de almacenamiento dentro de un sistema informático. El proceso de almacenamiento en sistemas informáticos consiste en la gestión organizada y eficiente de datos digitales dentro de dispositivos físicos o virtuales. El almacenamiento se realiza en estructuras jerárquicas que permiten localizar y acceder a los datos de manera eficiente.

Durante este proceso intervienen varios componentes críticos: el sistema de archivos organiza los datos en bloques, el controlador de almacenamiento gestiona la comunicación entre el sistema

operativo y el dispositivo, y la memoria de caché optimiza el acceso mediante el almacenamiento temporal de datos utilizados.

Proceso de replicación

La aplicación que se desarrolló también tiene como objetivo medir el proceso de replicación. El proceso de replicación en bases de datos consiste en la duplicación y sincronización de datos entre múltiples servidores o nodos, con el objetivo de mejorar la disponibilidad, tolerancia a fallos y rendimiento del sistema. Este mecanismo permite que una copia exacta de la base de datos principal (llamada nodo primario o master) sea distribuida a uno o más nodos secundarios (replicas o *slaves*), garantizando la continuidad del servicio en caso de fallos del nodo principal y permitiendo la distribución de la carga de lectura.

La estrategia de replicación utilizado es la asíncrona, en la cual los cambios se propagan con cierto retraso, priorizando el rendimiento sobre la consistencia en tiempo real. Este método en bases de datos NoSQL, que implementan réplicas en clústeres distribuidos para facilitar la escalabilidad horizontal y la alta disponibilidad.

La replicación también juega un papel fundamental en estrategias de recuperación ante desastres, copias de seguridad en tiempo real y balanceo de cargas, y es una práctica esencial en entornos empresariales donde la resiliencia de datos y el acceso continuo son requisitos críticos.

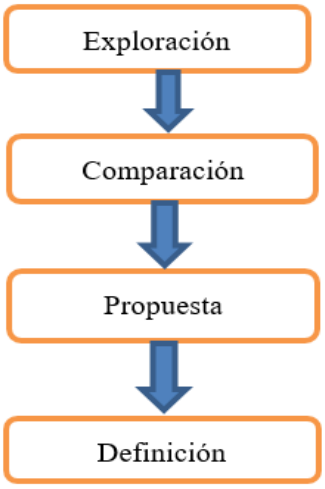
Aunque la simulación de replicación mediante *WebSockets* ofrece una aproximación válida para efectos comparativos entre los sistemas evaluados, es importante señalar que este enfoque puede no reflejar con exactitud los mecanismos de replicación nativos implementados por cada base de datos NoSQL. Por lo tanto, los resultados obtenidos deben interpretarse considerando esta diferencia en el modelo de replicación.

5.2 Metodología de intervención

En la metodología de intervención se describen las tareas llevadas a cabo teniendo en cuenta la recolección de la información y los conocimientos previos, para dar cumplimento con los objetivos específicos. La metodología de investigación se ilustra en la **Figura 2**.

Figura 2

Metodología de intervención



Fuente: Elaboración propia

A continuación, se explica las diferentes fases de la metodología de intervención ilustrada en la **Figura 2**.

Exploración

En el proceso de exploración se buscó identificar y analizar las tecnologías asociadas a los sistemas de almacenamiento de datos y a las bases de datos no relacionales (NoSQL) empleadas en entornos de dispositivos móviles. Aquí se permitió establecer un sistema actualizado sobre las soluciones esperadas para la gestión eficiente de datos no estructurados en plataformas móviles, considerando aspectos como el rendimiento, la escalabilidad, la sincronización y la compatibilidad con arquitecturas distribuidas.

Actividades realizadas:

Se revisó el sistema de ficheros y los sistemas de almacenamiento de datos NoSQL con sus características de la implementación entre ellas: *Couchbase Mobile, Snappy, DB4o, Realm, y ObjectBox*, mediante estudios primarios realizados en periodos no superiores a cinco años, en formularios de extracción de información en diferentes bases de datos tales como: IEEE, *Google Scholar*, ACM, *ScienceDirect y Couchdb.apache.org*.

Comparación:

Con la comparación se contrastaron las tecnologías relacionadas en la exploración realizada con los sistemas de almacenamiento NoSQL y sistemas de gestor de ficheros en los dispositivos móviles.

Actividades realizadas:

Se revisaron tiempos de respuesta, considerando el alcance y las especificaciones técnicas de sistema de gestor de ficheros tradicional y sistemas de almacenamiento NoSQL (*Couchbase, Mobile, Snappy, DB4o, Realm y ObjectBox);* y el alcance del sistema de transferencias *WebSockets* o comando de directorios, utilizando cuadros comparativos de las tecnologías utilizadas en el almacenamiento en los dispositivos móviles, presentados en la literatura para la extracción de datos.

Propuesta de recolección de datos:

Para la propuesta de recolección de datos se desarrolló una aplicación que permite comparar el tiempo de procesamiento de un sistema de ficheros tradicional con cada una de las bases de datos NoSQL.

A continuación, se presenta una lista de actividades realizadas:

- Se analizaron los objetos a almacenar considerando un *dataset* de imágenes multimedia, la mayoría normalizadas.
- Se realizó diseño y análisis de la aplicación móvil.
- Se revisó los diagramas, las tablas y los requerimientos necesarios para el cumplimiento del objetivo.
- Se realizó la aplicación para los dispositivos móviles.
- Utilizando la aplicación se tomaron tiempos de una base de datos almacenando cada uno de ellos, a diferente cantidad de objetos y tamaño en kilobytes para probar el sistema mediante el análisis del comportamiento en su ejecución.
- Utilizando la aplicación se calcularon tiempos de almacenamiento de información en todos los sistemas (ext4 fichero y bases de datos NoSQL) a diferente cantidad de objetos y tamaño en bytes para probar el sistema mediante el análisis del comportamiento en su ejecución.
- Se variaron los sistemas operativos y se tomaron nuevamente tiempos con la aplicación almacenando cada uno de ello a diferente cantidad de objetos y tamaño en kilobytes para probar el sistema mediante el análisis del comportamiento en su ejecución.
- Se varió la memoria RAM y se tomaron nuevamente tiempos con la aplicación almacenando cada uno de ello a diferente cantidad de objetos y tamaño en kilobytes para probar el sistema mediante el análisis del comportamiento en su ejecución.
- Se varió los procedimientos y se tomaron nuevamente tiempos con la aplicación almacenando cada uno de ello a diferente cantidad de objetos y tamaño en kilobytes en todos los sistemas de almacenamiento para probar el sistema mediante el análisis del comportamiento en su ejecución.

De tal manera, se realizaron todas las combinaciones de las variables memoria RAM, número de procesadores, versión del sistema operativo y tipo de sistema de almacenamiento.

Definición

Para la definición y selección del mejor sistema de almacenamiento de datos, se evaluó el sistema de almacenamiento de archivos mediante ficheros tradicional contra el almacenamiento de información en las bases de datos no relacionales en dispositivos móviles en su velocidad de respuesta.

A continuación, se presenta una lista de actividades realizadas:

- Se evaluó el almacenamiento mediante ficheros de forma tradicional y sistemas NoSQL usando *Timer y Loggers*, mediante la librería *Log4j*.
- Se evaluó el tiempo de inserción, búsqueda y replicación en almacenamiento mediante sistemas NoSQL y gestor de ficheros, usando *Timer* y *Loggers*. La información obtenida se mostró en tablas comparativas de tiempo de procesamiento luego exportadas en formato *Json*.
- Se realizaron análisis y comparaciones con los resultados obtenidos para los diferentes sistemas utilizando la Suite de análisis estadístico de Python.

5.3 Población y muestra

5.3.1 Población

Estadísticamente se conoce como población el conjunto de todos y cada uno de los elementos u objetos en estudio, para nuestro estudio serán los dispositivos móviles que contenga Android 10 como mínimo y que permitan manejar al menos 50 ficheros aleatorios, con el fin de poder generar las comparaciones antes mencionadas.

5.3.2 Muestra

Estadísticamente la muestra es el conjunto de la cantidad de elementos considerados para la investigación. Para nuestro estudio consideramos los procesos de almacenamiento del sistema de ficheros tradicional y el sistema de almacenamiento de bases de datos no relacionales (*ObjectBox, Realm, SnappyDB, DbO4, Couchbase Lite*)**.**

5.4 Técnicas e instrumentos de recolección de datos, validez y confiabilidad

5.4.1 Técnicas

Mediante la técnica de observación, y con base en conocimientos previos, es posible realizar de manera ágil un análisis preliminar de la situación a intervenir, lo cual resulta ideal para un proceso comparativo entre el sistema de almacenamiento de ficheros tradicionales y el sistema de almacenamiento basado en bases de datos no relacionales NoSQL.

Este tipo de investigación no requiere el uso de instrumentos estadísticos de recolección de información, como encuestas o entrevistas, dado que se fundamenta en mediciones directas que permiten obtener métricas de rendimiento del software. Estas métricas son esenciales para comparar los métodos de almacenamiento de información utilizados en cada base de datos e incluyen: la velocidad de procesamiento del equipo, el tiempo de respuesta ante operaciones de inserción, búsqueda y replicación (medido en milisegundos) y el tiempo total de procesamiento de la información (medido en milisegundos).

La confiabilidad se obtiene mediante el procesamiento del mismo software probado en diferentes condiciones y características, procesado en bloques de 50 hasta 300 imágenes algunas de ellas normalizadas, que generaban un tamaño igual en bytes para cada proceso.

5.4.2 Instrumentos

Los instrumentos utilizados en este estudio corresponden a herramientas diseñadas para recopilar datos pertinentes y dar respuesta a los objetivos planteados. En este caso, se aplicaron instrumentos de naturaleza cualitativa, cuantitativa y técnica para evaluar el rendimiento de diversas bases de datos en dispositivos móviles en relación con las operaciones de inserción, búsqueda y replicación. Entre los instrumentos empleados se incluyen, registros de *logs* del sistema para la captura de tiempos de ejecución, cronómetros de software para medición de tiempos de respuesta (en milisegundos), e instrumentos para validar los resultados obtenidos.

A continuación, se presenta una lista de instrumentos utilizados:

- Aplicación propuesta en lenguaje Java, que utiliza las de bases de datos NoSQL (*ObjectBox*, *Realm*, *SnappyDB*, *DbO4*, *CouchBaseLite*). Estos programas fueron fundamentales para obtener métricas de rendimiento como el tiempo de inserción (insertmiliTime), tamaño de inserción (insertsizekilobyte), y consumo de recursos (RAM, CPU).

- Scripts en lenguaje Python, encargado de realizar cálculos y operaciones con la información obtenida para el análisis estadístico de la información, sobre diferentes motores de bases de datos. Estos scripts fueron fundamentales para el análisis de las métricas de rendimiento como el tiempo de inserción (insertmiliTime), tamaño de inserción (insertsizekilobyte), entre otros.
- Registros de sistema y herramientas de monitoreo (Android 10, 12 y 13) que permitieron medir el comportamiento del dispositivo durante las pruebas por medio de *logs*, incluyendo uso de memoria, número de procesadores activos y versión del SDK.
- Se utilizó el método *one hot code* para la conversión de variables categóricas en un formato binario que permite proporcionar los algoritmos de aprendizaje automáticos para mejorar la predicción y normalizar los valores.
- Matrices de correlación y regresión lineal múltiple, generadas con bibliotecas estadísticas como pandas, *scikit-learn y matplotlib*, utilizadas para identificar la relación entre variables y determinar el impacto de diferentes factores sobre el rendimiento de las bases de datos.
- Se utilizaron herramientas estadísticas tales como el rango intercuartílico (IQR), la media, mediana, cuartiles, coeficientes para la regresión múltiple.
- Para el análisis de las hipótesis se consideró la herramienta estadística de *Boxplots y* los resultados de Tukey.

5.4.3 Métodos de análisis de datos

Los datos obtenidos se comparan mediante análisis estadísticos tales como el coeficiente de correlación de Spearman, comportamientos gráficos y el análisis de las líneas de regresión múltiple para determinar los comportamientos de las diferentes variables o características (bases de datos, versiones de dispositivos móviles, tamaño del archivo, número de núcleos, entre otras expresadas en la ecuación).

Resumen metodológico

El **Gráfico 3** presenta el diagrama de flujo que resume la metodología empleada para evaluar el rendimiento de diferentes sistemas de almacenamiento en dispositivos móviles, contemplado las métricas *insert, search, replication insert y replication search*, siguiendo una secuencia estructurada de actividades desde la preparación de datos hasta la obtención de resultados finales.

Inicio del proceso: El proceso inicia con la preparación del *dataset*, que consiste en organizar el conjunto de datos (imágenes) que serán utilizados durante las pruebas. Se empleó un conjunto de

datos compuesto por 400 imágenes normalizadas, comprimidas en un archivo ZIP para determinar los tiempos de *insert, search, replication insert y replication search*.

Configuración del entorno de pruebas: Se realiza la configuración de memoria RAM (4 y 6 GB) y SDK Versión (29; 31 y 33), ajustando las condiciones del entorno móvil (dispositivo físico o emulador) y las versiones de software necesarias para garantizar la coherencia en las pruebas. Se realizaron las operaciones con inserciones escalonadas en bloques de 50 en 50 hasta 300, evaluados en los diferentes sistemas de almacenamiento de bases de datos NoSQL y sistemas de ficheros.

Operaciones principales: A continuación, se definen las operaciones principales que constituyen el núcleo experimental:

- Inserción de datos: uso de bloques de 50 a 300 imágenes para evaluar el rendimiento en operaciones de inserción de los datos.
- Búsqueda de datos: pruebas realizadas sobre subconjuntos de 10 imágenes, simulando consultas frecuentes.
- Replicación de inserción de datos y replicación de búsqueda de datos: pruebas replicadas en entornos con emuladores y comunicación mediante *WebSockets* para simular escenarios de sincronización.

Ejecución Experimental: Cada prueba se ejecuta con una repetición de cinco veces (x5) para garantizar la robustez estadística de los resultados.

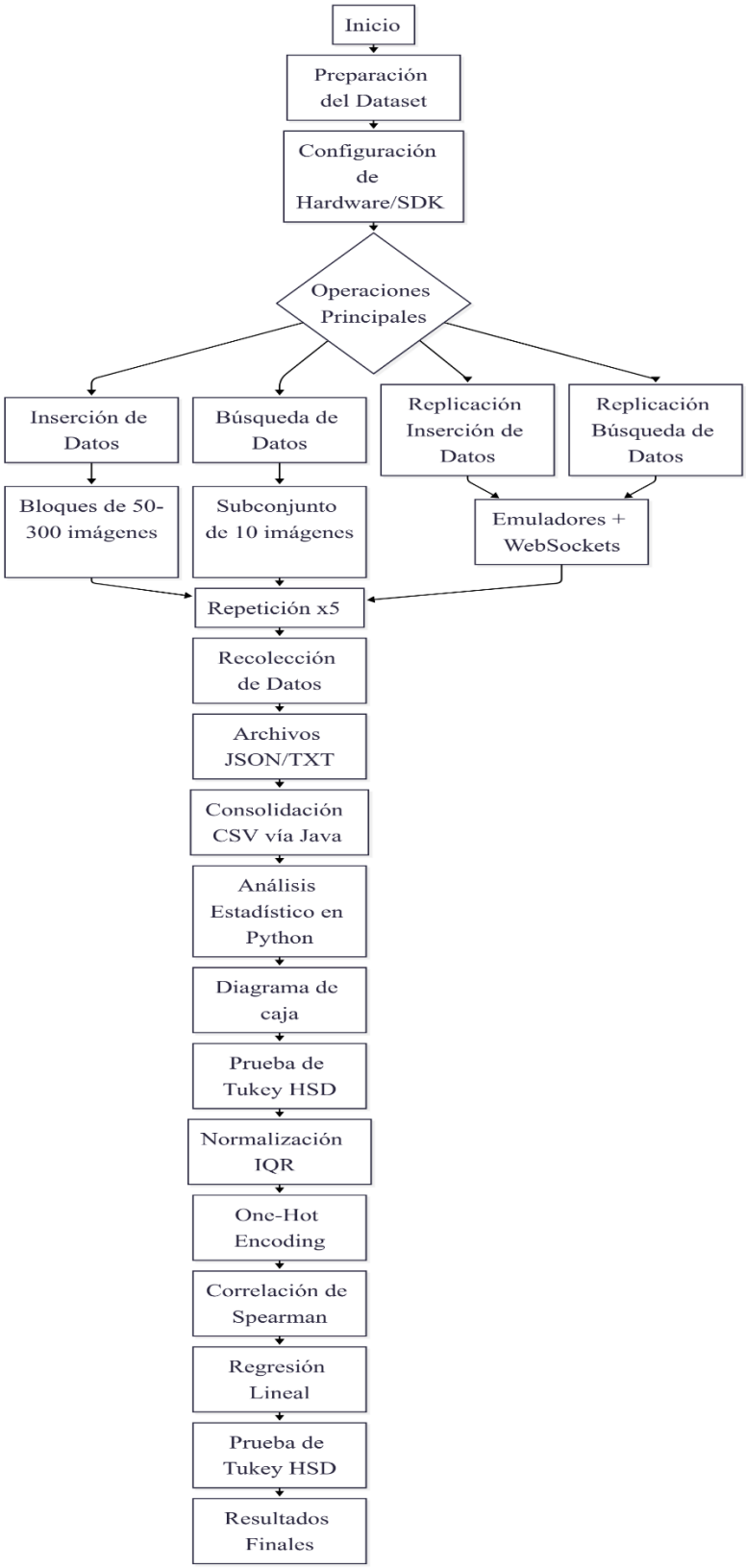
Recolección y Consolidación de Datos: Los datos generados se almacenan en archivos JSON/TXT y posteriormente se consolidan en formato CSV mediante una aplicación en Java para Android para facilitar el análisis posterior.

Análisis Estadístico: El análisis se realiza en Python e incluye las siguientes etapas:

- Limpieza de datos: eliminación de columnas irrelevantes, valores atípicos, nulos o inconsistentes y normalización IQR.
- Estadísticas descriptivas: cálculo de métricas básicas (media, mediana, desviación estándar y cuartiles).
- Diagramas de caja (*Boxplots*): visualización de la dispersión y valores atípicos.
- Codificación *One Hot Encoding*: transformación de variables categóricas para su inclusión en modelos de análisis.

- Modelado y Correlación: Se aplica la correlación de Spearman para identificar relaciones no lineales entre variables, y posteriormente se realiza una regresión lineal múltiple para modelar el comportamiento de las variables dependientes respecto a las independientes.
- Resultados Finales: Finalmente, se obtienen los resultados finales del análisis, que sirven de base para comparar el rendimiento de los distintos sistemas de almacenamiento evaluados.

Figura 3:
Resumen de la metodología de intervención.



Fuente: Elaboración propia

CAPITULO 6
RESULTADOS Y ANÁLISIS DE DATOS EXPERIMENTALES

6.1 Identificar las tecnologías relacionadas con los sistemas de almacenamiento de datos en los dispositivos móviles.

Los sistemas de bases de datos relacionales y no relacionales (NoSQL) para los dispositivos móviles encontrados en el estado del arte, se clasificaron de mayor a menor, considerando disponibilidad según plataformas y *frameworks* de desarrollo multiplataforma, información sobre el desarrollo y otras características específicas de las bases de datos consideradas para el estudio relevantes en este trabajo. Ver **Tabla 1**.

Tabla 1

Base de datos no relacionales para el estudio y Sistema de almacenamiento

Característica / Sistema	Couchbase Lite	Realm	ObjectBox	SnappyDB	db4o	Sistema de ficheros EXT4
Tipo	NoSQL Documental	NoSQL Orientado a objetos	NoSQL Orientado a objetos	Clave - valor	Orientado a objetos	Sistema de archivos (no BD)
Lenguaje nativo	JSON	Objetos (nativo)	Objetos (nativo)	String/Byte	Objetos (Java)	N/A
Sincronización en la nube	Con Couchbase Sync Gateway	Realm Sync	Con ObjectBox Sync	N/A	N/A	N/A
Persistencia local	Si	Si	Si	Si	Si	Si
Consultas	SQL-like (N1QL)	API orientado a objetos	API fluido en Java/Kotlin	No consultas solo por clave	Consultas por objetos	N/A
Velocidad	Buena Dependiendo del tamaño	Muy rápida	Muy rápida con índice automático	Muy rápida en claves directas	Buena en estructuras simples	Muy rápida para lectura/escritura
Consumo de recursos	Medio	Bajo	Bajo	Muy bajo	Medio	Bajo
Tamaño del binario	Grande (~5MB+)	Moderado (~2MB)	Pequeño (~1MB)	Muy pequeño menores 500KB	Antiguo, peso variable	N/A
Ideal para	Apps con sincronización offline	Apps móviles con objetos nativos	Apps IoT / móviles de alto rendimiento	Almacenamiento simple	Proyectos académicos /experimentales	Almacenamiento general de archivos

Fuente: Elaboración propia.

ObjectBox: Es una base de datos orientada a objetos con soporte para Android (entre otras plataformas) y que aboga por la simplicidad de su uso, la rapidez en sus operaciones y el reducido impacto en el rendimiento; Es rápida, eficiente y fácil de usar, ideal para Aplicaciones móviles modernas con necesidades de rendimiento (https://objectbox.io/mobile/).

Es clave el reducido impacto en el rendimiento debido a que prevé el uso de la aplicación sobre los dispositivos con capacidad de procesamiento reducida, en otras palabras, para dispositivos móviles Android de gama baja. Para su uso se debe considerar el repositorio de *jcenter*, bajo varios conceptos:

- **Entidad:** Se trata de clases que definen modelos físicos del objeto a guardar en la base de datos; para su correcta identificación e indexación se debe tener un campo de tipo *long*.
- ***ObjectBoxRepository*:** Es el capaz de devolver el curso instalado.
- ***Box:*** Es una clase genérica capaz de almacenar objetos de un determinado tipo.
- ***Mapper*:** es utilizado para transformar en un objeto dominio (Croche Andreu, 2021).

Realm: Es un sistema de gestión de bases de datos DBMS embebido que utiliza un modelo de datos orientado a objetos. Se puede utilizar en forma autónoma, o también de forma sincronizada con la base de datos *backend MongoDB*, ofreciendo la posibilidad de trabajar con objetos vivos (*live objects*) que se actualizan automáticamente ente cualquier cambio realizado en el servidor o en cualquier cliente. Está disponible para el desarrollo de aplicaciones nativas de Android e iOS, y aplicaciones móviles multiplataforma, permite guardar objetos directamente sin necesidad de descomponerlos en tablas (Nacional et al., 2015).

Para su uso en el desarrollo de aplicaciones móviles, esta base de datos cuenta con SDKs para Android, *iOS, Xamarin, TypeScript y Javascript*. Lo anterior muestra que es posible utilizarlo para el desarrollo de aplicaciones nativas, en enfoques multiplataforma interpretado y de compilación cruzada respecto al enfoque híbrido (Tesone, 2021).

SnappyDB: es una base de datos no relacional (NoSQL) embebida para dispositivos Android, diseñada para ofrecer almacenamiento de datos con alto rendimiento. Está construida sobre *LevelDB*, (biblioteca de almacenamiento rápida desarrollada por Google), que permite manejar operaciones de lectura y escritura de forma eficiente, incluso con grandes volúmenes de datos. *SnappyDB* permite almacenar tanto tipos de datos primitivos como objetos serializados para su integración en aplicaciones móviles. Su diseño es adecuado para escenarios donde se requiere persistencia local, acceso rápido a datos y bajo consumo de recursos, lo que la convierte en una opción atractiva frente a soluciones más pesadas como SQLite, cuando la estructura relacional no es una necesidad (Tesone, 2021).

Db4o: Es un motor de base de datos orientada a objetos. Sus siglas corresponden con la expresión "*DataBase 4 (for) Objects*", desarrollado por la compañía db4objects, Inc.

Es un producto de alto rendimiento (especialmente en el modo embebido) y el modelo proporciona aplicaciones para la capa de acceso a datos, en él proporciona un cambio total en paradigma relacional de las bases de datos tradicionales (Nacional et al., 2015).

CouchBaseLite: Es un DBMS embebido, que utiliza JSON como formato de los documentos, haciendo parte de los *CouchBaseLite* posibilita la utilización del sistema *Sync Gateway* para sincronizar datos con bases de datos remotas. Se utiliza para el desarrollo de aplicaciones nativas Android e iOS, y multiplataformas en los enforques hibrido, interpretado y compilación cruzada, excelente para aplicaciones que requieren sincronización entre dispositivos, es muy robusta y muy compleja de integrar (Nacional et al., 2015).

Sistema de ficheros EXT4: Es una extensión escalable del sistema de archivos predeterminado ext3 disponible en *Red Hat Enterprise Linux 5*. Ext4 es ahora el sistema de archivos predeterminado para *Red Hat Enterprise Linux 6*, y está soportado para un tamaño máximo de sistema de archivos de 16 TB y un tamaño máximo de un archivo individual de 16TB. También retira el límite de subdirectorio de 32.000 presente en ext3.

No es una base de datos como tal, es un sistema de archivos de Linux donde los datos se almacenan directamente como ficheros y no permite realizar búsquedas ni relaciones como lo puede hacer una base de datos

6.2 Identificar las variables que pueden impactar en la velocidad de acceso y replicación.

Entre las variables consideradas se presentan variables funcionales y operacionales. Las variables funcionales son aquellas que se consideran dependientes del funcionamiento de un sistema. Son variables cuyo valor se relaciona directamente con el sistema bajo ciertas condiciones y explican o afectan el rendimiento de sistema. Mientras que las variables operacionales son aquellas que han sido definidas de forma concreta y medible dentro del trabajo de investigación, con el fin de poder observarlas, cuantificarlas o evaluarlas en un contexto específico. Además, estas variables se clasifican en: dependientes e independientes, así como variables categóricas y continuas.

6.2.1 Variables que impactan en los tiempos de inserción, búsqueda y replicación.

En la Tabla 2 se presentan las variables consideradas en el estudio, las cuales fueron seleccionadas por su relevancia en el comportamiento y desempeño de las cuatro operaciones principales analizadas.

Estas variables influyen directamente en el rendimiento de las operaciones de inserción, búsqueda, replicación de inserción y replicación de búsqueda en los sistemas de almacenamiento evaluados.

La elección de dichas variables se fundamenta en una revisión exhaustiva del estado del arte, tomando como referencia estudios previos desarrollados por autores reconocidos en la literatura científica, tales como Ji, Cheng; Chang, Li-Pin; Pan, Riwei; Wu, Chao; Gao, Congming; Shi, Liang; Kuo, Tei-Wei; Xue y Chun Jason. Estos trabajos proporcionan evidencia empírica y conceptual sobre los factores críticos que impactan el rendimiento de las bases de datos NoSQL y sistemas tradicionales, lo cual respalda su inclusión en el presente estudio.

Tabla 2:

Variables que impactan los tiempos de inserción, búsqueda y replicación

Variable	Definición	Unidad
Velocidad /Tiempo	Tiempo de procesamiento de inserción, búsqueda y replicación	milisegundos
Sistema operativo	Procesadores que ejecutan operaciones de forma más eficiente.	SDK versión 29,31,33
Memoria RAM	Es la memoria de acceso aleatorio en dispositivos móviles.	Kb, Mb o Gb
Tamaño del archivo	La cantidad de información procesada en cada operación.	Kb, Mb o Gb
Motor de almacenamiento	Bases de datos o gestor de ficheros que permiten almacenar la información.	
Numero de núcleos	Es la unidad central de procesamiento.	Unidades

Fuente: Elaboración propia.

Modelo para el análisis de impacto de variables en el tiempo de replicación.

Se plantea un modelo de regresión lineal múltiple que considere las variables predictoras ponderadas por coeficientes (pesos), con el objetivo de pronosticar el tiempo de inserción, búsqueda y replicación. El objetivo es analizar los valores y signos de los coeficientes para determinar cuánto peso tienen en la variable tiempo. La ecuación definida para el estudio considerando las variables funcionales y operacionales es:

$$t = \beta_0 + \beta_1 N + \beta_2 M + \beta_{31} BD_1 + \beta_{31} BD_1 + \beta_{32} BD_2 + \beta_{33} BD_3 + \beta_{34} BD_4 + \beta_{35} BD_5 + \beta_{41} P_1 + \beta_{42} P_2 + \beta_{43} P_3 + \beta_5 T + \gamma Z$$

Donde:

t = Tiempo de inserción (instanceTimemili) – variable dependiente – Continua

β_0 = Intercepto, Término independiente o constante (intercept)

β_1 = Coeficiente para el numero de núcleos - (nprocesadores)

β_2 = Coeficiente para la memoria RAM - (ramdevicesize)

β_{31} *a* β_{35} = Coeficiente base de datos

β_{41} a β_{43} = Coeficiente para la versión del dispositivo (sdkversion)

β_5 = Coeficiente para el tamaño del archivo (insertsizekilobyte)

BD_1= Base de datos *CouchBaseLite* – Variable indicadora categórica

BD_2= Base de datos *Db4o* – Variable indicadora categórica

BD_3= Base de datos *ObjecBox* – Variable indicadora categórica

BD_4= Base de datos *Realm* – Variable indicadora categórica

BD_5= Base de datos *Snappy* – Variable indicadora categórica

N = Número de núcleos – Variable independiente numérica discreta

M = Tamaño de la memoria RAM – Variable independiente numérica

P_1 a P_3 = Versión del dispositivo – Variable categórica adicionales

T = Tamaño del archivo – Variable numérica

Z: Identificador de normalización (tipo de dispositivo, experimento o grupo).

γ: Coeficiente asociado al identificador de normalización.

El manejo de variables categóricas en la ecuación para el análisis en la regresión, es el método de codificación *one hot code* que convierte cada categoría en una nueva variable binaria (0 o 1).

6.3 Desarrollar una aplicación que permita comparar el tiempo de procesamiento de un sistema de ficheros tradicional contra el de base de datos NoSQL bajo diferentes configuraciones de hardware y software.

6.3.1 Entorno de desarrollo del proyecto.

Para el desarrollo de este trabajo se utilizó el entorno de desarrollo integrado (IDE) Android *Studio*, versión *Ladybug*, el cual proporciona una plataforma robusta y especializada para la creación de aplicaciones móviles basadas en el sistema operativo Android. Esta versión del IDE incluye soporte completo para desarrollo en Java, así como herramientas integradas de depuración, emulación de dispositivos y monitoreo de rendimiento, lo que facilitó una implementación más eficiente y controlada del sistema propuesto.

En este entorno se diseñó e implementó una aplicación móvil nativa en Java, cuyo propósito principal fue realizar pruebas comparativas entre dos enfoques de almacenamiento de datos: un sistema de ficheros tradicional y una base de datos NoSQL embebida. La aplicación fue estructurada para ejecutar procesos automatizados de inserción de datos bajo distintas condiciones controladas, con el

fin de medir y registrar tiempos de procesamiento, consumo de recursos y eficiencia en la gestión de datos.

En la **Tabla 3** se presenta una vista general del entorno integrado de desarrollo utilizado, donde se puede observar la configuración del proyecto, la estructura del código fuente, así como los recursos gráficos y de interfaz de usuario (UI) empleados durante el desarrollo. Esta configuración permitió evaluar el desempeño del sistema en un entorno reproducible y escalable, adecuado para fines de análisis y validación comparativa.

Tabla 3

Entorno de desarrollo integrado donde se configuró el proyecto

Android Studio Narwhal | 2025.1.1 Patch 1
Build #AI-251.25410.109.2511.13752376, built on July 8, 2025
Runtime version: 21.0.6+-13391695-b895.109 amd64
VM: OpenJDK 64-Bit Server VM by JetBrains s.r.o.
Toolkit: sun.awt.windows.WToolkit
Windows 11.0
GC: G1 Young Generation, G1 Concurrent GC, G1 Old Generation
Memory: 2048M
Cores: 20
Registry:
ide.experimental.ui=true
com.android.studio.ml.activeModel=com.android.studio.ml.AidaModel
Non-Bundled Plugins:
Dart (251.27623.5)
Lombook Plugin (251.25410.59)
plantuml-parser (0.0.9)
com.github.copilot (1.5.49-243)
io.flutter (86.0.2)
com.kn.diagrams.generator.generator (2022.2.0)

Fuente: Elaboración propia

6.3.2 Estructura de clases del proyecto.

En la **Figura 4** se muestra la estructura general de clases que conforman el código fuente del trabajo. Esta estructura refleja la organización modular del sistema, diseñada bajo principios de reutilización, mantenibilidad y separación de responsabilidades.

Una de las clases principales es *MainActivity*, que actúa como punto de entrada y controlador principal de la interfaz de usuario (UI). Esta clase permite al usuario configurar los parámetros de ejecución de las pruebas, incluyendo:

- La cantidad de datos que se desean insertar en cada ejecución.

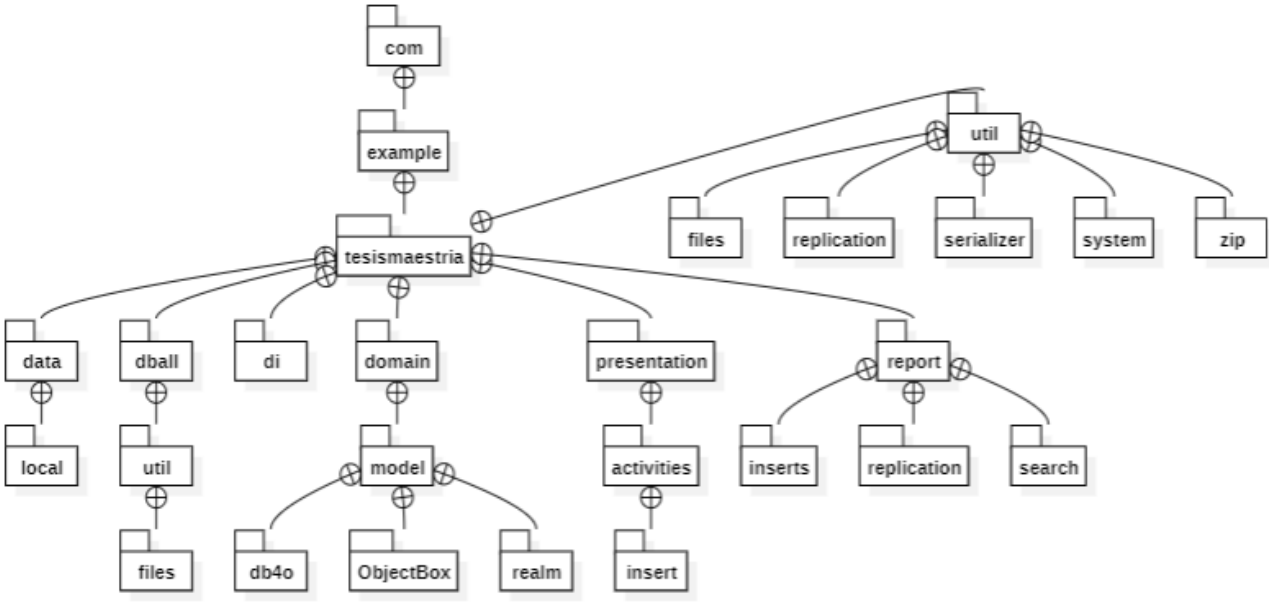
- El número de iteraciones o repeticiones del proceso de inserción, con el fin de garantizar resultados estadísticamente representativos.

A partir de esta configuración, la aplicación ejecuta pruebas de inserción sobre diferentes motores de bases de datos NoSQL embebidas, específicamente: *CouchBaseLite, Db4o, ObjectBox, Realm y SnappyDB*. Cada una de estas tecnologías fue encapsulada en clases independientes, siguiendo una arquitectura orientada a interfaces, lo que permite desacoplar la lógica de inserción del motor de persistencia subyacente.

Este diseño facilita tanto la extensión futura del sistema como la comparación sistemática del rendimiento entre las soluciones evaluadas, bajo condiciones controladas y homogéneas.

Figura 4

Estructura general de clases que conforman el código fuente del trabajo



Fuente: Elaboración Propia

6.3.3 Dispositivos de prueba emulados.

Para la validación funcional y la evaluación del rendimiento de la aplicación desarrollada, se utilizaron dispositivos emulados que representan distintos entornos de ejecución del sistema operativo Android. En particular, se configuraron y ejecutaron emuladores con versiones de Android 10 (SDK 29), Android 12 (SDK31) y Android 13 (SDK33), con el objetivo de verificar la compatibilidad de la

aplicación y observar posibles variaciones en el comportamiento del sistema bajo diferentes versiones del *framework* Android.

Estos emuladores fueron creados y gestionados mediante el *Android Virtual Device* (AVD) *Manager*, herramienta integrada en Android Studio, la cual permite simular múltiples configuraciones de hardware y software, incluyendo arquitectura de CPU, tamaño de memoria, resolución de pantalla, y versión de SDK.

En la **Tabla 4** se presenta el listado de dispositivos virtuales utilizados durante las pruebas. Cada emulador fue configurado para mantener condiciones homogéneas en cuanto a recursos del sistema (RAM, núcleos de CPU y almacenamiento), permitiendo así una evaluación comparativa más precisa del desempeño de las distintas bases de datos NoSQL integradas en la aplicación.

Tabla 4:

Listado de dispositivos virtuales

Device Manager		
Name	API	*Type*
Medium Phone API 33 *Android 13.0 ("Tiramisu") | x86_64*	33	*Virtual*
Medium Phone API 29 *Android 10.0 ("Q") | x86_64*	29	*Virtual*
Medium Phone API 31 *Android 12.0 ("S") | x86_64*	31	*Virtual*

Fuentes: Elaboración propia

6.3.4 Configuración del emulador.

El emulador Android fue configurado con parámetros específicos que permiten simular de forma precisa el comportamiento de un dispositivo físico, garantizando condiciones controladas y reproducibles para la ejecución de pruebas. La configuración incluye la selección de una versión particular del sistema operativo Android, junto con características de hardware virtuales como el tamaño de la memoria RAM, número de núcleos de CPU, tipo de arquitectura, resolución de pantalla, y espacio de almacenamiento interno.

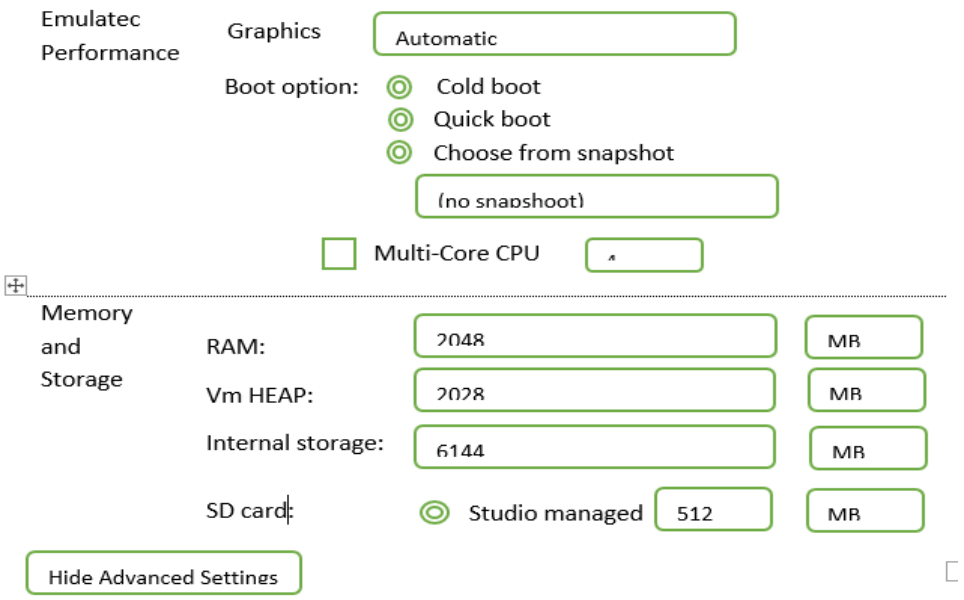
La **Tabla 5** muestra los detalles completos de esta configuración tal como aparece en el panel del Android *Virtual Device* (AVD) *Manager*, dentro de Android Studio. Esta configuración fue utilizada de manera consistente durante la ejecución de las pruebas para evaluar el rendimiento de los distintos motores de bases de datos embebidas integrados en la aplicación móvil.

Tabla 5:

Configuración del panel Android Virtual Device (AVD) Manager

Android Virtual Device (AVD)

Verify Configuration



Fuente: Elaboración propia

6.3.5 Caracterización del hardware con CPU - Z.

Para obtener información detallada sobre las especificaciones de hardware de los dispositivos móviles utilizados en las pruebas, se empleó la aplicación CPU-Z para Android, adicionalmente se usaron funciones de Java nativas para extraer la información técnica de los dispositivos y añadirla a los informes. Esta herramienta permite realizar una identificación precisa de los componentes internos del dispositivo, proporcionando datos esenciales para el análisis del rendimiento del sistema.

Entre las especificaciones extraídas mediante CPU-Z se incluyen:

- Número de núcleos del procesador.
- Arquitectura y modelo del procesador.
- Tamaño total de la memoria RAM instalada.
- Capacidad total del almacenamiento interno del dispositivo.

Estos parámetros son fundamentales para contextualizar los resultados de las pruebas de rendimiento, ya que influyen directamente en el comportamiento de las operaciones de inserción, lectura y escritura en las distintas bases de datos NoSQL evaluadas, como se muestra en la **Figura 5A y la Figura 5A.**

Figura 5A

Parámetros fundamentales de prueba

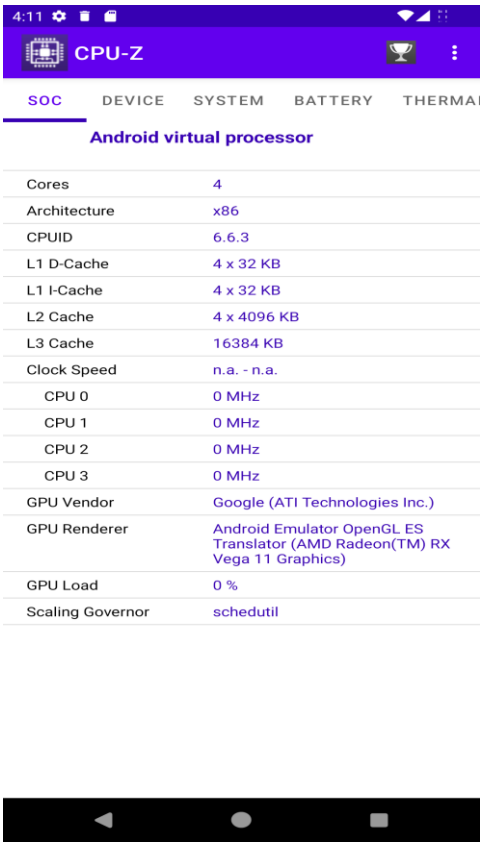
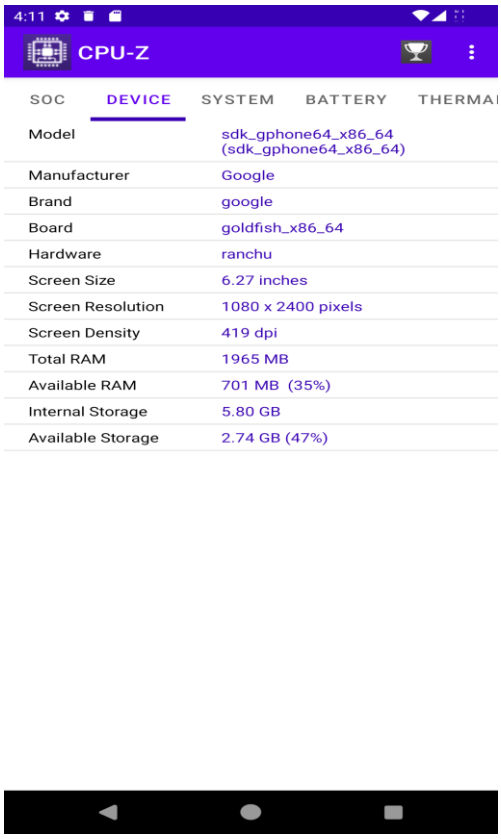


Figura 5B

Parámetros fundamentales de prueba



Fuente: Elaboración propia

6.3.6 Software para la aplicación de los dispositivos móviles.

En el proceso de desarrollo, se utilizó el lenguaje de programación Java, así como el entorno de desarrollo integrado (IDE) Android Studio. También el uso de *frameworks* multiplataforma como *Flutter o React Native*, que permiten desarrollar una única base de código para diferentes sistemas operativos.

6.3.7 Resultados obtenidos en la aplicación del software en los sistemas de almacenamiento de datos seleccionados.

En la implementación del software de aplicación realizado para este trabajo en el tiempo de inserción, búsqueda y replicación de los datos en los dispositivos móviles, se obtuvieron resultados para cada tipo de variable dada en la ecuación (1), diferentes sistemas de almacenamiento, memoria RAM, tamaño de inserción, número de procesadores, bases de datos, método y versión del SDK.

Los datos obtenidos para el tiempo de inserción, búsqueda y la replicación se analizaron mediante el coeficiente de correlación de Spearman y la regresión lineal múltiple como medidas estadísticas, además, de su comportamiento gráfico, implementados en el script de Python. El análisis estadístico de algunas variables relacionadas con la regresión múltiple y el coeficiente de Spearman. El análisis de los coeficientes de correlación de Spearman, indica la fuerza o importancia y dirección de la relación entre las variables independientes y la variable dependiente en cada caso de la medición. El análisis de regresión múltiple evalúa el impacto de las características o variables en la variable dependiente tiempo de inserción, búsqueda y replicación, que puede estar relacionada directa e indirectamente con el rendimiento de un sistema. En él se incluyen variables continuas (como tamaño de inserción, numero de procesadores, memoria RAM) y categóricas (como base de datos, método, versión del SDK) entre otras.

6.3.7.1 Análisis de la información obtenida en la implementación del software en la inserción.

La **Tabla 6** muestra un ejemplo de los resultados (datos) obtenidos en la inserción. Se omite la tabla completa por temas de espacio.

Tabla 6:

Resultados obtenidos en la implementación del software en la inserción

databasename	method	insertmiliTime	insertsegTime	insertminTime	insertsizekilobyte	datasize	repeticion	nprocesadores	ramdevicesize	sdkversion
CouchBaseLite	DATABASE	4145	4	0	45002	50	1	2	2	29
CouchBaseLite	DATABASE	9470	9	0	90005	100	1	2	2	29
CouchBaseLite	DATABASE	15245	15	0	135007	150	1	2	2	29
CouchBaseLite	DATABASE	18613	18	0	180010	200	1	2	2	29
CouchBaseLite	DATABASE	21267	21	0	225013	250	1	2	2	29
CouchBaseLite	DATABASE	25608	25	0	270015	300	1	2	2	29
Db4o	DATABASE	6291	6	0	45002	50	1	2	2	29
Db4o	DATABASE	9402	9	0	90005	100	1	2	2	29
Db4o	DATABASE	10905	10	0	135007	150	1	2	2	29
Db4o	DATABASE	16084	16	0	180010	200	1	2	2	29
Db4o	DATABASE	23562	23	0	225013	250	1	2	2	29
Db4o	DATABASE	23532	23	0	270015	300	1	2	2	29
FILES	FILES	42854	42	0	45002	50	1	2	2	29
FILES	FILES	84261	84	1	90005	100	1	2	2	29
FILES	FILES	126911	126	2	135007	150	1	2	2	29
FILES	FILES	133884	133	2	180010	200	1	2	2	29
FILES	FILES	79693	79	1	225013	250	1	2	2	29
FILES	FILES	59834	59	0	270015	300	1	2	2	29
ObjectBox	DATABASE	3698	3	0	45002	50	1	2	2	29
ObjectBox	DATABASE	7785	7	0	90005	100	1	2	2	29
ObjectBox	DATABASE	10210	10	0	135007	150	1	2	2	29
ObjectBox	DATABASE	15173	15	0	180010	200	1	2	2	29
ObjectBox	DATABASE	19593	19	0	225013	250	1	2	2	29
ObjectBox	DATABASE	23584	23	0	270015	300	1	2	2	29
Realm	DATABASE	3444	3	0	45002	50	1	2	2	29
Realm	DATABASE	6660	6	0	90005	100	1	2	2	29
Realm	DATABASE	9318	9	0	135007	150	1	2	2	29
Realm	DATABASE	14509	14	0	180010	200	1	2	2	29
Realm	DATABASE	17260	17	0	225013	250	1	2	2	29
Realm	DATABASE	19684	19	0	270015	300	1	2	2	29
Snappy	DATABASE	3321	3	0	45002	50	1	2	2	29
Snappy	DATABASE	15805	15	0	90005	100	1	2	2	29
Snappy	DATABASE	21783	21	0	135007	150	1	2	2	29

Snappy	DATABASE	37107	37	0	180010	200	1	2	2	29
Snappy	DATABASE	52645	52	0	225013	250	1	2	2	29
Snappy	DATABASE	75119	75	1	270015	300	1	2	2	29

Fuente: Elaboración propia

6.3.7.1.2 Análisis de la correlación en el tiempo de inserción de los datos.

La **Tabla 7** presenta para cada variable en estudio la correlación con su respectiva interpretación y análisis, entre el tiempo de inserción (insertmiliTime) y las distintas características que intervinieron en el proceso.

Tabla 7:

Coeficiente de correlación e interpretación en el tiempo de inserción para cada variable

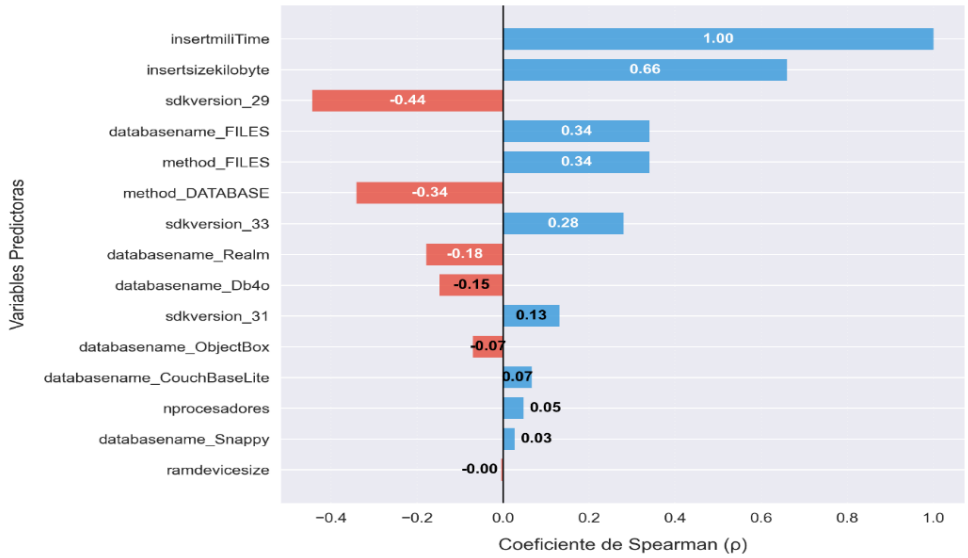
Feature	**Correlación Spearman**	**Interpretación**
insertmiliTime	1.000	Es la variable objetivo, correlación perfecta consigo misma.
insertsizekilobyte	0.660	Correlación positiva alta: mientras los datos a insertar aumentan o sea mayor, el tiempo de inserción va a aumentar.
sdkversion_33	0.280	Correlación positiva débil: usar la versión SDK 33 tiende a aumentar muy poco el tiempo de inserción.
method_FILES	0.340	Correlación positiva moderada: Uso de método o base de datos tipo "FILES" se asocia a mayor tiempo de inserción.
databasename_FILES	0.340	Correlación positiva moderada: el uso del método o base de datos tipo FILES se asocia a mayor tiempo de inserción.
databasename_ObjectBox	-0.070	Correlación negativa débil: poco efecto en la disminución del tiempo de inserción.
sdkversion_31	0.131	Correlación positiva débil: poco efecto en la disminución del tiempo de inserción.
databasename_Snappy	-0.027	Correlación negativa casi nula: muy poco impacto negativo en tiempo de inserción.
Databasename_Realm	-0.178	Correlación negativa moderada; muy poco impacto negativo en el tiempo de inserción.
ramdevicesize	-0.004	La RAM del dispositivo no muestra impacto relevante.
databasename_Db4o	-0.148	Correlación negativa moderada. asociadas a menor tiempo de inserción.
Databasename_CouchBaseLite	0.067	Correlación casi nula. No influye en el tiempo de inserción.
nprocesadores	0.048	El número de núcleos casi no afecta el tiempo (mejor rendimiento).
method_DATABASE	-0.340	Usar el método "DATABASE" en vez de "FILES" reduce el tiempo de inserción.
sdkversion_29	-0.443	Usar la versión 29 del SDK está asociada a un menor tiempo de inserción.

Fuente: Elaboración propia

El **Gráfico 1** muestra el comportamiento de cada variable mediante el coeficiente de correlación entre las diversas características de las variables y el tiempo de inserción (insertmiliTime).

Gráfico 1

Comportamiento del tiempo con el coeficiente de correlación en el tiempo de inserción



Fuente: Elaboración propia

Análisis de la correlación en los tiempos de inserción.

A partir del análisis del coeficiente de correlación, se identificaron variables que impactan significativamente en el tiempo de inserción. Las características o variables que tienden a incrementar dicho tiempo son: el tamaño de inserción (insertsizekilobyte), con una correlación alta de 0.660, la versión de SDK 33 (sdkversion_33) con una correlación débil de 0.280 y el uso del método FILES (method_FILES) con correlación débil de 0.340. Estas variables presentan una correlación positiva con mayores tiempos de operación.

Por otro lado, se identificaron variables que reducen los tiempos de inserción, entre las cuales destacan: la versión de SDK 29 (sdkversion_29) con una correlación 0.443, el uso del método DATABASE (method_DATABASE) con correlación de 0.340 y un mayor número de procesadores (nprocesadores) con correlación de 0.048. Dentro de las bases de datos analizadas, *Realm* con correlación de -178 y especialmente *Db4o (databasename_Db4o)* con correlación -0.148, mostraron mejoras notables en los tiempos de inserción. Estas variables presentan una correlación negativa con mayores tiempos de operación.

En cuanto a las variables con bajo impacto, correlación débil o nula, se encuentran bases de datos como *ObjectBox, Snappy, y CouchBaseLite,* así como las versiones de SDK 31 y la versión SDK 33.

Estas variables presentan correlaciones muy bajas, cercanas a cero, por lo que su influencia sobre el tiempo de inserción es mínima o nula.

6.3.7.1.3 Análisis de la regresión múltiple en el tiempo de inserción.

La **Tabla 8** presenta la interpretación y análisis de la regresión múltiple para el tiempo de inserción (insertmiliTime), considerando el intercepto y los coeficientes de las diferentes características o variables.

Tabla 8:

Análisis e interpretación de la regresión múltiple en el tiempo de inserción

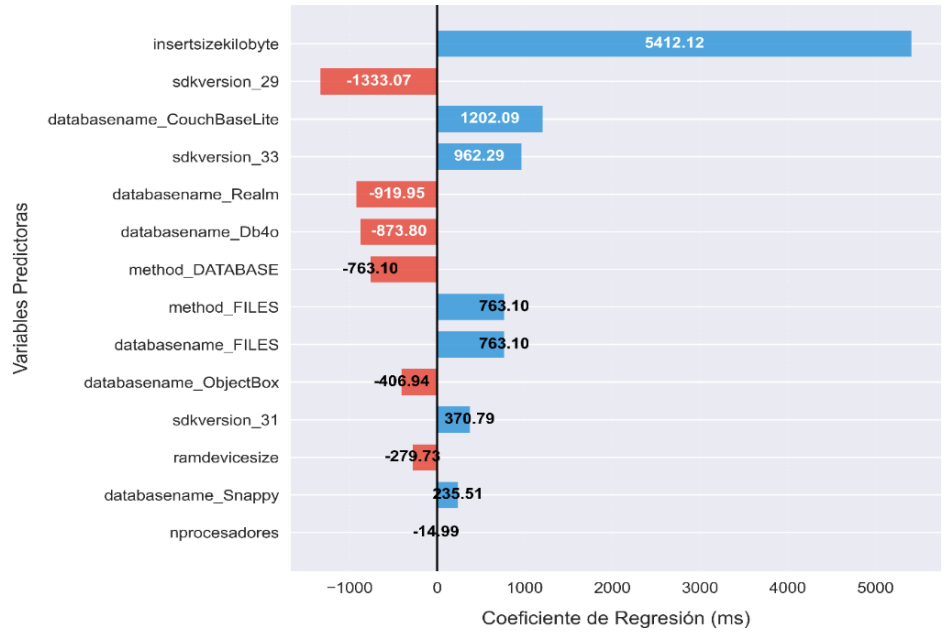
Feature	**Coeficiente**	**Interpretación**
Insertsizekilobyte	5412.12	Impacto significativo, a mayor tamaño del archivo mayor el tiempo a estimar. Aumenta en 5412.12 por cada KB.
Nprocesadores	-14.993	A mayor número de procesadores menor el valor del tiempo a estimar. Disminuye en 14.993 por cada procesador.
Ramdevicesize	-279.726	A mayor número de RAM menor el valor del tiempo a estimar. Disminuye 279.726 por unidad. Mejora el rendimiento.
databasename_FILES	763.1	Es la base de referencia implícita por su valor positivo similar al del método FILES. Es para comparar las bases de datos.
databasename_CouchBaseLite	1202.086	Aumenta el tiempo de inserción. Peor rendimiento que la referencia.
databasename_Db4o	-873.801	Disminuye el tiempo de inserción vs FILES. Mejor rendimiento.
databasename_ObjectBox	-406.944	disminuye el tiempo de inserción. Mejor que FILES.
databasename_Realm	-919.946	Disminuye el tiempo de inserción. Mejor que FILES.
databasename_Snappy	235.506	Aumenta el tiempo de inserción. Peor que FILES.
method_DATABASE	-763.1	Disminuye el tiempo de inserción. Similar a method_FILES.
method_FILES	763.1	Aumenta el tiempo de inserción. Similar a method_DATABASE.
sdkversion_29	-1333.075	Disminuye el tiempo de inserción.
sdkversion_31	370.789	Aumenta el tiempo de inserción.
sdkversion_33	962.286	Aumenta el tiempo de inserción. Peor que las versiones 29 y 31
Intercept	1010.289	Valor del tiempo cuando todas las variables son cero.

Fuente: Elaboración propia

En el **Gráfico 2** se muestra el comportamiento de los coeficientes en la regresión múltiple entre las diversas variables y el tiempo de inserción (insertmiliTime).

Gráfico 2

Análisis del comportamiento de los coeficientes en el tiempo de inserción



Fuente: Elaboración propia

Interpretación y análisis de la regresión múltiple en tiempos de inserción.

El tamaño de los *Insertsizekilobyte* es el factor con mayor impacto positivo sobre el tiempo de inserción, incrementándolo a medida que crece. En contraste, un mayor número de procesadores (Nprocesadores) con coeficiente -14.993 y una mayor cantidad de memoria RAM (Ramdevicesize) con un coeficiente de -279.726 contribuyen a reducir significativamente el tiempo de inserción, favoreciendo un mejor desempeño del sistema (considerando que un menor tiempo es deseable en este contexto).

La configuración basada en FILES con coeficiente -763.1 y method_FILES con coeficiente 763.1 representa la referencia base del sistema, dada su equivalencia en valores de configuración. En cuanto a las bases de datos evaluadas, *Db40* con coeficiente 873.801, *ObjectBox* con coeficiente 406.944 y *Realm* con coeficiente 919.947 presentan mejoras en comparación con la configuración FILES, mientras que *Snappy* con coeficiente 235.506 y *CouchBaseLite* con coeficiente 1202.086 muestran un desempeño inferior, aumentando los tiempos de inserción.

Respecto a las versiones de SDK, SDK 33 con coeficiente 962.286 exhibe un rendimiento superior frente a versiones anteriores como SDK 29 con coeficiente -1333.075 y SDK 31 con coeficiente 370.789, siempre bajo la interpretación de que un valor más alto corresponde a un mejor desempeño. Además, las combinaciones databasename_FILES y method_FILES mantienen coherencia,

reflejándose en correlaciones opuestas de 763.1 y -763.1 respectivamente, lo que sugiere estabilidad en la relación entre método y tipo de base de datos.

Finalmente, bases de datos como *Db40*, *ObjectBox* y *Realm* y versiones de SDK 29, muestran una influencia relativamente significativa en la disminución del tiempo de inserción.

6.3.7.2 Análisis de la información obtenida en la implementación del software en el tiempo de búsqueda.

En la **Tabla 9** se muestra una parte de los resultados para cada tipo de variable dada en la ecuación para los diferentes sistemas de almacenamiento, debido a la gran magnitud de los datos obtenidos.

Tabla 9:

Datos obtenidos en la implementación del software en el tiempo de búsqueda

method	databasename	filename	searchmiliTime	searchsegTime	searchminTime	datasize	repeticion	nprocesadores	ramdevicesize	sdkversion
DATABASE	CouchBaseLite	bike_241.bmp	167	0	0	921654	0	2	2	29
DATABASE	CouchBaseLite	bike_282.bmp	3	0	0	921654	0	2	2	29
DATABASE	CouchBaseLite	bike_160.bmp	12	0	0	921654	0	2	2	29
DATABASE	CouchBaseLite	bike_050.bmp	3	0	0	921654	0	2	2	29
DATABASE	CouchBaseLite	bike_075.bmp	4	0	0	921654	0	2	2	29
DATABASE	CouchBaseLite	bike_009.bmp	4	0	0	921654	0	2	2	29
DATABASE	CouchBaseLite	bike_080.bmp	4	0	0	921654	0	2	2	29
DATABASE	CouchBaseLite	bike_094.bmp	4	0	0	921654	0	2	2	29
DATABASE	CouchBaseLite	bike_016.bmp	3	0	0	921654	0	2	2	29
DATABASE	CouchBaseLite	bike_028.bmp	5	0	0	921654	0	2	2	29
DATABASE	CouchBaseLite	bike_241.bmp	160	0	0	921654	1	2	2	29
DATABASE	CouchBaseLite	bike_282.bmp	4	0	0	921654	1	2	2	29
DATABASE	CouchBaseLite	bike_160.bmp	4	0	0	921654	1	2	2	29
DATABASE	CouchBaseLite	bike_050.bmp	4	0	0	921654	1	2	2	29
DATABASE	CouchBaseLite	bike_075.bmp	5	0	0	921654	1	2	2	29
DATABASE	CouchBaseLite	bike_009.bmp	5	0	0	921654	1	2	2	29
DATABASE	CouchBaseLite	bike_080.bmp	3	0	0	921654	1	2	2	29
DATABASE	CouchBaseLite	bike_094.bmp	3	0	0	921654	1	2	2	29
DATABASE	CouchBaseLite	bike_016.bmp	5	0	0	921654	1	2	2	29
DATABASE	CouchBaseLite	bike_028.bmp	3	0	0	921654	1	2	2	29
DATABASE	CouchBaseLite	bike_241.bmp	3	0	0	921654	2	2	2	29
DATABASE	CouchBaseLite	bike_282.bmp	5	0	0	921654	2	2	2	29
DATABASE	CouchBaseLite	bike_160.bmp	4	0	0	921654	2	2	2	29
DATABASE	CouchBaseLite	bike_050.bmp	2	0	0	921654	2	2	2	29
DATABASE	CouchBaseLite	bike_075.bmp	3	0	0	921654	2	2	2	29
DATABASE	CouchBaseLite	bike_009.bmp	4	0	0	921654	2	2	2	29
DATABASE	CouchBaseLite	bike_080.bmp	5	0	0	921654	2	2	2	29
DATABASE	CouchBaseLite	bike_094.bmp	4	0	0	921654	2	2	2	29
DATABASE	CouchBaseLite	bike_016.bmp	4	0	0	921654	2	2	2	29
DATABASE	CouchBaseLite	bike_028.bmp	2	0	0	921654	2	2	2	29
DATABASE	CouchBaseLite	bike_241.bmp	3	0	0	921654	3	2	2	29
DATABASE	CouchBaseLite	bike_282.bmp	2	0	0	921654	3	2	2	29
DATABASE	CouchBaseLite	bike_160.bmp	4	0	0	921654	3	2	2	29
DATABASE	CouchBaseLite	bike_050.bmp	4	0	0	921654	3	2	2	29

DATABASE	CouchBaseLite	bike_075.bmp	3	0	0	921654	3	2	2	29
DATABASE	CouchBaseLite	bike_009.bmp	5	0	0	921654	3	2	2	29

Fuente: Elaboración propia

6.3.7.2.1 Análisis de los resultados en el tiempo de búsqueda.

En la **Tabla 10** se presentan los resultados obtenidos para cada variable con su respectiva interpretación y análisis, entre el tiempo de búsqueda (searchmiliTime) y las distintas características que intervinieron en el proceso.

Tabla 10:

Análisis e interpretación de la correlación en el tiempo de búsqueda

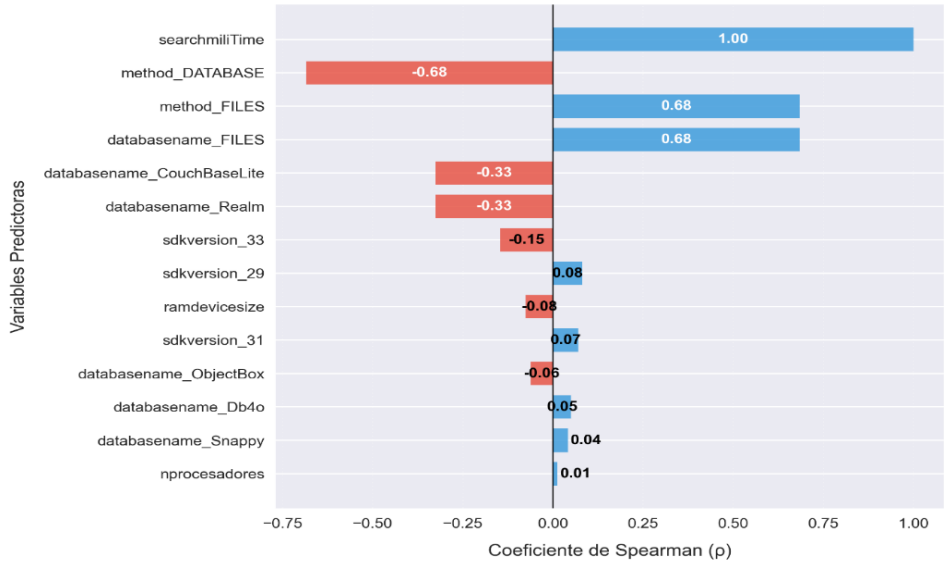
Feature	**Correlación Spearman**	**Interpretación**
searchmiliTime	1.000	Es la variable objetivo, correlación perfecta consigo misma.
databasename_FILES	0.703	Uso de método o base de datos tipo "FILES" se asocia a mayor tiempo de búsqueda.
method_FILES	0.703	Uso de método o base de datos tipo "FILES" se asocia a mayor tiempo de búsqueda.
databasename_Db4o	0.048	Correlación muy baja. Incremento casi nulo en el tiempo de búsqueda.
sdkversion_31	0.045	Correlación baja casi nula; poco efecto en el tiempo de búsqueda.
sdkversion_33	-0.164	Usar la versión 33 del SDK está asociada a un menor tiempo de búsqueda
nprocesadores	0.007	Correlación muy baja casi nula. No presenta ningún efecto sobre el tiempo de búsqueda.
ramdevicesize	-0.076	Dispositivos con más RAM tienden a tener tiempos de búsqueda levemente menores.
sdkversion_29	0.125	Usar la versión 29 del SDK está asociada a un mayor tiempo de búsqueda.
databasename_Snappy	0.025	Correlación positiva muy baja. Muy poco impacto en el tiempo de búsqueda.
Databasename_CouchBaseLite	-0.331	Bases de datos asociadas a menor tiempo de búsqueda.
databasename_ObjectBox	-0.076	Correlación negativa baja; con efecto mínimo en el tiempo de búsqueda.
Databasename_Realm	-0.331	Ligeramente negativo; con poco impacto el tiempo de búsqueda.
method_DATABASE	-0.703	Usar el método "DATABASE" en vez de "FILES" reduce el tiempo de inserción.

Fuente: Elaboración propia

En el **Gráfico 3** se muestra la correlación entre las diversas características de las variables y el tiempo de búsqueda (searchmiliTime), utilizando la correlación de Spearman como una medida estadística.

Gráfico 3

Correlación de las características en el tiempo de búsqueda



Fuente: Elaboración propia

Análisis del coeficiente de correlación en el tiempo de búsqueda.

El análisis de la correlación de Spearman revela que las variables databasename_FILES y method_FILES presentan una alta correlación positiva con el tiempo de búsqueda 0.703 para ambas correlaciones. Esto indica que el uso del método FILES está asociado a un incremento significativo en los tiempos de búsqueda.

En contraste, el method_DATABASE con correlación -0.703 y las bases de datos *Realm, ObjectBox y CouchBaseLite* con correlaciones -0.331, -0.076 y -0.331 respectivamente, muestran correlaciones negativas con el tiempo de búsqueda. Esta relación sugiere que la utilización de estas tecnologías tiende a disminuir los tiempos de búsqueda, favoreciendo un mejor desempeño en este aspecto.

La base de datos *Db4o* con coeficiente 0.048 se posiciona de manera intermedia, con una correlación positiva moderada, lo que implica un impacto menos pronunciado, pero todavía orientado a un aumento en los tiempos de búsqueda en comparación con las bases que presentan correlaciones negativas.

Por otro lado, las versiones de SDK 29 y 31 evaluadas, así como las características de hardware del dispositivo (cantidad de RAM y número de procesadores), presentan correlaciones positivas que

inciden en el aumento del tiempo de búsqueda muy bajo. Esto sugiere que su influencia sobre el tiempo de búsqueda es mínima o inconsistente, sin un patrón claro de afectación significativa.

6.3.7.2.2 Análisis de los coeficientes de la regresión múltiple en el tiempo de búsqueda.

La **Tabla 11** presenta la interpretación y análisis de los coeficientes de la regresión múltiple para el tiempo de búsqueda (searchmiliTime), considerando el intercepto y los coeficientes de las diferentes variables.

Tabla 11:

Análisis de la regresión múltiple en el tiempo de búsqueda

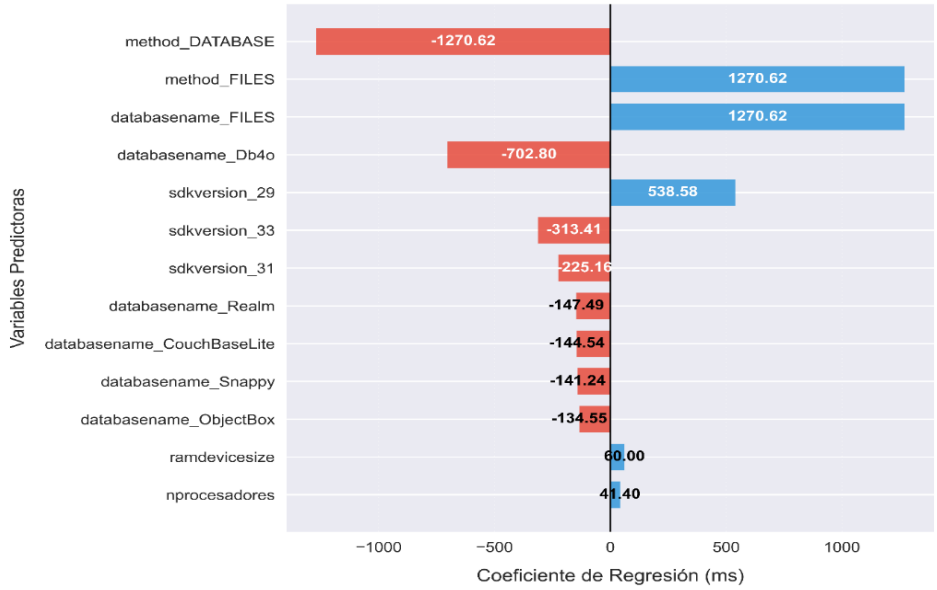
Feature	**Coeficiente**	**Interpretación**
nprocesadores	1.180	A mayor número de procesadores mayor el valor del tiempo a estimar.
ramdevicesize	106.477	A mayor número de RAM mayor el valor del tiempo a estimar.
databasename_FILES	2237.669	Es la base de referencia implícita por su valor positivo similar al del método FILES. Incrementa mucho el tiempo.
databasename_CouchBaseLite	-45.332	Disminuye el tiempo de búsqueda.
databasename_Db4o	-2021.334	Disminuye el tiempo de búsqueda en mayor proporción.
databasename_ObjectBox	-42.236	Disminuye el tiempo de búsqueda.
databasename_Realm	-59.781	Disminuye el tiempo de búsqueda.
databasename_Snappy	-68.976	Disminuye el tiempo de búsqueda.
method_DATABASE	-2237.669	Disminuye fuertemente el tiempo de búsqueda.
method_FILES	2237.669	Aumenta el tiempo de búsqueda. Este método tiene gran impacto negativo en el rendimiento.
sdkversion_29	1893.988	Incrementa el tiempo de búsqueda significativamente.
sdkversion_31	-937.894	Reduce bastante el tiempo de búsqueda.
sdkversion_33	-956.094	Reduce bastante el tiempo de búsqueda.
Intercept	2375.775	Valor del tiempo cuando todas las variables son cero.

Fuente: Elaboración propia

En el **Gráfico 4** se muestra el comportamiento de los coeficientes en la regresión múltiple entre las diversas variables y el tiempo de búsqueda (searchmiliTime).

Gráfico 4

Análisis del comportamiento de los coeficientes en el tiempo de búsqueda



Fuente: Elaboración propia

Análisis de la regresión múltiple sobre el tiempo de búsqueda.

En el modelo de regresión múltiple, cada coeficiente estimado representa el cambio esperado en el tiempo de búsqueda ante una variación de una unidad en la variable correspondiente, manteniendo constantes las demás variables.

Los resultados muestran que method_FILES y databasename_FILES con coeficiente 2237.334 cada uno presentan los coeficientes positivos más altos, indicando que su utilización se asocia con un incremento significativo en el tiempo de búsqueda. Por el contrario, *method_DATABASE* exhibe un coeficiente igual (-2237.334) pero negativo de gran magnitud, lo que sugiere una reducción importante en el tiempo de búsqueda cuando se emplea este método.

Asimismo, las bases de datos *Realm, ObjectBox, CouchBaseLite, Snappy y Db4o* (correlaciones -59.781, -42.236, -45.342, -68.976 y -2021.334 respectivamente), se correlacionan con disminuciones en el tiempo de búsqueda, reflejadas en los coeficientes negativos en el modelo.

En cuanto a las versiones del SDK, las versiones más recientes SDK 31 (-937.894) y SDK 33 -956.094) también presentan efectos reductores significativos sobre el tiempo de búsqueda, por su coeficiente negativo, mientras que la versión SDK 29 (1893.988) se asocia a un incremento del mismo por su coeficiente positivo.

De forma inesperada, tanto el aumento en el número de procesadores como en la cantidad de memoria RAM (1.180 y 106.477 respectivamente), muestran coeficientes positivos, indicando un incremento en el tiempo de búsqueda. Este comportamiento podría estar relacionado con una gestión ineficiente de recursos en el sistema o con un paralelismo no optimizado que genera sobrecarga en lugar de mejorar el desempeño.

6.3.7.3 Análisis de la información obtenida en la replicación inserción.

En la **Tabla 12** se muestra una parte de la información obtenida en la replicación inserción, por la gran magnitud de los datos obtenidos.

Tabla 12:

Datos obtenidos en la implementación del software en la replicación inserción

method	databasename	filename	replicationmiliTime	replicationsegTime	replicationminTime	datasize	repeticion	nprocesadores	ramdevicesize	sdkversion
Send	CouchBaseLite	bike_009.bmp	612	0	0	921654	0	2	2	29
Send	CouchBaseLite	bike_016.bmp	699	0	0	921654	0	2	2	29
Send	CouchBaseLite	bike_028.bmp	436	0	0	921654	0	2	2	29
Send	CouchBaseLite	bike_050.bmp	346	0	0	921654	0	2	2	29
Send	CouchBaseLite	bike_075.bmp	282	0	0	921654	0	2	2	29
Send	CouchBaseLite	bike_080.bmp	390	0	0	921654	0	2	2	29
Send	CouchBaseLite	bike_094.bmp	395	0	0	921654	0	2	2	29
Send	CouchBaseLite	bike_160.bmp	399	0	0	921654	0	2	2	29
Send	CouchBaseLite	bike_241.bmp	433	0	0	921654	0	2	2	29
Send	CouchBaseLite	bike_282.bmp	540	0	0	921654	0	2	2	29
Send	CouchBaseLite	bike_009.bmp	228	0	0	921654	1	2	2	29
Send	CouchBaseLite	bike_016.bmp	327	0	0	921654	1	2	2	29
Send	CouchBaseLite	bike_028.bmp	297	0	0	921654	1	2	2	29
Send	CouchBaseLite	bike_050.bmp	243	0	0	921654	1	2	2	29
Send	CouchBaseLite	bike_075.bmp	261	0	0	921654	1	2	2	29
Send	CouchBaseLite	bike_080.bmp	236	0	0	921654	1	2	2	29
Send	CouchBaseLite	bike_094.bmp	232	0	0	921654	1	2	2	29
Send	CouchBaseLite	bike_160.bmp	485	0	0	921654	1	2	2	29
Send	CouchBaseLite	bike_241.bmp	652	0	0	921654	1	2	2	29
Send	CouchBaseLite	bike_282.bmp	418	0	0	921654	1	2	2	29
Send	CouchBaseLite	bike_009.bmp	430	0	0	921654	2	2	2	29
Send	CouchBaseLite	bike_016.bmp	412	0	0	921654	2	2	2	29
Send	CouchBaseLite	bike_028.bmp	428	0	0	921654	2	2	2	29
Send	CouchBaseLite	bike_050.bmp	419	0	0	921654	2	2	2	29
Send	CouchBaseLite	bike_075.bmp	401	0	0	921654	2	2	2	29
Send	CouchBaseLite	bike_080.bmp	226	0	0	921654	2	2	2	29
Send	CouchBaseLite	bike_094.bmp	240	0	0	921654	2	2	2	29
Send	CouchBaseLite	bike_160.bmp	282	0	0	921654	2	2	2	29
Send	CouchBaseLite	bike_241.bmp	264	0	0	921654	2	2	2	29
Send	CouchBaseLite	bike_282.bmp	230	0	0	921654	2	2	2	29
Send	CouchBaseLite	bike_009.bmp	322	0	0	921654	3	2	2	29
Send	CouchBaseLite	bike_016.bmp	261	0	0	921654	3	2	2	29
Send	CouchBaseLite	bike_028.bmp	264	0	0	921654	3	2	2	29
Send	CouchBaseLite	bike_050.bmp	394	0	0	921654	3	2	2	29
Send	CouchBaseLite	bike_075.bmp	410	0	0	921654	3	2	2	29

Send	CouchBaseLite	bike_080.bmp	499	0	0	921654	3	2	2	29

Fuente: Elaboración propia

6.3.7.3.1 Análisis e interpretación de los resultados de la replicación inserción.

En la **Tabla 13** se presentan los resultados obtenidos para cada variable con su respectiva interpretación y análisis, entre el tiempo de la replicación inserción (replicationmiliTime) y las distintas características que intervinieron en el proceso.

Tabla 13:

Análisis e interpretación de la correlación de la replicación Inserción

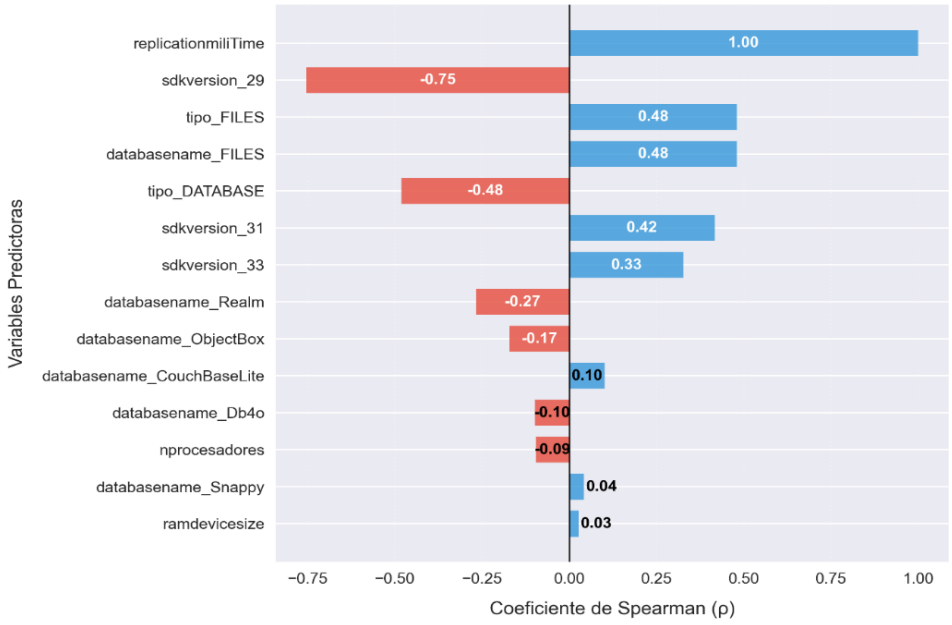
Feature	Correlación Spearman	Interpretación
replicationmiliTime	1.000	Es la variable objetivo, correlación perfecta consigo misma.
databasename_FILES	0.481	Alta correlación positiva con el tiempo de replicación. Sugiere que el uso del método FILES está asociado con un incremento en el tiempo de replicación.
tipo_FILES	0.481	Alta correlación positiva con el tiempo de replicación. Sugiere que el uso del método FILES está asociado con un incremento en el tiempo de replicación.
sdkversion_33	0.327	Correlación positiva moderada, las versiones más recientes de SDK tienen a aumentar ligeramente los tiempos de replicación.
sdkversion_31	0.417	Correlación positiva moderada, las versiones más recientes de SDK tienen a aumentar ligeramente los tiempos de replicación.
Databasename_CouchBaseLite	0.101	Correlación positiva muy débil, prácticamente nula, su impacto sobre el tiempo de replicación en mínimo o insignificante.
databasename_Snappy	0.042	Correlación positiva muy débil, prácticamente nula, su impacto sobre el tiempo de replicación en mínimo o insignificante.
nprocesadores	-0.095	Correlación negativa muy débil, un mayor número de procesadores tienden a asociarse con una ligera disminución.
ramdevicesize	0.027	Correlación positiva muy débil, no parece influir en el tiempo de replicación.
databasename_Db4o	-0.099	Correlación negativa débil, su uso puede reducir el tiempo de replicación modestamente.
databasename_ObjectBox	-0.171	Correlación negativa moderada, presenta mayor efecto en la reducción en el tiempo de replicación.
Databasename_Realm	-0.267	Correlación negativa moderada alta, presenta mayor efecto en la reducción en el tiempo de replicación.
sdkversion_29	-0.753	Correlación negativa muy fuerte, el uso de esta versión tiende a reducir significativamente el tiempo de replicación.
tipo_DATABASE	-0.481	Correlación negativa fuerte es significativamente más eficiente en términos de tiempo que el método DATABASE.

Fuente: Elaboración propia

En el **Gráfico 5** se muestra la correlación entre las diversas características de las variables y el tiempo de la replicación inserción.

Gráfico 5

Correlación en el tiempo de replicación inserción



Fuente: Elaboración propia

Análisis de la correlación de Spearman sobre el tiempo de replicación.

Se realizó un análisis de correlación de Spearman para evaluar el impacto de diferentes variables sobre el tiempo de replicación (replicationmiliTime**)**. Esta metodología mide la fuerza y dirección de la relación monotónica entre variables, permitiendo identificar cuáles factores incrementan o reducen el tiempo de replicación de forma significativa.

Las variables con alta correlación positiva son databasename_FILES y tipo_FILES con igual correlación (0.481). Esto indica que el uso de la configuración FILES (tanto a nivel de base de datos como de tipo de operación) incrementa de manera significativa el tiempo de replicación.

Las variables sdkversion_33 (correlación 0.327) y sdkversion_31 (correlación 0.417) presentan una correlación positiva moderada, aumentando el tiempo de replicación. Las versiones más recientes del SDK (31 y 33), tienden a aumentar ligeramente el tiempo de replicación en comparación con versiones anteriores.

Las variables databasename_CouchBaseLite (correlación 0.101) y databasename_Snappy (correlación 0.042) presentan correlaciones positivas débiles. El efecto de *CouchBaseLite y Snappy* sobre el tiempo de replicación es mínimo o casi nulo, sin cambios sustanciales detectados.

Las variables nprocesadores (correlación -0.095) y ramdevicesize (correlación -0.027) muestran correlaciones negativas débiles. Una mayor cantidad de procesadores y memoria RAM se asocia con ligeras mejoras en el tiempo de replicación, aunque su efecto es pequeño e inconsistente. Mientras que databasename_Db4o (correlación -0.099) indica una moderada reducción en el tiempo de replicación, en cambio databasename_ObjectBox (correlación -0.171) y databasename_Realm (correlación -0.267) tienen correlaciones negativas más marcadas y se destaca como una de las bases de datos que mejor contribuye a reducir los tiempos de replicación.

La variable sdkversion_29 (correlación -0.753) muestra una fuerte correlación negativa. Sorprendentemente, SDK 29 reduce considerablemente el tiempo de replicación, superando a versiones más recientes (31 y 33) en eficiencia. Y tipo_DATABASE (correlación - 0.481) presenta una correlación negativa alta. El método DATABASE mejora sustancialmente el tiempo de replicación frente al método FILES.

En consideración, El método FILES, tanto como base de datos como tipo de operación, es el principal responsable del incremento en los tiempos de replicación. Las bases de datos como *Realm y ObjectBox*, y el método DATABASE, ofrecen alternativas más eficientes, reduciendo notablemente el tiempo de operación. Las versiones más recientes del SDK no garantizan mejores tiempos de replicación; de hecho, SDK 29 muestra un comportamiento más favorable. Y la capacidad de hardware (número de procesadores y tamaño de RAM) tiene un efecto menor, pero sigue apuntando a mejoras cuando los recursos son mayores.

6.3.7.3.2 Análisis de los coeficientes de la regresión múltiple de la replicación inserción.

La **Tabla 14** presenta la interpretación y análisis de la regresión múltiple para el tiempo de replicación inserción (replicationmiliTime), considerando el intercepto y los coeficientes de las diferentes variables.

Tabla 14:

Análisis de la regresión múltiple en la replicación inserción

Feature	**Coeficiente**	**Interpretación**
nprocesadores	-101.409	Coeficiente negativo fuerte, el aumentar el número de procesadores disminuye el tiempo de replicación.

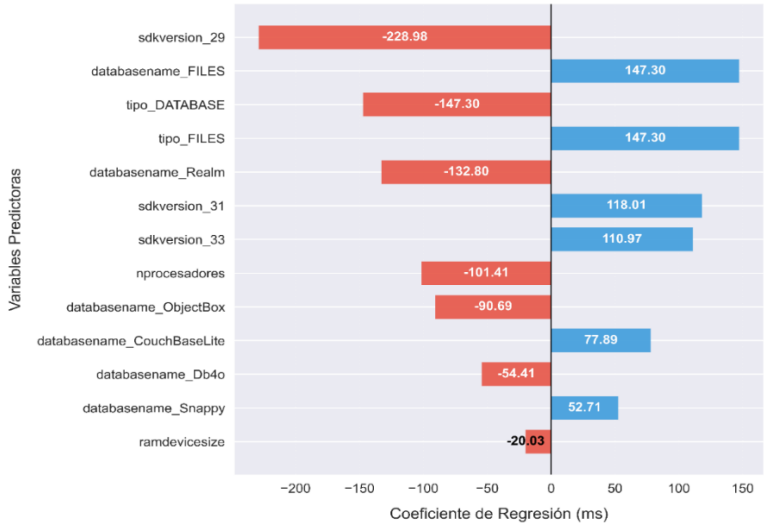
ramdevicesize	-20.032	Coeficiente negativo moderado, el aumentar el tamaño de la memoria disminuye el tiempo de replicación.
databasename_CouchBaseLite	77.892	Coeficiente positivo alto, aumenta moderadamente el tiempo de replicación, aunque mucho menos que FILE.
databasename_Db4o	-54.412	Coeficiente negativo, Es una base de datos que mejora los tiempos de replicación ligeramente.
databasename_FILES	147.298	Coeficiente positivo alto, al usar este método aumenta el tiempo de replicación.
databasename_ObjectBox	-90.691	Coeficiente negativo fuerte, Es una base de datos que mejora los tiempos de replicación moderadamente.
databasename_Realm	-132.796	Coeficiente negativo fuerte, Es una base de datos que mejora los tiempos de replicación sustancialmente.
databasename_Snappy	52.709	Coeficiente positivo, aumenta un poco el tiempo de replicación.
tipo_DATABASE	-147.298	Coeficiente negativo fuerte, Usar el método DATABASE en lugar de files reduce fuertemente el tiempo de replicación.
Tipo_FILES	147.298	Coeficiente positivo, al usar este método aumenta el tiempo de replicación.
sdkversion_29	-228.977	Coeficiente negativo fuerte, reduce ligeramente el tiempo de replicación.
sdkversion_31	118.066	Coeficiente positivo alto, aumenta muy poco el tiempo de replicación.
sdkversion_33	110.972	Coeficiente positivo alto, aumenta muy poco el tiempo de replicación.
Intercept	759.787	Es el valor promedio del tiempo de replicación cuando todas las variables son cero.

Fuente: Elaboración propia

En el **Gráfico 6** se muestra el comportamiento de los coeficientes en la regresión múltiple entre las diversas de las variables y el tiempo de replicación inserción (replicationmiliTime).

Gráfico 6

Comportamiento de los coeficientes para el tiempo de replicación inserción



Fuente: Elaboración propia

Interpretación del modelo de regresión múltiple para el tiempo de replicación inserción.

Cada coeficiente representa el cambio esperado en el tiempo de replicación por cada unidad de cambio en la variable correspondiente, manteniendo todas las demás variables constantes. El valor de la intersección (759.787) indica el valor promedio del tiempo de replicación cuando todas las variables predictoras están en cero (o en su nivel de referencia).

Las variables databasename_FILES y tipo_FILES con coeficiente iguales 147.298 presentan gran incremento en el tiempo de replicación al usar el método y base FILES. La variable databasename_CouchBaseLite (77.892) aumenta moderadamente el tiempo de replicación, aunque mucho menos que FILES. La variable databasename_Snappy (52.709) aumenta poco el tiempo de replicación. Mientras que, las variables o versiones sdkversion_31 (118.006) y sdkversion_33 (110.972) incrementan ligeramente el tiempo de replicación.

La variable tipo_DATABASE (coeficiente -147.298) usar el método DATABASE en lugar de FILES reduce fuertemente el tiempo de replicación, la variable databasename_Realm (-132.796) es una base de datos que mejora sustancialmente los tiempos, las variables databasename_ObjectBox (-90.691) y ObjectBox también tiene un impacto importante en la reducción del tiempo. Mientras que databasename_Db4o (-54.412) mejora el tiempo moderadamente, el número de procesadores nprocesadores (-101.409) ayuda a disminuir el tiempo de replicación, la memoria RAM ramdevicesize (-20.032) a mayor memoria favorece el tiempo de replicación y sdkversion_29 (-228.977) versión más antigua del SDK (29) reduce ligeramente el tiempo de replicación.

En consideración, el mayor impacto positivo y negativo en el tiempo de replicación está dado por el tipo de acceso a la base de datos (FILES vs DATABASE), FILES tanto en tipo como en base de datos empeora significativamente los tiempos, DATABASE, así como bases como *Realm y ObjectBox*, son las mejores opciones para optimizar el rendimiento, mejorar hardware con más procesadores y memoria RAM contribuye también a reducir el tiempo, aunque en menor medida que la elección de base de datos y método y la versión SDK 29 sigue mostrando mejores comportamientos que SDK 31 y 33.

6.3.7.4 Análisis e interpretación de la información en la replicación búsqueda.

En la **Tabla 15** se muestra una parte de la información obtenida en la replicación búsqueda, por la gran magnitud de los datos obtenidos.

Tabla 15:

Información obtenida en la implementación del software en la replicación búsqueda

method	databasename	filename	replicationmiliTime	replicationsegTime	replicationminTime	datasize	repeticion	nprocesadores	ramdevicesize	sdkversion
Search	CouchBaseLite	bike_009.bmp	610	0	0	921654	0	2	2	29
Search	CouchBaseLite	bike_016.bmp	238	0	0	921654	0	2	2	29
Search	CouchBaseLite	bike_028.bmp	1460	1	0	921654	0	2	2	29
Search	CouchBaseLite	bike_050.bmp	314	0	0	921654	0	2	2	29
Search	CouchBaseLite	bike_075.bmp	306	0	0	921654	0	2	2	29
Search	CouchBaseLite	bike_080.bmp	217	0	0	921654	0	2	2	29
Search	CouchBaseLite	bike_094.bmp	209	0	0	921654	0	2	2	29
Search	CouchBaseLite	bike_160.bmp	267	0	0	921654	0	2	2	29
Search	CouchBaseLite	bike_241.bmp	248	0	0	921654	0	2	2	29
Search	CouchBaseLite	bike_282.bmp	251	0	0	921654	0	2	2	29
Search	CouchBaseLite	bike_009.bmp	252	0	0	921654	1	2	2	29
Search	CouchBaseLite	bike_016.bmp	309	0	0	921654	1	2	2	29
Search	CouchBaseLite	bike_028.bmp	265	0	0	921654	1	2	2	29
Search	CouchBaseLite	bike_050.bmp	253	0	0	921654	1	2	2	29
Search	CouchBaseLite	bike_075.bmp	246	0	0	921654	1	2	2	29
Search	CouchBaseLite	bike_080.bmp	240	0	0	921654	1	2	2	29
Search	CouchBaseLite	bike_094.bmp	226	0	0	921654	1	2	2	29
Search	CouchBaseLite	bike_160.bmp	255	0	0	921654	1	2	2	29
Search	CouchBaseLite	bike_241.bmp	232	0	0	921654	1	2	2	29
Search	CouchBaseLite	bike_282.bmp	1763	1	0	921654	1	2	2	29
Search	CouchBaseLite	bike_009.bmp	1534	1	0	921654	2	2	2	29
Search	CouchBaseLite	bike_016.bmp	236	0	0	921654	2	2	2	29
Search	CouchBaseLite	bike_028.bmp	230	0	0	921654	2	2	2	29
Search	CouchBaseLite	bike_050.bmp	1684	1	0	921654	2	2	2	29
Search	CouchBaseLite	bike_075.bmp	223	0	0	921654	2	2	2	29
Search	CouchBaseLite	bike_080.bmp	252	0	0	921654	2	2	2	29
Search	CouchBaseLite	bike_094.bmp	274	0	0	921654	2	2	2	29
Search	CouchBaseLite	bike_160.bmp	356	0	0	921654	2	2	2	29
Search	CouchBaseLite	bike_241.bmp	252	0	0	921654	2	2	2	29
Search	CouchBaseLite	bike_282.bmp	250	0	0	921654	2	2	2	29
Search	CouchBaseLite	bike_009.bmp	365	0	0	921654	3	2	2	29
Search	CouchBaseLite	bike_016.bmp	269	0	0	921654	3	2	2	29
Search	CouchBaseLite	bike_028.bmp	268	0	0	921654	3	2	2	29
Search	CouchBaseLite	bike_050.bmp	236	0	0	921654	3	2	2	29
Search	CouchBaseLite	bike_075.bmp	246	0	0	921654	3	2	2	29
Search	CouchBaseLite	bike_080.bmp	389	0	0	921654	3	2	2	29

Fuente: Elaboración propia

6.3.7.4.1 Análisis e interpretación de los resultados de la replicación búsqueda.

La **Tabla 16** se muestra la interpretación y un análisis de la correlación utilizando entre distintas características y el tiempo de replicación búsqueda (replicationmiliTime).

Tabla 16:

Análisis de la correlación en la replicación búsqueda

Feature	Correlación Spearman	Interpretación
replicationmiliTime	1.000	Es la variable objetivo, correlación perfecta consigo misma.

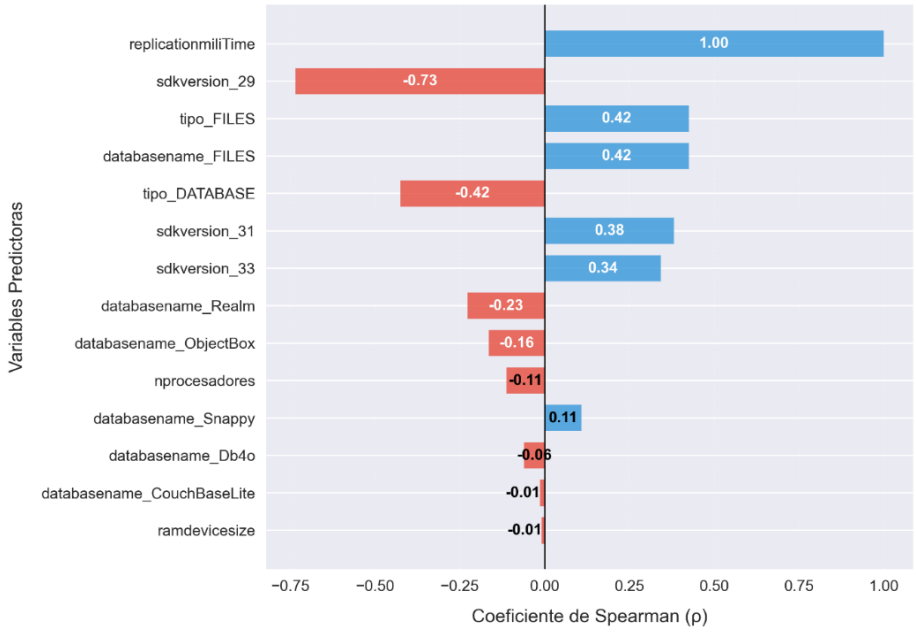
Feature	Correlación Spearman	Interpretación
databasename_FILES	0.425	Moderada correlación positiva con el tiempo de replicación. Sugiere que el uso del método FILES está asociado con un incremento en el tiempo de replicación.
tipo_FILES	0.425	Moderada correlación positiva con el tiempo de replicación. Sugiere que el uso del tipo FILES está asociado con un incremento en el tiempo de replicación.
sdkversion_33	0.343	Correlación positiva moderada, las versiones más recientes de SDK tienen a aumentar ligeramente los tiempos de replicación.
sdkversion_31	0.382	Correlación positiva moderada, las versiones más recientes de SDK tienen a aumentar ligeramente los tiempos de replicación.
databasename_Snappy	0.109	Correlación positiva débil, su impacto sobre el tiempo de replicación en mínimo o insignificante.
nprocesadores	-0.113	Correlación negativa débil, un mayor número de procesadores tienden a asociarse con una ligera disminución en el tiempo de replicación.
Databasename_CouchBaseLite	-0.0.14	Correlación negativa muy débil, prácticamente nula, su impacto sobre el tiempo de replicación en mínimo o insignificante.
databasename_Db4o	-0.06	Correlación negativa muy débil, su uso puede reducir el tiempo de replicación modestamente.
ramdevicesize	-0.009	Correlación negativa muy débil, un mayor tamaño de memoria RAM tienden a asociarse con una ligera disminución en el tiempo de replicación.
databasename_ObjectBox	-0.165	Correlación negativa baja, presenta mayor efecto en la reducción en el tiempo de replicación.
Databasename_Realm	-0.227	Correlación negativa moderada, presenta mayor efecto en la reducción en el tiempo de replicación.
sdkversion_29	-0.734	Correlación negativa muy fuerte, el uso de esta versión tiende a reducir significativamente el tiempo de replicación.
tipo_DATABASE	-0.425	Correlación negativa moderada es significativamente más eficiente en términos de tiempo que el método DATABASE

Fuente: Elaboración propia

En el **Gráfico 7** se muestra la correlación entre las diversas características de las variables y el tiempo de replicación de búsqueda (replicationmiliTime).

Gráfico 7

Correlación en la replicación búsqueda



Fuente: Elaboración propia

Análisis de la correlación sobre el tiempo de replicación búsqueda.

Se realizó un análisis de correlación de Spearman para evaluar el impacto de diferentes variables sobre el tiempo de replicación búsqueda (replicationmiliTime**)**. E**s**ta metodología mide la fuerza y dirección de la relación monotónica entre variables, permitiendo identificar cuáles factores incrementan o reducen el tiempo de replicación de forma significativa.

Las variables con moderación correlación positiva son databasename_FILES y tipo_FILES con igual correlación (0.425). Esto indica que el uso de la configuración FILES (tanto a nivel de base de datos como de tipo de operación) incrementa de manera significativa el tiempo de replicación.

Las variables sdkversion_33 (0.343) y sdkversion_31 (0.382) presentan una correlación positiva baja, aumentando el tiempo de replicación. Las versiones más recientes del SDK (31 y 33), tienden a aumentar ligeramente el tiempo de replicación en comparación con versiones anteriores. Tienen un impacto leve en el incremento del tiempo.

Las variables databasename_Snappy (0.109) y nprocesadores (-0.113) tienen una correlación muy baja, su efecto sobre el tiempo de replicación es mínimo o irrelevante.

La variable tipo_DATABASE (-0.425) presenta una moderada correlación negativa, el uso del tipo DATABASE contribuye significativamente a reducir el tiempo de replicación. La versión sdkversion_29 (-0.734) tiene una correlación negativa moderada, usar la versión SDK 29 tiende a

mejorar los tiempos en comparación con versiones más recientes. La variable databasename_Realm (-0.227) presenta una relación negativa moderada, esta base *Realm* ayuda a disminuir el tiempo de replicación. La variable databasename_ObjectBox (-0.165) presenta correlación negativa baja, esta se asocia también con una mejora del tiempo, aunque en menor medida que *Realm*. Mientras que databasename_Db4o (-0.006), y databasename_CouchBaseLite (-0.014) tienen correlaciones negativas muy bajas, aunque tienden a mejorar ligeramente los tiempos, su impacto es poco relevante.

En consideración FILES es la principal causa de aumento del tiempo de replicación, DATABASE, Realm, ObjectBox y el uso del SDK 29 reducen el tiempo de manera significativa, la influencia de parámetros de hardware como número de procesadores y RAM es mínima en este escenario y las versiones más recientes del SDK (31 y 33) no ayudan a mejorar el tiempo de replicación; incluso tienden a empeorarlo ligeramente.

6.3.7.4.2 Análisis e interpretación de los coeficientes de la regresión lineal múltiple de la replicación búsqueda.

La **Tabla 17** presenta la interpretación y un análisis de la regresión múltiple para el tiempo de replicación búsqueda (replicationmiliTime), considerando el intercepto y los coeficientes de las diferentes variables.

Tabla 17

Análisis de la regresión múltiple en la replicación búsqueda

Feature	**Coeficiente**	**Interpretación**
nprocesadores	-76.462	Coeficiente negativo fuerte, el aumentar el número de procesadores disminuye el tiempo de replicación.
ramdevicesize	-40.77	Coeficiente negativo, el aumentar el tamaño de la memoria disminuye el tiempo de replicación.
databasename_CouchBaseLite	11.201	Coeficiente positivo, aumenta el tiempo de replicación ligeramente.
databasename_Db4o	-13.034	Coeficiente negativo, Es una base de datos que mejora los tiempos de replicación ligeramente.
databasename_FILES	62.424	Coeficiente positivo alto, al usar este método aumenta el tiempo de replicación.
databasename_ObjectBox	-59.449	Coeficiente negativo, Es una base de datos que mejora los tiempos de replicación moderadamente.
databasename_Realm	-74.13	Coeficiente negativo fuerte, Es una base de datos que mejora los tiempos de replicación sustancialmente.
databasename_Snappy	72.988	Coeficiente positivo, aumenta significativamente el tiempo de replicación.
tipo_DATABASE	-62.424	Coeficiente negativo, Usar el método DATABASE en lugar de files reduce fuertemente el tiempo de replicación.
Tipo_FILES	62.424	Coeficiente positivo, al usar este método aumenta el tiempo de replicación.

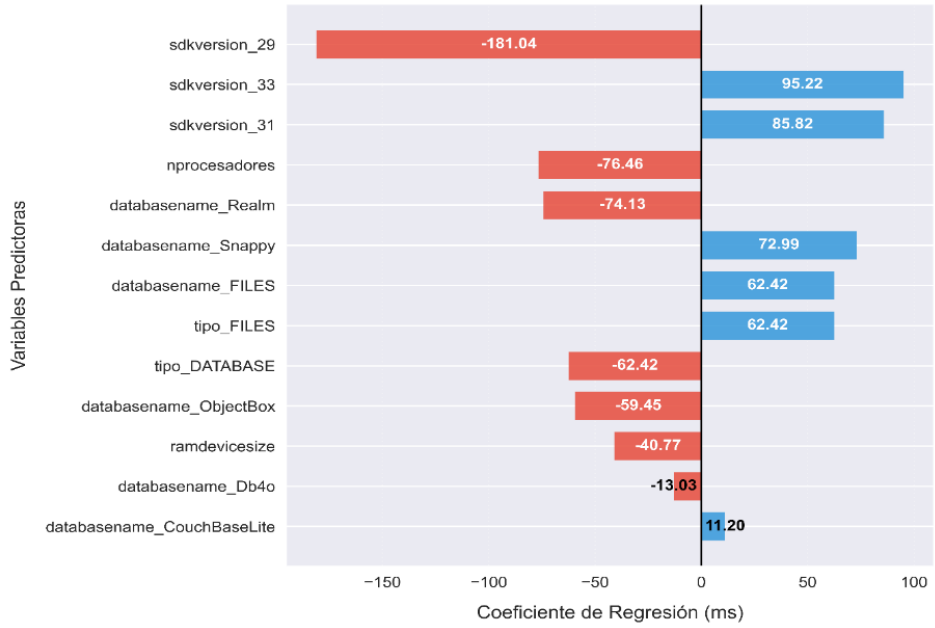
sdkversion_29	-181.041	Coeficiente negativo fuerte, disminuye notoriamente el tiempo de replicación.
sdkversion_31	85.823	Coeficiente positivo alto, aumenta el tiempo de replicación.
sdkversion_33	95.218	Coeficiente positivo, aumenta el tiempo de replicación.
Intercept	676.181	Es el valor promedio del tiempo de replicación cuando todas las variables son cero.

Fuente: Elaboración propia

En el **Gráfico 8** se muestra el comportamiento de los coeficientes en la regresión múltiple entre las diversas de las variables y el tiempo de replicación búsqueda (replicationmiliTime).

Gráfico 8

Análisis del comportamiento de los coeficientes en la replicación búsqueda



Fuente: Elaboración propia

Análisis de resultados de los coeficientes de la regresión lineal múltiple para el tiempo de replicación búsqueda.

Cada coeficiente representa el cambio esperado en el tiempo de replicación por cada unidad de cambio en la variable correspondiente, manteniendo todas las demás variables constantes. El valor de la intersección (676.181) indica el valor promedio del tiempo de replicación cuando todas las variables predictoras están en cero (o en su nivel de referencia).

Las variables databasename_FILES y tipo_FILES con coeficiente iguales 62.424 presentan gran incremento en el tiempo de replicación al usar el método y base FILES. La variable databasename_*Snappy* (72.988) y databasename_CouchBaseLite (11.201) aumentan ligeramente el

tiempo de replicación. Mientras que, las variables o versiones sdkversion_31 (85.823) y sdkversion_33 (95.218) también incrementan ligeramente el tiempo de replicación.

La variable databasename_Db4o (-13.034) disminuye levemente el tiempo de replicación, seguidos de la memoria RAM ramdevicesize (-40.77) a mayor memoria favorece el tiempo de replicación y el número de procesadores nprocesadores (-76.462) ayuda a disminuir el tiempo de replicación.

La variable tipo_DATABASE (-62.424) usar el método DATABASE en lugar de FILES reduce fuertemente el tiempo de replicación, la variable databasename_*Realm* (-74.988) es una base de datos que mejora sustancialmente los tiempos, la variable databasename_*ObjectBox* (-59.449) y *ObjectBox* también tiene un impacto importante en la reducción del tiempo. Mientras que mejora el tiempo moderadamente, el número de procesadores nprocesadores ayuda a disminuir el tiempo de replicación, la memoria RAM ramdevicesize favorece el tiempo de replicación y sdkversion_29 versión más antigua del SDK (29) disminuyen el tiempo de replicación, las versiones sdkversion_33 (95.218) y sdkversion_31 (85.823) aumentan ligeramente el tiempo de replicación.

En consideración; El mayor impacto positivo y negativo en el tiempo de replicación está dado por el tipo de acceso a la base de datos (FILES vs DATABASE), FILES tanto en tipo como en base de datos empeora significativamente los tiempos, DATABASE, así como bases como *Realm y ObjectBox*, son las mejores opciones para optimizar el rendimiento, mejorar hardware con más procesadores y memoria RAM contribuye también a reducir el tiempo, aunque en mayor medida que la elección de base de datos y método y la versión SDK 29 muestra mejores comportamientos que SDK 31 y 33.

6.3.8 Resultados comparativos sistemas de almacenamiento de datos NoSQL y el sistema de almacenamiento tradicional.

El análisis se hace comparativo de las bases de datos NoSQL (CouchBaseLite, Db4o, ObjectBox, Realm y Snappy) y el sistema de almacenamiento tradicional (FILES) en cuatro métricas clave: tiempo de inserción, tiempo de replicación de inserción, tiempo de búsqueda y tiempo de replicación de búsqueda. El análisis se basa en gráficos de caja y bigote (*boxplots*), que permiten evaluar tendencias centrales, dispersión y valores atípicos.

Este tipo de gráfico permite identificar de manera clara y precisa la mediana, el rango intercuartílico (IQR), los valores máximos y mínimos, así como los posibles valores atípicos en cada base de datos analizada. Al emplear los *boxplots*, se facilita la detección de la variabilidad de los datos y la simetría o asimetría de las distribuciones, lo cual es esencial para evaluar la consistencia y estabilidad del rendimiento de cada sistema de gestión de bases de datos (SGBD).

6.3.8.1 Análisis comparativo en la inserción.

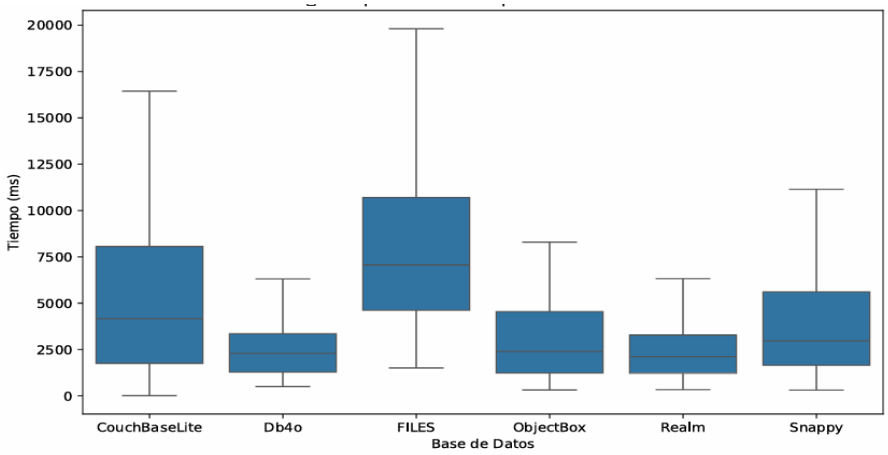
La **Tabla 18** presenta los datos estadísticos descriptivos fundamentales, incluyendo la media, la mediana, el primer cuartil (Q1) y el tercer cuartil (Q3), en milisegundos y en el **Gráfico 9** se muestra el comportamiento del tiempo de inserción en las bases de datos (en milisegundos) para el análisis comparativo.

Tabla 18:
Datos estadísticos descriptivos en la inserción en milisegundos

databasename	Media	Mediana	Desviación estándar	Q1	Q3
CouchBaseLite	5769.9	4165.0	6465.3	1755.0	8061.0
Db4o	2504.7	2289.0	1589.3	1284.5	3357.0
FILES	10651.7	7067.5	10050.9	4622.3	10705.0
ObjectBox	2940.9	2399.0	1936.2	1237.3	4544.8
Realm	2414.2	2116.5	1676.8	1223.0	3277.3
Snappy	3997.4	2966.0	3355.4	1653.0	5613.0

Fuente: Elaboración propia

Gráfico 9:
Tiempo de inserción en las bases de datos



Fuente: Elaboración propia

Interpretación.

Realm y Db4o son las bases de datos más eficientes para inserción, con tiempos bajos, con una ligera ventaja en la mediana y menor dispersión. *ObjectBox* es una opción intermedia. *CouchBaseLite y Snappy* presentan tiempos altos y variabilidad considerable, lo que puede comprometer la predictibilidad del rendimiento. FILES tiene el peor rendimiento, con tiempos muy elevados y alta variabilidad, lo que la hace poco recomendable para cargas de inserción elevadas.

6.3.8.2 Análisis comparativo en la búsqueda.

La **Tabla 19** presenta los datos estadísticos descriptivos fundamentales, incluyendo la media, la mediana, el primer cuartil (Q1) y el tercer cuartil (Q3) en milisegundos, y en el **Gráfico 10** se muestra el comportamiento del tiempo de búsqueda en las bases de datos para el análisis comparativo (en milisegundos).

Tabla 19:

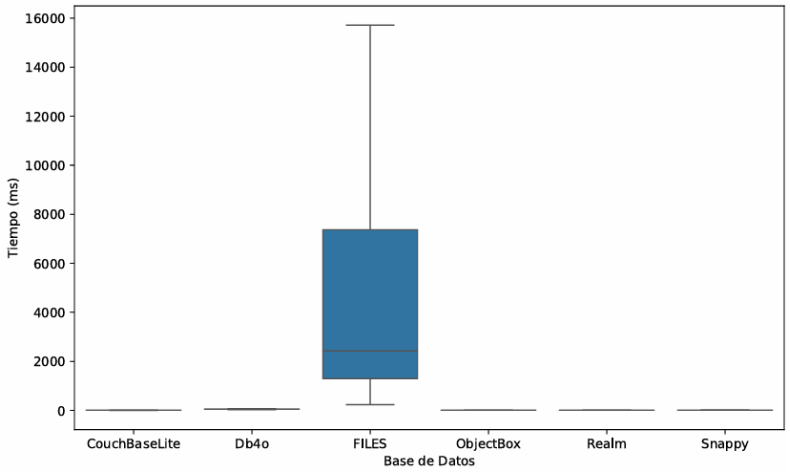
Datos estadísticos descriptivos en la búsqueda en milisegundos

databasename	Media	Mediana	Desviación estándar	Q1	Q3
CouchBaseLite	4.8	2.0	14.7	1.0	3.0
Db4o	48.0	45.0	15.3	40.0	51.0
FILES	6772.8	2426.0	10229.9	1294.0	7366.8
ObjectBox	4.2	3.0	4.8	2.0	4.0
Realm	3.6	2.0	8.0	1.0	3.0
Snappy	5.8	4.0	6.0	2.0	6.0

Fuente: Elaboración propia

Gráfico 10:

Tiempos de búsqueda en las bases de datos

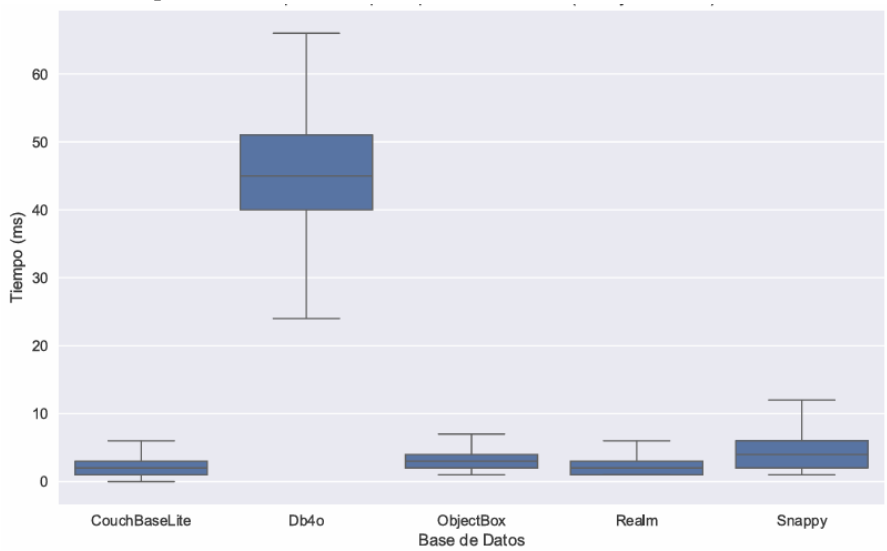


Fuente: Elaboración propia

El **Gráfico 10A** muestra el comportamiento de las bases de datos en la búsqueda para los sistemas de almacenamiento NoSQL (*Db4o, ObjectBox, CouchBaseLite,, Realm y Snappy)* indicualizados al Grafico 10 para su mejor comprension por los valores que toman en el tiempo.

Gráfico 10A:

Comportamiento del tiempo las bases de datos NoSQL

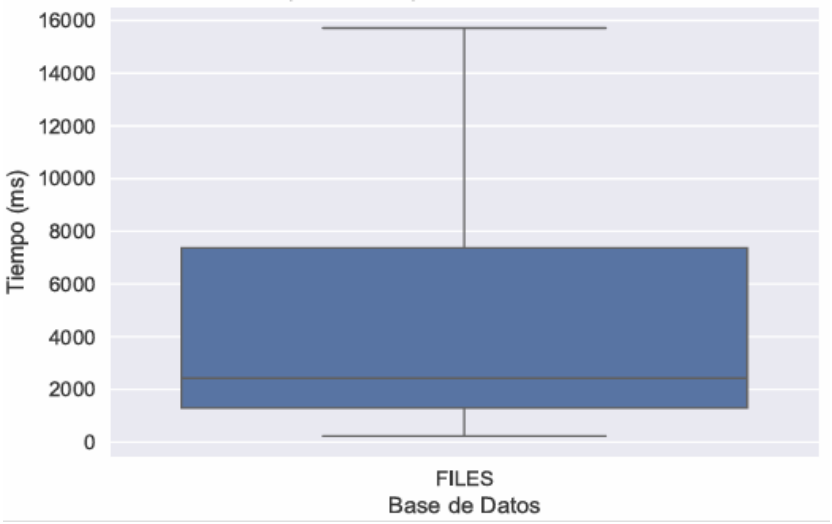


Fuente: Elaboración propia

El **Gráfico 10B** muestra el comportamiento de los sistemas de ficheros tradicionales (FILES) indicualizados al gráfico 10 para su mejor comprension por los valores que toman en el tiempo.

Gráfico 10A:

Comportamiento del tiempo en el sistema de ficheros



Fuente: Elaboración propia

Interpretación

Realm y ObjectBox son claramente las mejores opciones para las operaciones de búsqueda, con *Realm* teniendo una ligera ventaja. *CouchBaseLite y Snappy* presentan buena mediana, sobresalen como opciones altamente eficientes y consistentes en la búsqueda, con tiempos de búsqueda prácticamente instantáneos y sin valores atípicos detectables. *Db4o* tiene un rendimiento significativamente peor que los anteriores. FILES es completamente inadecuada para las operaciones de búsqueda.

6.3.8.3 Análisis comparativo en la replicación inserción.

La **Tabla 20** presenta los datos estadísticos descriptivos fundamentales, incluyendo la media, la mediana, el primer cuartil (Q1) y el tercer cuartil (Q3) en milisegundos, y en el **Gráfico 11** se muestra el comportamiento del tiempo de replicación inserción en las bases de datos para el análisis comparativo (en milisegundos).

Tabla 20:

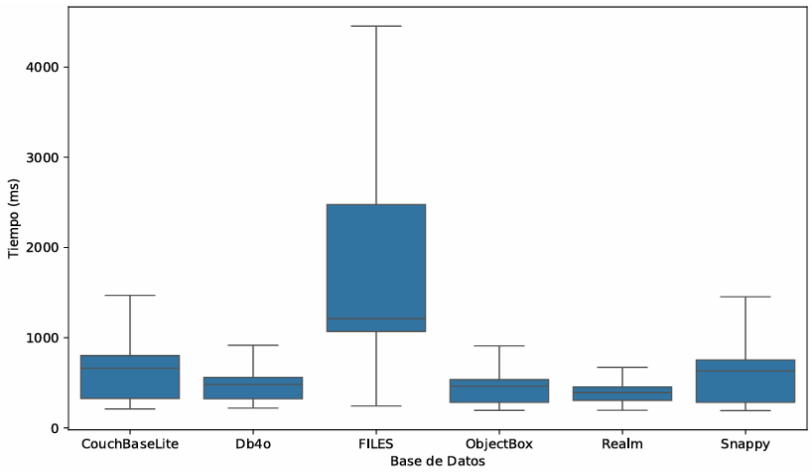
Datos estadísticos descriptivos en la replicación inserción en milisegundos

databasename	Media	Mediana	Desviación estándar	Q1	Q3
CouchBaseLite	696.1	660.0	526.7	327.0	799.8
Db4o	490.1	481.0	207.8	321.8	560.0
FILES	1624.7	1211.5	803.3	1069.3	2475.8
ObjectBox	452.6	463.5	178.7	282.5	535.0
Realm	420.4	393.0	279.1	305.3	451.8
Snappy	617.8	632.5	457.2	282.8	753.8

Fuente: Elaboración propia

Gráfico 11:

Tiempos de replicación inserción en las bases de datos



Fuente: Elaboración propia

Interpretación.

Las bases de datos NoSQL *Realm, ObjectBox* y *Db4o* muestran los tiempos de replicación más bajos y estables, siendo opciones preferibles para aplicaciones donde la velocidad es crucial. *CouchBaseLite, y Snappy* tienen un rendimiento intermedio por su variabilidad, pero deben ser evaluadas con cuidado debido a sus valores atípicos. FILES destaca negativamente con los mayores tiempos promedio y una alta variabilidad, por lo que no es recomendable para sistemas de almacenamiento en la replicación rápida.

6.3.8.4 Análisis comparativo en la replicación búsqueda.

La **Tabla 21** presenta los datos estadísticos descriptivos fundamentales, incluyendo la media, la mediana, el primer cuartil (Q1) y el tercer cuartil (Q3) en milisegundos, y en el **Gráfico 12** se muestra el comportamiento del tiempo de replicación búsqueda en las bases de datos para el análisis comparativo (en milisegundos).

Tabla 21:

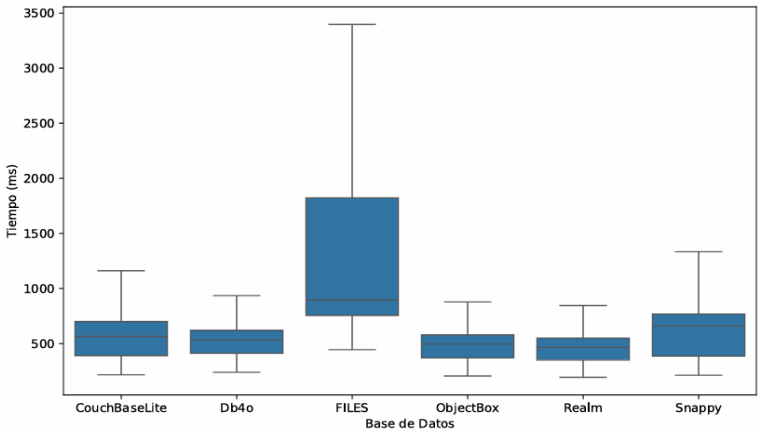
Datos estadísticos descriptivos de la replicación búsqueda en milisegundos

databasename	Media	Mediana	Desviación estándar	Q1	Q3
CouchBaseLite	635.1	565.0	505.9	390.3	700.0
Db4o	537.9	532.0	177.7	411.3	621.0
FILES	1207.6	894.5	597.8	756.0	1822.5
ObjectBox	496.1	496.0	170.1	371.0	578.8
Realm	469.4	465.5	155.7	350.3	548.8
Snappy	630.9	660.0	258.0	387.5	768.8

Fuente: Elaboración propia

Gráfico 12:

Tiempos de replicación búsqueda en las bases de datos



Fuente: Elaboración propia

Interpretación

Realm, ObjectBox y Db4o se destacan por ser más rápidos y consistentes, con menor dispersión en los tiempos de replicación búsqueda *y CouchBaseLite* tienen rendimientos aceptables, pero con varios casos extremos que podrían afectar el tiempo. Snappy mantiene un desempeño promedio sin destacar ni por velocidad ni por lentitud. FILES muestra el peor desempeño con tiempos de replicación de búsqueda altos y alta variabilidad.

6.3.8.5 Análisis comparativo de las bases de datos con las cuatro métricas mediante la *boxplots.*

La **Tabla 22** presenta la posición de las diferentes bases de datos en las cuatro métricas evaluadas en el estudio, lo que permite clasificarlas de mayor a menor según su rendimiento.

Tabla 22:

Clasificación de las métricas en inserción, búsqueda y replicación

Posición	Inserción	Búsqueda	Replicación Inserción	Replicación Búsqueda
1	*Realm*	*Realm*	*Realm*	*Realm*
2	*Db4o*	*ObjectBox*	*ObjectBox*	*ObjectBox*
3	*ObjectBox*	*CouchBaseLite*	*Db4o*	*Db4o*
4	*Snappy*	*Snappy*	*Snappy*	*CouchBaseLite*
5	*CouchBaseLite*	*Db4o*	*CouchBaseLite*	*Snappy*
6	*FILES*	*FILES*	*FILES*	*FILES*

Fuente: Elaboración propia

En general, FILES presenta el peor rendimiento en las cuatro métricas analizadas, con tiempos más altos y alta variabilidad. *Realm* es la mejor base de datos para las custro métricas. *Db4o y ObjectBox* sobresalen como las bases de datos más eficientes y estables en inserción, búsqueda y replicación

inserción, mientras que *CouchBaseLite y Snappy* tienen desempeños intermedios, con algunos casos de valores atípicos. Para aplicaciones que requieren alto rendimiento y consistencia, se recomienda priorizar *Realm* que presentan un peso más alto en las métricas analizadas.

6.3.9 Análisis de la prueba de hipótesis.

El análisis se realiza de forma comparativa entre las bases de datos NoSQL (CouchBaseLite, Db4o, ObjectBox, Realm y Snappy) y el sistema de almacenamiento tradicional (FILES), considerando cuatro métricas clave: tiempo de inserción, tiempo de búsqueda, tiempo de replicación de inserción y tiempo de replicación de búsqueda. Este análisis tiene como propósito evaluar las diferencias de rendimiento entre las soluciones tecnológicas y comprobar las hipótesis previamente planteadas mediante pruebas estadísticas apropiadas.

En cada tabla se encuentra las bases de datos en dos grupos de las bases de datos y el sistema de almacenamiento tradicional, que permite compararlos entre sí, la diferencia de la media, la probabilidad, los límites inferiores y superiores y la prueba.

6.3.9.1 Análisis de la prueba de hipótesis en los tiempos de inserción.

La **Tabla 23** muestra el resultado de la información estadística obtenida, para la prueba de hipótesis, que compara los tiempos de rendimiento entre diferentes bases de datos en los tiempos de inserción.

Tabla 23:
Datos de la prueba de hipótesis en los tiempos de inserción

Grupo 1	Grupo 2	Diferencia Medias	Probabilidad P	Límite inferior	Límite superior	Prueba
CouchBaseLite	Db4o	-3265.173	0	-4190.205	-2340.142	True
CouchBaseLite	FILES	4881.842	0	3956.810	5806.874	True
CouchBaseLite	ObjectBox	-2828.94	0	-3753.976	-1903.912	True
CouchBaseLite	Realm	-3355.648	0	-4280.680	-2430.616	True
CouchBaseLite	Snappy	-1772.477	0	-2697.509	-847.445	True
Db4o	FILES	8147.016	0	7214.432	9079.599	True
Db4o	ObjectBox	436.229	0.766	-496.354	1368.813	False
Db4o	Realm	-90.475	0.999	-1023.058	842.109	False
Db4o	Snappy	1492.696	0.0001	560.113	2425.279	True
FILES	ObjectBox	-7710.786	0	-8643.370	-6778.203	True
FILES	Realm	-8237.490	0	-9170.074	-7304.907	True
FILES	Snappy	-6654.319	0	-7586.903	-5721.736	True
ObjectBox	Realm	-526.704	0.592	-1459.287	405.879	False
ObjectBox	Snappy	1056.467	0.016	123.883	1989.05	True

Realm	Snappy	1583.171	0	650.587	2515.754	True

Fuente: Elaboración propia

Análisis, interpretación y conclusión de la prueba.

Realm muestra diferencias significativas siendo más rápido que casi todas las demás bases de datos. ObjectBox es similar a *Realm* en rendimiento (sin diferencia estadística significativa entre ellos). *Snappy* es más rápido que *Db4o y FILES*, pero más lento que *Realm y ObjectBox*. *Db4o* presenta un rendimiento intermedio, sin diferencias significativas con. *CouchBaseLit*e es más lento que *Realm, ObjectBox y Snappy*, pero más rápido que *Db4o y FILES*. *FILES* es claramente la opción más lenta en todas las comparaciones.

Las diferencias entre *Realm, ObjectBox y Db4o* no son estadísticamente significativas ($p > 0.05$). *CouchBaseLite* tiene un rendimiento mixto: mejor que Db4o y FILES, pero peor que las demás. *Snappy* se ubica en un punto intermedio, siendo significativamente más rápido que *Db4o y FILES*. *FILES* es significativamente más lento que todas las demás opciones.

Análisis comparativo en la prueba.

Realm y ObjectBox son los líderes sin diferencia estadística (probabilidad 0.592), *Realm* es 3,355 ms más rápido que *CouchBaseLite* y 8,237 ms más rápido que *FILES*, *ObjectBox* es 2,828 ms más rápido que *CouchBaseLite* y 7,710 ms más rápido que *FILES*.

Db4o tiene un comportamiento interesante, No es significativamente diferente de *Realm* ni a *ObjectBox* (Probabilidades 0.9998 y 0.7662 respectivamente), pero es significativamente más rápido que *Snappy y CouchBaseLite*.

Snappy es 1,772 ms más rápido que *CouchBaseLite,* pero 1,583 ms más lento que *Realm*; *CouchBaseLite* es el peor entre las bases de datos NoSQL, siendo mucho más lento que *Db4o, Snappy, Realm y ObjectBox*.

FILES es inaceptablemente lento, 4,881 ms más lento que *CouchBaseLite* y 8,237 ms más lento que *Realm.* En es peor que todas las demás opciones por un margen enorme.

6.3.9.2 Análisis de la prueba de hipótesis en los tiempos de búsqueda.

La **Tabla 24** muestra el resultado de la información estadística obtenida que compara la prueba de hipótesis los tiempos de rendimiento entre diferentes bases de datos en los tiempos de búsqueda.

Tabla 24:

Datos de la prueba de hipótesis en los tiempos de búsqueda

Grupo 1	Grupo 2	Diferencia Medias	Probabilidad	Límite inferior	Límite superior	Prueba
CouchBaseLite	Db4o	43.153	0.999	-530.572	616.877	False
CouchBaseLite	FILES	6767.982	0	6194.257	7341.707	True
CouchBaseLite	ObjectBox	-0.633	1	-574.358	573.092	False
CouchBaseLite	Realm	-1.178	1	-574.903	572.547	False
CouchBaseLite	Snappy	0.988	1	-572.737	574.713	False
Db4o	FILES	6724.829	0	6148.271	7301.387	True
Db4o	ObjectBox	-43.786	0.999	-620.344	532.772	False
Db4o	Realm	-44.331	0.999	-620.888	532.227	False
Db4o	Snappy	-42.165	0.999	-618.723	534.393	False
FILES	ObjectBox	-6768.615	0	-7345.174	-6192.057	True
FILES	Realm	-6769.16	0	-7345.718	-6192.602	True
FILES	Snappy	-6766.994	0	-7343.552	-6190.436	True
ObjectBox	Realm	-0.545	1	-577.103	576.014	False
ObjectBox	Snappy	1.621	1	-574.937	578.179	False
Realm	Snappy	2.166	1	-574.392	578.724	False

Fuente: Elaboración propia

Análisis, interpretación y conclusión de la prueba.

Realm, ObjectBox, Snappy y CouchBaseLite son base de datos equivalentes estadísticamente en búsquedas (no hay diferencias significativas), siendo mejor *Realm* con una diferencia muy mínima con respecto a las demás. *Db4o* también entra en el grupo anterior, pero en otras pruebas (como el análisis anterior) suele ser más lento. *FILES* es el peor por un margen enorme que lo hace mucho más lento comparado con los demás, se debe evitar si la velocidad de búsqueda es importante.

Análisis comparativo en la prueba.

Realm, ObjectBox, CouchBaseLite, Snappy y Db4o tienen diferencias estadísticamente insignificantes (probabilidad 1 en casi todas las comparaciones), las diferencias son mínimas (como se puede observar *Realm vs CouchBaseLite* 1.1779 ms, *Db4o vs Snappy* 42.1647 ms). *ObjectBox y Realm* son técnicamente los más rápidos, pero la diferencia es irrelevante en la práctica.

FILES es catastróficamente lento, 6,767 ms más lento que *CouchBaseLite*, 6,769 ms más lento que *Realm y ObjectBox* no hay solapamiento en los intervalos de confianza en todas las comparaciones.

6.3.9.3 Análisis de la prueba de hipótesis en los tiempos de replicación inserción.

La **Tabla 25** muestra el resultado de la información estadística obtenida que compara la prueba de hipótesis los tiempos de rendimiento entre diferentes bases de datos en los tiempos de replicación inserción.

Tabla 25:
Datos de la prueba de hipótesis en los tiempos de replicación inserción

Grupo 1	Grupo 2	Diferencia Medias	Probabilidad	Límite inferior	Límite superior	Prueba
CouchBaseLite	Db4o	-206.002	0	-269.990	-142.015	True
CouchBaseLite	FILES	928.568	0	864.581	992.556	True
CouchBaseLite	ObjectBox	-243.516	0	-307.504	-179.529	True
CouchBaseLite	Realm	-275.702	0	-339.690	-211.715	True
CouchBaseLite	Snappy	-78.292	0.007	-142.279	-14.304	True
Db4o	FILES	1134.570	0	1070.583	1198.558	True
Db4o	ObjectBox	-37.514	0.551	-101.502	26.474	False
Db4o	Realm	-69.7	0.024	-133.688	-5.712	True
Db4o	Snappy	127.711	0	63.723	191.698	True
FILES	ObjectBox	-1172.085	0	-1236.072	-1108.09	True
FILES	Realm	-1204.271	0	-1268.258	-1140.283	True
FILES	Snappy	-1006.86	0	-1070.848	-942.872	True
ObjectBox	Realm	-32.186	0.706	-96.174	31.802	False
ObjectBox	Snappy	165.225	0	101.237	229.213	True
Realm	Snappy	197.411	0	133.423	261.398	True

Fuente: Elaboración propia

Análisis, interpretación y conclusión de la prueba

Realm y ObjectBox son los líderes, con *Realm* siendo ligeramente mejor, pero sin diferencia estadísticamente significativa entre ellos por la probabilidad 0.7062. *CouchBaseLite* sigue de cerca, siendo significativamente más rápido que *Db4o y FILES*. *Snappy* muestra un rendimiento notablemente mejor que *Db4o y FILES*, pero peor que las opciones superiores. *Db4o* es significativamente más lento que las primeras opciones. *FILES* es claramente el peor, siendo más lento todos los demás.

Para operaciones de replicación/inserción, *Realm* es la mejor opción seguida de cerca por ObjectBox. *CouchBaseLite* es una buena alternativa si no se puede usar *Realm u ObjectBox*. Las diferencias entre *Realm, ObjectBox y CouchBaseLite* son significativas, pero entre *Realm y ObjectBox* no lo son. *FILES* debe evitarse completamente para este tipo de operaciones.

Análisis comparativo en la prueba.

Realm es la base de datos significativamente más rápida que *CouchBaseLite* (275 ms); Db4o (69 ms); *FILES* (1204 ms); Snappy (197 ms) y no difiere significativamente de *ObjectBox* (probabilidad 0.7062).

ObjectBox base de datos muy cercano a *Realm*, en rendimiento estadísticamente equivalente a Realm; significativamente más rápido que *CouchBaseLite* (243 ms); *Db4o* (37 ms, con probabilidad (0.5509); *FILES* (1172 ms); *Snappy* (165 ms).

CouchBaseLite presenta buen rendimiento, más rápido que *Db4o* (206 ms) y *FILES* (928 ms); más lento que *Realm* (275 ms); *ObjectBox* (243 ms) y Snappy (78 ms).

Snappy presenta rendimiento medio, más rápido que *Db4o* (127 ms) y *FILES* (1006 ms); más lento que las 3 primeras opciones.

Db4o presenta un rendimiento bajo, solo supera significativamente a *FILES y m*ás lento que todas las demás opciones

FILES el sistema tradicional es el más lento, significativamente más lento que todas las alternativas, entre 928 - 1204 ms más lento que otras opciones

6.3.9.4 Análisis de la prueba de hipótesis en los tiempos de replicación búsqueda.

La **Tabla 26** muestra el resultado de la información estadística obtenida que compara los tiempos de rendimiento entre diferentes bases de datos en los tiempos de replicación búsqueda.

Tabla 26:

Datos de la prueba de hipótesis en los tiempos de replicación búsqueda

Grupo 1	Grupo 2	Diferencia Medias	Probabilidad	Límite inferior	Límite superior	Prueba
CouchBaseLite	Db4o	-97.156	0	-146.521	-47.791	True
CouchBaseLite	FILES	572.509	0	523.144	621.874	True
CouchBaseLite	ObjectBox	-138.974	0	-188.339	-89.608	True
CouchBaseLite	Realm	-165.695	0	-215.060	-116.330	True
CouchBaseLite	Snappy	-4.207	0.999	-53.572	45.158	False
Db4o	FILES	669.665	0	620.301	719.031	True
Db4o	ObjectBox	-41.817	0.151	-91.182	7.547	False
Db4o	Realm	-68.538	0.001	-117.904	-19.173	True
Db4o	Snappy	92.949	0	43.584	142.314	True

FILES	ObjectBox	-711.483	0	-760.848	-662.118	True
FILES	Realm	-738.204	0	-787.569	-688.839	True
FILES	Snappy	-576.716	0	-626.081	-527.351	True
ObjectBox	Realm	-26.721	0.636	-76.086	22.644	False
ObjectBox	Snappy	134.767	0	85.401	184.132	True
Realm	Snappy	161.488	0	112.123	210.853	True

Fuente: Elaboración propia

Análisis, interpretación y conclusión de la prueba

Realm y ObjectBox son las mejores opciones (sin diferencia práctica entre ellos), *CouchBaseLite y Snappy* son alternativas decentes si no se pueden usar las anteriores, *FILES* es extremadamente lento, *Db4o* solo recomendable si no hay otra opción. *CouchBaseLite y Snappy* tienen un rendimiento estadísticamente igual en búsquedas (probabilidad 0.999), lo que los hace intercambiables en este aspecto.

Análisis comparativo en la prueba

Realm es la base de datos más rápida, significativamente más rápido que *CouchBaseLite* (165.7 ms), *Db4o* (68.5 ms), *FILES* (738.2 ms) y *Snappy* (161.5 ms) y presenta rendimiento equivalente a ObjectBox (probabilidad 0.6362).

ObjectBox es muy cercano a *Realm*, no hay diferencia significativa, más rápido que *CouchBaseLite* (139 ms), *Db4o* (41.8 ms, con probabilidad 0.151), *FILES* (711.5 ms) *y Snappy* (134.8 ms).

CouchBaseLite tercer mejor rendimiento, superior a *Db4o* (97.2 ms) y *FILES* (572.5 ms), similar a *Snappy* (con probabilidad 0.999) e inferior a *Realm y ObjectBox.*

Snappy presenta rendimiento intermedio, mejor que *Db4o* (92.9 ms) y F*ILES* (576.7 ms), estadísticamente igual a *CouchBaseLite y m*ás lento que *Realm y ObjectBox*

Db4o es de bajo rendimiento, solo supera significativamente a *FILES*, más lento que todas las demás opciones

FILES es significativamente más lento que todas las alternativas, entre 576 – 738 ms.

6.3.9.5 Análisis comparativo de las bases de datos con las cuatro métricas utilizando los resultados Tukey.

La **Tabla 27** presenta la posición de las diferentes bases de datos en las cuatro métricas evaluadas en el estudio, lo que permite clasificarlas de mayor a menor según su rendimiento.

Tabla 27:
Clasificación de las métricas en inserción, búsqueda y replicación

Posición	Inserción	Búsqueda	Replicación Inserción	Replicación Búsqueda
1	*Realm*	*Realm*	*Realm*	*Realm*
2	*ObjectBox*	*ObjectBox*	*ObjectBox*	*ObjectBox*
3	*Snappy*	*Snappy*	*CouchBaseLite*	*CouchBaseLite*
4	*Db4o*	*CouchBaseLite*	*Snappy*	*Snappy*
5	*CouchBaseLite*	*Db4o*	*Db4o*	*Db4o*
6	*FILES*	*FILES*	*FILES*	*FILES*

Fuente: Elaboración propia

En general, FILES presenta el peor rendimiento en las cuatro métricas analizadas, con tiempos más altos y alta variabilidad. *Realm* es la mejor base de datos para las custro métricas. *ObjectBox* sobresalen como las bases de datos más eficientes y estables en inserción, búsqueda y replicación inserción, siendo la segunda mejor, mientras que *CouchBaseLite, Snappy y Db4o* tienen desempeños intermedios. Para aplicaciones que requieren alto rendimiento y consistencia, se recomienda priorizar *Realm* que presentan un mejor valor comparado con las demás.

CAPITULO 7
CONCLUSIONES Y RECOMENDACIONES

7.1 Conclusiones generales

En este trabajo se realizó una evaluación comparativa del Sistema de Base de Datos NoSQL y el Gestor de Ficheros Tradicional en Dispositivos Móviles Android. La evaluación consistió en la elaboración de una aplicación para dispositivos móviles Android, creada con el entorno de desarrollo Android Studio, y la medición de los tiempos de inserción, búsqueda y replicación, en función de las variables número de procesadores, tamaño de RAM, sistema operativo, bases de datos y el sistema de ficheros dadas en la ecuación. Los datos obtenidos se analizan considerando el coeficiente de correlación de Spearman y el análisis estadístico de la regresión lineal múltiple.

A partir de los análisis de ajustes de normalización, IQR, *One Hot Code*, de los coeficientes de correlación de Spearman, de regresión múltiple y análisis estadístico descriptivo, realizados sobre los tiempos de inserción, búsqueda y replicación, se evidencia que diversas variables impactan de manera significativa en el desempeño de las operaciones de bases de datos en los dispositivos móviles. En contraste, un mayor número de procesadores y una mayor cantidad de memoria RAM contribuyen a reducir significativamente este tiempo, favoreciendo así un mejor desempeño del sistema. Asimismo, el método de almacenamiento utilizado desempeña un rol crucial: la configuración basada en FILES tiende a incrementar tanto el tiempo de inserción como de búsqueda y replicación, mientras que el método DATABASE permite una reducción considerable de estos tiempos, destacándose como una opción más eficiente.

En cuanto a las bases de datos evaluadas, *Db4o y ObjectBox* presentan mejores tiempos en la inserción, búsqueda y replicación inserción y frente a la configuración FILES, aunque en menor medida, mientras que *Realm, ObjectBox* muestran un mejor desempeño en la replicación búsqueda. Respecto a la búsqueda, las bases de datos *Realm*, *ObjectBox*, *CouchBaseLite*, *Snappy y Db4o* se asocian consistentemente con una disminución significativa en el tiempo requerido. Además, se observó que las versiones más recientes del SDK (31 y 33) tienden a reducir el tiempo de búsqueda, aunque en el caso de inserciones y replicaciones sus efectos son marginales o incluso negativos si se comparan con la versión SDK 29, que sigue mostrando un comportamiento más favorable.

Estos resultados pueden explicarse en parte a través del fenómeno de envejecimiento de software descrito por Cotroneo et al., 2020, donde se señala que, a medida que el sistema operativo Android

evoluciona tiende a consumir más recursos y a experimentar una degradación progresiva del rendimiento. Este envejecimiento podría justificar por qué versiones más recientes del sistema operativo no siempre traducen en mejoras lineales de rendimiento, especialmente en operaciones intensivas de inserción y replicación.

En lo referente a la replicación, tanto en operaciones de inserción como de búsqueda, el tipo de almacenamiento utilizado es determinante. El método FILES, tanto en su configuración como en su base de datos asociada, incrementa de manera significativa los tiempos de replicación, debido al bloqueo que genera en operaciones de escritura y lectura. Por el contrario, el método DATABASE y las bases de datos *Realm y ObjectBox* permiten reducciones sustanciales, posicionándose como las mejores alternativas para optimizar el rendimiento en escenarios móviles. Si bien la mejora en hardware, como el incremento en el número de procesadores y la ampliación de la memoria RAM, también contribuye a una reducción en los tiempos de replicación, su impacto es menor en comparación con la elección del método de almacenamiento y la base de datos.

El sistema de ficheros tradicionales se ve afectado en el tiempo de inserción, búsqueda y replicación debido al bloqueo presentado en la anotación (*JNI critical lock held for 222.067ms on Thread[4, tid=7411, Runnable, Thread=0x756507feec00, peer=0x132503e8, 'pool-2-thread-1'*]), presentado en el log.

Finalmente, el análisis integral de los modelos sugiere que, para optimizar el desempeño de aplicaciones móviles en términos de tiempos de inserción, búsqueda y replicación, resulta más determinante seleccionar adecuadamente el método de acceso (privilegiando DATABASE sobre FILES), emplear bases de datos eficientes como *Realm y ObjectBox*, y, en menor medida, considerar configuraciones de hardware robustas y versiones de SDK optimizadas, siendo SDK 29 una alternativa que sigue ofreciendo ventajas frente a versiones más recientes. Por consiguiente, la adecuada combinación de estos factores resulta clave para lograr sistemas móviles más eficientes, robustos y de mejor desempeño operativo.

7.2 Recomendaciones

Evaluar el tipo de operaciones en paralelo (programación reactiva o en hilos) para determinar la posibilidad de mejora y optimización en los procesos de inserción, búsqueda y replicación evitando los posibles bloqueos en operaciones de lectura/escritura vistos en el proyecto (*JNI critical lock held for 222.067ms on Thread[4, tid=7411, Runnable, Thread=0x756507feec00, peer=0x132503e8, 'pool-2-thread-1']*).

Evaluar la posibilidad emplear arquitecturas hibridas, NoSQL y SQL para la replicación y sincronización de datos con ficheros o archivos binarios.

Evaluar la forma en que el sistema realiza la fragmentación de archivos tanto en los ficheros binarios como las bases de datos NoSQL.

Analizar la posibilidad de usar otros métodos de replicación diferentes a *websockets* con el fin de poder garantizar el envío de información entre dispositivos.

Explorar en las nuevas versiones de Android con el fin de revisar el punto de variaciones de tiempo donde el factor del sistema operativo mejora o empeora el tiempo de cada operación.

Verificar la multicolinealidad a través del cálculo de factores de inflación de varianza (VIF).

7.3 Limitaciones del estudio.

Este trabajo presenta algunas limitaciones que deben considerarse al interpretar los resultados:

Simulación de replicación mediante *WebSockets* vs. mecanismos nativos.

Aunque el uso de WebSockets permite una comparación controlada y estandarizada de la replicación en sistemas NoSQL, esta aproximación no replica fielmente los protocolos nativos de cada base de datos. Por lo tanto, los resultados podrían variar en entornos reales donde la replicación se gestiona internamente mediante las estrategias propias de cada sistema.

Pruebas en emuladores vs. dispositivos reales.

Las evaluaciones de rendimiento se realizaron principalmente en entornos emulados, lo que puede introducir discrepancias en comparación con ejecuciones en dispositivos físicos. Los emuladores, aunque útiles para pruebas controladas, no siempre capturan aspectos como latencia de red real, consumo de recursos hardware o variaciones en el rendimiento debido a limitaciones del sistema operativo. Por ello, los resultados podrían diferir en implementaciones reales con hardware específico.

7.4 Cumplimiento de los objetivos

Tabla 28:

Complimiento de objetivos

Objetivo específico	Actividades	Producto
Identificar las tecnologías relacionadas con los sistemas de almacenamiento de datos en los dispositivos móviles.	Búsqueda de artículos en Bases de datos como: IEEE y *Google scholar* y realizando matriz de extracción identificando los diferentes tipos de bases de datos SQL y NoSQL y los sistemas de almacenamiento de datos en los dispositivos móviles. Resumen presentado en la Figura 1 en el capítulo 5 pagina 29 y en el capítulo 6 en la Tabla 1 página 37	Anteproyecto entregado a la Universidad de Medellín.
Identificar las variables que pueden impactar en las en la velocidad de acceso y replicación.	El sistema de ficheros y sistemas de almacenamiento de datos con sus características de la implementación de *Couchbase Mobile, Snappy, DB4o, Realm y ObjectBox,* utilizando estudios primarios realizados, formularios de extracción de información en diferentes bases de datos. Ilustrado en la ecuación determinada. Ilustrado en el capítulo 5 página 32.	Análisis de los resultados obtenidos presentados en el numeral 6.3.7 de este trabajo.
Desarrollar una aplicación que permita comparar el tiempo de procesamiento de un sistema de ficheros tradicional contra el de base de datos NoSQL bajo diferentes configuraciones de hardware y software.	Desarrollo de aplicación en el entorno de desarrollo Android Studio para la plataforma Android que por medio de emuladores se realizaron las pruebas que permitieron extraer la información para los respectivos análisis. Ilustrado en el capítulo 6 numeral 6.3	El APK de la aplicación instalable para los dispositivos móviles Android.
Evaluar el sistema de archivos de ficheros tradicional contra el de bases de datos no relacionales en dispositivos móviles Android, en su velocidad de respuesta y velocidad de replicación.	En la aplicación de Android APK instalada en los emuladores se obtuvieron 4 *datasets* que fueron convertidos a .csv. Elaboración de Scripts Python para el procesamiento de la información obtenida en los *datasets*, la cual fue interpretada mediante análisis estadísticos. Tanto para en *insert*, *search* y la replicación en *insert y search*, Se ilustra en el capítulo 6.	Análisis comparativo de los resultados obtenidos presentados en los numerales 6.3.7.3 y 6.3.7.4 de este trabajo.

REFERENCIAS

Abramova, V., Bernardino, J., & Furtado, P. (2014). Experimental Evaluation of NoSQL Databases. *International Journal of Database Management Systems*, *6*(3), 01–16. https://doi.org/10.5121/IJDMS.2014.6301

Baz Alonso, A., Ferreira Artime, I., Álvarez Rodríguez, M., & García Baniello, R. (2009). *Dispositivos móviles*.

Carvalho, I., Sá, F., & Bernardino, J. (2023). Performance Evaluation of NoSQL Document Databases: Couchbase, CouchDB, and MongoDB. *Algorithms 2023, Vol. 16, Page 78*, *16*(2), 78. https://doi.org/10.3390/A16020078

Cotroneo, D., Iannillo, A. K., Natella, R., & Pietrantuono, R. (2020). A Comprehensive Study on Software Aging across Android Versions and Vendors. *Empirical Software Engineering*, *25*, 3357–3395. https://doi.org/10.1007/s10664-020-09838-3

Croche Andreu, A. (2021). *Encuestas y evaluación del alumno en una aplicación móvil de cursos online individuales sin profesor*. https://oa.upm.es/68004/

Esquivel, W. C., & Sevilla, G. L. (2021). Paralelismos entre bases de datos relacionales y no relacionales (un enfoque en seguridad). *ReCIBE, Revista Electrónica de Computación, Informática, Biomédica y Electrónica*, *10*(2), C1-16. https://doi.org/10.32870/RECIBE.V10I2.189

Gómez, T. (2022). *Introducción a las bases de datos NoSQL. Sistemas de bases de datos orientados a grafos*. https://riunet.upv.es/handle/10251/186175

González Morte, D. (2019). *Estudio de dispositivos móviles, vulnerabilidades y auditoría de seguridad de aplicaciones móviles*. https://openaccess.uoc.edu/handle/10609/95126

Jianhong, L., & Xinyue, W. (2020). Design of mobile learning platform based on android. *15th International Conference on Computer Science and Education, ICCSE 2020*, 257–261. https://doi.org/10.1109/ICCSE49874.2020.9201659

Kuckartz, U., Rädiker, S., Ebert, T., & Schehl, J. (2013). *Statistik*. https://doi.org/10.1007/978-3-531-19890-3

Nacional, U., Plata, L., & Capdevila, G. E. (2015). *Replicación de bases de datos NoSQL en dispositivos móviles*. http://sedici.unlp.edu.ar/handle/10915/48085

Ortiz Pinilla, J., Felipe, A., & Rico, O. (2021). ¿Pearson y Spearman, coeficientes intercambiables? *Comunicaciones En Estadística*, *14*(1), 53–63. https://doi.org/10.15332/23393076.6769

Osorio Pérez, C. A. (2016). Sistema Basado En Una Arquitectura Distribuida Con Base De Datos Nosql Para La Recuperación De Información De Múltiples Bibliotecas Con Óptimos Niveles De Rendimiento. *Carlos Alfredo,Osorio Pérez*. https://repositorio.ucv.edu.pe/handle/20.500.12692/23187

Osorio Trejos, L. F. (2019). *BASES DE DATOS*. https://www.academia.edu/41135813/Investigacion_final_LUIS_fernando_Trejos_osorio

Repiso, S., Prats, C., & José, J. (2017). *Sistema de ficheros con control automático de versiones*. https://upcommons.upc.edu/handle/2117/100394

Rivero Hernández, D. de la C., Pérez Vázquez, R., Mora, C., & Vila Labrada, J. (2013). Bases de datos móviles. *Tlatemoani: Revista Académica de Investigación, ISSN-e 1989-9300, No. 14, 2013*, *14*, 4. https://dialnet.unirioja.es/servlet/articulo?codigo=7330824&info=resumen&idioma=ENG

Román Pérez, A. (2020). *Comparación de rendimiento entre bases de datos Relacionales, NoSQL y Blockchain*.

Tesone, F. (2021). *Un análisis comparativo de bases de datos para dispositivos móviles*. http://sedici.unlp.edu.ar/handle/10915/118824

Venegas Bravo, B., Alexander, J., Bravo Ruiz, M., & Arturo, J. (2022). Análisis comparativo de rendimiento de gestores de base de datos NOSQL documentales. *Repositorio Institucional - USS*. https://repositorio.uss.edu.pe/handle/20.500.12802/9220

Medellín, 18 de agosto de 2025

Señores
COMITÉ DE POSTGRADOS
Maestría en Ingeniería de Software
Universidad de Medellín.

Asunto: Radicación Trabajo de Grado corregido para optar el título de Magíster.

Cordial saludo:

Me permito presentar ante ustedes el trabajo de grado titulado **"Evaluación Comparativa del Sistema de Base de Datos NoSQL y el Gestor de Ficheros Tradicional en Dispositivos Móviles Android",** elaborado como requisito parcial para optar al título de **Magíster en Ingeniería de Software.**

Este trabajo aborda una propuesta de evaluación comparativa entre sistemas de bases de datos NoSQL y el gestor tradicional de ficheros en dispositivos móviles Android con sistema de archivos ext4., con el objetivo general "Evaluar las características velocidad de acceso y replicación de los sistemas de base de datos NoSQL de ficheros y gestor de ficheros tradicional en dispositivos móviles *Android* mediante un análisis comparativo basado en un caso de estudio". La investigación se desarrolló bajo la orientación del doctor Jesús Andrés Hincapié Londoño (director) y el doctor Mauricio González Palacio (codirector), adscritos a los programas asociados a Ingeniería de Sistemas, a quienes agradezco profundamente por su acompañamiento, comentarios y orientación a lo largo del proceso.

El trabajo que entrego para su evaluación representa el resultado de meses de estudio, reflexión, recolección y análisis de información, así como del compromiso por aportar al conocimiento en la Ingeniería de Software. Espero que sus observaciones y comentarios enriquezcan aún más el aporte de este trabajo.

Agradezco de antemano su lectura y consideración.

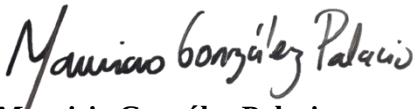
Atentamente,

Jorge Andrés García Zapata
Estudiante de Maestría
C.C. 1020 461 028
Email: jgrcia028@soyudemedellin.edu.co
Celular: 3217987335

Vo.Bo.

Jesús Andrés Hincapié Londoño
Director
C.C. 8.029.296

Vo.Bo.



Mauricio González Palacio
Codirector
C.C. 71.262.387