# Pontificia Universidad Católica del Ecuador Sede Esmeraldas

Escuela de

Ingeniería en sistemas

Informe de Estudio de Caso:

# Testing Entre Bases de Datos SQL y NoSQL

Previo a obtención del Grado académico de:

Ingeniero en Sistemas


Autor

Arroyo Lerma Diego

Asesor

Mg. Jaime Sayago Heredia


Esmeraldas – 2018

Estudio de caso aprobado luego de haber dado cumplimiento a los requisitos exigidos, previos a la obtención del título de INGENIERO EN SISTEMAS Y COMPUTACIÓN.

## TRIBUNAL DE GRADUACIÓN

**Título: "**Testing entre bases de datos SQL y NoSQL"

**Autor:** ARROYO LERMA DIEGO ISRAEL

_____
Mg. Jaime Sayago Heredia
ASESOR

_____
Mg. Wilson Chango Sailema
LECTOR

_____
Mg. Kléber Posligua Flores
LECTOR

_____
Mg. Xavier Quiñónez Ku
DIRECTOR DE ESCUELA

_____
Mg. Maritza Demera
SECRETARIA

## DECLARACIÓN DE AUTORÍA

Quien suscribe DIEGO ISRAEL ARROYO LERMA con número de cedula 080345486-7, hace constar que es el autor del estudio de caso titulado: Testing entre bases de datos SQL y NoSQL, el cual constituye una elaboración personal realizada únicamente con la dirección del asesor de dicho trabajo, Mg Jaime Sayago Heredia.

En tal sentido, manifiesto la originalidad de la conceptualización del trabajo, interpretación de datos y la elaboración del análisis, dejando establecido que aquellos aportes intelectuales de otros autores se han referenciado debidamente en el texto de dicho trabajo.

En la Ciudad de Esmeraldas, a los 08 días del mes de enero del dos mil dieciocho

ARROYO LERMA DIEGO ISRAEL
C.I: 080345486-7

# CERTIFICACIÓN

Mg. Jaime Sayago Heredia, Docente investigador de la PUCE-E, certifica que:

El estudio de caso realizado por DIEGO ISRAEL ARROYO LERMA bajo el título "TESTING ENTRE BASES DE DATOS SQL Y NOSQL" reúne los requisitos de calidad, originalidad y presentación exigibles a una investigación científica y que han sido incorporadas al documento final, las sugerencias realizadas, en consecuencia, está en condiciones de ser sometida a la valoración del Tribunal encargado de juzgarla.

Y para que conste a los efectos oportunos, firma la presente en Esmeraldas, el __ de mayo del 2017

Mg. Jaime Sayago Heredia
ASESOR.

# DEDICATORIA

Dedico esta investigación a mis padres por brindarme la oportunidad de acceder a la educación superior.

# AGRADECIMIENTO

Agradezco enteramente a Dios por darme la vida por hacerme parte de sus planes y darme propósito en la vida.

"Porque de Él, por El y para El son todas las cosas. A Él sea la gloria para siempre"
Romanos 11:36 (LBLA)

# ÍNDICE

# ÍNDICE DE FIGURAS

## RESUMEN

Con el desarrollo y expansión del internet, la computación en la nube y la aparición de aplicaciones como Facebook, YouTube y Twitter surge la necesidad de almacenar y procesar grandes volúmenes de datos. A raíz de estos acontecimientos el modelo tradicional de base de datos se ha visto enfrentado a varios desafíos especialmente en aplicaciones de alta concurrencia con manejo de grandes volúmenes de información, escenario bajo el cual emplear bases de datos relacionales no parase ser apropiado.

Es bajo este contexto que surge el modelo no relacional de base de datos (NoSQL), este nuevo enfoque surge como una solución a los constantes requerimientos de procesamiento y análisis de enormes cantidades de datos. Ambos paradigmas a pesar de tener enfoques distintos poseen algo en común y es que ambos nos permiten gestionar datos de manera prolija. Es por ello por lo que se torna interesante la cuestión de; ¿Cuál es mejor?, es decir cuál es más eficiente a nivel de rendimiento, y mejor aún seria cuestionarse ¿Por qué NoSQL es mucho más rápido que SQL?, y a que se debe tanta eficiencia. Este documento tiene como propósito presentar el proceso que conlleva el aplicar pruebas de rendimientos a ambos conceptos y posteriormente realizar un análisis de los datos obtenidos.

En este se hace una breve descripción de ambos paradigmas (SQL y NoSQL), se describe sus propiedades y se hace una comparación de sus propiedades a nivel conceptual.

Una vez esclarecido los conceptos claves que intervinieron en el proceso de investigación y posterior a haber desarrollado las diferentes pruebas y análisis de los datos obtenidos, podemos decir que; Indiscutiblemente el paradigma no relacional supera a su contra parte en todos los aspectos (rendimiento) , pero a pesar de esto sería apresurado decir que uno prevalece sobre el otro, tanto MySQL con sus estructuras de datos inflexibles, verificaciones de consistencia e integridad, así como MongoDB son dos enfoques que abarcan características distintas y que juegan un papel crucial a la hora de elegir un sistema gestor de base de datos para un proyecto determinado.



**Palabras Claves:** Base de datos, NoSQL, Base de datos Modelo Relacional, Performance Test, Big Data, Test de Estrés.

## ABSTRACT

With the development and expansion of the internet, cloud computing and the emergence of applications such as Facebook, YouTube and Twitter, there is a need to store and process large volumes of data. As a result of these events, the traditional database model has faced several special games in high concurrency applications with handling of large volumes of information, a scenario under which relational databases are required without being appropriate.

It is in this context that the non-relational database model (NoSQL) emerges, this new approach emerges as a solution to the constant requirements of processing and analyzing huge amounts of data. Both paradigms, although they have been converted, manage data in a neat way. That is why it becomes interesting the question of; What is the best answer? Why is NoSQL much faster than SQL? And why is it so efficient? This document aims to present the process that involves the use of performance tests to both concepts and then perform an analysis of the data obtained.

This is a brief description of both paradigms (SQL and NoSQL), describes their properties and makes a comparison of their properties at the conceptual level.

Once clarified the key concepts that intervened in the research process and after having developed the different tests and analysis of the data obtained, we can say that; Unquestionably the non-relational paradigm outperforms its part in all aspects (performance), but despite this it is affirmative that it prevails over the other, both MySQL with its inflexible data structures, checks for consistency and integrity, as well as MongoDB are two approaches that encompass different characteristics and that play a crucial role when choosing a database manager system for a given project.



**Keywords:** Database, NoSQL Database, Relational Database, Performance Testing Data Base, Big Data, Stress Test.

# INTRODUCCIÓN

Desde el principio de la historio el ser humano se ha esforzado por realizar una buena gestión de la información empleando toda clase de registros, desde administrar cosechas o hasta registrar censos. A lo largo del tiempo este proceso ha evolucionado hasta los complejos sistemas de gestión de información con los que contamos hoy en día.

La gestión de la información es una actividad que cada día se vuelve una acción más compleja y hace que las personas implicadas en esta área tengan que recurrir a herramientas tecnológicas con un enfoque diferente al tradicional en busca de herramientas que se adapten a las necesidades de la actualidad, esta nueva incursión se produce debido al estancamiento que han sufrido los gestores de bases de datos relacionales, y es que estos al no hacer frente a las necesidades de la actualidad y al no evolucionar en nuevo enfoque que haga cara a estas carencias, consecuentemente se tornan ineficientes en ciertas circunstancias.

El modelo de datos relacional, con sus tablas, relaciones, vistas, filas y columnas, ha sido muy útil y aceptado… pero presentan problemas, entre ellos, la relación con los lenguajes orientados a objetos y la dificultad para la adaptación a datos no estructurados (Romero, Sanabria, & Cuervo, 2012a).

Hoy en día se observa un crecimiento exponencial en cuanto a volúmenes de datos se trata, algo que denominamos Big Data, "Hoy los volúmenes de información crecen a un ritmo sin precedentes" (Romero et al., 2012a).

A raíz de este fenómeno diferentes empresas como Twitter, Facebook o Google se han visto en la necesidad no solo de almacenar estas grandes cantidades de información sino hacer uso eficiente de ella, es decir, generar análisis y consultas en tiempos aceptables para sacar ventaja a esta información, es aquí donde intervienen las bases de datos NoSQL las cuales en los últimos años han generado un gran interés.

NoSQL es un nuevo modelo o enfoque que "define un conjunto de tecnologías que se apartan de lo planteado por los gestores de bases relacionales, como por ejemplo, la interfaz de consulta para los usuarios en NoSQL no es soportada sobre SQL" (2012a).

# OBJETIVOS

## General

Desarrollar un estudio del rendimiento entre las bases de datos no relacionales (NoSQL) y las bases de datos relacionales (SQL) mediante la aplicación de pruebas de rendimiento.

## Específicos

- Describir las propiedades básicas de cada gestor
- Describir los parámetros a evaluar en ambos sistemas
- Plantear la metodología a emplear para la realización de las pruebas
- Desarrollar la aplicación para ejecutar las pruebas
- Analizar los resultados para ambos modelos con el propósito de realizar una breve comparación

# INFORME DE CASO

## Presentación de caso

El presente trabajo de investigación tiene como propósito poner a prueba el rendimiento de los sistemas de bases de datos relacionales frente a los sistemas no-relacionales (NoSQL), para ello se aplicarán pruebas de rendimiento a cada uno de los actores implicados, pruebas en base a las operaciones básicas que ejecuta cada gestor (CRUD).

Posteriormente se presentará un análisis comparativo de ambas herramientas con base en los resultados obtenidos y se describirán las ventajas de su posible implementación frente a diferentes situaciones o necesidades.

Para desarrollar la investigación se emplean los gestores MySQL y MongoDB, relacional y no relacional respectivamente. Selección que se estableció por cumplir con los parámetros de búsqueda establecidos en la investigación.

Para el desarrollo de las pruebas será necesario implementar un ambiente de prueba, para ello se utilizará un computador de escritorio de características Intel CORE i5 (5400U) 3.2 MHz con 8GB de memoria RAM con sistema operativo Windows 10 de 64 Bits, este computador será el pc anfitrión sobre el cual se instalarán dos máquinas virtuales de Ubuntu server (instalación mínima), las cuales desplegar los servicios para cada gestor.

# Marco Teórico

## Base de datos

### Orígenes

> Los orígenes de las bases de datos se remontan a la Antigüedad donde ya existían bibliotecas y toda clase de registros. Además, también se utilizaban para recoger información sobre las cosechas y censos. Sin embargo, su búsqueda era lenta y poco eficaz y no se contaba con la ayuda de máquinas que pudiesen reemplazar el trabajo manual. Posteriormente, el uso de las bases de datos se desarrolló a partir de las necesidades de almacenar grandes cantidades de información o datos. Sobre todo, desde la aparición de las primeras computadoras, el concepto de bases de datos ha estado siempre ligado a la informática.
>
> (Juan Anzaldo, 2005)

### Definición

Se denomina base de datos al "Conjunto de datos organizado de tal modo que permita obtener con rapidez diversos tipos de información"(ASALE, s. f.).

En el contexto de la informática una base de datos "es un conjunto de datos almacenados en memoria externa que están organizados mediante una estructura de datos normalmente predefinida para su posterior organización o consulta" (Marqués, 2009, p. 10).

En otras palabras, una base de datos no es más que un conjunto de información (conjunto de datos) relacionada que se encuentra agrupada o estructurada de manera ordenada.

### Historia

Antes de que existieran los sistemas gestores de bases de datos se empleaban sistemas de ficheros, los cuales "Están formados por un grupo de fichas de datos y un conjunto de programas de aplicación que permitían a los usuarios finales interactuar sobre esta colección de fichas" (Marqués, 2009, p. 10).

Estos sistemas de ficheros "surgieron producto de informatizar el manejo de los archivadores manuales para proporcionar un acceso más eficiente a los datos almacenados en los mismos" (Marqués, 2009, p. 10).

A pesar de la versatilidad que brindaban, estos sistemas de ficheros contaban con una serie de desventajas, entre ellas:

1. **Redundancia e inconsistencia de los datos. -** Redundancia significa tener el mismo dato guardado varias veces. Inconsistencia significa que hay contradicción en el contenido de un mismo dato, es decir, que un mismo dato tiene un valor en una parte de la memoria, mientras que en otra parte contiene otro valor diferente.

2. **Dificultad en el acceso a los datos. -** Era difícil que el usuario encontrara rápidamente un dato en especial.

3. **No existía el aislamiento de los datos. -** Debido a que los datos estaban dispersos en varios archivos y podían estar en diferentes formatos, era difícil escribir programas nuevos de aplicación para recuperar los datos apropiados.

4. **Problemas de integridad. -** Era complicado asegurarse que los valores almacenados satisficieran ciertos tipos de restricciones, por ejemplo, que tuvieran un valor mínimo y/o un valor máximo.

5. **Problemas de atomicidad. -** Era muy difícil asegurar que una vez que haya ocurrido alguna falla en el sistema y se ha detectado, los datos se restauraran al estado de consistencia que existía entes de la falla.

6. **Anomalías en el acceso concurrente. -** La cuestión de asegurar la consistencia de los datos se complica todavía más cuando se trata de sistemas en los que hay varios usuarios accediendo a un mismo archivo desde diferentes computadoras.

7. **Problemas de seguridad. -** No todos los usuarios de un sistema de información deberían poder acceder a todos los datos. En un sistema de archivos es muy difícil garantizar las restricciones de seguridad.

(María Del Carmen Gómez, 2012, p. 6,7)

No hay un momento concreto en el que los sistemas de ficheros hayan cesado y hayan dado comienzo los sistemas de bases de datos. De hecho, todavía existen sistemas de ficheros en uso.

Se dice que los sistemas de bases de datos tienen sus raíces en el proyecto estadounidense de mandar al hombre a la luna en los años sesenta, el proyecto Apolo. En aquella época, no había ningún sistema que permitiera gestionar la inmensa cantidad de información que requería el proyecto.

Fue la primera empresa encargada del proyecto NAA (North American Aviator) en desarrollar una aplicación que se basaba en el concepto de unir varias piezas

> pequeñas para formar una pieza más grande… fue así como la estructura resultante con forma de árbol se denominó estructura jerárquica.
>
> A mitad de los sesenta, General Electric desarrolló IDS (Integrated Data Store) … IDS era un nuevo tipo de sistema de bases de datos conocido como sistema de red… El sistema de red se desarrolló, en parte, para satisfacer la necesidad de representar relaciones entre datos más complejas que las que se podían modelar con los sistemas jerárquicos y, en parte, para imponer un estándar de bases de datos.
>
> (Marqués, 2009, p. 5)

Es así como los primeros gestores de base de dato, tanto el jerárquico como el de red se constituyeron como la primera generación de estos.

**Modelo relacional**

Posterior al modelo jerárquico y de red surgió un nuevo modelo de base de datos. Podría denominarse una segunda generación. El modelo relacional.

> En 1970, Edgar Frank Codd de los laboratorios de investigación de IBM, escribió un artículo presentando el modelo relacional. En este artículo presentaba también los inconvenientes de los sistemas previos, el jerárquico y el de red. Pasó casi una década hasta que se desarrollaron los primeros sistemas relacionales… esto condujo a dos grandes desarrollos como:
>
> - El desarrollo de un lenguaje de consultas estructurado denominado SQL, que se ha convertido en el lenguaje estándar de los sistemas relacionales.
>
> - La producción de varios SGBD relacionales durante los años ochenta, como DB2 y SLQ/DS, de IBM, y Oracle, de Oracle Corporation.
>
> Los SGBD relacionales constituyen la segunda generación de los SGBD. Sin embargo, el modelo relacional también tiene sus debilidades.
>
> (Marqués, 2009, p. 6)

**Sistema de gestión de base de datos**

Antes de indagar en la definición de lo que es un sistema de gestión de bases de datos (SGBD) vale la pena mencionar que no es lo mismo “base de datos” que “sistema de gestión de bases de datos”, aparentemente la diferencia es obvia, pero estos dos términos tienen a mezclarse y confundirse debido a que suele usarse el término “base de datos” para referirnos a los **SGBD** siendo estos por definición dos cosas totalmente distintas.

"Un sistema de gestión de la base de datos (SGBD) es una aplicación que permite a los usuarios definir, crear y mantener la base de datos, además de proporcionar un acceso controlado a la misma. Se denomina sistema de bases de datos al conjunto formado por la base de datos, el SGBD y los programas de aplicación que dan servicio a la empresa u organización" (Marqués, 2009, p. 3).

> Un SGBD es un conjunto de aplicaciones que nos permite definir, administrar y procesar una base de datos y sus aplicaciones "… es una herramienta de propósito general que permite crear bases de datos de cualquier tamaño y complejidad y con propósitos específicos distintos… El objetivo principal de un sistema de administración de bases de datos es proporcionar una forma de almacenar y recuperar la información de una base de datos de manera que sea tanto práctica como eficiente. Los SGBD se diseñan para gestionar grandes cantidades de información. La gestión de los datos implica tanto la definición de estructuras para almacenar la información como la provisión de mecanismos para la manipulación de la información. Además, los sistemas de bases de datos deben proporcionar la fiabilidad de la información almacenada, a pesar de las caídas del sistema o los intentos de acceso sin autorización. Si los datos van a ser compartidos entre varios usuarios, el sistema debe evitar posibles datos contradictorios.
>
> (María Del Carmen Gómez, 2012, lib. 10)

Normalmente, un Sistema Gestor de Bases de datos proporciona los siguientes servicios:

- Permite la definición de la base de datos mediante un lenguaje de definición de datos. Este lenguaje permite especificar la estructura y el tipo de los datos, así como las restricciones sobre los datos.
- Aprueba la inserción, actualización, eliminación y consulta de datos mediante un lenguaje de manejo de datos.
- Proporciona un acceso controlado a la base de datos mediante:
    - Un sistema de seguridad, de modo que los usuarios no autorizados no puedan acceder a la base de datos.
    - Un sistema de integridad que mantiene la integridad y la consistencia de los datos.
    - Un sistema de control de concurrencia que permite el acceso compartido a la base de datos.

- Un sistema de control de recuperación que restablece la base de datos después de que se produzca un fallo del hardware o del software.
- Un diccionario de datos o catálogo, accesible por el usuario, que contiene la descripción de los datos de la base de datos.

(Marqués, 2009, p. 3)

Los SGBD, a diferencia de los sistemas de ficheros (ancestros de los SGDB) que operan directamente sobre los datos, se encargan de la estructura física de los mismos y de su almacenamiento. Esto hace de los SGDB una gran herramienta de utilidad. No todos los SGBD presentan las mismas características mencionadas anteriormente.

"En general, los grandes SGBD multiusuario ofrecen todas las funciones que se acaban de citar e incluso más. Los sistemas modernos son conjuntos de programas extremadamente complejos y sofisticados, con millones de líneas de código y con una documentación consistente en varios volúmenes" (Marqués, 2009, p. 4).

**Modelo relacional**

"Un modelo de datos es un conjunto de conceptos que sirven para describir la estructura de una base de datos, es decir, los datos, las relaciones entre los datos y las restricciones que deben cumplirse sobre los datos … Cada SGBD soporta un modelo lógico, siendo los más comunes el relacional, el de red y el jerárquico".

(Marqués, 2009, p. 14)

Las bases de datos relacionales constituyen uno de los principales objetos de estudio en la presente investigación, y es importante dar a conocer los puntos más importantes referentes a este.

> Las bases de datos relacionales fueron definidas por el matemático Codd en los años 70. La teoría relacional ha ido evolucionando a lo largo del tiempo, incluyendo cada vez nuevas características. Este modelo representa los datos y las relaciones entre los datos mediante una colección de tablas, cada una de las cuáles tiene un número de columnas con nombres únicos.
>
> En el modelo relacional se utiliza un grupo de tablas para representar los datos y las relaciones entre ellos. Cada tabla está compuesta por varias columnas, y cada columna tiene un nombre único.
>
> Cada tabla contiene registros de un tipo particular. Cada tipo de registro define un número fijo de campos o atributos. Las columnas de la tabla corresponden a los atributos del tipo de registro. El modelo relacional oculta detalles de

> implementación de bajo nivel a los desarrolladores de bases de datos y a los usuarios.
>
> (María Del Carmen Gómez, 2012, p. 19)

"El modelo relacional representa la segunda generación de los SGBD. En él, todos los datos están estructurados a nivel lógico como tablas formadas por filas y columnas, aunque a nivel físico pueden tener una estructura completamente distinta" (Marqués, 2009, p. 15).

Hasta ahora bien se ha descrito que las tablas o relaciones constituyen la estructura básica para la gestión de los datos dentro del modelo relacional. Para que el concepto permanezca claro a continuación se define de manera conceptual cada uno de los elementos que conforman una tabla.

"El modelo relacional se basa en el concepto matemático de relación, que gráficamente se representa mediante una tabla. Codd, que era un experto matemático, utilizó una terminología perteneciente a las matemáticas, en concreto de la teoría de conjuntos y de la lógica de predicados" (Marqués, 2009, p. 16).

**Una relación** es el conjunto de filas y columnas (las tablas constituyen la estructura base del modelo relacional). Es el SGBD el encargado de recibir una base de datos como un conjunto de tablas, aunque esto sea real desde la perspectiva de una persona, esto no significa que a nivel físico la base de datos en cuestión esté representada de la misma manera, el SGBD puede implementar diferentes estructuras físicas para implementar una base de datos.

**Un atributo** es el nombre que se le da a una columna dentro de una tabla, este estará destinado a almacenar un dato y su naturaleza (tipo o clase) dependerá del formato que se le haya impuesto (todos los atributos almacenan un tipo de dato predefinido como cadenas, fechas, numéricos, etc.). Según Marqués "Una relación se representa gráficamente como una tabla bidimensional en la que las filas corresponden a registros individuales y las columnas corresponden a los campos o atributos de esos registros" (Marqués, 2009, p. 16).

**Una tupla o registro** corresponde simplemente al registro en una tabla. Ej. Se tiene una tabla para clientes llamada "clientesEJ" la cual cuenta con tres atributos "Nombre, Edad

y Teléfono", el registro de cada uno de los empleados en la tabla vendría siendo lo que denominamos tupla.

**La cardinalidad** es el número de registros o tuplas que existen en una tabla. Ej. De haber cinco empleados registrados en la tabla "clientesEJ" su cardinalidad sería igual a cinco. Se dice que una tabla o relación está **Normalizada** cuando en cada uno de sus campos, es decir cada intersección entre filas y columnas, existe un solo valor.

CLIENTES

| codcli | nombre | dirección | codpostal | codpue |
|---|---|---|---|---|
| 333 | Sos Carretero, Jesús | Mosen Compte, 14 | 12964 | 53596 |
| 336 | Miguel Archilés, Ramón | Bernardo Mundina, 132-5 | 12652 | 07766 |
| 342 | Pinel Huerta, Vicente | Francisco Sempere, 37-10 | 12112 | 07766 |
| 345 | López Botella, Mauro | Avenida del Puerto, 20-1 | 12439 | 12309 |
| 348 | Palau Martínez, Jorge | Raval de Sant Josep, 97-2 | 12401 | 12309 |
| 354 | Murría Vinaiza, José | Ciudadela, 90-18 | 12990 | 12309 |
| 357 | Huguet Peris, Juan Ángel | Calle Mestre Rodrigo, 7 | 12930 | 12309 |

**Ilustración 1:** Ejemplo Relación o Tabla (Marqués, 2009, p. 17)
[**Fuente:** https://goo.gl/rJmP6d]

| | | |
|---|---|---|
| Relación | Tabla | Fichero |
| Tupla | Fila | Registro |
| Atributo | Columna | Campo |
| Grado | No. de columnas | No. de Campos |
| Cardinalidad | No. de filas | No. de registros |

**Ilustración 2:** Terminología (María Del Carmen Gómez, 2012, p. 20)
[**Fuente:** https://goo.gl/hVcYAD]

## ACID

Es un compendio de requerimientos o propiedades que caracterizan a un determinado SGBD, esta propiedad es la que nos permite asegurar la integridad en las transacciones por parte del gestor en cuestión (Salazar C. 2014, P. 18).

- **Atomicidad:** Todas las operaciones de una transacción deben ser realizadas, si una de estas falla toda la transacción falla.
- **Consistencia:** Esta característica indica que toda transacción deberá llevar la base de datos de un estado valido a otro igualmente valido
- **Aislamiento:** Esta propiedad permite en las transacciones concurrentes cada una se comporte como la única transacción en ejecución en el sistema a pesar de haber otras transacciones en ejecución.

- **Durabilidad:** Esta propiedad garantiza la ejecución de la transacción a pesar de que algo falle durante el proceso, Ej. Si falla la alimentación eléctrica del servidor esta propiedad es la que permite que la transacción se complete cuando el servidor se restablezca.


**Ilustración 3**: A. C. I. D.
[**Fuente:** https://goo.gl/evqLUE]

ACID es un conjunto de propiedades que se aplican cuando se realizan modificaciones en una base de datos. Una transacción es un conjunto de operaciones que están relacionadas, estas se emplean para lograr garantizar ACID.

**Atomicidad** significa que se puede garantizar que todas o ninguna de las instrucciones se ejecuten. Se pueden realizar complejas operaciones considerándolas como un todo, ningún error, bloqueo o caída de energía permitirá que solo se ejecuten una parte de las instrucciones relacionadas.

**Consistencia** significa que los datos serán consistentes, es decir, ninguna de las restricciones que tenemos en relación con los datos será violada.

**Aislamiento** Ninguna transacción podrá leer datos que están siendo empleados por otra hasta que esta termine, si dos transacciones se están ejecutando concurrencialmente cada una vera la base de datos como si se estuvieran ejecutando secuencialmente y si alguna transacción necesita leer datos que otra transacción está escribiendo, esta deberá esperar a que la otra termine.

**Durabilidad** significa que una vez que se completa una transacción, se garantiza que todos los cambios se hayan registrado en un medio duradero (como un disco duro) y que también se registre el hecho de que la transacción se completó.

**Teorema de CAP**

El teorema de CAP es una propuesta planteada por Eric Brewer en el año 2000 que dice que ningún sistema es capaz de mantener consistencia, tolerancia a fallos y disponibilidad perfecta en un entorno distribuido (Espinoza & Fernanda, s. f.).

- **Consistencia:** Es decir que todos los nodos obtengan la misma información.
- **Disponibilidad:** Cada petición que se haga al sistema debe recibir una confirmación si se ha resuelto o no la petición.
- **Tolerancia a fallos:** Es decir el sistema continuará funcionando a pesar de la falla de alguno de los nodos.

Es importante considerar este concepto ya que nos permite comprender que en los sistemas distribuidos de bases de datos es necesario sacrificar al menos uno de estos parámetros para ganar en velocidad.

Con base en esto se puede decir que un sistema puede hacer una combinación de estas propiedades en función de sus requerimientos:

- **CP:** Asegura la consistencia, aunque se pierda comunicación entre nodos esto en función de la disponibilidad sacrificada.
- **AP:** Asegura una buena disponibilidad a pesar de la pérdida de algún nodo, pero pierde en consistencia.
- **CA:** El sistema responderá correctamente y mantendrá la consistencia de datos, pero no tolerará la pérdida de algún nodo.

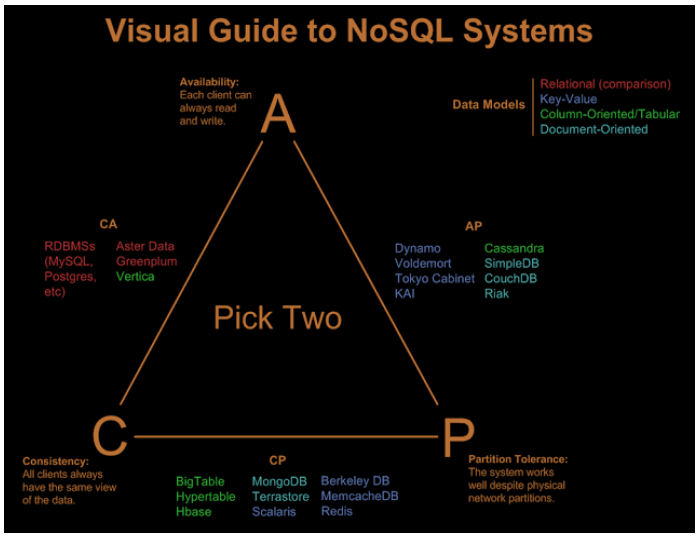


**Ilustración 4:** Teorema de CAP
[**Fuente:** https://goo.gl/UHWU6m]

**Base de datos modelo no relacional (NoSQL)**

El modelo no relacional es un paradigma de gestión de bases de datos que difiere del modelo clásico relacional en varios aspectos transcendentes, siendo uno de estos la inexistencia de un lenguaje de consulta.

> Las bases de datos NoSQL son sistemas de almacenamiento de información que no cumplen con el esquema entidad-relación. Mientras que las tradicionales bases de datos relacionales basan su funcionamiento en tablas, joins y transacciones. Las bases de datos NoSQL no imponen una estructura de datos en forma de tablas y relaciones entre ellas, sino que proveen un esquema mucho más flexible.
>
> (Martín, Chávez, Rodríguez, Valenzuela, & Murazzo, 2013a, p. 1)

Este sistema ofrece flexibilidad en la forma de almacenar los datos, es decir, no emplea estructuras fijas como tablas en el modelo relacional, normalmente no emplea operaciones tipo JOIN y no garantiza la propiedad ACID (atomicidad, consistencia, aislamiento y durabilidad). Regularmente tienen un buen desempeño de escalado horizontal.

Las bases de datos NoSQL surgen a raíz del gran incremento del volumen de la datos o información, fenómeno que se le atribuye a la llegada de la web 2.0, hasta antes de este suceso solo eran un puñado de empresas las que poseían portales para subir información a la red, luego con la aparición de aplicaciones como Twitter, Facebook o YouTube muchas más personas tuvieron acceso a subir contenido a la red provocando así un incremento exponencial en los datos.

"Las bases de datos NoSQL surgen como una solución a los constantes requerimientos de procesamiento y análisis a gran escala de enormes cantidades de datos, y para los cuales los sistemas tradicionales de base de datos son insuficientes" (Martín et al., 2013a, p. 2).

**Ventajas:**

- **Por lo general emplean pocos recursos:** Estos a diferencia de los gestores SQL pueden montarse y funcionar perfectamente sobre hardware de pocos recursos.
- **Escalabilidad Horizontal:** Agregar nodos al sistema mejora su rendimiento debido a la propiedad que posee de escalamiento horizontal.
- **Manejo de grandes cantidades de datos:** Los gestores NoSQL prácticamente nacieron para cubrir esta necesidad, no es de extrañarse que se desarrollen mejor en este campo que los SQL, esto es posible gracias a que maneja una estructura distribuida.
- **Uso eficiente de los recursos:** aprovecha bien tecnologías como los discos sólidos, la memoria RAM y los sistemas distribuidos en general.
- **Esquema de datos flexible:** Esta propiedad nos otorga mayor libertar para modelar los datos, al no hacer uso de esquemas rígidos como las tablas en el caso de los gestores relacionales.
- **Modelo de concurrencia débil:** No implementa ACID, lo que quiere decir que reúne las características necesarias para que una serie de instrucciones sean consideradas una transacción.
- **Simplicidad en las consultas:** las consultas son más naturales por lo tanto se gana en simplicidad y eficiencia.

(Romero et al., 2012a, p. 6)

**Clasificación NoSQL**

Las bases de datos no relacionales suelen clasificarse según la forma en que almacenan su información en:

**A) Clave-Valor:** En este caso cada registro almacenado está estructurado por dos componentes, una clave y un valor.

La novedad con este tipo de bases de datos NoSQL es la posibilidad de tener los datos en un ambiente distribuido. Se recomienda su uso en casos donde se necesita velocidad en las consultas o se tienen muchos datos con estructura simple que requieren ser procesados una y otra vez y tienen valores cambiantes, por ejemplo, listas de los más vendidos, carritos de la compra, las preferencias-cliente, gestión de sesiones, rango de venta y catálogo de productos.

(Romero et al., 2012a, p. 27)

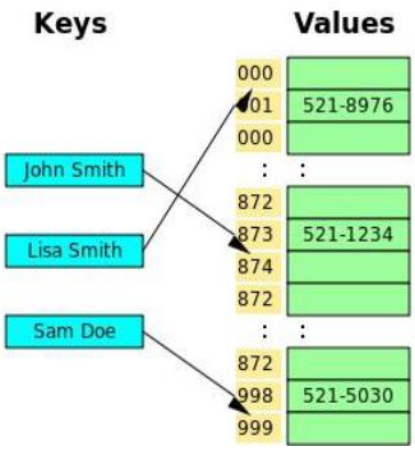


**Ilustración 5:** Tipo Clave/Valor

[**Fuente:** https://goo.gl/HkSv8X]

**B) Orientadas a Documentos:** El almacenamiento de los datos en esta categoría se realiza a través de documentos generalmente de tipo JSON (JavaScript Object Notation), estas ofrecen gran versatilidad y por ello se han implementado en gran cantidad de proyectos.

"Esta categoría NoSQL posee gran capacidad de manejar millones de lecturas simultáneas, puesto que ya tienen una lectura simple (un documento que contiene toda la información requerida). En un solo documentos se mantiene toda la información relacionada a una entidad" (Romero et al., 2012a, p. 27).

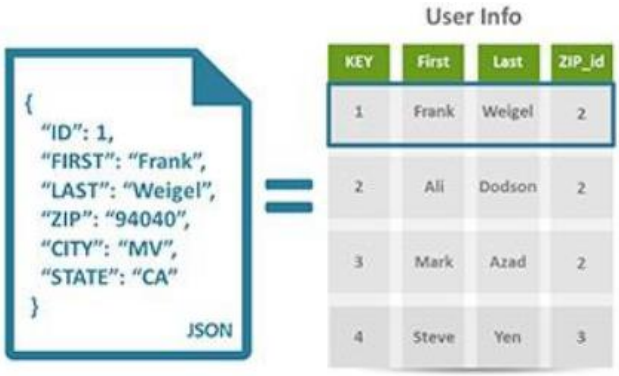


**Ilustración 6:** Orientada a Documentos
[**Fuente:** https://goo.gl/HkSv8X]

**C) Orientadas a Grafos:** Esta clasificación emplea grafos como estructura de almacenamiento, en donde los nodos vienen a ser las entidades u objetos y las aristas las relaciones entre estos. Los grafos son estructuras más flexibles y se integran más fácilmente en aplicaciones con estructuras orientadas a objetos.

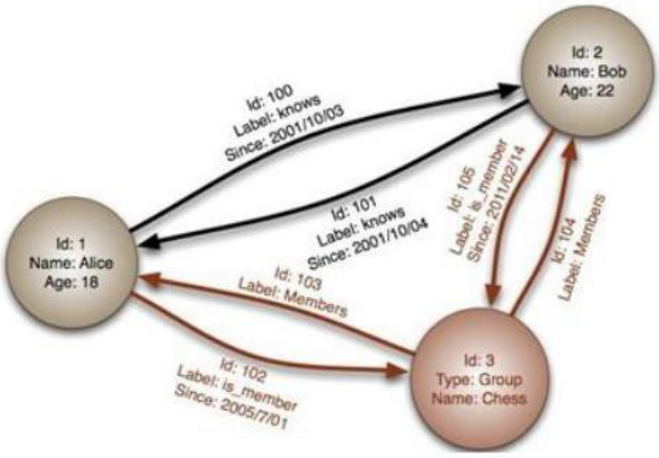


**Ilustración 7:** Orientada a Grafos
[**Fuente:** https://goo.gl/HkSv8X]

**Principales diferencias con el modelo relacional.**

- No emplea SQL (Structured Query Language) como lenguaje de consultas.
- No emplea estructuras inflexibles como las tablas para almacenar los datos.
- No suelen permitir operaciones JOIN.
- Posee una arquitectura distribuida.
- Normalmente escalan bien horizontalmente
- Buen desempeño en sistemas de escasos recursos

**Tabla 1**
*Diferencia entre ambos modelos*

| Modelo Relacional | Modelo No-relacional |
|---|---|
| **Escalabilidad Baja** | Escalabilidad Alta |
| **Rendimiento Bajo** | Alto rendimiento |
| **Alta fiabilidad** | Baja Fiabilidad |
| **Garantiza transacciones ACID** | No garantiza ACID en sus transacciones |
| **Costo de implementación elevado** | Costo de implementación moderado |
| **Alto nivel de seguridad** | Bajo nivel de seguridad |
| **Procesamiento de datos bajo** | Alto nivel de procesamiento de datos |

**Nota.** Descripción de las principales diferencias entre ambos enfoques.
**Fuente:** https://goo.gl/LqiFoh

## Diferencias de Sintaxis y Esquema

A continuación, se presenta una comparación de sintaxis para ambos SGBD. En la ilustración 17 se muestra las sentencias de creación de base de datos o colección, creación y eliminación de tablas o documentos (JSON) y creación de índices.

| Operación | SQL | MongoDB |
|---|---|---|
| Creación base de datos | **CREATE DATABASE** nombreBD; | **USE** nombreBD |
| Creación tabla (en SQL) / Colección (en MongoDB) | **CREATE TABLE** nombreTabla ( atributo1 tipo_de_dato restricciones <, atributo2...>); | **db.createCollection**("nombreColección") |
| Creación índice | **CREATE INDEX** nombreIndice **ON** nombreTabla (atributo1 <, atributo2... >); | **db.nombreColeccion.ensureIndex(** {"atributo": 1}) |
| Destrucción tabla (en SQL) / Colección (en MongoDB) | **DROP TABLE** nombreTabla; | **db.nombreColeccion.drop()** |

**Ilustración 8:** Base de dato / Colección
[**Fuente:** https://goo.gl/v5NcJm]

```
CREATE TABLE dpto (
 dep NUMBER(3) PRIMARY KEY,
 nom VARCHAR(20) NOT NULL);

CREATE TABLE empleado (
 id NUMBER(4) PRIMARY KEY,
 nombre VARCHAR(20) NOT NULL,
 edad VARCHAR(20) NOT NULL,
 grado NUMBER(3) NOT NULL,
 dep NUMBER(3) REFERENCES dpto);
```

```
db.createCollection("dpto")

db.createCollection("empleado")
```

**Ilustración 9:** Tabla /Colección
[**Fuente:** https://goo.gl/v5NcJm]

| SQL | MongoDB | Resultados en MongoDB |
|---|---|---|
| **INSERT INTO** dpto **VALUES**(10, 'Ventas');<br><br>**INSERT INTO** empleado **VALUES**(1, 'Juan', 23, 8, 10); | db.dpto.insert({_id: 10, nom: "Ventas"})<br><br>db.empleado.insert({_id: 1, nombre: "Juan", edad: 23, grado: 8, dep: 10}) | Los datos son insertados. |
| **UPDATE** empleado **SET** grado = 9 **WHERE** edad = 25; | db.empleado.update({edad:25}, {$set: {grado: 9}}) | {**"_id"**: 1, **"nombre"**: "Juan", **"edad"**: 23, **"grado"**: 8, **"dep"**: 10}<br>{**"_id"**: 2, **"nombre"**: "Pablo", **"edad"**: 25, **"grado"**:9, **"dep"**: 10}<br>{**"_id"**: 3, **"nombre"**: "Ana", **"edad"**: 11, **"grado"**: 6, **"dep"**: 20} |
| **DELETE FROM** empleado **WHERE** edad >= 18; | db.empleado.remove({edad: {$gt:18}}) | {**"_id"**: 3, **"nombre"**: "Ana", **"edad"**: 11, **"grado"**: 6, **"dep"**: 20} |

**Ilustración 10:** INSERT, UPDATE, DELETE
[**Fuente:** https://goo.gl/v5NcJm]

**Prueba de aplicaciones**

Parte indispensable dentro del desarrollo de una aplicación son las pruebas de rendimiento, estas tienen como propósito determinar cuán rápido ejecuta una tarea un sistema en condiciones particulares de trabajo. Estas también sirven para validar y verificar otros aspectos que influyen en la calidad de una aplicación, tales como la escalabilidad, fiabilidad y uso de los recursos.

> La prueba de software es una investigación realizada para proporcionar a las partes interesadas información sobre la calidad del producto o servicio de software sometido a prueba. Las pruebas de software también pueden proporcionar una visión objetiva e independiente del software para permitir que la empresa aprecie y comprenda los riesgos de la implementación del software. Las técnicas de prueba incluyen el proceso de ejecución de un programa o aplicación con la intención de encontrar errores de software (errores u otros defectos) y verificar que el producto de software sea apto para su uso.
>
> (Cem Kaner, 2016)

Las pruebas de rendimiento pertenecen a una rama de la ingeniería de pruebas, esta es una práctica informática que se enfoca en mejorar el rendimiento, el diseño y la arquitectura de una aplicación, incluso antes del esfuerzo inicial de la codificación.

Esta práctica se enfoca en diferentes propósitos tales como:

- Demostrar que un sistema o aplicación cumple con las expectativas de rendimiento de esta
- Comparar dos aplicaciones con el propósito de demostrar cual posee un mejor rendimiento
- Demostrar en que partes del sistema existe un mal funcionamiento

Para el desarrollo de esta actividad el experto emplea herramientas de software que monitorean y diagnostican que partes del sistema contribuyen al mal rendimiento, esta tarea también puede consistir en establecen un umbral en el sistema en donde los tiempos de respuestas de este son aceptables.

Es fundamental para alcanzar un buen nivel de rendimiento de un nuevo sistema, que los esfuerzos en estas pruebas comiencen en el inicio del proyecto de desarrollo y se amplíe durante su construcción. Cuanto más se tarde en detectar un defecto de rendimiento, mayor es el coste de la solución. Esto es cierto en el caso de las pruebas

funcionales, pero mucho más en las pruebas de rendimiento, debido a que su ámbito de aplicación es de principio a fin.

En las pruebas de rendimiento, a menudo es crucial (y con frecuencia difícil de conseguir) que las condiciones de prueba sean similares a las esperadas en el uso real. Esto es, sin embargo, casi imposible en la práctica. La razón es que los sistemas en producción tienen un carácter aleatorio de la carga de trabajo y aunque en las pruebas se intente dar lo mejor de sí para imitar el volumen de trabajo que pueda tener el entorno de producción, es imposible reproducir exactamente la variabilidad de ese trabajo, salvo en el sistema más simple.

(Molyneaux, 2009, p. 35)

Existen varios tipos de pruebas, a continuación, se listan las más conocidas:

**Pruebas de carga**

La prueba de carga es el proceso de poner demanda en un sistema y medir su respuesta. Las pruebas de carga generalmente se refieren a la práctica de modelar el uso esperado de una aplicación de software simulando múltiples usuarios que acceden al programa al mismo tiempo. Esta se perpetra por lo general para observar el desenvolvimiento de la aplicación bajo un entorno controlado y en respuesta a un número de peticiones dadas. Por lo general estas pruebas abarcan tanto al servidor de la aplicación como al servidor de base de datos.

**Pruebas de estrés**

Esta prueba básicamente consiste en ir doblando la carga en el servidor de tal manera que este llega a un punto en que se rompe. Este tipo de prueba nos ayuda a determinar cuan solida es la aplicación en los momentos de carga extrema, ayuda a los administradores a saber que decisiones tomar cuando el flujo de usuarios es mayor al esperado. Este tipo de prueba nos permite prever y garantizar la disponibilidad en la aplicación.

**Pruebas de estabilidad**

Por lo general el propósito de esta prueba es determinar si la aplicación puede mantenerse en producción durante una carga continua. Sirve para localizar fugas de memoria en la aplicación.

## Ámbito de estudio

El perímetro temario bajo el que se desarrolla la presente investigación es, programación, manejo de base de datos, testeo de aplicaciones, diseño de bases de datos y análisis de rendimiento.

Esta sección se define con el propósito de dar una perspectiva y rápida descripción de los gestores de base de datos más populares y disponibles en el medio, tanto SQL como NoSQL para posteriormente hacer una selección en base a los parámetros de selección de la presente investigación.

**Base de datos relacionales**

**MySQL**

Es un sistema de administración de bases de datos relacionales de código abierto con todas las funciones que originalmente fue construido por MySQL AB y actualmente es propiedad de Oracle Corporation. Almacena datos en tablas que están agrupadas en una base de datos, utiliza el Lenguaje de consulta estructurado (SQL) para acceder a datos y comandos como ‘SELECT’, ‘UPDATE’, ‘INSERT’ y ‘DELETE’ para administrarlo.

La información relacionada se puede almacenar en diferentes tablas, pero el uso de la operación JOIN le permite correlacionarla, realizar consultas en varias tablas y minimizar las posibilidades de duplicación de datos.

MySQL es compatible con casi todos los sistemas operativos, disponibles, Windows, Linux, Unix, Apple, FreeBSD y muchos otros. Admite varios motores de almacenamiento, como InnoDB (es el predeterminado), Federated, MyISAM, Memory, CSV, Archive, Blackhole y Merge.

> Es la base de datos de código abierto más popular del mundo. Es una base de datos integrada y segura para transacciones, compatible con ACID con capacidades completas de compromiso, retrotracción, recuperación de fallas y bloqueo a nivel de fila. Ofrece facilidad de uso, escalabilidad y alto rendimiento, así como un conjunto completo de controladores de bases de datos y herramientas visuales para ayudar a los desarrolladores y administradores de bases de datos a crear y administrar.
>
> («MySQL», 2018)

MySQL es desarrollado, distribuido y respaldado por Oracle. La base de datos MySQL proporciona las siguientes características:

- Alto rendimiento y escalabilidad para satisfacer las demandas de usuarios y cargas de datos en crecimiento exponencial.
- Clústeres de replicación autor regenerativos para mejorar la escalabilidad, el rendimiento y la disponibilidad.
- Cambio de esquema en línea para cumplir con los cambiantes requisitos comerciales.
- Esquema de rendimiento para supervisar el rendimiento a nivel de usuario y aplicación y el consumo de recursos.
- SQL y NoSQL Access para realizar consultas complejas y operaciones simples y rápidas de calve valor.
- Independencia de plataforma que le brinda la flexibilidad para desarrollar e implementar en múltiples sistemas operativos.
- Big Data Interoperability utilizando MySQL como el almacén de datos operativos para Hadoop y Cassandra.

(Cárdenas, 2014)

**SQL Server**

Microsoft SQL Server es un sistema de administración de bases de datos relacionales, o RDBMS, que admite una amplia variedad de aplicaciones de procesamiento de transacciones, inteligencia empresarial y análisis en entornos corporativos de TI. Es una de las tres tecnologías de bases de datos líderes en el mercado.

> Al igual que otros RDBMS, Microsoft SQL Server está construido sobre SQL, un lenguaje de programación estandarizado que los administradores de bases de datos (DBA) y otros profesionales de TI usan para administrar bases de datos y consultar los datos que contienen. SQL Server está vinculado a Transact-SQL (T-SQL), una implementación de SQL de Microsoft que agrega un conjunto de extensiones de programación propietarias al lenguaje estándar.
>
> («Microsoft SQL Server», 2018)

El código original de SQL Server fue desarrollado en la década de 1980 por la antigua Sybase Inc., que ahora es propiedad de SAP. Sybase inicialmente construyó el software para ejecutar en sistemas Unix y plataformas de miniordenadores. Microsoft y Ashton-Tate Corp., entonces el proveedor líder de bases de datos de PC, se unieron para producir la primera versión de lo que se convirtió en Microsoft SQL Server, diseñado para el sistema operativo OS / 2 y lanzado en 1989.

Componentes clave en Microsoft SQL server:

- Transact-SQL (T-SQL): implementación de SQL utilizada para programar comandos y consultas.
- Flujo de datos tabulares: protocolo que conecta a los clientes del servidor SQL con los servidores de bases de datos
- Motor de base de datos de SQL Server: gestiona el proceso y el almacenamiento de datos, además de la seguridad de los datos.
- Sistema operativo SQL Server (SQLOS): supervisa las funciones del sistema y de la base de datos de nivel inferior

> SQL Server 2016 también aumentó el soporte para big data analytics y otras aplicaciones analíticas avanzadas a través de SQL Server R Services, que permite que DBMS ejecute aplicaciones analíticas escritas en el lenguaje de programación de código abierto R , y PolyBase, una tecnología que permite a los usuarios de SQL Server acceder a datos almacenado en clústeres de Hadoop o almacenamiento de blobs de Azure para su análisis. Además, SQL Server 2016 fue la primera versión del DBMS que se ejecutó exclusivamente en servidores de 64 bits basados en microprocesadores x64. Y agregó la capacidad de ejecutar SQL Server en contenedores Docker , una tecnología de virtualización que aísla aplicaciones entre sí en un sistema operativo compartido.
>
> («Searchsqlserver», s. f.)

**PostgreSQL**

Es un potente sistema de base de datos relacional de condigo abierto que utiliza y amplía el lenguaje SQL combinado con muchas características que almacenan y escalan de forma segura las cargas de trabajo de datos más complicadas («PostgreSQL: About», s. f.-a).

Los orígenes de PostgreSQL se remontan a 1986 como parte del proyecto POSTGRES en la Universidad de California en Berkeley y cuenta con más de 30 años de desarrollo activo en la plataforma central.

PostgreSQL se ejecuta en todos los principales sistemas operativos, ha sido compatible con ACID desde 2001 y tiene complementos potentes, como el popular extensor de base de datos geoespaciales PostGIS.

> PostgreSQL intenta cumplir con el estándar SQL donde dicha conformidad no contradice las características tradicionales o podría llevar a decisiones arquitectónicas deficientes. Muchas de las funciones requeridas por el estándar SQL son compatibles, aunque a veces con una sintaxis o función ligeramente diferente. Se pueden esperar más movimientos hacia la conformidad a lo largo del tiempo. A partir del lanzamiento de la versión 10 en octubre de 2017, PostgreSQL cumple con al menos 160 de las 179 características obligatorias para SQL: conformidad con el estándar 2011, donde a partir de este momento, ninguna base de datos relacional cumple totalmente con este estándar.
>
> («PostgreSQL: About», s. f.-a)

Principales características de PostgreSQL:

**Tipos de datos**

- Primitivas: Entero, Numérico, Cadena, Booleano
- Estructurado: fecha / hora, matriz, rango, UUID
- Documento: JSON / JSONB, XML, clave-valor (Hstore)
- Geometría: punto, línea, círculo, polígono
- Personalizaciones: tipos compuestos y personalizados

**Integridad de los datos**

- ÚNICO, NO NULO
- Llaves primarias
- Llaves extranjeras
- Restricciones de exclusión
- Bloqueos explícitos, bloqueos de asesoramiento

**Fiabilidad, recuperación de desastres**

- Registro de escritura anticipada (WAL)
- Replicación: asíncrona, síncrona, lógica
- Punto-entiempo-recuperación (PITR), activos
- Tablespaces

**Seguridad**

- Autenticación: GSSAPI, SSPI, LDAP, SCRAM-SHA-256, Certificado y más
- Robusto sistema de control de acceso
- Columna y seguridad a nivel de fila

**Extensibilidad**

- Procedimientos almacenados
- Lenguajes de procedimiento: PL / PGSQL, Perl, Python (y muchos más)
- Contenedores de datos externos: conéctese a otras bases de datos o flujos con una interfaz SQL estándar
- Muchas extensiones que proporcionan funcionalidad adicional, incluido PostGIS

> PostgreSQL es altamente escalable tanto por la gran cantidad de datos que puede gestionar como por la cantidad de usuarios simultáneos que puede acomodar. Hay clústeres PostgreSQL activos en entornos de producción que administran muchos terabytes de datos y sistemas especializados que administran petabytes.
>
> (PostgreSQL: About, s. f.)

**Base de datos No-Relacionales**

**MongoDB**

> Es la base de datos NoSQL más conocida. En lugar de guardar los datos en tablas como se hace en las bases de datos relacionales, los guarda en documentos tipo JSON con un esquema dinámico (llamado formato BSON), haciendo que la integración de los datos en ciertas aplicaciones sea más fácil y rápida.
>
> («What Is MongoDB?», s. f.)

MongoDB es una base de datos orientada a documentos que admite nativamente el formato JSON.

Es extremadamente fácil de usar y operar, por lo que es muy popular entre los desarrolladores y no requiere un administrador de base de datos (DBA) para arrancar.

MongoDB es bastante robusto desde el punto de vista funcional y permite una replicación flexible para fragmentar entre nodos (Bit, s. f.).

Como tal, MongoDB es adecuado para escenarios con altas cargas de escritura y volúmenes Big Data. La replicación, la fragmentación y la conciencia del centro de datos le permiten permanecer disponible en entornos no confiables (por ejemplo, la nube).

Ofrece consultas dinámicas y agregados potentes, soporte de índice y funciones de map/reduct. Siendo una base de datos NoSQL, también es muy conveniente usarlo temprano en la fase de desarrollo, cuando un esquema no se ha establecido completamente.

**Redis**

Desarrollada en C y de código abierto, es utilizada por Craiglist y Stack Overflow. Redis es uno de los almacenes de datos más rápidos disponibles en la actualidad («Who's using Redis? – Redis», s. f.).

Es base de datos de código abierto, en memoria y NoSQL conocida por su velocidad y rendimiento, Redis se ha vuelto popular entre los desarrolladores y tiene una comunidad creciente y vibrante. Cuenta con varios tipos de datos que hacen que la implementación de varias funcionalidades y flujos sea extremadamente simple.

> Para entregar un rendimiento superior, Redis almacena todos los datos en la RAM. Dado que Redis es el rey en velocidad y rendimiento, la base de datos se utiliza mejor cuando el tiempo es un problema, incluida la administración de trabajos, las colas, el análisis en tiempo real, los mensajes, las capacidades sociales en la aplicación y la búsqueda geográfica.
>
> («Introduction to Redis – Redis», s. f.)

Redis resuelve de manera sencilla y eficiente problemas que no necesitan la complejidad de las bases de datos relacionales, como lo es en la mayoría de los casos de usos el almacén y gestión de estructuras de datos temporales, por lo cual Redis es mayoritariamente usado para incorporar soluciones sofisticadas de cache de datos o como backend de operaciones en línea en escenarios de alta demanda («SQLite vs MySQL vs PostgreSQL», s. f.).

Redis no pierde tiempo pensando en relaciones, restricciones ni tipos de datos, se enfoca en hacer eficientemente su trabajo que es establecer y recuperar (set y get) datos sobre las estructuras con las que cuenta («Introduction to Redis – Redis», s. f.).

> Redis está escrito en ANSI C y funciona en la mayoría de los sistemas POSIX como Linux, * BSD, OS X sin dependencias externas. Linux y OSX son los dos sistemas operativos en los Redis se desarrolla y más probados, y que recomiendan el uso de Linux para el despliegue. Redis puede funcionar en los sistemas derivados de Solaris como SmartOS, pero el apoyo es el mejor esfuerzo. No hay soporte oficial para Windows construye, pero Microsoft desarrolla y mantiene un puerto de Redis Win-64.
>
> (abraham, 2009)

**Principales características:**

- Transacciones: Permiten la ejecución de un grupo de comandos en una sola etapa, con dos importantes garantías.
- Pub: están programados para enviar sus mensajes a receptores específicos (usuarios registrados).
- Lua scripting: se utilizan para evaluar las secuencias de comandos utilizando el intérprete de Lua incorporado en Redis partir de la versión 2.6.0.
- Expiran claves segundos: Establezca un tiempo de espera en clave. Una vez transcurrido el tiempo de espera, se eliminará automáticamente la clave.

(«Introduction to Redis – Redis», s. f.)

**Cassandra**

Base de datos escrita en Java, de tipo Column Family, de código abierto creada por Facebook en 2008 actualmente cassandra se encuentra en producción en Facebook, pero aún se encuentra bajo fuerte desarrollo («Apache Cassandra», 2018).

> Altamente escalable, eventualmente consistente, distribuida y almacenamiento estructurado key-value. Agrupa las tecnologías de sistemas distribuidos de Dynamo y el modelo de datos BigTable de Google. Como Dynamo, es eventualmente consistente. Como BigTable, provee modelo de datos basado en ColumnFamily más enriquecido que los sistemas comunes key-value.

(Apache Cassandra, 2012).

Existen varias herramientas para la visualización y administración de los datos, la más destacada es OpsCenter que ofrece gestión y administración para los cluster, esta contiene una edición comunitaria y una empresarial que incluye características adicionales: alertas, balanceo automático de cargas, respaldos en vivo, entre otras. Adicional a esta se pueden encontrar otras: Cassandra Cluster Admin, Cassandra Explorer y Helenos, Lenguaje de consulta: CQL (Cassandra Query Language).

Principales características:

- **Mantenibilidad**: Al utilizar el lenguaje de consultas CQL, similar al estándar SQL facilita un poco el entendimiento e identificación para los desarrolladores que ya han utilizado otros motores. Características de la máquina: Requiere la última versión estable de Java 1.6 JRE o una más actualizada. En cuanto a sistema operativo y versión no se encuentra explícitamente, por lo que depende de la instalación del ambiente de java.
- **Instalación/Implementación:** En el sitio web oficial cassandra.apache.org se encuentra una sección dedicada a la descarga y otra a instalación y requerimientos necesarios.
- **Usabilidad:** Comandos ingresados por medio de cassandra CLI (interfaz de línea de comandos) o por medio de las herramientas gráficas y de gestión.
- **Versión:** La última versión estable es la 2.0.0, lanzada el 3 de septiembre de 2013. Soporte y solución de inquietudes: Contiene una sección llamada Wiki con información general y configuraciones; y otra nombrada FAQ (Frequently Asked Questions) con un listado de posibles preguntas.
- **Madurez:** Gran recorrido ya que fue inicialmente desarrollo interno de Facebook, y en 2008 salió a la luz como código abierto, se encuentra en versión estable. La primera versión para el público fue lanzada durante el segundo semestre de 2009.
- **Documentación:** Como se mencionó en los ítems anteriores, contiene varios apartados que pueden guiar al usuario en diferentes actividades. Para el caso del lenguaje CQL, contiene un repositorio con las consultas soportadas, sus notaciones y respectivos ejemplos.
- **Ventajas:** Orientada a columnas, tolerante a fallos, ya que replica los datos de forma automática a múltiples nodos; cuando un nodo falla pueden ser reemplazado sin ningún periodo de inactividad. Permite replicación a múltiples data centers; almacenamiento de los datos tipo ColumnFamily.
- **Limitaciones:** El valor de una columna no debe ser mayor a 2GB, el máximo número de columnas por fila es de 2 billones, la llave y los nombres de las columnas deben ser menores a 64 KB.

(Peña, s. f.)

**Herramientas de prueba**

Para desarrollar las pruebas de estrés y de carga será necesario emplear una herramienta software que se adapte a las necesidades de la investigación. En este apartado se presentan los candidatos que más se acercan a los parámetros de búsqueda.

**LoadUI**

Es un software de prueba de carga, dirigido principalmente a servicios web («LoadUI», 2018). LoadUI se ejecuta en varios entornos como Windows, Linux y Mac OS. LoadUI permite obtener una vista previa de los comportamientos de rendimiento de la API antes de lanzarlos a entornos de producción y cambiar las estadísticas de rendimiento a la izquierda («LoadUI», 2018).

> Había una versión de código abierto, pero se suspendió en julio de 2014. En 2015, LoadUI pasó a formar parte de ReadyAPI con el nombre LoadUI Pro. La mayoría de la interfaz de usuario, así como todos los eye-candies de JavaFX fueron eliminados. La interfaz de uso se simplificó y dividió en cinco partes visuales. Los proyectos antiguos ahora se pueden importar desde el archivo con algunas limitaciones.
>
> («About LoadUI | ReadyAPI Documentation», s. f.)

**Principales características:**

- Pruebas de carga basadas en la nube
- Prueba de carga paralela
- Supervisión del servidor
- Reutilizar pruebas funcionales existentes
- Generadores de carga distribuida
- Prueba de carga aislada
- Prueba de carga de punto final

> («API Performance Testing with LoadUI Pro | Easily Reuse SoapUI Tests», s. f.)


**Ilustración 11**: LoadUI
[**Fuente**: https://goo.gl/vpWGTH]

**HammerDB**

HammerDB es una aplicación de benchmarking de bases de datos de alto rendimiento de código abierto con modos de GUI y Command Line y compilaciones nativas de Linux y Windows para probar bases de datos que se ejecutan en cualquier sistema operativo («HammerDB», s. f.).

HammerDB es automatizado, multiproceso y extensible con soporte dinámico de scripting.


**Ilustración 12**: HammerDB
[**Fuente**: https://goo.gl/Q7C6GL]

> HammerDB es automatizado, multiproceso y extensible con soporte dinámico de scripting. HammerDB es compatible con Oracle, SQL Server, DB2, TimesTen, MySQL, MariaDB, PostgreSQL, Greenplum, Postgres Plus Advanced Server, Redis y Trafodion SQL en Hadoop. HammerDB incluye completas cargas de trabajo integradas basadas en los estándares de la industria TPC-C y TPC-H benchmarks, así como la captura y reproducción para la base de datos Oracle.
>
> («HammerDB», s. f.)

HammerDB es utilizado por todas las principales compañías de bases de datos y tecnología. Se ha descargado cientos de miles de veces a más de 180 países en el mundo. El uso de ejemplos se puede ver en la sección Benchmarks de compañías como Oracle, IBM, Intel, Dell / EMC HPE, Huawei, Lenovo y cientos más.

**LoadRunner**

Es una herramienta de prueba de software de Micro Focus. Se usa para probar aplicaciones, medir el comportamiento del sistema y el rendimiento bajo carga («LoadRunner», 2018).

LoadRunner puede simular miles de usuarios al mismo tiempo utilizando software de aplicación, grabando y luego analizando el rendimiento de los componentes clave de la aplicación.

> LoadRunner simula la actividad del usuario generando mensajes entre los componentes de la aplicación o simulando interacciones con la interfaz del usuario, como pulsaciones de teclas o movimientos del mouse. Los mensajes y las interacciones que se generarán se almacenan en secuencias de comandos. LoadRunner puede generar las secuencias de comandos grabándolas, como el registro de solicitudes HTTP entre un navegador web de un cliente y el servidor web de una aplicación.

(«LoadRunner», 2018)

Hewlett Packard Enterprise adquirió LoadRunner como parte de su adquisición de Mercury Interactive en noviembre de 2006.

**Ilustración 13**: Load Runner
[**Fuente**: https://goo.gl/jhbdor]

Los componentes clave de LoadRunner son:

- **Load Generator** genera la carga contra la aplicación siguiendo guiones
- **VuGen** (Virtual User Generator) para generar y editar guiones
- **El controlador** controla, inicia y secuencia instancias de Load Generator, especificando qué script usar, durante cuánto tiempo, etc. Durante las ejecuciones, el controlador recibe datos de monitoreo en tiempo real y muestra el estado.
- **El proceso del agente** gestiona la conexión entre las instancias de Controller y Load Generator.
- **El análisis** reúne registros de varios generadores de carga y formatea informes para la visualización de datos de resultados de ejecución y datos de supervisión.

(«Resources & Collateral - LoadRunner», s. f.)

### Apache JMeter

Apache JMeterTM es un software de código abierto escrito en Java, que fue desarrollado por primera vez por Stefano Mazzocchi de la Apache Software Foundation, diseñado para cargar el comportamiento funcional de la prueba y medir el rendimiento. Puede usar JMeter para analizar y medir el rendimiento de la aplicación web o la variedad de servicios. La prueba de rendimiento significa probar una aplicación web contra una carga pesada, tráfico de usuarios múltiples y concurrentes. JMeter originalmente se usa para probar aplicaciones web o aplicaciones FTP. Hoy en día, se usa para pruebas funcionales, pruebas de servidor de bases de datos, etc.


**Ilustración 14**: Apache JMeter
[**Fuente**: https://goo.gl/RKkf8s]

**Principales características:**

- Licencia de código abierto: JMeter es totalmente gratuito, permite a los desarrolladores usar el código fuente para el desarrollo.
- GUI amigable: JMeter es extremadamente fácil de usar y no lleva tiempo familiarizarse con él.
- Plataforma independiente: JMeter es una aplicación de escritorio Java 100% pura. Por lo tanto, puede ejecutarse en múltiples plataformas.
- Full multi-threading framework: JMeter permite el muestreo concurrente y simultáneo de diferentes funciones por un grupo de hilos separado
- Visualizar resultado de la prueba: el resultado de la prueba se puede visualizar en un formato diferente, como gráfico, tabla, árbol y archivo de registro
- Instalación sencilla: solo copie y ejecute el archivo *.bat para ejecutar JMeter. No necesita instalación
- Altamente extensible: puede escribir sus propias pruebas. JMeter también es compatible con complementos de visualización que le permiten extender sus pruebas
- Estrategia de prueba múltiple: JMeter admite muchas estrategias de prueba, como la prueba de carga, la prueba distribuida y la prueba funcional.
- Simulación: JMeter puede simular múltiples usuarios con subprocesos concurrentes, crear una gran carga contra la aplicación web bajo prueba.
- Soporte multiprotocolo: JMeter no solo es compatible con las pruebas de aplicaciones web, sino que también evalúa el rendimiento del servidor de la base de datos. Todos los protocolos básicos como HTTP, JDBC, LDAP, SOAP, JMS y FTP son compatibles con JMeter.
- Grabar y reproducir: registre la actividad del usuario en el navegador y simúlelos en la aplicación web utilizando JMeter
- Prueba de script: JMeter se puede integrar con Bean Shell & Selenium para realizar pruebas automatizadas.

(Apache JMeter - Apache JMeter ™, s. f.)

**Tabla 2**
*Criterio de selección de herramientas de prueba*

| Herramientas de prueba | | | | |
|---|---|---|---|---|
| **Herramienta** | **LoadUI** | **HammerDB** | **LoadRunner** | **Apache JMeter** |
| **Criterio** | | | | |
| Es compatible con ambos gestores | No | No | No | Si |
| Es de código abierto | No | Si | Si | Si |
| Opera sobre Windows | Si | Si | Si | Si |
| Posee una versión estable | Si | Si | Si | Si |
| Está bien documentada | No | Si | Si | Si |

**Nota**. Cuadro comparativo en base a los criterios de selección preestablecidos
**Fuente**: Elaboración Propia

Al igual que en la elección de los actores implicados los parámetros para la selección de la herramienta de prueba serán en base al nivel de popularidad, la cantidad de documentación accesible, la compatibilidad con el sistema operativo, la gratuidad de esta y adicionalmente debe ser compatible con ambos gestores dado que por ejemplo existen herramientas como HammerDB, que me permite emplearla contra MySQL más contra MongoDB no. Aparte de que emplear herramientas diferentes para desarrollar este tipo de pruebas puede interferir en los resultados. Con base en lo anterior se concluye que Apache JMeter cuenta con las cualidades que más se acerca a los requerimientos y por consiguiente se ha seleccionado como la herramienta de prueba.

## Elección de base de datos

El criterio de selección entre los posibles candidatos se basa en varios aspectos que se listan a continuación, principalmente se considera el nivel de popularidad y la cantidad de documentación accesible. Criterio que prevalece para ambos paradigmas, tanto el relacional como el no relacional.

**Tabla 3**
*Selección de base de datos*

| Paradigma | | SQL | | | NoSQL | | |
|---|---|---|---|---|---|---|---|
| | **Base de Datos** | **MySQL** | **SQL Server** | **PostgreSQL** | **MongoDB** | **Cassandra** | **Redis** |
| | **Criterio** | | | | | | |
| **1** | ¿Tiene marco de trabajo gráfico? | Sí | Sí | Sí | Sí | Si | No |
| **2** | ¿Soporte por parte del Proveedor? | Sí | Sí | Sí | Sí | No | No |
| **3** | ¿Se puede integrar con otras aplicaciones? | Sí | Sí | Sí | Sí | Sí | Sí |
| **4** | ¿Tiene una versión estable de este año? | Sí | Sí | Sí | Sí | No | No |
| **5** | ¿Trabaja bajo sistema operativo Windows? | Sí | Sí | Sí | Sí | Sí | No |
| **6** | ¿Tiene herramientas que apoyen la administración de los datos? | Sí | Sí | Sí | Sí | Sí | Si |
| **7** | ¿Es de código abierto? | Sí | No | Sí | Sí | Sí | Sí |

**Nota.** Cuadro comparativo en función de varios criterios para la selección de los gestores involucrados en la investigación.
**Fuente:** https://goo.gl/LmbJba

Según los datos presentados en la tabla tres, queda claro que tanto mongoDB como MySQL son los candidatos que más se acercan a los parámetros de búsqueda que propone la presente investigación. Los datos presentados en la ilustración quince, demuestra que tanto MySQL como mongoDB son herramientas que cuentan con suficiente reconocimiento dentro de la comunidad de desarrolladores además de poseer una enorme cantidad de documentación accesible, por consiguiente, estas herramientas quedan selectas como los actores que intervendrán en esta investigación.


**Ilustración 15:** Most popular databases in 2018 according to StackOverflow survey
[**Fuente:** https://goo.gl/wr1yj9]

## Actores implicados

### MongoDB

Es una base de datos de tipo no relacional creada por "10gen" de código abierto. Posee un esquema libre para datos, es decir que el formato o propiedades de cada registro en la colección puede ser diferente o independiente de los demás. MongoDB es la base de datos NoSQL líder en el mercado.

> Organizaciones de todos los tamaños están usando MongoDB para crear nuevos tipos de aplicaciones, mejorar la experiencia del cliente, acelerar el tiempo de comercialización y reducir costes. Es una base de datos ágil que permite a los esquemas evolucionar rápidamente cuando las aplicaciones cambian, proporcionando siempre la funcionalidad que los desarrolladores esperan de las bases de datos tradicionales, tales como índices secundarios, un lenguaje completo de búsquedas y consistencia estricta.
>
> (Cárdenas, 2014, p. 35).


**Ilustración 16:** MongoDB
[**Fuente:** https://goo.gl/RKkf8s]

MongoDB es orientada a documentos, es decir, emplea documentos tipo JSON para almacenar sus registros. El cuerpo del documento JSON está encerrado entre llaves {}. Este está conformado por parejas de capos tipo clave/valor separadas por comas, donde valor puede ser un carácter, una cadena de caracteres, un arreglo de valores e incluso otro documento tipo JSON.

> **JSON** (Notación de Objetos de JavaScript) es un formato ligero de intercambio de datos. Leerlo y escribirlo es simple para humanos, mientras que para las máquinas es simple interpretarlo y generarlo. Está basado en un subconjunto del Lenguaje de Programación JavaScript, JSON es un formato de texto que es completamente independiente del lenguaje, pero utiliza convenciones que son ampliamente conocidos por los programadores de la familia de lenguajes C, incluyendo C, C++, Java y muchos otros. Estas propiedades hacen que JSON sea un lenguaje ideal para el intercambio de datos.
>
> (JSON, s. f.)

```
{
        "id": 1,
        "nombre": "Juan",
        "edad": 23,
        "grado": 8,
        "dep": 10
}
```


**Ilustración 17:** Ej. Documento tipo JSON
[**Fuente:** https://goo.gl/v5NcJm]

Un documento JSON es el equivalente a una tupla o registro en el modelo relacional, al conjunto de documentos se les denomina colección, lo que en un SGBD relacional sería una tabla y sus registros.

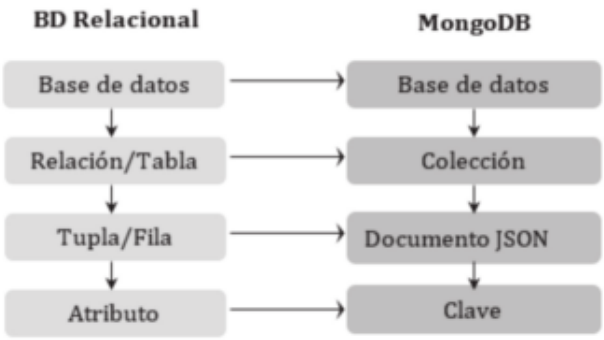


**Ilustración 18:** Paralelismo SQL/NoSQL
[**Fuente:** https://goo.gl/v5NcJm]

> MongoDB ha sido creado para brindar escalabilidad, rendimiento y gran disponibilidad, escalando de una implantación de servidor único a grandes arquitecturas complejas de centros multidatos. MongoDB brinda un elevado rendimiento, tanto para lectura como para escritura, potenciando la computación en memoria (in-memory). La replicación nativa de MongoDB y la tolerancia a fallos automática ofrece fiabilidad a nivel empresarial y flexibilidad operativa.
>
> (Cárdenas, 2014, p. 36)

**Principales características:**

**Tabla 4**
*Principales características de MongoDB*

| | |
|---|---|
| **Alta disponibilidad** | Perdura su funcionamiento a pesar de la caída de alguno de sus nodos |
| **Escalabilidad** | Este podría ir desde un servidor aislado a arquitecturas distribuidas de grandes clusters (Permite particionar, de manera transparente para el usuario, nuestra base de datos en tantas shards como tengamos disponible. Esto aumenta el rendimiento del procesado de los datos al ser cada una de ellas más pequeña que la original). |
| **Auto balanceado de carga** | El balanceador decide cuándo migrar los datos, y a qué Shard, para que estén uniformemente distribuidos entre todos los servidores del cluster. Cada shard aloja los datos correspondientes a un rango de la clave escogida para particionar nuestra colección. |
| **Replicación nativa** | Sincronización de datos entre servidores. |
| **Actualizaciones en producción** | Es capaz de actualizarse sin dejar de dar servicio. |
| **JSON** | Utiliza objetos JSON para guardar y transmitir la información. JSON es el estándar web hoy en día, por lo que supone una gran ventaja que web y base de datos hablen el mismo lenguaje. |

**Nota**. Se describen las principales características de MongoDB.
**Fuente:** https://goo.gl/LqiFoh

**Ventajas:**

- Orientada a documentos
- Alto rendimiento
- Escalado horizontal alta
- Fácil replicación
- Esquema de datos flexible
- Auto balanceo de carga
- Costo de implementación bajo

**Desventajas:**

- No emplea SQL
- No garantiza transacciones ACID
- Seguridad Baja
- Propenso a la inconsistencia de datos
- No emplea relaciones entre tablas (JOINS)
- Limitaciones de memoria

**Seguridad**

MongoDB al igual que muchos de los SGBD disponibles en el mercado cuenta con mecanismos de seguridad tales como autenticación, control de acceso y encriptación.

**Tabla 5**

*Mecanismos de seguridad MongoDB*

| **Autenticación** | Autorización | TLS/SSL | Empresarial |
|---|---|---|---|
| Autenticación | Control de acceso basado en rol | Transporte de cifrado | Autenticación Kerberos |
| SCRAM | Habilitar autenticación | Configuración mongo y mongod para TLS/SSL | Autenticación LDAP Proxy |
| x.509 | Gestión de usuarios y roles | TLS/SSL configuración para clientes | Cifrado en reposo |
| | | | Auditoria |

**Nota.** Mecanismos de seguridad empleados por mongoDB
**Fuente:** https://goo.gl/4HgYtX

Sin embargo, este no siempre fue tan robusto, mongoDB al igual que muchas herramientas software que están en continuo desarrollo, atraviesan por etapas que permiten ir mejorando su estructura en general, incluyendo aquí por supuesto la seguridad.

Según (Okman, Gal-Oz, Gonen, Gudes, & Abramov, 2011) mongoDB en sus inicios no fue concebida precisamente tomando en cuenta en gran manera los mecanismos de seguridad como los mencionados previamente, como resultado de esto mongoDB incurrió en serias carencias incorporadas en su diseño, a continuación, se mencionar varias de ellas.

**Archivos no encriptados**

Los archivos de datos de Mongo no están cifrados, y Mongo no proporciona un método para automáticamente Encriptar estos archivos. Esto significa que cualquier atacante con el acceso al sistema de archivos puede extraer directamente la información de los

archivos. Para mitigar esto, la aplicación debe encriptar explícitamente cualquier información sensible antes de escribirlo en la base de datos.

**Posibilidades de ataques de inyección**

El ataque por inyección es un método que aprovecha la mala filtración de los parámetros por parte de la aplicación permitiendo así al atacante inyectar código dentro del servido de base de datos.

> Mongo utiliza mucho JavaScript como lenguaje de scripting interno. La mayoría de los comandos internos disponibles para el desarrollador son en realidad cortos scripts de JavaScript. Incluso es posible almacenar funciones de JavaScript en la base de datos en db.system.js colección que están disponibles para los usuarios de la base de datos. Como JavaScript es un lenguaje interpretado, hay es un potencial para ataques de inyección.
>
> (2011, p. 2)

Un ejemplo de esta debilidad se puede apreciar en las siguientes peticiones hechas a la base de datos empleado una cláusula "where".

```
db.myCollection.find(
  { a : { $gt: 3 } }
);
db.myCollection.find(
  { $where: "this.a > 3" }
);
db.myCollection.find(
  "this.a > 3"
);
db.myCollection.find(
  { $where: function() {
    return this.a > 3;}
  }
);
```

**Ilustración 19**: Ejemplo ataque de inyección
[**Fuente**: https://goo.gl/2KGEpQ]

En el segundo y tercer enunciado, la cláusula "where" se pasó como una cadena que podría contener valores que eran concatenados directamente con valores pasados por el usuario. En la cuarta declaración, el objeto $where es una función de JavaScript que se evalúa por cada registro en la colección "myCollection". Esto no podría modificar la base de datos directamente si esta se ejecutase en un contexto de solo lectura, sin

embargo, si una aplicación utiliza este tipo de cláusula where sin depurar correctamente la entrada del usuario es posible que un ataque de este tipo se lleve a cabo.

**Autenticación parcial**

Mongo cuando se ejecuta en modo "Sharded" no soporta autenticación, sin embargo, cuando se ejecuta en modo "Standalone" o "Replica-set-mode" la autenticación puede estar habilitada en mongo. La principal diferencia entre estos dos es que en el modo "Replica-set-mode" además de contar con la autenticación de usuarios a la base de datos, cada réplica del servidor debe autenticarse con los otros servidores antes de poder acceder al cluster.

**Autenticación en modo shared**

Mongo cuando se ejecuta en modo Sharded, no es compatible con la autenticación, y por lo tanto no tiene soporte para la autorización. Si la autenticación ha sido habilitada, entonces mongoDB admite dos tipos de usuarios: de solo lectura y de lectura-escritura. Los usuarios de solo lectura pueden consultar todo en la base de datos en la que se encuentran definidos, mientras que los usuarios de lectura-escritura tienen acceso completo a todos los datos en la base de datos en la que están definidos. Cualquier usuario definido en la base de datos de administrador tiene plena acceso de lectura y escritura a todas las bases de datos definidas en cluster.

**Auditoria**

MongoDB no provee ninguna facilidad a la hora de ejecutar tareas de auditoria, por ejemplo, al crear una nueva colección de datos lo único que hace mongo es agregar una línea al archivo de log indicando que este ha sido creado, pero cuando esta colección empieza a alojar registro nada nuevo ocurre, es decir no se registra ningún tipo petición ya sea este de escritura o lectura.

**MySQL**

Es un sistema de gestión de bases de datos que pertenece a Oracle Corporación y que se apega al paradigma de base de datos relacional, esta fue liberada bajo licencia tipo GPL/comercial, es estimada una de las bases de datos de fuente abierta más popular en mundo.

**Principales características:**

- Escrita en C y C++
- Trabaja bajo diferentes plataformas
- Utiliza un diseño de servidor de varias capas con módulos independientes
- Diseñado para ser completamente multiproceso usando hilos de kernel, para usar fácilmente varias CPU si están disponibles
- Proporciona motores de almacenamiento transaccionales y no transaccionales
- Utiliza un sistema de asignación de memoria basado en hilos muy rápido
- Ejecuta JOINS muy rápidas usando una combinación optimizada de bucle anidado
- Implementa tablas hash en memoria, que se usan como tablas temporales
- Implementa funciones SQL usando una biblioteca de clases altamente optimizada que debe ser lo más rápido posible. Por lo general, no hay asignación de memoria después de la inicialización de la consulta.

(MySQL, s. f.)


**Ilustración 20**: MySQL
[**Fuente:** https://goo.gl/rjak3v]

MySQL está escrita en su mayor parte en ANSI C y C++, es ampliamente utilizada por muchos sitios populares, como Wikipedia, Google, Facebook, Twitter y YouTube.

En sus inicios, MySQL carecía de elementos esenciales en las bases de datos relacionales, tales como las transacciones e integridad referencial. A pesar de todo ello,

logro alcanzar un gran nivel de popularidad entre la comunidad de desarrolladores de sitios web, debido a su simplicidad.

Poco a poco los elementos de los que carecía MySQL fueron incorporados. Entre las características principales de este se puede destacar:

- Disponibilidad en gran cantidad de plataformas y sistemas
- Transacciones y claves foráneas
- Búsqueda e indexación de campos de texto
- Extenso subconjunto del lenguaje SQL

**Ventajas:**

- Soporta transacciones atómicas ACID
- Soporta operaciones JOIN
- Emplea sistema de contraseñas y privilegios
- Es de código abierto
- Relativamente fácil de emplear

**Desventajas:**

- Escalado pobre
- Estabilidad no muy consistente
- Su desarrollo no es impulsado por la comunidad
- Rendimiento tiende a degradarse

**Seguridad**

Hoy en día las bases de datos representan la columna vertebral de cualquier organización. Independientemente del tipo de organización, muy probablemente será necesario almacenar información sensible, tales como; usuarios, contraseñas, correos electrónicos, información financiera y mucho más. La seguridad e integridad de una base de datos resulta un aspecto muy esencial para cualquier organización ya que cualquier violación de las políticas de privacidad del usuario podría derivar en algún tipo de acción legal por parte de estos.

> La seguridad de la base de datos es más que importante: es una parte absolutamente esencial de cualquier empresa. Evita el compromiso o la pérdida de datos contenidos en la base de datos, un evento que podría tener graves consecuencias para cualquier empresa.
>
> (Gasystems, 2017)

Son varios los aspectos que juegan un papel crucial en la seguridad de una base de datos, sin embargo, los más relevantes son; integridad, confidencialidad y disponibilidad, estos constituyen los fundamentos de la seguridad de la información.

**Confidencialidad:** Es el aspecto más importante de la seguridad de la base de datos, generalmente se consigue atreves de la encriptación de los datos. la encantación debe llevarse a cabo tanto para los datos en tránsito como para los datos en reposo.

**Integridad:** La integridad es otro aspecto crucial de la seguridad de la base de datos, ya que garantiza que solo las personas correctas podrán ver la información privilegiada de la empresa. La integridad de una base de datos se aplica a través de un sistema de control de acceso de usuario que define los permisos para quién puede acceder a qué datos. Sin embargo, el aspecto de integridad va más allá de los simples permisos. Implementaciones de seguridad como protocolos de autenticación, políticas de contraseñas seguras y asegurar que las cuentas no utilizadas se bloquean o eliminan, lo que fortalece aún más la integridad de una base de datos.

**Disponibilidad:** La disponibilidad se relaciona con la necesidad de que las bases de datos estén disponibles y listas para su uso. Las bases de datos deben ser confiables para ser funcionales, lo que requiere que estén en funcionamiento siempre que la organización lo sea

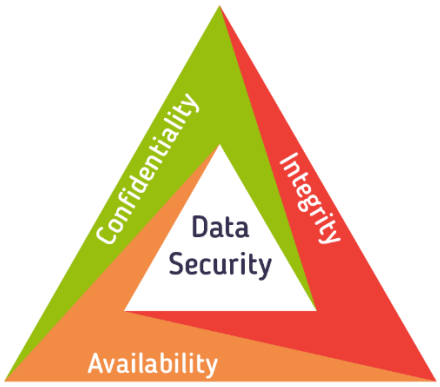


**Ilustración 21**: CIA
**[Fuente:** https://goo.gl/88fdeY]

Al hablar de la seguridad de cualquier instancia de un SGBD inevitablemente se abarca un amplio conjunto de temas que influyen en este aspecto, temas que van desde la seguridad física del servidor, correcta gestión de usuarios y contraseñas, restricción de roles de usuarios etc.

“Al pensar en la seguridad dentro de una instalación de MySQL, debe considerar una amplia gama de posibles temas y cómo afectan la seguridad de su servidor MySQL y las aplicaciones relacionadas”(«MySQL: Security», s. f.).

A continuación, se mencionar varios de los aspectos principales que toman parte en este tema.

- Factores generales que afectan la seguridad. Estos incluyen elegir buenas contraseñas, no otorgar privilegios innecesarios a los usuarios, garantizar la seguridad de las aplicaciones al evitar las inyecciones de SQL y la corrupción de datos, y otros.
- Seguridad de la instalación en sí. Los archivos de datos, los archivos de registro y todos los archivos de la aplicación deben estar protegidos para garantizar que no sean legibles ni modificables por terceros no autorizados.
- Control de acceso y seguridad dentro del sistema de la base de datos, incluidos los usuarios y las bases de datos a los que se otorga acceso a las bases de datos, vistas y programas almacenados en uso dentro de la base de datos.
- Las características que ofrecen los complementos relacionados con la seguridad (Plugin de autenticación, control de conexiones, validador de contraseña, etc.)
- Seguridad de red de MySQL y su sistema. La seguridad está relacionada con las concesiones para usuarios individuales, pero también puede restringir MySQL para que esté disponible solo localmente en el host del servidor MySQL, o para un conjunto limitado de otros hosts.
- Asegúrese de tener copias de seguridad adecuadas y adecuadas de sus archivos de base de datos, configuración y archivos de registro. Además, asegúrese de contar con una solución de recuperación y probar que puede recuperar con éxito la información de sus copias de seguridad.

(«MySQL: Security», s. f.)

# METODOLOGÍA

## Entorno de Prueba

A continuación, se presenta en detalle las características de cada prueba realizada, las herramientas software que intervinieron en este proceso y el entorno bajo el que se realizaron estas pruebas.

Para realización de las pruebas se empleó un computador de escritorio con procesador Intel CORE i5 (6540-U) 3.2 MHz S.O. Windows 10 y 8GB de memoria RAM, se implementaron dos servidores virtuales, uno para cada motor de base de datos, el sistema operativo empleado para desplegar los servicios fue Ubuntu Server versión 16.04 Para virtualizar cada servidor se empleó VirtualBox versión 5.2.0.

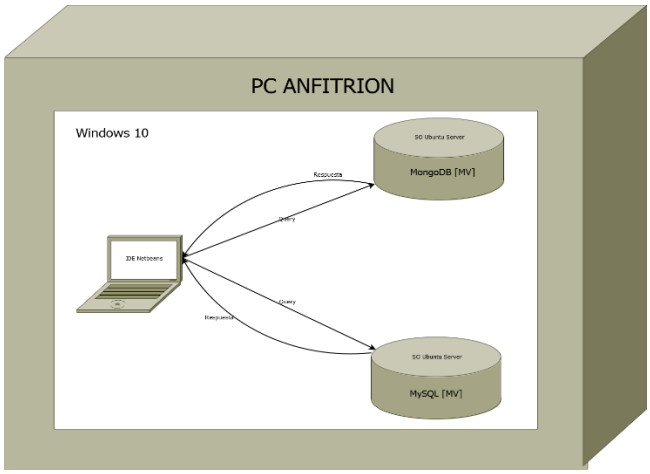


**Ilustración 22**: Entorno de Prueba
[**Fuente:** Elaboración Propia]

PHP fue el lenguaje selecto para realizar la conexión y las consultas a los servidores, también se empleó la librería phpFaker para la captura de los tiempos y la reproducción de los datos.

Es indispensable garantizar la igualdad de condiciones para ambos servidores ya que de otra manera los resultados serían irrelevantes, por ello se tomó en cuenta varias observaciones como:

- Que ambos servidores emplearan la misma cantidad de recursos
- Que cada parámetro (RAM, número de núcleos asignados, etc.) de configuración que influya en el rendimiento de las máquinas virtuales fuesen configurados de la misma manera

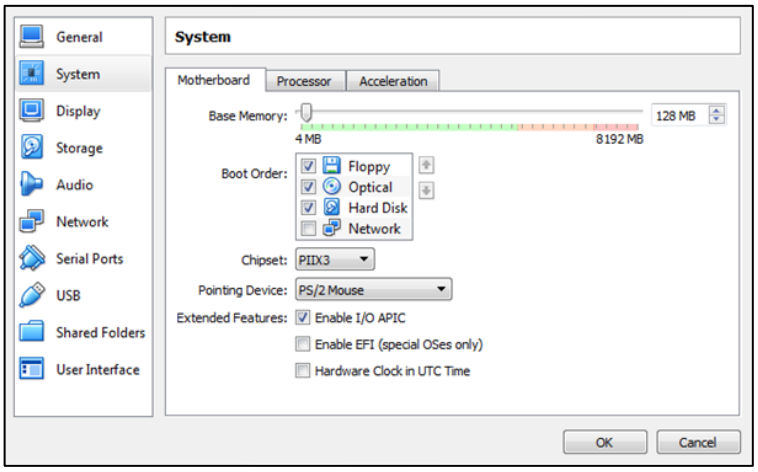


**Ilustración 23**: Propiedades Máquina Virtual
[**Fuente:** Elaboración Propia]

- Que la instalación de Ubuntu sea mínima para emplear exactamente los servicios necesarios y así ganar en rendimiento

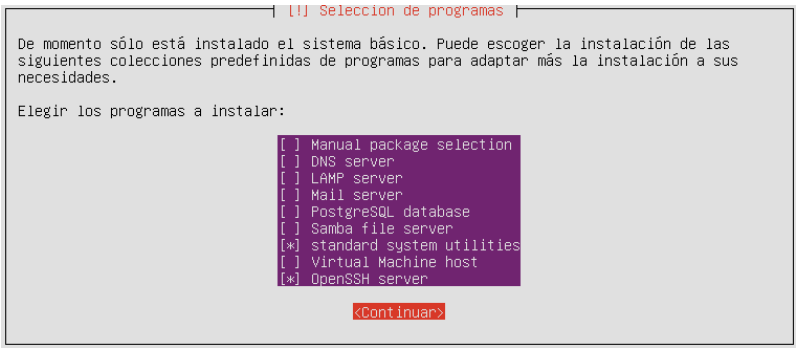


**Ilustración 24:** Ubuntu Instalación Mínima
[**Fuente:** Elaboración Propia]

El modelo de datos sobre el que se realizaron las pruebas se describe a continuación en el siguiente diagrama entidad relación.

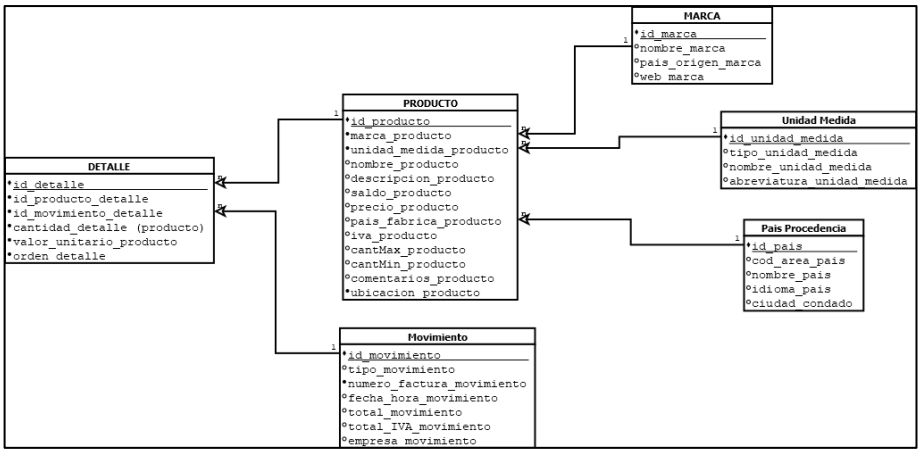


**Ilustración 25:** Entidad Relación (Elaboración Propia)
[**Fuente:** Elaboración Propia]

Este modelo de datos describe un sencillo sistema de control de inventario (compra y venta) de productos, a continuación, se hace una breve descripción de cada relación.

**Tabla 6**
*Descripción de campos y relaciones*

| Nombre | Descripción |
|---|---|
| **Marca** | Cada producto tiene una marca de procedencia, para cada marca podrán existir varios productos y cada producto tendrá una única marca de procedencia. |
| **Unidad Medida** | Cada producto poseerá una unidad de medida bien detallada, cada producto tendrá una única unidad de medida. |
| **País Procedencia** | Cada producto tendrá un país de procedencia. Varios productos podrían tener un único país de procedencia |
| **Producto** | Cada producto se asume es un objeto susceptible de comerciar |
| **Detalle** | Cada detalle encapsula la cantidad de producto que solicita un movimiento, cada movimiento podría tener varios detalles, varios detalles podrían pertenecer a un único movimiento. |
| **Movimiento** | Cada movimiento representa al menos la transacción de un detalle o mas |

**Nota.** Descripción de los campos y relaciones del diagrama entidad relación
**Fuente:** Elaboración propia

**Diseño de la aplicación**

El Lenguaje Unificado de modelado (UML) nos permite crear un modelo visual de la arquitectura diseño y comportamiento de la aplicación (Una imagen vale más que mil palabras). Básicamente esta representación semántica de nuestra aplicación correspondería analógicamente a los planos de un edificio. Esencialmente los diagramas UML describen los límites, comportamiento y los objetos que interactúan entre sí en el sistema.

El propósito de UML básicamente se compone en:

- Brindar las herramientas para el diseño, análisis e implementación de sistemas software a arquitectos, ingenieros y desarrolladores de software.
- Permite el progreso continuo del sistema a través de la representación visual de la interoperabilidad de objetos. Facilita el intercambio de información entre objetos y herramientas. Establece una definición formal de basado en un modelo de software

**Diagrama de componentes**

En el lenguaje de modelado unificado, un diagrama de componentes muestra cómo los componentes se conectan entre sí para formar componentes más grandes o sistemas de software. Ilustra las arquitecturas de los componentes de software y las dependencias entre ellos. Esos componentes de software, incluidos los componentes de tiempo de ejecución, los componentes ejecutables también los componentes del código fuente.

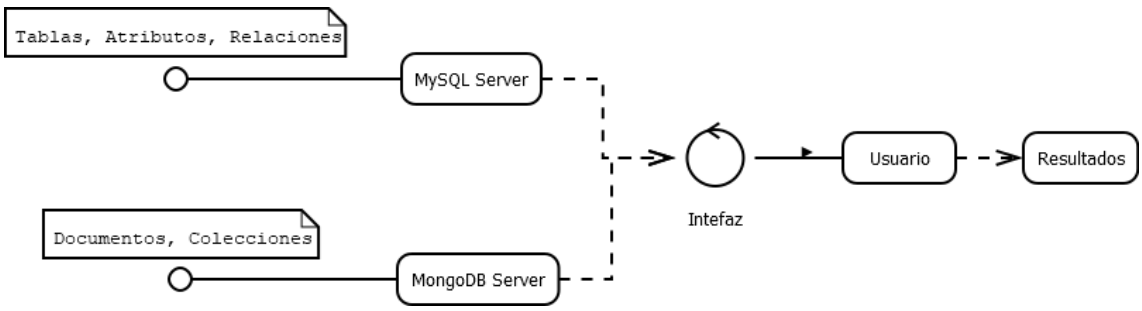


**Ilustración 26**: Diagrama de Componentes
[**Fuente:** Elaboración Propia]

## Diagrama de implementación

Un diagrama de implementación modela la estructura física de la aplicación, es decir su estructura y los componentes hardware. Este se encarga de mostrar donde y como operan los distintos componentes de una aplicación.

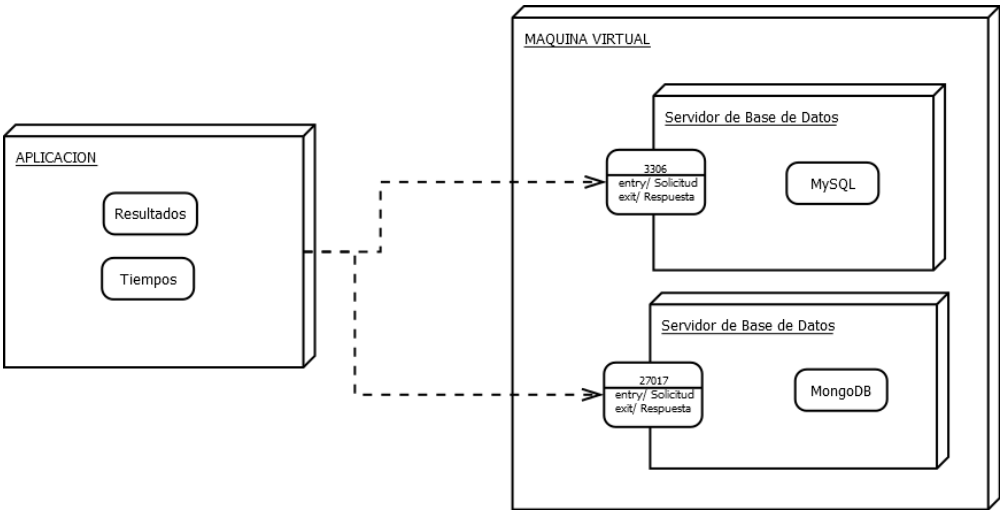


**Ilustración 27:** UML Implementación
[**Fuente**: Elaboración Propia]

## Diagrama de Caso de Uso

Un modelo de caso de uso describe los requisitos funcionales de un sistema en términos de casos de uso. Es un modelo de la funcionalidad prevista del sistema (casos de uso) y su entorno (actores). Los casos de uso le permiten relacionar lo que necesita de un sistema con la forma en que el sistema cumple con esas necesidades. Debido a que es un instrumento de planificación muy poderoso, el modelo de caso de uso es generalmente utilizado en todas las fases del ciclo de desarrollo por todos los miembros del equipo.

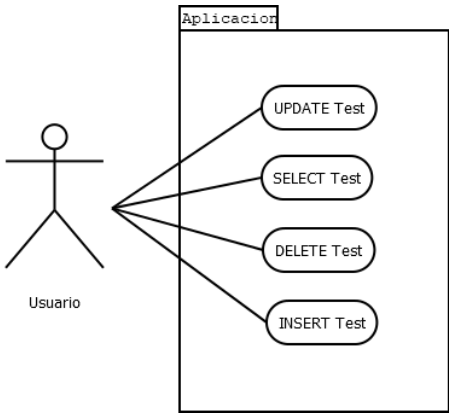


**Ilustración 28**: UML Caso de Uso
[**Fuente**: Elaboración Propia]

**Diagrama de Secuencia**

El Diagrama de Secuencia modela la colaboración de objetos basada en una secuencia de tiempo. Muestra cómo los objetos interactúan con otros en un escenario particular de un caso de uso. Con la capacidad avanzada de modelado visual, puede crear un diagrama de secuencia complejo en pocos clics. Además, alguna herramienta de modelado como Visual Paradigm puede generar un diagrama de secuencia a partir del flujo de eventos que ha definido en la descripción del caso de uso.

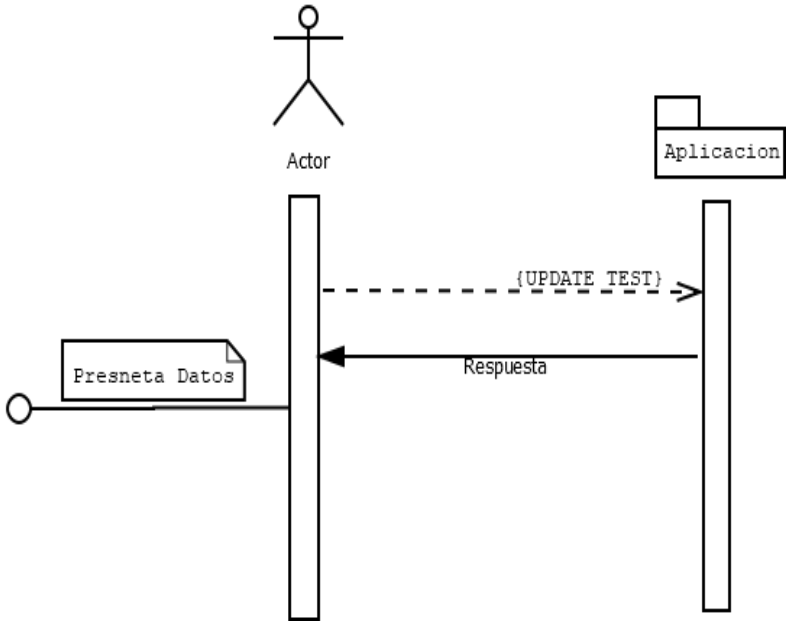


**Ilustración 29**: UML Diagrama de Secuencia
[**Fuente**: Elaboración Propia]

El único propósito de la aplicación es llevar a cabo ciertas operaciones (carga) sobre los motores de base de datos (bajo condiciones controladas). Su importancia radica en la parte de lógica de negocio, es decir la relación que hay entre el gestor y las peticiones que se realizan a este. Con el propósito de poder presentar una mejor perspectiva del funcionamiento de esta se desarrolló una pequeña vista para poder observar a detalle las operaciones que se realizan sobre cada gestor.

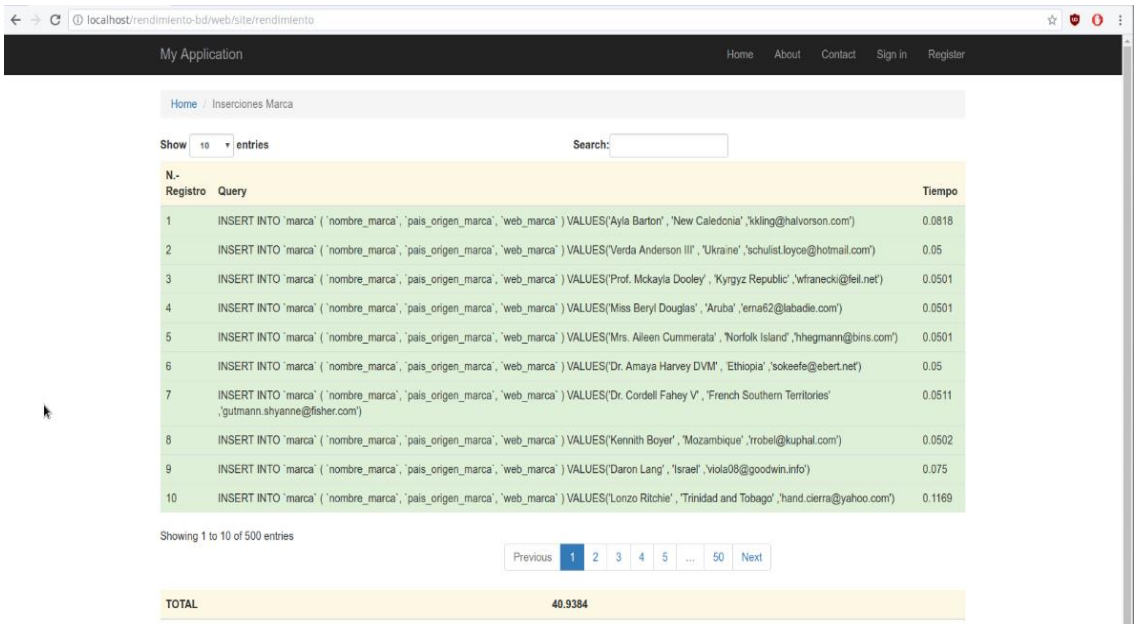


**Ilustración 30**: Ejecución de aplicación – Ejemplo
[**Fuente**: Elaboración Propia]

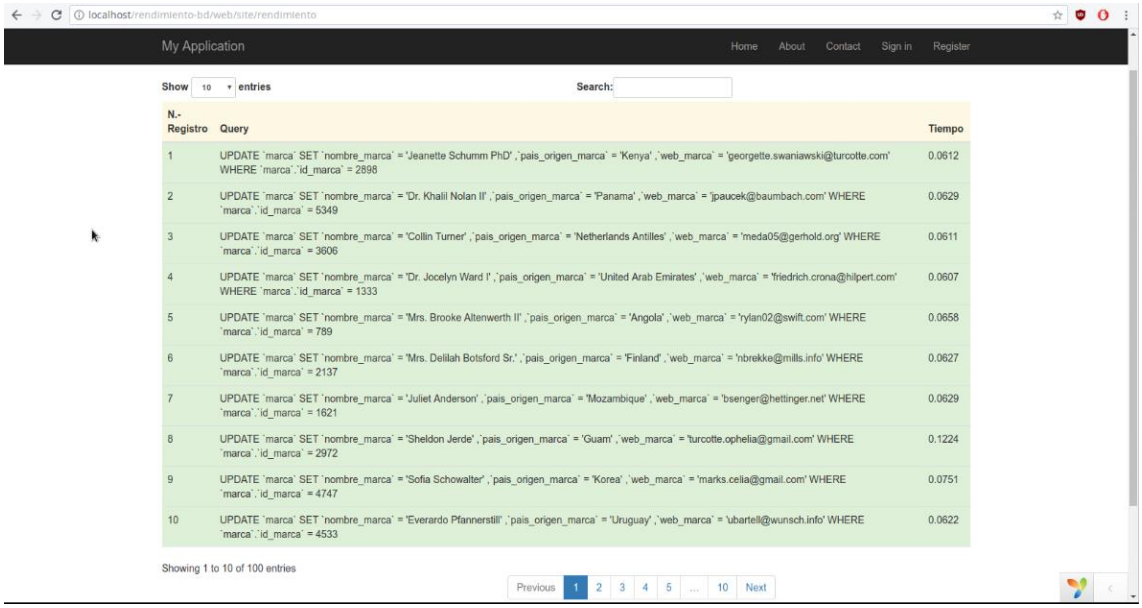


**Ilustración 31**: Ejecución de aplicación - Ejemplo
[**Fuente**: Elaboración Propia]

**Prueba de carga**

El propósito de las pruebas de carga es observar el comportamiento del sistema cuando este está siendo pedido por un número de usuarios concurrentes. Esta prueba nos permite saber cuál es el tiempo de respuesta de la aplicación con relación a la carga que se le está imponiendo.

La prueba de carga es el proceso de poner demanda en un sistema y medir su respuesta. El término prueba de carga se usa de diferentes maneras en la comunidad profesional de pruebas de software. Las pruebas de carga generalmente se refieren a la práctica de modelar el uso esperado de un programa de software mediante la simulación de múltiples usuarios que acceden al programa al mismo tiempo. Como tal, esta prueba es más relevante para sistemas multiusuario; a menudo uno construido usando un modelo cliente / servidor, como servidores web. Sin embargo, otros tipos de sistemas de software también pueden someterse a prueba de carga. Por ejemplo, un procesador de textos o editor de gráficos puede verse obligado a leer un documento extremadamente grande; o un paquete financiero puede ser forzado a generar un informe basado en varios años de datos. La prueba de carga más precisa simula el uso real, a diferencia de las pruebas que utilizan modelos teóricos o analíticos.

Las pruebas de carga le permiten medir el rendimiento de la calidad de servicio (QOS) de su sitio web en función del comportamiento real del cliente. La prueba de carga y rendimiento analiza el software destinado a una audiencia multiusuario sometiendo el software a diferentes números de usuarios virtuales y en vivo mientras se monitorean las mediciones de rendimiento bajo estas cargas diferentes. Las pruebas de carga y rendimiento generalmente se llevan a cabo en un entorno de prueba idéntico al entorno de producción antes de que el sistema de software pueda iniciarse.

**Plan de prueba**

Un plan de prueba de es un documento que describe el alcance y las actividades de prueba. Es la base para probar formalmente cualquier sistema.

> Un plan de prueba describe una serie de pasos que JMeter ejecutará cuando se ejecute. Un plan de prueba completo constará de uno o más grupos de subprocesos, controladores lógicos, controladores de generación de muestras, oyentes, temporizadores, aserciones y elementos de configuración
>
> (Apache JMeter - User's Manual: Building a Test Plan, s. f.)

A continuación, se describe el plan de pruebas para ambos motores de base de datos, los datos obtenidos fueron plasmados en cuadro organizador que más adelante se pueden contemplar. El estado inicial de la base de datos es de 200 mil registros.

CASO A) 200 usuarios virtuales para operaciones tipo select, cada usuario recuperara el 100% de los registros de la base de datos

CASO B) 50 usuarios virtuales para operaciones tipo update, se actualizará el 100% de los registros en la base de datos.

CASO C) 50 usuarios virtuales para operaciones tipo insert, se crearán mil nuevos registros.

**Grupo de hilos**

Los componentes del grupo de hilos son los elementos más importantes de cualquier plan de prueba. Todos los controladores y peticiones deben estar debajo de un grupo de hilos. Otros elementos, por ejemplo, los oyentes, pueden colocarse directamente bajo el plan de prueba, en cuyo caso se aplicarán a todos los grupos de subprocesos. Como su nombre lo indica, el elemento de grupo de hilos controla el número de hilos que JMeter utilizará para ejecutar su prueba. Los controles para un grupo de subprocesos le permiten:

- Establecer el número de hilos (Usuarios)
- Establecer el período de aumento gradual
- Establezca el número de veces para ejecutar la prueba

Es importante recordar que cada subproceso ejecutará el plan de prueba en su totalidad y completamente independientemente de otros subprocesos de prueba. Múltiples hilos se utilizan para simular conexiones concurrentes a su aplicación de servidor.

El período de aceleración le dice a JMeter cuánto tiempo llevar a la "aceleración" hasta la cantidad total de hilos elegidos. Si se utilizan 10 subprocesos y el período de aceleración es de 100 segundos, entonces JMeter tardará 100 segundos en poner en funcionamiento los 10 subprocesos. Cada hilo comenzará 10 (100/10) segundos después de que el hilo anterior haya comenzado. Si hay 30 subprocesos y un período de aceleración de 120 segundos, cada subproceso se retrasará 4 segundos.

La aceleración debe ser lo suficientemente larga como para evitar una carga de trabajo demasiado grande al comienzo de una prueba, y lo suficientemente corta como para que

los últimos hilos comiencen a funcionar antes de que los primeros terminen (a menos que uno quiera que eso suceda).

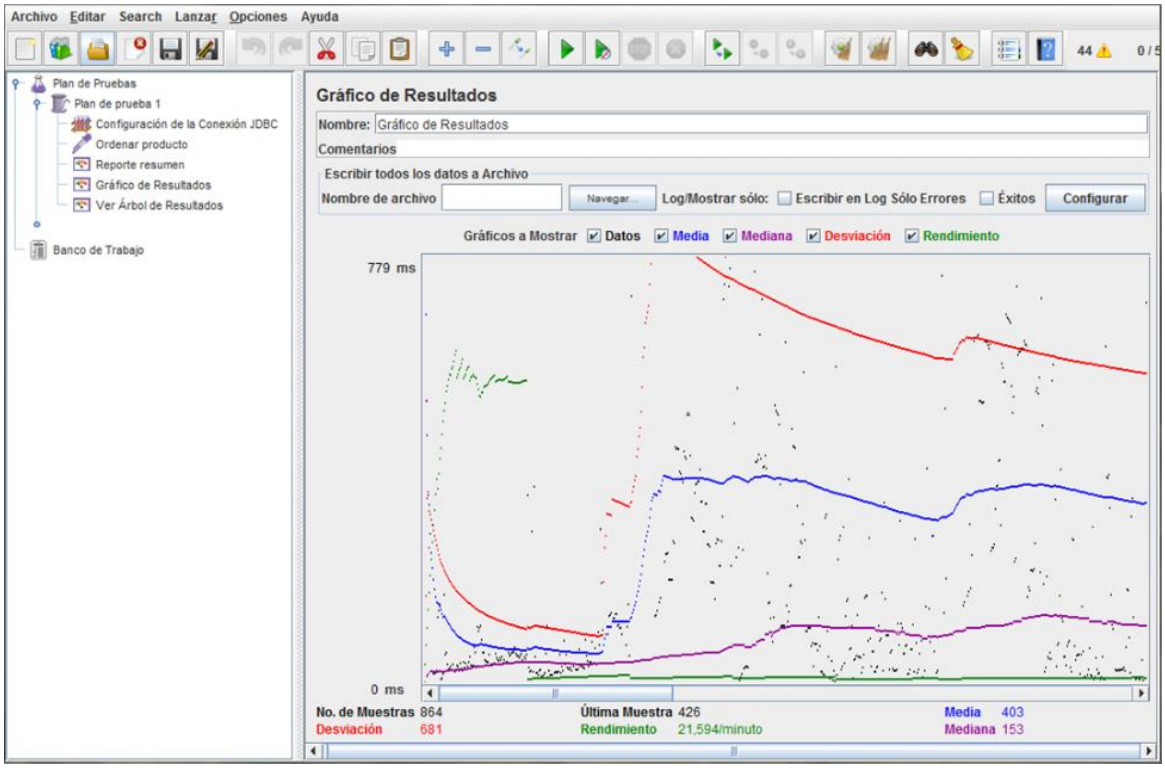


**Ilustración 32:** Ejecución plan de prueba 1 con GUI
[**Fuente:** Elaboración Propia]

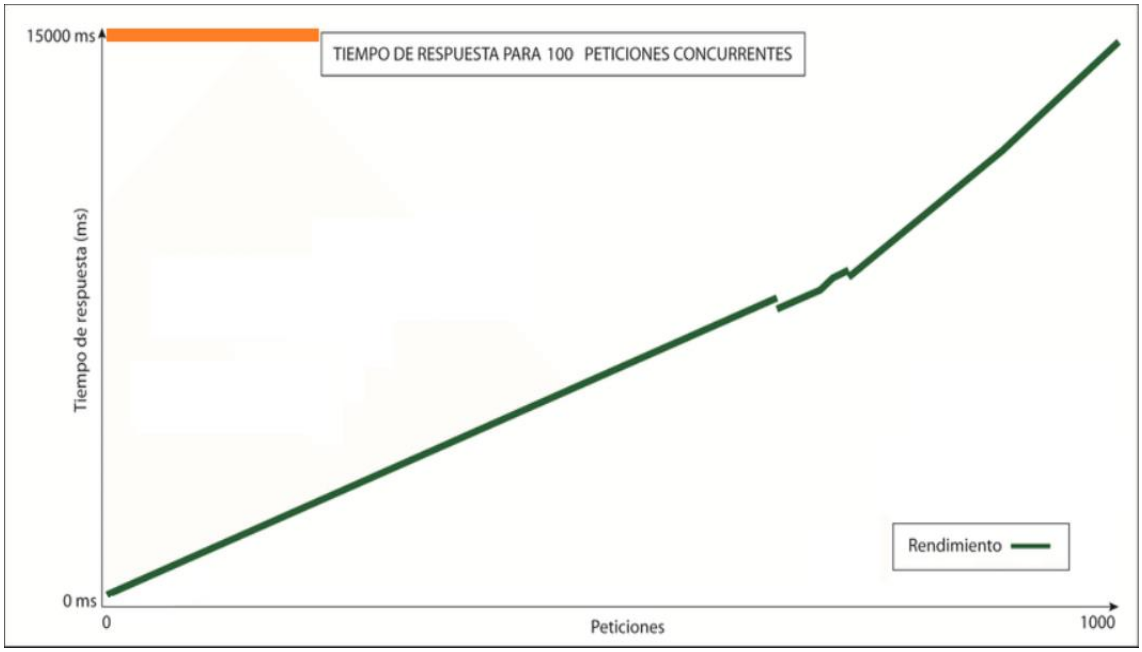


**Ilustración 33:** Ejemplo Grafico de resultados JMeter
[**Fuente:** Elaboración Propia]

**Oyentes**

Los oyentes brindan acceso a la información que JMeter reúne sobre los casos de prueba mientras se ejecuta JMeter. El oyente "Cuadro resumen de resultados" describe los tiempos de respuesta en un cuadro organizativo.

**Tabla 7**
*Detalles de Configuración Plan de prueba: Grupo de Hilos*

| | |
|---|---|
| Nombre test | NA |
| Dirección del servidor | Localhost 3306/27017 |
| Threads | Depende del caso |
| Periodo de aceleración | Número de Threads * 5s (Cinco segundos para cada hilo) |
| Oyentes | Cuadro resumen |
| Servidor Proxy | No |
| Tareas Opcionales | No |

**Nota.** Detalles para la configuración del grupo de hilos en JMeter
**Fuente:** Elaboración propia

**Método**

A continuación, se describen los procedimientos necesarios para desarrollar las pruebas de carga descritas con anterioridad sobre JMeter.

1. **Desplegar JMeter.**

Para ejecutar JMeter solamente se necesita acceder al directorio donde fue descargada la herramienta, en este caso esta se encuentra en el directorio '**C:\Users\...\Downloads\JMeter\bin**', una vez ubicados en el directorio ejecutamos el archivo "JMeter.bat".

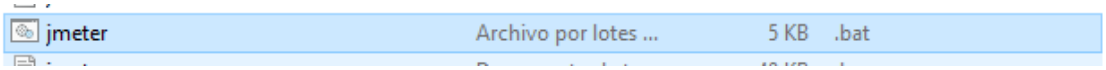


**Ilustración 34**: JMeter.bat
**Fuente**: Elaboración Propia

2. **JDBC conector.**

Antes que nada, es imprescindible adquirir y configurar esta librería, ya que esta API (**Java Database Connectivity**) permite la interacción con la base de datos (MySQL) desde la aplicación (**JMeter**), esto es necesario ya que a diferencia de MongoDB esta librería no viene incorporada por defecto.

**Tabla 8**

*MySQL JDBC*

| | |
|---|---|
| **Fuente** | https://goo.gl/JNFo5A |
| **Versión** | 8.0 |

**Nota.** JDBC para MySQL, fuente y versión
**Fuente**: Elaboración propia

Una vez descargado este archivo lo siguiente es; descomprimirlo y colocarlo en su directorio correspondiente '**C:\Users\...\Downloads\JMeter\lib**'.


**Ilustración 35**: MySQL
[**Fuente**: Elaboración Propia]

Después de haber realizado todo esto ya es posible empezar a configurar JMeter de tal manera que refleje la actividad del plan de prueba.

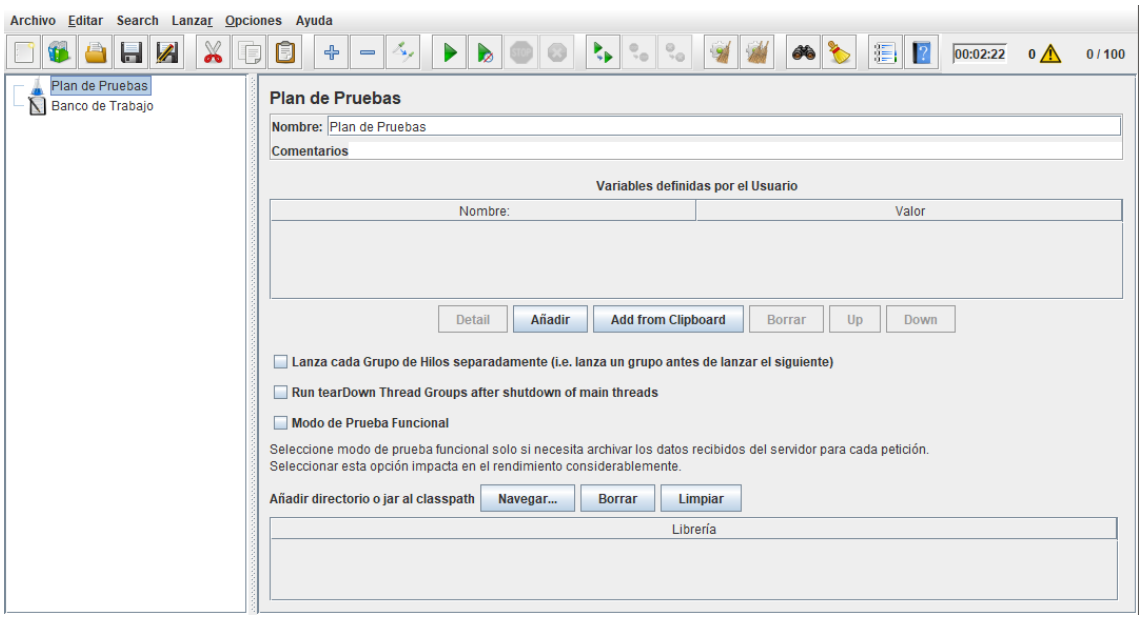


**Ilustración 36:** JMeter Interface
**Fuente**: Elaboración Propia

El elemento principal dentro de JMeter es el plan de pruebas o test plan, este contendrá elementos de configuración y todos los elementos que definen como se ejecutará las peticiones al servidor.

### 3. Configuración de la conexión

Este paso permite establecer los parámetros de conexión con el servidor de base de datos, para ello ubicamos sobre el **plan de prueba**, **clic derecho → añadir → elemento de configuración → configuración de la conexión JDBC**.

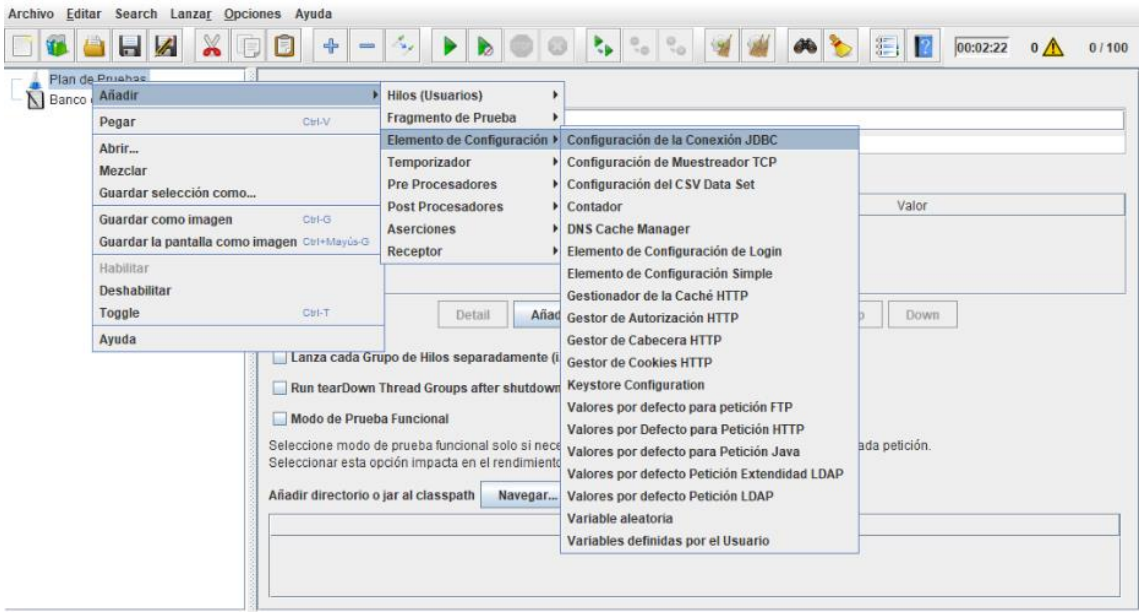


**Ilustración 37:** Configuración de Conexión JDBC
[**Fuente**: Elaboración Propia]

Una vez hecho esto se puede observar este elemento de configuración dentro del plan de prueba y los varios parámetros que lo caracterizan.

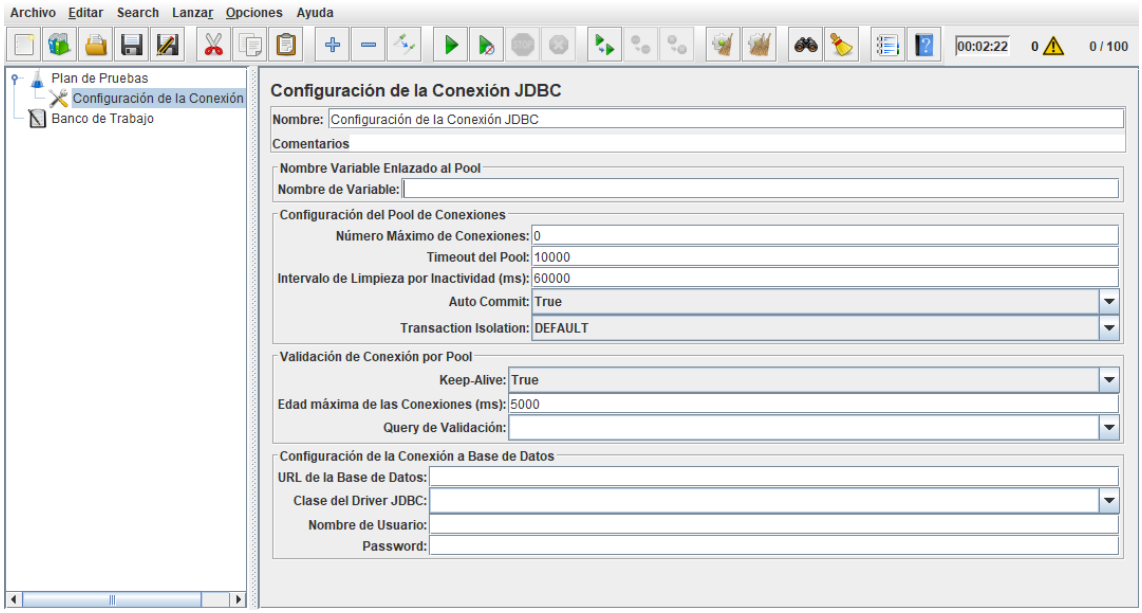


**Ilustración 38**: Configuración de Conexión JDBC
[**Fuente**: Elaboración Propia]

A continuación, se describen el propósito de cada parámetro y la configuración que es empleada para estos propósitos.

**Nombre de variable:** Ya que es posible configurar varias conexiones en un mismo test plan, es necesario identificar configuración de conexión con un nombre de variable único.

**URL de la base de datos:** El formato de este campo está compuesto por: JDBC + "nombre del gestor" + "dirección IP o dominio" + "puerto de comunicación empleado" + "nombre de la base de datos".

**Clase del driver JDBC:** Determina el tipo de JDBC a emplear, ya sea para MySQL o MongoDB.

**Nombre de usuario:** Este parámetro refiere al usuario que empleamos para acceder a la administración de nuestra base de datos.

**Contraseña:** Refiere a la contraseña para acceder a través de nuestro usuario administrador de la base de datos.

**Tabla 9**

*Configuración de conexión JDBC*

| SGBD | MySQL | MongoDB |
|---|---|---|
| Nombre de variable | Test | Test |
| URL de la base de datos | jdbc:<br>mysql:<br>//192.168.1.136:<br>3306<br>/mysqlvsmongodb | jdbc:<br>mongodb:<br>//192.168.1.137<br>:27017<br>/mysqlvsmongodb |
| Clase del driver JDBC | com.mysql.jdbc.driver | com.mongodb.jdbc.driver |
| Nombre de usuario | root | root |
| Contraseña | 12345678 | 12345678 |

**Nota.** Parámetros de configuración para la conexión JDBC
**Fuente**: Elaboración propia

### 4. Grupo de hilos

El siguiente elemento que configurar es el "grupo de hilos", para ello seguimos la siguiente secuencia **clic derecho sobre el test plan → añadir → hilos** (usuarios) → **grupo de hilos**.

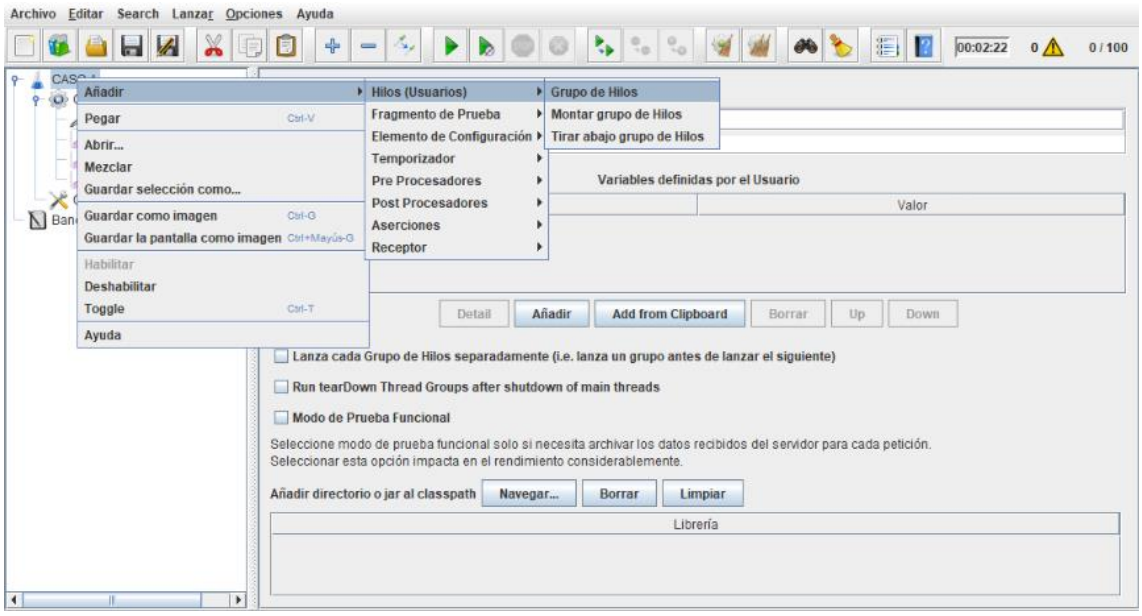


**Ilustración 39**: Grupo de hilos
**Fuente**: Elaboración Propia

Este elemento indica a JMeter cuantos usuarios o hilos empleará para realizar una terminada acción o grupo de acciones. Las acciones están determinadas por elementos que están dentro del grupo de hilo en cuestión. Lo que se pretende aquí es que JMeter emule el comportamiento de usuarios concurrentes haciendo peticiones al servidor.

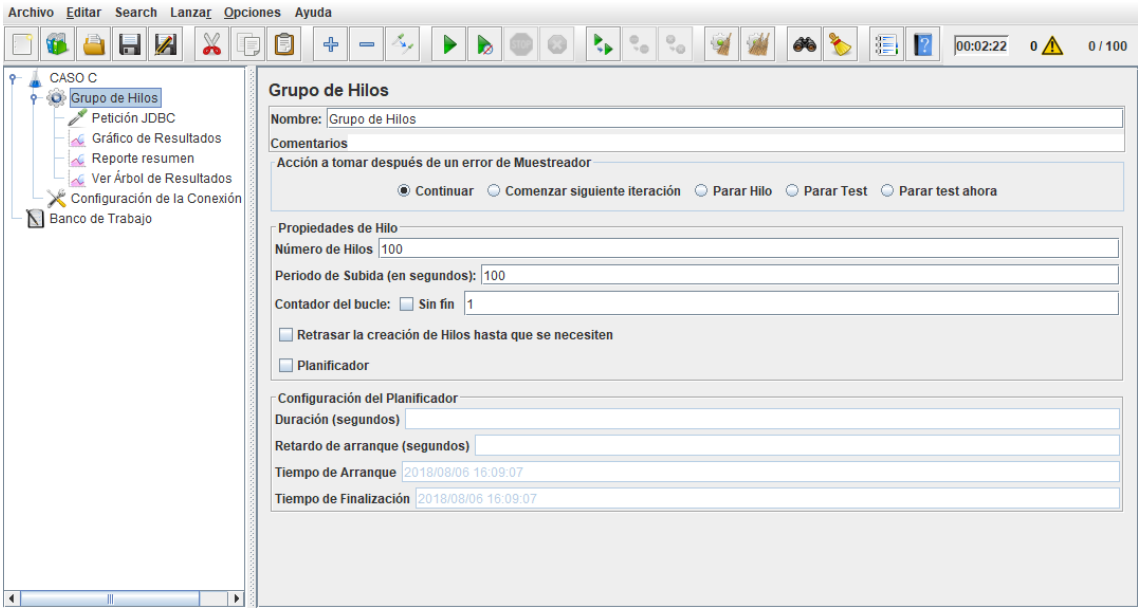


**Ilustración 40**: Grupo de hilos
[**Fuente**: Elaboración Propia]

**Numero de hilos:** Indica la cantidad de usuarios que realizaran peticiones al servidor.

**Periodo de subida (ramp-up-period):** Este parámetro indica el lapso de tiempo en el que va a ser lanzada la carga es decir la cantidad de usuarios conectados.

**Contador del bucle:** Esta propiedad indica cuantas veces va a ejecutar la petición cada usuario.

Suponiendo que se requiere desplegar el plan de prueba para el caso C los parámetros de configuración quedarían establecidos del siguiente modo.

**Tabla 10**

*Configuración de grupo de hilos*

| SGBD | MySQL | MongoDB |
|---|---|---|
| Numero de hilos | 50 | 50 |
| Periodo de subida | 100 | 100 |
| Contador del bucle | 20 | 20 |

**Nota.** Parámetros de configuración para grupo de hilos plan de prueba caso C
**Fuente**: Elaboración propia

### 5. Petición JDBC

Este elemento pertenece al grupo de hilos y sirve para indicar que tipo de peticiones van a ejecutar los usuarios que previamente fueron configurados. Recuerde que el test plan para esta investigación involucra operaciones de lectura y escritura (insert, update, select). Añadimos un elemento tipo petición JDBC siguiendo la secuencia; **clic derecho grupo de hilos → añadir → muestreador → petición JDBC**

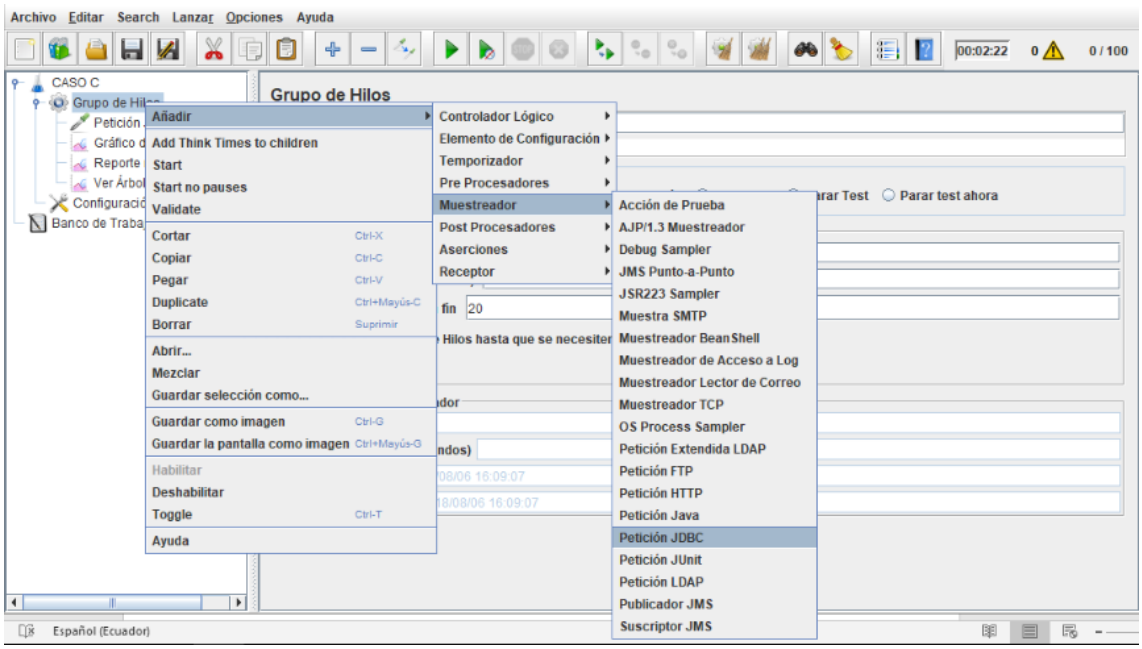


**Ilustración 41**: Petición JDBC
[**Fuente**: Elaboración Propia]

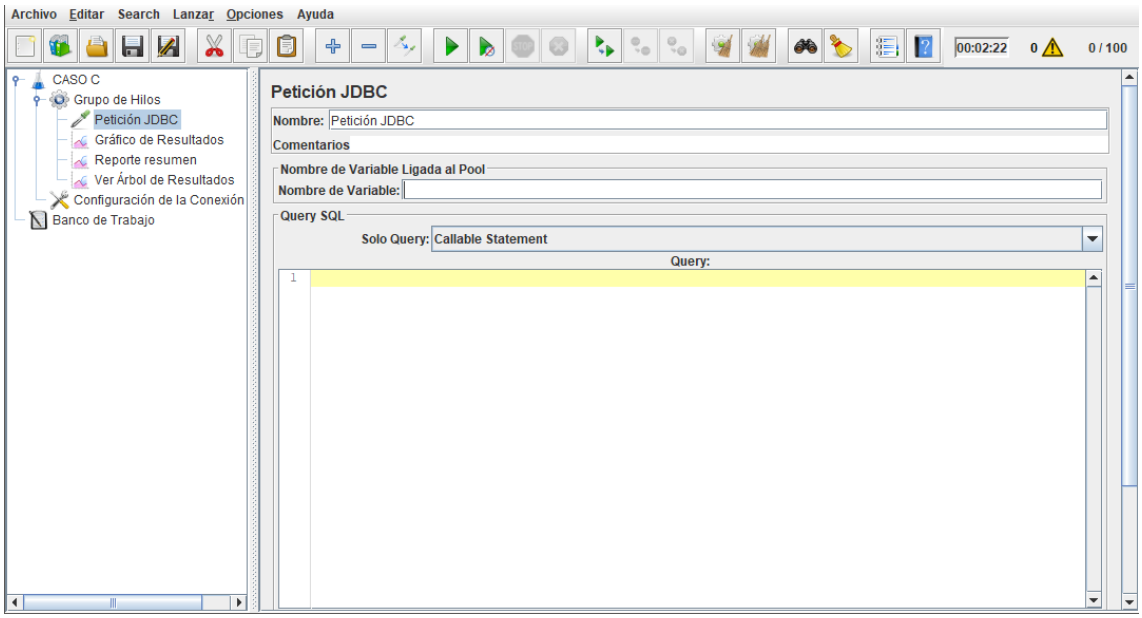


**Ilustración 42**: Petición JDBC
[**Fuente**: Elaboración Propia]

**Nombre de variable (ligada al pool):** Este campo debe ser igual al campo de mismo nombre que está en el JDBC de configuración de la conexión, ya que esta petición hace referencia a los parámetros de configuración situados allí con el propósito de interactuar con la base de datos.

**Query SQL:** Esta sección está compuesta por

- **Solo Query:** El tipo de petición a realizarse ya sea de tipo insert, update o select.

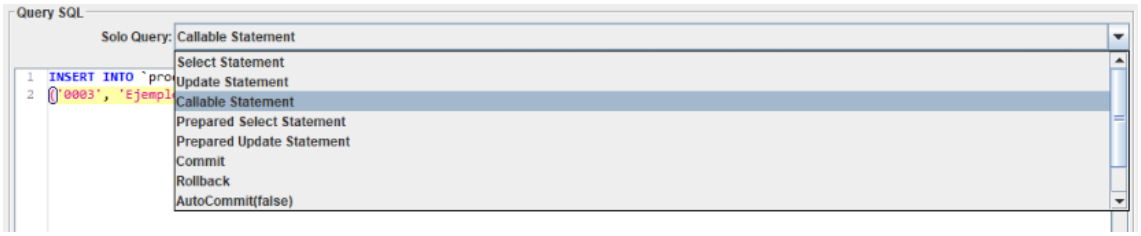


**Ilustración 43**: Parámetro Solo Query
[**Fuente**: Elaboración Propia]

- **Query:** El detalle de la petición en sí mismo.

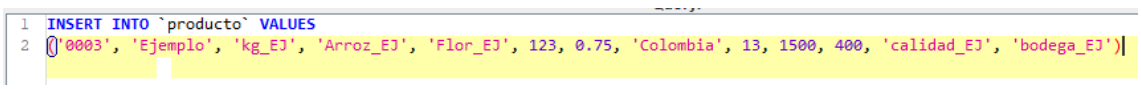


**Ilustración 44**: Parámetro Query
[**Fuente**: Elaboración Propia]

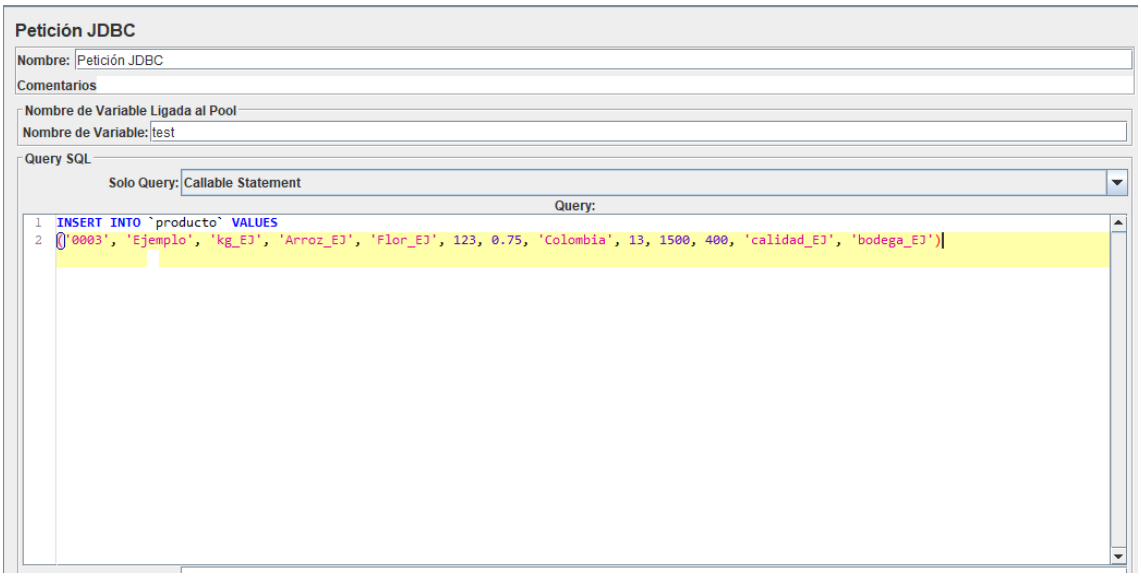


**Ilustración 45**: Petición JDBC
[**Fuente**: Elaboración Propia]

## 6. Oyentes

Los oyentes constituyen el último elemento del test plan. Estos tienen como propósito procesar la información proveniente de las pruebas. Existen varios, en este caso se emplean **grafico de resultados, reporte resumen y árbol de resultados.** Para agregar los oyentes siga la secuencia **clic derecho grupo de hilos → añadir → receptor → grafico de resultados**.

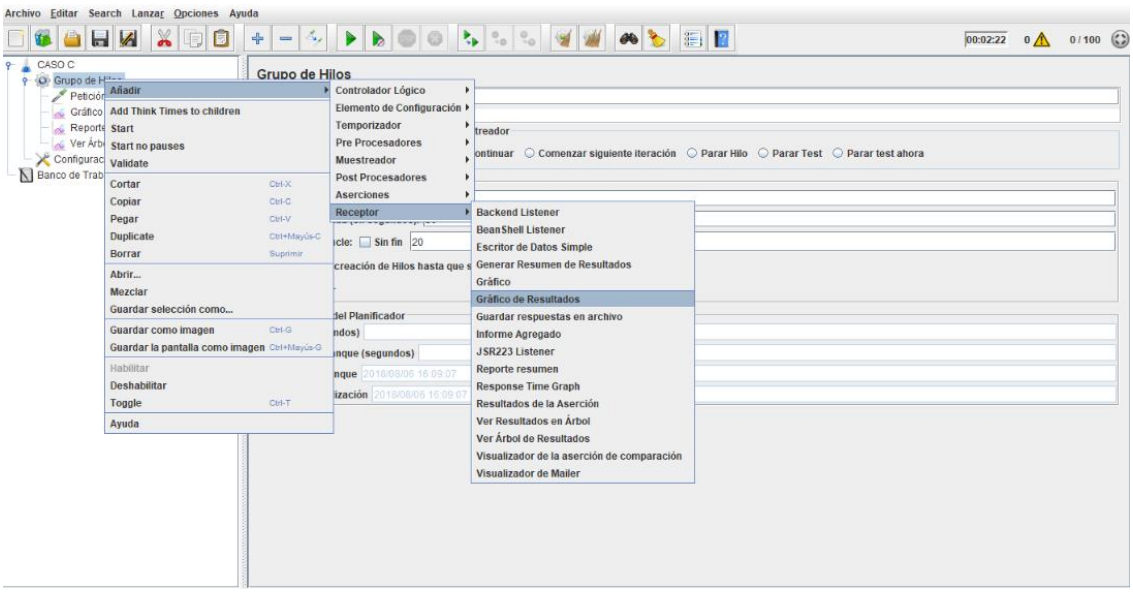


**Ilustración 46**: Oyentes
[**Fuente**: Elaboración Propia]

Finalmente, el plan de pruebas queda estructurado de la siguiente manera:

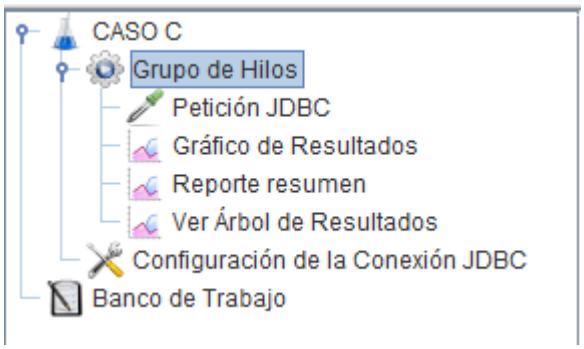


**Ilustración 47**: Plan de pruebas
**[Fuente**: Elaboración Propia]

Una vez ejecutado el plan de pruebas se presentan los diferentes resultados obtenidos.

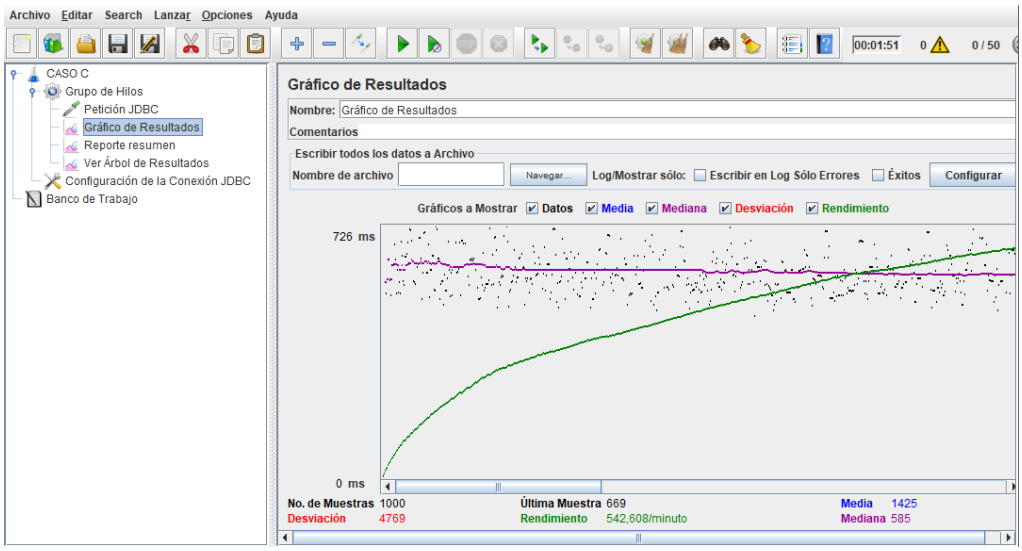


**Ilustración 48**: Gráfico de resultados
**[Fuente**: Elaboración Propia]

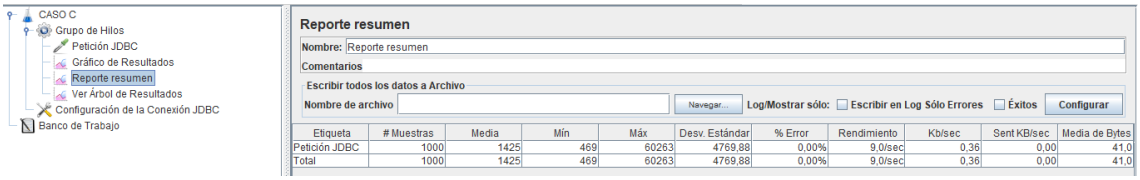


**Ilustración 49**: Cuadro resumen
[**Fuente**: Elaboración Propia]

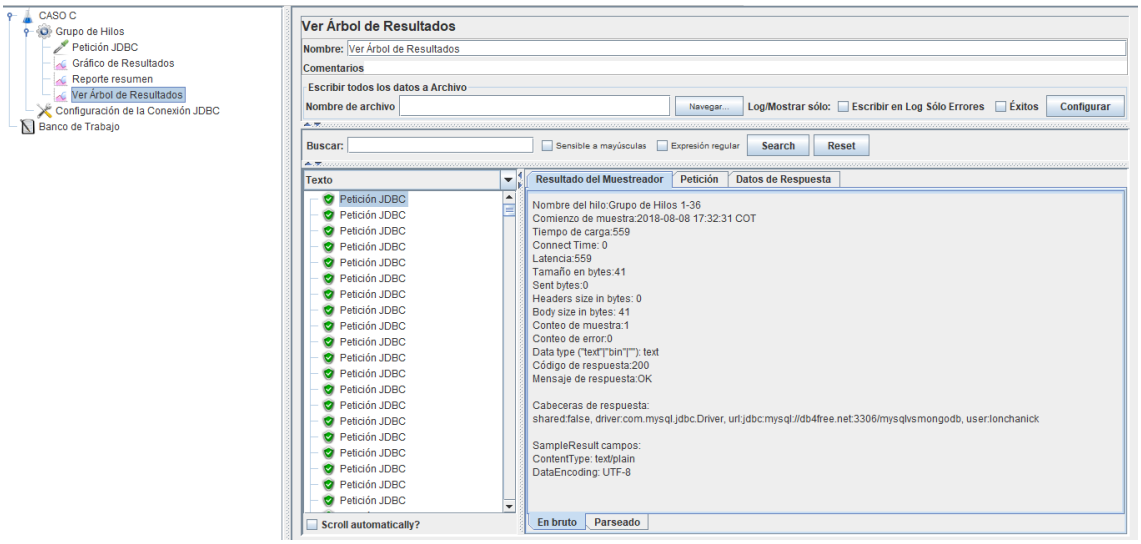


**Ilustración 50**: Árbol de resultados
[**Fuente**: Elaboración Propia]

**Pruebas de rendimiento**

El propósito de las pruebas es medir el tiempo de respuesta de los gestores en cuanto a operaciones de inserción, actualización, lectura y borrado, para cada una de estas operaciones se determinó una serie de ordenamientos que se describen a continuación.


**Ilustración 51:** Paradigma CRUD
[**Fuente**: https://goo.gl/nXwVpr]

**Inserción**

Se realizarán inserciones de 500, 1.000, 5.000, 10.000 y 100.000 registros con datos aleatorios para ambas bases de datos, ambas no contarán con ningún tipo de registro previo al inicio de cada carga.

**Actualización**

Para las operaciones de actualización la base de datos contara con un estado inicial de 5000.000 registros, se evaluarán dos casos; "A" Se actualizarán en primera instancia el 100% de los registros y "B" donde se efectuará la misma operación con el 50% de los registros para luego tomar el tiempo de respuesta. Se ejecutarán varias veces la prueba y se tomará un promedio.

**Lectura**

Para las operaciones de recuperación se evaluarán tres casos en donde: "A" se recuperará primeramente un único registro, "B" se recuperará el 25% de los registros y "C" Se recuperará el 100% de los registros. La base de datos contará con un estado inicial de 500.000 registros. Las pruebas se realizarán varias veces y se tomará un promedio (se procuró que la memoria cache no intervenga en los resultados).

### Borrado

Para las operaciones de borrado se hará una eliminación del 100% de los registros cuando la base de datos cuente con 50.000, 100.000 y 500.000 registros para cada caso, en cada caso la prueba se realizará varias veces y se tomará un promedio.

## Análisis y resultados

### Resultado pruebas de rendimiento

A continuación, se presentan los resultados de las pruebas realizadas a ambos gestores (SQL y MongoDB) y un breve análisis de estos. Para ejecutar las pruebas se empleó MongoDB en su versión 3.4 y MySQL 5.7.

Las características de los equipos empleados y todo lo relacionado al ambiente de prueba esta descrito en la metodología más arriba en este mismo documento. Todos los parámetros (memoria cache, procesos en segundo plano, etc..) que pudieran afectar directamente al rendimiento de cada gestor y por ende al resultado de las pruebas fueron debidamente gestionados con la finalidad de garantizar la igualdad de condiciones en el ambiente de prueba.

Se evaluaron las cuatro operaciones básicas que ejecuta cada gestor Inserción, Borrado, consulta y actualización.

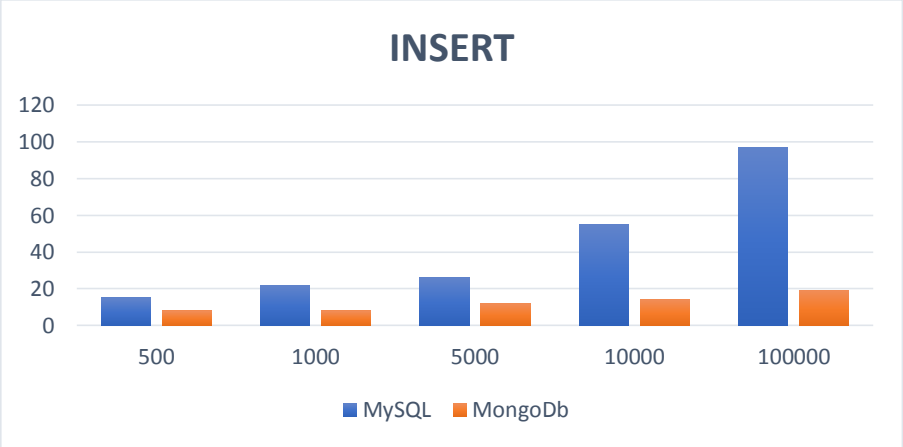


**Ilustración 52:** Resultados Pruebas de Inserción
[**Fuente:** Elaboración Propia]

Podemos observar en el gráfico que los resultados benefician a MongoDB, esto se debe a que MySQL emplea varias restricciones, como, por ejemplo, chequear que los campos de tipo no NULL estén correctos, que no se rompa ninguna regla de integridad de los datos, las claves foráneas correspondan, que las claves primarias o que los parámetros

coincidan con los pre estructurados, etc.… todo esto con el fin de garantizar la integridad de la información.

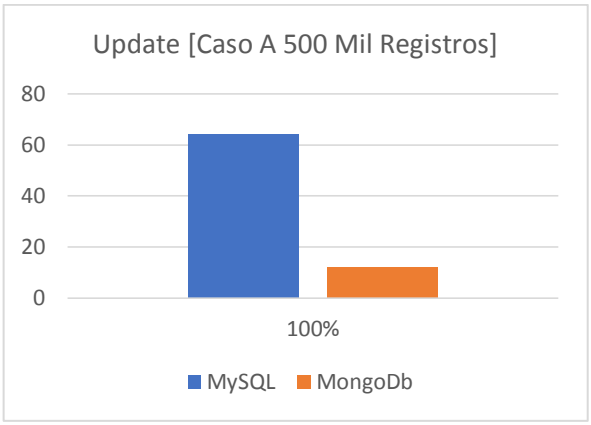


**Ilustración 53:** Resultados Pruebas de Actualización A
[**Fuente:** Elaboración Propia]

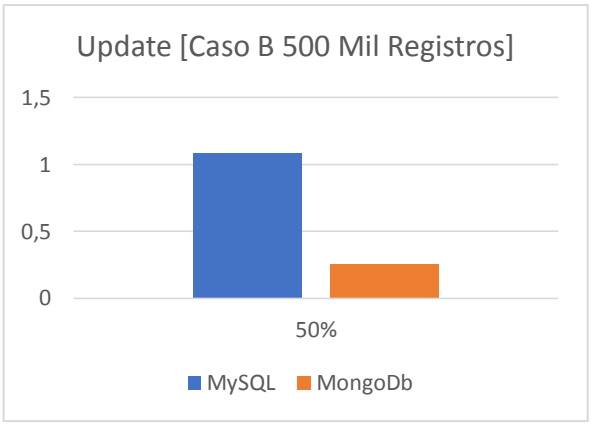


**Ilustración 54:** Resultados Pruebas de Actualización B
[**Fuente:** Elaboración Propia]

Podemos observar que los resultados para las pruebas de actualización favorecen nuevamente a MongoDB, debido a las mismas razones por las que este resultado prevalece en las pruebas de inserción.

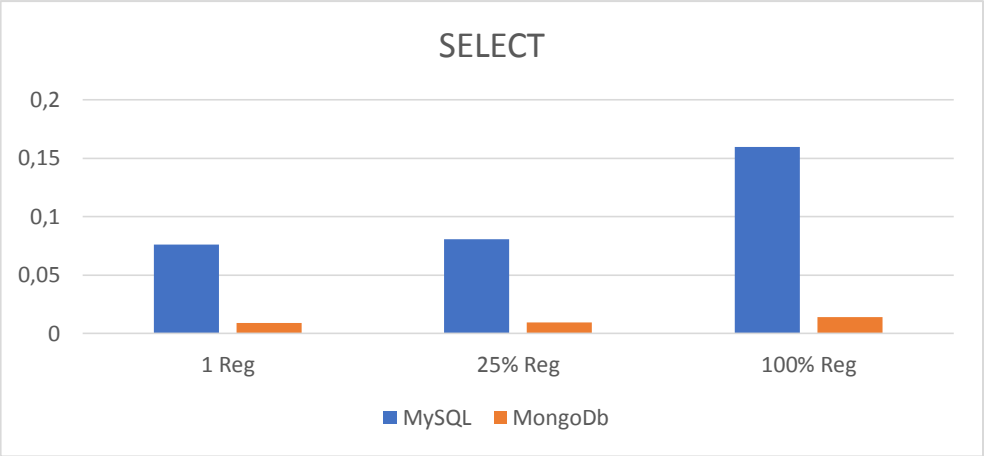


**Ilustración 55:** Resultado Pruebas de Lectura
[**Fuente:** Elaboración Propia]

La particularidad de los resultados en este caso es que las diferencias fueron minúsculas, todas las operaciones requirieron menos de un segundo, de igual manera MongoDB vuelve a triunfar en este caso superando a MySQL por poco. Importante no olvidar que el volumen de datos para este caso de prueba es de 500.000 registros para los tres parámetros de evaluación.

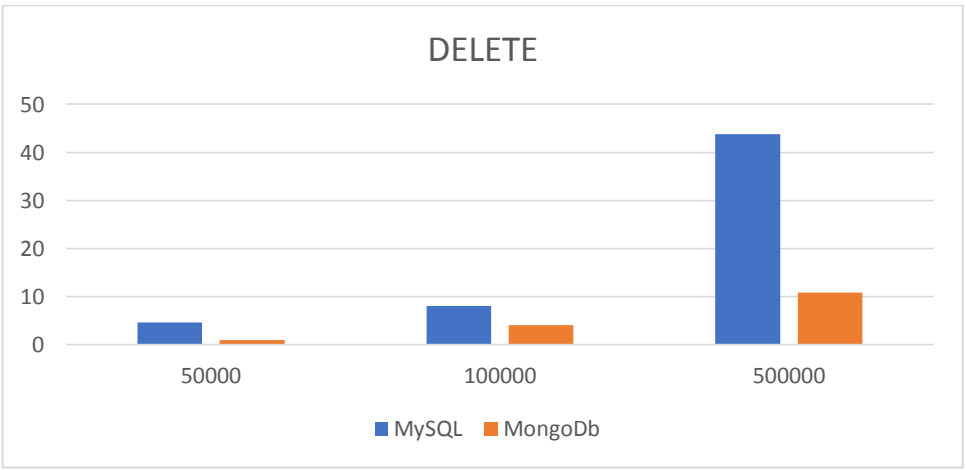


**Ilustración 56:** Resultado Pruebas de Borrado
[**Fuente:** Elaboración Propia]

Una vez más podemos observar que MongoDB prevalece en este aspecto, esto seguramente por las mismas razones expuestas en el caso de las inserciones, adicional a esto se podría decir que en este caso MySQL realiza procesos adicionales de integridad que afectan en su rendimiento.

**Resultados pruebas de carga**

**Tabla 10**
*Resultados pruebas de carga caso A*

| Gestores | MySQL | MongoDB |
|---|---|---|
| **Número de Usuarios** | 200 | 200 |
| **Operación tipo** | Select | Select |
| **Carga** | Select 100% Registros | Select 100% Registros |
| **Tiempo de respuesta** | 1.63m | 1.25m |
| **Peticiones fallidas** | 0% | 0% |
| **Punto de quiebre** | 397 usuarios | 460 usuarios |

**Nota.** Tabla de resultados para pruebas de carga CASO A
**Fuente:** Elaboración propia

Según el cuadro de resultado para las operaciones tipo SELECT podemos observar una diferencia significativa en tiempo de respuesta, esto debido a las relaciones y restricciones que emplea el motor de MySQL en el instante en que realizar la recuperación de los registros, MongoDB por su lado es significativamente más vertiginoso debido la naturaleza de sus transacciones que no garantizan ACID en las mismas.

**Tabla 11**
*Resultados CASO B*

| Gestores | MySQL | MongoDB |
|---|---|---|
| **Número de Usuarios** | 50 | 50 |
| **Operación tipo** | Update | Update |
| **Carga** | Update 100% Registros | Update 100% Registros |
| **Tiempo de respuesta** | 10.21m | 7.38m |
| **Peticiones fallidas** | 0% | 0% |
| **Punto de quiebre** | 90 usuarios | 120 usuarios |

**Nota.** Tabla de resultados para el caso B de las pruebas de carga
**Fuente:** Elaboración propia

El caso B que corresponde con las operaciones de actualización podemos observar que existe una diferencia significativa en tiempo de respuestas entre ambos gestores, en cuanto al punto de quiebre del sistema es importante mencionar que la cantidad de

usuarios se fueron aumentando gradualmente de tal manera que se halla el punto en que el sistema empieza a encolar las peticiones y posteriormente colapsa.

**Tabla 12**
*Resultados CASO C*

| **Gestores** | **MySQL** | **MongoDB** |
|---|---|---|
| **Número de Usuarios** | 50 | 50 |
| **Operación tipo** | Insert | Insert |
| **Carga** | mil registros | mil registros |
| **Tiempo de respuesta** | 11.40m | 6.74m |
| **Peticiones fallidas** | 0 | 0 |
| **Punto de quiebre** | 85 usuarios | 130 usuarios |

**Nota.** Tabla de resultados para el caso C de las pruebas de carga
**Fuente:** Elaboración propia

En este caso, que corresponde a las operaciones de inserción, según los antecedentes y las conclusiones de las anteriores pruebas, se podría decir que los resultados son predecibles. Una vez más MongoDB saca una diferencia significativa a su contra parte en cuanto a tiempo de respuesta se trata, esto debido a las restricciones (No nulidad, tipo de dato, relaciones, etc.) y diferentes factores que intervienen en una operación de este tipo para MySQL y que MongoDB ignora casi que por completo. En cuanto al punto de quiebre es evidente que MongoDB al ser más ligera con sus transacciones empleará menos recursos y por consiguiente responderá a muchas más peticiones que MySQL.

Indiscutiblemente MongoDB gana a su contra parte en todos los aspectos, pero a pesar de esto sería apresurado decir que uno prevalece sobre el otro, tanto MySQL con sus estructuras de datos rígidas, verificaciones de consistencia e integridad, así como MongoDB, son dos enfoques que abarcan características distintas y que juegan un papel crucial a la hora de elegir un sistema gestor de base de datos para un proyecto determinado.

Por lo tanto, podemos decir que MongoDB es una alternativa a los sistemas relacionales de base de datos en donde cuyo contexto de desarrollo prima la velocidad y manejo de grandes volúmenes de datos en un entorno distribuido.

Una de las grandes limitaciones que se presentaron durante este trabajo de investigación fue la complejidad de las herramientas empleadas durante el mismo, particularmente

JMeter que es un tanto complejo especialmente a la hora de desplegar el plan de pruebas sin la participación de la GUI. Otro obstáculo digno de mención fue la falta de componentes informático (Hardware) con una capacidad de procesamiento más avanzado ya que este tipo de herramientas suele ejecutarse en entornos hardware de alto rendimiento.

Una cuestión interesante que surgió durante esta investigación fue que el tema está bastante bien documentado a pesar de que varios expertos en el tema supieron expresar su desacuerdo en cuanto al desarrollo del mismo por considerarlo aparentemente sin propósito, esto debido a que tanto MongoDB como MySQL a pesar de que son herramientas que gestionan lo mismo (datos), cada una se constituye con un enfoque diferente, es decir cubren necesidades diferentes, cual comparación "autos y camionetas"; aparentemente tienen funciones similares pero enfoques distintos.

Como recomendación en base a la presente investigación comento que; debido a que este tipo de aplicaciones (una vez en producción) suelen ejecutarse en entornos distribuidos estaría interesante medir el tiempo de recuperación de las mismas en respuesta a la falla o caída de uno de sus nodos, es decir ¿qué gestor tarda más tiempo en recuperarse de cuál error?

# ANEXOS

Aplicación:
https://drive.google.com/open?id=11pm0XASioFrtFkKb6t2zpWKMJLyCHwsV

# BIBLIOGRAFÍA


About LoadUI | ReadyAPI Documentation. (s. f.). Recuperado 17 de agosto de 2018, de https://support.smartbear.com/readyapi/docs/loadui/intro/about.html

abraham, A. (2009, agosto 18). Redis, un nuevo modelo de base de datos ligero. Recuperado 17 de agosto de 2018, de https://novanebula.net/blog/gnulinux/redis-nuevo-modelo-base-datos-ligero/

Apache Cassandra. (2018). En *Wikipedia*. Recuperado de https://en.wikipedia.org/w/index.php?title=Apache_Cassandra&oldid=850826292

Apache JMeter - Apache JMeter™. (s. f.). Recuperado 13 de agosto de 2018, de https://jmeter.apache.org/

Apache JMeter - User's Manual: Building a Test Plan. (s. f.). Recuperado 9 de julio de 2018, de https://jmeter.apache.org/usermanual/build-test-plan.html

API Performance Testing with LoadUI Pro | Easily Reuse SoapUI Tests. (s. f.). Recuperado 17 de agosto de 2018, de https://www.soapui.org/professional/loadui-pro/features.html

Arboleda, M., Javier, F., Quintero Rendón, J. E., & Rueda Vásquez, R. (2016). A PERFORMANCE COMPARISON BETWEEN ORACLE AND MONGODB. *Ciencia e Ingeniería Neogranadina*, *26*(1), 109–129.

ASALE, R.-. (s. f.). diccionario. Recuperado 10 de diciembre de 2017, de http://dle.rae.es/?id=5ASmP2Z

Bases de datos NoSql en cloud computing. (s. f.).

Bases de datos NoSQL que son y tipos que nos podemos encontrar.pdf. (s. f.).

Bit. (s. f.). MongoDB para Big Data – replicación y sharding (I). Recuperado 17 de agosto de 2018, de https://www.bit.es/knowledge-center/mongodb-para-big-data-replicacion-y-sharding-i/

Boicea, A., Radulescu, F., & Agapin, L. I. (2012). MongoDB vs Oracle -- Database Comparison (pp. 330-335). IEEE. https://doi.org/10.1109/EIDWT.2012.32

Camargo-Vega, J. J., Camargo-Ortega, J. F., & Joyanes-Aguilar, L. (2015). Knowing the Big Data. *Facultad de Ingeniería*, *24*(38), 63–77.

Cardenas, J. E. S. (2014). ANÁLISIS COMPARATIVO DE DOS BASES DE DATOS SQL Y DOS BASES DE DATOS NO SQL, (2014), 80.

Cattaneo, M. F. P., Nocera, M. L., & Rottoli, G. D. (2014). Rendimiento de tecnologías NoSQL sobre cantidades masivas de datos. *CUADERNO ACTIVA*, (6), 11–17.

Cordova, Cuzco - 2013 - Análisis comparativo entre bases de datos relacionales con bases de datos no relacionales.pdf. (s. f.).

Cuervo, M. C. (2007). Evaluación del rendimiento. *Revista Universidad EAFIT*, *43*(148), 78–90.

de Datos, B. (2012). Bases de datos. *Gestión*, *6*(7), 9.

del Busto, H. G., & Enríquez, O. Y. (2013). Bases de datos NoSQL. *Revista Telem@ tica*, *11*(3), 21–33.

[ES] MongoDB vs Oracle - Comparación de base de datos.pdf. (s. f.).

Espinoza, C., & Fernanda, R. (s. f.). Análisis comparativo entre bases de datos relacionales con bases de datos no relacionales, 88.

Final De Carrera et al. - 2015 - Estudio comparativo de BBDD relacionales y NoSQL en un entorno industrial.pdf. (s. f.).

Gasystems. (2017, febrero 9). What Is Database Security And Why Is It Important? Recuperado 12 de agosto de 2018, de https://www.gasystems.com.au/database-security-important/

HammerDB. (s. f.). Recuperado 17 de agosto de 2018, de https://www.hammerdb.com/

Han, J., Haihong, E., Le, G., & Du, J. (2011). Survey on NoSQL database. En *Pervasive computing and applications (ICPCA), 2011 6th international conference on* (pp. 363–366). IEEE.

Introduction to Redis – Redis. (s. f.). Recuperado 17 de agosto de 2018, de https://redis.io/topics/introduction

Investigating storage solutions for large data.pdf. (s. f.).

JSON. (s. f.). Recuperado 16 de mayo de 2018, de https://json.org/json-es.html

Juan Anzaldo. (2005, diciembre 6). Breve Historia de las Bases de datos. Recuperado 18 de julio de 2018, de https://janzaldo.wordpress.com/2005/12/06/breve-historia-de-las-bases-de-datos/

LoadRunner. (2018). En *Wikipedia*. Recuperado de https://en.wikipedia.org/w/index.php?title=LoadRunner&oldid=852613355

LoadUI. (2018). En *Wikipedia*. Recuperado de https://en.wikipedia.org/w/index.php?title=LoadUI&oldid=825853120

María Del Carmen Gómez. (2012). Bases de datos. *Gestión*, *6*(7), 9.

Marqués, M. (2009). *Bases de datos*. Castelló de la Plana: Universitat Jaume I, Servei de Comunicació i Publicacions.

Martín, A., Chávez, S. B., Rodríguez, N. R., Valenzuela, A., & Murazzo, M. A. (2013b). Bases de datos NoSQL en cloud computing. En *XV Workshop de Investigadores en Ciencias de la Computación*.

Microsoft SQL Server. (2018). En *Wikipedia*. Recuperado de https://en.wikipedia.org/w/index.php?title=Microsoft_SQL_Server&oldid=855141341

MySQL. (2018). En *Wikipedia, la enciclopedia libre*. Recuperado de https://es.wikipedia.org/w/index.php?title=MySQL&oldid=109696577

MySQL. (s. f.). Recuperado 14 de junio de 2018, de http://www.oracle.com/technetwork/database/mysql/index.html

MySQL: Security. (s. f.). Recuperado 12 de agosto de 2018, de https://dev.mysql.com/doc/mysql-security-excerpt/5.7/en/security.html

Okman, L., Gal-Oz, N., Gonen, Y., Gudes, E., & Abramov, J. (2011). Security Issues in NoSQL Databases (pp. 541-547). IEEE. https://doi.org/10.1109/TrustCom.2011.70

Pavlo, A., Paulson, E., Rasin, A., Abadi, D. J., DeWitt, D. J., Madden, S., & Stonebraker, M. (2009). A comparison of approaches to large-scale data analysis. En *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data* (pp. 165–178). ACM.

Peña, C. A. L. (s. f.). ANÁLISIS DE LAS BASES DE DATOS NOSQL COMO ALTERNATIVA A LAS BASES DE DATOS SQL, 58.

PostgreSQL: About. (s. f.-a). Recuperado 17 de agosto de 2018, de https://www.postgresql.org/about/

PostgreSQL: About. (s. f.-b). Recuperado 14 de junio de 2018, de https://www.postgresql.org/about/

Resources & Collateral - LoadRunner. (s. f.). Recuperado 17 de agosto de 2018, de https://software.microfocus.com/es-es/products/loadrunner-load-testing/resources

Romero, A. C., Sanabria, J. S. G., & Cuervo, M. C. (2012b). Utilidad y funcionamiento de las bases de datos NoSQL. *Facultad de Ingeniería*, *21*(33), 21–32.

Ruiz, Franciso, Mateos - 2014 - Procesos de Markov.pdf. (s. f.).

Salazar Cárdenas, J. E. (2014). *Análisis comparativo de dos bases de datos SQL y dos bases de datos no SQL* (PhD Thesis). Universidad Tecnológica de Pereira.

Sharma, S., Tim, U. S., Gadia, S., Wong, J., Shandilya, R., & Peddoju, S. K. (2015). Classification and comparison of NoSQL big data models. *International Journal of Big Data Intelligence*, *2*(3), 201–221.

SQLite vs MySQL vs PostgreSQL: A Comparison Of Relational Database Management Systems. (s. f.). Recuperado 17 de agosto de 2018, de https://www.digitalocean.com/community/tutorials/sqlite-vs-mysql-vs-postgresql-a-comparison-of-relational-database-management-systems

Suarez-Cabal, M. J., de la Riva, C., & Tuya, J. (2010). Populating test databases for testing SQL queries. *IEEE Latin America Transactions*, *8*(2), 164–171.

Venegas-Martınez, F. (2012). Procesos markovianos en la toma de decisiones: contribuciones de Onésimo Hernández-Lerma. *Morfismos*, *16*(2), 1–28.

What is Microsoft SQL Server? - Definition from WhatIs.com. (s. f.). Recuperado 17 de agosto de 2018, de https://searchsqlserver.techtarget.com/definition/SQL-Server

What Is MongoDB? (s. f.). Recuperado 17 de agosto de 2018, de https://www.mongodb.com/what-is-mongodb

Who’s using Redis? – Redis. (s. f.). Recuperado 17 de agosto de 2018, de https://redis.io/topics/whos-using-redis