

Sistema de eventos MongoDB y Oracle sql

Reporte de proyecto

Johan Daniel Aguirre Arias

Alejandro Amu Garcia

Miguel Angel Gonzalez Arango

Sistemas intensivos de datos II

Santiago de Cali

Junio 3, 2024

Mario Julian Mora Cardona

Monica Maria Rojas Rincon

ÍNDICE

- [Historias de usuario y criterios de aceptación](#)
- [Comparación mongodb con otros sistemas nosql](#)
- [Comparación mongodb con sistemas relacionales](#)
- [Razones para usar mongodb para este aplicativo](#)
- [Modelo de datos](#)
- [Ventajas y desventajas del modelo de datos](#)
- [Estrategia a seguir](#)
- [Implementación de la estrategia](#)
- [Conclusiones](#)
- [Referencias bibliográficas](#)

Historias de usuario y criterios de aceptación

HU1 Creacion, modificacion y eliminacion de un evento

Como administrador, **quiero** poder crear nuevos eventos en la aplicación usando su título, descripción, categorías, fecha, lugar del evento, conferencista, asistentes y facultad que lo organiza. Así mismo quiero poder modificar esta misma información, y en caso de ser necesario tener la posibilidad de eliminar un evento. **Para** poder realizar una gestión eficiente de los eventos realizados en la universidad.

Criterios de aceptación:

Given soy administrador

And ingreso a la sección de “registrar evento”

And ingreso todos los campos requeridos de manera correcta

When doy click al botón “enviar”

Then el sistema debe mostrar un mensaje de que se ha creado un nuevo evento y el mismo debe poder ser visualizado.

Given soy administrador

And ingreso a la sección de “registrar evento”

And ingreso la mayoría de los campos requeridos

When doy click al botón “enviar”

Then el sistema debe mostrar un mensaje de que no se ha rellenado toda la información necesaria.

Given soy administrador
And ingreso a la sección de “registrar evento”
And ingreso a todos los campos requeridos pero alguno de manera incorrecta
When doy click al botón “enviar”
Then el sistema debe mostrar un mensaje de que la información que se ha rellenado no es válida.

Given soy administrador
When doy click en la opción “eventos”
Then el sistema debe mostrar una lista de todos los eventos

Given soy administrador
And ingreso a la sección de “eventos”
And seleccione un evento
And doy en la opción modificar evento
And ingreso los cambios que deseo realizar de manera correcta
When doy click al botón “enviar”
Then el sistema debe mostrar un mensaje de que se ha modificado correctamente
And la información modificada debe reflejarse en el evento

Given soy administrador
And ingreso a la sección de “eventos”
And seleccione un evento
And doy en la opción modificar evento
And ingreso los cambios que deseo realizar de manera incorrecta
When doy click al botón “enviar”
Then el sistema debe mostrar un mensaje de que no se ha podido modificar.

Given soy administrador
And ingreso a la sección de “eventos”
And seleccione un evento
When doy en la opción eliminar evento
Then el sistema debe mostrar una página con toda la información del evento solicitando la confirmación de la eliminación.

Given soy administrador
And ingreso a la sección de “eventos”
And seleccione un evento
And doy en la opción eliminar evento
When doy click al botón “confirmar eliminación”
Then el sistema debe mostrar un mensaje de que se ha eliminado correctamente.
And el evento debe desaparecer de la lista de eventos

HU2 Creacion, modificacion y eliminacion de espacios autorizados

Como administrador, **quiero** poder crear nuevos espacios en la aplicación usando su nombre, dirección y ciudad. Así mismo quiero poder modificar esta misma información, y en caso de ser necesario tener la posibilidad de eliminarla. **Para** poder realizar una gestión óptima y evitar percances con los espacios.

Criterios de aceptación:

Given soy administrador
And ingreso a la sección de “registrar espacio autorizado”
And ingreso todos los campos requeridos de manera correcta
When doy click al botón “enviar”
Then el sistema debe mostrar un mensaje de que se ha creado un nuevo espacio y el mismo debe poder ser visualizado.

Given soy administrador
And ingreso a la sección de “registrar espacio autorizado”
And ingreso la mayoría de los campos requeridos
When doy click al botón “enviar”
Then el sistema debe mostrar un mensaje de que no se ha rellenado toda la información necesaria.

Given soy administrador
And ingreso a la sección de “registrar espacio autorizado”
And ingreso la todos los campos requeridos pero alguno de manera incorrecta
When doy click al botón “enviar”
Then el sistema debe mostrar un mensaje de que la información que se ha rellenado no es válida.

Given soy administrador

When doy click en la opción “espacio autorizado”

Then el sistema debe mostrar una lista de todos los espacios disponibles.

Given soy administrador

And ingreso a la sección de “espacio autorizado”

And seleccione un espacio

And doy en la opción modificar espacio

And ingreso los cambios que deseo realizar de manera correcta

When doy click al botón “enviar”

Then el sistema debe mostrar un mensaje de que se ha modificado correctamente

And la información modificada debe reflejarse en el espacio correspondiente

Given soy administrador

And ingreso a la sección de “espacio autorizado”

And seleccione un espacio

And doy en la opción modificar espacio

And ingreso los cambios que deseo realizar de manera incorrecta

When doy click al botón “enviar”

Then el sistema debe mostrar un mensaje de que no se ha podido modificar.

Given soy administrador

And ingreso a la sección de “espacio autorizado”

And seleccione un espacio

When doy en la opción eliminar espacio

Then el sistema debe mostrar una página con toda la información del espacio solicitando la confirmación de la eliminación.

Given soy administrador

And ingreso a la sección de “espacio autorizado”

And seleccione un espacio

And doy en la opción eliminar espacio

When doy click al botón “confirmar eliminación”

Then el sistema debe mostrar un mensaje de que se ha eliminado correctamente.

And el espacio debe desaparecer de la lista de espacios

Como administrador, **quiero** poder registrar nuevos conferencistas y/o asistentes en la aplicación usando su cédula, nombre de usuario y nombre completo, tipo de relación con la institución, email y ciudad . Así mismo quiero poder modificar esta misma información, y en caso de ser necesario tener la posibilidad de eliminarla. **Para** poder realizar de manera dinámica todo lo relacionado a la asistencia a los eventos.

Criterios de aceptación:

Given soy administrador

And ingreso a la sección de “registrar asistente o conferencista”

And ingreso todos los campos requeridos de manera correcta

When doy click al botón “enviar”

Then el sistema debe mostrar un mensaje de que se ha creado un nuevo asistente y/o conferencista y el mismo debe poder ser visualizado.

Given soy administrador

And ingreso a la sección de “registrar asistente o conferencista”

And ingreso la mayoría de los campos requeridos

When doy click al botón “enviar”

Then el sistema debe mostrar un mensaje de que no se ha rellenado toda la información necesaria.

Given soy administrador

And ingreso a la sección de “registrar asistente o conferencista”

And ingreso la todos los campos requeridos pero alguno de manera incorrecta

When doy click al botón “enviar”

Then el sistema debe mostrar un mensaje de que la información que se ha rellenado no es válida.

Given soy administrador

When doy click en la opción “lista de asistentes o conferencistas”

Then el sistema debe mostrar una lista de todos los espacios disponibles.

Given soy administrador

And ingreso a la sección de “lista de asistentes o conferencistas”

And seleccione un conferencista o asistente

And doy en la opción modificar

And ingreso los cambios que deseo realizar de manera correcta

When doy click al botón “enviar”

Then el sistema debe mostrar un mensaje de que se ha modificado correctamente
And la información modificada debe reflejarse en el asistente o conferencista correspondiente

Given soy administrador

And ingreso a la sección de “lista de asistentes o conferencistas”

And seleccione un asistente o conferencista

And doy en la opción modificar

And ingreso los cambios que deseo realizar de manera incorrecta

When doy click al botón “enviar”

Then el sistema debe mostrar un mensaje de que no se ha podido modificar.

Given soy administrador

And ingreso a la sección de “lista de asistentes o conferencistas”

And seleccione un asistente o conferencista

When doy en la opción eliminar

Then el sistema debe mostrar una página con toda la información del espacio solicitando la confirmación de la eliminación.

Given soy administrador

And ingreso a la sección de “lista de asistentes o conferencistas”

And seleccione un asistente o conferencista

And doy en la opción eliminar

When doy click al botón “confirmar eliminación”

Then el sistema debe mostrar un mensaje de que se ha eliminado correctamente.

And el asistente o conferencista debe desaparecer de la lista de asistentes o conferencistas

HU4 Creación y visualización de comentarios

Como asistente, **quiero** poder registrar comentarios sobre los eventos en la aplicación usando mi nombre de usuario y el comentario en si. Así mismo quiero poder modificar este mismo comentario, y en caso de ser necesario tener la posibilidad de eliminarlo. **Para** poder expresar mi opinión sobre las actividades realizadas y cómo las mismas se desarrollaron.

Criterios de aceptación:

Given soy un usuario o el admin

And ingreso a la sección de “eventos”

And doy click en el botón agregar comentario

And ingreso todos los campos requeridos de manera correcta

When doy click al botón “enviar”

Then el sistema debe mostrar un mensaje de que se ha creado un nuevo comentario y debe poder ser visualizado.

Given soy un usuario o el admin

And ingreso a la sección de “eventos”

And doy click en el botón agregar comentario

And ingreso todos los campos requeridos de manera incorrecta

When doy click al botón “enviar”

Then el sistema debe mostrar un mensaje de que no se puede crear un comentario con datos incorrectos .

Given soy un usuario o el admin

When ingreso a la sección de “eventos”

And doy click en ver detalles de un evento

Then el sistema debe mostrar todos los comentarios de ese evento.

HU5 visualizar información detallada de un evento

Como asistente o administrador , **quiero** poder visualizar detalladamente un evento desde su información básica pasando por todos los usuarios que están en ese evento y en caso de que los asistentes sean miembros de la universidad poder ver los detalles almacenados así como los comentarios que tiene el mismo. **Para** poder explorar con más detalle los eventos a los cuales quiero asistir o los cuales estoy administrando.

Criterios de aceptación:

Given soy un usuario o el admin

And ingreso a la sección de “eventos”

When doy click en el botón ver detalles de un evento específico sin asistentes de la universidad

Then el sistema debe mostrar un vistazo detallado del evento seleccionado.

Given soy un usuario o el admin

And ingreso a la sección de “eventos”

When doy click en el botón ver detalles de un evento específico con asistentes de la universidad

Then el sistema debe mostrar un vistazo detallado del evento seleccionado y cargar los datos más detallados de los miembros de la universidad.

HU6 realizar sugerencias de eventos

Como asistente o administrador, **quiero** que a la hora de visualizar eventos en general se ubiquen más arriba aquellos que estén relacionados por categoría a otros eventos a los cuales he asistido, **para** poder tener un sistema más personalizado y adaptable a mis necesidades.

Criterios de aceptación:

Given soy un usuario o el admin

When ingreso a la sección de “eventos”

Then el sistema debe mostrar al principio de la página aquellos eventos con las mismas categorías a las que ya he asistido.

Comparación mongodb con otros sistemas nosql

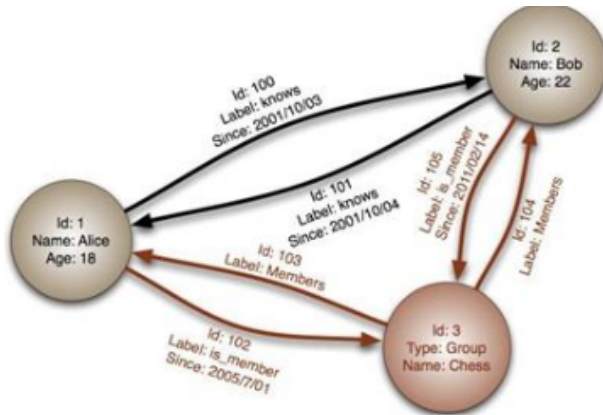
para este punto usaremos los siguientes sistemas nosql como puntos de comparación:

- neo4j
- cassandra
- Db4o

se han seleccionado estos pues son los más conocido dentro de lo que buscan realizar (bases de datos orientadas a columnas, orientadas a grafos y orientadas a objetos) posteriormente se listaran todas las ventajas y desventajas de cada uno desde cada parte de las propiedades acid pasando por seguridad y eficiencia al final se resumirá las conclusiones de la sección en una tabla de comparación

Neo4j

Neo4j es una base de datos orientada a grafos con un enfoque en la escalabilidad de los datos a largo plazo y en ser un sistema lo más fácil de comprender y manipular posible.



Ventajas:

Propiedades ACID: Neo4j cumple con las propiedades ACID, garantizando transacciones seguras y coherentes.

Modelado de Grafos: Ideal para datos conectados y consultas complejas de grafos.

Eficiencia en Consultas de Grafos: Muy eficiente en la ejecución de consultas que involucran múltiples relaciones.

Escalabilidad Horizontal: Aunque es más comúnmente escalada verticalmente, también admite escalabilidad horizontal con Causal Clustering.

Comunidad y Soporte: Gran comunidad de usuarios y soporte robusto.

Desventajas:

Rendimiento en Datos Tabulares: No es ideal para datos altamente tabulares o transacciones con gran volumen de datos no conectados.

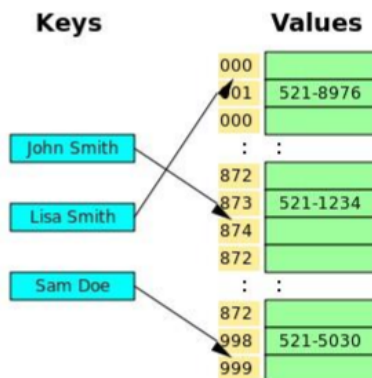
Complejidad de Implementación: Puede ser más complejo de implementar y mantener en comparación con bases de datos relacionales.

Costos: Las versiones empresariales pueden ser costosas.

Escalabilidad Compleja: La escalabilidad horizontal puede ser más complicada y menos intuitiva que en otras bases de datos NoSQL.

Cassandra

Cassandra es un sistema nosql centrado en los sistemas por columnas o clave valor enfocada en el rendimiento de consultas simples y en la clusterización de datos.



Ventajas:

Propiedades ACID: No cumple completamente con ACID, pero proporciona AP (Availability y Partition Tolerance) en el teorema CAP.

Alta Disponibilidad: Diseñada para ser altamente disponible y tolerante a fallos.

Escalabilidad Horizontal: Escala fácilmente añadiendo más nodos al clúster.

Eficiencia en Escrituras: Optimizada para operaciones de escritura rápida y masiva.

Soporte para Multi-Datacenter: Soporte nativo para replicación entre múltiples datacenters.

Desventajas:

Propiedades ACID Limitadas: No es completamente ACID, lo que puede ser un inconveniente para algunas aplicaciones que requieren transacciones estrictas.

Atomicidad:

Ventajas: Ofrece atomicidad a nivel de fila, garantizando que las operaciones en una sola fila son atómicas.

Desventajas: No garantiza atomicidad a nivel de múltiples filas o tablas, lo cual puede ser limitante para algunas aplicaciones.

Consistencia:

Ventajas: Permite configuraciones de consistencia ajustables (e.g., ONE, QUORUM, ALL), ofreciendo flexibilidad para equilibrar entre consistencia y rendimiento.

Desventajas: No es estrictamente consistente, lo que significa que pueden existir lecturas sucias y datos eventualmente consistentes.

Aislamiento:

Ventajas: Proporciona aislamiento básico para operaciones dentro de una sola partición.

Desventajas: Carece de niveles de aislamiento robustos, lo que puede llevar a condiciones de carrera y lecturas inconsistentes en transacciones complejas.

Durabilidad:

Ventajas: Diseñada para ser altamente duradera con replicación de datos y mecanismos de tolerancia a fallos.

Desventajas: La durabilidad puede depender de la configuración de la replicación y puede implicar una sobrecarga de rendimiento.

Eficiencia en Consultas Complejas: Las consultas complejas pueden ser menos eficientes que en bases de datos relacionales o de grafos.

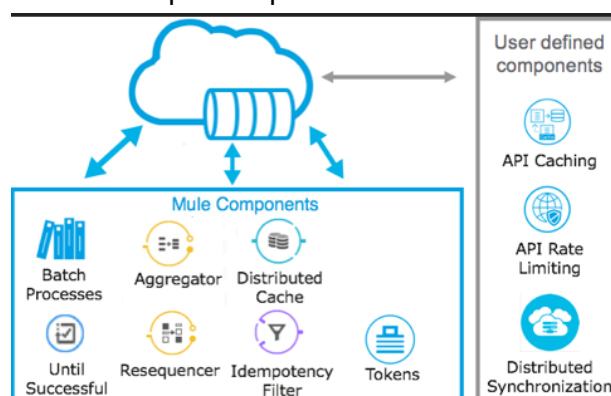
Mantenimiento y Configuración: Requiere un mantenimiento y configuración cuidadosos para garantizar el rendimiento y la consistencia.

Lenguaje de Consulta Limitado: CQL (Cassandra Query Language) es menos expresivo que muchos otros lenguajes de consultas.

Consistencia Eventual: La consistencia eventual puede no ser adecuada para todas las aplicaciones.

Db4o

Db4o es una base de datos orientada a objetos donde tiene un enfoque es ser amigable con los lenguajes de programación del mismo ayudando con su implementación de la manera más natural posible pues no es necesario conocimiento extra además del propio POO



Ventajas:

Propiedades ACID: Db40 cumple con las propiedades ACID, garantizando transacciones seguras y coherentes.

Orientación a Objetos: Ideal para aplicaciones con estructuras de datos complejas y orientadas a objetos.

Integración con Lenguajes OO: Fácil integración con lenguajes de programación orientados a objetos como Java y C#.

Fácil de usar: Menor curva de aprendizaje para desarrolladores familiarizados con programación orientada a objetos.

Almacenamiento Embebido: Puede ser embebido directamente en aplicaciones, lo que es útil para aplicaciones móviles y de escritorio.

Desventajas:

Eficiencia en Consultas Complejas: Las consultas pueden no ser tan eficientes como en bases de datos relacionales y NoSQL diseñadas para grandes volúmenes de datos.

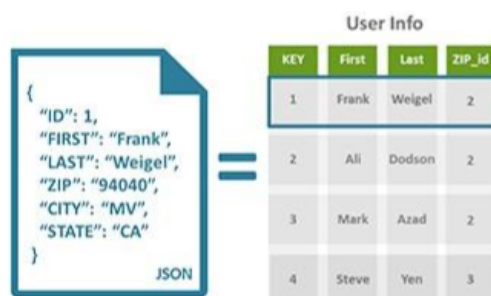
Escalabilidad Limitada: No está diseñada para una escalabilidad horizontal masiva.

Adopción Limitada: Menos adopción en la industria en comparación con otros sistemas de bases de datos, lo que puede limitar el soporte y la comunidad.

Rendimiento en Operaciones Masivas: Puede no ser tan eficiente para operaciones masivas de datos en comparación con otras bases de datos NoSQL.

MongoDB

Mongodb es un sistema nosql enfocado a documentos donde se prioriza la disponibilidad y la eficiencia en lectura de las consultas.



Ventajas:

Propiedades ACID: Desde la versión 4.0, soporta transacciones ACID multi-documento.

Escalabilidad horizontal: Facilita la escalabilidad horizontal con sharding.

Flexibilidad de Esquema: Esquema flexible que permite almacenar documentos JSON/BSN.

Eficiencia en Consultas y Escrituras: Muy eficiente en operaciones de lectura y escritura, especialmente con índices adecuados.

Amplia Adopción y Comunidad: Amplia adopción en la industria y gran comunidad de soporte.

Desventajas:

Uso Intensivo de Memoria: Requiere una gran cantidad de memoria para funcionar eficientemente.

Complejidad de Sharding: La configuración y el mantenimiento del sharding pueden ser complejos.

Seguridad: Históricamente ha tenido problemas de seguridad, aunque se han mejorado significativamente en versiones recientes.

Consistencia Eventual por Defecto: Por defecto, ofrece consistencia eventual que puede no ser adecuada para todas las aplicaciones.

Tamaño de Documentos: El tamaño de documentos individuales está limitado a 16MB, lo que puede ser una restricción para algunos casos de uso.

Resumen general:

Eficiencia:

Neo4j: Muy eficiente en consultas de grafos, pero menos en datos tabulares.

Cassandra: Muy eficiente en escrituras masivas y lecturas rápidas, pero menos en consultas complejas.

Db4o: Eficiente para aplicaciones embebidas y orientadas a objetos, pero no para grandes volúmenes de datos.

MongoDB: Muy eficiente en operaciones de lectura y escritura con índices adecuados, pero puede ser intensivo en memoria.

Seguridad:

Neo4j: Buenas características de seguridad, pero la complejidad puede aumentar en implementaciones grandes.

Cassandra: Seguridad robusta pero requiere configuración cuidadosa.

Db4o: Seguridad básica adecuada para aplicaciones embebidas.

MongoDB: Mejoras significativas en seguridad, pero históricamente ha tenido vulnerabilidades.

Escalabilidad:

Neo4j: Escalabilidad vertical más común, aunque soporta escalabilidad horizontal.

Cassandra: Excelente escalabilidad horizontal.

Db4o: Limitada escalabilidad, más adecuada para aplicaciones embebidas.

MongoDB: Buena escalabilidad horizontal con sharding.

Tabla general:

	Propiedades ACID	Eficiencia	Seguridad	Escalabilidad
Neo4j	Cumple con todas las propiedades ACID	eficiente en consulta nodulares cercanas	buena pero complicada de mantener a largo plazo	escalabilidad vertical, la escalabilidad horizontal está restringida
Cassandra	No cumple ACID proporciona AP (Availability y Partition Tolerance)	eficiente en datos masivos pero no en consultas complejas	seguridad robusta pero requiere de una fuerte configuración	excelente escalabilidad horizontal sin escalabilidad vertical
Db4o	Cumple con todas las propiedades ACID	buena para apps orientadas a objetos y con un volumen de datos moderado	seguridad basica apta para aplicaciones pequeñas embebidaas	limitada
Mongodb	Cumple con todas las propiedades ACID	bueno en lectura y escritura pero con alto coste de memoria	buena seguridad actualmente sufrio problemas en el pasado	buena escalabilidad gracias al sharding

después de analizar las ventajas y desventajas de cada uno de los sistemas y viendo los requerimientos (alto volumen de datos simples con formas desestructuradas y consultas complejas) hemos decidido optar por el uso de mongodb

Comparación mongodb con sistemas relacionales

Bueno pero la pregunta que surge con todo es: ¿por qué una base de datos nosql? Si ya tenemos una base de datos relacional ¿no podríamos o bien ampliar la misma para este sistema o crear una nueva para el manejo de este sistema?

Para resolver esta pregunta compararemos mongodb con dos de los sistemas relacionales más importantes:

- oracle
- Postgresql

MongoDB

Ventajas:

Atomicidad:

Soporta transacciones ACID multi-documento desde la versión 4.0, lo que permite garantizar que todas las operaciones en una transacción se completen o se reviertan.

Consistencia:

Ofrece consistencia fuerte dentro de una transacción, asegurando que los datos estén actualizados y coherentes.

Aislamiento:

Proporciona niveles de aislamiento para prevenir interferencias entre transacciones concurrentes, aunque más limitado en comparación con RDBMS.

Durabilidad:

Utiliza técnicas de journaling para garantizar que una vez confirmada una transacción, los datos son persistentes.

Escalabilidad:

Facilita la escalabilidad horizontal mediante sharding, permitiendo manejar grandes volúmenes de datos.

Flexibilidad de Esquema:

Esquema flexible que permite almacenar documentos JSON/BSON, adaptándose a cambios en los datos sin necesidad de alteraciones complejas del esquema.

Comunidad y Ecosistema:

Amplia adopción en la industria y una gran comunidad de soporte.

Desventajas:

Atomicidad:

Antes de la versión 4.0, la atomicidad solo se garantizaba a nivel de documento individual.

Consistencia:

La consistencia eventual es el comportamiento por defecto fuera de las transacciones, lo que puede no ser adecuado para todas las aplicaciones.

Aislamiento:

Los niveles de aislamiento más altos pueden afectar el rendimiento, especialmente en sistemas con alta concurrencia.

Durabilidad:

La durabilidad puede impactar en el rendimiento debido a la sobrecarga de las operaciones de journaling y replicación.

Uso Intensivo de Memoria:

Requiere una gran cantidad de memoria para funcionar eficientemente.

PostgreSQL



Ventajas:

Atomicidad:

Soporta transacciones ACID completas, asegurando que todas las operaciones dentro de una transacción se completen completamente o no se apliquen en absoluto.

Consistencia:

Mantiene la consistencia de los datos aplicando todas las reglas y restricciones durante las transacciones.

Aislamiento:

Ofrece varios niveles de aislamiento (READ COMMITTED, REPEATABLE READ, SERIALIZABLE) para prevenir interferencias entre transacciones concurrentes.

Durabilidad:

Utiliza WAL (Write-Ahead Logging) para asegurar que una vez que una transacción se ha confirmado, los datos estarán persistentes.

Eficiencia en Consultas Complejas:

Muy eficiente en la ejecución de consultas SQL complejas, con soporte para índices avanzados y optimización de consultas.

Ampliable:

Altamente extensible con soporte para procedimientos almacenados, tipos de datos personalizados y extensiones.

Seguridad:

Ofrece robustas características de seguridad, incluyendo autenticación, autorización y cifrado.

Desventajas:

Escalabilidad Horizontal:

La escalabilidad horizontal puede ser más complicada y menos intuitiva que en algunas bases de datos NoSQL.

Rendimiento en Escritorios Masivos:

Puede no ser tan eficiente como algunas bases de datos NoSQL para operaciones masivas de escritura.

Complejidad de Configuración:

Requiere una configuración y mantenimiento cuidadosos para garantizar el rendimiento óptimo.

Costos de Hardware:

Puede requerir hardware más potente para manejar grandes volúmenes de datos y altas tasas de transacciones.

Oracle



Ventajas:

Atomicidad:

Garantiza transacciones ACID completas, asegurando que todas las operaciones dentro de una transacción se completen completamente o no se apliquen en absoluto.

Consistencia:

Mantiene la consistencia de los datos aplicando todas las reglas y restricciones durante las transacciones.

Aislamiento:

Ofrece niveles de aislamiento robustos para prevenir interferencias entre transacciones concurrentes.

Durabilidad:

Utiliza técnicas avanzadas de registro de transacciones para asegurar que los datos sean persistentes una vez confirmados.

Eficiencia en Consultas Complejas:

Muy eficiente en la ejecución de consultas SQL complejas y en la gestión de grandes volúmenes de datos.

Seguridad:

Proporciona características de seguridad avanzadas, incluyendo cifrado de datos, auditoría y control de acceso granular.

Desventajas:

Costos:

Las licencias y el soporte pueden ser muy costosos, lo que puede ser prohibitivo para pequeñas y medianas empresas.

Complejidad de Configuración:

La configuración y el mantenimiento pueden ser complejos y requerir personal altamente capacitado.

Escalabilidad Horizontal:

Aunque ofrece soluciones de escalabilidad, pueden ser más complejas y menos flexibles en comparación con bases de datos NoSQL.

Rendimiento en Escrituras Masivas:

Puede no ser tan eficiente como algunas bases de datos NoSQL para operaciones masivas de escritura.

Curva de Aprendizaje:

Requiere un conocimiento profundo y especializado para maximizar su potencial y rendimiento.

Resumen Adicional

Eficiencia:

MongoDB: Muy eficiente en operaciones de lectura y escritura con índices adecuados, pero puede ser intensivo en memoria.

PostgreSQL: Muy eficiente en consultas SQL complejas y optimización de consultas.

Oracle: Muy eficiente en consultas complejas y gestión de grandes volúmenes de datos.

Seguridad:

MongoDB: Mejoras significativas en seguridad, pero históricamente ha tenido vulnerabilidades.

PostgreSQL: Ofrece robustas características de seguridad.

Oracle: Proporciona características de seguridad avanzadas y control de acceso granular.
Escalabilidad:

MongoDB: Buena escalabilidad horizontal con sharding.

PostgreSQL: Escalabilidad horizontal más complicada que en NoSQL.

Oracle: Ofrece soluciones de escalabilidad, pero pueden ser más complejas.

	Propiedades ACID	Eficiencia	Seguridad	Escalabilidad
Postgresql	Cumple con todas las propiedades ACID	gran eficiencia en consultas complejas y bueno para optimizar consultas sobre datos estructurados	robusta a nivel de acceso a tablas y datos	difícil de escalar
Oracle	Cumple con todas las propiedades ACID	eficiente con consultas complejas y grandes volúmenes de datos estructurados	robusta incluso a nivel granular	pocas soluciones útiles de escalabilidad
Mongodb	Cumple con todas las propiedades ACID	bueno en lectura y escritura pero con alto coste de memoria	buena seguridad actualmente sufrió problemas en el pasado	buena escalabilidad gracias al sharding

aunque como podemos observar pareciera que las bases de datos relacionales son mejores para esta clase de sistemas podemos observar un detalle importante son mejores siempre y cuando los datos están estructurados cosa que aquí no es el caso por lo que seguiremos usando mongodb

Razones para usar mongodb para este aplicativo

razones por la que se prefirió mongo db por encima de otras bases de datos:

- enfoque en el rendimiento sobre consultas de modificación y agregación a diferencia de las bases de datos sql donde se enfoca más al rendimiento de consultas complejas y sistema transaccionales.

- esquemas flexibles donde se pueden agregar datos a libertad sin tener la necesidad de modificar todos los esquemas lo cual lo hace más fácil de adaptar a los distintos requerimientos y menos costoso en cuestiones de tiempo al realizar cambios sobre los requerimientos

- escalabilidad progresiva más fácil de aplicar dado el uso de clusters que nos permiten añadir recursos sobre diferentes servidores distribuidos mientras que la mayoría de bd sql solo escalan de manera vertical de manera eficiente (el escalamiento horizontal es muchísimo mas costoso que en mongodb) lo cual lo hace complejo cuando se tienen grandes volúmenes de datos.

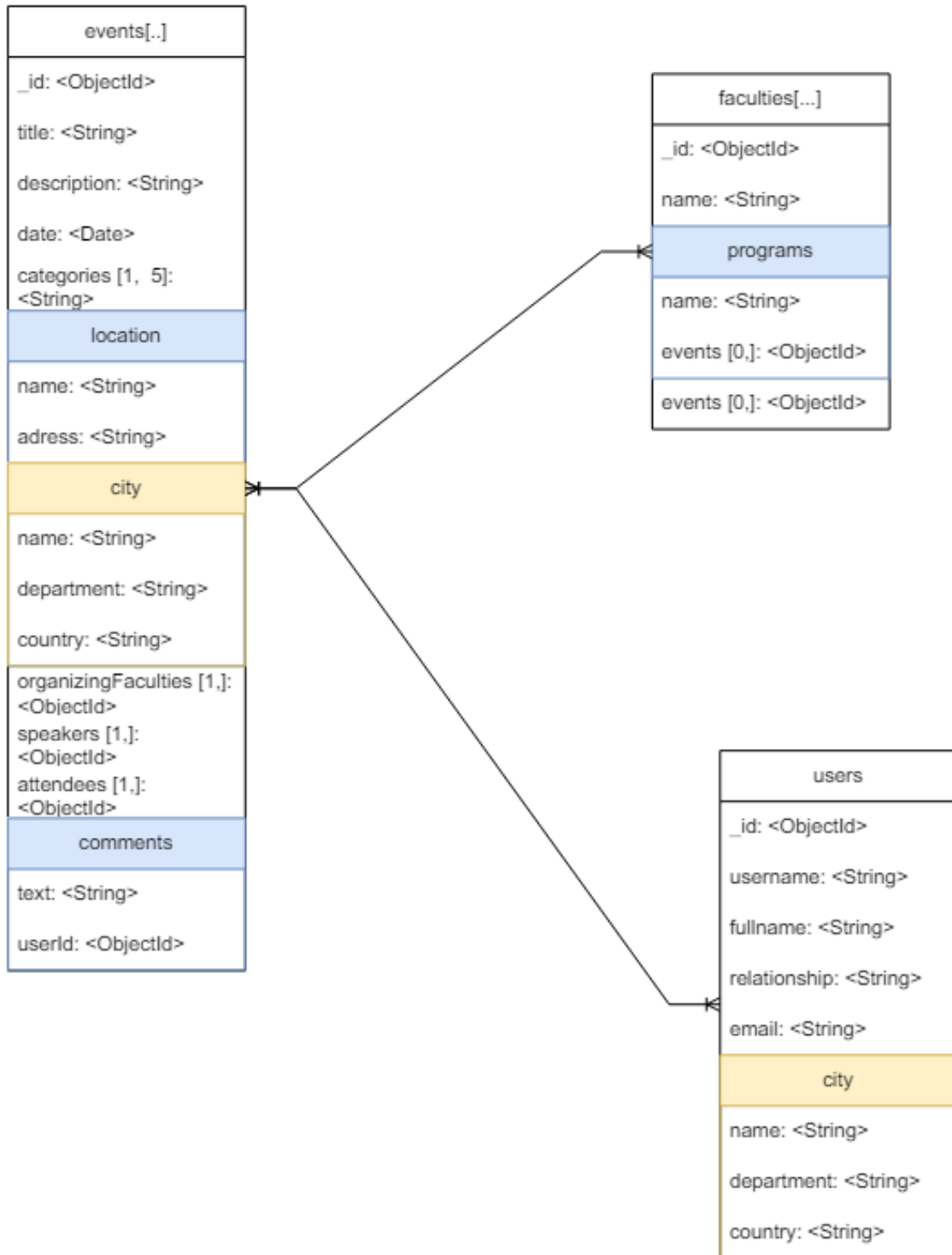
- costo mongodb es económico a nivel de licencia manteniendo aun así un gran nivel de rendimiento además de contar con un soporte de gran calidad pagando un costo reducido.

- manejo de datos desestructurados de manera sencilla ya que mongodb está diseñado para este tipo de datos tales como los comentarios y descripciones de los eventos que varían en tamaño, formato y contenido .

- seguridad: en mongodb se cuentan con capas relativamente robustas de seguridad que no son implementadas en todas las db sql del mercado (al menos no aquellas soluciones que compiten en el rango de precios de mongodb) tales como la autenticación del tipo SCRAM-SHA-256, LDAP para autenticación dentro de entornos empresariales. Cuenta también con características a nivel de bloqueos de ip pues solo se podrán realizar operaciones sobre la db aquellas ips que estén en la whitelist de ese cluster, mongodb también utiliza RBAC desde su gestor atlas donde se pueden asignar roles por ip dándoles ciertas características a cada rol. Por último mongodb cuenta con dos clases de cifrado mientras que la mayoría de bds relacionales solo cuentan con 1, mongodb cuenta con: cifrado en tránsito con soporte TSL/SSL y cifrado en reposo por lo general con el uso de KMIP.

aunque la seguridad sea un punto altamente destacable no se aprovechara realmente el potencial de mongodb aquí dado el alcance del proyecto.

Modelo de datos



Dado los requerimientos obtenidos y lo mencionado en la sección “Razones para usar mongodb para esta aplicación” se ha concluido que este modelo es el mejor para las necesidades del cliente dado que el mismo no requería de sistemas robustos de seguridad ni nada similar se ha hecho alarde única y

exclusivamente de la eficiencia a nivel de consultas y de escritura de mongo sobre datos no estructurados como los comentarios.

Ventajas y desventajas del modelo de datos

para comparar las ventajas y desventajas de este modelo se compara con otra solución que sería tener los documentos relacionados más no embebidos:

(Documentos Embebidos)

Ventajas:

Lectura Rápida:

Consulta de Datos Completa: Toda la información relacionada con un evento (título, descripción, categorías, fecha, lugar, organizadores, asistentes, comentarios) se encuentra en un único documento, lo que permite recuperar todos los detalles del evento con una sola consulta.

Eficiencia de Lectura: Al tener los datos embebidos, se reduce la necesidad de realizar múltiples consultas o uniones entre diferentes colecciones, mejorando la velocidad de lectura para consultas comunes.

Simplicidad de Estructura:

Modelo Intuitivo: La estructura del documento refleja directamente la estructura de los datos que se manejan en la aplicación, lo que facilita la comprensión y el desarrollo.

Desarrollo Rápido: Al no necesitar relaciones complejas, el desarrollo y la implementación pueden ser más rápidos y sencillos.

Atomicidad y Consistencia Local:

Actualizaciones Atómicas: Las actualizaciones a los datos embebidos dentro de un único documento son atómicas, garantizando que las modificaciones se aplican de manera coherente.

Menor Riesgo de Inconsistencia: Al tener los datos relacionados en un solo documento, se minimiza el riesgo de inconsistencias entre documentos relacionados.

Menor Sobrecarga de Red:

Reducción de Llamadas: Menos llamadas a la base de datos y menos tráfico de red ya que se necesitan menos consultas para obtener datos completos del evento.

Facilidad de Indexación:

Índices Simples: Es más fácil y eficiente crear índices en campos dentro de un documento que en múltiples colecciones relacionadas.

Desventajas:

Duplicación de Datos:

Redundancia: La información de usuarios, facultades y programas se puede duplicar en múltiples documentos, lo que aumenta el espacio de almacenamiento y puede llevar a inconsistencias si los datos no se actualizan correctamente en todos los documentos donde aparecen.

Tamaño del Documento:

Documentos Grandes: Los documentos pueden crecer significativamente en tamaño, especialmente para eventos con muchos asistentes, comentarios o relaciones, lo que puede afectar el rendimiento y el límite de tamaño de documentos de MongoDB (16MB).

Actualizaciones Complejas:

Dificultad en Modificaciones: Las actualizaciones en datos embebidos pueden ser más complejas y menos eficientes, especialmente si los datos están profundamente anidados.
Eficiencia en Operaciones de Escritura:

Sobrecarga de Escritura: Cada vez que se actualiza una sección del documento, todo el documento debe ser reescrito, lo que puede ser ineficiente para documentos grandes.

Mantenimiento y Escalabilidad:

Mantenimiento de Datos: La duplicación de datos requiere mantener la consistencia manualmente, lo que puede ser propenso a errores y aumentar la complejidad del mantenimiento.

Escalabilidad Limitada: Aunque MongoDB maneja bien la escalabilidad horizontal, documentos muy grandes o muy anidados pueden ser difíciles de manejar y optimizar en un clúster distribuido.

Modelo Basado en Relaciones (Referencias)

Ventajas:

Normalización:

Evita Redundancia: Minimiza la duplicación de datos al almacenar información común (como usuarios, facultades y programas) en colecciones separadas.

Menor Tamaño de Documento: Mantiene los documentos más pequeños y manejables, ya que solo se almacenan las referencias a otros documentos.

Flexibilidad y Mantenimiento:

Actualizaciones Sencillas: Actualizar información centralizada (como detalles del usuario) es más sencillo y requiere menos cambios en la base de datos.

Consistencia: Garantiza que todos los documentos referenciados tienen información actualizada y consistente.

Integración con RDBMS:

Facilidad de Sincronización: Facilita la sincronización con la base de datos relacional, ya que los datos pueden estar más alineados con las estructuras normalizadas del RDBMS.

Integridad Referencial: Aunque MongoDB no garantiza integridad referencial, tener datos normalizados y referenciados puede acercar más el modelo de datos a un esquema relacional.

Escalabilidad:

Distribución de Carga: Distribuye la carga de datos en múltiples colecciones, lo que puede mejorar el rendimiento en consultas que no necesitan acceder a datos embebidos.

Menor Impacto de Sharding: Las colecciones más pequeñas y normalizadas pueden facilitar la distribución de datos y el equilibrio de carga en un entorno de sharding.

Desventajas:

Complejidad en las Consultas:

Consultas Más Lentas: Las consultas pueden ser más lentas debido a la necesidad de realizar múltiples consultas y joins entre colecciones.

Mayor Latencia: Aumenta la latencia de las operaciones debido a la necesidad de buscar en múltiples colecciones.

Mayor Complejidad de Desarrollo:

Gestión de Relaciones: Requiere una gestión manual de las relaciones entre colecciones, lo que puede aumentar la complejidad del código y la posibilidad de errores.

Transacciones Limitadas:

Soporte de Transacciones: Aunque MongoDB soporta transacciones multi-documento, su rendimiento puede no ser tan eficiente como las transacciones simples en un documento único.

Rendimiento: Las transacciones que abarcan múltiples documentos y colecciones pueden tener un impacto en el rendimiento general.

Consistencia Eventual:

Sincronización: Mantener la sincronización de datos referenciados puede ser desafiante, especialmente en sistemas distribuidos donde la consistencia eventual es un problema.

Inconsistencias Temporales: Pueden ocurrir inconsistencias temporales debido a la latencia en la propagación de actualizaciones.

a modo de conclusión se prefirió un modelo basado en documentos embebidos por que este esta mas enfocado a mejorar la eficiencia pues como se sabe al cargar en memoria estos documentos los mismos se encontraran espacialmente más cercanos por lo que podríamos aprovechar mejor los recursos propuestos por el cluster como su memoria caché.

Estrategia a seguir

para implementar el modelo anteriormente seleccionado haremos acopio de la siguiente estrategia:

- construiremos los modelos respectivos para cada una de las bases de datos
- a estos modelos ya creados se les agregaran índices que mejoren sus tiempos de lectura y sacrificaremos un poco la escritura pero estos índices serán hechos sobre documentos y tablas intensivas en lectura y no escritura como lo pueden ser los eventos pues estos no se hacen constantemente mientras que por ejemplo los comentarios si sufren de este problema.
- seguidamente se implementará un sistema web para realizar las HU solicitadas por el cliente pues es la manera más efectiva de dar gran alcance a la aplicación.
- se solventara el tema de costo haciendo uso de los clusters limitados y la base de datos limitada de oracle

- después de esto se realizará todo lo que tenga que ver con acceder y cambiar la base de datos nosql
- posteriormente se realizará todo el tema de consultas conjuntas dentro del sistema o sea la realización de consultas que involucren a ambas bases de datos
- finalmente se realizará todo el tema de extracción de información de ambas bases de datos y mostrarlas conjuntamente ejemplo mostrar los detalles de los asistentes que a su vez son empleados.

Implementación de la estrategia

para la implementación de la estrategia se hará uso de las siguientes herramientas:

- Python 3.9: un lenguaje de programación multiparadigma que tiene la particularidad de no ser fuertemente tipado y de compilar en ejecución.
- Django 3.0: Django es un framework web para trabajar con python que cimienta su filosofía en realizar los trabajos de un middleware y dejar que el desarrollador se enfoque en realizar la aplicación en sí.
- cx_oracle: cx_Oracle es un módulo de extensión de Python que permite el acceso a la base de datos Oracle.
- PyMongo: PyMongo es una distribución de Python que contiene herramientas para trabajar con MongoDB y es la forma recomendada de trabajar con MongoDB desde Python. Se eligió esta por encima de otras librerías dada su extensa documentación y gran compatibilidad con el framework que se está usando.

el modelo realizado a partir del modelo diseñado en mongo es el siguiente:

```
collections.events
{
  "_id": ObjectId(),
  "title": "Título del Evento",
  "description": "Descripción del Evento",
  "categories": ["categoría1", "categoría2"],
  "date": ISODate("2024-05-25T09:00:00Z"),
  "location": {
    "name": "Nombre del Lugar",
    "address": "Dirección del Lugar",
    "city": {
      "name": "Nombre de la Ciudad",
      "department": "Departamento",
      "country": "País"
    }
  }
},
```

```

"organizingFaculties": [
  {
    "facultyId": ObjectId(), // Referencia a la colección faculties
    "name": "Nombre de la Facultad"
  }
],
"organizingPrograms": [
  {
    "programId": ObjectId(), // Referencia al programa
    "name": "Nombre del Programa"
  }
],
"speakers": [
  {
    "userId": ObjectId(), // Referencia a la colección users
    "fullName": "Nombre Completo del Conferencista"
  }
],
"attendees": [
  {
    "userId": ObjectId(), // Referencia a la colección users
    "fullName": "Nombre Completo del Asistente"
  }
],
"comments": [
  {
    "text": "Texto del comentario",
    "userId": ObjectId() // Referencia a la colección users
  }
]
}

```

collections.users

```

{
  "_id": ObjectId(),
  "username": "nombreUsuario",
  "fullName": "Nombre Completo",
  "relationship": "profesor",
  "email": "usuario@correo.com",
  "city": {
    "name": "Nombre de la Ciudad",
    "department": "Departamento",
    "country": "País"
  }
}

```

collecioions.faculties

```

{

```

```

    "_id": ObjectId(),
    "name": "Nombre de la Facultad"
  }

```

collections.programs

```

{
  "_id": ObjectId(),
  "name": "Nombre del Programa",
  "facultyId": ObjectId() // Referencia a la colección faculties
}

```

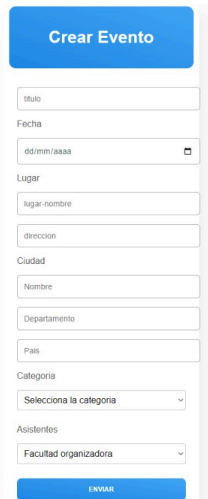
Estos son los índices que se agregaron para mejorar el rendimiento:

```

db.events.createIndex({ categories: 1 });
db.events.createIndex({ date: 1 });
db.events.createIndex({ "location.city.name": 1 });
db.events.createIndex({ "organizingFaculties.name": 1 });
db.events.createIndex({ "organizingPrograms.name": 1 });
db.events.createIndex({ "attendees.userId": 1 });

```

En cuanto al aplicativo en sí, se realizaron primeramente todos los formularios para creación y modificación de los diferentes elementos.



El formulario 'Crear Evento' contiene los siguientes campos:

- Título:
- Fecha: (con icono de calendario)
- Lugar:
- Lugar nombre:
- Dirección:
- Ciudad:
- Nombre:
- Departamento:
- País:
- Categoría: (menú desplegable)
- Asistentes: (menú desplegable)

En la parte superior hay un botón azul 'Crear Evento' y en la parte inferior un botón azul 'ENVIAR'.

este paso va desde la creación del código interno (en django se llama view) hasta la parte de interacción con el usuario (html y css)

Posteriormente se realizó todo el código relacionado con el listado de objetos que se encuentren en una sola base de datos por ejemplo eventos.

Adicionalmente y en paralelo se implementó la extracción de información de la base de datos relacional que luego será usada para realizar consultas conjuntas.

Finalmente se realizó todo el tema de consultas conjuntas y para generar resultados que involucren a ambas bases de datos así mismo pues se implementó los temas de detallado de la información específica

Conclusiones

El desarrollo de esta aplicación proporciona una plataforma robusta para la gestión integral de eventos universitarios, mejorando la eficiencia administrativa y la experiencia del usuario. La validación rigurosa de datos y las confirmaciones previas aseguran la integridad de la información gestionada. Las funcionalidades de visualización detallada y sugerencias personalizadas nos ayudaron a comprender el funcionamiento de algoritmos de sugerencia simples basándonos en la información de un usuario en específico.

El proyecto cumplió con los objetivos establecidos en las historias de usuario y los criterios de aceptación, proporcionando una herramienta valiosa para la gestión de eventos universitarios.

A modo de conclusión el desarrollo de esta aplicación nos permite conocer a profundidad cómo se desarrolla en mongo y como es realizar trabajo conjunto con sistemas legado, además de ayudarnos a concretar el tema de propiedades ACID y su importancia además de conocer bien los puntos fuertes y débiles de cada uno de los sistemas de bases de datos que día a día nos prestan servicio en las diferentes aplicaciones que usamos.

Referencias bibliográficas

Cielen, D., Meysman, A., & Ali, M. (2016). *Introducing Data Science: Big Data. Machine Learning and More, Using Python Tools*. Manning, Shelter Island, US, 322

Harrison, G. (2015). *Next Generation Databases: NoSQL and Big Data*. Apress.

DataStax. (s. f.). *DataStax Docs*. DataStax Documentation.

<https://docs.datastax.com/en/home/index.html>

CICS TX 11.1.0. (s. f.).

<https://www.ibm.com/docs/es/cics-tx/11.1?topic=processing-acid-properties-transactions>

¿Qué es una base de datos NoSQL? | IBM. (s. f.).

<https://www.ibm.com/es-es/topics/nosql-databases#:~:text=NoSQL%2C%20tambi%C3%A9n%20conocido%20como%20%22no,las%20bases%20de%20datos%20relacionales>

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

<https://www.acens.com/comunicacion/wp-content/images/2014/02/bbdd-nosql-wp-acens.pdf>.

Walther. (2023, 24 agosto). *SQL vs NoSQL: Ventajas y Desventajas Detalladas de Cada*

Sistema. Tutoriales Dongee.

<https://www.dongee.com/tutoriales/sql-vs-nosql-ventajas-y-desventajas-detalladas-de-cada-sistema/>

