CouchDB vs. RavenDB: Una Comparativa para el Manejo de Citas en Odontología

Elizabeth Araya, Isaac Barrantes, Guillermo Castillo, Miguel Obregón
Escuela de Estadística, Universidad de Costa Rica
San José, Costa Rica
elizabeth.arayamurillo@ucr.ac.cr
isaac.barrantesramirez@ucr.ac.cr
guillermo.castillo@ucr.ac.cr
miguel.obregonzuniga@ucr.ac.cr

***Abstract*—En respuesta a los desafíos de gestionar citas perdidas, sustituciones de pacientes y optimización de los recursos en el Servicio de Odontología del Área de Salud Alajuela Norte, este trabajo propone el uso de bases de datos no estructurales de tipo documento como solución al problema debido a su flexibilidad, escalabilidad y capacidad de manejar grandes volúmenes de datos de manera eficiente. Se analizaron en CouchDB y RavenDB aspectos como la consistencia, facilidad de uso, inserción, herramientas de gestión, recuperación de datos, entre otros. Se concluye que ambos software son opciones viables, cada uno destacando en distintos aspectos, CouchDB por su simplicidad en la manipulación de documentos cuando su estructura no es muy similar y RavenDB por su flexibilidad y velocidad en consultas.**

Index terms: Bases de datos NoSQL, gestión de citas, facilidad de uso, documentos.

I. INTRODUCCIÓN

El uso apropiado de las tecnologías de datos es fundamental para la era actual, en donde todos podemos conectarnos con la necesidad de pocos clicks. Sin embargo, con la ausencia de estos es complicado realizar una correcta toma de decisiones. Por mencionar uno de ellos, está el modelo SQL el que facilitó la tarea del programado desde 1970 hasta la actualidad [1]. Además de mejorar considerablemente el manejo de bases de datos y basarse en las operaciones ACID, dichas se consideran muy firmes a la hora de brindar seguridad de los datos.

Como contraparte de los sistemas SQL se conocen los noSQL que se conocen por su versatilidad y comodidades que los sistemas relacionales no poseen. Las bases de datos NoSQL no requieren satisfacer las operaciones ACID, por lo que no es necesario soportar directamente el álgebra relacional [2]. El uso de estas estructuras de datos siempre depende del tipo de datos y de los requisitos de procesamiento requeridos. En vista de esto, el propósito de este trabajo es comparar las similitudes y diferencias entre la implementación de bases de datos NOSQL utilizando los dos sistemas de almacenamiento BD.

Es importante poder comparar el rendimiento de las bases de datos de tipo documentos, los cuales son sistemas de almacenamiento BD, que están construidos para ofrecer prestaciones de alto rendimiento ideales para el manejo de volúmenes masivos de información [3].

En el siguiente repositorio de github se pueden encontrar los archivos de datos utilizados, así como el código para generarlos, además de archivos pdf con instrucciones para instalar los software, crear bases e importar los datos: https://github.com/MiguelObregon/NoSQL_XS3210

II. PLANTEAMIENTO DEL PROBLEMA

El Servicio de Odontología del Área de Salud Alajuela Norte enfrenta problemas para gestionar de forma eficiente sus actividades diarias, especialmente en lo que respecta a citas perdidas, sustitución de pacientes y horas no utilizadas en los servicios de consulta externa y procedimientos. Esto genera una gran ineficiencia en el uso de los recursos y una disminución en la satisfacción de los pacientes. Además, la ausencia de una herramienta que centralice la información sobre citas, inasistencias y sustituciones complica la toma de decisiones basadas en los datos. Por lo tanto, el uso de una base de datos de tipo documento permite almacenar y gestionar esta información de manera flexible y eficiente, mejorando así el control sobre los recursos y la programación de citas [4].

III. PROCESO DE LA SOLUCIÓN IMPLEMENTADA

Como solución a dicho problema, se planteó la construcción de una base de datos no estructurada que ya facilita y vuelve más rápido el proceso de lectura y escritura de datos masivos, lo que es ideal para el seguimiento de inasistencias y citas sustituidas.

Del mismo modo, los datos pueden estar semiestructurados, lo que permite agregar o modificar campos de forma más flexible sin alterar toda la estructura de la base. A su vez, hace que la base de datos sea más manejable ya que almacena todo en un solo documento, lo que reduce hacer joins o consultas relacionales complejas. También, permite escalar horizontalmente por lo cual se puede agregar más servidores lo que facilita el manejo de grandes volúmenes de datos y consultas distribuidas, por ejemplo, el historial de citas y consultas de pacientes [5].

La construcción de dicha base se realizó en dos software diferentes que trabajan con bases de datos de tipo documento, los cuales fueron: CouchDB y RavenDB. Se seleccionó CouchDB porque permite almacenar y recopilar datos de este tipo, además de que implementa modelos de datos sin esquema, que simplifica la gestión de registros [6], asimismo, se decidió seleccionar RavenDB porque utiliza motores que hacen que las consultas sean más rápidas y eficientes, de la misma forma facilita la manipulación y extracción de datos de manera intuitiva y flexible [7].

Para la construcción de la base de datos en ambos software, se codificaron 400 documentos, los cuales se

dividen en 100 documentos para cada tipo (citas, pacientes, inasistencias y sustituciones). Para uniformizar las propiedades se mantuvo el mismo identificador en todos los documentos, el cual se utilizó para diferenciar el tipo de documento, este se añadió la como "tipo_documento" .

Cabe resaltar que tanto en CouchDB como RavenDB se logró visualizar que la base de datos de tipo documento es una solución viable para el problema expuesto. Ambos software tuvieron un comportamiento similar, sin embargo, CouchDB resultó tener un poco más de facilidad en la inserción de datos tipo JSON, aunque en gran escala haya que usar el cmd.

IV. COMPARACIÓN ENTRE COUCHDB Y RAVENDB

En esta sección se analizarán por diversos criterios las similitudes, diferencias, ventajas o desventajas entre CouchDB y RavenDB.

A. Consistencia

CouchDB implementa un modelo de consistencia eventual, lo que significa que los cambios realizados en un nodo no se reflejan de inmediato en todos los nodos, sino que estos cambios se propagan con el tiempo hasta que todos los nodos alcanzan un estado consistente. Este enfoque es útil en sistemas distribuidos que priorizan la disponibilidad sobre la consistencia inmediata, lo que está alineado con el teorema CAP (Consistencia, Disponibilidad y Tolerancia a Particiones) [8].

Para gestionar las actualizaciones concurrentes, CouchDB utiliza Multiversion Concurrency Control (MVCC) para gestionar las actualizaciones concurrentes sin necesidad de bloqueos. Cada vez que se modifica un documento, se crea una nueva versión del mismo, lo que permite que varios usuarios trabajen con versiones diferentes sin interferir entre sí. Esto mejora la disponibilidad del sistema bajo cargas elevadas, pero requiere que los conflictos se gestionan manualmente o mediante la lógica de la aplicación[8].

Las propiedades ACID en CouchDB solo se aplican a nivel de documento individual. Esto significa que las operaciones que involucran un único documento (como crear, editar o eliminar) son atómicas, consistentes, aisladas y duraderas. Cada operación en un documento se realiza en su totalidad o no se realiza en absoluto, lo que garantiza que no haya estados intermedios o parciales en la base de datos. Esta capacidad se logra a través del uso de Multiversion Concurrency Control (MVCC), que permite mantener múltiples versiones de un documento, asegurando que las actualizaciones concurrentes no bloqueen el acceso a los datos[9].

Por otro lado, RavenDB, ofrece una mayor flexibilidad, permitiendo configuraciones tanto de consistencia eventual como de consistencia fuerte, lo que le otorga una ventaja en aplicaciones donde se requieren diferentes niveles de consistencia.

La gestión de las consultas en RavenDB se hacen mediante un almacén de índices. A medida que se escriben los documentos, se copian datos del almacén de documentos al almacén de índices para que las consultas utilicen esos índices. Sin embargo, las consultas pueden acceder a índices desactualizados si los datos aún no se han sincronizado completamente entre los nodos. Esto permite que el sistema priorice la velocidad de respuesta y la disponibilidad, a costa de una consistencia inmediata.

Además, RavenDB permite a los usuarios ajustar el nivel de consistencia según el tipo de operación. Por ejemplo, un cliente que necesite ver los datos más recientes puede configurar la consulta para que espere hasta que los índices estén actualizados, mientras que en otros casos, la velocidad puede ser la prioridad, accediendo a índices no actualizados[10]. Esto hace que sea ideal para aplicaciones que necesitan un equilibrio entre disponibilidad y consistencia.

Con relación a lo anterior en la documentación de RavenDB, se menciona la consistencia fuerte en el contexto de garantizar que los cambios se reflejen de manera inmediata en todos los nodos cuando se configura de esta forma. Esto es esencial en aplicaciones donde la precisión de los datos es crítica y no se puede tolerar la inconsistencia temporal. Esta capacidad de ofrecer consistencia fuerte asegura que los datos sean uniformes en todo el sistema, sin depender del tiempo para sincronizarse [11].

La consistencia eventual de CouchDB se adapta mejor a sistemas distribuidos que priorizan la disponibilidad sobre la sincronización inmediata de los datos. Este modelo es adecuado para aplicaciones donde los datos pueden propagarse gradualmente y no requieren consistencia instantánea en todos los nodos. Por otro lado, RavenDB ofrece más flexibilidad al permitir tanto consistencia eventual como consistencia fuerte, lo que lo hace más adecuado para aplicaciones críticas donde la precisión de los datos es esencial y no se puede tolerar la inconsistencia temporal.

B. Inserción de datos

En cuanto a cómo se importan los datos se tiene que a pesar de que CouchDB y RavenDB trabajan con documentos en formato JSON, sus formas de inserción de información no son precisamente las mismas cuando se trabaja con cantidades grandes de documentos. Empezando por CouchDB se encontró que es posible insertar documentos uno por uno en la base desde la interfaz gráfica, sin embargo, esta forma de inserción podría ser práctica o rentable solo cuando se trabaje con bases pequeñas, si la base tiene muchos documentos se incrementa el tiempo que se necesitaría para insertar todos los documentos, lo cual es lo opuesto a lo que se busca ahora en una época de competitividad y de optimización en la gestión de bases de datos.

Cuando la base de datos ya es muy grande se busca realizar una inserción de documentos en masa. Para poder insertar los documentos en CouchDB de esta manera se debe hacer uso del cmd del ordenador. Se debe proporcionar un archivo JSON cuya estructura sea un arreglo de los documentos que se quieren ingresar en la base, si los documentos son de diferentes tipos o estructuras lo más recomendable es incluir un identificador para saber cuál tipo de documento es y poder diferenciarlos en la base [12]. Esta opción requiere del uso de un comando específico que se indica en el documento de cómo crear y cargar datos en CouchDB que se encuentra en el repositorio y puede resultar no tan simple de implementar a la primera, ya que si no se indican bien los datos de usuario de CouchDB o se indica mal la ubicación del archivo JSON va a tirar errores lo que puede dificultar el proceso de inserción para una persona que no esté tan familiarizada con el uso del cmd.

En cuanto a RavenDB tenemos que de igual manera, una de sus formas de inserción de información es mediante la

creación de documentos en su interfaz uno por uno, sin embargo, vuelve a presentar el factor de que si la base es muy grande no sería la forma más óptima de realizar la inserción, aunque por su interfaz pueda inclinar a los usuarios a hacerlo, pero de nuevo se tiene el asunto del tiempo.

Para insertar una cantidad de documentos mayor en RavenDB hay algunas opciones que se encuentran en su interfaz, las cuales se observan en una sección del archivo pdf dentro del repositorio, pero ninguna de estas es para un archivo JSON, aunque sí permite archivos CSV. En RavenDB se pueden insertar los documentos de un mismo tipo como una colección, ya que al trabajar con archivos CSV se esperaría que tengan una cierta uniformidad en cuanto a su estructura y por lo tanto si se tiene un archivo JSON para los documentos de cada tipo entonces una de las formas es convertir cada archivo JSON a CSV y posteriormente realizar su inserción en RavenDB. Esa sería como una de las limitantes de este último, que para los archivos JSON no tiene una opción directa en su interfaz para cargar documentos en masa, requeriría una programación más compleja.

C. Recuperación de datos

CouchDB se destaca por su capacidad de replicación bidireccional (master-master), lo que permite que varios nodos sincronizados mantengan sus datos actualizados incluso en escenarios offline. Esta característica es esencial para aplicaciones distribuidas o móviles que requieren sincronización de datos sin interrupciones cuando los dispositivos vuelven a estar en línea. Esta capacidad asegura la recuperación de datos sin pérdida en caso de fallos en alguno de los nodos. Otra ventaja es en las aplicaciones que requieren funcionamiento offline, ya que permite que los usuarios continúen operando y sincronizando datos localmente, lo que facilita la recuperación de datos cuando el dispositivo se reconecta [8].

Por su parte, RavenDB tiene un sistema robusto de respaldos completos e incrementales que permite restaurar fácilmente la base de datos en caso de fallos. Además, ofrece opciones para configurar planes automáticos de respaldo, asegurando que los datos estén siempre actualizados y listos para una recuperación rápida y eficiente.

Una característica relevante es la capacidad de restaurar datos en un nuevo clúster o en un entorno diferente, lo que asegura la integridad de los datos incluso en casos de fallos completos del sistema. Esta flexibilidad lo hace ideal para entornos empresariales que requieren alta disponibilidad [13]. En conclusión, CouchDB es más adecuado para entornos distribuidos con necesidades de sincronización activa entre nodos, mientras que RavenDB es la mejor opción cuando se requiere recuperación rápida y resiliencia ante fallos críticos.

D. Facilidad de uso

En el caso de CouchDB se podría decir que sin tomar en cuenta la parte de inserción de datos en masa mediante el cmd, realmente llega a tener un nivel de uso que es fácil para un usuario una vez estén ingresados los datos, ya que la interfaz cuenta con identificación de qué es cada pestaña, también de cómo ver las bases de datos, cómo mostrar u ocultar algunas columnas y hasta de cómo poder llegar a modificar un documento. El estar trabajando con documentos permite que se puedan agregar o eliminar elementos de un documento único sin alterar la estructura de otros documentos a diferencia de los modelos relacionales donde hay más restricciones para este tipo de acciones ya que modificaría la estructura inicial con la que fue planteada.

Para RavenDB, su interfaz es muy moderna e intuitiva y permite que sea fácil de utilizar en cuanto a la visualización de los documentos, edición y creación de nuevos valores en los mismos, así como una identificación de las diferentes colecciones de datos que se tengan. Su inserción en masa también se facilita al no tener que utilizar código mediante el cmd. De igual manera se pueden editar documentos sin alterar a los demás.

Con respecto a la facilidad en la realización de consultas tanto CouchDB como RavenDB tienen sus elementos que dependen de si la persona tiene un cierto conocimiento en el lenguaje SQL a pesar de tratarse de bases NoSQL, pero esto se detallará más en la sección de Consultas, donde se explicará cómo se pueden realizar y de si puede o no llegar a ser muy complejo.

E. Herramientas de Gestión

Empezando por CouchDB se tiene que su interfaz, la cual ya se ha lleva como nombre Fauxton, la cual ya viene integrada en la instalación de CouchDB. Su interfaz como ya se ha mencionado en la sección de facilidad de uso permite la capacidad para crear, actualizar, eliminar y ver documentos, así como acceso a parámetros de configuración y una interfaz para replicación [15]. Fauxton igual permite crear bases de datos y también tiene una característica relacionada con la gestión de consultas, que es el uso de Vistas, estas se mencionan un poco más en la sección de consultas. De igual manera Fauxton permite que los administradores de las bases puedan tener un mejor control, seguridad y manejo de las mismas.

En cuanto a RavenDB su principal herramienta de gestión es la misma interfaz RavenDB Document Database Management Studio. Tiene la ventaja de ser incluida en las licencias que ofrece RavenDB al realizar su instalación. Sin embargo, la licencia gratis tiene un período de duración de 6 meses. De igual manera se puede ver que su interfaz es muy interactiva y el usuario puede comprender rápidamente cómo utilizarla. Además esta interfaz permite monitorizar métricas operacionales y de rendimiento en las bases de datos, también permite ver cuánto espacio y memoria del disco se está utilizando gracias a sus cluster dashboards [16]. De igual manera que CouchDB permite crear, modificar, eliminar y ver documentos a nivel de interfaz. Desde este punto de vista la interfaz de RavenDB llega a ser un poco más potente en cuanto a indicadores para el usuario a nivel visual.

F. Método utilizado para hacer consultas

CouchDB cuenta con un par de formas de poder realizar consultas en los documentos. Una de ellas es por medio de las vistas que se mencionaron anteriormente, estas permiten filtrar documentos de forma que se pueda extraer información de interés de la base de datos para algún análisis o toma de decisiones. Las vistas en CouchDB se crean de manera dinámica, lo que quiere decir que no van a afectar almacenamientos de documentos, además hay diversas maneras en que se pueden realizar las vistas y se pueden replicar en varias partes de las bases de datos [17]. Sin embargo, la manera en que se realizan las consultas de esta manera implica que se haga uso del cmd ya que la vista

genera un código que se debe correr, por lo que de esta manera no se puede ver de inmediatamente el resultado de la vista en la interfaz, si no que será en el mismo cmd luego de correr el código, lo que puede ser incómodo de manera visual para algunos usuarios.

CouchDB también cuenta con Mango Query, una opción de lenguaje de consultas que fue diseñado específicamente para este programa y que sí permite realizar consultas de manera que se puedan observar en la misma interfaz de Fauxton los resultados de los documentos que se requieran en la consulta. Las consultas que se realizaron para los documentos de este artículo para CouchDB fueron realizadas con esta última opción ya que se pueden observar de manera más rápida los resultados y ver si se obtienen los resultados esperados. Pero es importante mencionar que este lenguaje es propio para este programa y por lo tanto requiere de un estudio previo de sus funciones ya que aunque tenga ciertas similitudes con el lenguaje SQL no es igual [18].

Por ejemplo, un WHERE de SQL acá se puede asociar con el comando selector, mientras que un SELECT para columnas sería como el comando fields en Mango Query. En el Anexo 1 se pueden observar algunas consultas realizadas con este lenguaje, las cuales para este caso de los datos de gestión de citas realmente no fueron muy complejas, pero si se llegara a tener una base de un caso más complejo sí se podría volver más largo de implementar el código para una consulta.

En el caso de RavenDB, este cuenta igual con su propio lenguaje de consultas, que se asimila a un lenguaje SQL, el cual tiene como nombre Raven Query Language. Las consultas se pueden implementar igual en la interfaz de RavenDB Studio y muestran los resultados de una vez al ejecutar la consulta. Si se tienen conocimientos en SQL se podrá aprender más rápido el lenguaje para las consultas ya que los comandos utilizados acá sí son muy similares como WHERE para filtrar, SELECT para las columnas deseadas. Además, el FROM también sirve para elegir de cuál de las colecciones se quieren seleccionar los documentos recordando que en RavenDB se pueden ingresar de esta manera los distintos tipos de documentos. De igual manera hay comandos similares como ORDER BY y GROUP BY, entre otros que se pueden consultar [19].

En el Anexo 2 se muestran algunos ejemplos de consultas. Igual se debe mencionar que si la persona no tiene conocimientos previos en SQL puede que su curva de aprendizaje sea más lenta para la realización de consultas en RavenDB, mientras que en CouchDB podría intentarlo con las vistas, pero aún así, las consultas tienden a ser más limitadas en CouchDB.

G. Almacenamiento

Los documentos JSON manejan un formato liviano y fácil de interpretar tanto para máquinas como para humanos [14]. Por esta razón se decidió que la codificación principal se realizará en este formato. Los documentos se identifican únicamente en forma de texto específico para cada documento, lo que le permite organizar la información en un contexto relevante. CouchDB crea automáticamente una nueva versión cada vez que se actualiza un documento.

Por otra parte, RavenDB también usa el formato JSON para su base de datos, pero utiliza técnicas para optimizar la base de datos. A diferencia de CouchDB, RavenDB no crea revisiones nativas en cada actualización. Más bien, garantiza que se mantenga la coherencia de los datos a nivel de documento para las transacciones ACID. Esto asegura que las operaciones de lectura y escritura sean fiables sin la sobrecarga de múltiples versiones.

Añadiendo que CouchDB no cuenta con una opción para exportar la base de datos de una manera simple, más bien se basa en replicación de los datos. A diferencia de RavenDB, este permite exportar y respaldar la base completa o partes específicas, facilitando la recuperación ante fallos y la migración de datos.

V. CONCLUSIONES

Después de analizar diferentes aspectos para comparar CouchDB y RavenDB como posibles opciones que aprovechan el uso de documentos para un sistema de gestión de citas de odontología se ha podido determinar que ambos llegan a ser opciones viables. Los dos tienen sus diferencias por ejemplo en la parte de inserción o recuperación de datos, pero también tienen sus similitudes en que ambos permiten a su manera aprovechar la estructura de los documentos y realizar las consultas o modificaciones que podrían permitir mejorar la gestión en la asignación de citas y a la vez encontrar indicadores de las principales causas que hacen que no se aprovechen los espacios al máximo o que se tengan que reprogramar.

Se debe tener siempre presente que estos softwares están en constante actualización, así como otros en la industria de los datos por lo que cada uno siempre buscará la manera de ser la primera opción para los distintos usuarios, pero como se puede ver en este caso, ambos logran contribuir de una u otra forma a un sistema de gestión de citas. Por lo que tomando en cuenta lo investigado se podría determinar que CouchDB se preferiría cuando la estructura de los datos no sea muy similar ya que permite un buen manejo y visualización de documentos con esta característica, mientras que RavenDB es una mejor opción cuando la estructura de los documentos sea más parecida por las colecciones.

REFERENCIAS

[1] J. Salazar Cárdenas, “Análisis comparativo de dos bases de datos SQL y dos bases de datos noSQL”. Pereira, Colombia: Universidad Tecnológica de Pereira, 2014. Disponible en: https://hdl.handle.net/11059/511

[2] D. Robles, S. Maury, R. Serrano, B. Ad, y D. H. Vizca, "¿Qué características tienen los esquemas NoSQL?," *Investigación y desarrollo en TIC*, vol. 6, no. 1, pp. 40-44, 2015.

[3] H. Saltos Viteri y M. Franco Bayas, “La transición del manejo de bases de datos entre el modelo SQL al NOSQL en la enseñanza de carreras tecnológicas”, *JSR*, vol. 5, n.º 1, pp. 29–48, ene. 2020.

[4] F. Tablado, “Bases de datos documentales. ¿Qué es? Tipos y ejemplos,” Ayuda Ley Protección Datos, Nov. 04, 2020. https://ayudaleyprotecciondatos.es/bases-de-datos/documentales/

[5] “Modelado de datos con Amazon DocumentDB,” Amazon Web Services, Inc. [Online]. Available: https://aws.amazon.com/es/nosql/document/

[6] “Explicación de Apache CouchDB”, *Ibm.com*, 26-abr-2023.

[7] M. B. Gil, “Análisis de las Ventajas y Desventajas de RavenDB: ¿La mejor opción para tu base de datos?”, *Ventajas y desventajas top*, 02-sep-2023. [En línea]. Disponible en: https://ventajasydesventajastop.com/ravendb-ventajas-y-desventajas/. [Consultado: 16-oct-2024].

[8] O'Reilly, *CouchDB: The Definitive Guide*, 2016. [Online]. Accedido en: Oct, 15, 2024. Available: https://www.oreilly.com/library/view/couchdb-the-definitive/9780596158156/ch02.html.

[9] Severalnines, "Battle of NoSQL databases: Comparing MongoDB and CouchDB," 2021. [Online]. Accedido en: Oct, 15, 2024. Available: https://severalnines.com/blog/battle-nosql-databases-comparing-mongodb-and-couchdb.

[10] RavenDB, "Understanding eventual consistency," *RavenDB Documentation*, 2023. [Online]. Accedido en: Oct, 15, 2024. Available: https://ravendb.net/docs/article-page/6.2/csharp/users-issues/understanding-eventual-consistency#understanding-eventual-consistency.

[11] RavenDB, "How to enable optimistic concurrency," *RavenDB Documentation*, 2022. [Online]. Accedido en: Oct, 15, 2023. Available: https://ravendb.net/docs/article-page/6.0/Csharp/client-api/session/configuration/how-to-enable-optimistic-concurrency.

[12] Apache Software Foundation, 1.3.2. /{db}/_all_docs. [Online]. Accedido en: Oct. 14, 2024. Available: https://docs.couchdb.org/en/stable/api/database/bulk-api.html

[13] RavenDB, "Backups and restores," *Inside RavenDB*, 2023. [Online]. Accedido en: Oct, 15, 2024. Available: https://ravendb.net/learn/inside-ravendb-book/reader/4.0/17-backups-and-restores#:~:text=RavenDB%20attempts%20to%20make%20the,select%20New%20database%20from%20backup%20

[14] D. Y. Chango Gavilánez y J. J. Jácome Salas, "Bases de datos no relacionales: Utilización de Mongo DB como base de datos no relacional empleando formato GEO JSON," *Bachelor's thesis*, 2016.

[15] CouchDB Apache Software Foundation, Fauxton Visual Guide. [Online]. [Accessed: Oct. 15, 2024]. [Available]: https://couchdb.apache.org/fauxton-visual-guide/index.html

[16] E. Oren. RavenDB NoSQL Management Studio GUI. [Online]. Accedido en: Oct, 15, 2024. Available: https://ravendb.net/articles/ravendb-nosql-document-database-management-studio-gui

[17] .IBM. “¿Qué es CouchDB?” [Online]. Accedido en: Oct 14, 2024. Available: https://www.ibm.com/es-es/topics/couchdb

[18] J. Soo. Query in Apache CouchDB: Mango Query [Online]. Accedido en: Oct 15, 2024. Available: https://dev.to/yenyih/query-in-apache-couchdb-mango-query-1fd

[19] RavenDB. RQL - Raven Query Language. [Online]. Accedido en: Oct, 13, 2024. Available: https://ravendb.net/docs/article-page/6.2/csharp/client-api/session/querying/what-is-rql

ANEXOS

Anexo 1. Consultas en CouchDB

-Para obtener las citas que han sido canceladas (muestra todas las columnas):

```
{
 "selector": {
  "tipo_documento": "cita",
  "estado": "cancelada"
 }
```

-Para obtener las citas que han sido canceladas (seleccionando solo las columnas indicadas):

```
{
 "selector": {
  "tipo_documento": "cita",
  "estado": "cancelada"
 },
 "fields": [
  "_id",
  "cita_id",
  "fecha_hora",
  "paciente_id",
  "estado"
 ]
}
```

-Cita de un paciente específico:

```
{
 "selector": {
  "tipo_documento": "cita",
  "paciente_id": "PAC00023"
 },
 "fields": [
  "_id",
  "cita_id",
  "fecha_hora",
  "paciente_id",
  "estado"
 ]
}
```

-Inasistencias antes de una fecha específica:

```
{
 "selector": {
  "tipo_documento": "inasistencia",
  "fecha_cita": {
   "$lt": "2024-11"
```

```
    }
  }
}
```

-Citas confirmadas de Oct 2024 en adelante:

```
{
  "selector": {
    "tipo_documento": "cita",
    "estado": "confirmada",
    "fecha_hora": {
      "$gt": "2024-10-01"
    }
  }
}
```

-Citas entre un rango de fechas:

```
{
  "selector": {
    "tipo_documento": "cita",
    "fecha_hora": {
      "$gte": "2024-09-01 00:00:00",
      "$lte": "2024-09-15 23:59:59"
    }
  }
}
```

Anexo 2. Consultas en RavenDB

-Para obtener las citas que han sido canceladas (muestra todas las columnas):

```
from citas
where estado = 'cancelada'
```

-Para obtener las citas que han sido canceladas (seleccionando solo las columnas indicadas):

```
from citas
where estado = 'cancelada'
select cita_id, fecha_hora, paciente_id, estado
```

-Cita de un paciente específico:

```
from citas
where paciente_id = "PAC00023"
select cita_id, fecha_hora, paciente_id, estado
```

-Inasistencias antes de una fecha específica:

```
from "Inasistencias"
where fecha_cita < "2024-11"
```

-Citas confirmadas de Oct 2024 en adelante:

```
from "Citas"
where fecha_hora > "2024-10-01" and estado = "confirmada"
```

-Citas entre un rango de fechas:

```
from "Citas"
where fecha_hora > "2024-09-01 00:00:00" and fecha_hora < "2024-09-15 23:59:59"
```

-Agrupar por motivo de sustitución y contar:

```
from 'Sustituciones'
group by notas
select notas, count()
```