



Base de datos en entornos móviles

Diseño y Programación de Software Multiplataforma DPS941

ELABORADO POR:

Samuel Fernando Calderon Reyes CR202814

Walter Daniel Mejia Palacios MP202829

Docente:

Alexander Alberto Siguenza Campos

Facultad de Ingeniería

Soyapango, septiembre 2023

Índice

Índice	2
Introducción	3
Diferencias fundamentales entre SQL y NoSQL	4
¿Cuáles son las diferencias específicas entre Cloud Firestore y Realtime Database?	6
Basándose en su investigación, ¿cuál de estas bases de datos consideran que sería la mejor opción para implementar en una aplicación desarrollada en React Native?	8
Comparación y Conclusiones	10
Ejemplo con base de datos SQL	12
Ejemplo con base de datos NoSQL	13
Tabla de Tipo Evaluación	13
Tabla de Materia	13
Tabla de alumno	14
Tabla de subcoleccion de materias inscritas	14
Tabla de subcoleccion de notas	15

Introducción

El proyecto "Sistema de Calificación Automatizada de Exámenes de Opción Múltiple con Arduino" tiene como objetivo desarrollar una solución tecnológica innovadora para agilizar y mejorar el proceso de calificación de exámenes de opción múltiple. Utilizando la plataforma de hardware Arduino y tecnologías electrónicas, el sistema automatiza la evaluación de las respuestas seleccionadas por los estudiantes en un examen de opción múltiple.

Diferencias fundamentales entre SQL y NoSQL

Una **base de datos SQL**, también conocida como una base de datos relacional, es un tipo de sistema de gestión de bases de datos (DBMS) que utiliza un modelo de datos relacional para almacenar y administrar la información. Sus datos están relacionados entre sí y se almacenan como texto de manera estructurada en tablas que constan de registros (filas) que es la propia información relacionada entre sí, en base a su contexto, y campos (columnas) que son los distintos tipos de información que se relacionan. Algunos ejemplos de SGBD SQL son: SQLite, PostgreSQL, Oracle, MySQL, SQL Server, entre otros.

Las características principales de las bases de datos SQL podemos encontrar:

- Almacenan datos de manera estructurada en tablas, con filas y columnas. Cada tabla tiene un esquema definido que especifica los tipos de datos que se pueden almacenar en cada columna.
- Utilizan el lenguaje SQL para realizar consultas y manipular los datos. SQL es un lenguaje de programación que se utiliza para interactuar con la base de datos y realizar operaciones como la inserción, actualización y eliminación de datos.
- Son adecuadas para aplicaciones que requieren una alta integridad de los datos y una consistencia transaccional. Esto significa que las bases de datos SQL son ideales para aplicaciones que requieren una alta precisión y coherencia de los datos, como los sistemas de contabilidad y finanzas.
- Proporcionan una sólida seguridad de los datos, escalabilidad, alto rendimiento y facilidad de uso. Las bases de datos SQL son altamente seguras y escalables, lo que significa que pueden manejar grandes volúmenes de datos y un gran número de usuarios simultáneamente.
- Son menos adecuadas para manejar grandes volúmenes de datos no estructurados, como imágenes, videos o texto. Las bases de datos SQL no son ideales para manejar grandes volúmenes de datos no estructurados, ya que estos datos no se pueden almacenar en tablas.

Por otro lado, las **bases de datos NoSQL** (Not Only SQL) aparecen debido a la necesidad de flexibilidad para almacenar distintos tipos de información no estructurada, de una manera que difiere significativamente del enfoque tradicional de las bases de datos SQL (estructurales y relacionales). Algunos ejemplos de SGBD NoSQL son: MongoDB, Firebase, Cassandra, CouchDB, Redis, Neo4j, entre otros.

En las características principales de las bases de datos NoSQL podemos encontrar:

- Almacenan datos en su formato original, sin estructurarlos en tablas. Los datos se almacenan en documentos, gráficos o pares clave-valor.
- No utilizan SQL para realizar consultas y manipular los datos, sino que utilizan otros lenguajes específicos. Cada tipo de base de datos NoSQL tiene su propio lenguaje de consulta.
- Son adecuadas para aplicaciones que requieren una alta escalabilidad y flexibilidad, como el Big Data. Las bases de datos NoSQL son ideales para aplicaciones que requieren una alta escalabilidad y flexibilidad, como el análisis de Big Data y la gestión de contenido web.
- Proporcionan una mayor capacidad de escalar que las bases de datos SQL. Las bases de datos NoSQL son altamente escalables y pueden manejar grandes volúmenes de datos y un gran número de usuarios simultáneamente.
- Son más adecuadas para manejar grandes volúmenes de datos no estructurados. Las bases de datos NoSQL son ideales para manejar grandes volúmenes de datos no estructurados, como imágenes, videos o texto.

¿Cuáles son las diferencias específicas entre Cloud Firestore y Realtime Database?

Cloud Firestore y Realtime Database son dos bases de datos en tiempo real ofrecidas por Google en su plataforma Firebase, y aunque ambas son bases de datos NoSQL que pueden ser utilizadas para aplicaciones en tiempo real, tienen diferencias clave en sus características y funcionamiento. Aquí hay algunas diferencias específicas entre Cloud Firestore y Realtime Database:

1. Modelo de datos:

- Cloud Firestore: Utiliza un modelo de datos basado en documentos y colecciones. Los datos se almacenan en documentos que se organizan en colecciones. Cada documento es un objeto JSON que puede contener campos anidados.
- Realtime Database: Utiliza un modelo de datos basado en árbol JSON. Los datos se almacenan en una única estructura de árbol con referencias a diferentes ubicaciones en la base de datos.

2. Escalabilidad:

- Cloud Firestore: Cloud Firestore está diseñado para ser altamente escalable. Puede manejar grandes cantidades de datos y tráfico, lo que lo hace adecuado para aplicaciones a gran escala.
- Realtime Database: Aunque Realtime Database es escalable, no escala tan bien como Cloud Firestore en términos de consultas complejas y grandes conjuntos de datos.

3. Consultas:

- Cloud Firestore: Permite realizar consultas más complejas, incluyendo consultas compuestas, filtrado y ordenación directamente en la base de datos.
- Realtime Database: Realtime Database ofrece capacidades de consulta más limitadas en comparación con Cloud Firestore. La mayoría de las consultas avanzadas requieren indexación adicional y planificación.

4. Soporte offline:

- Cloud Firestore: Cloud Firestore ofrece un sólido soporte para el modo fuera de línea, lo que significa que las aplicaciones pueden leer y escribir datos incluso cuando no están conectadas a internet. Los datos se sincronizan automáticamente cuando la conexión se restablece.
- Realtime Database: Realtime Database también admite el modo fuera de línea, pero su soporte es menos robusto que el de Cloud Firestore.

5. Facturación:

- Cloud Firestore: La facturación de Cloud Firestore se basa en la cantidad de operaciones que realizas, como lecturas, escrituras y eliminaciones, así como en el almacenamiento de datos.
- Realtime Database: La facturación de Realtime Database se basa en el ancho de banda y la cantidad de datos transferidos, lo que puede hacer que sea más difícil de predecir en comparación con Cloud Firestore.

6. Seguridad:

- Cloud Firestore y Realtime Database: Ambas bases de datos permiten configurar reglas de seguridad personalizadas para restringir el acceso a los datos. Puedes definir quién puede leer y escribir en diferentes ubicaciones de la base de datos.

Basándose en su investigación, ¿cuál de estas bases de datos consideran que sería la mejor opción para implementar en una aplicación desarrollada en React Native?

Firestore puede ser una mejor opción para implementar en una aplicación desarrollada en React Native por varias razones:

1. Modelo de datos más flexible y escalable: Firestore utiliza un modelo de datos basado en documentos y colecciones, lo que lo hace más flexible y escalable en comparación con el modelo de árbol JSON de Realtime Database. Esto facilita la organización y estructuración de tus datos de una manera más intuitiva y permite escalar eficazmente a medida que tu aplicación crece.
2. Consultas avanzadas: Firestore permite realizar consultas avanzadas directamente en la base de datos, lo que simplifica la obtención de datos específicos que coincidan con ciertos criterios. Se puede realizar consultas compuestas, filtrado y ordenación sin tener que realizar indexación adicional. Esto es especialmente útil en aplicaciones donde se requiere una búsqueda o filtrado complejo de datos.
3. Soporte para el modo fuera de línea robusto: Firestore ofrece un sólido soporte para el modo fuera de línea. Esto significa que los datos se almacenan en la memoria caché local de la aplicación y se pueden leer e escribir incluso cuando la aplicación no está conectada a internet. Cuando la conexión se restablece, los cambios se sincronizan automáticamente con la base de datos en la nube. Esto es crucial para aplicaciones que necesitan funcionar sin conexión, que puede ser el caso en una aplicación móvil.

4. Integración con Firebase: Firestore se integra estrechamente con otros servicios de Firebase, como Firebase Authentication, Firebase Cloud Functions y Firebase Hosting. Esto facilita la construcción de aplicaciones completas con Firebase como backend.

5. Escalabilidad y rendimiento: Firestore está diseñado para ser altamente escalable, lo que lo hace adecuado para aplicaciones que prevén un crecimiento significativo en el número de usuarios y datos. La arquitectura subyacente de Firestore se encarga de la escalabilidad sin que el desarrollador tenga que preocuparse demasiado por la infraestructura.

En resumen, Firestore es una opción sólida para aplicaciones desarrolladas en React Native, especialmente si valoras la flexibilidad en el modelo de datos, la capacidad de realizar consultas avanzadas, el soporte para el modo fuera de línea y la integración con otros servicios de Firebase.

Comparación y Conclusiones

Luego de investigar sobre ambas bases de datos, podemos realizar una comparación sobre ambas bases de datos, dicha comparación será a partir de las categorías más fundamentales que debe cumplir un gestor de base de datos:

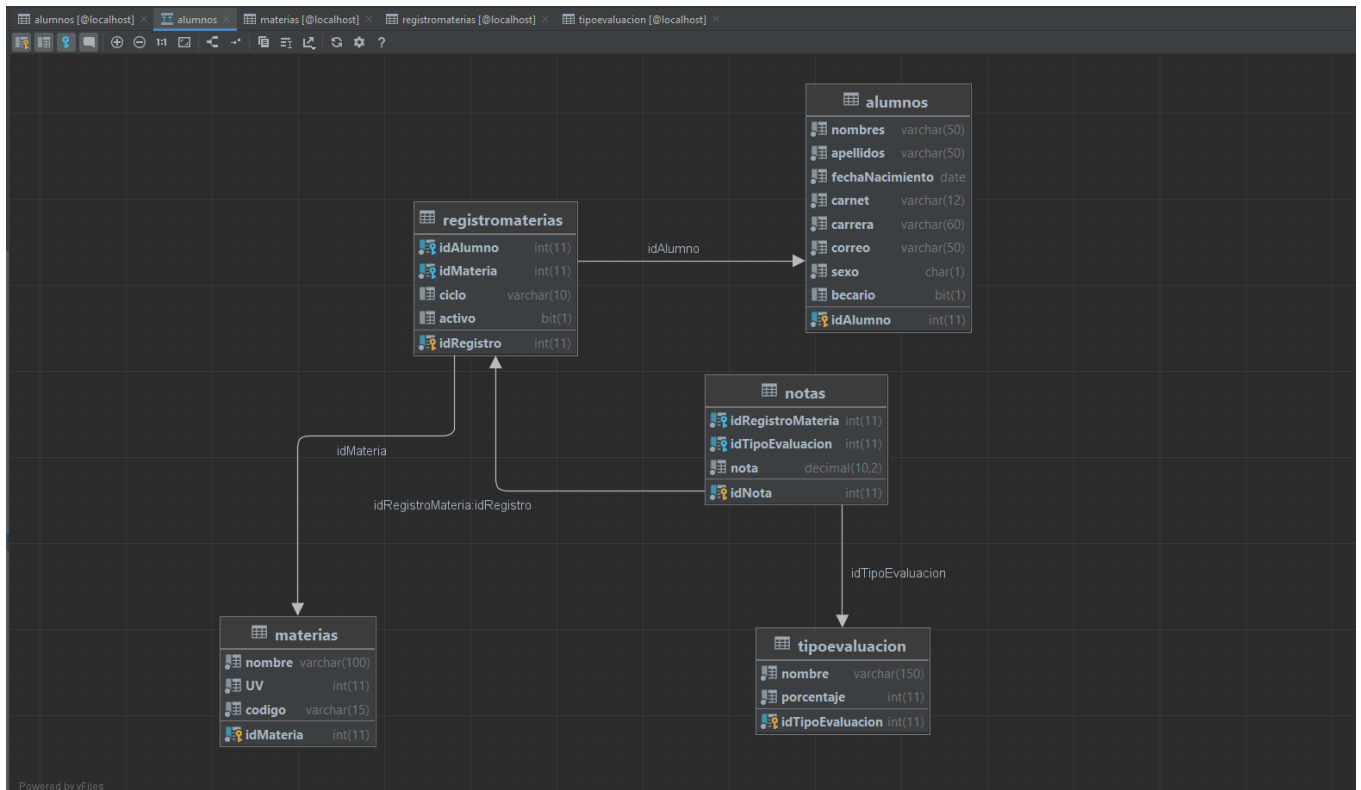
	SQL	NoSQL
Modelo de datos	Utiliza un modelo de datos relacional, que organiza la información en tablas con filas y columnas. Los datos se almacenan en estructuras tabulares con esquemas predefinidos.	Emplea un modelo de datos no relacional o flexible. Los datos se pueden almacenar en varios formatos, como documentos, grafos, pares clave-valor o columnas, lo que permite una mayor versatilidad en la estructura de los datos.
Escalabilidad	Tradicionalmente, las bases de datos SQL han sido desafiantes de escalar horizontalmente, lo que significa que agregar más servidores para aumentar la capacidad puede ser complicado.	Las bases de datos NoSQL están diseñadas para ser altamente escalables, lo que facilita la distribución de datos en varios servidores y la administración de grandes volúmenes de información.
Consistencia	Las bases de datos relacionales priorizan la consistencia de los datos, lo que significa que garantizan que los datos siempre cumplan con ciertas restricciones y reglas predefinidas.	A menudo, las bases de datos NoSQL sacrifican la consistencia en favor de la disponibilidad y la tolerancia a particiones, siguiendo el teorema CAP (Consistency, Availability, Partition tolerance).
Transacciones	Las bases de datos SQL son conocidas por admitir transacciones ACID (Atomicidad, Consistencia, Aislamiento, Durabilidad) que garantizan la integridad de los datos en todo momento.	Las bases de datos NoSQL a menudo utilizan el enfoque BASE (Basically Available, Soft State, Eventually Consistent), que permite una mayor disponibilidad y rendimiento a costa de posiblemente no

		garantizar la consistencia inmediata.
Escalabilidad de lectura/escritura	Por lo general, las bases de datos SQL son más adecuadas para aplicaciones con patrones de lectura intensiva y escritura moderada debido a la estructura de tablas y las transacciones.	Las bases de datos NoSQL son adecuadas para aplicaciones con patrones de escritura intensiva y lectura moderada, ya que pueden distribuir eficazmente las operaciones de escritura en varios nodos.
Consultas	Ofrece un lenguaje de consulta poderoso y estandarizado (SQL) que permite realizar consultas complejas y realizar análisis de datos sofisticados.	El acceso a los datos en bases de datos NoSQL suele ser más sencillo y flexible, pero las capacidades de consulta pueden ser limitadas en comparación con SQL, dependiendo del tipo de base de datos NoSQL.
Casos de uso	Tradicionalmente se utiliza en aplicaciones que requieren integridad de datos, transacciones complejas y análisis de datos estructurados. Ejemplos incluyen sistemas de contabilidad, ERP y aplicaciones empresariales.	Se utiliza en aplicaciones que requieren alta escalabilidad, flexibilidad en la estructura de datos y un alto rendimiento, como aplicaciones web, redes sociales, análisis de big data y juegos en línea.

En resumen, las bases de datos SQL y NoSQL tienen propiedades y estructuras bastante diferentes. Las bases de datos SQL son ideales para aplicaciones que requieren una alta integridad de los datos y una consistencia transaccional, mientras que las bases de datos NoSQL son ideales para aplicaciones que requieren una alta escalabilidad y flexibilidad. Es importante destacar que no existe una opción universalmente superior; la elección entre SQL y NoSQL depende de los requisitos y las características específicas de tu proyecto. A menudo, las aplicaciones modernas utilizan una combinación de ambas para abordar diferentes necesidades de almacenamiento y recuperación de datos.

Ejemplo con base de datos SQL

Esquema de base de datos SQL que permite almacenar las notas de los alumnos becarios de UDB VIRTUAL



Ejemplo con base de datos NoSQL

Tabla de Tipo Evaluación

<div> <div> <div>🏠</div> <div>> TipoEvaluacion > Rr1XQ46YbN7P..</div> </div> <div> <div>☰ (default)</div> <div> <div>+ Iniciar colección</div> <div>Alumno</div> <div>Materia</div> <div>TipoEvaluacion ></div> </div> </div> </div>	<div> <div>📁 TipoEvaluacion</div> <div>☰ ⋮</div> </div>	<div> <div>📄 Rr1XQ46YbN7Pq1XTNgJo</div> <div>☰ ⋮</div> </div>
	<div> <div>+ Agregar documento</div> <div>Rr1XQ46YbN7Pq1XTNgJo ></div> </div>	<div> <div>+ Iniciar colección</div> <div>+ Agregar campo</div> <div>Nombre: "Parcial"</div> <div>Porcentaje: 20</div> </div>

Tabla de Materia











<div>  > Materia > Lph6KH7i3JxaC. </div> <div>  Más funciones en Google Cloud  </div>		
<div>  (default) </div> <div> <div>+ Iniciar colección</div> <div>Alumno</div> <div> <div>Materia</div> <div>TipoEvaluacion</div> </div> </div>	<div>  Materia <div>   </div> </div> <div> <div>+ Agregar documento</div> <div>Lph6KH7i3JxaC9NbNoGu</div> </div>	<div>  Lph6KH7i3JxaC9NbNoGu <div>   </div> </div> <div> <div>+ Iniciar colección</div> <div>+ Agregar campo</div> <div> <div>UV: 4</div> <div>codigo: "DPS941 "</div> <div>nombre: "Diseño y Programación de Software Multiplataforma"</div> </div> </div>

Tabla de alumno

🏠 > Alumno > mOB00zwoYG...			☁ Más funciones en Google Cloud ▾		
(default)		📁 Alumno		📄 mOB00zwoYGMzj8hL6PrN	
+ Iniciar colección		+ Agregar documento		+ Iniciar colección	
Alumno >		mOB00zwoYGMzj8hL6PrN >		MateriasInscritas	
Materia					
TipoEvaluacion					
				+ Agregar campo	
				Becario: true	
				apellidos: "Mejia"	
				carnet: "MP202829"	
				carrera: "Ingeniería en Ciencias de la computacion"	
				correo: "waltermejia61@hotmail.com"	
				fechaNacimiento: "27/11/2002"	
				nombres: "Walter"	
				sexo : "H"	

Tabla de subcoleccion de materias inscritas

🏠 > Alumno > mOB00zwoYG... > MateriasInscrita... > ZazlnqIDXJkNN..			☁ Más funciones en Google Cloud ▾		
📄 mOB00zwoYGMzj8hL6PrN		📁 MateriasInscritas		📄 ZazlnqIDXJkNN5Bfj608	
+ Iniciar colección		+ Agregar documento		+ Iniciar colección	
MateriasInscritas >		ZazlnqIDXJkNN5Bfj608 >		notas	
+ Agregar campo				+ Agregar campo	
Becario: true				Ciclo: "02-2023"	
apellidos: "Mejia"				Nombre: "Diseño y Programación de Software Multiplataforma"	
carnet: "MP202829"				activa: true	
carrera: "Ingeniería en Ciencias de la computacion"				codigo: "DPS941 "	
correo: "waltermejia61@hotmail.com"					
fechaNacimiento: "27/11/2002"					
nombres: "Walter"					
sexo : "H"					

Tabla de subcoleccion de notas

<div><div><div>🏠</div><div>></div><div>...</div><div>></div><div>MateriasInscrita...</div><div>></div><div>ZazlnqlDXJkNN...</div><div>></div><div>notas</div><div>></div><div>zHJqqVKmyoM..</div></div><div>Más funciones en Google Cloud</div></div>		
<div><div>ZazlnqlDXJkNN5Bfj608</div><div>⋮</div></div>	<div><div>notas</div><div>☰ ⋮</div></div>	<div><div>zHJqqVKmyoMT3J0t5k2i</div><div>⋮</div></div>
<div>+ Iniciar colección</div>	<div>+ Agregar documento</div>	<div>+ Iniciar colección</div>
<div>notas ></div>	<div>zHJqqVKmyoMT3J0t5k2i ></div>	<div>+ Agregar campo</div>
<div>+ Agregar campo</div> <div>Ciclo: "02-2023"</div> <div>Nombre: "Diseño y Programación de Software Multiplataforma"</div> <div>activa: true</div> <div>codigo: "DPS941 "</div>		<div>Nota: 7.5</div> <div>TipoEvaluacion: "Parcial"</div>