UNIVERSIDAD DON BOSCO

FACULTAD DE INGENIERÍA

ESCUELA DE COMPUTACIÓN

Diseño y Programación de Software Multiplataforma - DPS941

Desarrollo de software multiplataforma y base de datos en entornos móviles

Integrantes:

1. Alberto Elena, Bryan Josué - AE210567
2. Colocho Díaz, Jairo Rafael - CD210488
3. González Crespín, Fátima Argentina – GC200404
4. Montano González, Fernando Josué - MG210111
5. Menjívar Ramírez, David Gerardo - MR210455
6. Salas Bojórquez, Kallahan Andrea - SB210537

Encargado

Alexander Alberto Sigüenza Campos

21/04/2024

ÍNDICE

INTRODUCCIÓN 3

INVESTIGACIÓN COMPARATIVA: CLOUD FIRESTORE VS. REALTIME DATABASE 4

Cloud firestore 4

Realtime database 4

Diferencias entre Cloud Firestore y Realtime Database 4

Diferencias fundamentales entre las bases de datos SQL y NoSQL 6

Selección de la base de datos ideal para una aplicación React Native 7

DISEÑO DE BASE DE DATOS SQL 9

DISEÑO DE BASE DE DATOS NOSQL 10

CONCLUSIÓN INVESTIGATIVA 11

CONCLUSIÓN DE IMPLEMENTACIÓN 12

INTRODUCCIÓN

Firebase es una plataforma alojada en la nube que cumple principalmente la funcionalidad para el desarrollo de aplicaciones web y móviles. Está disponible para diversas plataformas de alta demanda, como iOS, Android y web. La disponibilidad en estas plataformas otorga flexibilidad para trabajar en el desarrollo sin perder la calidad que se requiere.

Firebase ofrece un sinfín de herramientas, pero en la investigación actual nos interesa principalmente su gestión de tareas para el desarrollo, ya que facilita al desarrollador la parte del backend tanto en desarrollo como en mantenimiento. En Firebase, las principales herramientas de desarrollo son las bases de datos NoSQL, como Cloud Firestore o Realtime Database, que están alojadas en la nube. Firebase envía automáticamente eventos a las aplicaciones cuando los datos se actualizan, almacenando los nuevos datos en el disco o en caché. Esto permite que el usuario siempre tenga acceso a la última versión de los datos disponible sin necesidad de conexión, y una vez que vuelva a estar en línea, los datos se sincronizarán nuevamente.

La elección de optar por alguna de las bases de datos mencionadas requiere una investigación exhaustiva para conocer sus funciones y seleccionar según nuestras necesidades. A continuación, se presentará de forma objetiva esta información para obtener un resultado óptimo.

INVESTIGACIÓN COMPARATIVA: CLOUD FIRESTORE VS. REALTIME DATABASE

Cloud firestore

Cloud Firestore es una base de datos del tipo NoSQL alojada en la nube, que al igual que cualquier base de datos, permite almacenar información, sincronizar y consultar de forma eficaz, independientemente de su uso. Esta puede ser utilizada para aplicaciones web y/o dispositivos móviles. El uso de Cloud Firestore permite que se pueda consultar y estructurar los datos de la manera que el usuario desee. Su funcionalidad crea los datos en documentos y los almacena en colecciones, creando jerarquías a través de los datos relacionados para que la recuperación de los datos sea flexible y eficaz. Al realizar una consulta, no considera el tamaño total de la base de datos, sino los resultados esperados por la consulta realizada.

Realtime database

Firebase Realtime es una base de datos del tipo NoSQL que se aloja en la nube, como lo son las bases de datos. Su función principal es poder almacenar información, sincronizar y consultar. Firebase Realtime permite la conexión en tiempo real incluso cuando la app no tiene conexión, gracias a su almacenamiento local de datos. Los datos se almacenan en formato JSON y se sincronizan simultáneamente con cada cliente conectado. Una vez compilado, todos los usuarios y/o clientes comparten una instancia de esta misma y, de forma automatizada, se actualiza con los datos más recientes, por lo que busca datos de la app a escala global.

Diferencias entre Cloud Firestore y Realtime Database

Tabla 1 (Creación propia)

Cloud Firestore vs Realtime Database

Funciones	Cloud Firestore	Realtime Database
Tiempo real	Usa la sincronización de datos para su actualización y al mismo tiempo ofrece manejar consultas más complejas y estructuradas de manera eficiente.	Sincroniza datos siempre y cuando se reciba una actualización.
Sin conexión	Almacena en caché los datos que son usados recurrentemente en la app, al estar con conexión nuevamente sincroniza y actualiza.	Su SDK permite que los datos permanezcan en el disco, al tener conexión nuevamente actualiza los cambios que faltaban.
Escalamiento	1. Replicación automática de datos multirregión. 2. Garantías de coherencia sólida. 3. Operaciones atómicas por lotes. 4. Asistencia real sobre transacciones.	1. Escalabilidad con el plan Blaze de Firebase. 2. Optimización de la autenticación con Firebase Authentication. 3. Control de acceso con reglas de seguridad personalizadas en Firebase Realtime Database.
Crear app sin servidores	1. SDK web y para dispositivos móviles. 2. Conjunto completo de reglas de seguridad. 3. Cloud Functions para ejecutar código backend. 4. Acceso a través de bibliotecas cliente tradicionales	1. SDK web y para dispositivos móviles. 2. Cloud Functions para ejecutar código backend.

Además de tomar en cuenta la figura anterior (Tabla 1), debemos considerar que, con Cloud Firestore, como se mencionó previamente, está diseñado creando datos en documentos y documentos en colecciones. Por ende, para realizar consultas, lo realiza de forma horizontal, lo que facilita generar consultas mucho más complejas que lo que sería con Realtime Database. Este último tiene una escalabilidad vertical, lo que limita que las consultas sean más sencillas o simples, siendo una estructura de árbol con nodos en formato JSON.

Diferencias fundamentales entre las bases de datos SQL y NoSQL

Tabla 2 (Creación propia)

SQL vs NoSQL

Función	SQL	NoSQL
Lenguaje	Como sus siglas lo indican es un Lenguaje de Consulta Estructurado. Solo puede acceder para leer y escribir sus datos comprendiendo su lenguaje.	No utiliza un Lenguaje de Consulta Estructurado, sino, utiliza documentos JSON para almacenar los datos. Al ser NoSQL se puede acceder o interactuar con sus datos a través de diferentes métodos.
Escalabilidad y rendimiento	Usualmente se utiliza escalabilidad horizontal con fragmentación y para hacerla crecer se debe de aumentar la potencia del procesamiento del hardware actual.	Estas generalmente utilizan la escalabilidad vertical y para hacerlas crecer puede ser implementando más servidores o nodos.
Estructura de datos	Las bases de datos SQL tienen un formato tabular compuesto por columnas que representa los campos y las filas los registros y se relacionan entre tablas.	Las bases de datos NoSQL generalmente almacenan documentos con propiedades que contienen valores. Es una estructura flexible que se puede

		relacionar sin el uso de tablas.
Propiedades de la base de datos	ACID (Atomicidad, Consistencia, Aislamiento y Durabilidad)	CAP (Coherencia, Disponibilidad y Tolerancia de Partición)
Soporte y comunidades	Amplio soporte y comunidad establecida	Menor soporte y comunidad en comparación con SQL, pero creciendo con el tiempo.

Selección de la base de datos ideal para una aplicación React Native

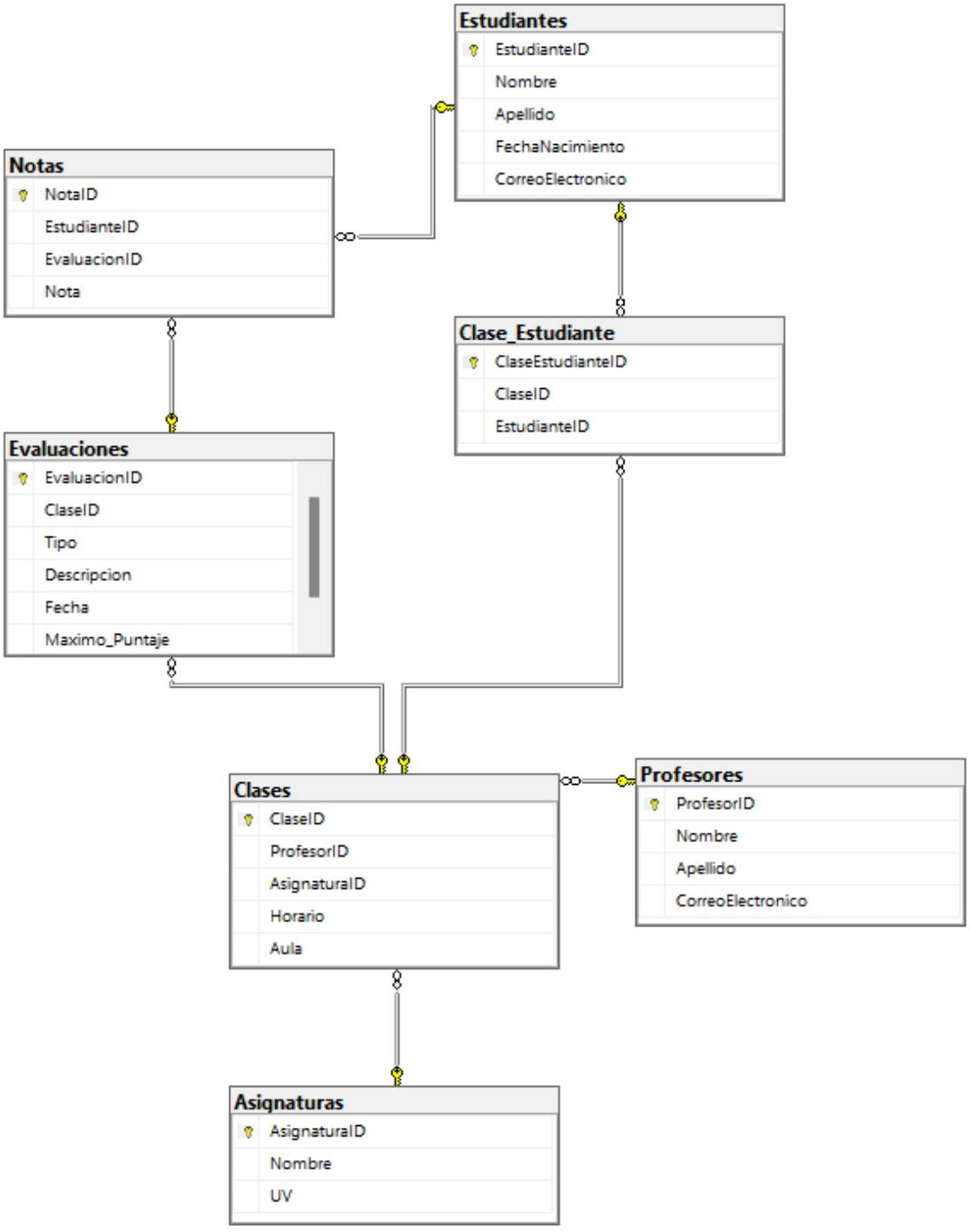
Al desarrollar una aplicación en React Native debemos considerar diversos factores para una elección apropiada. Se puede optar por delimitar inicialmente si será una base de datos SQL o NoSQL. Esta elección dependerá de la escalabilidad que tendrá nuestra aplicación móvil respecto a sus datos y la flexibilidad deseada para su estructura de datos. También, se debe tomar en cuenta cómo una base de datos SQL o NoSQL se integrará con React Native. En este caso de prueba en el que es requisito que sea desarrollada con React Native y valorando que algunas bases de datos NoSQL proporcionan bibliotecas y SDK específicos para React Native, a diferencia de las SQL, la mejor opción parece ser una base de datos NoSQL por la facilidad de integración brindada.

Ahora bien, ya delimitado lo previamente mencionado, debemos optar por seleccionar una opción entre Cloud Firestore o Realtime Database. La primera nos permitirá trabajar con datos complejos y relacionados, una escalabilidad horizontal, lo que puede ser de mucha utilidad si son datos en crecimiento constantemente y así mismo con la recurrencia de los usuarios. Por otro lado, la segunda, Realtime Database, está basada en modelo de árboles JSON, donde es mucho más fácil manejar datos simples y sin estructura estricta. Esta opción es viable si es muy necesario sincronizar datos en tiempo real como lo

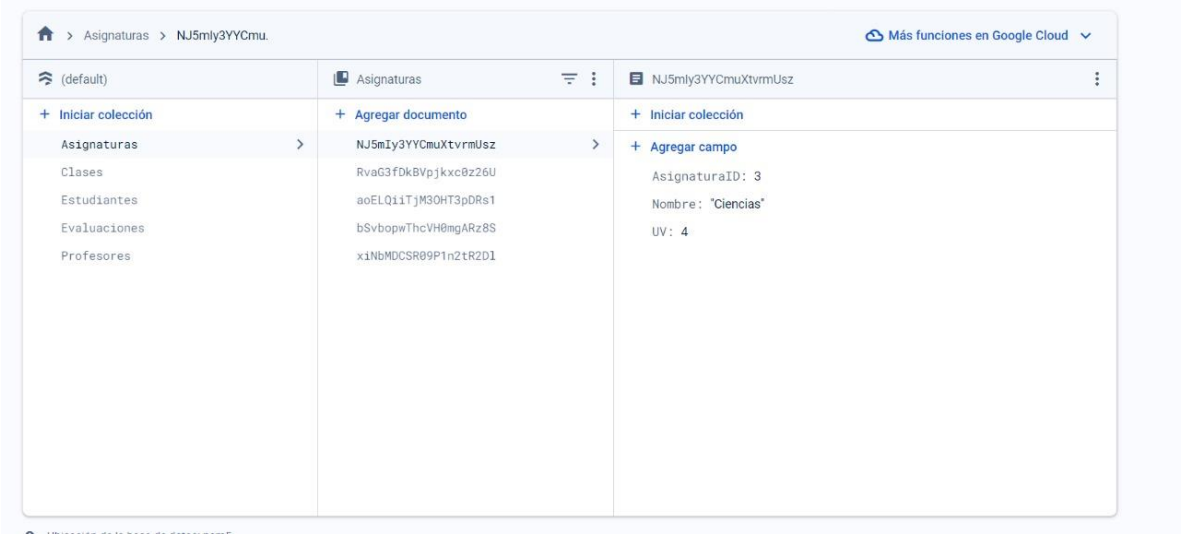
podrían ser los chats. Su escalabilidad es mucho menos limitada permitiendo consultas más sencillas o simples.

La selección entre Cloud Firestore o Realtime Database también se basará en el tipo de aplicación a desarrollar. Si esta requiere de datos complejos y relacionados, lo ideal sería usar Cloud Firestore. Si, por el contrario, es una aplicación con datos simples y cuyo objetivo principal es la sincronización en tiempo real, lo recomendable sería Realtime Database. Haciendo uso del caso de prueba para la segunda fase de la investigación, que implica un ejercicio práctico para el diseño de base de datos que almacenará las notas de los alumnos becarios de la Universidad Don Bosco campus virtual, y por supuesto, desarrollada en React Native como lo indica el planteamiento de este apartado, lo adecuado sería optar por realizar la base de datos con Cloud Firestore.

DISEÑO DE BASE DE DATOS SQL



DISEÑO DE BASE DE DATOS NOSQL



CONCLUSIÓN INVESTIGATIVA

Una vez realizada y leída la investigación previa, es conocido que Firebase ofrece dos soluciones de bases de datos almacenadas en la nube del tipo NoSQL, las cuales admiten la sincronización en tiempo real.

Cloud Firestore fue lanzada posteriormente que Realtime Database, por lo que toma las mejores ventajas o funcionalidades de este último, pero con una estructura de la base de datos mucho más compleja o menos sencilla, lo cual permite las relaciones entre sí, una escalabilidad mucho más alta que da la oportunidad al usuario de realizar consultas más complejas, pero con respuestas rápidas. La estructura de datos de Cloud Firestore puede manejar un volumen de datos y usuarios mucho más grande que lo que sería con Realtime Database, permitiendo mucha más flexibilidad.

Realtime Database, al ser la base de datos NoSQL original lanzada por Firebase, tiende a ser mucho más sencilla en estructura de datos, por lo que su solución tiende a ser eficiente siempre y cuando las consultas sean sencillas. Realtime Database, por su sencillez en estructura de datos, también es fácil de manejar los datos en tiempo real. Por ende, para la aplicación en la que será utilizada, los datos serán sincronizados instantáneamente, generando una visualización e interacción fluida, incluso cuando haya muchos usuarios.

La elección de la base de datos a utilizar siempre dependerá de los requisitos de la aplicación, por lo que los desarrolladores en el equipo deberán evaluar la mejor opción que se adapte a sus necesidades.

CONCLUSIÓN DE IMPLEMENTACIÓN

El implementar una base de datos en SQL Server, basada en la estructura tradicional de tablas, columnas, relaciones, llaves primarias y foráneas, y luego migrar esa misma lógica a Cloud Firestore, un sistema NoSQL, implica un cambio significativo en la forma de implementación y en el funcionamiento de la base de datos. Aunque, ambos sistemas comparten el mismo objetivo de almacenar y gestionar los datos, la lógica detrás de esta es bastante diferente.

En SQL Server, como ya conocemos la estructura de datos se define mediante tablas que están relacionadas entre sí a través de llaves primarias y foráneas. Además, se aplican principios de normalización para mantener la consistencia y reducir la redundancia. Este modelo relacional es ideal para aplicaciones con transacciones complejas y requisitos de consistencia estricta.

En cambio, Cloud Firestore utiliza un modelo basado en colecciones, subcolecciones y documentos. Cada transacción es representada por un documento, y las relaciones no se definen con llaves foráneas, sino que se gestionan a través de referencias, como subcolecciones. Este enfoque ofrece mayor flexibilidad y escalabilidad, pero puede ser menos estricto en términos de consistencia.

Al comparar ambos casos de implementación, se puede observar que SQL Server se enfoca en el modelo relacional, diseñado para aplicaciones con transacciones complejas y que requieren alta demanda. Por otro lado, Cloud Firestore está optimizado para aplicaciones que valoran la escalabilidad y la disponibilidad, permitiendo un flujo más fluido de datos a través de colecciones y documentos, porque será mucho más fácil acceder

a documentos y colecciones quienes dentro de sí mismas tienen su referencia, que escalar entre diferentes relaciones y tablas como lo sería en una base de datos SQL.