UNIVERSIDAD DON BOSCO

DISEÑO Y PROGRAMACIÓN DE SOFTWARE MULTIPLATAFORMA

Docente: Alexander Alberto Sigüenza Campos

Foro de Discusión [15%]

Proyecto de Carrito de Compras en JavaScript con Facturación

Integrantes:

- Jabes Alfredo Flores Reyes FR230108
- Alexandra María Henríquez Miranda HM232507
- Eduardo Ronald Gómez Miguel GM212911
- Jesús Alejandro Campos Landaverde CL212345
- Diego Alessandro Abarca Soto AS230109

28 de abril de 2024

Fase 1:

Investigar sobre las dos bases de datos NoSQL que ofrece Firebase:

1. Cloud Firestore

2. Realtime Database.

Por cada tipo de base de datos, se espera que investiguen lo siguiente:

1. ¿Qué es Cloud Firestore?

2. ¿Qué es Realtime Database?

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

4. ¿Cuáles son las diferencias fundamentales entre las bases de datos SQL y NoSQL?

5. 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?. Justifique su respuesta.

Fase 2:

1. Ambas bases de datos deben permitir almacenar las notas de los alumnos becarios de UDB VIRTUAL. (considerar todas las tablas o estructuras necesarias para almacenar notas) (profesores, materias, asignaturas, evaluaciones, etc.)

2. Cada base de datos debe tener por lo menos 5 registros

3. En la base de datos SQL, se requiere que esté normalizada y que siga las buenas prácticas de diseño de bases de datos relacionales.

4. Para la base de datos NoSQL, se espera que diseñen una estructura que aproveche las ventajas de Firestore o Realtime Database en Firebase. (dejar capturas del trabajo en Firebase en el documento pdf)

5. Finalmente después de la implementación de las bases Sql y NoSQL proporcionar conclusiones basadas en su experiencia y en la investigación realizada. (dejarlo en el documento pdf)

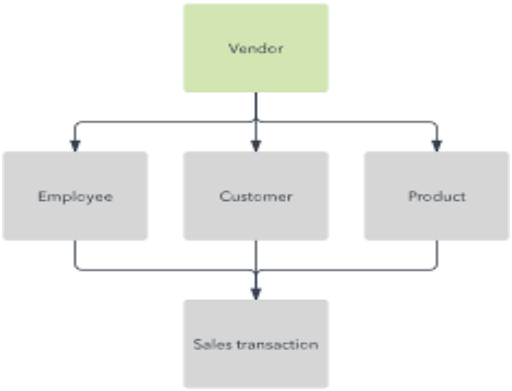
1. ¿Qué es Cloud Firestore?

Cloud Firestore es una base de datos NoSQL de Google que forma parte de Firebase. Está diseñada para almacenar y sincronizar datos en tiempo real entre aplicaciones y servidores, utilizando una estructura basada en documentos y colecciones en lugar de tablas y filas.

La sincronización en tiempo real es una característica destacada, permitiendo a múltiples usuarios ver cambios al instante. Cloud Firestore es escalable, adaptándose a grandes cantidades de datos y tráfico, y ofrece consultas avanzadas para buscar y filtrar datos eficientemente. También tiene soporte offline, permitiendo a las aplicaciones seguir funcionando incluso sin conexión a Internet. Cloud Firestore es ideal para aplicaciones modernas que requieren datos en tiempo real, escalabilidad y funcionalidad offline.

Principales características:

Modelo de datos flexible: Utiliza documentos y colecciones para almacenar datos, permitiendo diversos tipos como cadenas, números y listas.

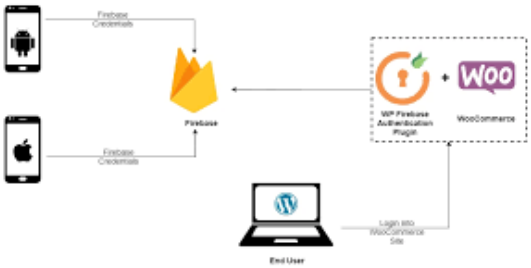


Sincronización en tiempo real: Actualiza datos en tiempo real, permitiendo que los usuarios vean cambios al instante.

Escalabilidad: Escala automáticamente para manejar grandes volúmenes de datos y tráfico.

Consultas avanzadas: Permite filtrar, ordenar y limitar resultados con eficiencia.

Integración con Firebase: Se integra con otros servicios de Firebase, facilitando el desarrollo de aplicaciones.



Soporte offline: Permite que las aplicaciones funcionen sin conexión, sincronizando los datos cuando se restablece la conexión.

Como funciona:

Cloud Firestore es una base de datos NoSQL de Google, parte de Firebase, que almacena datos en documentos y colecciones. Ofrece sincronización en tiempo real, permitiendo que los datos se actualicen automáticamente entre clientes y servidores. La estructura flexible de documentos y colecciones facilita la organización de datos variados.

Las consultas avanzadas permiten filtrar, ordenar y limitar resultados de manera eficiente, y la escalabilidad de Firestore le permite manejar grandes volúmenes de datos y tráfico. Se integra bien con otros servicios de Firebase, como Firebase Authentication, para autenticación de usuarios y Firebase Cloud Functions para lógica de backend.

El soporte offline de Firestore permite que las aplicaciones funcionen sin conexión a Internet, sincronizando datos cuando se restablece la conexión.

Ventajas:

- **Sincronización en tiempo real:** Actualiza datos automáticamente entre clientes y servidores, ideal para aplicaciones de chat y colaboración.
- **Escalabilidad:** Se adapta a grandes volúmenes de tráfico y datos.
- **Consultas avanzadas:** Permite filtrar, ordenar y limitar resultados eficientemente.
- **Integración con Firebase:** Funciona bien con otros servicios de Firebase como Authentication y Cloud Functions.
- **Soporte offline:** Permite que las aplicaciones funcionen sin conexión, sincronizando datos cuando la conexión se restaura.
- **Flexibilidad:** Usa documentos y colecciones, facilitando una estructura de datos adaptable.

Desventajas:

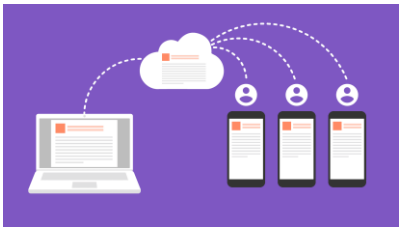
- **Costos:** Puede volverse costoso con un alto volumen de operaciones.
- **Límites de uso:** Existen restricciones en tamaño de documentos y tasas de lectura/escritura.
- **Complejidad para algunas consultas:** Algunas operaciones pueden requerir índices personalizados o ser menos eficientes.
- **Dependencia de Google:** El servicio depende de la infraestructura de Google Cloud, por lo que cualquier interrupción o cambio en esa infraestructura podría afectar la aplicación.

2. ¿Qué es Realtime Database?

Realtime Database de Firebase es una base de datos en la nube que sincroniza datos en tiempo real entre los clientes conectados. Utiliza JSON para almacenar datos y ofrece escalabilidad, seguridad y diversas integraciones con otras herramientas de Firebase. Esto nos permite crear aplicaciones multiplataforma con funciones avanzadas como sincronización offline y control de acceso personalizado.

Características principales:

Colabora entre dispositivos con facilidad



La sincronización en tiempo real permite que los usuarios accedan a sus datos desde cualquier dispositivo, web o móvil, con facilidad, y los ayuda a trabajar en conjunto.

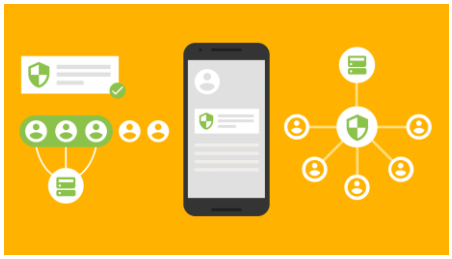
Crear aplicaciones sin servidores.

Realtime Database se incluye en los SDK web y para dispositivos móviles, de manera que puedas crear aplicaciones sin la necesidad de usar servidores. También puedes ejecutar un código de backend que responda a los eventos que activan la base de datos a través de Cloud Functions para Firebase .

Optimizada para el uso sin conexión

Cuando los usuarios se desconectan, los SDK de Realtime Database usan la caché local del dispositivo para publicar y almacenar cambios. Cuando el dispositivo se conecta, los datos locales se sincronizan de manera automática.

Seguridad sólida basada en usuarios



Realtime Database se integra en Firebase Authentication para brindar una autenticación intuitiva y sencilla a los desarrolladores. Puedes usar nuestro modelo de seguridad declarativa para permitir el acceso según la identidad de los usuarios o patrones que coincidan con tus datos.

¿Como funciona?

Con Firebase Realtime Database, puedes compilar aplicaciones dinámicas e interactivas, ya que permite el acceso seguro a la base de datos directamente desde el código del cliente. Los datos persisten de forma local. Incluso cuando no hay conexión, se siguen activando los eventos en tiempo real, lo que proporciona una experiencia adaptable al usuario final. Cuando el dispositivo vuelve a conectarse, Realtime Database sincroniza los cambios de los datos locales con las actualizaciones remotas que ocurrieron mientras el cliente estuvo sin conexión, lo que combina los conflictos de forma automática.

Realtime Database proporciona un lenguaje flexible de reglas basadas en expresiones, llamado reglas de seguridad de Firebase Realtime Database, para definir cómo se deberían estructurar los datos y en qué momento se pueden leer o escribir. Integrar Firebase Authentication permite que los desarrolladores definan quién tiene acceso a qué datos y cómo acceden a ellos.

Realtime Database es una base de datos NoSQL y, como tal, tiene diferentes optimizaciones y funcionalidades en comparación con una base de datos relacional. La API de Realtime Database está diseñada para permitir solo operaciones que se puedan ejecutar rápidamente. Eso permite crear una excelente experiencia de tiempo real que puede servir a millones de usuarios sin afectar la capacidad de respuesta. Es importante pensar cómo deben acceder a los datos los usuarios y estructurarlos según corresponda.

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

Estructura de datos

- Cada base de datos NoSQL almacena datos de diferentes maneras. Firebase Realtime Database almacena datos como un gran árbol JSON, que es un grupo de documentos JSON. Esto funciona bien para manejar datos simples, pero puede fallar si necesita su base de datos para organizar grandes cantidades de datos o manejar datos de manera jerárquica.
- Google Cloud Firestore utiliza documentos que contienen campos a los que se asignan a valores reales. Estos documentos se almacenan como colecciones y subcolecciones, que se pueden organizar para recopilar datos relacionados o facilitar consultas a la base de datos. La base de datos puede admitir muchos tipos de datos, incluso objetos anidados y estructuras de datos jerárquicas complejas.

Soporte fuera de línea

- Firebase Realtime Database admite un modo fuera de línea en el que la persistencia se mantiene a través de un caché local, incluso cuando la aplicación está desconectada de internet. La base de datos se actualiza automáticamente y se sincroniza con su instancia cuando la aplicación se reconecta a internet. Así, la aplicación puede seguir funcionando incluso cuando el acceso a la red es lento o intermitente.
- Google Cloud Firestore también proporciona un modo sin conexión que almacena en caché los datos del cliente y permite que las aplicaciones continúen leyendo y escribiendo en la base de datos sin acceso a internet. Sin embargo, también se integra con otros servicios de Google Cloud, como Cloud Functions, así como con bibliotecas de código abierto. Esto proporciona acciones más versátiles y flexibilidad en el diseño de la aplicación: solo los cambios en la base de datos se intercambian durante la sincronización.

Seguridad

- Firebase Realtime Database está construida para manejar datos no estructurados con un lenguaje de reglas llamado Firebase Realtime Database Security Rules. Las reglas definen las estructuras de datos y controlan cómo se escriben, leen, validan e indexan los datos. Los equipos de TI también pueden usarlos junto con Firebase Authentication, que permite el control sobre el acceso a datos a nivel de usuario.
- Google Cloud Firestore también usa su propio lenguaje de reglas llamado Cloud Firestore Security Rules. Al igual que con Realtime Database, las reglas pueden proporcionar protección granular a los contenidos y el acceso a la base de datos. Estas reglas también se pueden usar junto con Firebase Authentication para aplicaciones móviles y web, y las empresas también pueden usar Google Cloud Identity and Access Management para los idiomas del lado del servidor.

Consultas

- Firebase Realtime Database puede ordenar o filtrar, pero no ambos. Las consultas pueden ser granulares, pero devolverán todo el subárbol, lo que ofrece más resultados de los necesarios. Si bien la base de datos no requiere indexación, esto puede afectar el rendimiento en grandes conjuntos de datos. En comparación.
- Google Cloud Firestore admite necesidades más complejas, como consultas indexadas que simultáneamente ordenan y filtran para refinar los resultados de búsqueda.

Escritura y transacciones

- Firebase Realtime Database utiliza escrituras y transacciones simples. Los datos se escriben a través de conjuntos dedicados y operaciones de actualización periódicas. Los datos también se guardan a través de transacciones.
- Google Cloud Firestore agrega flexibilidad a las escrituras y transacciones, escribiendo datos con conjuntos y actualizaciones, además de permitir operaciones más complejas con transformaciones. Las transacciones también se pueden leer y escribir en cualquier parte de la base de datos, mientras que Firebase Realtime solo puede usar transacciones en un subárbol de datos específico.

Fiabilidad y escalabilidad

- En términos de confiabilidad y rendimiento, Firebase Realtime Database proporciona operaciones extremadamente rápidas, pero la base de datos está limitada a las zonas de disponibilidad de una sola región. Además, la base de datos deberá dividirse en varias instancias si supera las 100,000 conexiones concurrentes y 1,000 escrituras por segundo.
- En comparación, Google Cloud Firestore ofrece la confiabilidad de un servicio nativo de varias regiones. Además, es más escalable con límites actuales de 1 millón de conexiones concurrentes y 10,000 escrituras por segundo, y se espera que aumente en el futuro.

4. ¿Cuáles son las diferencias fundamentales entre las bases de datos SQL y NoSQL?

- Las BBDD SQL almacenan datos de manera estructurada y las NoSQL lo hacen en su formato original.

- Las SQL proporcionan una capacidad de escalar baja, en comparación con las NoSQL. Esta es una de las principales ventajas de las NoSQL, ya que están pensadas para grandes volúmenes de información como el Big Data. Lo anterior es debido a que las SQL están centralizadas y las NoSQL distribuidas, posibilitando que se ejecuten en múltiples máquinas, pero con muy pocos recursos (RAM, CPU, disco…).

- La adaptación a los cambios de las SQL es poca y puede ser compleja. Sin embargo, las NoSQL son totalmente flexibles.

- Las BBDD SQL están totalmente estandarizadas y las NoSQL carecen de homogeneización.

- Las SQL se utilizan en múltiples aplicaciones de todo tipo, las NoSQL se emplean principalmente para el Big Data (por ejemplo, en redes sociales).

- Las BBDD SQL proporcionan consistencia en los datos (integridad). Sin embargo, las NoSQL, al buscar rapidez, no ponen el foco en esta característica.

- La rapidez de ambas BBDD dependerá del contexto o de su uso: en datos estructurados las SQL son más rápidas, pero el Big Data no se estructura y ahí consiguen más rapidez las NoSQL.

5. 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?. Justifique su respuesta.

Elegir entre Cloud Firestore y Realtime Database para una aplicación desarrollada en React Native depende de varios factores, como la estructura de datos, la escalabilidad, la complejidad de las consultas y las características específicas de cada base de datos. Aquí está la comparación para ayudarte a decidir cuál sería la mejor opción para tu aplicación:

Cloud Firestore

-Estructura de datos: Ofrece una estructura basada en documentos y colecciones, que es más flexible y permite datos anidados. Ideal para aplicaciones que requieren un modelo de datos complejo.

-Consultas avanzadas: Soporta consultas más complejas, como filtrado, ordenación y limitación de resultados, con índices automáticos y personalizados.

-Escalabilidad: Está diseñado para escalar automáticamente, lo que lo hace adecuado para aplicaciones con crecimiento rápido.

-Sincronización en tiempo real: Aunque no es tan rápida como Realtime Database, ofrece sincronización casi en tiempo real.

-Integración con Firebase: Se integra fácilmente con otros servicios de Firebase.

-Soporte offline: Ofrece soporte offline para aplicaciones móviles, permitiendo operaciones sin conexión y sincronización posterior.

Realtime Database

-Estructura de datos: Utiliza un modelo basado en árboles, más simple pero menos flexible que Firestore.

-Sincronización en tiempo real: Proporciona sincronización más rápida que Firestore, ideal para aplicaciones que requieren actualización instantánea.

-Escalabilidad: Escala, pero puede ser menos eficiente que Firestore para grandes volúmenes de datos o aplicaciones con un alto tráfico.

-Consultas: Soporta consultas simples, pero es menos adecuado para consultas complejas.

-Soporte offline: También tiene soporte offline, pero menos robusto que Firestore.

¿Cuál es la mejor opción?

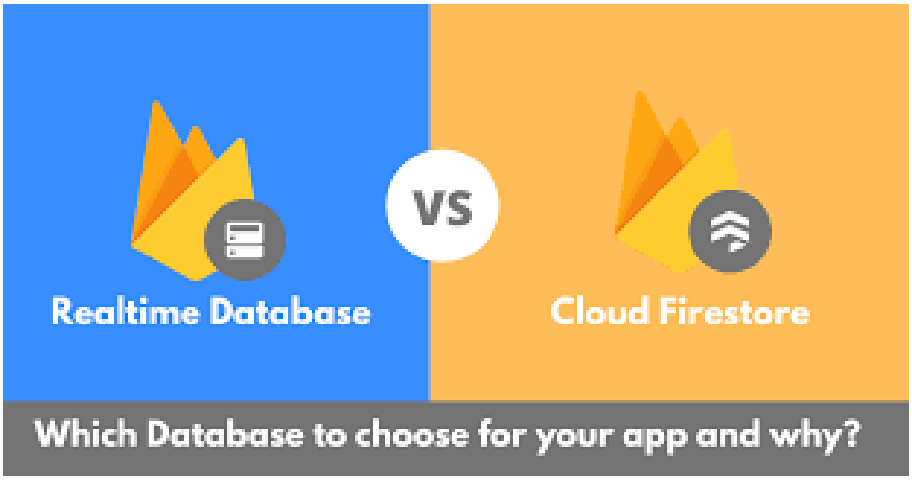
Si la aplicación en React Native requiere:

- Consultas avanzadas o estructuras de datos complejas, Cloud Firestore es la mejor opción.

- Sincronización rápida en tiempo real con estructuras de datos más simples, Realtime Database es más adecuado.

Si se necesita una solución escalable con una estructura de datos flexible y consultas avanzadas, elige Cloud Firestore. Si la prioridad es la sincronización en tiempo real y la aplicación tiene una estructura de datos simple, Realtime Database puede ser una mejor

opción. Considera también el soporte offline y la integración con otros servicios de Firebase según tus necesidades específicas.



¿Qué es la base de datos SQL?

Una base de datos SQL es un base de datos relacional que organiza los datos en tablas con filas y columnas. SQL significa Lenguaje de consulta estructurado, que es el lenguaje estándar utilizado para consultar y manipular datos en una base de datos relacional.

Algunas características clave de una base de datos SQL incluyen:

- Los datos se almacenan en tablas que contienen filas y columnas. Cada fila representa un registro y cada columna representa un atributo de ese registro.
- Hay relaciones entre tablas que se imponen mediante el uso de claves foráneas. Esto asegura la integridad de los datos y reduce la redundancia.
- El lenguaje SQL se utiliza para consultar y manipular datos. SQL proporciona comandos como SELECCIONAR, INSERTAR, ACTUALIZAR y ELIMINAR para interactuar con la base de datos.
- Las propiedades ACID (Atomicidad, Consistencia, Aislamiento, Durabilidad) se aplican para garantizar la confiabilidad e integridad de los datos. Las transacciones se completan o no se completan en absoluto.

¿Qué es una base de datos NoSQL?

Una base de datos NoSQL es un base de datos no relacional que almacena datos en un formato distinto de filas y columnas. Las bases de datos NoSQL vienen en una variedad de tipos según su modelo de datos. Los principales tipos son:

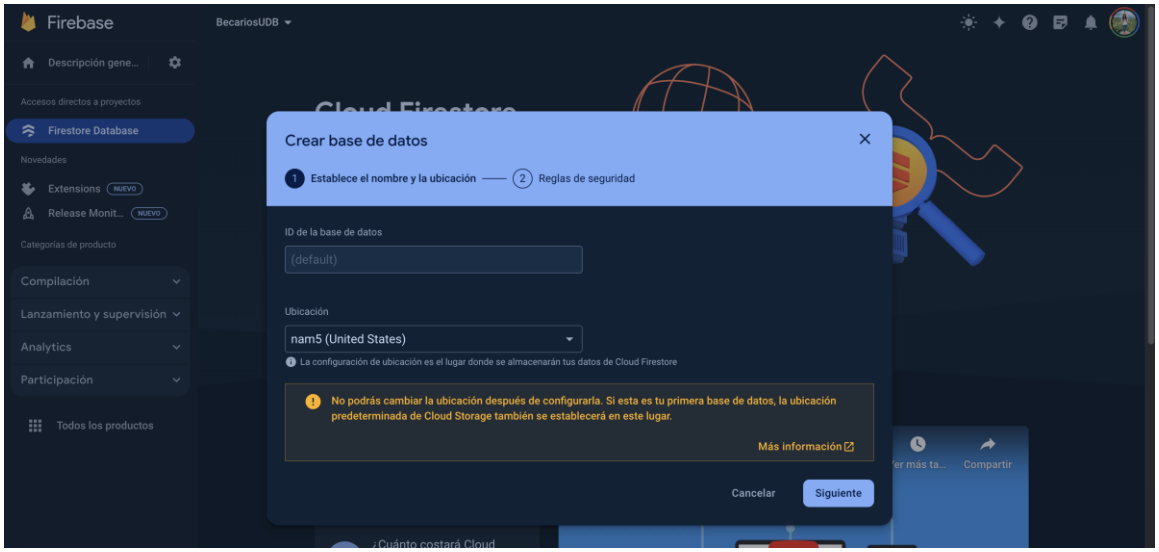
- Almacenes de clave-valor: los datos se almacenan en un formato no estructurado con una clave única para recuperar valores. Los ejemplos son Redis y DynamoDB.

- Bases de datos de documentos: los datos se almacenan en formato de documento, como JSON. Los ejemplos son MongoDB y CouchDB.
- Bases de datos de gráficos: los datos se almacenan en nodos y bordes, optimizados para las relaciones de datos. Los ejemplos son Neo4j y JanusGraph.
- Bases de datos en columnas: los datos se almacenan en columnas en lugar de filas. Algunos ejemplos son Cassandra y HBase.

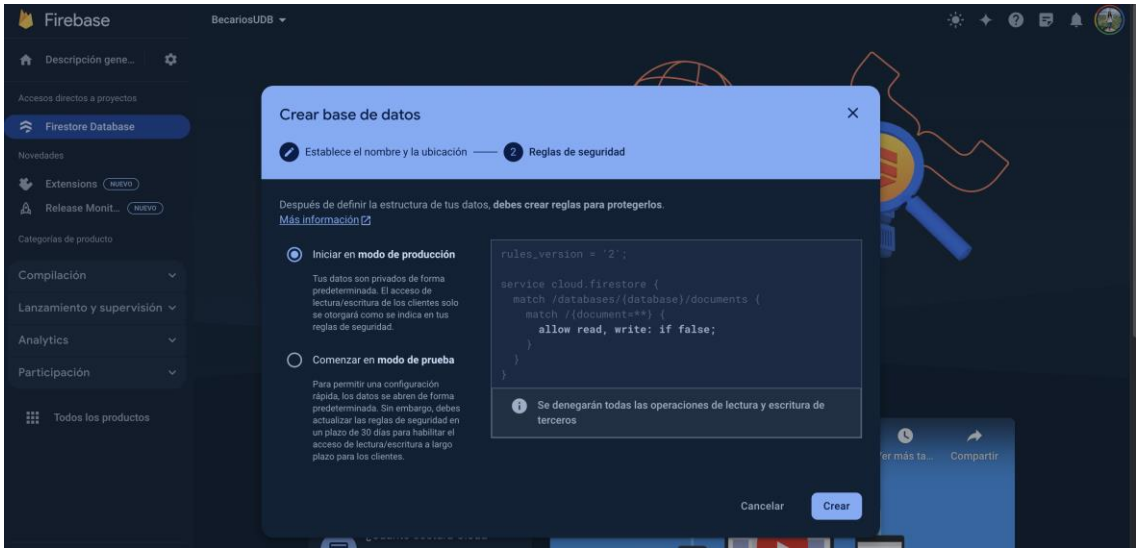
Base de datos NoSQL:

1- Click en Crear base de datos:

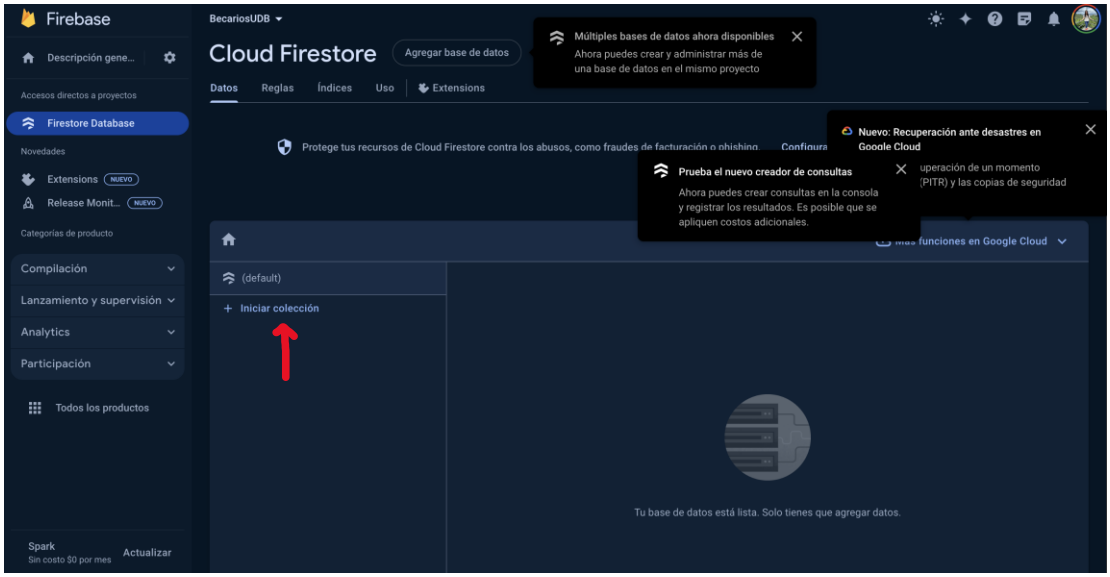
2- Click en Siguiente:



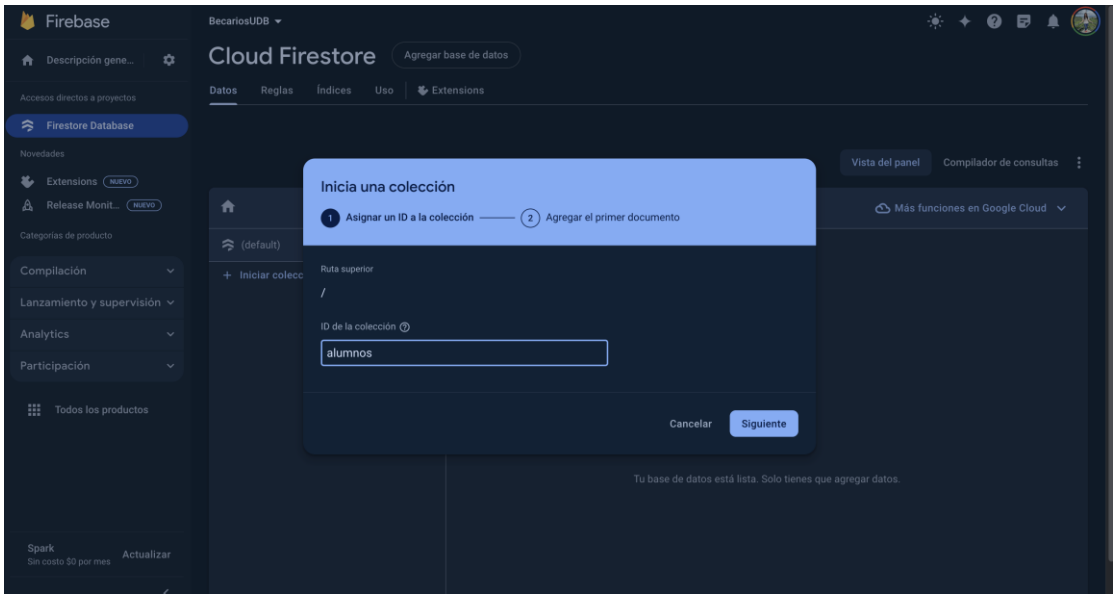
3- Click en Crear:



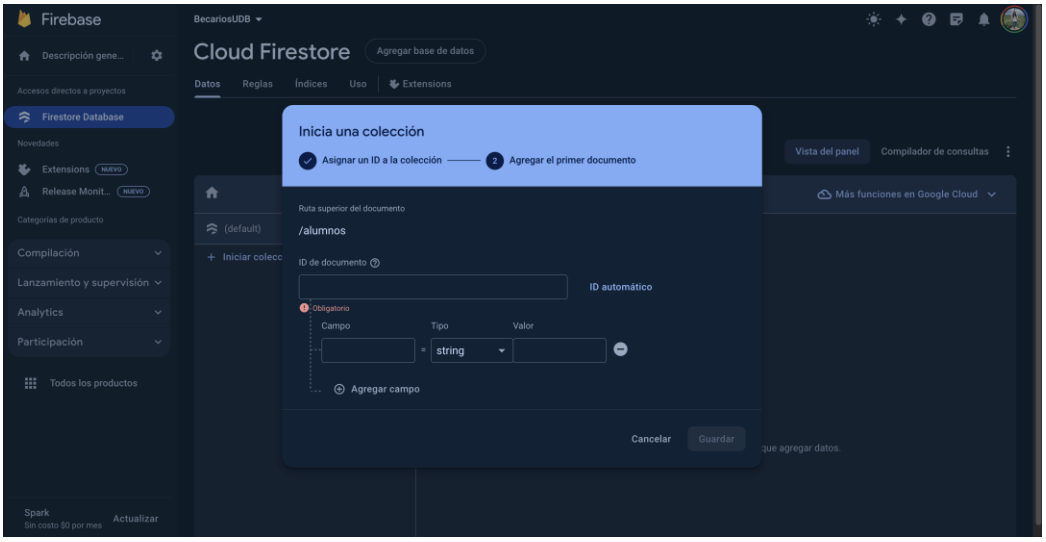
4- La base de datos ha sido creada, click en Iniciar colección:



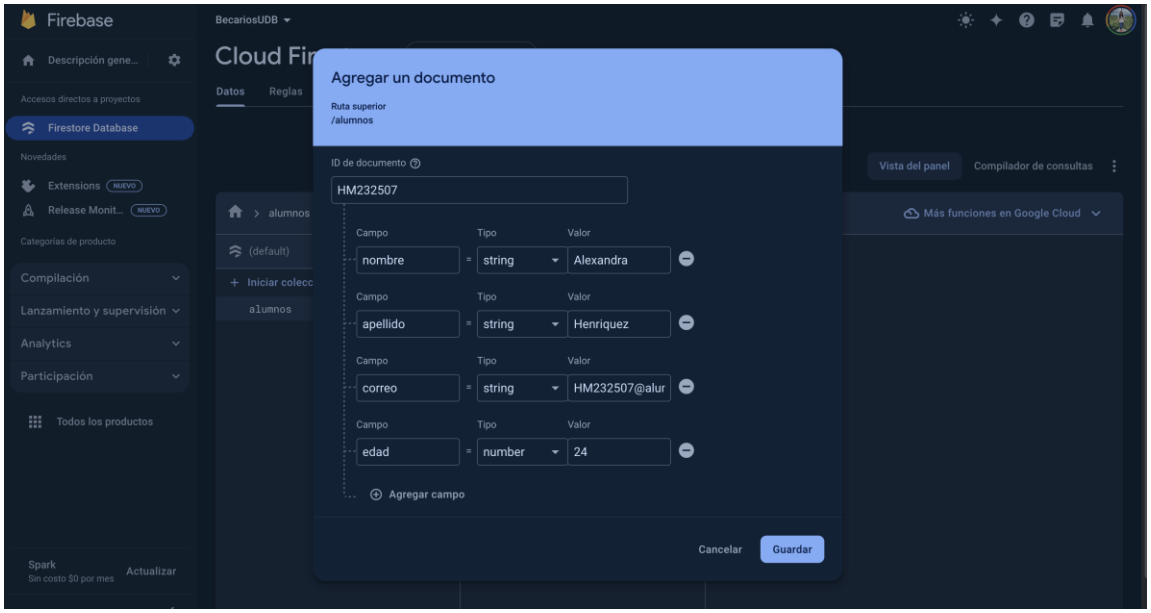
5- Inserte el nombre de la colección:



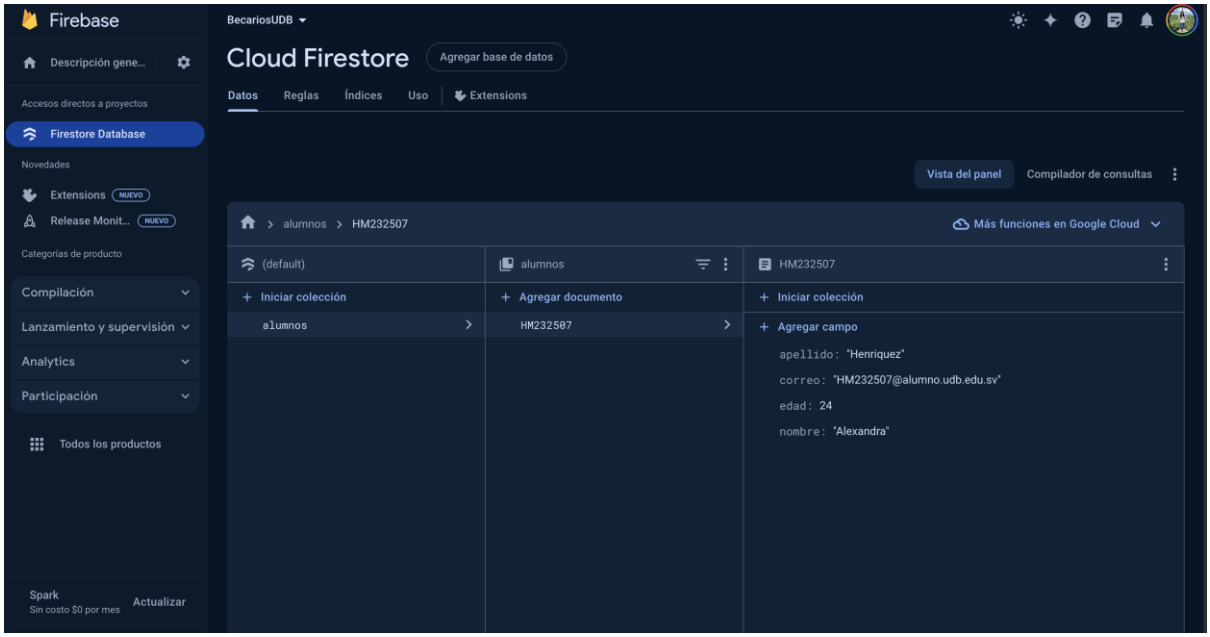
6- Click en Siguiente y el modal desplegará los siguientes campos:



7- Agregue la información en el documento:

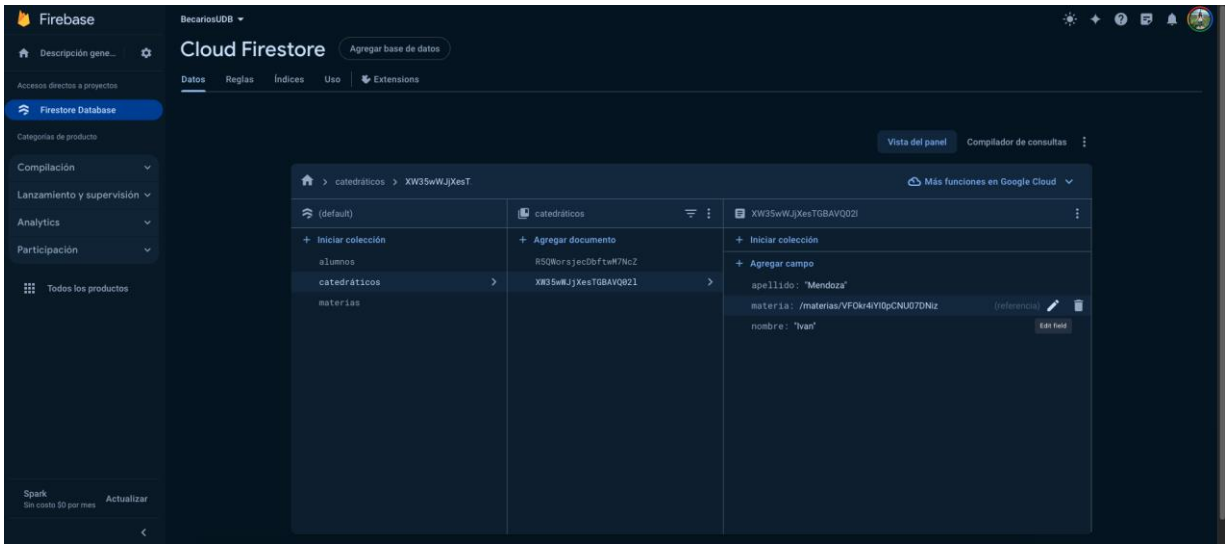


8- Guarde la información y el documento será desplegado en la colección:



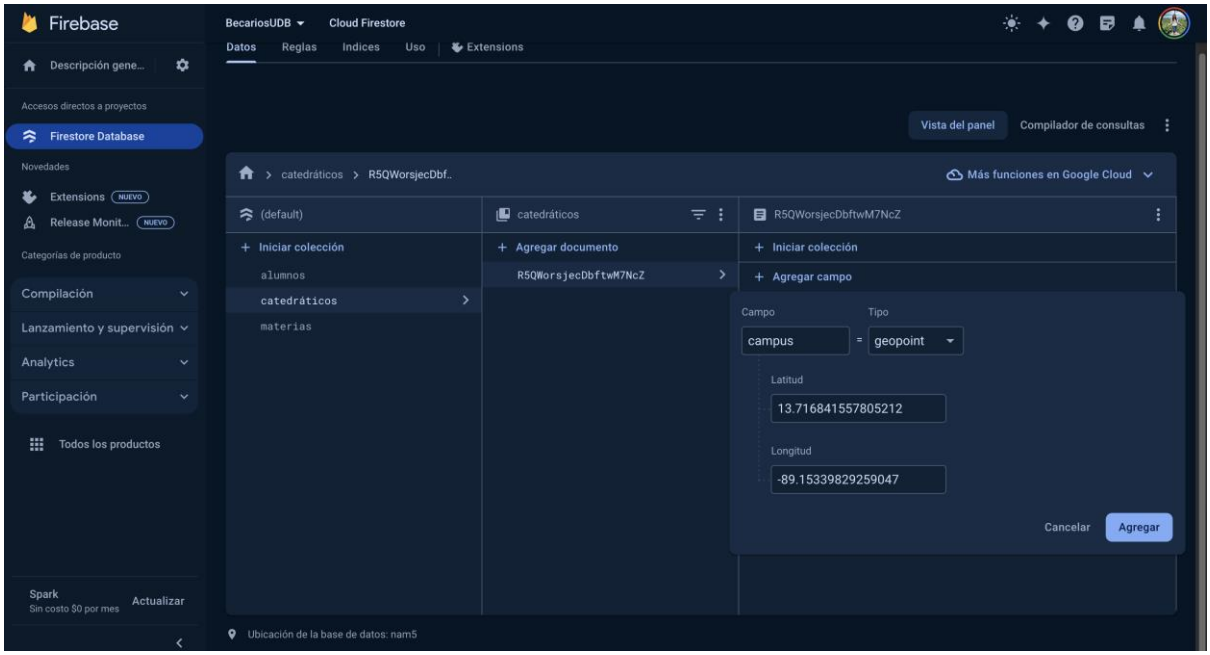
9- Agregue más colecciones y relaciónelas, por ejemplo:

Se agrega la colección “catedráticos”, ya hemos creado previamente la colección “materias”. Podemos relacionar de qué materia es el catedrático en el documento, creando un campo de tipo reference “materia” y escribiendo la colección y el ID del documento de la materia deseada.



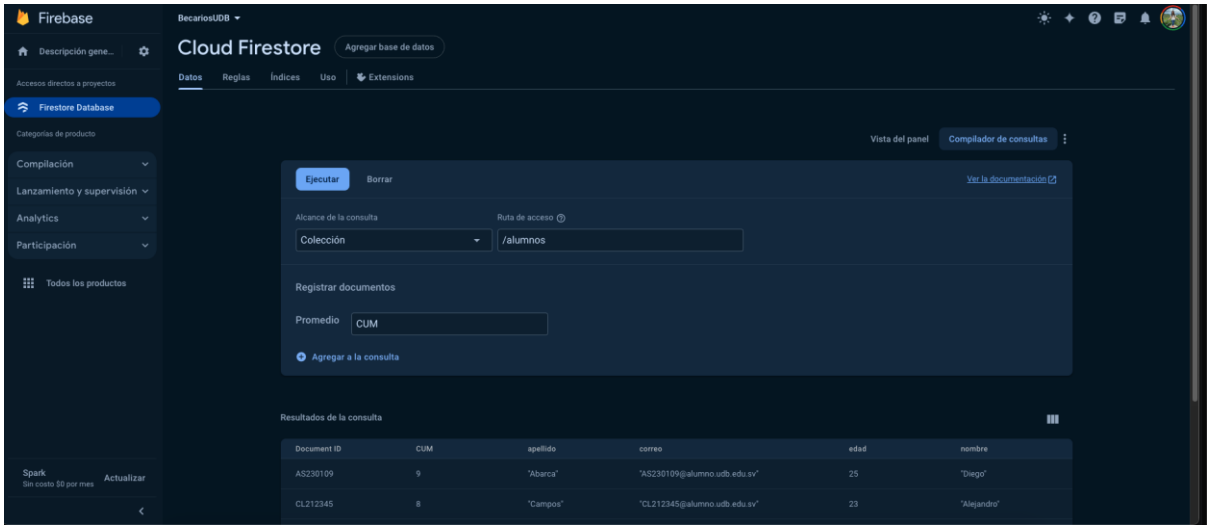
10- Podemos agregar también la ubicación de un elemento.

Por ejemplo, agregamos el campus al que pertenece el catedrático.

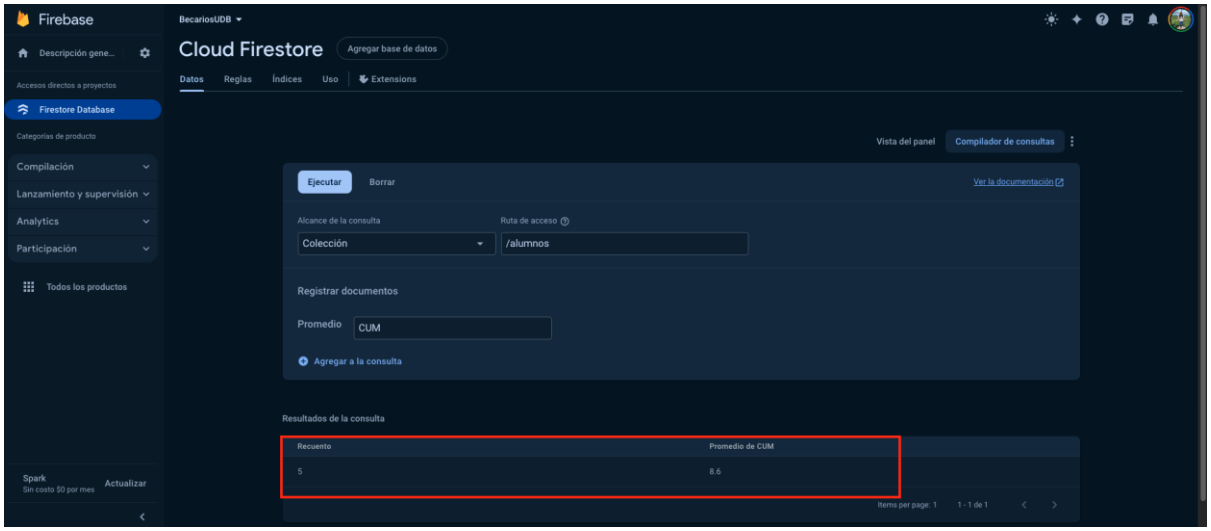


11- Llene las colecciones de datos

12- Use la opción de Compilador de Consultas para conocer valores como Promedio:

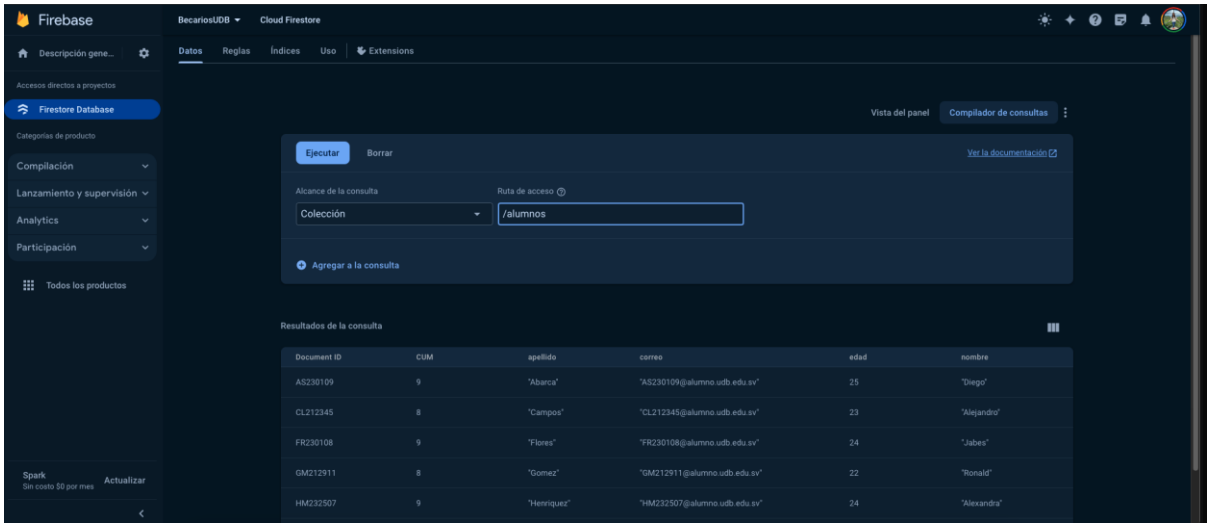


13- Ejecute el compilador de consultas y revise los resultados:

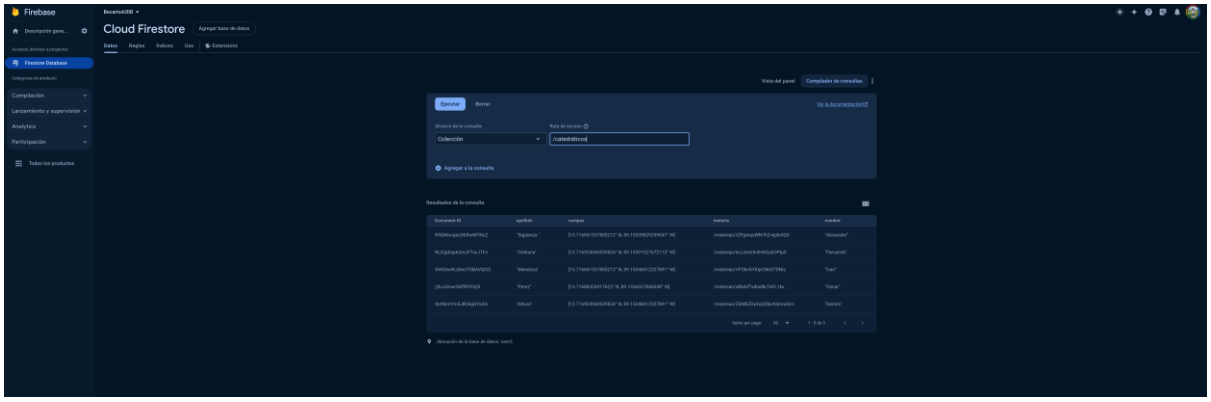


14-También puede visualizar la colección dentro de una tabla:

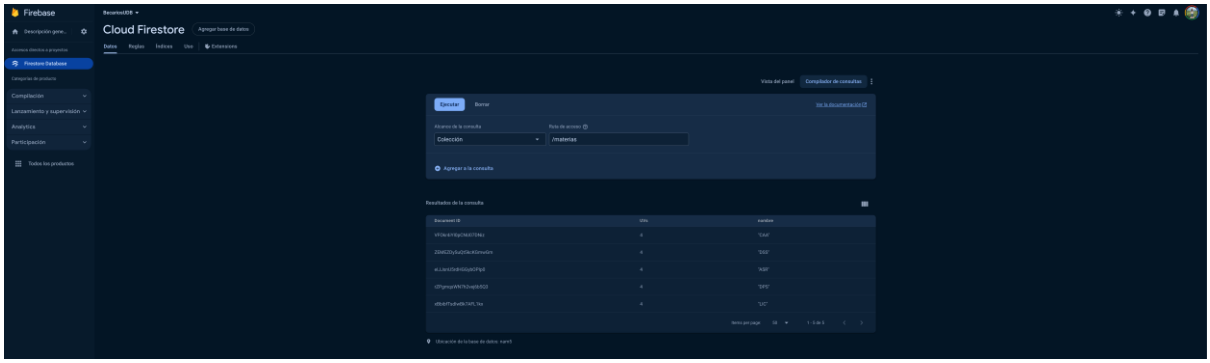
Alumnos:



Catedráticos:

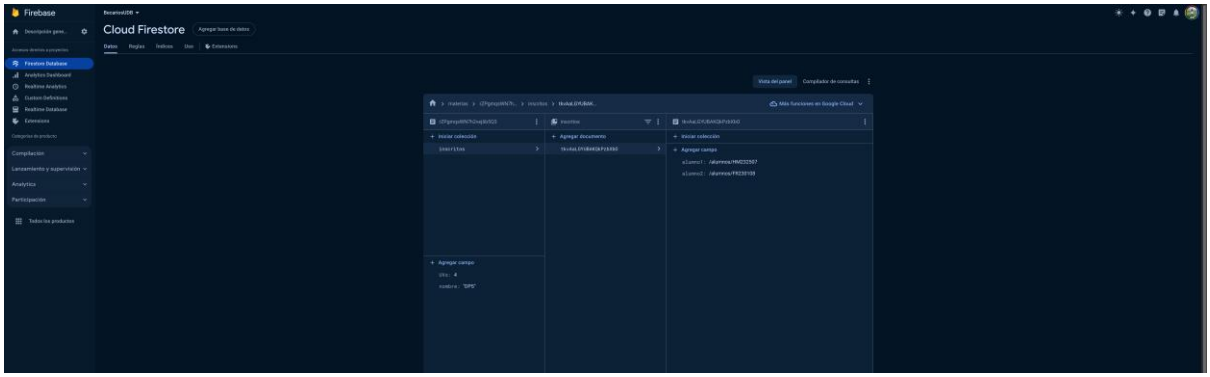


Materias:



15- También, puede crear una colección dentro de otra. Por ejemplo:

En la colección “materias” añadir una nueva colección llamada “inscritos” que tendrá una lista de los alumnos inscritos en esta materia.



Base de datos mySQL

```
CREATE DATABASE UDBVirtual;
USE UDBVirtual;

CREATE TABLE Profesores (
  id_profesor INT PRIMARY KEY,
  nombre VARCHAR(100),
  email VARCHAR(100)
);

CREATE TABLE Materias (
  id_materia INT PRIMARY KEY,
  nombre_materia VARCHAR(100)
);

CREATE TABLE Alumnos (
  id_alumno INT PRIMARY KEY,
  nombre VARCHAR(100),
  id_profesor INT,
  FOREIGN KEY (id_profesor) REFERENCES Profesores(id_profesor)
);

CREATE TABLE Asignaturas (
  id_asignatura INT PRIMARY KEY,
  nombre_asignatura VARCHAR(100),
  id_materia INT,
  FOREIGN KEY (id_materia) REFERENCES Materias(id_materia)
);
```

```
CREATE TABLE Evaluaciones (
  id_evaluacion INT PRIMARY KEY,
  tipo_evaluacion VARCHAR(50),
  nota DECIMAL(5,2),
  id_alumno INT,
  id_asignatura INT,
  FOREIGN KEY (id_alumno) REFERENCES Alumnos(id_alumno),
  FOREIGN KEY (id_asignatura) REFERENCES Asignaturas(id_asignatura)
);


INSERT INTO Profesores VALUES (1, 'Profesor 1', 'profesor1@udbvirtual.edu');
INSERT INTO Profesores VALUES (2, 'Juan Carlos', 'juancarlos@udbvirtual.edu');
INSERT INTO Profesores VALUES (3, 'Alejando Campos',
'alejandrocampos@udbvirtual.edu');
INSERT INTO Profesores VALUES (4, 'Jabes reyes', 'jabesreyes@udbvirtual.edu');
INSERT INTO Profesores VALUES (5, 'Alexandra', 'alexandra@udbvirtual.edu');
INSERT INTO Profesores VALUES (6, 'Diego', 'diego@udbvirtual.edu');



INSERT INTO Materias VALUES (1, 'Materia 1');
INSERT INTO Materias VALUES (2, 'Ciencia');
INSERT INTO Materias VALUES (3, 'Calculo');
INSERT INTO Materias VALUES (4, 'Ingles');
INSERT INTO Materias VALUES (5, ' Aplicaciones');
INSERT INTO Materias VALUES (6, 'Auditoria');
INSERT INTO Materias VALUES (7, 'Programacion');
```

```
INSERT INTO Alumnos VALUES (1, 'Alumno 1', 1);
INSERT INTO Alumnos VALUES (2, ' oscar', 2);
INSERT INTO Alumnos VALUES (3, 'erika', 3);
INSERT INTO Alumnos VALUES (4, 'karla', 4);
INSERT INTO Alumnos VALUES (5, 'carolina', 5);
INSERT INTO Alumnos VALUES (6, 'luis', 6);

INSERT INTO Asignaturas VALUES (1, 'Asignatura 1', 1);
INSERT INTO Asignaturas VALUES (2, 'Ciencia', 2);
INSERT INTO Asignaturas VALUES (3, 'Calculo' , 3);
INSERT INTO Asignaturas VALUES (4, 'Ingles' ,4);
INSERT INTO Asignaturas VALUES (5, ' Aplicaciones' ,5);
INSERT INTO Asignaturas VALUES (6, 'Auditoria' ,6);
INSERT INTO Asignaturas VALUES (7, 'Programacion' ,7);

INSERT INTO Evaluaciones VALUES (1, 'Examen 1', 95.00, 1, 1);
INSERT INTO Evaluaciones VALUES (2, 'Examen 2', 55.20, 2, 2);
INSERT INTO Evaluaciones VALUES (3, 'Examen 3', 60.00, 3, 3);
INSERT INTO Evaluaciones VALUES (4, 'Examen 4', 90.50, 4, 4);
INSERT INTO Evaluaciones VALUES (5, 'Examen 5', 75.10, 5, 5);
INSERT INTO Evaluaciones VALUES (6, 'Examen 6', 85.00, 6, 6);
```

La base de datos se guia de las siguientes tablas, profesores, materias, alumnos, asignaturas y evaluaciones.

Profesores, almacena informacion sobre los profesores.

Materias, almacena informacion sobre las materias.

Alumnos, almacena información sobre los alumnos y a que profesor estan asignados.

Asignaturas, almacena informacion sobre las asignaturas y a que profesor estan asignados.

Evaluaciones, almacena informacion sobre las evaluaciones incluyendo el tipo de evaluacion, la nota obtenida, y a que alumno y a asignatura pertenecen.

Finalmente, después de la implementación de las bases sql y Nosql proporcionar conclusiones basadas en su experiencia y en la investigación realizada

Con respecto a la escalabilidad, una de las ventajas de las bases NOsql suelen ser más escalables de forma horizontal, es decir que pueden manejar grandes volumenes de datos.

Las bases de datos NoSQL a menudo son más económicas de escalar y mantener, ya que pueden ejecutarse en hardware de menor costo y requerir menos administración. Las bases de datos SQL pueden requerir una inversión inicial mayor en infraestructura y pueden ser más costosas de mantener especialmente a medida que crece el tamaño de la base de datos.
la elección entre las bases de datos SQL y NoSQL depende de los requisitos específicos del proyecto incluyendo la estructura de datos, el volumen de datos, el rendimiento deseado y las necesidades de escalabilidad y disponibilidad.