UNIVERSIDAD DON BOSCO

7-5-2023

Foro de discusión 1

Asignatura:
Diseño y Programación de Software Multiplataforma

Docente:
Ing. Alexander Sigüenza

Alumnos:
Gerardo Antonio Cabezas Vaquero CV152055
Yosselin Beatriz Martínez Alvarado MM110166
Gabriela María Pineda González PG120866

Tabla de contenido

1. Investigue sobre los dos tipos de base de datos NoSQL que ofrece Firebase ... 3
Ventajas y Desventajas ... 4
Ventajas ... 4
Desventajas ... 4
Cloud Firestore ... 5
Funcionamiento ... 5
Documentos ... 5
Colecciones ... 6
Referencias ... 7
Subcolecciones ... 7
Tipo de Datos ... 8
Orden de los tipos de valor ... 8
Implementación ... 9
Realtime Database ... 9
Ventajas y Funciones Claves ... 10
Funcionamiento ... 10
Implementación ... 10
Diferencia entre Base de Datos NoSQL y SQL ... 11
Base de Datos Relacionales ... 11
Base de Datos No Relacionales ... 11
Diferencias ... 12
¿Por qué usar una base de datos NoSQL? ... 13
¿Las diferencias entre Firestore y Realtime? ... 14
Modelo de datos ... 14
Compatibilidad sin conexión y tiempo real ... 14
Realizar consultas ... 14
Escrituras ... 15
Seguridad ... 15
Rendimiento ... 15
¿La mejor opción para implementar React Native? ... 16
Función de la base de datos ... 16

Operaciones con datos 16
Modelo de datos 16
Disponibilidad 16
Consultas sin conexión en datos locales 17
Cantidad de instancias de base de datos 17
2. En base a la investigación realizada en el punto anterior, desarrolle el siguiente ejercicio, tomando en cuenta que deberá seleccionar una de las dos opciones que ofrece Firebase: Firestore o Realtime. 17
Creando base de datos 17
Base de Datos transaccional SQL server 19
Diagrama de relación: 19
Bade de Datos Firestores 20
Colección de Alumno 23
Colección Asignatura 23
Colección de Becas 24
Colección de Detalle_Alumno_Asignatura 24
Colección Detalle_Profesor_Asignatura 25
Colección NivelesAcademicos 25
Colección Persona 26
Colección Profesor 26
Conclusiones 27
Bibliografía 28

1. INVESTIGUE SOBRE LOS DOS TIPOS DE BASE DE DATOS NOSQL QUE OFRECE FIREBASE

Firebase es la plataforma móvil de Google en la nube para el desarrollo de aplicaciones web y móvil. Está disponible para distintas plataformas (iOS, Android y web), con lo que es más rápido trabajar en el desarrollo.

Su función esencial es hacer más sencilla la creación de tanto aplicaciones webs como móviles y su desarrollo, procurando que el trabajo sea más rápido, pero sin renunciar a la calidad requerida.

Sus herramientas son variadas y de fácil uso, considerando que su agrupación simplifica las tareas de gestión a una misma plataforma. Las finalidades de estas se pueden dividir en cuatro grupos: desarrollo, crecimiento, monetización y análisis.

Firebase dispone de diferentes funcionalidades, que se pueden dividir básicamente en 3 grupos: Desarrollo (Develop), Crecimiento (Grow) y Monetización (Earn), a los que hay que sumar la Analítica (Analytics).

Desarrollo: Como su nombre indica, incluye los servicios necesarios para el desarrollo de un proyecto de aplicación móvil o web.

Crecimiento: está enfocado al proceso de crecimiento de la aplicación, que contempla tanto la gestión de aquellos que ya son usuarios de esta, como herramientas para la captación de nuevas audiencias.

Monetización: En este caso, la búsqueda de ganancias viene ligada a la publicidad que se puede insertar en las aplicaciones.

Analítica: Con Firebase Analytics, puedes controlar diversos parámetros y obtener mediciones variadas desde un mismo panel de manera gratuita.

Algunas características pueden ser:

Es multiplataforma: Soportada por Android, iOS y web.

Monetización: A través de Firebase podemos ganar dinero esto a través de AdMob con anuncios y publicidad.

Gran poder de crecimiento: Gracias a la fácil gestión de los usuarios de las aplicaciones es posible obtener un alto crecimiento según los objetivos planteados. Esta herramienta cuenta con el valor añadido de que podemos llegar a nuevos usuarios con el envío de notificaciones e invitaciones.

Es Ágil: Ofrece el desarrollo y gestión de apps multiplataforma gracias a sus APIs integradas a SDK tanto para JavaScript como para iOS y Android, permitiendo gestionar diferentes aplicaciones sin la necesidad de la salir de la plataforma.

VENTAJAS Y DESVENTAJAS

Ventajas:

I. Muy recomendable para aplicaciones que necesiten compartir datos en tiempo real.
II. Sus funcionalidades, además de ser variadas, se complementan muy bien y se pueden gestionar de forma sencilla desde un único panel. Además, no es necesario usar todas estas opciones para la aplicación, pudiendo elegir solo aquellas que más nos interesen.
III. Facilita el envío de notificaciones: son muy sencillas de implementar y gestionar, además de ser extremadamente útiles para mantener la atención de los usuarios.
IV. Permite la monetización: desde el propio Firebase se puede agregar publicidad a la app, permitiendo fácilmente rentabilizarla obteniendo el ROI.
V. Gran ventaja Cloud Storage, la cual nos permite contar con una base de datos para que el usuario pueda contar con un espacio de almacenamiento y compartir imágenes y ficheros; sin recurrir a bases de datos propias.
VI. En líneas generales, sus funcionalidades se complementan entre sí a pesar de ser variadas. Firebase facilita los eventos en cuanto a el envío de notificaciones que gracias a su simpleza son de fácil uso y nos permiten centrar la atención de los usuarios. Cuenta con SSL (Secure Sockets Layer).
VII. Soporte gratuito vía email, sin importar si el desarrollador utiliza la versión gratuita o de pago.
VIII. Escalabilidad: los inicios son gratuitos, pero permite ir adaptándose a las necesidades de la aplicación con diferentes planes de pago.

Desventajas:

I. La desventaja más comúnmente mencionada es el precio. Se ha hablado de la escalabilidad de Firebase, donde el inicio con el plan Spark es gratuito.

CLOUD FIRESTORE

Cloud Firestore es una base de datos de documentos NoSQL que permite almacenar, sincronizar y consultar fácilmente datos en tus apps web y para dispositivos móviles a escala global.

Mantiene tus datos sincronizados entre apps cliente a través de objetos de escucha en tiempo real y ofrece soporte sin conexión para dispositivos móviles y la Web, por lo que puedes compilar apps con capacidad de respuesta que funcionan sin importar la latencia de la red ni la conectividad a Internet. Cloud Firestore también ofrece una integración sin interrupciones con otros productos de Firebase y Google Cloud, incluido Cloud Functions.

Funcionamiento:

Cloud Firestore es una base de datos NoSQL alojada en la nube a la que pueden acceder a las aplicaciones para Apple, Android y la Web directamente desde los SDK nativos. Cloud Firestore también está disponible en los SDKs nativos de Node.js, Java, Python, Unity, C++ y Go, además de las APIs de REST y RPC.

A partir del modelo de datos NoSQL de Cloud Firestore, almacenas los datos en documentos que contienen campos que se asignan a valores. Estos documentos se almacenan en colecciones, que son contenedores para tus documentos y que puedes usar para organizar tus datos y compilar consultas. Los documentos admiten varios tipos de datos diferentes, desde strings y números simples, hasta objetos anidados complejos. También puedes crear subcolecciones dentro de documentos y crear estructuras de datos jerárquicas que se ajustan a escala a medida que tu base de datos crece. El modelo de datos de Cloud Firestore admite cualquier estructura de datos que funcione mejor con tu aplicación.

Protege el acceso a tus datos en Cloud Firestore con Firebase Authentication y las reglas de seguridad de Cloud Firestore para Android, las plataformas de Apple y JavaScript, o bien con la administración de identidades y accesos (IAM) para los lenguajes del servidor.

A diferencia de una base de datos SQL, no hay tablas ni filas. En su lugar, almacenas los datos en documentos, que se organizan en colecciones.

Documentos:

Cada documento contiene un conjunto de pares clave-valor. Cloud Firestore está optimizado para almacenar grandes colecciones de documentos pequeños.

Ejemplo: Documento que representa a un usuario gvaquero puede tener el siguiente aspecto:

gvaquero

firstName : “Gerardo”

lastName : “Vaquero”

born : 1993

Los objetos complejos anidados en un documento se llaman mapas. Por ejemplo, podríamos estructurar el nombre del usuario del ejemplo anterior con un mapa como este:

gvaquero

name:

firstName : “Gerardo”

lastName : “Vaquero”

born : 1993

Colecciones:

Los documentos viven en colecciones, que simplemente son contenedores de documentos.

Ejemplo con una colección llamada users:

users

gvaquero

firstName : “Gerardo”

lastName : “Vaquero”

born : 1993

klucero

firstName : “Katherine”

lastName : “Lucero”

born : 1993

Referencias:

Cada documento de Cloud Firestore se identifica de manera inequívoca por su ubicación dentro de la base de datos. El ejemplo anterior muestra un documento gvaquero en la colección users. Para hacer referencia a esta ubicación en tu código, puedes crear una referencia a ella.

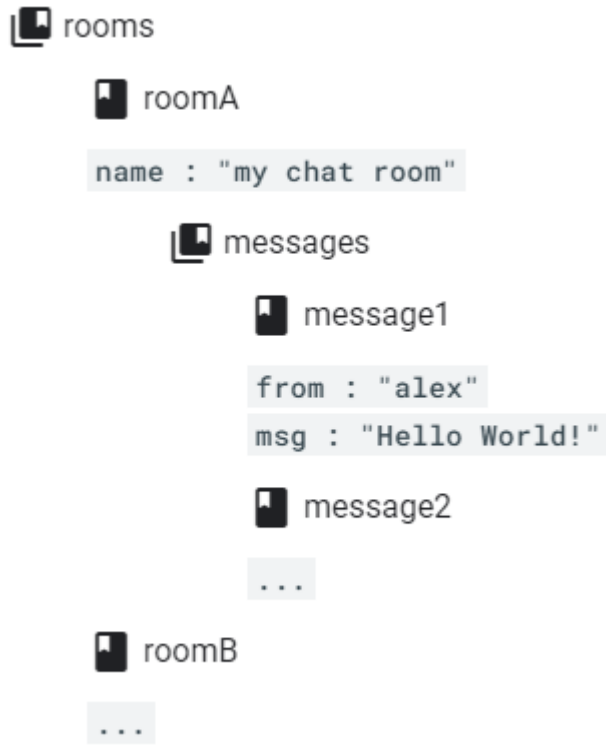
```
import { doc } from "firebase/firestore";

const gvaqueroDocumentRef = doc(db, 'users', 'gvaquero');
```

Subcolecciones:

La mejor manera de almacenar mensajes en este caso es usar subcolecciones. Una subcolección es una colección asociada con un documento específico.

Puedes crear una subcolección llamada messages para cada documento de la sala que integra la colección rooms:



Los documentos de las subcolecciones también pueden contener subcolecciones, lo que te permite anidar datos en más niveles. Puedes anidar datos hasta 100 niveles de profundidad.

Tipo de Datos:

Tipo de datos	Clasificación	Descripción
Arreglo	Según los valores de los elementos	Un arreglo no puede contener otro valor de arreglo como uno de sus elementos.
Booleano	false < true	
Bytes	Orden de bytes	Hasta 1,048,487 bytes (de 1 MiB a 89 bytes). Las consultas solo consideran los primeros 1,500 bytes.
Fecha y Hora	Cronológico	Cuando se almacena en Cloud Firestore, la precisión máxima es de microsegundos; cualquier precisión adicional se redondea hacia abajo.
Numero de coma flotante	Numérico	Precisión doble de 64 bits, IEEE 754.
Punto geográfico	Por latitud, luego por longitud	Por el momento, no recomendamos usar este tipo de datos debido a las limitaciones de consultas.
Número entero	Numérico	64 bits, firmado
Mapa	Por claves y, luego, por valor	Representa un objeto incorporado en un documento. Cuando se indexa, puedes hacer consultas por subcampos. Si excluyes este valor de la indexación, también se excluyen todos los subcampos.
Nulo	Ninguno	
Referencia	Por elementos de ruta de acceso (colección, ID de documento, colección, ID de documento…)	Por ejemplo, projects/[PROJECT_ID]/databases/[DATABASE_ID]/documents/[DOCUMENT_PATH].
String de texto	Orden de bytes con codificación UTF-8	Hasta 1,048,487 bytes (de 1 MiB a 89 bytes). Las consultas solo consideran los primeros 1,500 bytes de la representación UTF-8.

Orden de los tipos de valor:

Cuando una consulta incluye un campo con valores de varios tipos, Cloud Firestore usa un orden determinista basado en representaciones internas. La lista siguiente muestra el orden:

1. Valores nulos
2. Valores booleanos
3. Valores de números enteros y de coma flotante, en orden numérico
4. Valores de fechas
5. Valores de strings de texto
6. Valores de bytes
7. Referencias de Cloud Firestore
8. Valores de puntos geográficos
9. Valores de matrices
10. Valores de mapas

Implementación:

Integra los SDK de Cloud Firestore: Incluye clientes rápidamente mediante Gradle, CocoaPods o un script.

Protege los datos: Usa las reglas de seguridad de Cloud Firestore o Identity and Access Management (IAM) a fin de proteger tus datos para la programación para dispositivos móviles o la Web y en el servidor, respectivamente.

Agrega datos: Crea documentos y colecciones en tu base de datos.

Obtén datos: Crea consultas o usa agentes de escucha en tiempo real para recuperar datos de la base de datos.

Pasos para crear una base de datos Cloud Firestore:

1. En Firebase console, haz clic en Agregar proyecto y, luego, sigue las instrucciones en pantalla a fin de crear un proyecto de Firebase o agregar servicios de Firebase a un proyecto de GCP existente.
2. Ve a la sección Cloud Firestore de Firebase console. Se te pedirá que selecciones un proyecto de Firebase existente. Sigue el flujo de trabajo para crear la base de datos.
3. Selecciona uno de los siguientes modos de inicio para las reglas de seguridad de Cloud Firestore:
    - Modo de prueba
    - Modo bloqueado
4. Selecciona una ubicación para tu base de datos.

REALTIME DATABASE

Firebase Realtime Database es una base de datos NoSQL alojada en la nube que te permite almacenar y sincronizar datos entre tus usuarios en tiempo real.

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.

Este producto de Firebase lo podemos considerar una respuesta a la situación de que en el desarrollo de aplicaciones es necesario contemplar muchas cosas para almacenar y compartir datos de un servidor. Es oportuno decir que puede resultar demasiado complicado el hecho de configurar y mantener una base de datos.

Teniendo en cuenta lo anterior y que, de pronto, quieras sincronizar los datos en tiempo real para tu app o necesites asistencia cuando estás desconectado, surge Firebase Realtime Database, con la que puedes ahorrar mucho tiempo de trabajo y automatizar muchos procesos.

Ventajas y Funciones Claves

Cuando actualizas datos en Realtime Database, estos se almacenan en la nube y la plataforma envía una notificación a todos los dispositivos que se encuentran conectados a tu app en cuestión de segundos.

Se encuentra configurado para permitirte usarla sin conexión. Esto significa que en caso de que el usuario pierda la conexión a internet, el SDK de la base de datos hará uso de una caché local en el dispositivo, con el propósito de procesar y almacenar los cambios realizados durante ese tiempo y, cuando el usuario pueda conectarse otra vez, se sincronizarán los datos locales de forma automática.

Cuenta con un SDK para Android, iOS y JavaScript.

Se puede acceder a Firebase Realtime Database directamente desde un dispositivo móvil o un navegador web; no se necesita un servidor de aplicaciones. La seguridad y la validación de datos están disponibles a través de las reglas de seguridad de Firebase Realtime Database: reglas basadas en expresiones que se ejecutan cuando se leen o se escriben datos.

Podrás dividir la información en diversas instancias de bases de datos dentro del mismo proyecto de Firebase.

Usa Firebase Authentication para optimizar el proceso de autenticación en el proyecto.

Podrás autenticar a usuarios en varias instancias de la base de datos. Controla el acceso a la información de cada base de datos.

Funcionamiento

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.

Implementación

1. Integrar los SDK de Firebase Realtime Database.

Incluye clientes rápidamente mediante Gradle, CocoaPods o una secuencia de comandos.

2. Crear referencias de Realtime Database.

Haz referencia a tus datos JSON, como "users/user:1234/phone_number" para establecer datos o suscribirte a cambios de datos.

3. Configurar datos y detectar cambios.

Usa estas referencias para escribir datos o suscribirte a cambios.

4. Habilitar la persistencia sin conexión.

Permite que se escriban los datos en el disco local del dispositivo para que estén disponibles sin conexión.

5. Proteger los datos.

Usa reglas de seguridad de Firebase Realtime Database para proteger los datos.

DIFERENCIA ENTRE BASE DE DATOS NOSQL Y SQL

Base de Datos Relacionales

Son una colección de elementos de datos organizados en un conjunto de tablas formalmente descritas, desde donde se puede acceder a los datos o volver a montarlos de muchas maneras diferentes sin tener que reorganizar las tablas de la base. La interfaz estándar de programa de usuario y aplicación a una base de datos relacional es el Lenguaje de Consultas Estructuradas (SQL). Los comandos SQL se utilizan tanto para consultas interactivas como para obtener información de una base de datos relacional y la recopilación de datos para informes.

Las bases de datos relacionales se basan en la organización de la información en partes pequeñas que se integran mediante identificadores; a diferencia de las bases de datos no relacionales que, como su nombre lo indica, no tienen un identificador que sirva para relacionar dos o más conjuntos de datos.

Base de Datos No Relacionales

Están diseñadas específicamente para modelos de datos específicos y tienen esquemas flexibles para crear aplicaciones modernas. Son ampliamente reconocidas porque son fáciles de desarrollar, tanto en funcionalidad como en rendimiento a escala. Usan una variedad de modelos de datos, que incluyen documentos, gráficos, clave-valor, en-memoria y búsqueda.

Diferencias

student_id	age	score
1	12	77
2	12	68
3	11	75


```
[
 {
  "student_id":1,
  "age":12,
  "score":77
 },
 {
  "student_id":2,
  "age":12,
  "score":68
 },
 {
  "student_id":3,
  "age":11,
  "score":75
 }
]
```


NoSQL se refiere a una base de datos no relacional o no SQL. Una base de datos relacional es un formato de bases de datos muy estructurado basado en una tabla, como MySQL u Oracle. Las bases de datos NoSQL están orientadas a los documentos y le permiten almacenar y recuperar datos en formatos que no sean tablas. Algunas de las plataformas NoSQL más conocidas son MongoDB, Elasticsearch® y Redis®.

Las bases de datos relacionales utilizan un lenguaje de consulta estructurado para la manipulación de datos, estas se conforman por filas, columnas y registros y se almacenan por tablas. Para manipular los datos en SQL, se requiere primero determinar la estructura de estos, si se cambia la estructura de uno de los datos, puede perjudicar todo el sistema, ya que las tablas están relacionadas.

Las aplicaciones modernas usan y generan tipos de datos complejos y que cambian constantemente, y las bases de datos relacionales no fueron diseñadas para gestionar este tipo de almacenamiento y recuperación de datos. Las bases de datos NoSQL son más flexibles y escalables.

Al trabajar con una base de datos NoSQL, usted puede agregar datos nuevos, sin tener que definirlos previamente en el esquema de la base de datos, lo que le permite procesar rápidamente grandes volúmenes de datos sin estructura, semiestructurados y estructurados.

El esquema dinámico de bases de datos NoSQL permite realizar desarrollos ágiles, que requieren iteraciones rápidas y significativas y durante los que no puede haber tiempo de inactividad.

¿Por qué usar una base de datos NoSQL?

- Costos reducidos
- Mejor rendimiento
- Mejores datos
- Velocidad de llegada al mercado

¿LAS DIFERENCIAS ENTRE FIRESTORE Y REALTIME?

Firebase ofrece dos soluciones de bases de datos basadas en la nube y accesibles para los clientes que admiten sincronización en tiempo real:

Cloud Firestore: es la base de datos más reciente de Firebase para el desarrollo de apps para dispositivos móviles. Aprovecha lo mejor de Realtime Database con un modelo de datos nuevo y más intuitivo. Con Cloud Firestore también se pueden realizar consultas más ricas y rápidas, y el escalamiento se ajusta a un nivel más alto que Realtime Database.

Realtime Database: es la base de datos original de Firebase. Es una solución eficiente y de baja latencia destinada a las apps para dispositivos móviles que necesitan estados sincronizados entre los clientes en tiempo real.

Modelo de datos

Realtime Database	Cloud Firestore
Almacena datos como un gran árbol JSON.	Almacena datos como colecciones de documentos.
Los datos simples son muy fáciles de almacenar.	Los datos simples son fáciles de almacenar en documentos, que son muy similares a JSON.
Los datos complejos y jerárquicos son más difíciles de organizar a escala.	Los datos complejos y jerárquicos son más fáciles de organizar a escala, con subcolecciones dentro de los documentos.
	Necesita menos desnormalización y compactación de datos.

Compatibilidad sin conexión y tiempo real

Realtime Database	Cloud Firestore
Soporte sin conexión para clientes de Apple y Android.	Soporte sin conexión para clientes de Apple, Android y la Web.

Realizar consultas

Realtime Database	Cloud Firestore
Consultas directas con funciones de ordenamiento y filtrado limitadas.	Consultas indexadas con ordenamiento y filtrado compuestos.
Las consultas pueden ordenar o filtrar en una propiedad, pero no ambas opciones.	Las consultas se indexan de forma predeterminada: el rendimiento de las consultas es proporcional al tamaño del conjunto de resultados, no del conjunto de datos.
Las consultas no requieren un índice. Sin embargo, el rendimiento de ciertas consultas se degrada a medida que crece el conjunto de datos.	Las consultas son superficiales: solo muestran documentos de colecciones o grupos de colecciones específicos y no muestran datos de subcolecciones.
Las consultas son profundas según la configuración predeterminada: siempre muestran el subárbol completo.	Las consultas siempre deben mostrar documentos completos.
Las consultas pueden acceder a los datos en cualquier nivel de detalle, hasta valores de nodo de hoja individuales en el árbol JSON.	Puedes encadenar filtros y combinar filtrado con ordenamiento según una propiedad en la misma consulta.

Escrituras

Realtime Database	Cloud Firestore
Operaciones básicas de escritura y transacción.	Operaciones avanzadas de escritura y transacción.
Las transacciones son atómicas en un subárbol de datos específico.	Las transacciones pueden leer y escribir datos atómicamente desde cualquier parte de la base de datos.
Permite escribir datos mediante operaciones de configuración y actualización.	Permite escribir datos a través de operaciones de configuración y actualización, así como transformaciones avanzadas, como operadores de arreglos y numéricos.

Seguridad

Realtime Database	Cloud Firestore
Lenguaje de reglas en cascada que separa la autorización de la validación.	Reglas sin formato de cascada que combinan autorización y validación.
Transmisión en cascada de reglas de lectura y escritura.	Las reglas pueden restringir consultas: Si los resultados de una consulta pudiesen contener datos a los que el usuario no tiene acceso, la consulta completa falla.
Las reglas de seguridad de Realtime Database protegen las operaciones de lectura y escritura desde los SDK para dispositivos móviles.	Identity and Access Management (IAM) protege las operaciones de lectura y escritura desde los SDK para servidores.
Debes validar los datos por separado con la regla validate.	Las reglas no se aplican en cascada, a menos que uses un comodín.
	Las reglas de seguridad de Cloud Firestore protegen las operaciones de lectura y escritura desde los SDK para dispositivos móviles.

Rendimiento

Realtime Database	Cloud Firestore
Realtime Database es una solución regional	Cloud Firestore es una solución regional y multirregional con ajuste de escala automático
Latencia extremadamente baja, es la opción ideal para sincronizar estados con frecuencia.	Está disponible en configuraciones regionales o multirregionales en todo el mundo.
Está disponible en configuraciones regionales. Las bases de datos se limitan a la disponibilidad zonal de una región.	Aloja los datos en varios centros de datos de distintas regiones, lo que garantiza una escalabilidad global y una confiabilidad sólida.

¿LA MEJOR OPCIÓN PARA IMPLEMENTAR REACT NATIVE?

De acuerdo con lo que hemos estudiado de las bases de datos SQL y NoSQL podemos analizar el tipo y las necesidades que nuestra aplicación va a solventar y lo mucho que puede llegar a crecer en un tiempo determinado.

Hemos aprendido que una base de datos no transaccional puede ayudar a una aplicación que va a almacenar una gran cantidad de información e incluso su estructura puede cambiar con el tiempo a comparación de las transaccionales que su estructura no puede llegar a cambiar tan fácilmente.

Firebase nos da la opción de poder elegir entre Realtime Database y Cloud Firestore pero hay un punto muy importante que debemos analizar y es que al implementar nuestra aplicación con React Navite deducimos que esta tecnología se utiliza para desarrollar aplicaciones móviles para Android o iOS, por lo cual Cloud Firestore es una base de datos más reciente y está diseñada para apps en dispositivos móviles pero Realtime Database es una solución eficiente y de baja latencia destinada a las apps para dispositivos móviles.

Concluimos que dependerá de la necesidad y del análisis que realicemos que podremos elegir una base de datos de acuerdo con nuestra aplicación.

Algunas consideraciones claves que pueden ayudar a elegir una opción serían las siguientes:

Función de la base de datos: Es decir para qué va a funcionar mi base de datos.

Realtime Database: Si no necesitas realizar consultas, transacciones ni ordenamientos avanzados.

Cloud Firestore: Si necesitas realizar interacciones complejas con tus datos.

Operaciones con datos: Cantidad de espacio en GB que podemos analizar que puede llegar a crecer la información.

Realtime Database: Si tu app envía un flujo de pequeñas actualizaciones.

Cloud Firestore: Para conjuntos de datos muy grandes y cuando se necesitan operaciones por lotes.

Modelo de datos: Tipo de estructura de la base de datos si deseamos un árbol JSON o colecciones.

Realtime Database: Para datos JSON no estructurados.

Cloud Firestore: Para los documentos y colecciones estructurados.

Disponibilidad: Garantía en tiempo de actividad.

Realtime Database: Cuando se acepta una disponibilidad muy alta, pero que no llega a ser crítica.

Cloud Firestore: Si la disponibilidad es de suma importancia.

Consultas sin conexión en datos locales: Mi aplicación deberá realizar consultas en dispositivos con conectividad limitada o sin conexión.

Realtime Database: Si esperas que tus usuarios estén constantemente en línea.

Cloud Firestore: Para capacidades de consulta sofisticadas en datos locales cuando el usuario no tiene conexión.

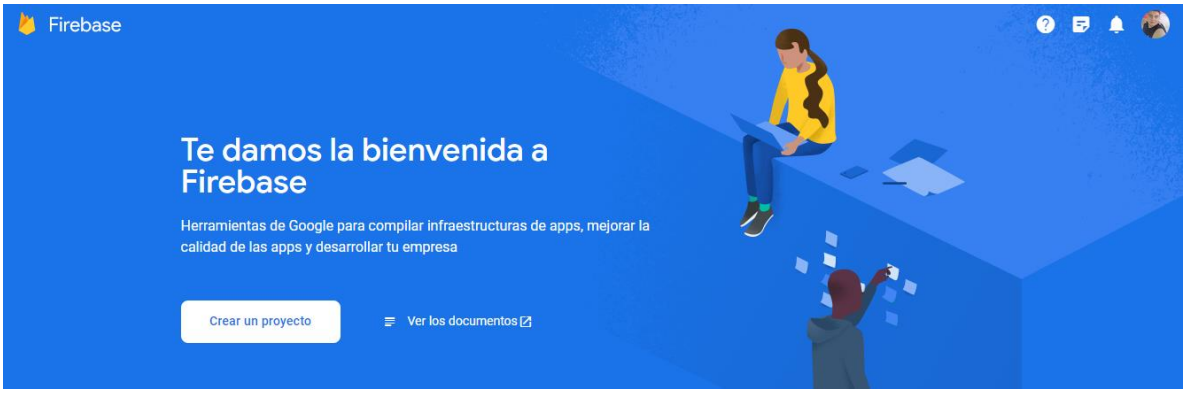
Cantidad de instancias de base de datos: Si necesito usar una o más bases de datos.

Realtime Database: permite agregar varias bases de datos a un solo proyecto.

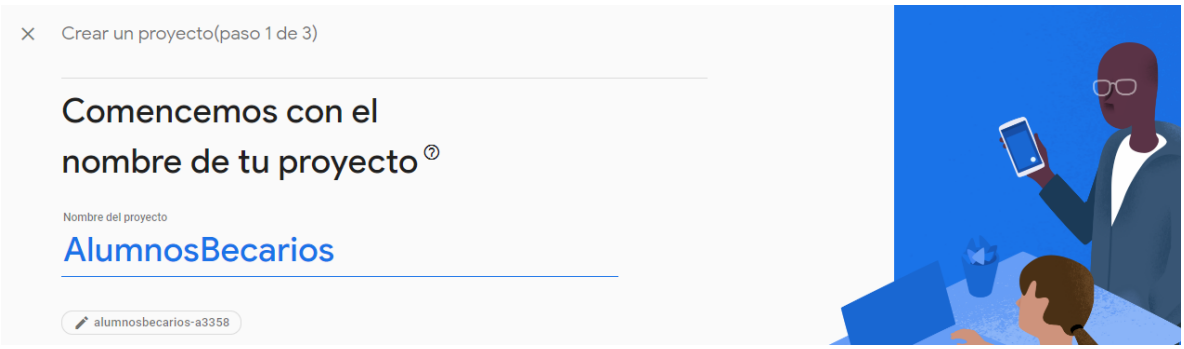
Cloud Firestore: Si necesitas una sola base de datos.

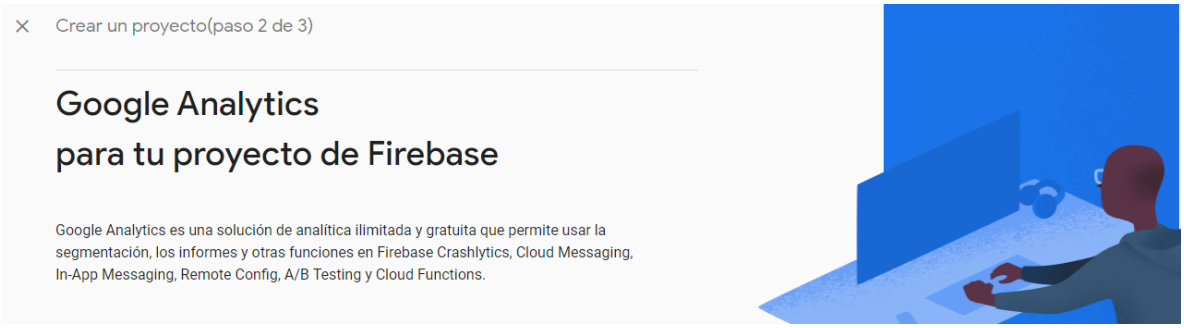
2. EN BASE A LA INVESTIGACIÓN REALIZADA EN EL PUNTO ANTERIOR, DESARROLLE EL SIGUIENTE EJERCICIO, TOMANDO EN CUENTA QUE DEBERÁ SELECCIONAR UNA DE LAS DOS OPCIONES QUE OFRECE FIREBASE: FIRESTORE O REALTIME.

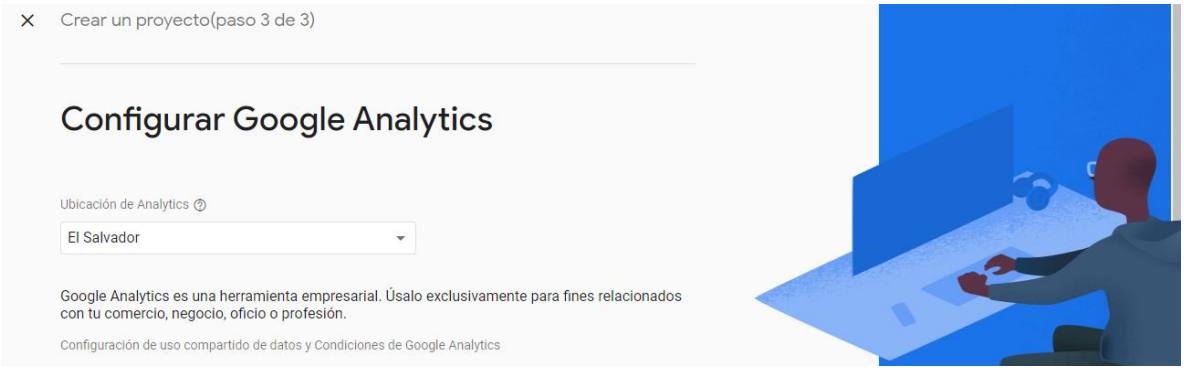
Creando base de datos:

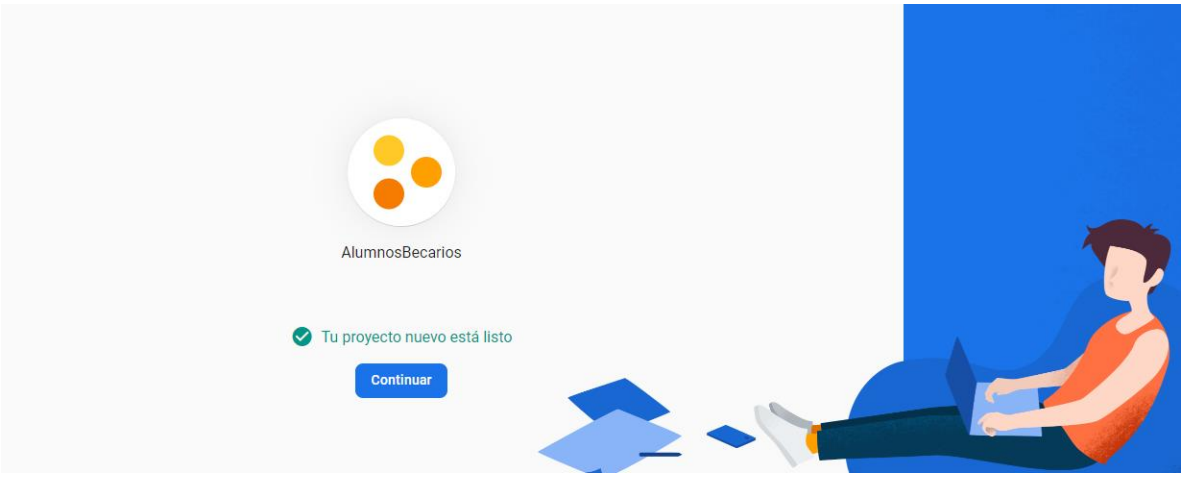


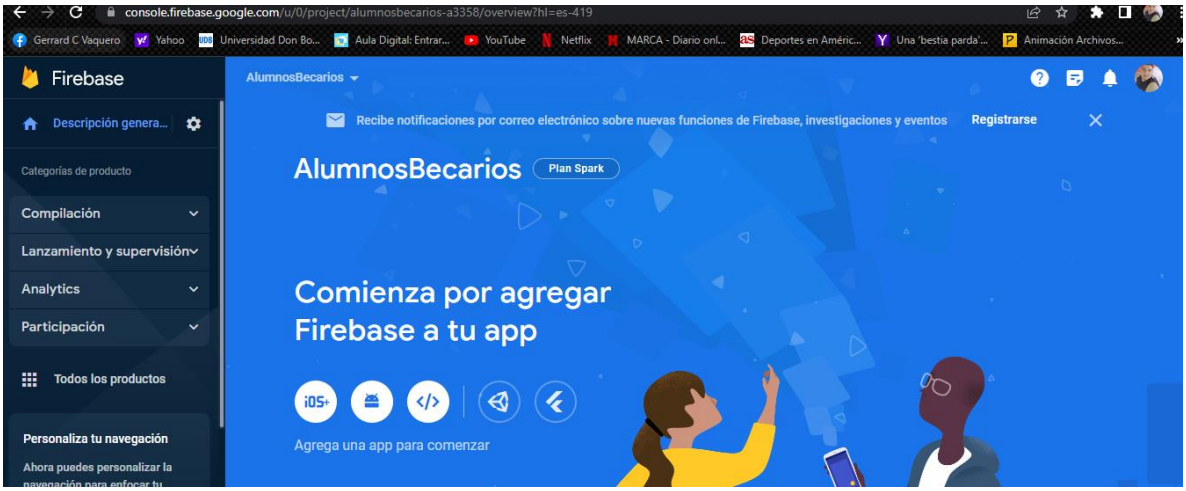
alumnosbecarios-a3358


Crear un proyecto(paso 1 de 3)
Comencemos con el nombre de tu proyecto
Nombre del proyecto
AlumnosBecarios
alumnosbecarios-a3358


Crear un proyecto(paso 2 de 3)
Google Analytics para tu proyecto de Firebase
Google Analytics es una solución de analítica ilimitada y gratuita que permite usar la segmentación, los informes y otras funciones en Firebase Crashlytics, Cloud Messaging, In-App Messaging, Remote Config, A/B Testing y Cloud Functions.

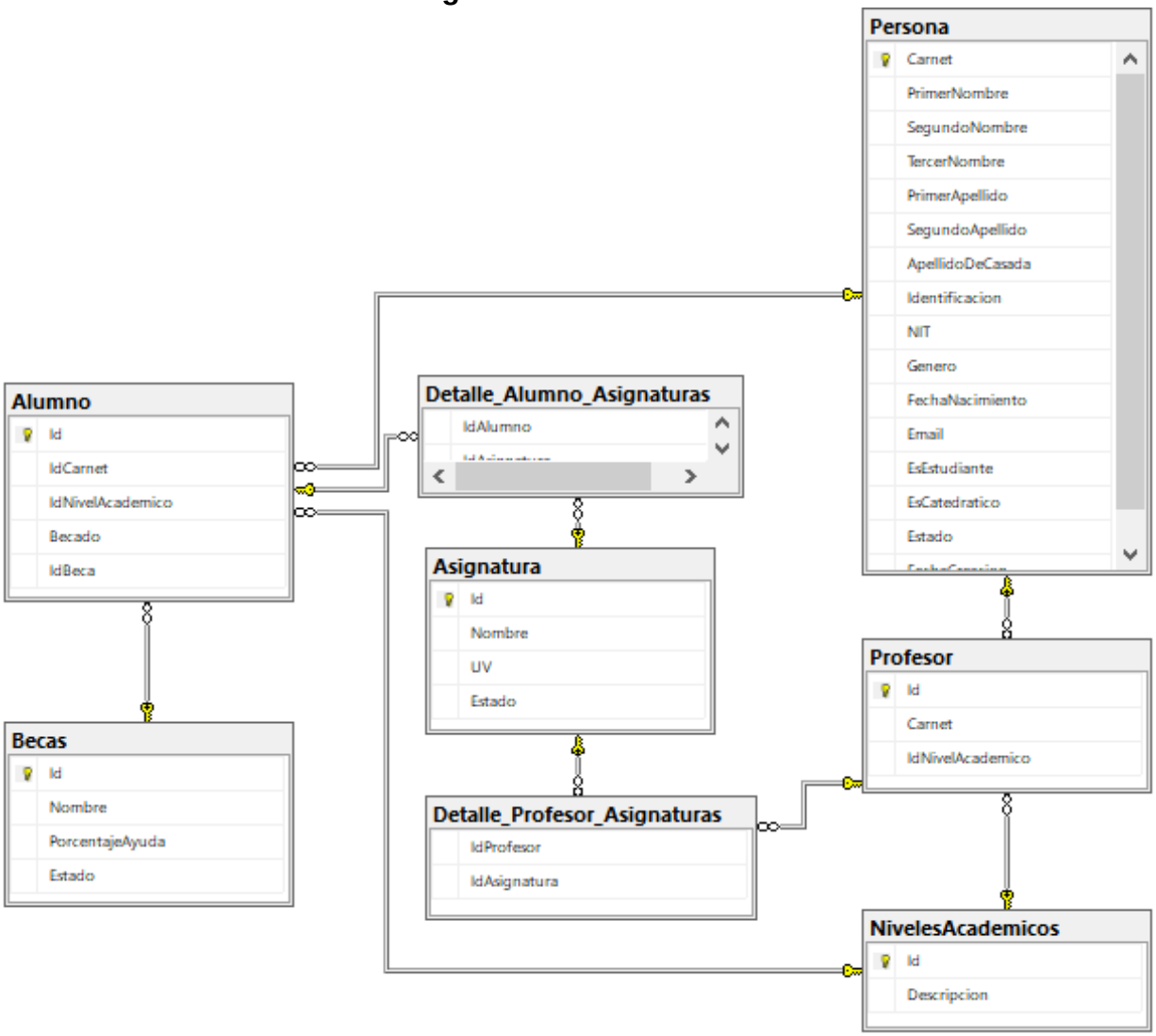

Crear un proyecto(paso 3 de 3)
Configurar Google Analytics
Ubicación de Analytics
El Salvador
Google Analytics es una herramienta empresarial. Úsalo exclusivamente para fines relacionados con tu comercio, negocio, oficio o profesión.
Configuración de uso compartido de datos y Condiciones de Google Analytics


AlumnosBecarios
Tu proyecto nuevo está listo
Continuar



BASE DE DATOS TRANSACCIONAL SQL SERVER

Diagrama de relación:



Querys donde se mira la relación de llaves primarias y foráneas:

```sql
--YA CON REGISTROS PODEMOS VER LA RELACION DE LLAVES PRIMARIAS Y FORANEAS EN UN SELECT
--PARA ALUMNO
SELECT P.Carnet,P.PrimerNombre,P.PrimerApellido,B.Nombre,B.PorcentajeAyuda,
N.Descripcion FROM Persona P INNER JOIN Alumno A ON A.IdCarnet=P.Carnet
INNER JOIN Becas B ON B.Id = A.IdBeca
INNER JOIN NivelesAcademicos N ON N.Id=A.IdNivelAcademico
--PARA PROFESOR
SELECT P.Carnet,P.PrimerNombre,P.PrimerApellido,
N.Descripcion FROM Persona P INNER JOIN Profesor A ON A.Carnet=P.Carnet
INNER JOIN NivelesAcademicos N ON N.Id=A.IdNivelAcademico
```

.00 %

Results | Messages

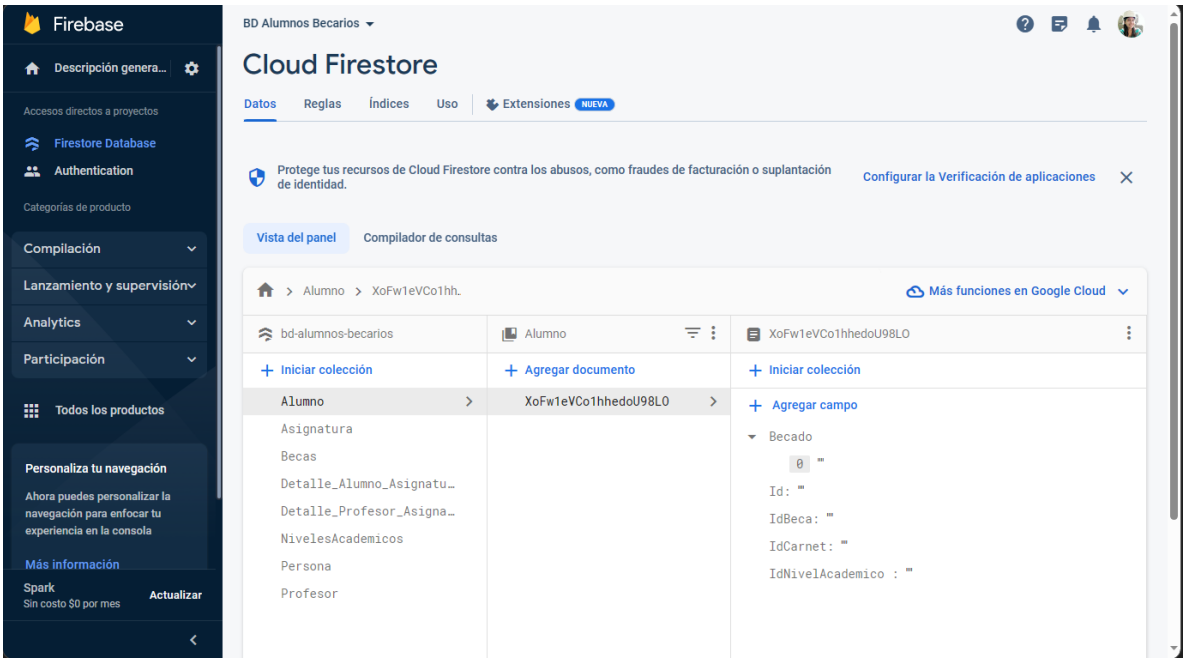
	Carnet	PrimerNombre	PrimerApellido	Nombre	PorcentajeAyuda	Descripcion
1	CV152055	GERARDO	CABEZAS	BECA FUNDACION KASMA	80	Estudiante
2	BJ152032	JOSSELIN	ALVARADO	BECA EMPRESA SELECTOS	80	Estudiante

	Carnet	PrimerNombre	PrimerApellido	Descripcion
1	GP875052	GABRIELA	PINEDA	Ingeniero

BADE DE DATOS FIRESTORES

Se crea BD utilizando Firebase con la opción de Firestores que permite realizar consultas más complejas y rápidas, y el escalamiento se ajusta a un nivel más alto que Realtime Database.

La Base de Datos se llama: bd-alumnos-becarios



Se crearon diferentes colecciones que son lo equivalente a crear una tabla

- Alumno
- Asignatura
- Becas
- Detalle _Alumno_Asignatura
- Detalle_Profesor_Asignatura
- NivelesAcademicos
- Persona
- Profesor

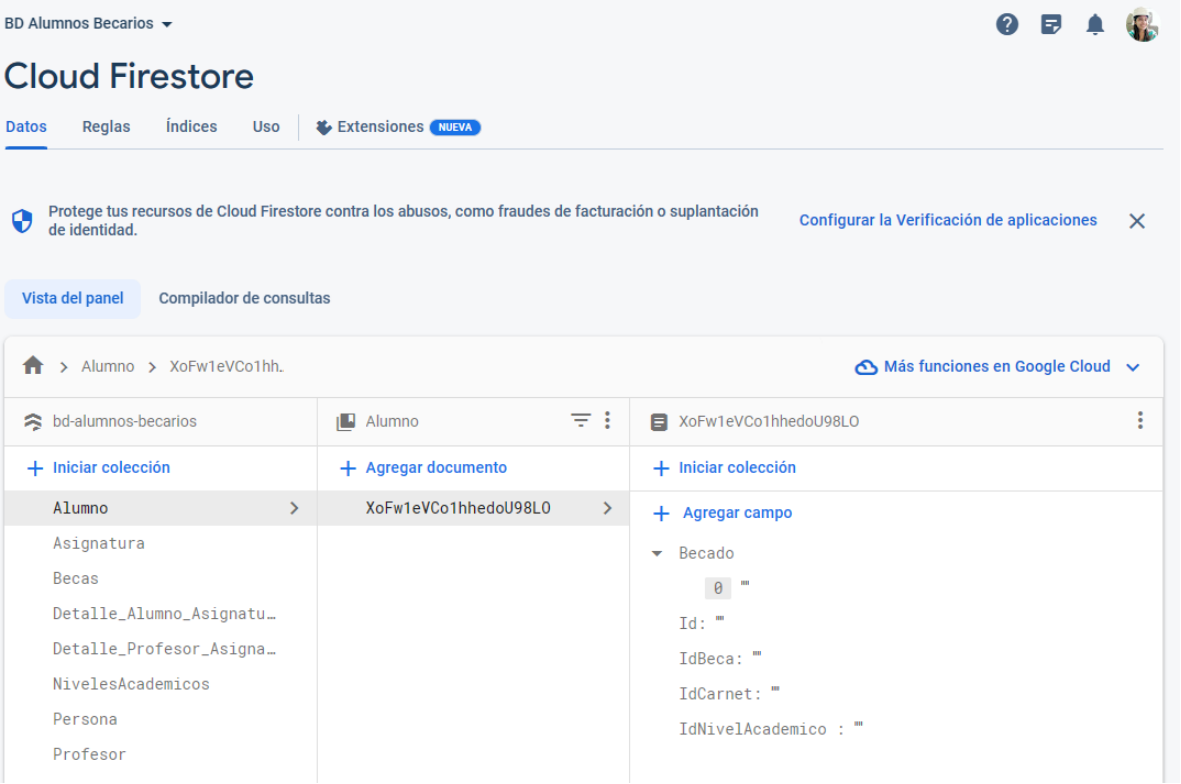
También ser agregaron los campos a cada colección, que se establecen diferente tipo de datos que son similares a los de otros gestores de BD

Tipo de datos	Orden de clasificación	Notas
Arreglo	Según los valores de los elementos	Un arreglo no puede contener otro valor de arreglo como uno de sus elementos. En los arreglos, los elementos conservan la posición que se les asignó. Para ordenar dos o más arreglos, se debe realizar con base en los valores de sus elementos. Cuando se comparan dos arreglos, se comparan los primeros elementos de cada uno de ellos. Si son iguales, se deben comparar los segundos elementos y así sucesivamente, hasta que se encuentre una diferencia. Si un arreglo no tiene más elementos que comparar, pero es igual al otro arreglo hasta ese punto, debe ordenarse antes del arreglo más largo. Por ejemplo, `[1, 2, 3] < [1, 2, 3, 1] < [2]`. El arreglo `[2]` tiene el valor más alto en el primer elemento. El arreglo `[1, 2, 3]` tiene elementos iguales a los primeros tres de `[1, 2, 3, 1]`, pero su longitud es menor.
Booleano	`false` < `true`	-
Bytes	Orden de bytes	Hasta 1,048,487 bytes (de 1 MiB a 89 bytes). Las consultas solo consideran los primeros 1,500 bytes.
Fecha y hora	Cronológico	Cuando se almacena en Cloud Firestore, la precisión máxima es de microsegundos; cualquier precisión adicional se redondea hacia abajo.
Número de coma flotante	Numérico	Precisión doble de 64 bits, IEEE 754.
Punto geográfico	Por latitud, luego por longitud	Por el momento, no recomendamos usar este tipo de datos debido a las limitaciones de consultas. En general, es mejor almacenar la latitud y longitud como campos numéricos separados. Si tu app necesita consultas geográficas simples basadas en la distancia, revisa Consultas geográficas.
Número entero	Numérico	64 bits, firmado

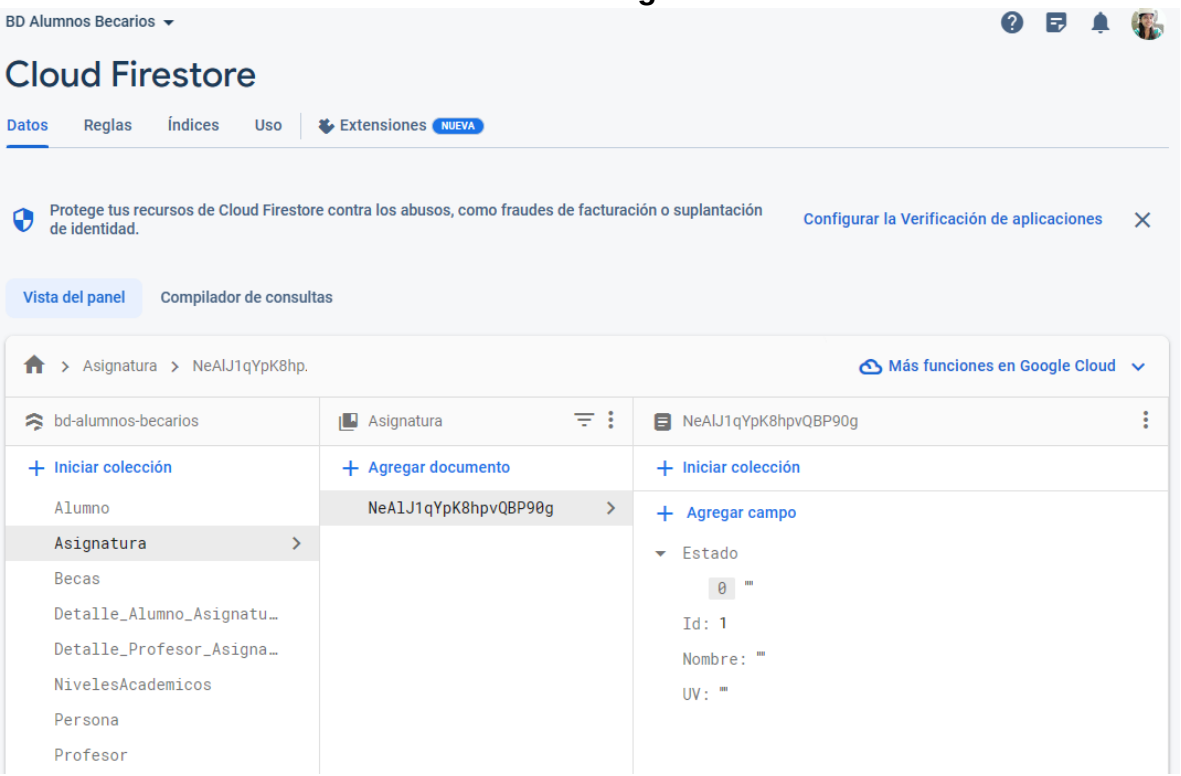
Mapa	Por claves y, luego, por valor	Representa un objeto incorporado en un documento. Cuando se indexa, puedes hacer consultas por subcampos. Si excluyes este valor de la indexación, también se excluyen todos los subcampos.
		
		Siempre se realiza el ordenamiento por clave. Por ejemplo, si escribes `{c: "foo", a: "bar", b: "qux"}`, el mapa se ordena por clave y se guarda como `{a: "bar", b: "qux", c: "foo"}`.
		
		Los campos de mapas se ordenan por clave y se comparan con pares clave-valor. Primero se comparan las claves y, luego, los valores. Si los primeros pares clave-valor son iguales, se compararán los siguientes, y así sucesivamente. Si dos mapas comienzan con los mismos pares clave-valor, se considerará la longitud del mapa. Por ejemplo, estos mapas se ordenan en sentido ascendente:
		
		`{a: "aaa", b: "baz"}`
		`{a: "foo", b: "bar"}`
		`{a: "foo", b: "bar", c: "qux"}`
		`{a: "foo", b: "baz"}`
		`{b: "aaa", c: "baz"}`
		`{c: "aaa"}`

Nulo	Ninguno	—
Referencia	Por elementos de ruta de acceso (colección, ID de documento, colección, ID de documento…)	Por ejemplo, `projects/[PROJECT_ID]/databases/[DATABASE_ID]/documents/[DOCUMENT_PATH]`.

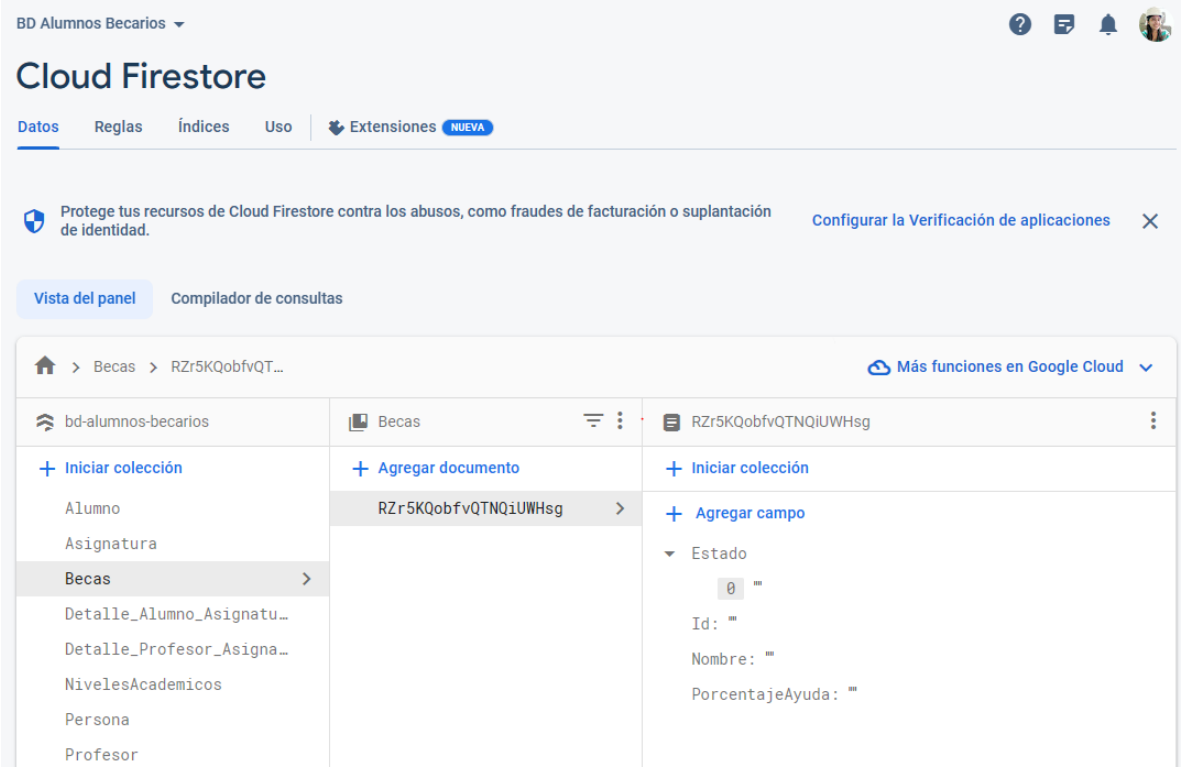
Colección de Alumno



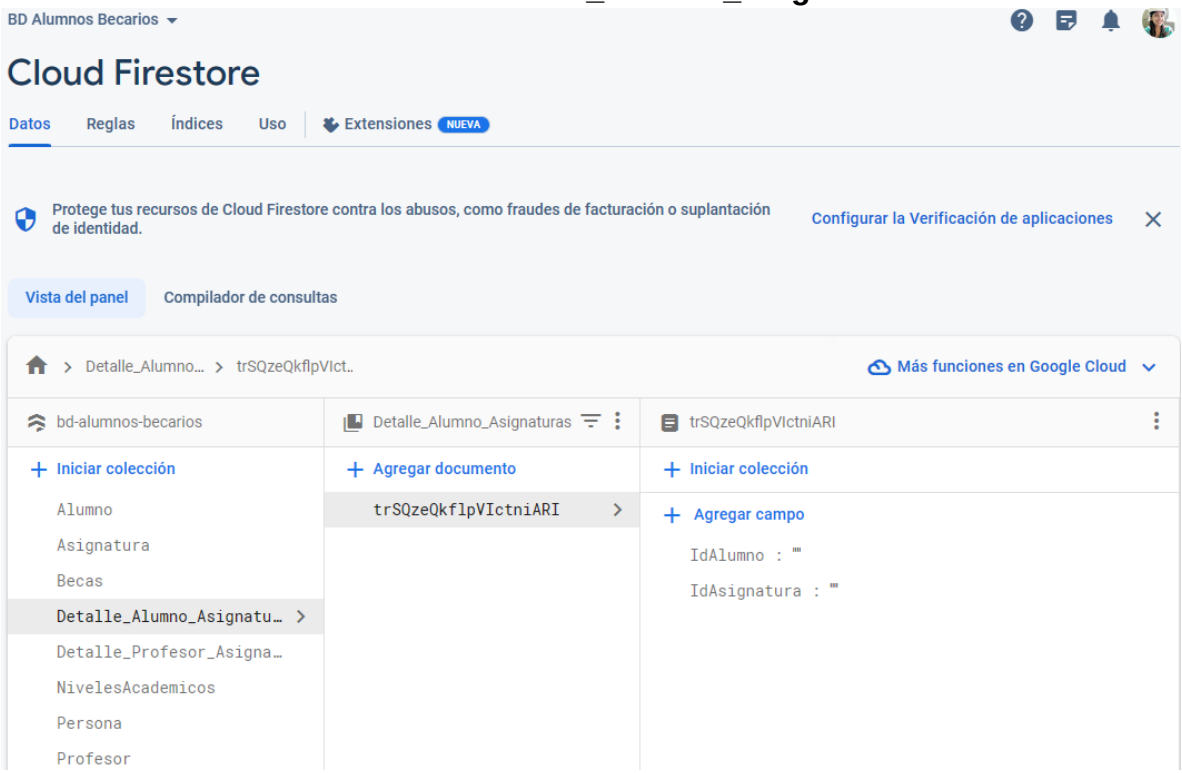
Colección Asignatura



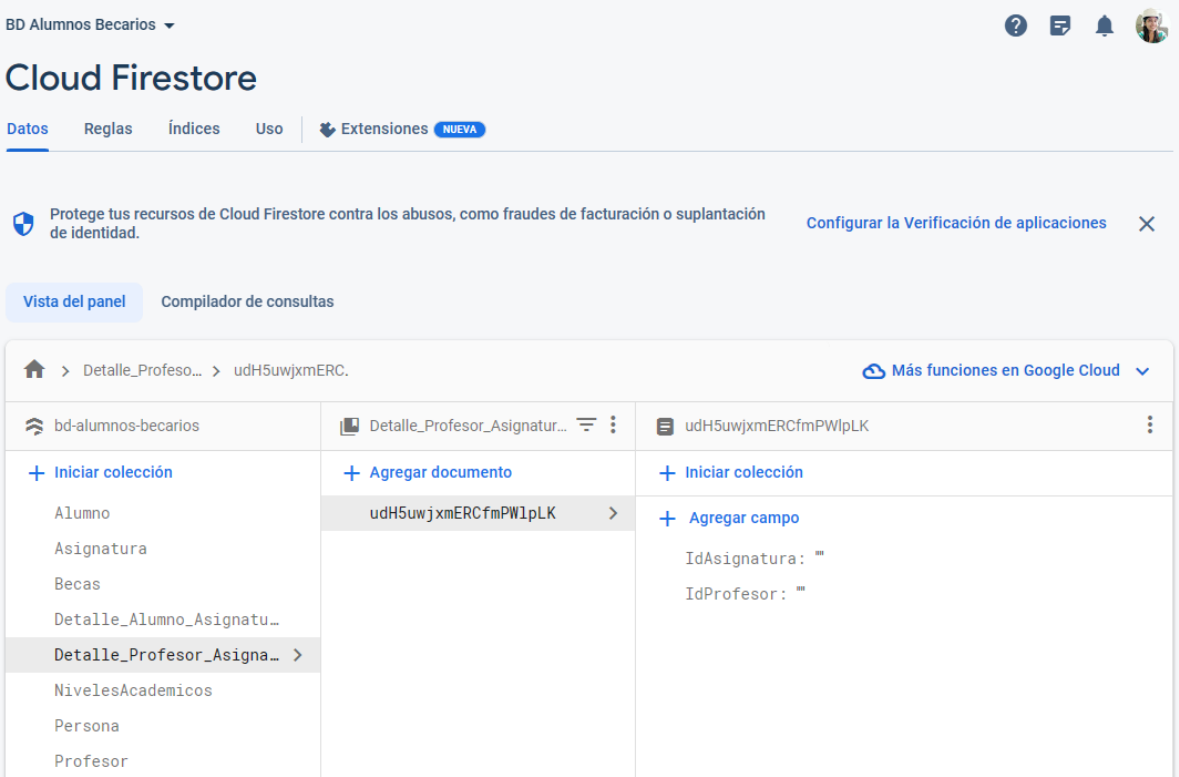
Colección de Becas



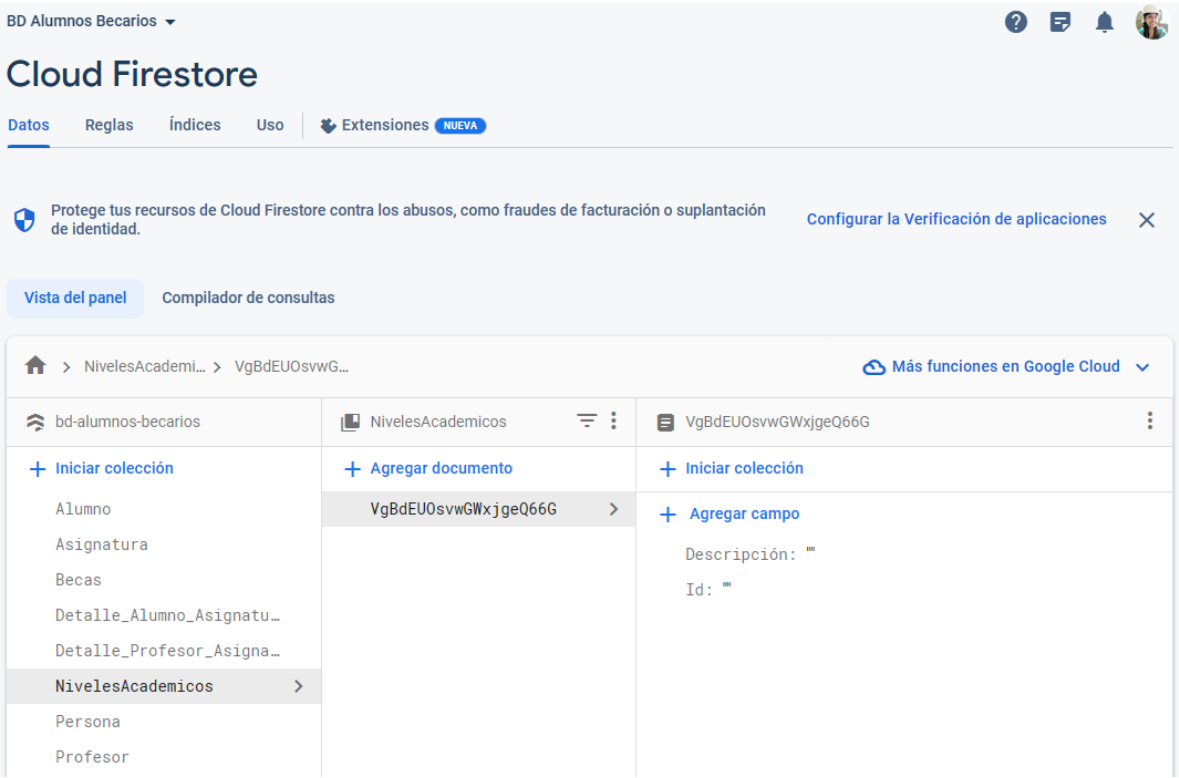
Colección de Detalle_Alumno_Asignatura



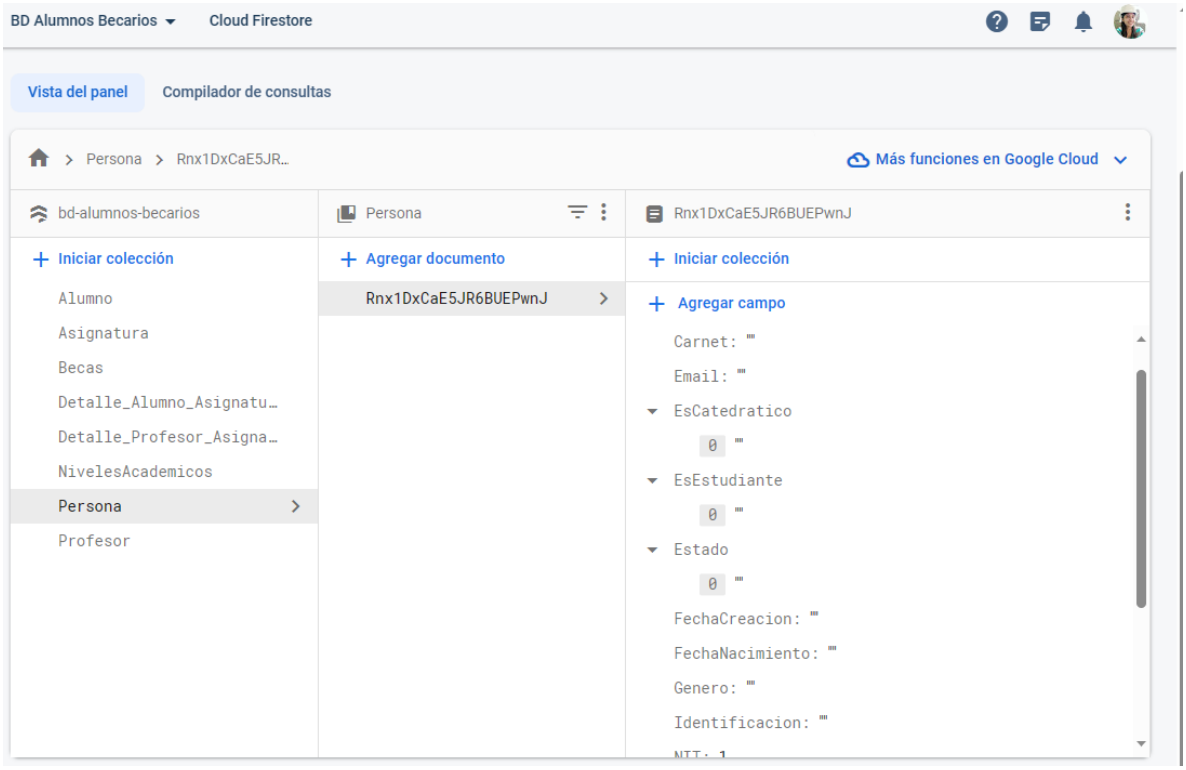
Colección Detalle_Profesor_Asignatura



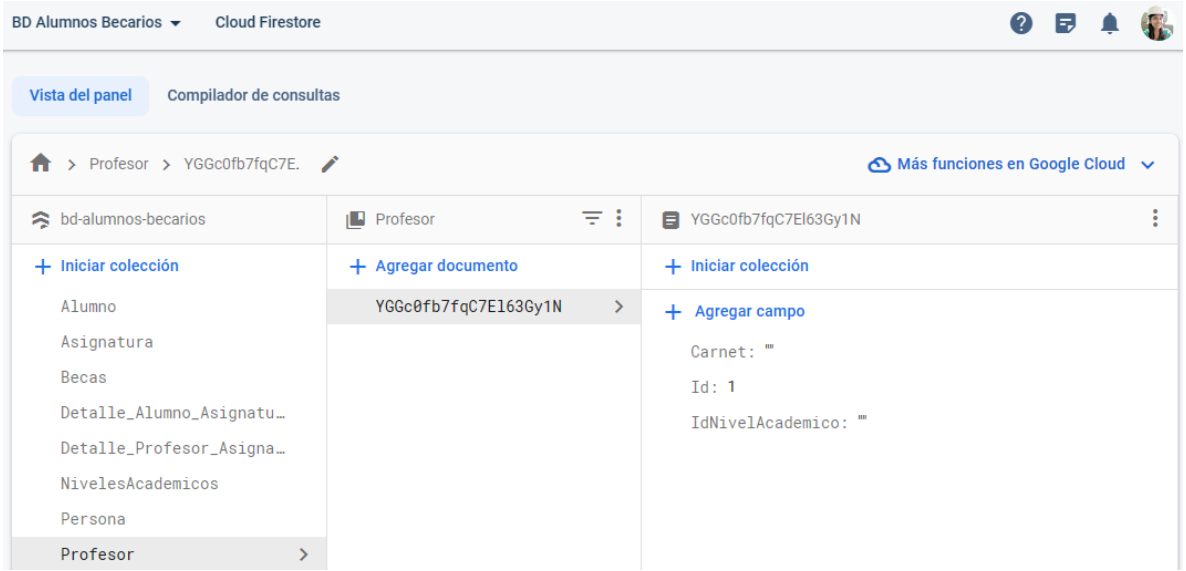
Colección NivelesAcademicos



Colección Persona



Colección Profesor



CONCLUSIONES

Con Cloud Firestore, podemos tener una base de datos donde podemos almacenar estructuras de datos complejas, realizar consultas potentes, anidar consultas, filtrar, ordenar resultados y más; si lo que necesitamos es una base de datos clásica, se recomienda Cloud Firestore.

Por otro lado, si no tenemos estructuras de datos complejas o necesitamos realizar consultas muy potentes, Realtime Database se adaptará muy bien y las diferentes configuraciones de la aplicación cambiarán de acuerdo con estos archivos.

Ambas estructuras son muy buenas, pero se deberá de elegir la adecuada dependiendo del alcance y la necesidades de nuestro proyecto.

BIBLIOGRAFÍA

Mora, S. L. (2022, octubre 4). Firebase: qué es, para qué sirve, funcionalidades y ventajas. DIGITAL55. https://digital55.com/blog/que-es-firebase-funcionalidades-ventajas-conclusiones/

Elige una base de datos: Cloud Firestore o Realtime Database | Firebase. (s. f.). Firebase. https://firebase.google.com/docs/firestore/rtdb-vs-firestore?hl=es-419

KeepCoding, R. (2022, 19 mayo). Qué es Firebase Realtime Database | KeepCoding Bootcamps. KeepCoding Bootcamps. https://keepcoding.io/blog/que-es-firebase-realtime-database/

Información básica. (s. f.). Firebase. https://firebase.google.com/docs/guides?hl=es

Qué es una base de datos NoSQL | Base de datos no relacional de Rackspace. (s. f.). Rackspace Technology. https://www.rackspace.com/es/library/what-is-a-nosql-database

Rendón, Y. A. (s. f.). Bases de datos relacionales vs. no relacionales. https://www.pragma.com.co/academia/lecciones/bases-de-datos-relacionales-vs.-no-relacionales