Universidad Don Bosco.

Campus Virtual

Asignatura:

Diseño y Programación de Software Multiplataforma

Grupo Teórico:

01

Foro 1

Integrantes:

Alas Linares Alejandro Antonio AL192188

De Paz Velásquez Vicente Daniel DV192307

Duran Meléndez Gilberto Emmanuel DM192201

Gutiérrez Borja Rafael Armando GB192205

Docente:

Alexander Alberto Siguenza Campos

Fecha: Domingo 28 de abril del 2024

Contenido

Introducción

En el paisaje actual del desarrollo de aplicaciones móviles y web, la selección precisa de una base de datos puede ser determinante para la eficiencia, escalabilidad y rendimiento de una aplicación.

En este análisis, dirigiremos nuestra atención hacia dos servicios de base de datos ofrecidos por Google Firebase: Cloud Firestore y Realtime Database. Además, examinaremos detalladamente las diferencias entre ambos servicios y compararemos las bases de datos NoSQL con sus contrapartes SQL.

Concluiremos evaluando cuál de estas alternativas sería la más idónea para su implementación en una aplicación desarrollada en React Native.

¿Qué es Cloud Firestore?

Según (Google, 2024) “Cloud Firestore es una base de datos flexible y escalable para el desarrollo en servidores, dispositivos móviles y la Web desde Firebase y Google Cloud.”. Este servicio es una base de datos NoSQL completamente gestionada y altamente escalable.

Las características que definen a Cloud Firestore son las siguientes:

Característica	Descripción
Persistencia en tiempo real	Con la capacidad de almacenar datos estructurados en formato JSON, este servicio va más allá, ofreciendo sincronización instantánea entre los clientes y los servidores. Esto implica que cada modificación realizada en la base de datos se refleja al instante en todos los dispositivos conectados. Aunque también puede realizar consultas de recuperación únicas.
Escalabilidad	Cloud Firestore está diseñado para expandirse de manera autónoma, adaptándose a las necesidades cambiantes de los usuarios. Desde pequeñas aplicaciones hasta gigantescas plataformas empresariales con millones de usuarios
Protección y autorización	En su núcleo, este servicio ofrece robustas políticas de seguridad y autorización, otorgando el poder de controlar quién puede acceder y modificar los datos almacenados en la base de datos.
Integración con otras herramientas de Google Cloud Platform	Cloud Firestore se entrelaza de manera fluida con un conjunto de herramientas y servicios de Google Cloud Platform, como Firebase, Cloud Functions, Google App Engine, y más

Diversidad de plataformas	Cloud Firestore despliega SDKs para una variedad de plataformas, desde Android e iOS hasta JavaScript y Node.js

Desde el cimiento de su modelo de datos NoSQL, Cloud Firestore ofrece una plataforma donde los datos se encapsulan en documentos, donde cada uno de los cuales alberga campos vinculados a valores específicos. Estos documentos, a su vez, se guardan en colecciones, proporcionando una organización estructurada para los datos y un punto de partida para consultas futuras.

Los documentos en Cloud Firestore son capaces de alojar una amplia gama de tipos de datos, desde simples strings y números, hasta complejos objetos anidados. Esta versatilidad se extiende aún más con la capacidad de crear subcolecciones dentro de los documentos, permitiendo la construcción de estructuras de datos jerárquicas que se expanden de manera fluida a medida que la base de datos crece en tamaño y complejidad.

¿Qué es Realtime Database?

Según (Google, 2023) “Firebase Realtime Database es una base de datos alojada en la nube. Los datos se almacenan en formato JSON y se sincronizan en tiempo real con cada cliente conectado”

Este servicio se distingue por su capacidad de reflejar modificaciones en cuestión de milisegundos en todos los dispositivos conectados, eliminando la necesidad de recargar páginas o enviar nuevas solicitudes.

Algunos aspectos cruciales de Realtime Database son:

Característica	Descripción
Sincronización en tiempo real	Los datos en Realtime Database se sincronizan al instante entre los clientes conectados, permitiendo que los usuarios observen cambios en tiempo real sin interrupciones ni esperas.
Almacenamiento en formato JSON	La flexibilidad y facilidad de uso se ven impulsadas por el almacenamiento en formato JSON, facilitando la adaptabilidad a diversas aplicaciones y escenarios.
Escalabilidad	Diseñada para manejar volúmenes masivos de datos y tráfico concurrente, Realtime Database brinda estabilidad y rendimiento incluso ante un crecimiento exponencial de usuarios.
Seguridad y autorización	Las reglas de seguridad personalizables permiten un control detallado sobre quién puede acceder y modificar los datos.
Integración con Firebase	Realtime Database se fusiona perfectamente con las características de Firebase, desde la autenticación hasta las funciones en la nube y el hosting.

Firebase Realtime Database abre las puertas a la creación de aplicaciones dinámicas y colaborativas, al permitir un acceso seguro a la base de datos directamente desde el código del cliente. Los datos persisten localmente, asegurando una experiencia fluida incluso en momentos de desconexión. Aún en ausencia de conexión, los eventos en tiempo real continúan activos, garantizando una experiencia adaptable para el usuario final. Al reconectarse el dispositivo, Realtime Database sincroniza automáticamente los cambios locales con las actualizaciones remotas, gestionando conflictos de manera automática.

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

Cloud Firestore y Realtime Database son dos pilares de almacenamiento de datos proporcionados por Google Firebase. Aunque ambos comparten el objetivo fundamental de almacenar y sincronizar datos en tiempo real, divergen en varios aspectos clave:

	Realtime Database	Cloud Firestore
Modelo de datos	Adopta una estructura de árbol JSON, que, si bien es flexible, puede dificultar la escalabilidad en aplicaciones con datos anidados de gran volumen.	Se basa en una estructura de documentos y colecciones. Cada documento sigue un esquema clave-valor similar a JSON, ofreciendo una organización más rica y jerárquica de los datos.
Escalabilidad	Destacado por su capacidad para manejar cargas de lectura y escritura intensivas, pero puede enfrentar desafíos en aplicaciones con datos complejos o consultas elaboradas.	Ofrece una escalabilidad mejorada, especialmente para aplicaciones con consultas sofisticadas y grandes volúmenes de datos anidados.
Consultas y ordenación	Aunque permite consultas básicas y ordenación según una sola clave, las consultas complejas pueden ser limitadas y requieren manipulación adicional en el cliente.	Se destaca por su soporte para consultas flexibles y avanzadas, incluyendo consultas compuestas, ordenación y filtrado directamente en el servidor, lo que alivia la carga en el cliente.
Precio	Opera bajo un esquema de precios basado en el ancho de banda y el almacenamiento, siendo las lecturas y escrituras relativamente económicas. Sin embargo, los costos pueden aumentar con una gran cantidad de datos transferidos.	También sigue un modelo de precios basado en el ancho de banda y el almacenamiento, aunque generalmente es un poco más costoso que Realtime Database. A pesar de ello, su mejor escalabilidad y funcionalidad pueden justificar

		el costo adicional para ciertas aplicaciones.

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

En el campo del desarrollo y manejo de aplicaciones, es crucial escoger la base de datos correcta según las demandas del proyecto. Hay dos grandes grupos de bases de datos: SQL (Structured Query Language) y NoSQL (Not Only SQL), cada una con atributos distintivos.

Las bases de datos SQL o relacionales emplean un modelo de datos estructurado con esquemas predeterminados, ideal para mantener la consistencia de los datos. Este tipo es extensamente utilizado en aplicaciones que implican transacciones complejas, como los sistemas de gestión financiera, donde los datos se organizan en tablas interrelacionadas para preservar la integridad de los datos mediante claves foráneas y otras restricciones.

En contraste, las bases de datos NoSQL ofrecen una mayor flexibilidad con respecto a los esquemas, lo cual no es necesario fijar, permitiendo una integración más sencilla de diversos tipos y formatos de datos. Esto es beneficioso en aplicaciones que requieren una rápida escalabilidad o gestionan grandes volúmenes de datos no estructurados, como las redes sociales o las plataformas de análisis de big data. Las bases de datos NoSQL incluyen tipos como documentales, de clave-valor, de grafos o de columnas, optimizadas para distintos tipos de consultas y operaciones.

Respecto a la escalabilidad, las bases de datos SQL tienden a escalar verticalmente, añadiendo más potencia de procesamiento a un servidor único, lo que puede resultar costoso y con limitaciones físicas. Por otro lado, las bases de datos NoSQL están diseñadas para escalar horizontalmente, distribuyendo la carga entre varios servidores, lo que mejora el manejo de grandes datos y aumenta la disponibilidad y la tolerancia a fallos, siendo ideales para entornos de nube donde la capacidad de expansión rápida es fundamental.

En el aspecto de transacciones, las bases de datos SQL admiten transacciones ACID (Atomicity, Consistency, Isolation, Durability), garantizando que todas las operaciones

de una transacción se ejecuten con éxito o no se realicen en caso de fallo, esencial para aplicaciones críticas como el comercio electrónico o los servicios bancarios. Las bases de datos NoSQL, por su parte, suelen ofrecer transacciones BASE (Basically Available, Soft state, Eventual consistency), que fomentan la disponibilidad y la tolerancia a fallos pero pueden sacrificar la consistencia inmediata, adecuado en aplicaciones donde la velocidad de respuesta y el volumen de datos son prioritarios sobre la precisión exacta de los datos en tiempo real.

Finalmente, la decisión entre SQL y NoSQL dependerá de los requisitos específicos del proyecto, incluyendo el tipo de datos, la necesidad de escalabilidad, la integridad de las transacciones y las consultas requeridas.

¿Cuál de estas bases de datos consideran que sería la mejor opción para implementar en una aplicación desarrollada en React Native?.

En el campo del desarrollo y manejo de aplicaciones, es crucial escoger la base de datos correcta según las demandas del proyecto. Hay dos grandes grupos de bases de datos: SQL (Structured Query Language) y NoSQL (Not Only SQL), cada una con atributos distintivos.

Las bases de datos SQL o relacionales emplean un modelo de datos estructurado con esquemas predeterminados, ideal para mantener la consistencia de los datos. Este tipo es extensamente utilizado en aplicaciones que implican transacciones complejas, como los sistemas de gestión financiera, donde los datos se organizan en tablas interrelacionadas para preservar la integridad de los datos mediante claves foráneas y otras restricciones.

En contraste, las bases de datos NoSQL ofrecen una mayor flexibilidad con respecto a los esquemas, lo cual no es necesario fijar, permitiendo una integración más sencilla de diversos tipos y formatos de datos. Esto es beneficioso en aplicaciones que requieren una rápida escalabilidad o gestionan grandes volúmenes de datos no estructurados, como las redes sociales o las plataformas de análisis de big data. Las bases de datos NoSQL incluyen tipos como documentales, de clave-valor, de grafos o de columnas, optimizadas para distintos tipos de consultas y operaciones.

Respecto a la escalabilidad, las bases de datos SQL tienden a escalar verticalmente, añadiendo más potencia de procesamiento a un servidor único, lo que puede resultar costoso y con limitaciones físicas. Por otro lado, las bases de datos NoSQL están diseñadas para escalar horizontalmente, distribuyendo la carga entre varios servidores, lo que mejora el manejo de grandes datos y aumenta la disponibilidad y la tolerancia a fallos, siendo ideales para entornos de nube donde la capacidad de expansión rápida es fundamental.

En el aspecto de transacciones, las bases de datos SQL admiten transacciones ACID (Atomicity, Consistency, Isolation, Durability), garantizando que todas las operaciones de una transacción se ejecuten con éxito o no se realicen en caso de fallo, esencial para aplicaciones críticas como el comercio electrónico o los servicios bancarios. Las bases de datos NoSQL, por su parte, suelen ofrecer transacciones BASE (Basically Available, Soft state, Eventual consistency), que fomentan la disponibilidad y la tolerancia a fallos pero pueden sacrificar la consistencia inmediata, adecuado en aplicaciones donde la velocidad de respuesta y el volumen de datos son prioritarios sobre la precisión exacta de los datos en tiempo real.

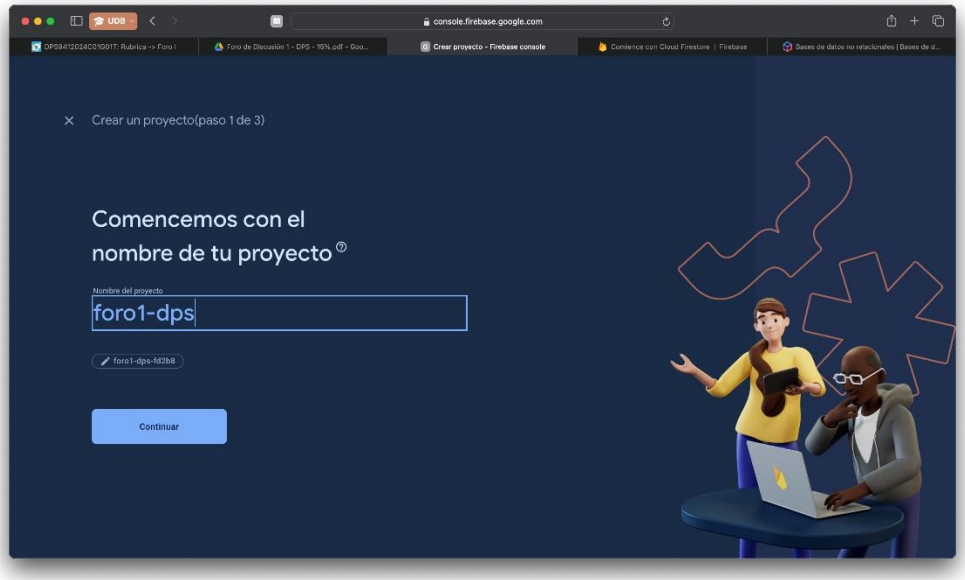
Finalmente, la decisión entre SQL y NoSQL dependerá de los requisitos específicos del proyecto, incluyendo el tipo de datos, la necesidad de escalabilidad, la integridad de las transacciones y las consultas requeridas.

Fase 2: Implementación de Base de Datos NoSQL usando Firebase Firestone

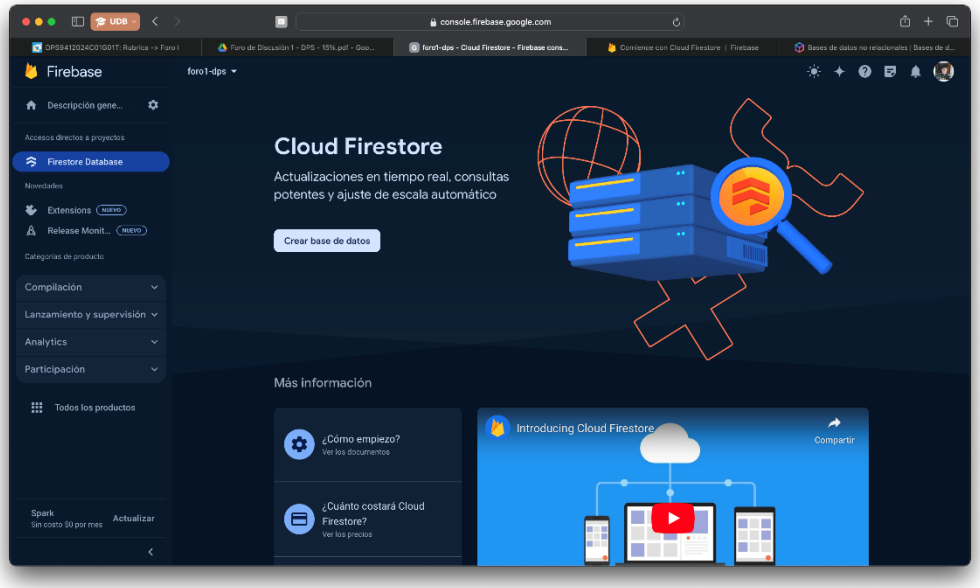
Para la implementación de una base de datos NoSQL usando Firebase es necesario tener una cuenta de Google e iniciar sesión en: https://console.firebase.google.com/u/0/?hl=es.

Al haber iniciado sesión se nos presentará una interfaz similar a la siguiente:

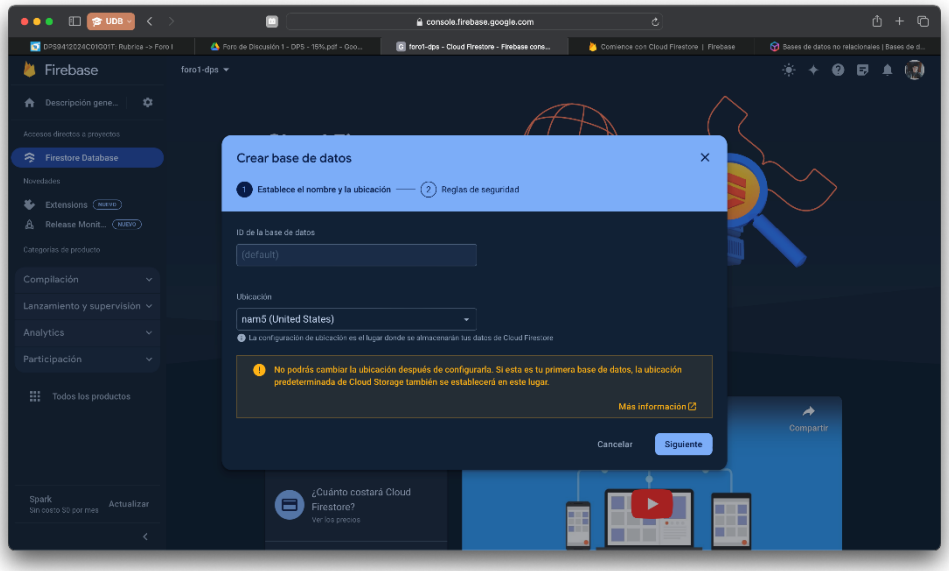
Damos clic en crear proyecto y nos pedirá crear un nombre para nuestro nuevo proyecto, en esta ocasión colocamos “foro1-dps”.



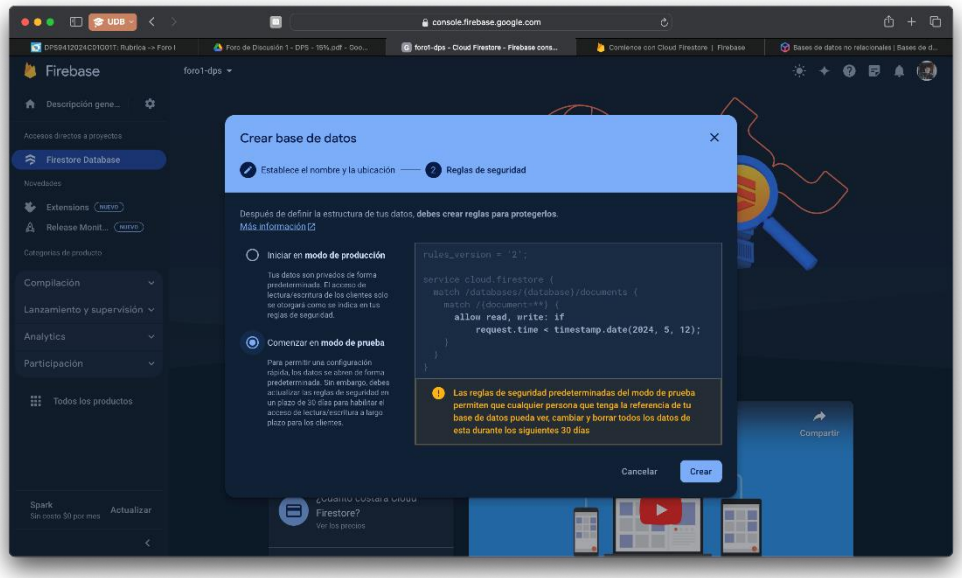
Posteriormente se nos creará un proyecto en Firebase, en la barra lateral izquierda debemos buscar en el directorio “Compilación” la opción llamada Cloud Firestone. Al seleccionarla nos mostrará una interfaz similar a la siguiente:



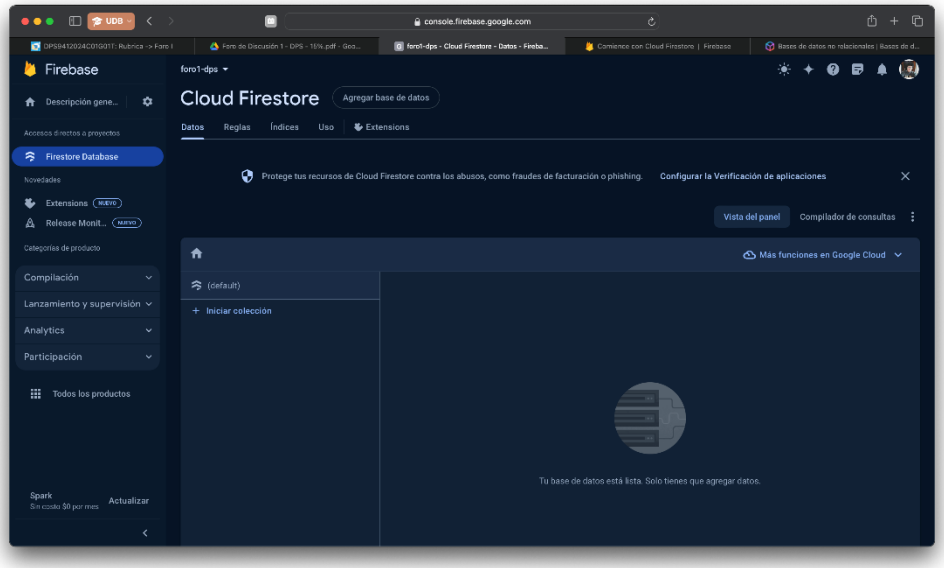
Al dar clic en “Crear base de datos” se nos mostrará una ventana emergente en la que dejaremos todo como está:



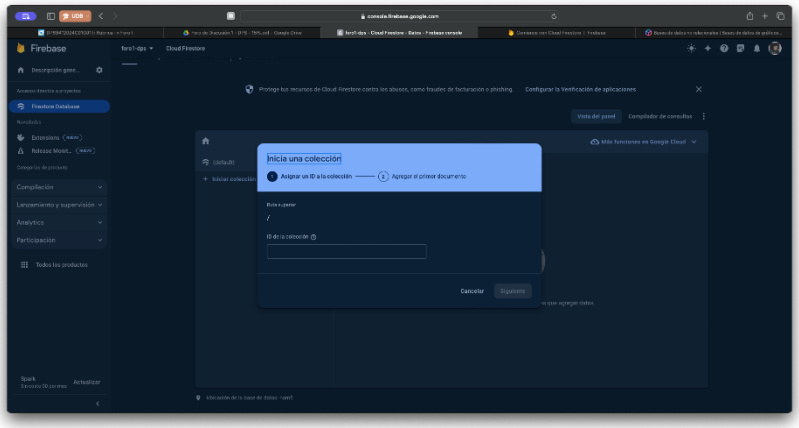
Al dar clic en “Siguiente” se nos preguntará acerca de las reglas de seguridad, para este ejemplo dejaremos seleccionada la opción “Comenzar en modo de prueba”:



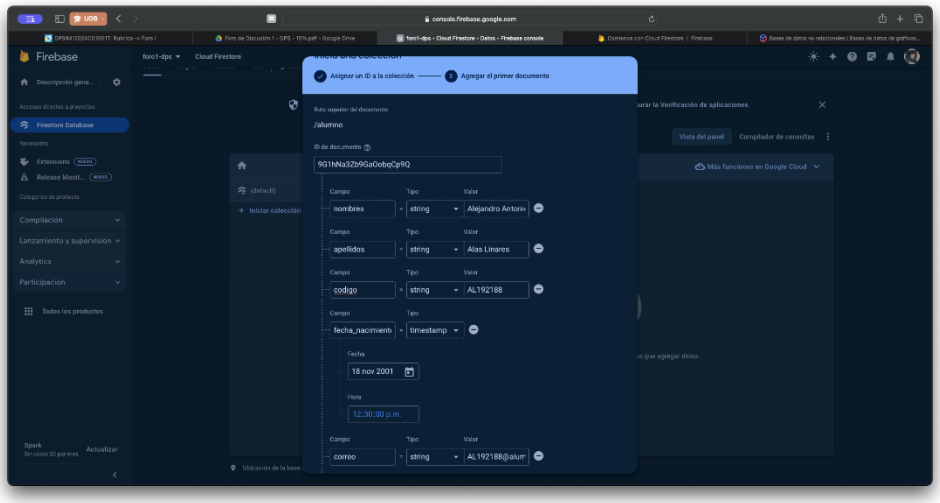
Al haberse creado nuestra base de datos se nos mostrará la siguiente interfaz:



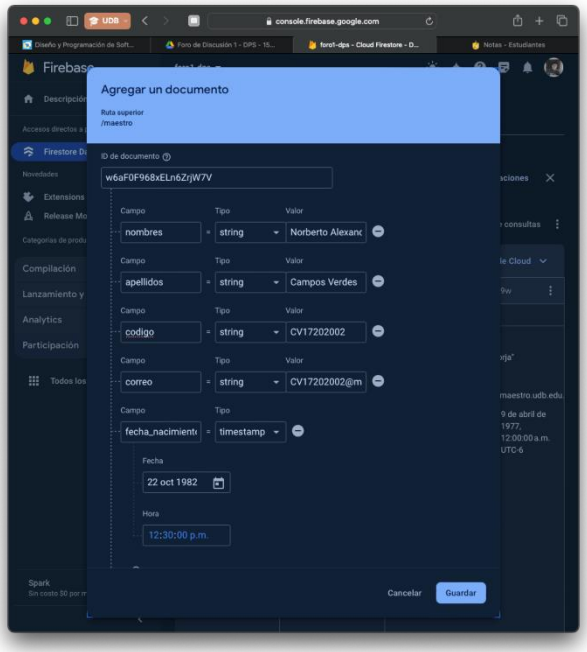
Haremos clic en “Iniciar colección” y se nos presentará un menú donde nos pedirá crear un nombre para la colección, en este caso colocaremos de nombre “alumno”.



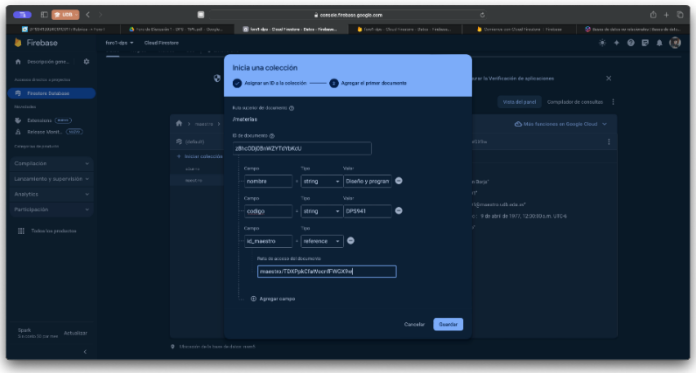
Al dar a “Siguiente” nos pedirá crear un nuevo documento para añadirse a la colección. Este documento sirve como un registro dentro de la colección (las colecciones actuan como tablas). Aquí creamos la estructura de nuestra tabla.



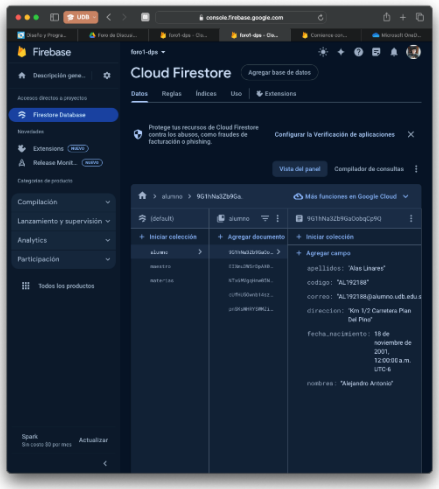
Repetiremos el mismo proceso de dar a “Iniciar colección”, en esta ocación crearemos la colección "maestro" y tendrá la siguiente estructura:

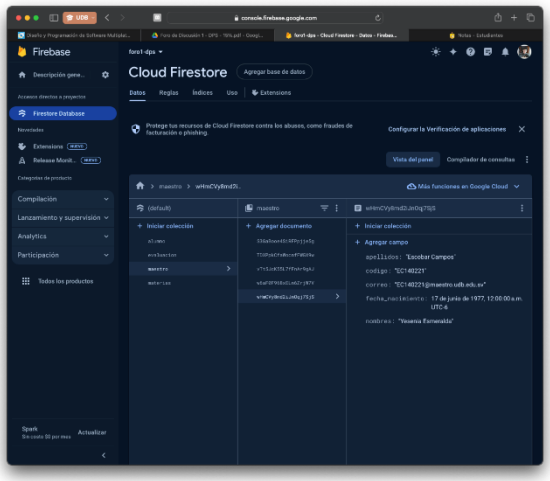


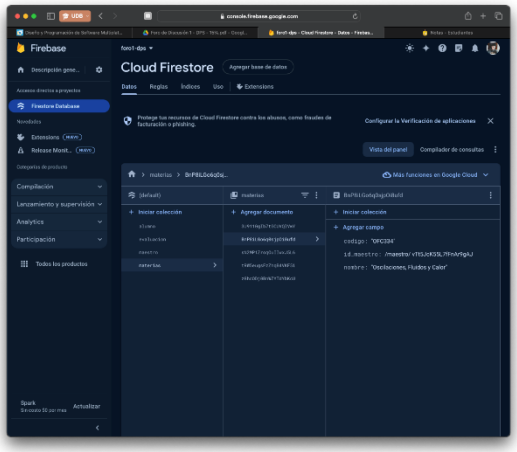
Repetiremos este proceso con las colecciones “materias” y “evaluaciones”, con la diferencia de que llevarán un tipo de dato llamado referencia. Este tipo de dato permite hacer referencia a un documento de otra colección y generar relaciones entre los diversos registros de la base de datos.

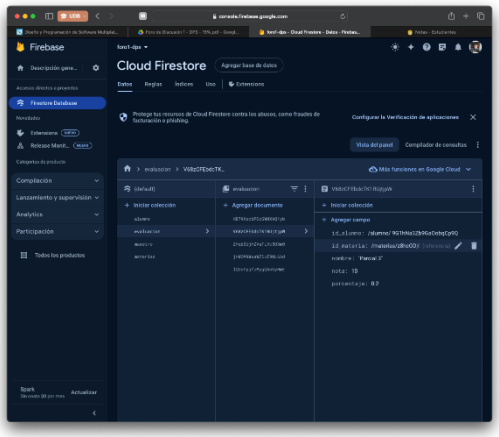


Una vez teniendo esta estructura, crearemos registros en todas las colecciones y eso nos permitirá tener un primer modelo de nuestra base de datos.






Firebase
Cloud Firestore
Agregar base de datos
Datos
Reglas
Índices
Uso
Extensiones
Vista del panel
Compilador de consultas
Compilación
Lanzamiento y supervisión
Analytics
Participación
Todos los productos
Spark
Actualizar


Firebase
Cloud Firestore
Agregar base de datos
Datos
Reglas
Índices
Uso
Extensiones
Vista del panel
Compilador de consultas
Compilación
Lanzamiento y supervisión
Analytics
Participación
Todos los productos
Spark
Actualizar

Conclusiones de investigación

Al desarrollar aplicaciones, especialmente proyectos que utilizan React Native, es necesario elegir la tecnología de base de datos adecuada debido a la variedad de funciones disponibles. Por ejemplo, Firebase Realtime Database es una aplicación ideal que requiere datos en tiempo real y datos en tiempo real. Además, la estructura de costos depende del ancho de banda y el uso de almacenamiento. En cambio, Cloud Firestore es más adecuado para administrar bases de datos complejas y grandes, admite consultas complejas y tiene una mejor escalabilidad, lo que puede generar costos más altos.

La elección entre bases de datos SQL y NoSQL también es importante para la arquitectura de aplicaciones. Las bases de datos SQL son conocidas por su poder en la gestión empresarial y la integridad de los datos, lo que las hace perfectas para aplicaciones que requieren una alta coherencia, como las que se utilizan en el mundo financiero. Por otro lado, las bases de datos NoSQL aportan flexibilidad y escalabilidad horizontal, ideales para gestionar grandes cantidades de datos no estructurados, algo habitual en análisis de big data o redes sociales.

Para proyectos que utilizan React Native, la distinción entre SQL y NoSQL debería depender de las necesidades específicas del proyecto. NoSQL Database, Firebase Realtime Database y Cloud Firestore son útiles para proyectos que buscan escalabilidad, desarrollo rápido y flexibilidad en la gestión de datos. Estas funciones son especialmente útiles para aplicaciones web y móviles dinámicas donde la escalabilidad y la comparación de datos en tiempo real son esenciales.

Por último, los factores de coste y rendimiento son importantes a la hora de elegir una acción. Estimar los costos operativos en función de la cantidad de datos y el tipo de operación (lectura, escritura, almacenamiento) es clave. Cada tecnología de base de datos tiene características que deben cumplir requisitos específicos del proyecto, incluidos los tipos de datos, las necesidades de escalabilidad, la integridad de las transacciones y las características de las solicitudes.

Conclusiones de implementación

La implementación nos logra confirmar que usar bases de datos SQL o NoSQL depende plenamente del enfoque y del tipo de arquitectura que se desee utilizar, y sobre todo en requerimientos de la aplicación.

En la implementación que realizamos el usar una base de datos SQL ayuda a mantener un esquema predefinido que funcionará en todas partes de la aplicación. Esto porque el escenario de ingreso de notas en una universidad es predecible y la información que utilizará el sistema cumplirá con el formato del esquema. Esto no quiere decir que usar NoSQL no sea conveniente para este escenario, pues siempre se pueden crear estructuras complejas bien definidas haciendo uso de estructuración y referencias a colecciones externas, pero no estaríamos aprovechando las ventajas de Firebase como su flexibilidad, además que utilizar Firebase o Realtime Database en este tipo de aplicaciones resulta impráctico, pues una de las grandes ventajas de este tipo de bases de datos es su velocidad de escritura, pero en aplicaciones de ingreso de notas la cantidad de escritura no es grande durante todo el tiempo, solo durante tiempo de evaluaciones, por lo que las ventajas de estos servicios no se verían utilizadas. Además, por lo general las universidades y/o centros educativos tienden a mantener sus datos almacenados en data centers locales y únicos, esto para reducir logísticas, por lo que las ventas de SQL se vuelven obvias, ya que estos sistemas escalan mejor de forma vertical.

Bibliografía

Google. (13 de Septiembre de 2023). *Firebase Realtime Database*. Obtenido de firebase.google.com: https://firebase.google.com/docs/database?hl=es-419

Google. (19 de Enero de 2024). *Cloud Firestore*. Obtenido de firebase.google.com: https://firebase.google.com/docs/firestore?hl=es-419

Connolly, T., & Begg, C. (2014). Database Systems: A Practical Approach to Design, Implementation, and Management. Pearson.

Sadalage, P. J., & Fowler, M. (2012). NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence. Addison-Wesley.

Silberschatz, A., Korth, H. F., & Sudarshan, S. (2010). Database System Concepts (6th ed.). McGraw-Hill Education.