UNIVERSIDAD DON BOSCO

DISEÑO Y PROGRAMACION DE SOFTWARE MULTIPLATAFORMA

FORO I
Categoría GRUPAL

GRUPO 01T

INTEGRANTES:

Apellidos	Nombres	Carnet
Cardoza de León	Sergio Alexander	CD120928

DOCENTE:
Ing. Alexander Sigüenza

FECHA DE PRESENTACIÓN:
Sábado , 28 de Octubre de 2023

Índice

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

Diferencias	SQL	NoSQL
Modelo de Datos	Son bases de datos relacionales que utilizan tablas para organizar los datos. Los datos se almacenan en filas y columnas, y se establecen relaciones entre las tablas.	Son bases de datos no relacionales que pueden utilizar varios modelos de datos, como documentos, gráficos, clave-valor o columnas. No tienen una estructura fija de tabla como las bases de datos SQL.
Escalabilidad	Suelen escalar verticalmente, lo que significa que se pueden ampliar agregando más recursos (CPU, RAM) a un servidor. Esto puede ser costoso y tiene un límite práctico.	Suelen escalar horizontalmente, lo que permite agregar más servidores a la infraestructura para manejar cargas de trabajo crecientes de manera más eficiente.
Esquema	Utilizan un esquema rígido y predefinido, lo que significa que se debe definir una estructura de tabla antes de ingresar datos. Cualquier desviación del esquema puede ser problemática.	Son esquema flexible, lo que permite agregar campos o cambiar la estructura de datos sin afectar la aplicación. Esto es útil en situaciones donde los requisitos de datos son cambiantes.
Transacciones ACID	Son conocidas por cumplir con las propiedades ACID (Atomicidad, Consistencia, Aislamiento, Durabilidad), lo que garantiza la integridad de los datos en transacciones críticas.	Pueden ofrecer propiedades ACID, pero no todas lo hacen. Algunas bases de datos NoSQL priorizan la escalabilidad y pueden sacrificar la consistencia en ciertos escenarios.
Consultas	Son altamente eficientes en consultas complejas y ad-hoc utilizando SQL, lo que las hace ideales para aplicaciones que requieren consultas complejas.	Pueden ser menos eficientes en consultas complejas y suelen ser más adecuadas para aplicaciones con patrones de acceso de datos específicos.
Uso de Casos	Son comunes en aplicaciones donde la estructura de datos es fija y se requiere un alto nivel de integridad de los datos, como sistemas de contabilidad, sistemas de gestión de inventario y aplicaciones empresariales.	Son adecuadas para aplicaciones web, móviles y proyectos que tienen requisitos de escalabilidad, flexibilidad y velocidad, como redes sociales, análisis de big data y aplicaciones en tiempo real.

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

Diferencias	Cloud Firestore	Realtime Database
Modelo de Datos	Utiliza un modelo de base de datos NoSQL basado en documentos. Los datos se almacenan en documentos, que pueden agruparse en colecciones. Esto permite una estructura jerárquica de datos y es escalable y flexible.	Base de datos en tiempo real basado en árboles de JSON. Los datos se almacenan en nodos y se pueden sincronizar en tiempo real. Esto lo hace adecuado para aplicaciones que requieren actualizaciones en tiempo real.
Consultas	Permite consultas más ricas y flexibles, incluyendo filtrado, ordenación y límites, lo que facilita la obtención de datos específicos.	Ofrece consultas más simples basadas en la ubicación de nodos y no permite filtrado y ordenación tan complejos como Cloud Firestore.
Escalabilidad	Ofrece una mejor escalabilidad horizontal para manejar grandes volúmenes de datos y tráfico. Es adecuado para aplicaciones con una gran cantidad de lecturas y escrituras simultáneas.	Es menos escalable que Cloud Firestore y puede enfrentar problemas de rendimiento en aplicaciones con una alta concurrencia.
Seguridad	Ofrece un sistema de reglas de seguridad más granular y flexible, que permite controlar el acceso a nivel de colección y documento. Puedes definir reglas personalizadas para cada colección.	Las reglas de seguridad son más simples en comparación con Cloud Firestore y se aplican a nivel de nodo. Esto puede hacer que sea más difícil implementar ciertas lógicas de seguridad.
Precio	Tiene una estructura de precios basada en la cantidad de operaciones de lectura, escritura y ancho de banda utilizadas. Puede ser más costoso en aplicaciones con un alto volumen de operaciones.	Utiliza una estructura de precios similar, pero suele ser más económico que Cloud Firestore en aplicaciones con muchas lecturas y escrituras en tiempo real.

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

Eso dependerá de las exigencias que se vaya a desarrollar en la aplicación; si se requiere una estructura de datos altamente flexible, consultas ricas y complejas, y una escalabilidad superior, Cloud Firestore es la mejor elección. Si la actualización sera en tiempo real y la simplicidad en el modelo de datos son más importantes en la aplicación, Firebase Realtime Database es la opción adecuada.

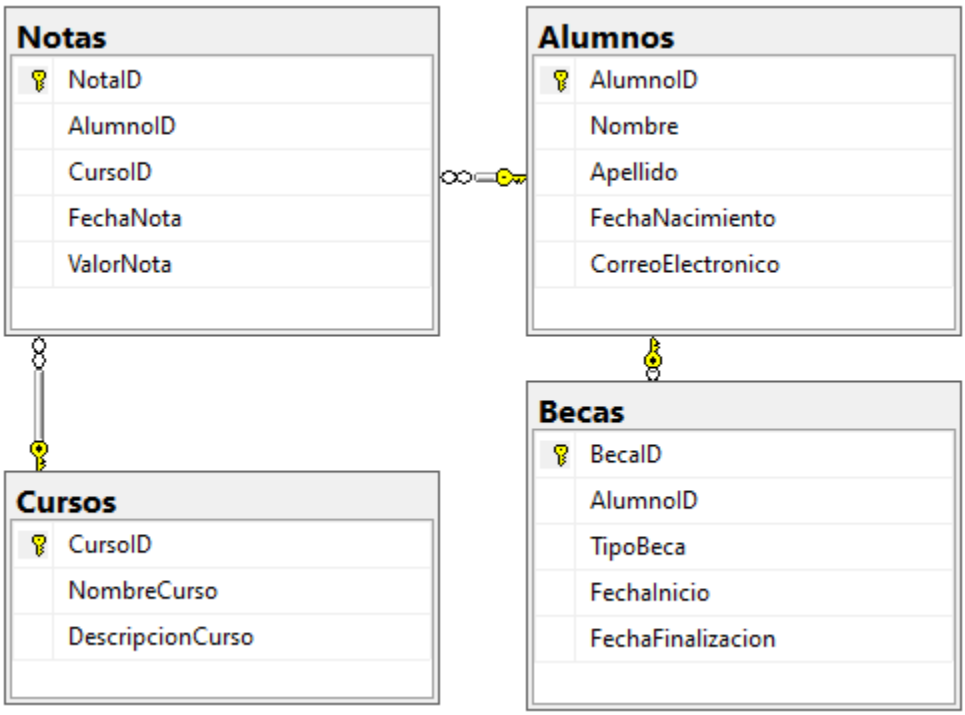
En esta fase, deben elaborar dos estructuras de base de datos: una utilizando SQL y otra utilizando NoSQL (Firebase Firestore o Realtime Database). Ambas bases de datos deben permitir almacenar las notas de los alumnos becarios de UDB VIRTUAL.

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

Script de BD con SQL Server

```
CREATE DATABASE NotasUDB;

-- Crear la tabla Alumnos

CREATE TABLE Alumnos (
    AlumnoID INT PRIMARY KEY,
    Nombre VARCHAR(50),
    Apellido VARCHAR(50),
    FechaNacimiento DATE,
    CorreoElectronico VARCHAR(100),
    -- Otros campos de información personal
);

-- Crear la tabla Cursos
CREATE TABLE Cursos (
    CursoID INT PRIMARY KEY,
    NombreCurso VARCHAR(100),
    DescripcionCurso TEXT,
    -- Otros campos relacionados con el curso
);

-- Crear la tabla Notas
CREATE TABLE Notas (
    NotaID INT PRIMARY KEY,
    AlumnoID INT,
    CursoID INT,
    FechaNota DATE,
    ValorNota DECIMAL(5, 2),

-- Relacion de tablas
```

```sql
    FOREIGN KEY (AlumnoID) REFERENCES Alumnos(AlumnoID),
    FOREIGN KEY (CursoID) REFERENCES Cursos(CursoID)
);

-- Crear la tabla Becas
CREATE TABLE Becas (
    BecaID INT PRIMARY KEY,
    AlumnoID INT,
    TipoBeca VARCHAR(50),
    FechaInicio DATE,
    FechaFinalizacion DATE,
    -- Otros detalles de la beca
    FOREIGN KEY (AlumnoID) REFERENCES Alumnos(AlumnoID)
);


use NotasUDB
-- Insertar datos en la tabla Alumnos

INSERT INTO Alumnos (AlumnoID, Nombre, Apellido, FechaNacimiento, CorreoElectronico)
VALUES
    (1, 'Juan', 'Pérez', '2023-03-15', 'juanperez@example.com'),
    (2, 'María', 'Gómez', '2023-06-20', 'mariagomez@example.com'),
    (3, 'Carlos', 'Rodríguez', '2023-12-10', 'carlosrodriguez@example.com');

-- Insertar datos en la tabla Cursos
INSERT INTO Cursos (CursoID, NombreCurso, DescripcionCurso)
VALUES
    (101, 'Matemáticas', 'Matemáticas Avanzadas'),
    (102, 'Historia', 'Historia Mundial'),
    (103, 'Programación', 'Desarrollo Web con JavaScript');

-- Insertar datos en la tabla Notas
INSERT INTO Notas (NotaID, AlumnoID, CursoID, FechaNota, ValorNota)
VALUES
    (1, 1, 101, '2023-10-10', 9.5),
    (2, 1, 102, '2023-10-12', 8.0),
    (3, 2, 101, '2023-10-11', 9.5);

-- Insertar datos en la tabla Becas
INSERT INTO Becas (BecaID, AlumnoID, TipoBeca, FechaInicio, FechaFinalizacion)
VALUES
    (201, 1, 'Beca Completa', '2023-09-01', '2024-08-31'),
    (202, 2, 'Beca Parcial', '2023-09-15', '2024-08-31'),
    (203, 3, 'Sin Beca', NULL, NULL);
```




```
select * from Alumnos;
```

100 %

Results | Messages

	AlumnoID	Nombre	Apellido	FechaNacimiento	CorreoElectronico
1	1	Juan	Pérez	2023-03-15	juanperez@example.com
2	2	María	Gómez	2023-06-20	mariagomez@example.com
3	3	Carlos	Rodríguez	2023-12-10	carlosrodriguez@example.com

```
select * from Becas;
```

100 %

Results | Messages

	BecaID	AlumnoID	TipoBeca	FechaInicio	FechaFinalizacion
1	201	1	Beca Completa	2023-09-01	2024-08-31
2	202	2	Beca Parcial	2023-09-15	2024-08-31
3	203	3	Sin Beca	NULL	NULL

```
select * from Cursos;
```

100 %

Results | Messages

	CursoID	NombreCurso	DescripcionCurso
1	101	Matemáticas	Matemáticas Avanzadas
2	102	Historia	Historia Mundial
3	103	Programación	Desarrollo Web con JavaScript

```
select * from Notas;
```

100 %

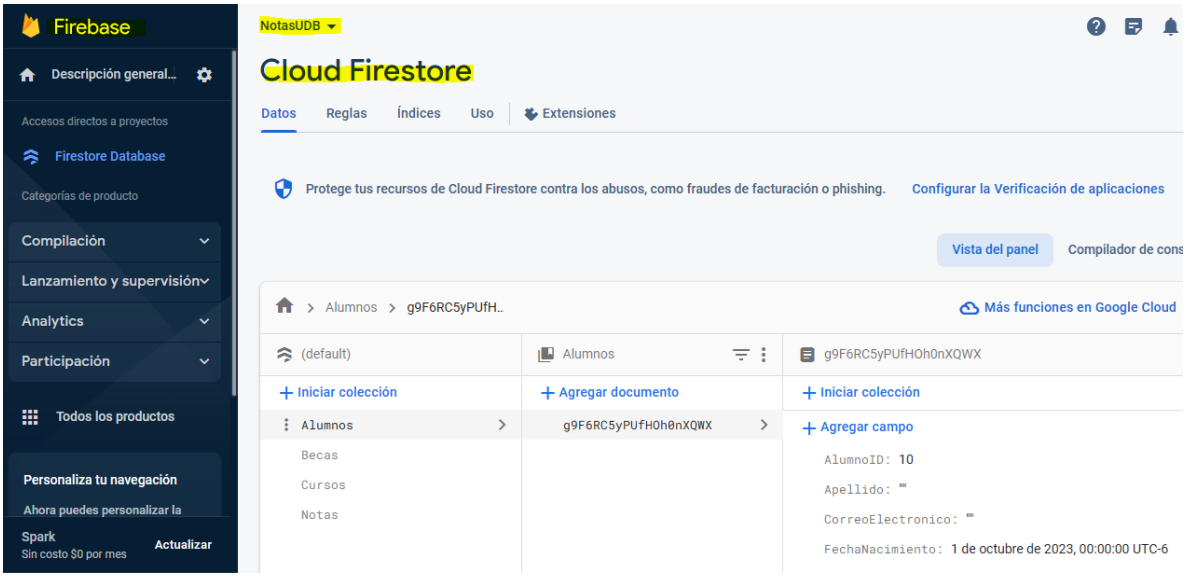
Results | Messages

	NotaID	AlumnoID	CursoID	FechaNota	ValorNota
1	1	1	101	2023-10-10	9.50
2	2	1	102	2023-10-12	8.00
3	3	2	101	2023-10-11	9.50

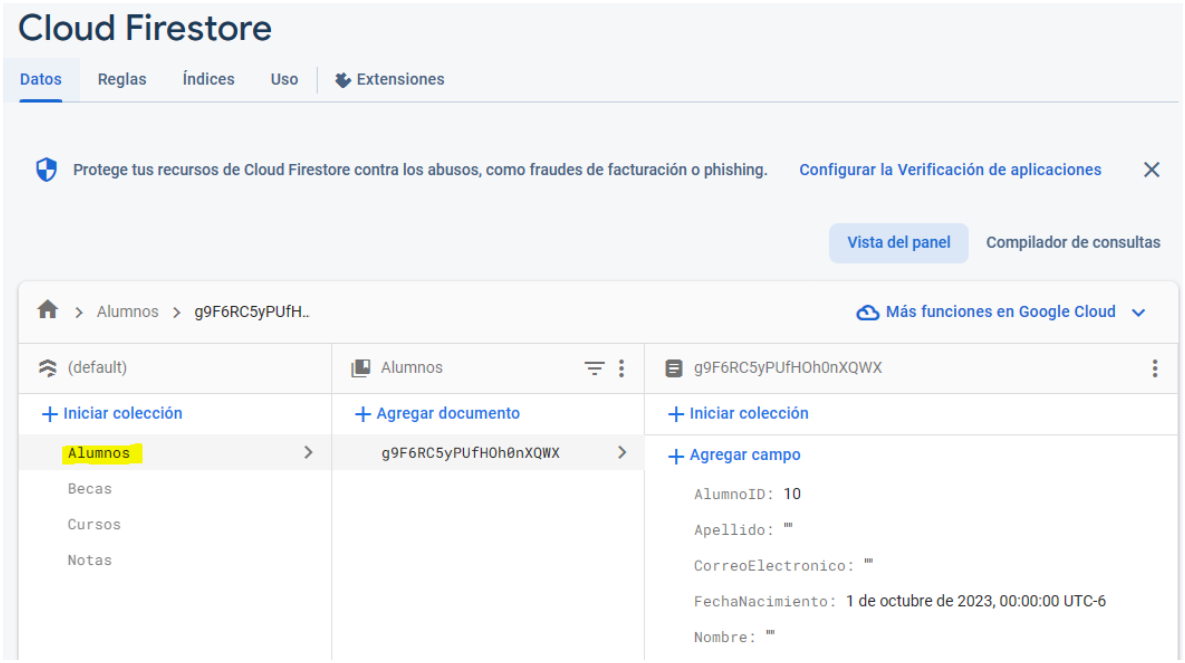
- **Para la base de datos NoSQL, se espera que diseñen una estructura que aproveche las ventajas de Firestore o Realtime Database en Firebase.**

La base que se utilizo es Firebase (Firestore Database), se agrupan diferentes colecciones, esto permite una estructura jerárquica de datos de manera escalable.

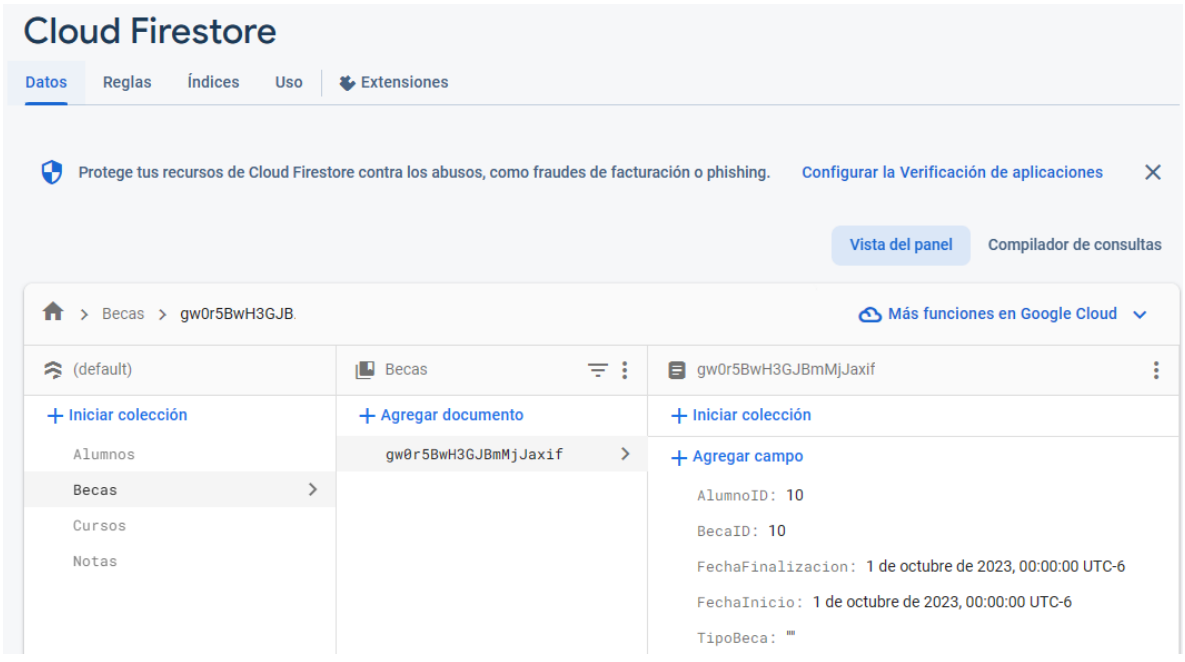
Se creó la base de datos con nombre de NotaUDB, las colecciones a utilizar: Alumnos, Becas, Cursos, Notas.



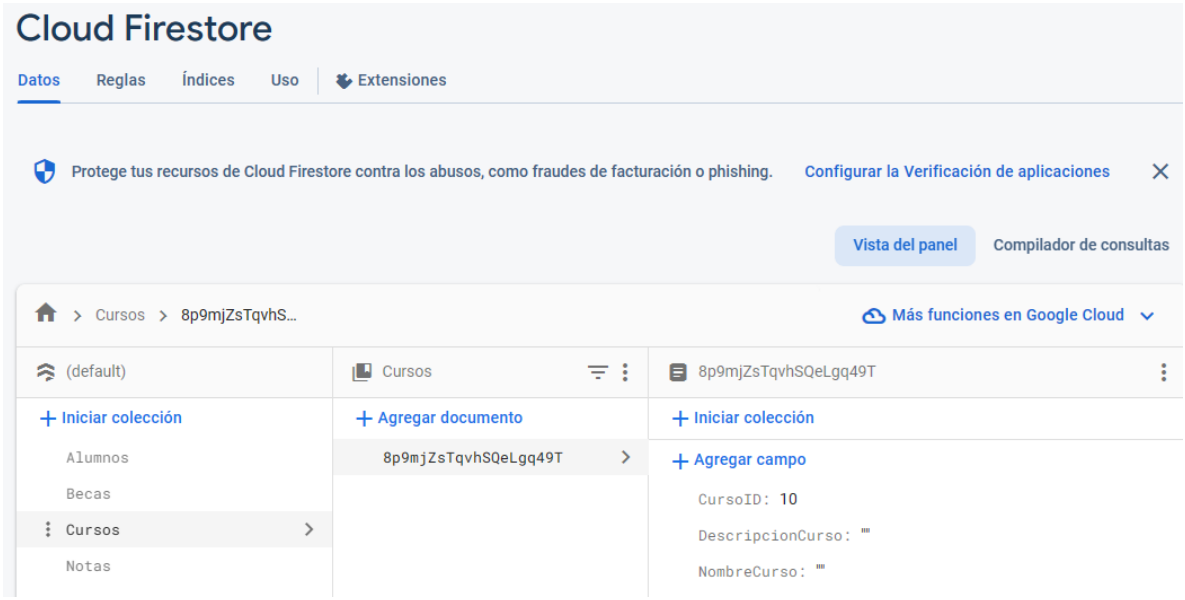
Colección de Alumnos



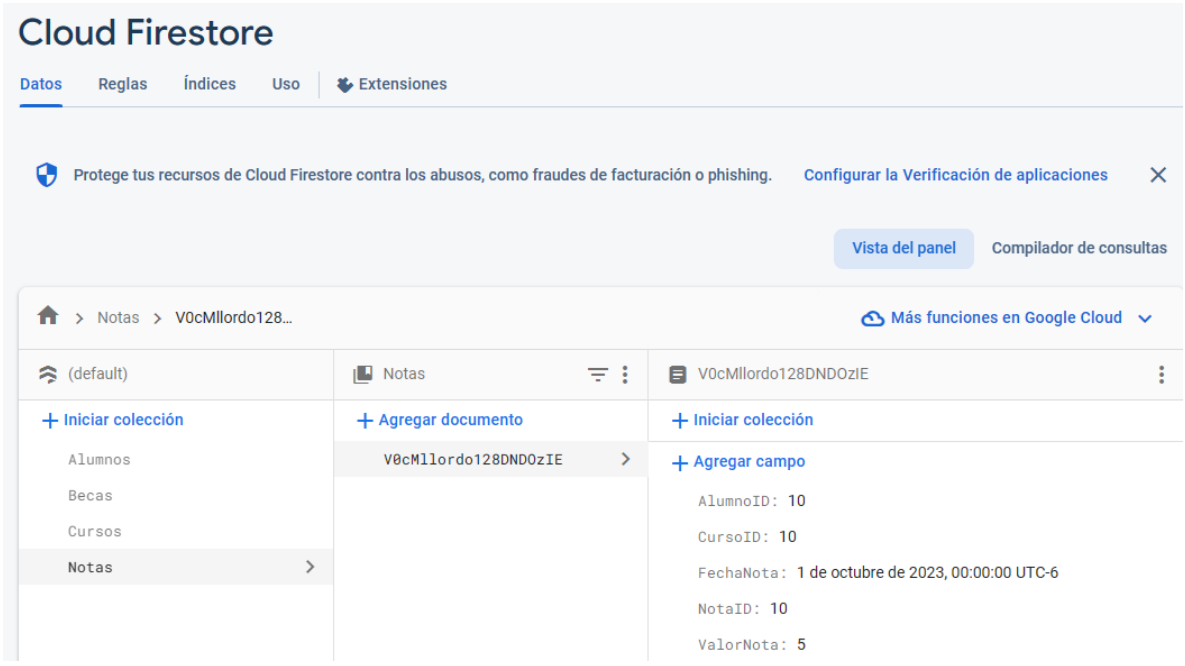
Colección de Becas



Colección Cursos



Colección Notas



Finalmente, se les solicita realizar una comparación de ambas bases de datos y proporcionar conclusiones basadas en su experiencia y en la investigación realizada.

SQL Server es más adecuado para aplicaciones desktop o web que requieren estructuras de datos altamente normalizadas, consultas complejas y cumplimiento de transacciones ACID.

Firebase Realtime Database es ideal para aplicaciones en tiempo real, como aplicaciones móviles, donde la sincronización en tiempo real es fundamental. Ofrece un modelo de datos más flexible.

La elección entre SQL Server y Firebase Realtime Database depende de los requisitos específicos del proyecto a desarrollar, como la estructura de datos, la escalabilidad, la sincronización en tiempo real y el presupuesto. A menudo, se utilizan enfoques mixtos, como Firebase para funciones en tiempo real y SQL Server para el almacenamiento y análisis de datos.