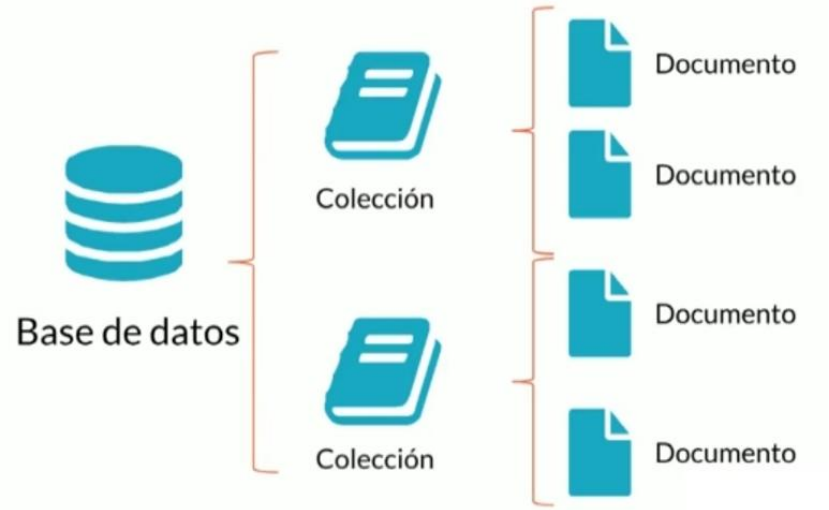
Firebase

Una **base de datos documental** es un tipo de base de datos NoSQL y NO Relacional que organiza y almacena los datos como **documentos**. Cada documento es una unidad individual que contienen información en un formato clave-valor, y posee diferentes estructuras y tipos de datos (como texto, números, listas, objetos anidados, etc.). En lugar de usar tablas y filas, las bases de datos documentales utilizan colecciones de documentos, lo que ofrece una estructura flexible y escalable. Normalmente, se forman jerarquías (**colecciones**) para organizar estos documentos.

Los **documentos** pueden contener cualquier tipo de datos y no necesitan seguir un esquema fijo, lo que permite a los desarrolladores agregar, modificar o eliminar campos sin afectar otras partes de la base de datos.



Quizás, la forma más simple de tener una imagen mental de cómo es una base de datos documental es pensar en un armario y en un montón de papeles. Tú puedes dejar el currículum de un alumno en el cajón “Alumnos 2DAM”, donde estarán todos los currículums de todos los alumnos de esa clase. Ese armario se llama “Alumnos de DAM”, y está en una sala con más armarios que se llama “Alumnos del centro”. No existen reglas en cuanto a cómo es ese currículum, qué campos tiene, o si lo metes en el cajón “Zapatillas de Deporte”. Eso lo controlas tú a parte. La única “regla” es que, a la hora de clasificar ese currículum en su lugar correspondiente, no (deberías) tenerlo duplicado en dos sitios diferentes.

Un ejemplo MUY simple. Supongamos que estamos desarrollando una aplicación para **gestionar libros**. En lugar de usar una tabla con filas y columnas, podríamos usar una base de datos documental. Cada libro sería un **documento** en una colección llamada libros. Cada documento podría tener campos diferentes, como el título, autor, año de publicación, etc.

Libros

├── Misión en París – **Documento**

│ ├── Arturo Pérez Reverte

│ └── 2025

├── El último Secreto – **Documento**

├── Dan Brown

└── 2025

Es posible definir estructuras en árbol más complejas para clasificar la información. La relación que se establece es similar a la de padre-hijo que ya estamos habituados. Un hijo sólo puede tener un padre, pero un padre sólo puede tener un hijo. Obviamente, esto tiene unas limitaciones que otras bases de datos NO tienen.

Un ejemplo un poco más elaborado. Supongamos que queremos ampliar la aplicación **gestionar libros**. Podríamos hacer que cada libro tenga que estar clasificado según su género. Esto nos deja una una estructura similar a esta:

Libros

├── Novela

│ ├── Misión en París – **Documento**

│ ├── Arturo Pérez Reverte

│ └── 2025

│ ├── El último Secreto – **Documento**

│ ├── Dan Brown

│ └── 2025

├── Ensayo

│ ├── El infinito en un junco – **Documento**

│ ├── Irene Vallejo

│ └── 2019

Diseñando una Base de Datos Documental

Cuando diseñamos una Base de Datos Documental, lo primero es entender que no tenemos tablas sino **documentos**. Por tanto, lo ideal sería comenzar definiendo los documentos que conforman nuestra Base de Datos. En el ejemplo anterior, definiríamos que cada 'libro' con su información relevante sería un tipo de documento.

El siguiente paso sería definir las **colecciones**. En una Base de Datos Documental, los Documentos se agrupan en colecciones. En nuestro ejemplo anterior, una colección es 'Ensayo' y otra es 'Novela'. Un Documento pertenece a una única colección, pero no existe más restricciones. De hecho, puedes tener documentos de estructura diferente en una misma colección, aunque la idea es (obviamente) no hacerlo para evitarse complicaciones.

Ahora toca definir cada uno de los **campos** de cada uno de los Documentos. En nuestro ejemplo, el título del libro y su fecha de publicación.

Por último, si bien no podemos tener el mismo Documento en dos partes de la jerarquía, sí podemos tener **referencias** de un documento a otro, o incluso añadir **documentos anidados**.

Para hacer una referencia a otro Documento, podríamos añadir los campos "precuela" y "secuela" al documento libro. En dicho campo añadiríamos el título del anterior libro de la saga y el siguiente (de haberlos), pasa así almacenar nueva información a nuestra Base de Datos Documental.

Por otro lado, para hacer un documento anidado estaríamos almacenando documentos completos dentro de otros documentos. Por ejemplo, podría ser que nuestra Base de Datos Documental almacenase cada Autor como un Documento, pero que la información de la bibliografía del mismo (sus Libros), fuesen una parte íntegra del Documento Autor.

¿Qué es Firebase?

Firebase es una plataforma de **desarrollo de aplicaciones móviles (y web)** creada por Google en el 2014. Ofrece un conjunto de servicios en la nube que permiten a los desarrolladores centrarse más en la lógica de su app que en la infraestructura.

Firebase ofrece:

- **Base de datos en tiempo real y Firestore**: permiten almacenar y sincronizar datos entre clientes y servidores de forma instantánea.

- **Autenticación**: facilita la gestión de usuarios con login mediante correo, redes sociales o proveedores externos.

- **Hosting**: para el despliegue rápido y seguro de aplicaciones web.

- **Cloud Functions**: para ejecutar código backend sin necesidad de tener servidores propios.

- **Analytics**: estadísticas de uso sobre el comportamiento de los usuarios.

- **Mensajería (FCM)**: envío de notificaciones push a móviles y web.

La plataforma es en principio gratuita, aunque como siempre, de necesitar más espacio (etc.) sería necesario cambiar a la modalidad de pago.

A nosotros nos interesa de Firebase la opción de usarla como Base de Datos online (**Cloud Firestore**) que es de tipo Documental (No SQL).

Podemos acceder a todas las posibilidades de Firebase desde:

https://firebase.google.com/

Ejercicio Propuesto - 1

Diseña el esquema de una BBDD Documental que almacenará la información de todos los alumnos que asisten a Elorrieta. La información que tenemos que almacenar de cada alumno es la siguiente:

- Nombre
- Apellido
- Edad

Los alumnos pertenecerán a una única clase (1º o 2º) y a un único módulo (DAM o DAW).

Solución Propuesta nº1

Podríamos hacer una **estructura jerárquica** similar a esta:

centros (Colección)
└── centro01 (Documento)
    └── ciclos (Colección)
        ├── ciclo DAM (Documento)
        │   └── clases (Colección)
        │   ├── clase1 (Documento)
        │   │   └── alumnos (Colección)
        │   └── clase2 (Documento)
        │   └── alumnos (Colección)
        └── ciclo DAW (Colección)
            └── clases (Colección)
                ├── clase1 (Documento)
                │   └── alumnos (Colección)
                └── clase2 (Documento)
                    └── alumnos (Colección)

De forma que cada uno de los alumnos sea un Documento similar a este:

```
{
  "id": "A001",
  "nombre": "Juan",
  "apellido": "Pérez"
}
```

Para realizar una búsqueda de información, tendríamos que ir navegando por la estructura de los documentos de forma sucesiva para ir recuperando la información necesaria. Esta solución permite recuperar de forma sencilla conjuntos de alumnos; pero quizás no es la mejor para averiguar la clase a la que pertenece Juan.

Solución Propuesta nº2

Podríamos hacer una **estructura con referencias** similar a esta:

centros (Colección)

└── centro01 (Documento)

└── ciclos (Colección)

└── ciclo DAM (Documento)

└── clases (Colección)

└── clase1 (Documento)

De forma que cada uno de los alumnos sea un Documento similar a este, con referencias a otros documentos mediante:

```
{
  "id": "A001",
  "nombre": "Juan",
  "apellido": "Pérez",
  "centroId": "centro01",
  "cicloId": "ciclo DAM",
  "claseId": "clase1"
}
```

En este caso, lo tendríamos un poco más sencillo. Para recuperar información de un alumno, disponemos de referencias a otros documentos: si queremos averiguar el ciclo de Juan, recuperamos su “cicloId” y realizamos una búsqueda secundaria. Puede que no sea tan sencillo recuperar un conjunto de alumnos, en cambio.

Este tipo de solución se supone que es la más adecuada para **Firestore**.