¡Les damos la bienvenida!

¿Comenzamos?

Esta clase va a ser

- grabada

Clase 14. REACTS JS

Firebase II

Objetivos de la clase

- **Insertar y actualizar** información usando Firestore
- **Flujos de control**

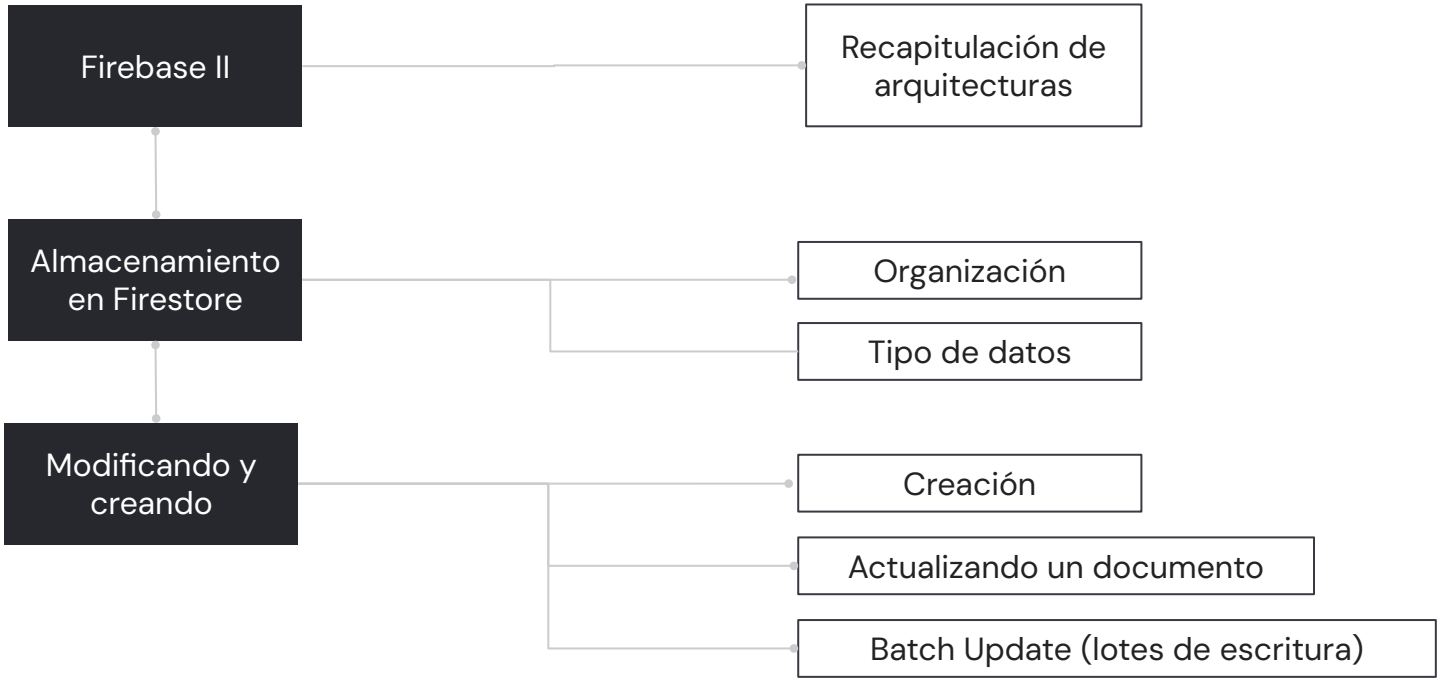
CODERHOUSE

CLASE N°13

Glosario

Firebase: Es un servicio provisto por google para satisfacer las distintas necesidades que puede tener una aplicacion y su ciclo de desarrollo.

MAPA DE CONCEPTOS



Temario

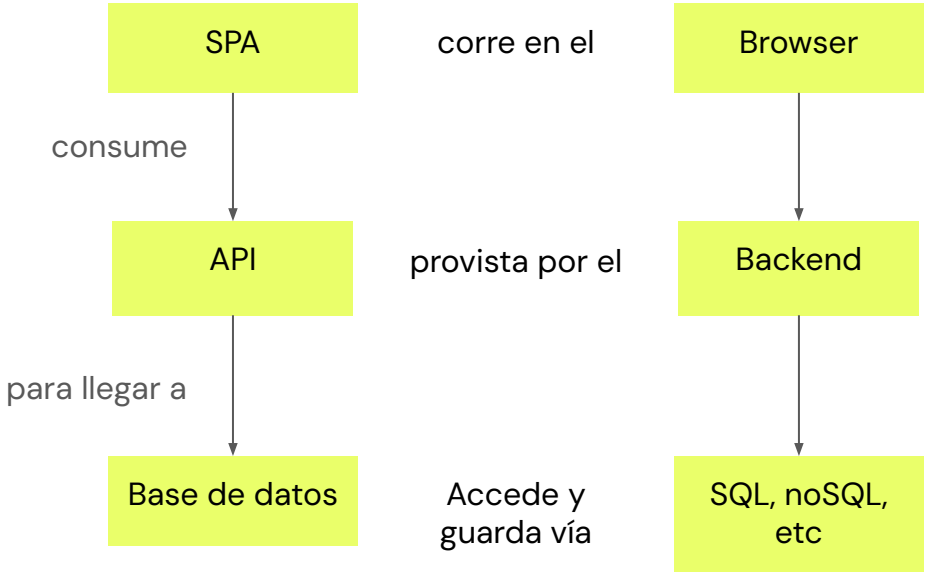
13	14	15
Firebase I	**Firebase II**	**Workshop**
✓ Firebase	✓ Firebase II	✓ Optimización de código
✓ ¿Por dónde empiezo?	✓ Almacenando en Firestore	✓ Prepara tu proyecto
✓ Firestore	✓ Modificando y creando	
✓ Configurando nuestra app		

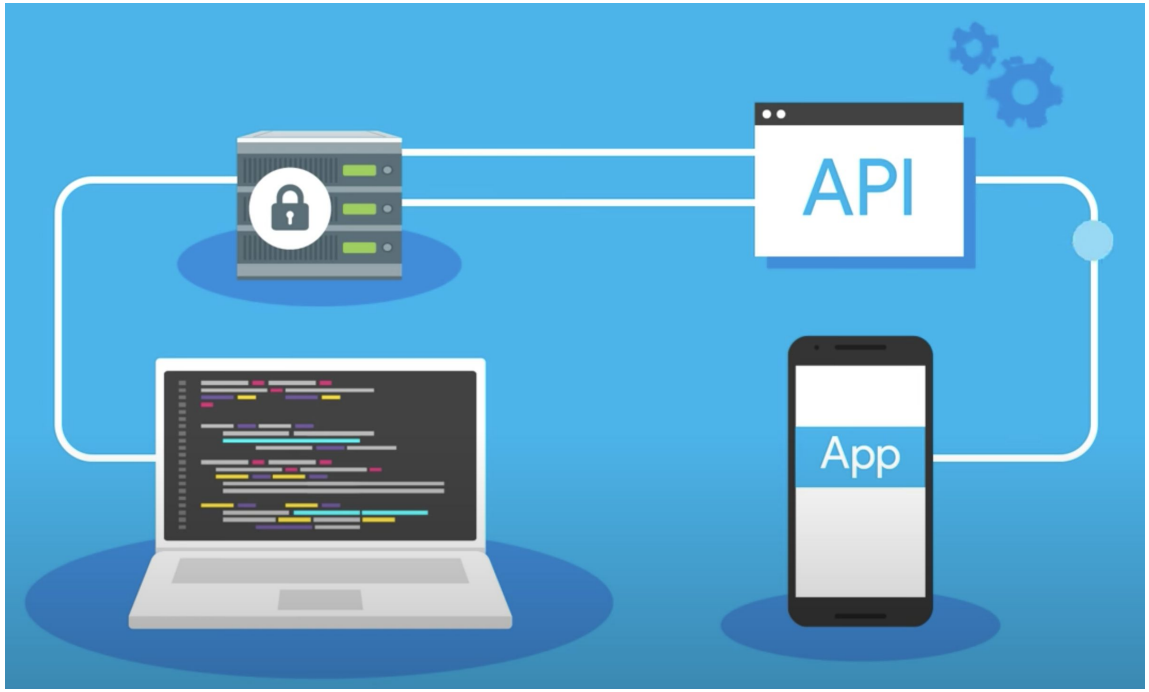
CODERHOUSE

Firebase II

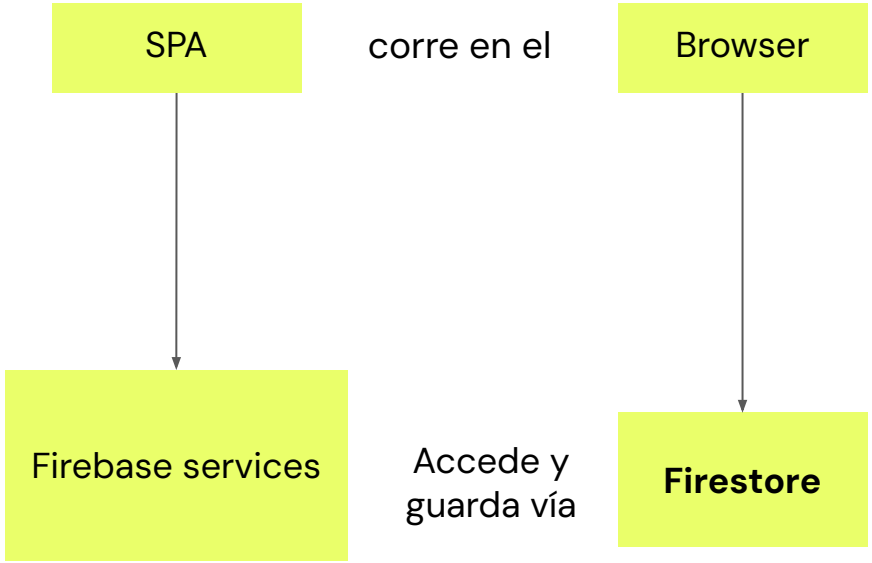
Recapitulación de arquitecturas

Si venimos de los desarrollos clásicos, tenemos una arquitectura que es la más conocida. Puede estar vinculada a un patrón como el siguiente:





La api decide quién accede, implementa su modelo de seguridad y determina la respuesta



Firestore decide quién accede mediante mecanismos propios y filtros de seguridad.

Almacenamiento en firestore

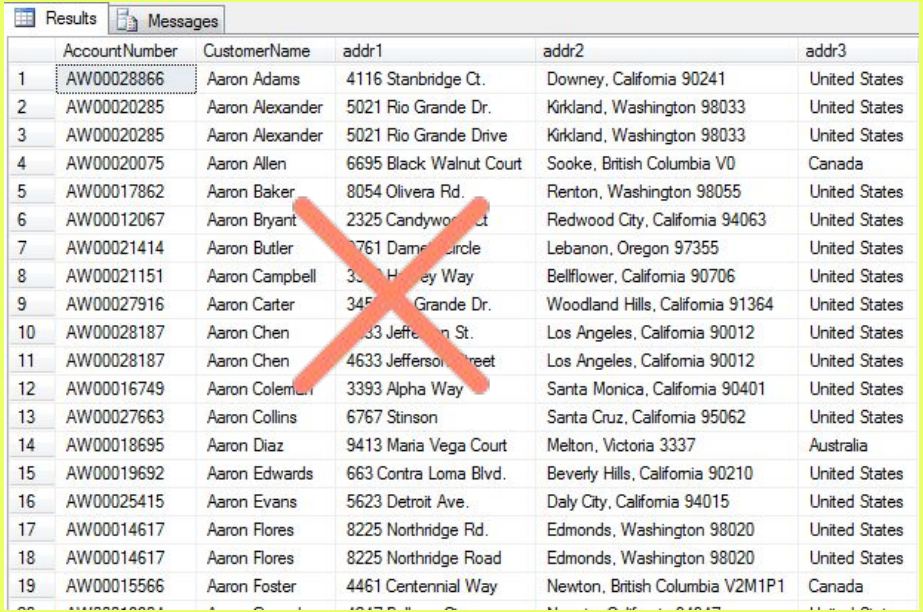
Firestore y el almacenamiento

Firestore tiene ciertos límites en cuanto a cómo organizamos la información. Veamos los distintos aspectos, para hacerlo de manera eficiente.

CODERHOUSE

Organización

FIRESTORE



Results | Messages

	AccountNumber	CustomerName	addr1	addr2	addr3
1	AW00028866	Aaron Adams	4116 Stanbridge Ct.	Downey, California 90241	United States
2	AW00020285	Aaron Alexander	5021 Rio Grande Dr.	Kirkland, Washington 98033	United States
3	AW00020285	Aaron Alexander	5021 Rio Grande Drive	Kirkland, Washington 98033	United States
4	AW00020075	Aaron Allen	6695 Black Walnut Court	Sooke, British Columbia V0	Canada
5	AW00017862	Aaron Baker	8054 Olivera Rd.	Renton, Washington 98055	United States
6	AW00012067	Aaron Bryant	2325 Candywo[illegible]ct	Redwood City, California 94063	United States
7	AW00021414	Aaron Butler	[illegible]761 Dame[illegible]ircle	Lebanon, Oregon 97355	United States
8	AW00021151	Aaron Campbell	3[illegible] H[illegible]ey Way	Bellflower, California 90706	United States
9	AW00027916	Aaron Carter	345[illegible] Grande Dr.	Woodland Hills, California 91364	United States
10	AW00028187	Aaron Chen	[illegible]33 Jeffe[illegible]n St.	Los Angeles, California 90012	United States
11	AW00028187	Aaron Chen	4633 Jefferso[illegible]reet	Los Angeles, California 90012	United States
12	AW00016749	Aaron Colem[illegible]	3393 Alpha Way	Santa Monica, California 90401	United States
13	AW00027663	Aaron Collins	6767 Stinson	Santa Cruz, California 95062	United States
14	AW00018695	Aaron Diaz	9413 Maria Vega Court	Melton, Victoria 3337	Australia
15	AW00019692	Aaron Edwards	663 Contra Loma Blvd.	Beverly Hills, California 90210	United States
16	AW00025415	Aaron Evans	5623 Detroit Ave.	Daly City, California 94015	United States
17	AW00014617	Aaron Flores	8225 Northridge Rd.	Edmonds, Washington 98020	United States
18	AW00014617	Aaron Flores	8225 Northridge Road	Edmonds, Washington 98020	United States
19	AW00015566	Aaron Foster	4461 Centennial Way	Newton, British Columbia V2M1P1	Canada

En Firestore, el almacenamiento es de tipo no estructurado/noSQL:

- ✓ No hay tablas.
- ✓ No hay filas/records.

CODERHOUSE

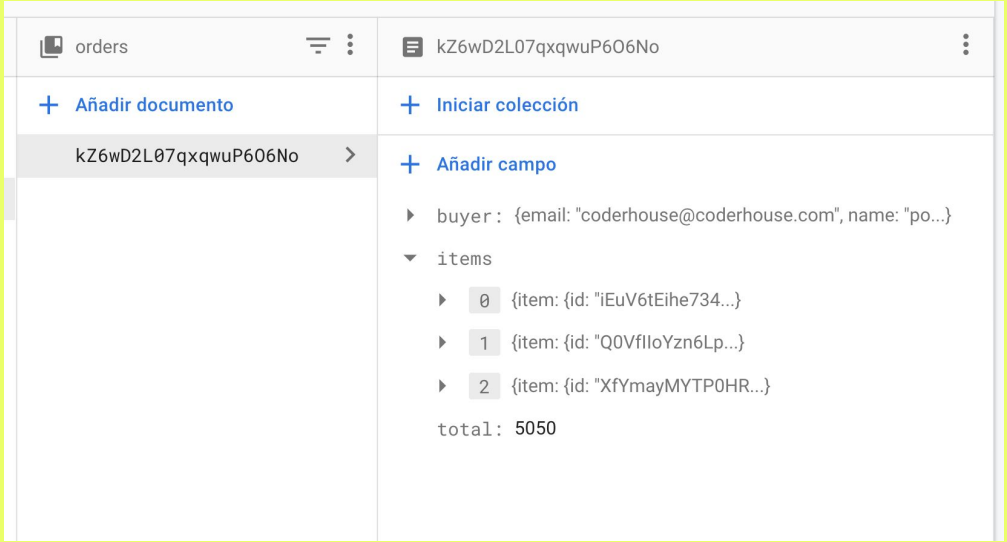
FIRESTORE: DOCUMENTOS

En Firestore hay documentos.

- ✓ Tienen pares key/value.
- ✓ Estos key value pueden tener ciertos tipos de datos.
- ✓ Tienen un límite de 1MB

CODERHOUSE

FIRESTORE: DOCUMENTOS

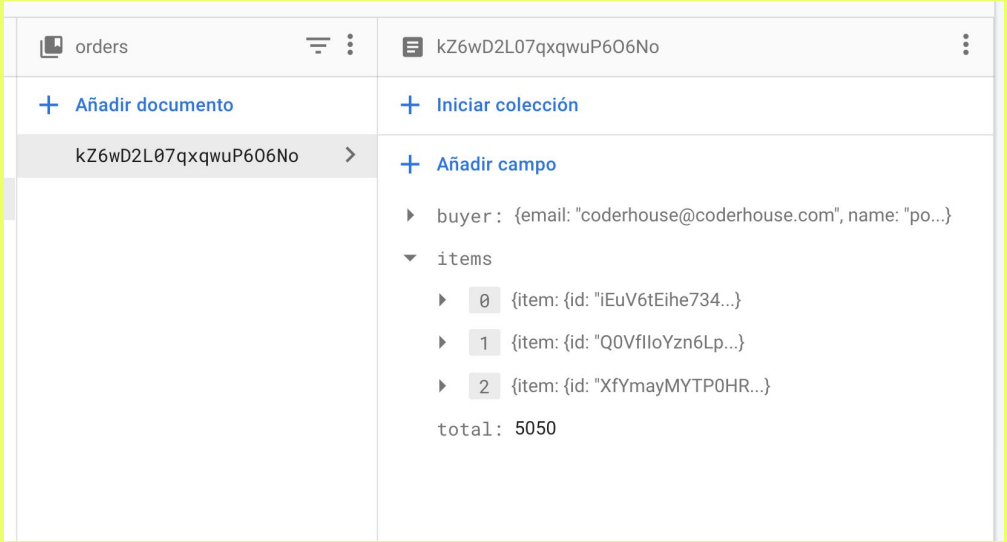


Los documentos pueden ser complejos y anidados, pueden contener arrays, fechas (timestamps), números, y otros objetos (maps)

¡Juega y experimenta con el editor online y descubre más datos!

CODERHOUSE

FIRESTORE: COLECCIONES

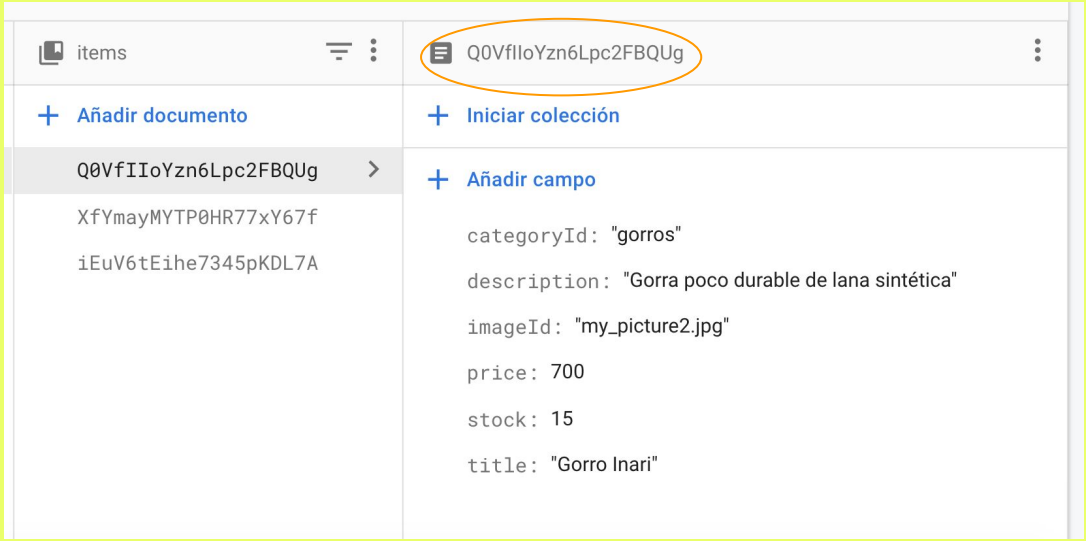


- ✓ Son contenedores de documentos.
- ✓ Los documentos se agrupan obligatoriamente dentro de ellas.
- ✓ Los mismos documentos pueden tener colecciones dentro.

Borrar un documento no elimina sus sub-colecciones ¡debes hacerlo manualmente!

CODERHOUSE

FIRESTORE: COLECCIONES



Si bien tienen máximo de 1MB, esto puede ser bastante para registros de texto, y aparte podemos definir hasta 100 niveles de sub-colecciones.

Esto permite multiplicar exponencialmente el tamaño bruto de un documento.

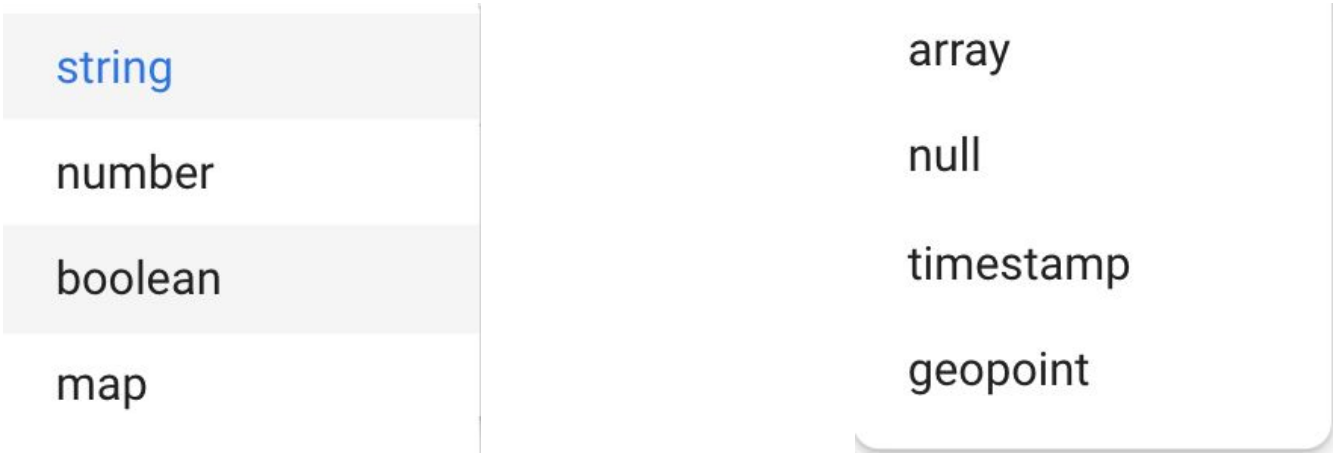
Ejemplo en vivo

Vamos al código.

Tipo de datos

CODERHOUSE

FIRESTORE: TIPOS DE DATOS



Cuidado: arrays no pueden tener sub-arrays entre sus elementos

Modera tus orders

Usa tus items del cart para modelar tu orden

Duración: **10 minutos**

CODERHOUSE

ACTIVIDAD EN CLASE

¡A PRACTICAR!

Descripción de la actividad

Usa tu tus ítems del cart para modelar tu orden al siguiente formato:

{ buyer: { name, phone, email }, items: [{id, title, price}], total }, si todavía no creaste el formulario de compra puedes usar un objeto hardcodeado de tipo { name, phone, email }.

Break

¡10 minutos y volvemos!

¡Ya estás llegando al fin de la cursada!
Recuerda que luego de la corrección de tu proyecto final, se notificará por Slack y email si quedaste en el TOP10

No lo dejes estar. Solo tienes hasta 2 semanas desde que te notificamos para solicitar los beneficios.

¡10 minutos y volvemos!

CODERHOUSE

Firestore: checklist challenge

- Tienes tu Car Provider
- Puedes agregar items a tu Cart
- Puedes navegar de tu List a tu Detail
- Puedes clikear en el Cart
- Conectaste el listado y el detalle a Firebase
- ¡Conectemos la compra y generemos la orden!

CODERHOUSE

Modificando
y creando

Creación

Firestore: preparar documento

```
import { collection, getFiresore } from "firebase/firestore";

// ...

const sendOrder = () => {
    const order = {
      buyer: { name: "Agustin", phone: "1111", email: "a@a.com" },
      items: [{ name: "Bici", price: 100 }],
      total: 100
    };
    const db = getFirestore();

    const ordersCollection = collection(db, "orders");

    // ...
}
```

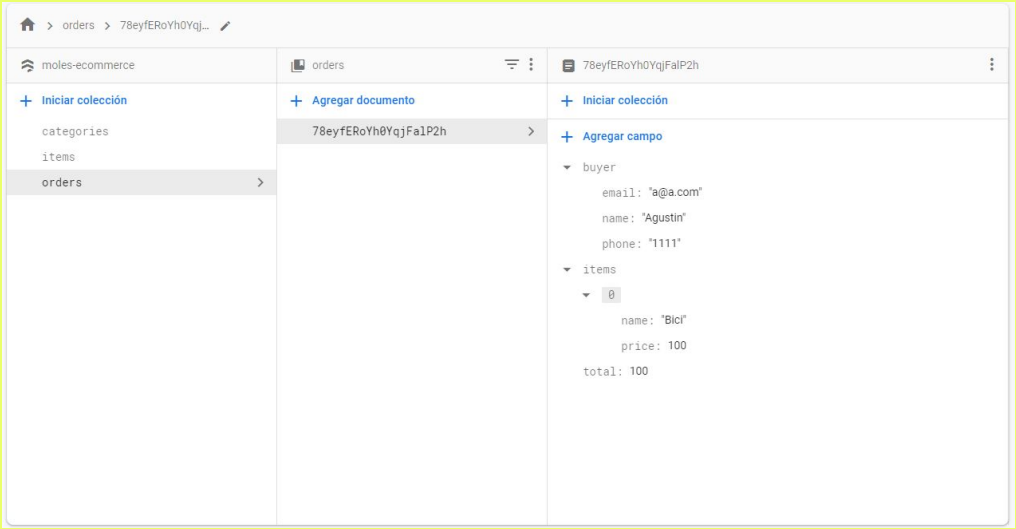
Debemos generar una referencia a la colección, y crear el objeto que queremos crear en Firestore.

Firestore: crear documento

```
import { addDoc, collection, getFiresore } from "firebase/firestore";

// ...

const sendOrder = () => {
    const order = {
      buyer: { name: "Agustin", phone: "1111", email: "a@a.com" },
      items: [{ name: "Bici", price: 100 }],
      total: 100
    };
    const db = getFirestore();

    const ordersCollection = collection(db, "orders");

    addDoc(ordersCollection, order).then(({ id }) => setOrderId(id));
}
```

El método addDoc devuelve una promise. Si todo sale bien el then devolverá un objeto con el id autogenerado del nuevo documento, de otro modo pasará por el flow de error.

Firestore: verifica tu creación



¡Verifica en la consola que se haya creado!

Firestore crea el id por nosotros si utilizamos el método **addDoc**

CODERHOUSE

Actualizando un documento

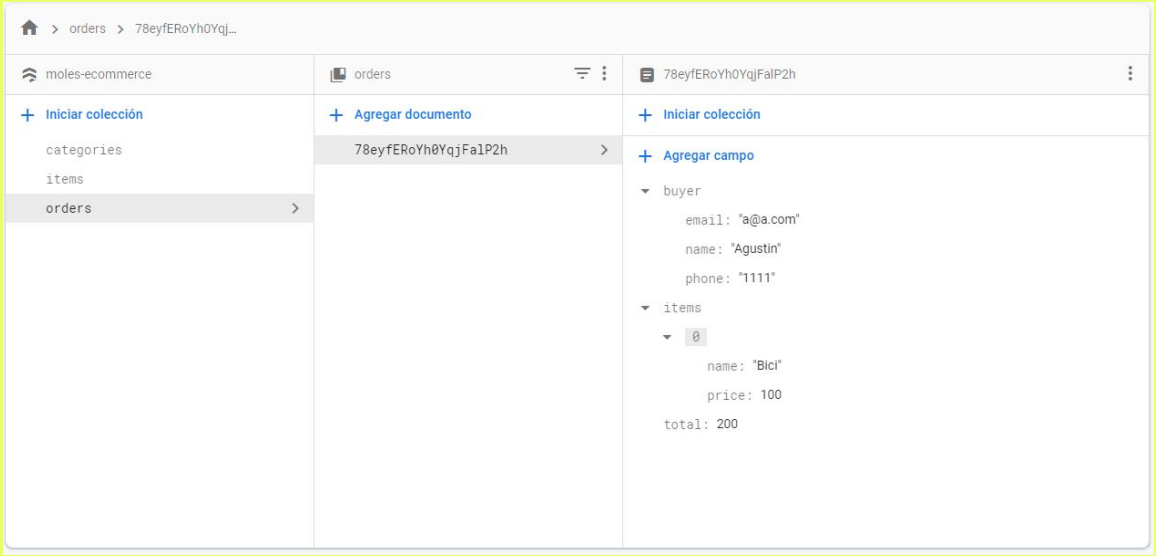
CODERHOUSE

Firestore: actualizar un documento

```
import { doc, getFiresore, updateDoc } from "firebase/firestore";

// ...

const updateOrder = () => {
    const db = getFirestore();

    const orderDoc = doc(db, "orders", "78eyfERoYh0YqjFalP2h");
    updateDoc(orderDoc, { total: 200 });
}
```

Puedo darle únicamente los campos que quiero actualizar. Firestore combinará este nuevo campo y dejará lo que haya habido previamente

Firestore: verifica tu creación



¡Verifica en la consola que se haya modificado!

CODERHOUSE

Batch update
(lotes de escritura)

Firestore: actualizar un lote de documentos

Puedes usar una operación batch para actualizar muchos documentos en una misma operación

```
import { getFiresore, writeBatch } from "firebase/firestore";

// ...

const updateOrder = () => {
    const db = getFirestore();

    const batch = writeBatch(db);

    // ... Obtener las referencias a los docs con doc() ...

    batch.update(doc1, { total: 150 });
    batch.set(doc2, { field: 'new field value' });

    batch.commit();
}
```

Pasos:

1. Obtener una instancia de firestore.
2. Obtener un batch.
3. Ejecutar las operaciones requeridas.
4. Ejecutar el commit()

#Coderalert

Ingresa al manual de prácticas y realiza la octava actividad “Item Collection II”. Ten en cuenta que el desarrollo de la misma será importante para la resolución del Proyecto Final.

Actividad Nº 8

Proyecto Final

Item Collection II

Descripción de la actividad.

- ✓ Crea tu colección de órdenes.

Recomendaciones

- ✓ Utiliza las operaciones de inserción para insertar tu orden en la colección y dale al user su id de orden auto-generada
- ✓ Crea los mappings para poder grabar un objeto del formato { buyer: { name, phone, email }, items: [{id, title, price}], date, total }
- ✓ Pista: Puedes controlar los stocks con multi-gets utilizando los itemId de tu cart.

¿Preguntas?

Resumen
de la clase hoy

- ✓ Firebase.
- ✓ Firestore.
- ✓ Queries.

Muchas gracias.

Opina y valora esta clase