



# ¡Les damos la bienvenida!

¿Comenzamos?

Esta clase va a ser

• grabada

Clase 13. REACT JS

# Firebase I

# Objetivos de la clase



- Diferenciar** el modelo de serverless-computing de un modelo tradicional.
- Conectar** nuestra app a servicios cloud-first de alta disponibilidad.
- Configurar** un storage, sus colecciones y acceder a la información.



# MAPA DE CONCEPTOS



# Temario

12

## Renderizado Condicional

- ✓ Recapitulando: principios básicos react
- ✓ Rendering condicional
- ✓ Render optimization

13

## Firebase I

- ✓ [Firebase](#)
- ✓ [¿Por dónde empiezo?](#)
- ✓ [Firestore](#)
- ✓ [Configurando](#) nuestra App

14

## Firebase II

- ✓ Firebase II
- ✓ Almacenando en Firestore
- ✓ Modificando y creando

**Firebase**

**CODERHOUSE**

# WHYW?

# WHYW: El proceso clásico de explorar nuevas tecnologías



Fuente: extraído de la presentación cómica de Brian D. Gilbert

[How to make a perfect E3 press conference \(or drinking game\) | Unraveled](#)

# WHYW: El proceso clásico de explorar nuevas tecnologías

## 1. What?

Sorpresa, intriga ¿cómo hace eso?

## 2. Hell Yeah!

Nuestra emoción cuando nos responden esa pregunta solo con los beneficios infinitos y lo “simple” de adoptarla.

## 3. What?

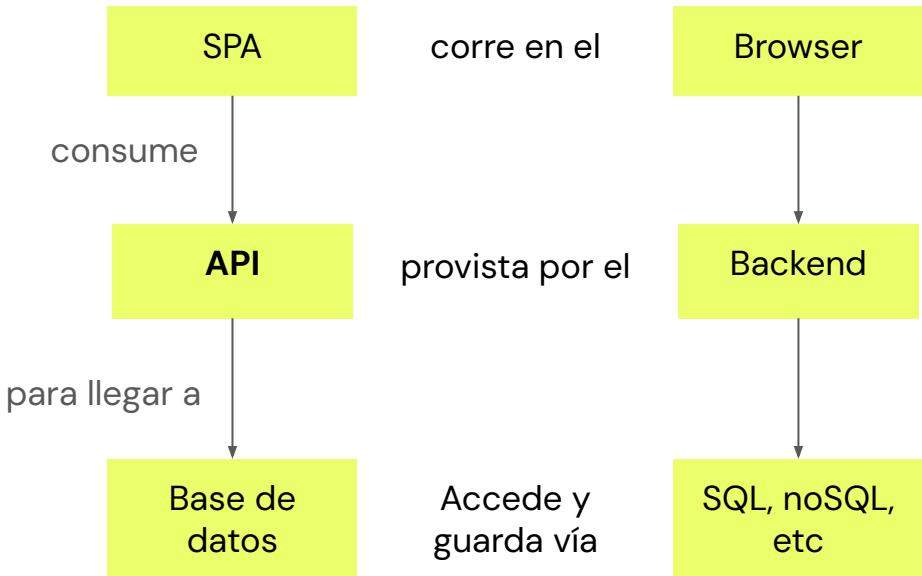
Cuando empezamos a usarla, se comporta mágicamente y realmente no sabemos qué estamos haciendo.



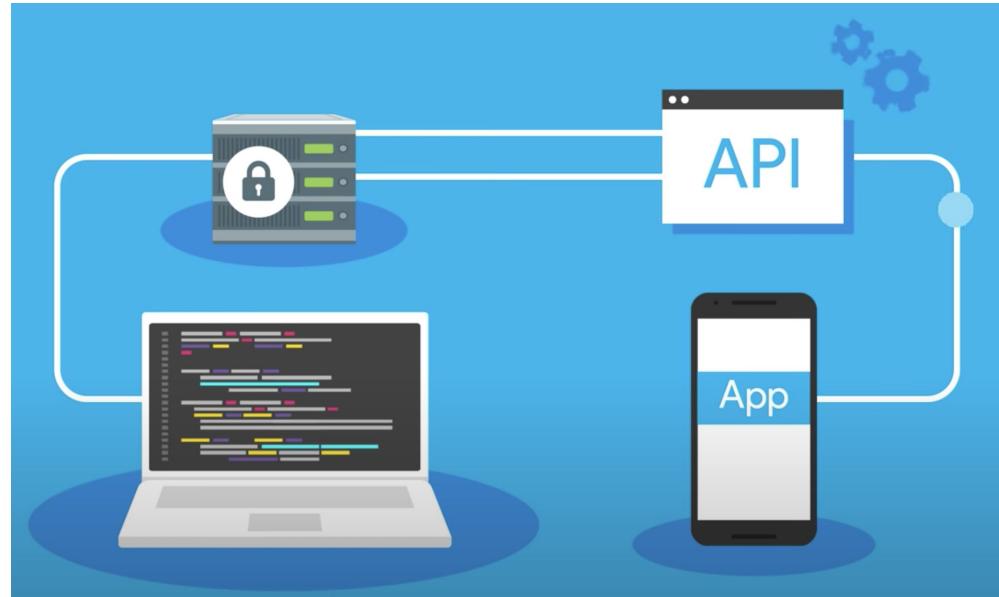
# Hablemos de arquitecturas

Si venimos de los desarrollos **clásicos**, tenemos una arquitectura que es la más conocida.

Puede estar vinculada a un patrón como el siguiente

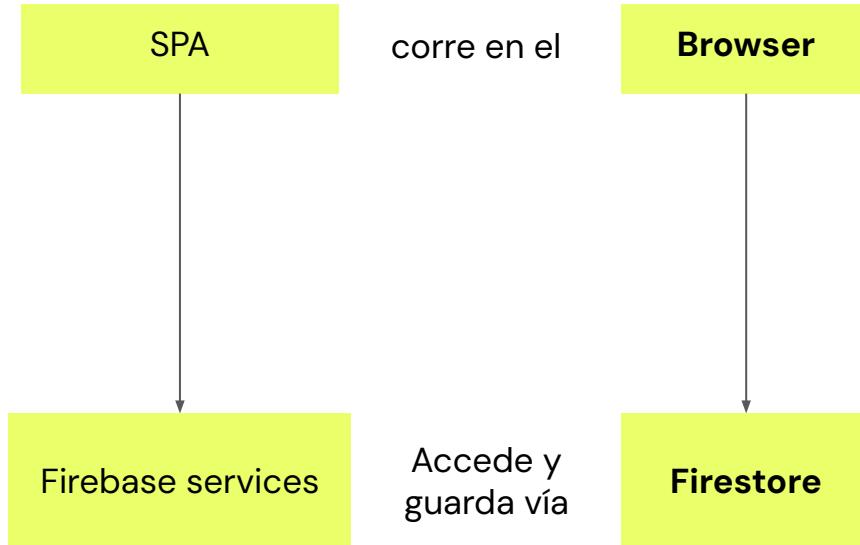


En muchos casos esta sigue siendo una alternativa cómoda para desarrollar, cuando vamos ganando experiencia en backend.



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

Hoy las tecnologías nos permiten rearmar este esquema, para que todos los desarrollos **no necesariamente requieran este patrón**, llegando a algo como lo siguiente:



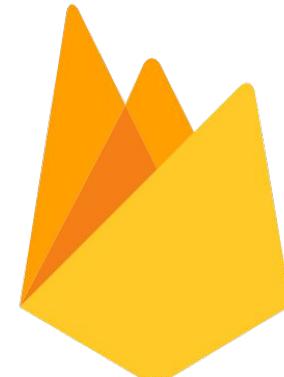
Firestore decide quién accede mediante mecanismos propios y filtros de seguridad.

# ¿Qué es entonces Firebase?

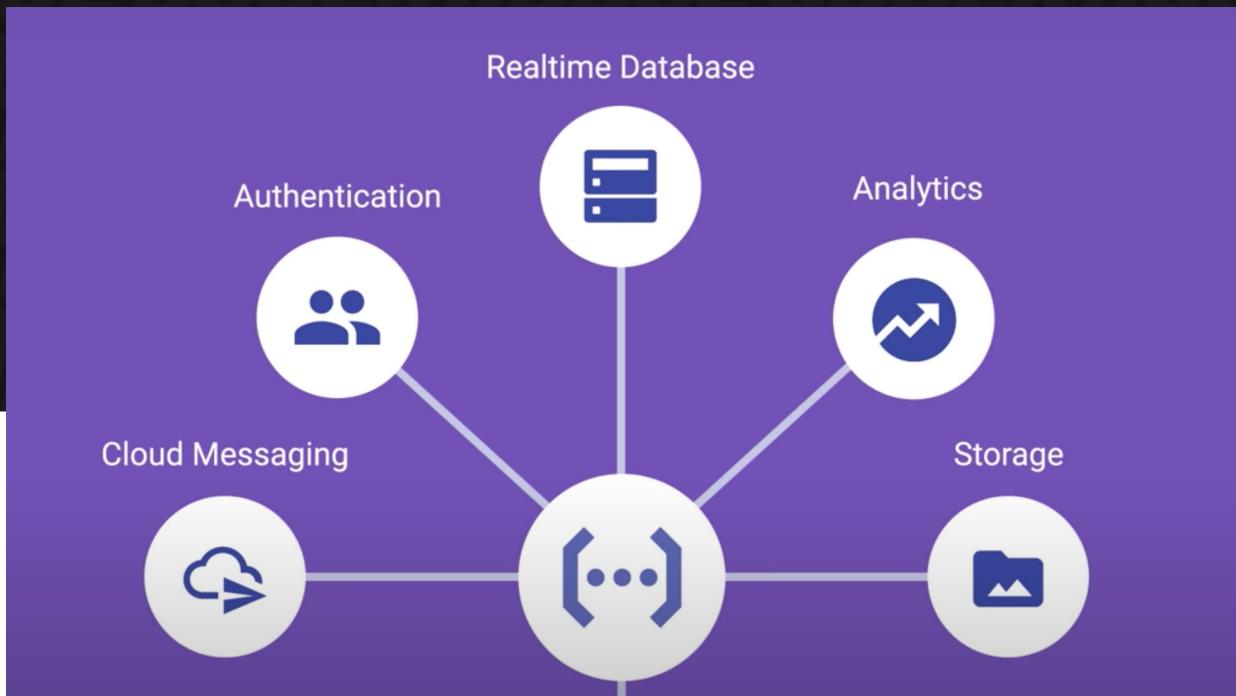
# Firebase

Es un servicio provisto por **Google** para satisfacer las distintas necesidades que puede tener una aplicación y su ciclo de desarrollo, dentro de las cuales encontramos:

- ✓ Seguridad y autenticación
- ✓ Almacenamiento y consulta
- ✓ Hosting
- ✓ Monitoreo
- ✓ Functions y más



# Firebase



¿Por dónde empiezo?

# Creando un proyecto de Firebase

# Accediendo a Firebase: registro

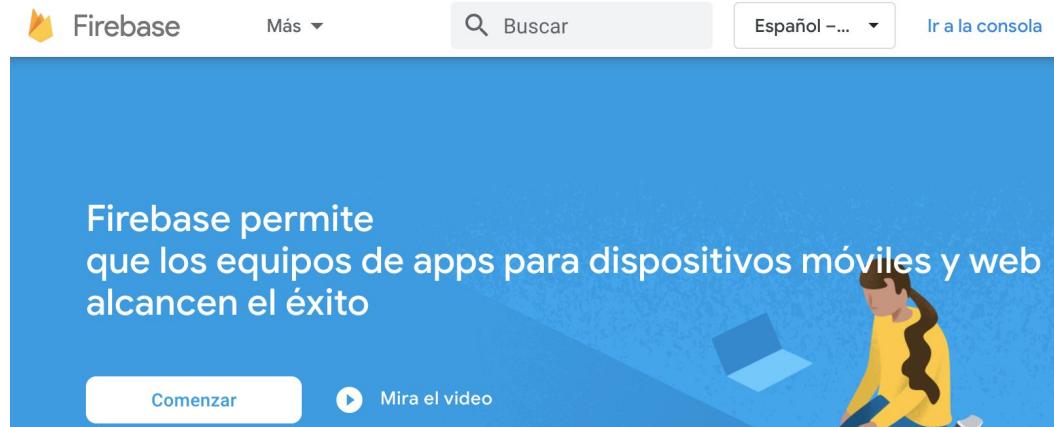
Para acceder a los servicios de **firebase** necesitas una cuenta de Google o Gmail, si no la tienes ve ahora y regístrate, es muy sencillo:



The screenshot shows the Google sign-in interface. It features the Google logo and the word 'Acceder' (Sign in). Below that is a link 'Ir a Google Developers'. The main area has a text input field labeled 'Correo electrónico o teléfono' with a placeholder 'Introduce tu correo electrónico'. Below the input field is a link '¿Olvidaste el correo electrónico?'. At the bottom left is a 'Crear cuenta' (Create account) button, and at the bottom right is a 'Siguiente' (Next) button.

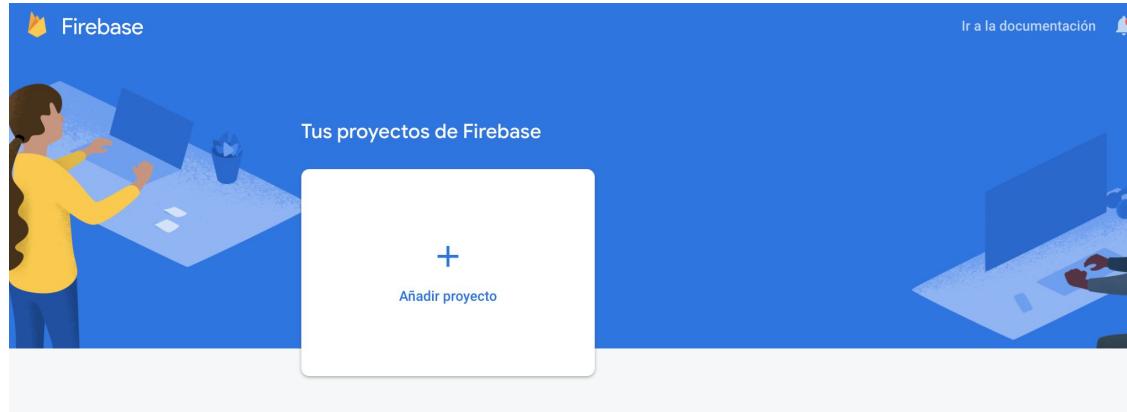
# Accediendo a Firebase: acceso a la consola

Una vez logueados, podemos ver la consola:



# Accediendo a Firebase: crear un proyecto

Aquí veremos todos nuestros proyectos de firebase, lo esencial es crear uno como punto de partida para nuestro e-commerce.



# Accediendo a Firebase: crear un proyecto

En este paso podremos indicarle con qué nombres queremos conocer este proyecto, utiliza un nombre significativo para diferenciarlo de otros

The image shows two screenshots of the Firebase project creation interface. On the left, 'Crear un proyecto (paso 1 de 3)' shows a text input field with 'coderhouse-e-commerce' and a 'Continuar' button. A pink arrow points from this screen to the right one. On the right, 'Crear un proyecto (paso 2 de 2)' lists Google Analytics features: A/B Testing, Segmentación de usuarios en productos de Firebase, Predicción del comportamiento de usuarios, Usuarios sin fallos, Activadores de Cloud Functions basados en eventos, and Informes gratuitos ilimitados. It also has a 'Habilitar Google Analytics en este proyecto' toggle (set to off) and 'Anterior' and 'Crear proyecto' buttons.

Al no ser que ya seas user y conozcas **analytics** omite el paso 2 por simpleza.

# Accediendo a Firebase: crear un proyecto

Una vez que esté todo, ¡Firebase se encargará de dejar todo listo para que podamos empezar a trabajar!

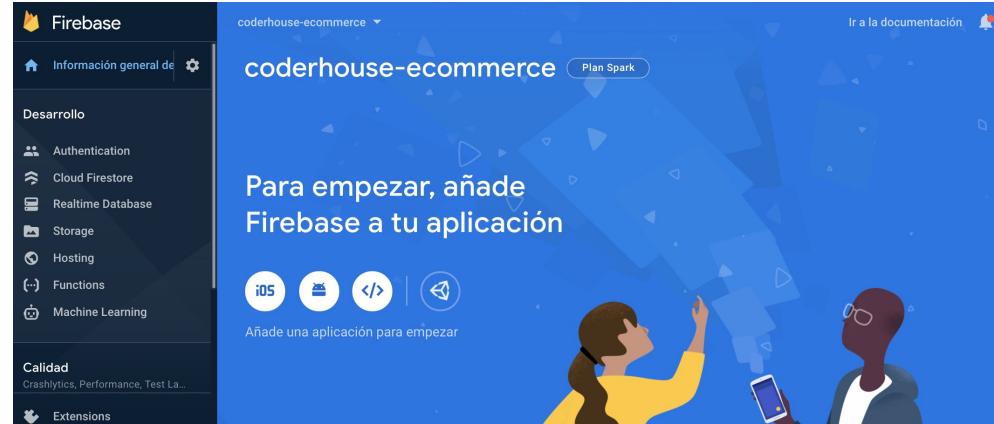


Al no ser que ya seas user y conozcas **analytics** omite el paso 2 por simpleza.

# Panel de control

# Accediendo a Firebase: overview

Ahora en el panel podemos observar que estamos consumiendo el  
**“Plan Spark” (Tier gratuito)**



Este panel te permitirá configurar otras funcionalidades extra muy interesantes.

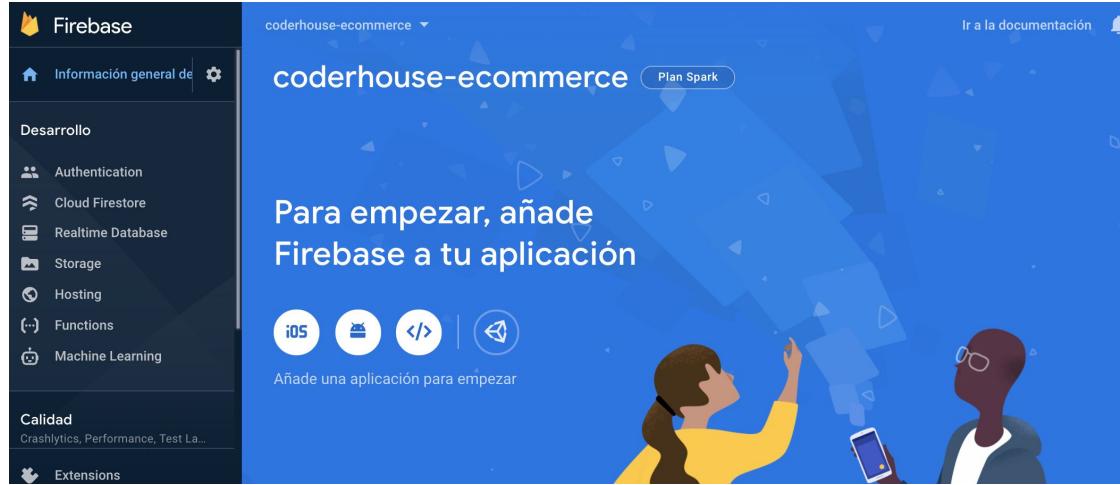
# Firebase

- ✓ Es una **plataforma** de servicios para el desarrollo de apps.
- ✓ Permite de manera simple y rápida crear aplicaciones ricas en contenido, interactividad y persistencia.
- ✓ Si bien nosotros usaremos el free-tier (spark), una vez que se excedan las cuotas podemos configurarlo para utilizar un plan “pay-as-you-go”, que cobra de acuerdo al consumo.
- ✓ No requerimos extensos conocimientos de infraestructura o mantenimiento, para acceder a nuestra app escalable.

**Firestore**

# Firestore

En el panel podemos acceder a **firestore** para empezar a diseñar nuestra capa de datos dinámicos, y poder enriquecer la app:



Firestore es una de las soluciones para acceder a la información.

# Creación

# Firestore: creando un store



Desde el mismo panel, podemos crear una base de datos:

A screenshot of the Firebase console interface. On the left, a sidebar shows navigation links for Authentication, Cloud Firestore (which is highlighted with a red oval), Realtime Database, Storage, Hosting, Functions, and Machine Learning. The main content area is titled 'Cloud Firestore' and describes it as providing real-time updates, efficient queries, and automatic scaling. A prominent 'Crear base de datos' (Create database) button is centered in the middle of the page, also highlighted with a red oval. To the right of the text, there's an illustration of a stack of three servers with a magnifying glass focusing on the top one, which displays the Firestore logo.

No confundir con **storage**, que es otro servicio de Firebase dedicado a ofrecer almacenamiento de archivos.



# Firestore: creando un store- inicio

Desde el mismo panel podemos crear una base de datos. Por ahora, elegiremos el **modo de prueba** (sin control de acceso):

The screenshot shows the 'Crear base de datos' (Create database) dialog in the Firebase console. It has two steps: 'Reglas de seguridad para Cloud Firestore' (Step 1) and 'Define la ubicación de Cloud Firestore' (Step 2). Step 1 is active, showing the following content:

Tras definir la estructura de los datos, tendrás que escribir reglas para protegerlos.  
[Más información](#)

Iniciar en modo de producción  
Tus datos serán privados de forma predeterminada. Solo se concederá acceso de lectura y escritura de cliente según se indique en tus reglas de seguridad.

Iniciar en modo de prueba  
De forma predeterminada, se podrá acceder libremente a tus datos para agilizar la configuración. Si no actualizas las reglas de seguridad, dejarán de concederse los accesos de lectura y escritura de cliente dentro de 30 días.

rules\_version = '2';  
service cloud.firestore {  
 match /databases/{database}/documents {  
 match /{document=\*\*} {  
 allow read, write: if  
 request.time < timestamp.date(2020, 9, 29);  
 }  
 }  
}  
  
1 Cualquier usuario que tenga la referencia de tu base de datos podrá ver, editar y eliminar todos los datos de la base de datos durante 30 días.

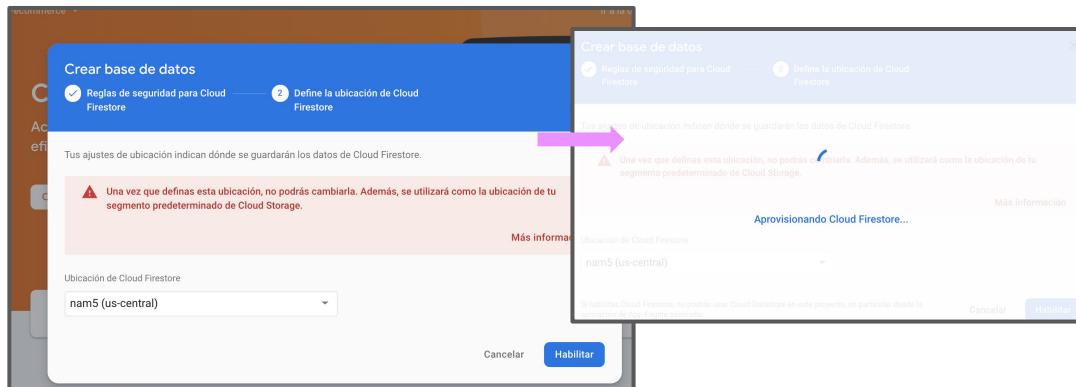
Cancelar Siguiente

Es una opción fácil para desarrollar rápido, aunque debes considerar que la base estará expuesta por 30 días, o hasta que restrinjas el acceso como indica.

# Firestore: creando un store-datacenter (DC)



Usualmente viene preconfigurado para DC en us-central, y está bien por ahora ya que es un DC multi-región.



Los DC impactan en los costos efectivos, verifica estos specs en la sección de **pricing**.

# Firestore: creando una colección



Almacenaremos distintos datos en distintas colecciones con sus respectivos campos. ¡Presiona **iniciar colección** y empecemos!

A screenshot of the Cloud Firestore interface. On the left is a sidebar with icons for Home, Settings, Users, and other database-related functions. The main area shows a table with one row. The first column contains a home icon and the text 'coderhouse-ecommerce'. The second column is empty. At the bottom of the table is a blue button labeled '+ Iniciar colección'. The top navigation bar shows the project name 'coderhouse-ecommerce' and links for 'Ir a la documentación' and tabs for 'Datos', 'Reglas', 'Índices', and 'Uso'.

Los datos se almacenan en colecciones y cada una define un tipo de entidad distinta.

# Firestone: configurando una colección



Almacenaremos distintos datos en distintas colecciones, con sus respectivos campos. ¡Presiona **iniciar colección** y empecemos!

The image shows two side-by-side screenshots of the Cloud Firestore interface. On the left, a blue dialog box titled 'Iniciar una colección' (Create Collection) is displayed. It has two steps: '1 Proporciona un ID a la colección' (Provide a collection ID) and '2 Añadir el primer documento' (Add the first document). The 'ID de colección' field contains 'items'. On the right, another blue dialog box titled 'Iniciar una colección' is shown, with a pink arrow pointing from the left dialog to it. This second dialog also has steps 1 and 2. Step 1 is checked with the note 'Proporciona un ID a la colección'. Step 2 says 'Añadir el primer documento'. Below these steps, the 'Ruta principal del documento' is set to '/items'. Under 'ID del documento', there is a field labeled 'ID automático'. At the bottom, there are 'Cancelar' and 'Guardar' buttons.

Recuerda poner atención al tipo de dato que brindas a cada campo para que tenga sentido con el dato que almacena, ante la duda podes recurrir a un string ;)

# Firestore: creando el primer documento



Crea los campos que contengan la información para los ítems de tu **ItemList** alojada en "/".

Puedes **autogenerar** ids, así que no es necesario que crees un campo específico id, y usar el provisto por firestore da beneficios de tolerancia a alta concurrencia.

No te preocupes si todavía no sabes todos los campos que quieres implementar a futuro. Firestore te dejará crear, agregar o quitarlos como quieras.

The screenshot shows the Firestore console's "Create Document" dialog. The document ID is "iEuV6tEihe7345pKDL7A". The data fields are:

Campo	Tipo	Valor
title	= string	Gorro Yokai
description	= string	Gorro altamente...
price	= number	350
imageId	= string	my_picture.jpg
categoryId	= string	gorros
stock	= number	20

Buttons at the bottom right include "Cancelar" and "Guardar". A pink arrow points to the document ID input field.

# Firestore



¡Una vez logrado, tendrás tu primer colección de firestore creada!

A screenshot of the Firestore console interface. The top navigation bar shows the path: Home > items > iEuV6tEihe7345... The left sidebar has a 'coderhouse-eCommerce' project selected, with 'items' listed under it. The main area displays a single document with the ID 'iEuV6tEihe7345pKDL7A'. The document contains the following fields and values:

categoryId:	"gorros"
description:	"Gorro altamente durable resistente a manchas y de talle flexible"
imageId:	"my_picture.jpg"
price:	350
stock:	20
title:	"Gorro Yokai"

No estamos muy lejos de poder acceder a la información, ¡sigue así!



# Crea tu Item Collection

Configura tu cuenta de Firebase, y crea un cloud Firestore con una nueva colección de items.

Duración: **15 minutos**



## ACTIVIDAD EN CLASE

# Crea tu item collection

### ¡A practicar!

Configura tu cuenta de Firebase, y crea un cloud Firestore con una nueva colección de items, con los siguientes atributos (como mínimo): categoryId, title, description, image, price, stock.

Extra-mile: si estás optando por el challenge-extra opcional, crea también tu colección de categorías dinámicas (id, key y nombre). Cuentas con 15 minutos para resolver la actividad.



# Break

¡10 minutos y volvemos!

¿Sabías que  
premiamos a nuestros estudiantes  
por su dedicación?

Conoce los [beneficios](#) del Top 10

# Configurando nuestra app

# Firestore: checklist

- Te has registrado en Firebase.
- Has creado un nuevo proyecto.
- Creación de un **Firestore**.
- Inicialización de la primera colección (items).
- Create tu primer item en la colección.

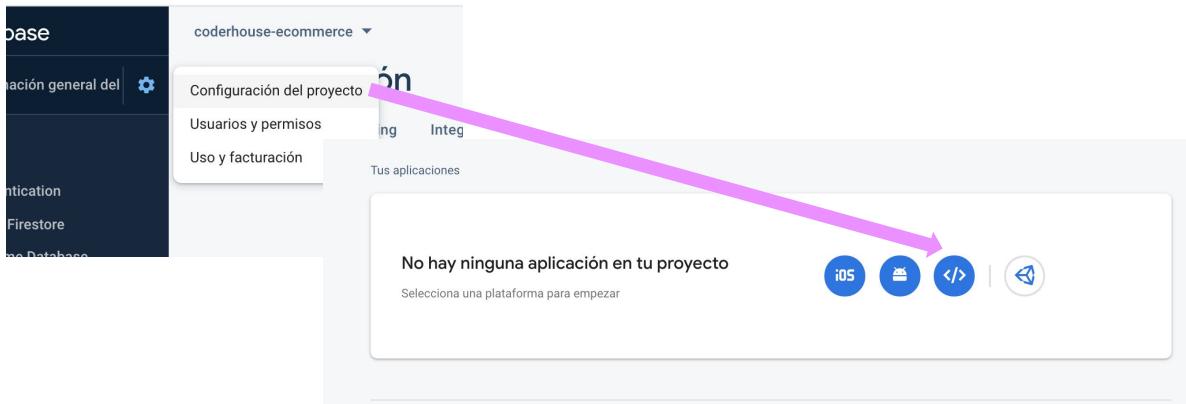
# Firestore: checklist para conectar nuestra react- app



- Configura tu integración con una webapp.
- Instalar firebase en el proyecto vía **NPM**.
- Configurar el client con tu api key.
- Configurar un client factory.

# Firebase: configura tu integración con la web app

Busca tus configuraciones de aplicación:



Agrega una integración de tipo web a tu APP.

# Firebase: configura tu integración con la web app

Crea tu web app en tu proyecto:

The image shows two screenshots from the Firebase console. The left screenshot, titled 'Añadir Firebase a tu aplicación web', shows step 1 'Registrar la aplicación' where the app name 'tu-apodo-de-app' is entered. Step 2 'Añadir SDK de Firebase' is shown below. A large pink arrow points from this step to the right screenshot. The right screenshot, titled 'Añadir SDK de Firebase', shows the generated JavaScript code for initializing the Firebase SDK. The code includes configuration for the apiKey, authDomain, databaseURL, projectId, storageBucket, messagingSenderId, and appId.

```
<!-- The core Firebase JS SDK is always required and must be listed first -->
<script src="https://www.gstatic.com/firebasejs/7.19.1.firebaseio-app.js"></script>

<!-- TODO: Add SDKs for Firebase products that you want to use
      https://firebase.google.com/docs/web/setup#available-libraries -->

<script>
  // Your web app's Firebase configuration
  var firebaseConfig = {
    apiKey: "AIzaSy_UHWW9n6zppEqD5_CnvRTfa",
    authDomain: "coderhouse-commerce.firebaseioapp.com",
    databaseURL: "https://coderhouse-commerce.firebaseio.com",
    projectId: "coderhouse-commerce",
    storageBucket: "coderhouse-commerce.appspot.com",
    messagingSenderId: "9171",
    appId: "1:web:67c6618b276d70c6410ed6"
  };
  // Initialize Firebase
  firebase.initializeApp(firebaseConfig);
</script>
```

Y obtendrás tus “api keys” para acceder al servicio.

# Firebase: instalación local

Instala el cliente de firebase vía **NPM**:



```
npm install firebase
```

Puedes hacerlo como vimos en la clase 2, usando el terminal dentro del proyecto.

# Inicializa Firebase

# Inicializa firebase

Inicia la **firebase** en el punto de entrada de tu aplicación de React, que es **index.js**

```
import { StrictMode } from "react";
import ReactDOM from "react-dom";
import { initializeApp } from "firebase/app";
import App from "./App";

const firebaseConfig = {
  apiKey: "AIzaSyB7S_Fyxmp1C7ZUJU430PydfDI0EsaLVLA",
  authDomain: "moles-ecommerce.firebaseio.com",
  projectId: "moles-ecommerce",
  storageBucket: "moles-ecommerce.appspot.com",
  messagingSenderId: "850435110368",
  appId: "1:850435110368:web:4324547d9f4886cbdd6785"
};

initializeApp(firebaseConfig);

const rootElement = document.getElementById("root");
ReactDOM.render(
  <StrictMode>
    <App />
  </StrictMode>,
  rootElement
);
```

# Accede a un documento

# Firebase: consulta un documento

Importa firestore desde nuestro factory.



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

1. Configura a qué **colección** accederás con el segundo parámetro de **doc()**
2. Configura qué **documento** buscas con el tercer parámetro de **doc()**

¡Haz tu query contra **firestore**!

```
import { doc, getDoc, getFiresore } from "firebase/firestore";  
// ...  
  
useEffect(() => {  
  const db = getFirestore();  
  
  const biciRef = doc(db, "items", "Uyho0HTTsC40A0IVxXyA");  
  getDoc(biciRef).then((snapshot) => {  
    if (snapshot.exists()) {  
      setProduct({ id: snapshot.id, ...snapshot.data() });  
    }  
  });  
}, []);
```

Y tiene un método **.data()** que nos da el contenido que buscamos

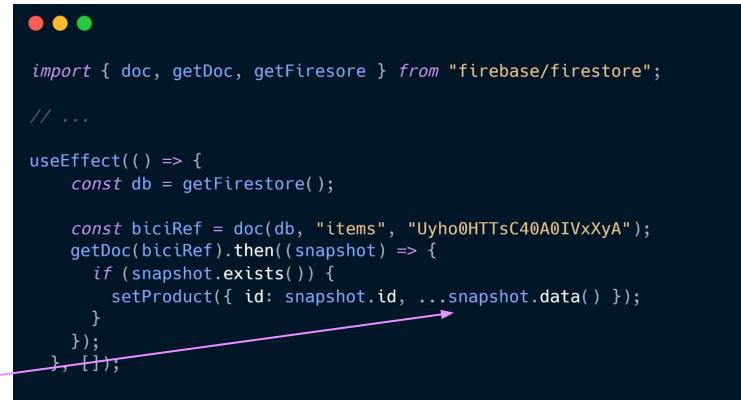
# Firebase: contenido de la respuesta

Por el diseño, en **firestore** el id **no es parte del documento**

Cada consulta a firestore nos devolverá un objeto que es la referencia de firestore que adentro tiene los siguientes campos:

- id, ref, metadata

Y tiene un método **.data()** que nos da el contenido que buscamos



```
import { doc, getDoc, getFirestore } from "firebase/firestore";
// ...
useEffect(() => {
  const db = getFirestore();

  const biciRef = doc(db, "items", "Uyho0HTTsC40A0IVxXyA");
  getDoc(biciRef).then((snapshot) => {
    if (snapshot.exists()) {
      setProduct({ id: snapshot.id, ...snapshot.data() });
    }
  });
}, []);
```

# Firebase: contenido de la respuesta

Si necesitas algo de info extra de esta consulta la puedes buscar en esa respuesta de tipo **DocumentSnapshot**

Puedes combinar el resultado con el id si después lo necesitas para algo realizando lo siguiente:

```
{ id: doc.id, ...doc.data() }
```



```
import { doc, getDoc, getFirestore } from "firebase/firestore";  
  
// ...  
  
useEffect(() => {  
  const db = getFirestore();  
  
  const biciRef = doc(db, "items", "Uyho0HTTsC40A0IVxYxA");  
  getDoc(biciRef).then((snapshot) => {  
    if (snapshot.exists()) {  
      setProduct({ id: snapshot.id, ...snapshot.data() });  
    }  
  });  
}, [ ]);
```

# Accede a una colección

# Firebase: consulta una colección sin filtros

¡Haz tu query contra **firestore**!

```
import { collection, getDocs, getFirestore } from "firebase/firestore";  
  
// ...  
  
useEffect(() => {  
  const db = getFirestore();  
  
  const itemsCollection = collection(db, "items");  
  getDocs(itemsCollection).then((snapshot) => {  
    setProducts(snapshot.docs.map((doc) => ({ id: doc.id, ...doc.data() })));  
  });  
}, []);
```

# Firebase: consulta una colección sin filtros

¡Haz tu query contra **firestore**!

```
import { collection, getDocs, getFirestore } from "firebase/firestore";
// ...
useEffect(() => {
  const db = getFirestore();

  const itemsCollection = collection(db, "items");
  getDocs(itemsCollection).then((snapshot) => {
    if (snapshot.size === 0) {
      console.log("No results");
    }
    setProducts(snapshot.docs.map((doc) => ({ id: doc.id, ...doc.data() })));
  }, []);
}, []);
```

1. Inicializa el cliente
2. Setea a la colección
3. Invoca el llamado
4. Accede al snapshot (resultado)
5. (opcional) verifica su tamaño con **.size**
6. Accede al set de datos mediante el accesror **querySnapshot.docs**
7. Persiste en algún estado la respuesta

# Firebase: consulta una colección con filtros

¡Haz tu query contra **firestore**!

Implementa un filtro simple sobre la colección utilizando el método **query()** y **where()**

```
import { collection, getDocs, getFirestore, query, where } from "firebase/firestore";
// ...

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

  const q = query(collection(db, "items"), where("price", ">", 100));
  getDocs(q).then((snapshot) => {
    if (snapshot.size === 0) {
      console.log("No results");
    }
    setProducts(snapshot.docs.map((doc) => ({ id: doc.id, ...doc.data() })));
  }, []);
}, []);
```

# Firebase: consulta una colección con filtros

Implementa un filtro combinado utilizando **where()** las veces que necesites en los parámetros de la función **query()**

```
import { collection, getDocs, getFirestore, query, where } from "firebase/firestore";  
// ...  
  
useEffect(() => {  
  const db = getFirestore();  
  
  const q = query(  
    collection(db, "items"),  
    where("price", ">", 100),  
    where("category", "==", "tecnology")  
  );  
  getDocs(q).then((snapshot) => {  
    if (snapshot.size === 0) {  
      console.log("No results");  
    }  
    setProducts(snapshot.docs.map((doc) => ({ id: doc.id, ...doc.data() })));  
  });  
}, []);
```

# Firebase: aplica un límite de resultados

Un query masivo puede ser poco ideal, pon un límite de acuerdo a tus caso de uso

Agrega **limit()** para poner un tope de resultados

Una combinación de size y las referencias de documentos que vimos en query a un documento te servirán luego para cuando necesites paginar tus resultados.



```
import { collection, getDocs, getFiresore, limit, query, where } from "firebase/firestore";  
// ...  
  
useEffect(() => {  
  const db = getFirestore();  
  
  const q = query(  
    collection(db, "items"),  
    where("price", ">", 100),  
    limit(1)  
  );  
  getDocs(q).then((snapshot) => {  
    if (snapshot.size === 0) {  
      console.log("No results");  
    }  
    setProducts(snapshot.docs.map((doc) => ({ id: doc.id, ...doc.data() })));  
  });  
}, []);
```



# Ejemplo en vivo

Vamos al código.



# #Coderalert

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



Actividad N° 7

Proyecto Final

# Item Collection I

## Descripción de la actividad.

- ✓ Conecta tu nueva ItemCollection de google Firestore a tu ItemListContainer y ItemDetailContainer

## Recomendaciones

- ✓ Conecta tu colección de firestore con el listado de ítems y con el detalle de ítem.
- ✓ Elimina los async mocks (promises) y reemplazalos por los llamados de Firestore.
- ✓ Si navegas a /item/:id, debe ocurrir una consulta de (1) documento.
- ✓ Si navegas al catálogo, debes consultar (N) documentos con un query filtrado, implementando la lógica de categorías y obteniendo el id de categoría del parámetro de react-router :categoryId.

**¿Preguntas?**

# Resumen de la clase hoy

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

**Opina y valora  
esta clase**

Muchas gracias.

#DemocratizandoLaEducación