wawiwa
React
2022 © Wawiwa Tech | Confidential

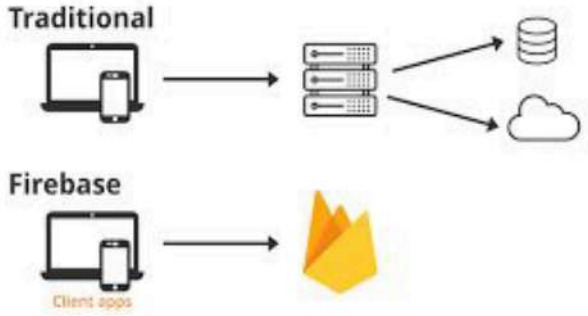

wawiwa
24 - Firebase
visual design
2022 © Wawiwa Tech | Confidential

wawiwa

Introducción a Firebase

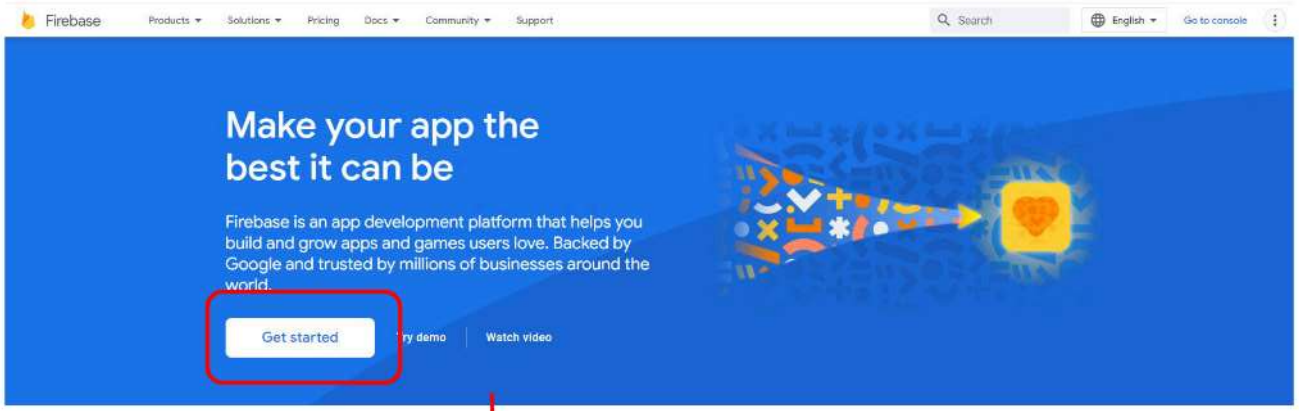
Firebase es una base de datos NoSQL en tiempo real en la nube que te ayuda a crear aplicaciones sin tener que crear el backend.
Puede guardar y traer objetos JSON, crear autenticación de usuario y obtener actualizaciones de datos en tiempo real en todos los dispositivos conectados.

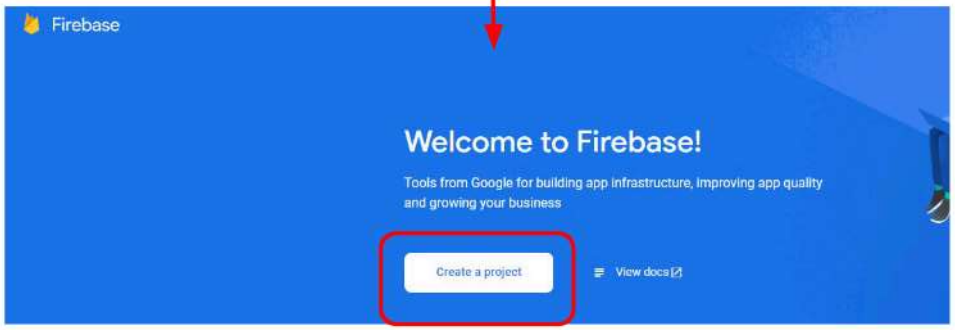
El plan gratuito de Firebase está limitado a 50 conexiones y 100 MB de almacenamiento.



wawiwa

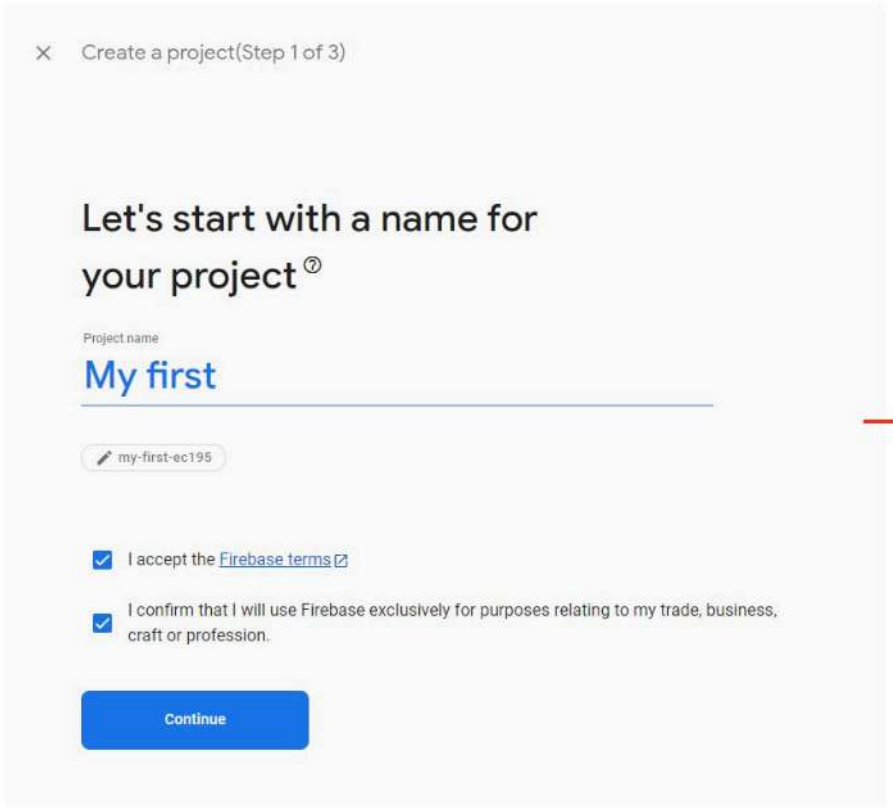
Crea un nuevo Proyecto

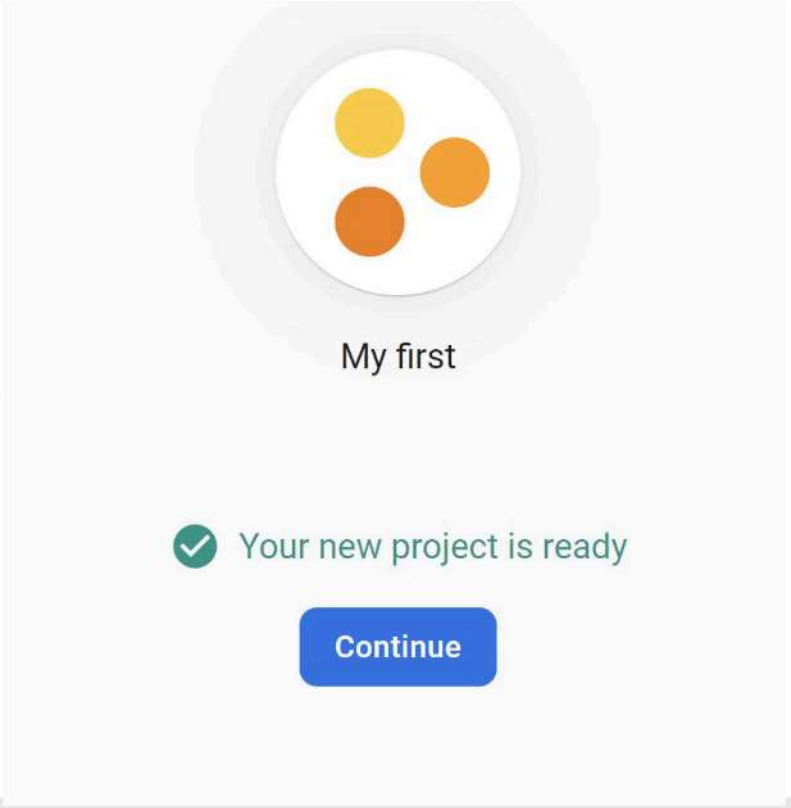




wawiwa

Create un nuevo Proyecto



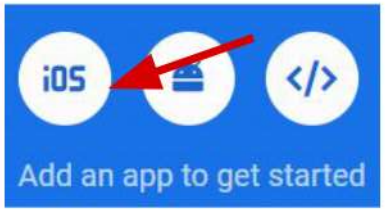


wawiwa

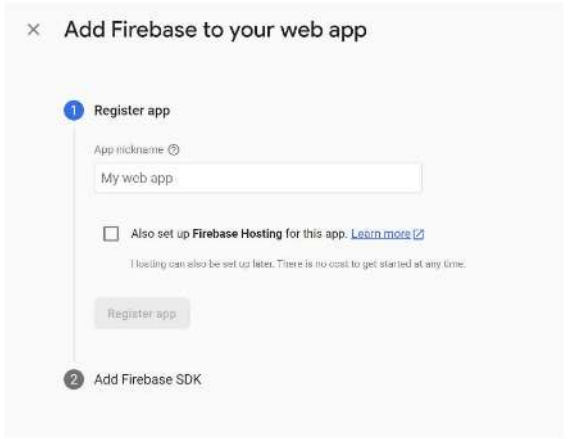
Conectando a tu Aplicación Web

Ahora necesitamos conectar el proyecto Firebase a nuestra aplicación web.

Da Click a la opción de la derecha:



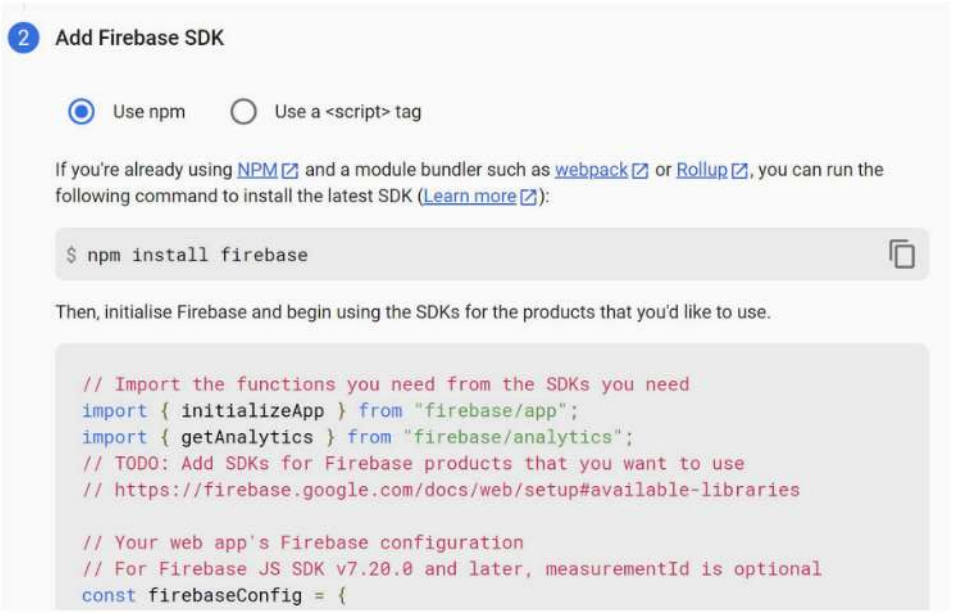
Ingresa el nombre de la app:



wawiwa

Conectando a tu aplicación Web

Corre: **npm i firebase**



wawiwa

Agrega el archivo .env

Debemos agregar los pares key-value del objeto firebaseConfig en nuestra aplicación en React.
Sin embargo, debemos encubrir esta información usando un archivo .en.

Los archivos .env contienen credenciales en formato key-value para servicios usados en el programa que están construyendo. **Deben almacenarse localmente y no cargarse en repositorios de códigos en línea para que todos puedan leerlos**.

De otra manera, la configuración de nuestra base de datos de Firebase sería expuesta a cualquiera, incluyendo usuarios potencialmente maliciosos.

Afortunadamente, “create-react-app” viene automáticamente con dotenv, el paquete npm que usamos en Javascript para guardar keys de API sensibles en un archivo .env.

wawiwa

Agrega el archivo .env a .gitignore

Primero, debemos agregar .env a nuestro archivo .gitignore.

Tenga en cuenta que create-react-app agrega automáticamente varios de estos tipos de archivos a nuestro .gitignore, incluidos .env.local, .env.development.local, etc.

create-react-app hace esto porque en proyectos más grandes, puede resultar útil tener varios archivos para las variables de entorno. Se pueden dividir para pruebas, producción y desarrollo.

Como nuestra aplicación es pequeña, crearemos un archivo .env básico.
Agrega .env al .gitignore y luego confirma y envía el archivo .gitignore actualizado a GitHub.
No crees el archivo .env todavía. Como recordarás de JavaScript, si enviamos un archivo .gitignore actualizado al mismo tiempo que enviamos el archivo que debe ignorarse, GitHub no sabrá que debe ignorarlo, lo que significa que se agregará al repositorio.

wawiwa

Agrega el archivo .env a .gitignore

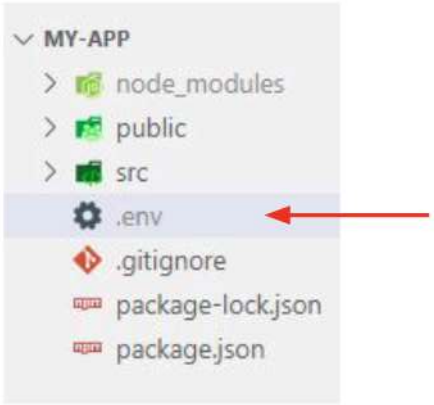
1. **Agrega el archivo .env a .gitignore.**

- git add .
- git commit -m "update gitignore"
- git push origin main

wawiwa

Crea el archivo .env

2. Lo siguiente es crear un archivo .env en el directorio del proyecto:



wawiwa

Configura las variables en el archivo .env

3. Lo siguiente es configurar las variables en el archivo .env

Las variables “environment” pueden ser configuradas solamente a strings, no objetos. Por esa razón cada par de key-value en el objeto firebaseConfig necesita ser descompuesto en sus propias constantes así:

```
REACT_APP_FIREBASE_API_KEY = "YOUR-UNIQUE-CREDENTIALS"
REACT_APP_FIREBASE_AUTH_DOMAIN = "YOUR-PROJECT-NAME.firebaseapp.com"
REACT_APP_FIREBASE_MEASUREMENT_ID = "YOUR-MEASUREMENT-ID"
REACT_APP_FIREBASE_PROJECT_ID = "YOUR-PROJECT-FIREBASE-PROJECT-ID"
REACT_APP_FIREBASE_STORAGE_BUCKET = "YOUR-PROJECT-NAME.appspot.com"
REACT_APP_FIREBASE_MESSAGING_SENDER_ID = "YOUR-PROJECT-SENDER-ID"
REACT_APP_FIREBASE_APP_ID = "YOUR-PROJECT-APP-ID"
```

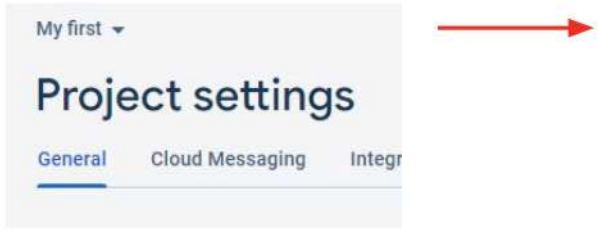
Nota: Es muy importante que cada variable de entorno en su aplicación esté precedida por REACT_APP. De lo contrario, la variable de entorno no funcionará. Esta es una protección implementada por create-react-app para garantizar que las variables de entorno sensibles no queden expuestas accidentalmente en nuestras aplicaciones.

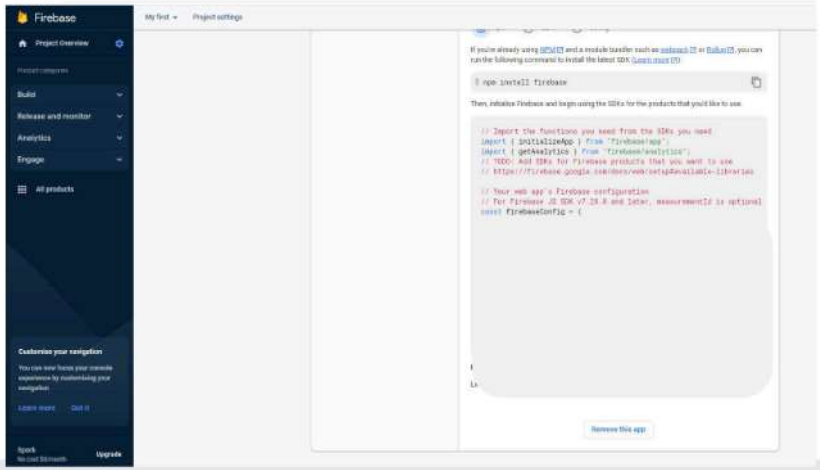
wawiwa

Configura las variables en el archivo .env

3. Lo siguiente es configurar las variables en el archivo .env

Reemplaza los placeholders en los valores anteriores con el valor de cada clave de tu propia aplicación Firebase. (Si perdiste esta información, haz click en el engranaje en la parte superior izquierda de la página, haz click en configuración del proyecto y desplácese hasta la parte inferior de la página).





wawiwa

Crea el archivo de configuración de referencia a Firebase

4. Crea el archivo de configuración de referencia a la base de datos

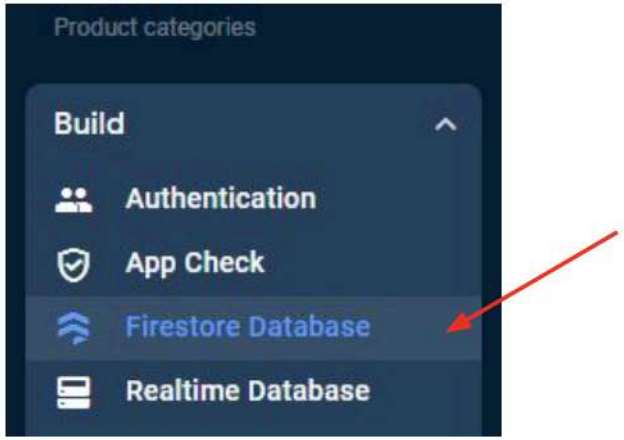
- Crea un archivo en el directorio “src” llamado “firebase.js”.
- Agrega el siguiente código:

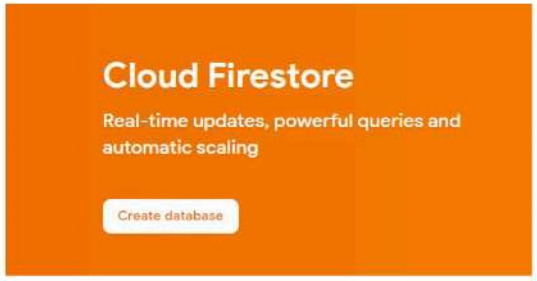
```
import { initializeApp } from "firebase/app";

const firebaseConfig = {
    apiKey: process.env.REACT_APP_FIREBASE_API_KEY,
    authDomain: process.env.REACT_APP_FIREBASE_AUTH_DOMAIN,
    projectId: process.env.REACT_APP_FIREBASE_PROJECT_ID,
    storageBucket: process.env.REACT_APP_FIREBASE_STORAGE_BUCKET,
    messagingSenderId: process.env.REACT_APP_FIREBASE_SENDER_ID,
    appId: process.env.REACT_APP_FIREBASE_APP_ID,
    measurementId: process.env.REACT_APP_FIREBASE_MEASUREMENT_ID
}

const app = initializeApp(firebaseConfig);
```

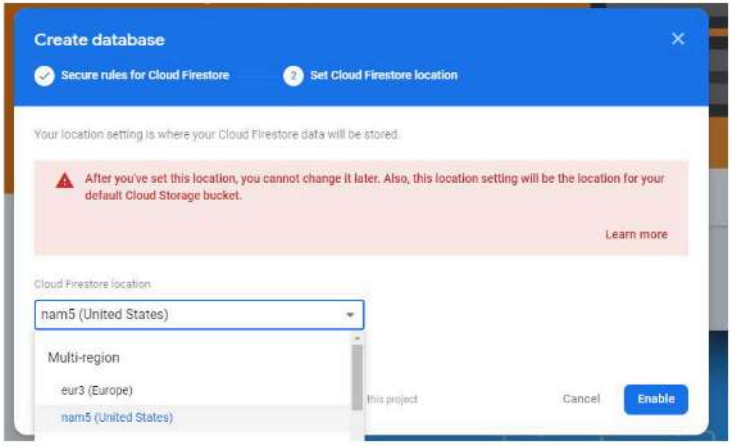
wawiwa

Configura la base de datos de Firestore

1. Da click en “Firestore Database”
2. Da click en el botón “Create database”
3. Usa el “production mode” y da click en “Next”
4. Escoge la ubicación más cerca de ti. Esto va a ser donde tu servidor estará ubicado. Después da click en “Enable”.





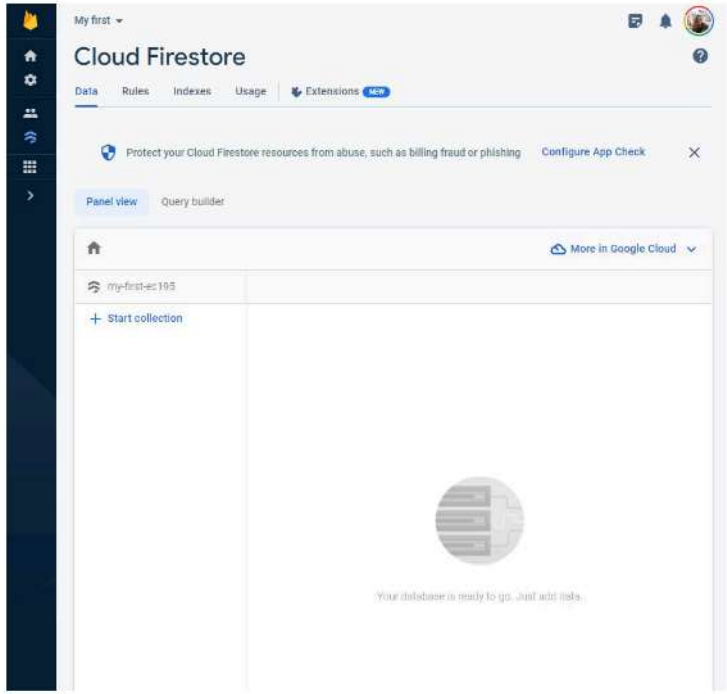


wawiwa

Firestore

Podemos ver que nuestra base de datos se terminó de crear.

- Firestore es una base de datos NoSql (como mongoDB).
- Firestore tiene colecciones - similar a tablas.



wawiwa

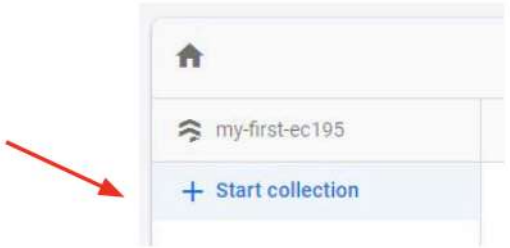
Firestore - Reglas

Las reglas determinan que va a pasar en la base de datos.

En nuestro caso, tenemos que permitir que nuestro sitio web lea y escriba de la base de datos (por ejemplo agregar datos a la base de datos).
Lo único que tenemos que hacer es cambiar “false” a “true”.
Después hacer click en “Publish”.

wawiwa

Firestore - Crea una Colección

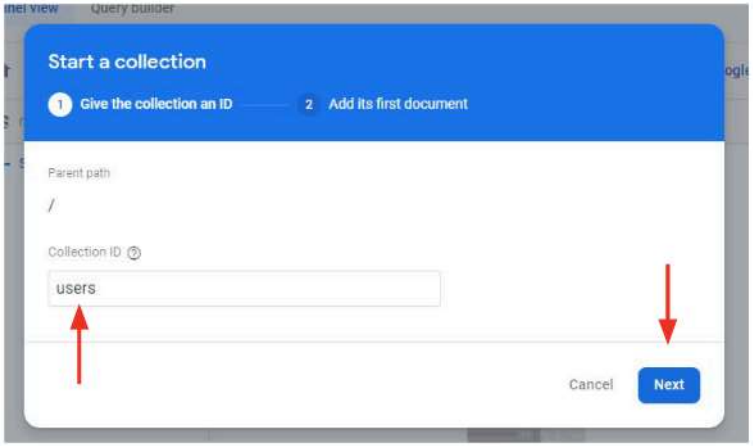


Para crear una colección:

1. Da click en “Start collections”.
2. Dale un nombre.

   En este ejemplo vamos a crear una colección de “users”.

1. Da click en “Next” para continuar.

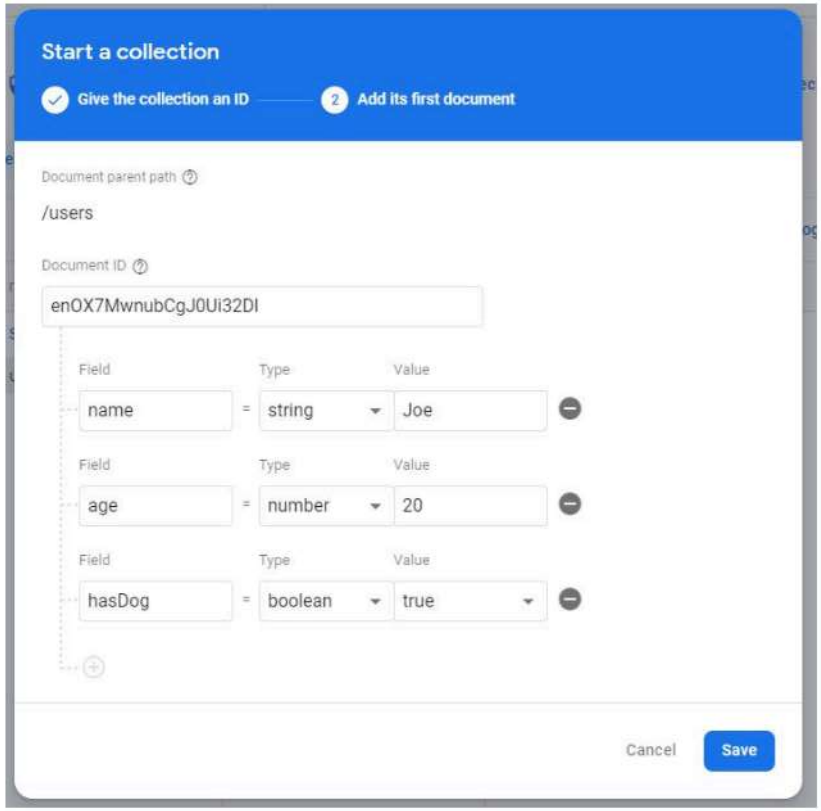


wawiwa

Firestore - Crea una colección

- Una colección va a incluir múltiples documentos.
- Cadah **documento es un "user"**.

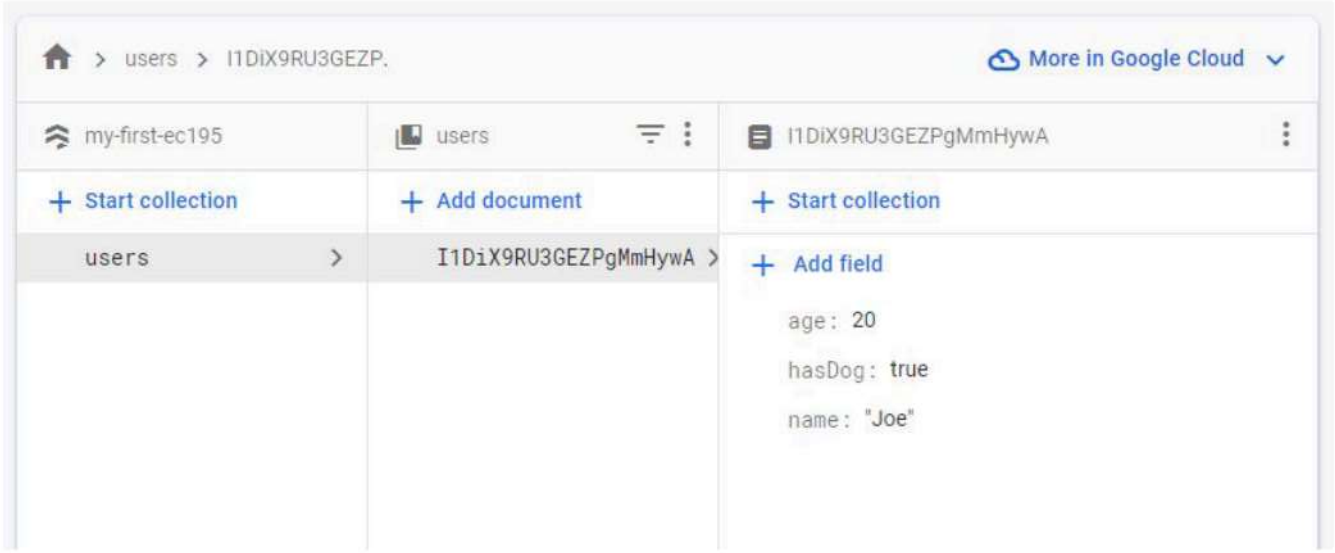
1. Marca el "Document ID" con Auto-ID.
2. Crea un documento de ejemplo para que nuestra colección incluya ya un usuario.
3. Cada campo contenerá un atributo del "user".
4. Cuando termines, haz click en "Save".



wawiwa

Firestore - Crea una Colección

Ahora nuestra base de datos de firestore incluye una colección de “users”.
Tiene un solo documento adentro.



wawiwa

Usa la nube de Firestore (base de datos)

Después de que nuestra base de datos esté lista para ser usada, la vamos a conectar a nuestra app.

Para inicializar la base de datos dentro de nuestro proyecto, en firebase.js:

```
1. import { getFirestore } from "firebase/firestore";
2. export const db = getFirestore(app);
```

wawiwa

Obten los “users”

En nuestro componente, vamos a mandar una solicitud para obtener los “users” y presentarlos.

- Para guardar los “users”, crea un estado:

```
const [users, setUsers] = useState([]);
```

- Para buscar la información, debemos usar el hook useEffect():

```
useEffect(() => {

},[]);
```

- No olvides importar los dos hooks:

```
import React, { useState, useEffect } from 'react';
```

wawiwa

Función “Async”

Para obtener la información debemos crear una función “async”.

Una función “async” es una función de programación que permite que función asíncronas sucedan.

Las operaciones asíncronas son las que pueden correr en el fondo mientras el resto del programa corre.

wawiwa

Obten los “users”

Importa el siguiente código:

```
import { db } from './firebase';
import { collection, getDocs } from "firebase/firestore";


  const usersCollectionRef = collection(db, "users");

  const getUsers = async () => {
    const data = await getDocs(usersCollectionRef);
    console.log(data);
  }

  useEffect(() => {
    getUsers();
  },[]);
```

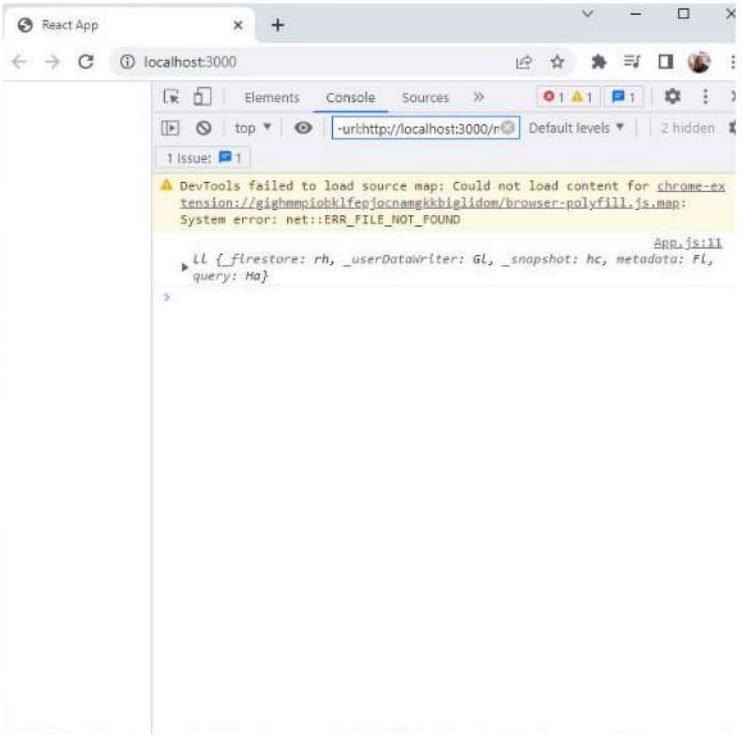
El método **collection()** de Firestore se usa para crear, acceder y manipular colecciones en tu base de datos. Es un método del objeto Firestore, que se usa para interactuar con tu base de datos de Firestore.

- Creamos una función porque no podemos hacer que el hook useEffect sea una función async.
- Usamos la clave "await" para esperar a la operación de complete antes de continuar. Si no usamos “await”, el efecto puede no comportarse como se espera.

wawiwa

Obten los “users”

Los “users” que obtenemos están escondidos:

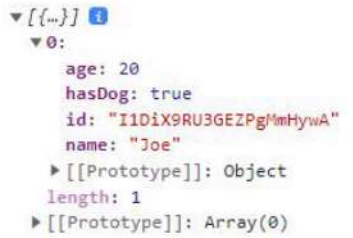


wawiwa

Obten los “users”

Eciste una función que nos permite manejar información fácilmente:

```
setUsers(data.docs.map((doc) => ({ ...doc.data(), id: doc.id })));
```



Esta línea de código utiliza el método map() para transformar una serie de instantáneas de documentos devueltas por una consulta de Firestore en una serie de objetos JavaScript simples que incluyen tanto los datos como el ID único de cada documento. Aquí hay un desglose de lo que está sucediendo:

1. **data.docs:** Este es un arreglo de instancias de los documentos retornados por una solicitud a Firestore. Cada instancia contiene la información de cada documento y un ID único que se usa para identificar el documento en la base de datos de Firestore.
2. **map():** El método map() se usa para transformar cada instancia del documento del arreglo en un nuevo objeto.
3. **(doc) => ({ ...doc.data(), id: doc.id }):** Es una función de flecha que toma una instancia del documento como argumento y retorna un nuevo objeto que incluye la información y ID del documento. El operador de propagación (...) es usada para copiar los campos de la información del objeto del documento a un nuevo documento, y el ID es agregado como un campo diferente.

wawiwa

Muestra los “Users”

Para mostrar una lista de “users”, usa el método map():

```
return (
  <div>
    {users.map((user) => {
      return (
        <div>
          {user.name} is {user.age} years old and he {user.hasDog ? "has" : "hasn't"} a dog
        </div>
      )
    })}
  </div>
);
```

wawiwa

Agrega un documento en Firestore

Para un solo documento a una colección en Firestore, usa el método **addDoc**:

- Importalo de Firebase:

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

- Crea una referencia a la colección a la que quieres agregar un documento:

```
const usersRef = collection(db, "users");
```

- Pas un objeto representando la información que quieres guardar como un argumento a la función **addDoc**.

  Por ejemplo:

```
  await addDoc(usersRef, {
    name: "John Doe",
    age: 30,
    email: "johndoe@example.com"
});
```

wawiwa

Agrega un documento a la Colección

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

const nameRef = useRef();
  const ageRef = useRef();
  const hasDogRef = useRef();

  const addUser = async () => {
    await addDoc(usersCollectionRef, {
      name: nameRef.current.value,
      age: Number(ageRef.current.value),
      hasDog: hasDogRef.current.checked
    });
  }
```


```
<input placeholder='Name' ref={nameRef}/>
<input placeholder='Age' ref={ageRef} />
<input type='checkbox' ref={hasDogRef} />
Has a dog<button onClick={addUser}>Add User</button>
```

wawiwa

Método Doc

En Firebase, el método doc() es usado para obtener una referencia a un documento específico de una colección de Firestore.

Retorna un objeto DocumentReference que representa la ubicación del documento en la base de datos.

Para usar éste método, impórtalot:

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

wawiwa

Obtén un documento de una Colección

Para obtener un documento de una colección en Firestore, usa los métodos **doc** y **getDoc**:

1. Importa de Firebase:

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

2. Crea una referencia al documento que quieres obtener usando la función doc.
   Por ejemplo, si tienes una colección llamada "users" y un documento con ID "abc123", puedes crear una referencia así:

```
const userRef = doc(db, "users", "abc123");
```

3. Pasa la referencia del “user que quieres obtener como un argumento en la función getDoc:
   Por ejemplo:

```
await getDoc(userRef);
```

wawiwa

Obtén un documento de una Colección

Un ejemplo:

```
const userRef = doc(db, "users", "423324234234234");
const res = await getDoc(userRef);
const user = res.data();
console.log("user");
console.log(user);
```

```
user
{firstName: 'Super', birthDate: 'Sat, 01 Jan 2000 17:13:02 GMT', la
n@gmail.com'}
  birthDate: "Sat, 01 Jan 2000 17:13:02 GMT"
  email: "superman@gmail.com"
  firstName: "Super"
  lastName: "Man"
  password: "123456"
▶ [[Prototype]]: Object
```

wawiwa

Actualiza un documento en Firestore

Para actualizar un documento en una colección en Firestore, use los métodos **doc** y **updateDoc**:

1. Importa de Firebase:

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

2. Crea una referencia al documento que quieres actualizar usando la función “doc”.
   Por ejemplo, si tienes una colección llamada “users” y un documento con ID "abc123", puedes crear a un referencia así:

```
const userRef = doc(db, "users", "abc123");
```

3. Pasa un objeto representado de los campos que quieres actualizar como argumento a esta función.
   Por ejemplo:

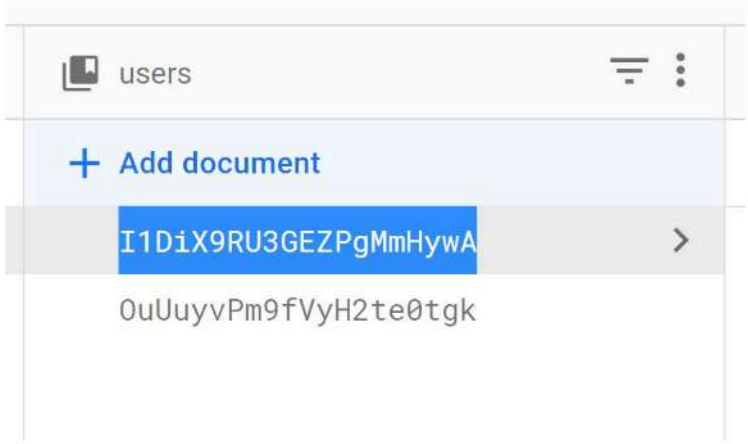
```
await updateDoc(userRef, {
  name: "Jane Doe",
  age: 35
});
```

wawiwa

Actualiza un documento en una Colección

Un ejemplo de una actualización para nuestra app:

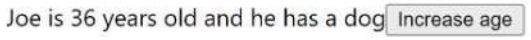
```
const updateUser = async() => {
  const userRef = doc(db, "users", "I1DiX9RU3GEZPgMmHywA");
  await updateDoc(userRef, {
    age: 35
  });
}
```



wawiwa

Ejercicio 23

Actualiza un documento en la colección:



- Escribe una función que incremente la edad de un ID dado
- Agrega un botón junto a cada usuario que incremente la edad del “user”

wawiwa

Borra un documento en Firestore

Para borrar un documento en una colección en Firestore, usa los métodose **doc** y **deleteDoc**:

1. Importa deFirebase:

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

1. Crea una referencia al documento que quieres borrar usando la función doc.
   Por ejemplo, si tienes una colección llamada “users” y un documento con ID "abc123", puedes crear una referencia así:

```
const userRef = doc(db, "users", "abc123");
```

1. Pasa la referencia del usuario que quieres borrar como argumento de la función deleteDoc:
   Por ejemplo:

```
await deleteDoc(userRef);
```

wawiwa

Ejercicio 24

Borra un documento en la colección:

- Escribe una función que borre un usuario por su ID
- Usa los métodos then() y catch() de un objeto de una promesa que calendarize una función que sea llamada cuando la promesa sea rechazada. Imprime en la consola si fue borrado exitosamente o si hubo un error:
- Agrega un botón junto a cada “user” que lo borre
- Haz Git add, commit y push a todos los cambios a tu repositorio en Github

wawiwa

Conclusión

Los documentos en Firestore sean guardados como pares key-value y cada documento es identificado por un ID único.

Firebase provee cuatro operaciones básicas CRUD (Create, Read, Update, Delete) para manipular información en su base de datos Firestore:

- addDoc - agrega un nuevo documento a una colección en Firestore
- getDoc - obten data de Firestore
- updateDoc - actualiza un documento existente en Firestore
- deleteDoc - borra un documento existente en Firestore

Para obtener una referencia a un documento específico en una colección de FireStore - usa el método **doc()**


wawiwa
Proyecto Final
visual design
2022 © Wawiwa Tech | Confidential