«Talento Tech»

Node JS

Clase 14

Buenos Aires Ciudad

Clase N° 14 - Datos en la nube

Temario:

1. Firebase
2. Firestore
    - Nuevo Proyecto
    - Creando una colección
3. Configuración de Firebase Firestore en tu Proyecto

Objetivos de la Clase

En esta clase, los estudiantes comprenderán el funcionamiento de Firebase Firestore como servicio de almacenamiento en la nube y su integración en aplicaciones web mediante una API. Aprenderán a configurar y estructurar una base de datos en Firestore, gestionar colecciones y documentos, y aplicar reglas de seguridad para un acceso seguro. Además, explorarán cómo consumir datos desde Firestore utilizando su SDK y consultas avanzadas, integrando esta funcionalidad dentro de una API propia. Finalmente, implementarán una arquitectura REST para estructurar y devolver los datos de manera eficiente, garantizando una comunicación clara y escalable entre el frontend y el backend.

Firebase

Firebase es una plataforma de desarrollo de aplicaciones creada por Google que proporciona una amplia gama de herramientas y servicios para facilitar la creación y escalabilidad de aplicaciones web y móviles. Su ecosistema incluye soluciones para autenticación, bases de datos en tiempo real, almacenamiento en la nube, hosting, funciones serverless y análisis, permitiendo a los desarrolladores enfocarse en la lógica de negocio sin preocuparse por la infraestructura.

Uno de los componentes más destacados de Firebase es su capacidad para ofrecer sincronización en tiempo real y escalabilidad, lo que lo hace ideal para aplicaciones colaborativas y sistemas que requieren una actualización constante de datos. Además, su integración con servicios como Google Cloud, su facilidad de configuración y su compatibilidad con múltiples plataformas lo convierten en una opción poderosa para desarrolladores que buscan eficiencia y rendimiento en sus proyectos.

Firestore, la base de datos de Firebase, es un claro ejemplo de la filosofía de la plataforma, ya que permite almacenar y gestionar datos de manera flexible y segura, facilitando la comunicación entre el frontend y el backend sin necesidad de configurar servidores complejos. Gracias a su arquitectura NoSQL basada en documentos y su capacidad de procesamiento eficiente de consultas, Firestore se ha convertido en una de las opciones más populares para el desarrollo de aplicaciones modernas.

Firestore

Firestore es una base de datos en la nube de Google, diseñada para manejar datos en tiempo real con una arquitectura NoSQL basada en documentos. A diferencia de bases de datos relacionales, Firestore estructura la información en colecciones y documentos, lo que permite una mayor flexibilidad en el almacenamiento y recuperación de datos. Su integración con Firebase facilita la sincronización de datos entre múltiples dispositivos y plataformas sin necesidad de configurar servidores complejos.

Uno de los principales beneficios de Firestore es su capacidad para manejar datos en tiempo real, permitiendo que las aplicaciones respondan instantáneamente a los cambios en la base de datos sin necesidad de realizar constantes peticiones al servidor. Además, Firestore cuenta con un sistema de seguridad basado en reglas que define quién puede leer o escribir en la base de datos, asegurando un control preciso sobre el acceso a la información.

En cuanto a su funcionamiento, los datos se organizan en colecciones, dentro de las cuales se almacenan documentos. Cada documento es un conjunto de pares clave-valor que pueden contener datos primitivos como cadenas, números y booleanos, así como estructuras más complejas como arreglos y mapas anidados. Firestore permite realizar consultas eficientes utilizando índices automáticos, lo que mejora el rendimiento en la búsqueda y filtrado de datos.

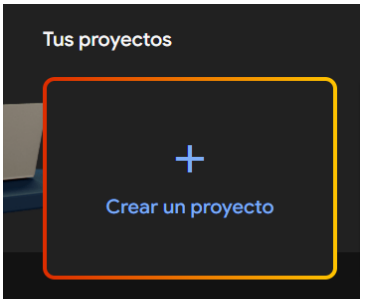
Para interactuar con Firestore, se utiliza el SDK de Firebase, que proporciona métodos para agregar, actualizar, eliminar y consultar documentos. Además, Firestore es compatible con peticiones REST y bibliotecas como Firebase Admin SDK, lo que permite su integración en entornos backend. A lo largo de esta clase, exploraremos cómo configurar Firestore, estructurar nuestra base de datos y consumir la información de manera eficiente dentro de una API.

Nuevo Proyecto

Para comenzar a trabajar con Firestore, debemos dirigirnos al sitio web de **Firebase** (https://firebase.google.com/) y loguearnos con una cuenta de google válida.

Una vez dentro debemos acceder al enlace “Go to console” disponible en el header del sitio.

Allí tendremos la oportunidad de crear un nuevo proyecto de **Firebase**, esto nos habilitará a utilizar todos los servicios que ofrece, incluido **Firestore** que almacena datos en la nube:

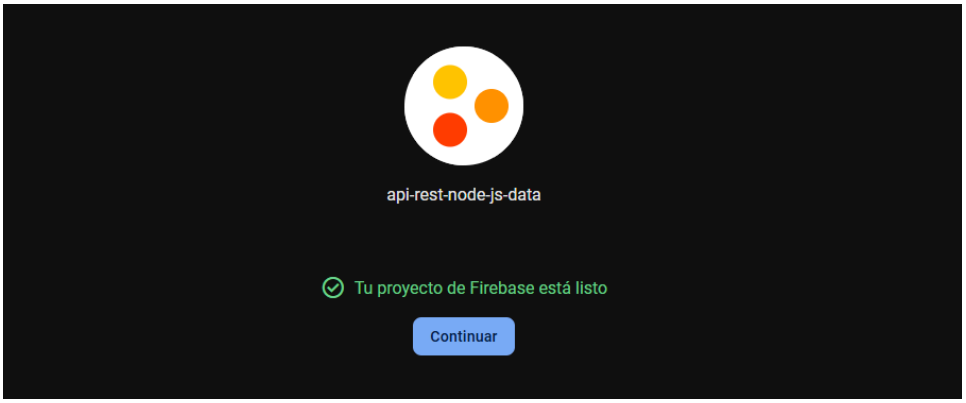


A partir de aquí nos solicitará que completemos la información necesaria para configurar nuestro proyecto, comenzando por el nombre, por ejemplo:

`api-rest-node-js-data`

Luego nos preguntará si deseamos habilitar la IA Gemini al proyecto y finalmente si queremos vincularlo con Google Analytics (podemos deshabilitar ambas opciones sin problema).

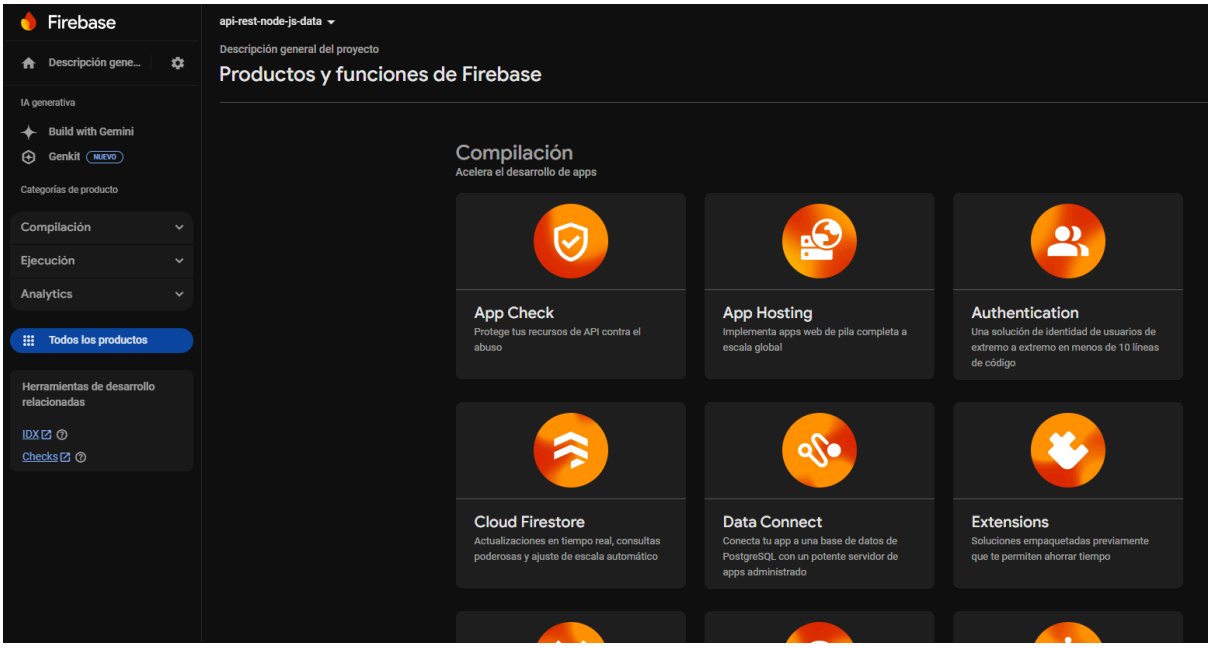
Una vez finalizado el proceso obtendremos el siguiente mensaje:



Ahora contamos con nuestro panel de control que nos da acceso a todas las herramientas ofrecidas por **Firebase**:

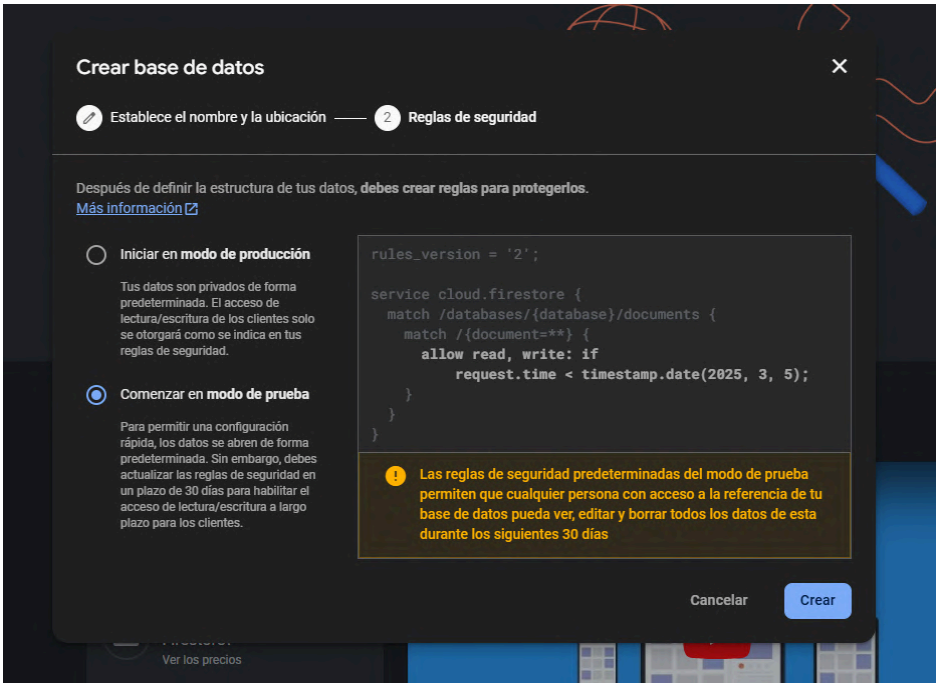
Una vez aquí toca crear un nuevo servicio de **Firestore**.

Desde el botón en el panel lateral izquierdo, seleccionamos la opción “**Todos los productos**” para acceder al listado completo de herramientas de Firebase:

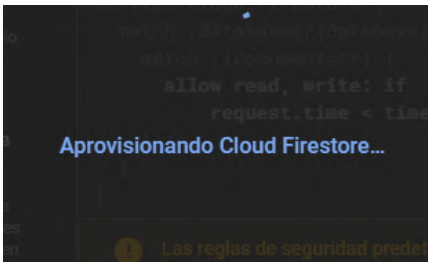


Ahora buscamos y presionamos sobre el módulo “Cloud Firestore” para crear una nueva instancia de este servicio.

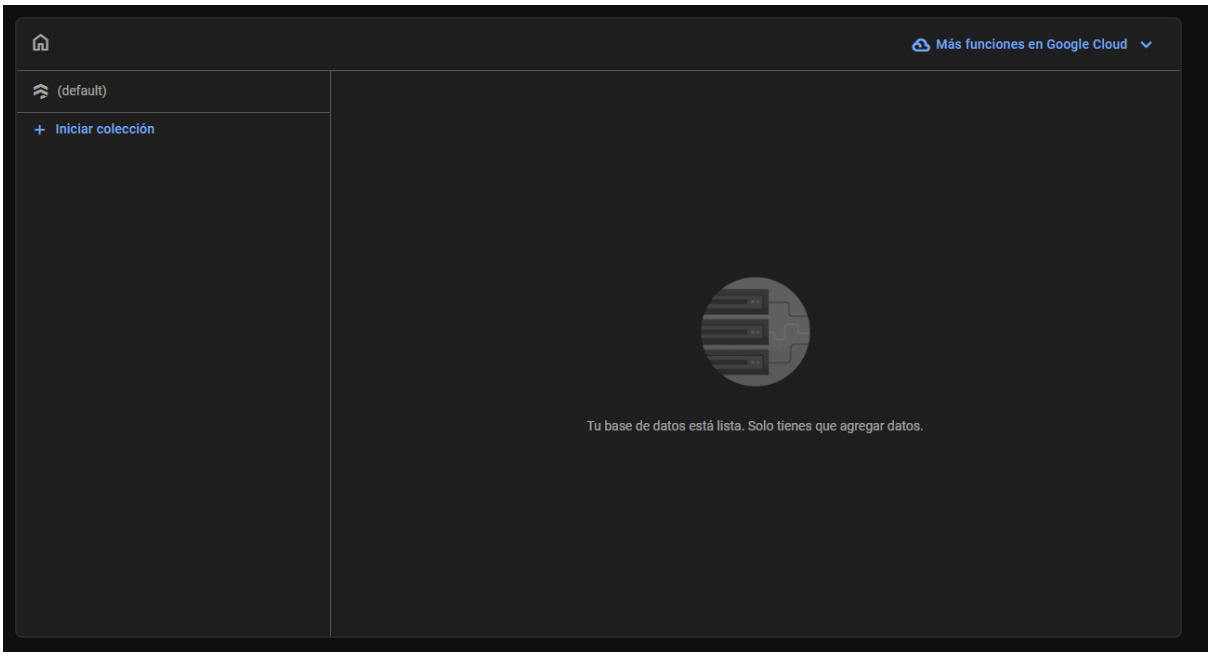
Seleccionamos la opción “Crear base de datos” y damos “siguiente” en el primer paso y en el segundo, marcamos la opción “Comenzar en **modo de prueba**”



Firestore ofrece dos modos de seguridad: "**modo de prueba**" y "**modo bloqueado**". El primero permite el acceso abierto a la base de datos por un tiempo limitado, ideal para pruebas iniciales, mientras que el segundo restringe el acceso y requiere reglas de seguridad configuradas manualmente. Para entornos de desarrollo es recomendable iniciar en modo de prueba y luego ajustar las reglas para producción.

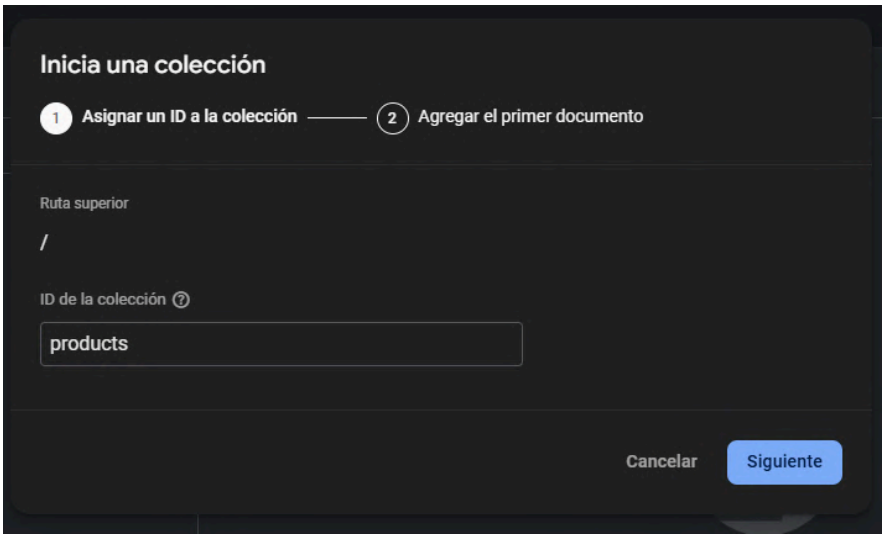


Damos al botón crear y luego de un breve período de carga tendremos a disposición nuestro panel de control para manejar la base de datos proporcionada por Firestore.

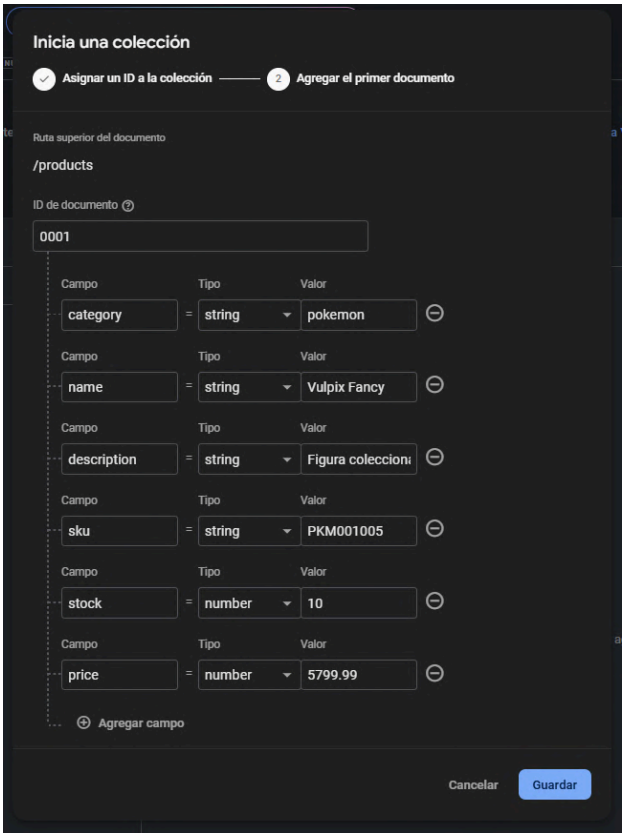


Creando una colección

Presionando en el botón “**Iniciar colección**” podremos crear una nueva colección de datos para nuestro proyecto y le colocaremos un ID para identificar el conjunto de datos.



Una vez creada la colección que nos permitirá agrupar productos, es momento de crear nuestro primer documento:

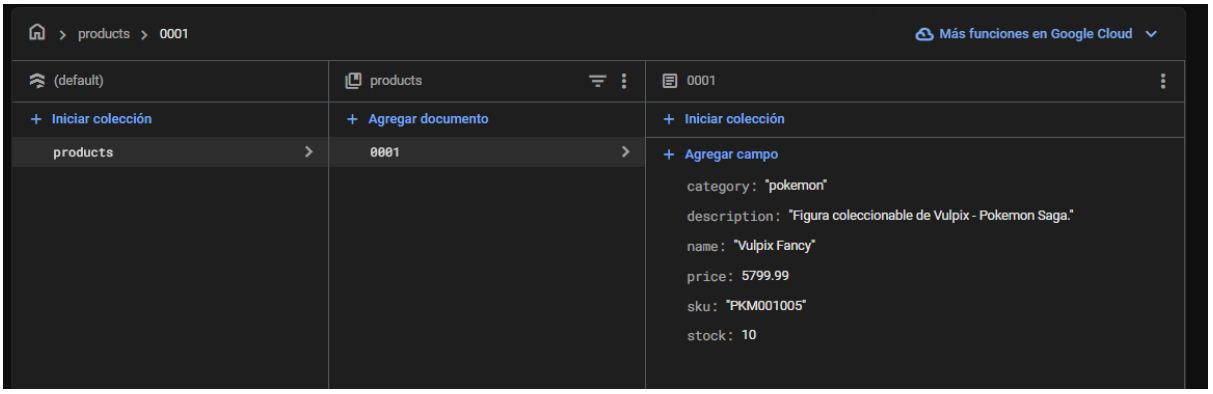


Los documentos funcionan como registros en una base de datos convencional. La primera vez que creamos una colección debemos establecer un primer documento para determinar los campos que tendrán este y los siguientes documentos que creemos relacionados a esa colección.

En este caso, agregamos un primer producto y le asignamos el tipo de dato correspondiente a cada propiedad.

Una propiedad puede ser del tipo **string**, **number**, **boolean**, **array**, **map** (objeto) entre otros y nos ayuda a guardar determinado formato en nuestros documentos.

Una vez guardado el primer registro, ya tendremos disponible nuestra colección y podremos seguir agregando nuevos documentos en el futuro.

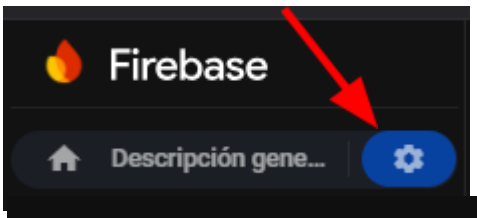


Si bien no existe una manera para agregar todos los productos desde un simple archivo de forma directa, Firestore tiene métodos que nos permitirán hacer “bulk inserts” o guardado masivo de registros desde el código.

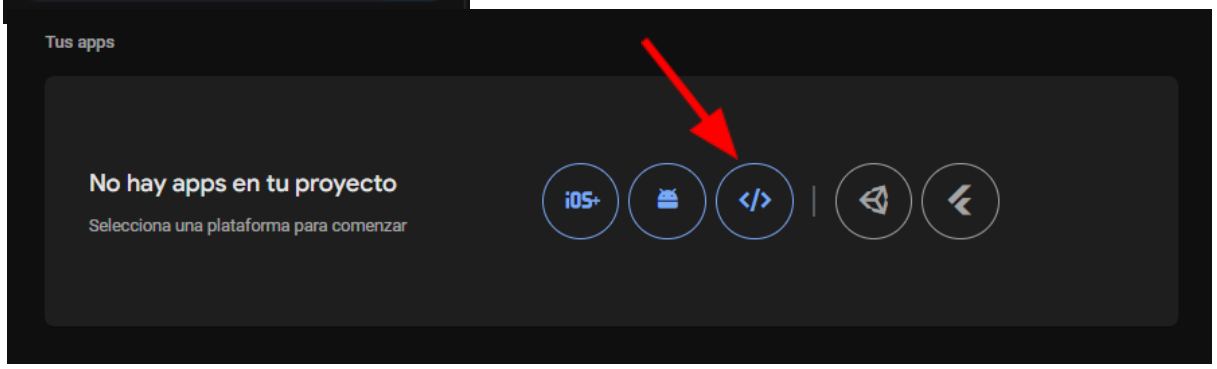
Configuración de Firebase Firestore en tu Proyecto

Creamos una nueva App en Firebase

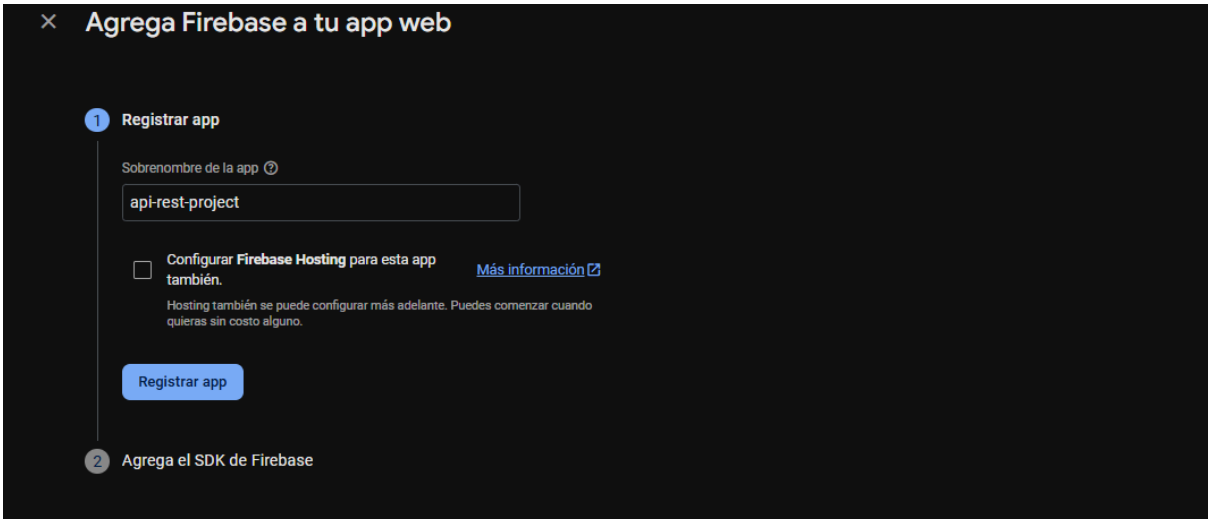
Lo primero a realizar es configurar una App para nuestro proyecto, para eso nos dirigimos a este ícono:



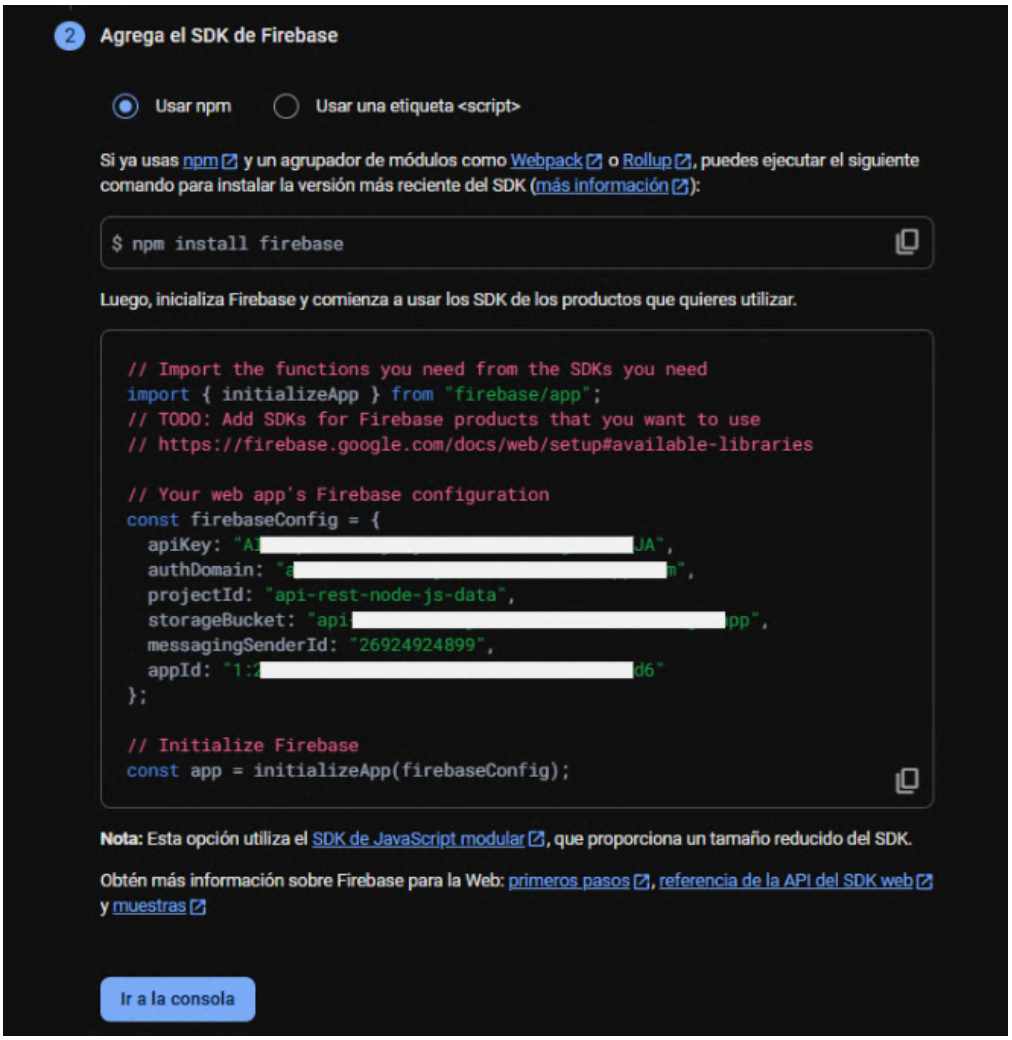
y luego seleccionamos el botón para agregar una nueva App de código:



Le colocamos un sobrenombre:



y nos entregará el comando para instalar firebase en nuestro proyecto y las credenciales de acceso necesarias para realizar la vinculación:



Instalación del SDK de Firebase

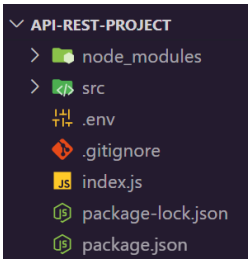
Primero, necesitas instalar el SDK de Firebase en tu proyecto mediante npm:

```
npm install firebase
```

Configuración del Archivo `.env`

Instalaremos la librería `dotenv` para manejar variables de entorno y no exponer nuestras credenciales:

```
npm install dotenv
```



Crea un archivo `.env` en la raíz de tu proyecto y agrega tus credenciales de Firebase.

Las credenciales son las que nos otorgó Firebase listadas en la `firebaseConfig`.

```
FIREBASE_API_KEY=KAjsneuYe_prX8Str-aeSDjHDJS5aieS9012SkhA
FIREBASE_AUTH_DOMAIN=your-project-name-data.firebaseapp.com
FIREBASE_STORAGE_BUCKET=your-project-name-data.firebasestorage.app
FIREBASE_APP_ID=9:12345678907:web:h5x76swsajh3476sde
```

Inicialización de Firebase

Crea un archivo `data.js` en la carpeta `data` dentro de `src` para inicializar Firebase y Firestore. Asegúrate de cargar las variables de entorno desde el archivo `.env`.

```
import 'dotenv/config';

import { initializeApp } from "firebase/app";
import { getFirestore } from 'firebase/firestore';

const firebaseConfig = {
```

```
  apiKey: process.env.FIREBASE_API_KEY,
  authDomain: process.env.FIREBASE_AUTH_DOMAIN,
  projectId: "api-rest-node-js-data",
  storageBucket: process.env.FIREBASE_STORAGE_BUCKET,
  messagingSenderId: "26924924899",
  appId: process.env.FIREBASE_APP_ID
};

// Initialize Firebase
const app = initializeApp(firebaseConfig);

// Initialize Firestore
const db = getFirestore(app);

export { db };
```

Con `initializeApp` y `firebaseConfig` creamos una nueva instancia del servicio de **Firebase** en nuestro proyecto y luego mediante `getFirestore` configuramos el acceso a nuestros documentos de **Firestore**.

Finalmente exportamos `db` para tener acceso a nuestros documentos desde otros archivos de nuestro proyecto.

Implementación del Modelo de Productos

Modifica el archivo `products.model.js` para utilizar Firestore en lugar de un archivo JSON local. Este archivo contendrá métodos para interactuar con la colección de productos en Firestore.

```
// products.model.js
import { db } from '../data/data.js';
import {
    collection,
    getDocs,
    getDoc,
    addDoc,
    deleteDoc,
    doc
} from 'firebase/firestore';

const productsCollection = collection(db, 'products');

// Método para buscar un producto por su ID
```

```
export async function getProductById(id) {
  const productDoc = await getDoc(doc(productsCollection, id));
  if (productDoc.exists()) {
      return productDoc.data();
  } else {
      return null;
  }
};

// Método para obtener todos los productos
export async function getAllProducts() {
  const querySnapshot = await getDocs(productsCollection);
  const products = [];
  querySnapshot.forEach((doc) => {
      products.push({ id: doc.id, ...doc.data() });
  });
  return products;
};

// Método para guardar un producto en Firestore
export async function saveProduct(product) {
  await addDoc(productsCollection, product);
};

// Método para eliminar un producto por su ID
export async function deleteProduct(id) {
  await deleteDoc(doc(productsCollection, id));
};
```

Veamos en detalle los métodos de firestore para acceder a nuestros datos:

1. `collection`:
    - Este método se utiliza para obtener una referencia a una colección en Firestore.

    ```
    const productsCollection = collection(db, 'products');
    ```

    - `db` es la instancia de firestore que importamos desde `data.js` y `'products'` es el nombre de nuestra colección.

2. `getDocs`:
    - Este método se utiliza para obtener todos los documentos de una colección.

```
const querySnapshot = await getDocs(productsCollection);
```

- Recibe como parámetro la referencia a la colección que queremos consultar.

3. `getDoc`:
   - Este método se utiliza para obtener un único documento de una colección por su ID.

```
const productDoc = await getDoc(doc(productsCollection, id));
```

- `productsCollection` referencia a la colección de la cual se quieren obtener los documentos, `id` es el id con el que se encuentra identificado el documento y se utilizan dentro de `doc` que sirve para obtener una referencia a un documento específico dentro de una colección.

4. `addDoc`:
   - Este método se utiliza para agregar un nuevo documento a una colección.

```
addDoc(productsCollection, product);
```

- `productsCollection` referencia a la colección de la cual se quieren obtener los documentos y `product` contiene los datos del nuevo documento que se desea agregar, respetando los campos definidos al momento de crear la colección.

5. `deleteDoc`:
   - Este método se utiliza para obtener una referencia a un documento específico dentro de una colección.

```
deleteDoc(doc(productsCollection, id));
```

- recibe los mismo parámetros que `getDoc`.

De esta manera ya conocemos todo lo que necesitamos para acceder a nuestros datos en la nube, almacenados dentro de una instancia de **Firebase** mediante su servicio **Firestore**.

Ajustes adicionales

Una vez migrado nuestro modelo para consumir los datos desde la nueva fuente, es momento de revisar nuestros archivos de `controllers` y `services` para verificar que todo siga funcionando correctamente.

Una ventaja de la arquitectura de capas utilizada es que, cambiar la implementación de una determinada capa, no significa cambios en las otras capas, por ejemplo, en este caso, al mantener el mismo nombre en los métodos de los modelos (getProductById, getAllProducts, etc.) el impacto en los archivos que utilizaban estos métodos debería ser nulo.

Uno de los problemas con el que podemos encontrarnos es el manejo del asincronismo, mientras que con `filesystem` podíamos acceder a los archivos de manera síncrona o asíncrona, con Firestore nuestras consultas siempre devuelven promesas por lo que debemos utilizar el manejo de asincronismo.

Revisa tus capas para verificar que realizaste un correcto manejo de código asíncrono con `async` y `await` o `then` y `catch` en cada uno de tus métodos que van desde las rutas hasta los modelos.

Ejercicio Práctico

Migración a Firestore

Después de completar la tarea de acceso a los datos mediante archivos, Sabrina y Matías vuelven con un nuevo reto. Esta vez, Sabrina te desafía con una sonrisa:

"Ahora que has organizado bien los datos, es momento de llevarlos a la nube. Vamos a reemplazar los archivos JSON locales con una base de datos real: Firestore en Firebase.."

Matías interviene con un tono más analítico:

"Firestore nos permitirá manejar datos de manera más flexible y escalable. Tu tarea será conectar tu API a esta base de datos y asegurarte de que todo funcione perfectamente."

Misión:

1. **Configurar Firestore en tu Proyecto**
    - Crea un nuevo proyecto en Firebase y configura Firestore.
    - Diseña la estructura de la base de datos siguiendo el esquema de los JSON que usaste previamente.

2. **Configurar el servicio de Firebase/Firestore en tu proyecto:**
    - Instala las dependencias necesarias.
    - Crea un archivo de configuración con las credenciales aportadas por Firebase.
    - Coloca tus credenciales en un archivo `.env` y utilizalas en tu archivo de configuración.

3. **Verifica la compatibilidad con otras capas:**
    - Valida que el código de los servicios y controladores siga funcionando correctamente.
    - Utiliza POSTMAN para consultar tu API Rest y recibir los datos provenientes desde Firestore.

Materiales y Recursos Adicionales:

- **Documentación Oficial de Firebase Firestore:**
  Guía de Firestore – Revisión detallada de cómo estructurar, leer y escribir datos en Firestore.

- **Guía de Firebase para Node.js:**
  Documentación de Firebase para Node.js – Configuración y uso del SDK de Firebase en proyectos backend.

- **Curso sobre API REST con Node.js y Firestore:**
  Tutoriales y videos en YouTube como este curso gratuito que explican paso a paso cómo construir una API utilizando Firestore.

Preguntas para Reflexionar:

- ¿Qué ventajas tiene usar Firestore en una aplicación escalable y en tiempo real?
- ¿Cómo aseguraste que tu API pudiera leer y escribir datos en Firestore sin afectar la arquitectura RESTful?
- ¿Qué cambios realizarías si tu API manejara un volumen mayor de datos en Firestore?
- ¿Qué dificultades encontraste al migrar desde archivos JSON a Firestore y cómo las solucionaste?

Próximos Pasos:

- **Autenticación y Autorización**: Manejando el acceso público y privado de nuestros datos.
- **Fin de la cursada**: daremos cierre al curso mediante la presentación de proyectos finales

Buenos Aires Ciudad