**M2S16: React CRUD con Firebase**

**Hola, nos volvemos a encontrar, en esta semana 16 tenemos mucho contenido por trabajar y por ende bastante que hacer. Por lo que te recomiendo que leas con calma y puedas realizar paso a paso todo lo que se mostrará en esta semana. Recuerda que si tienes alguna duda puedes consultar con tus coaches o compañeros de clase; también puedes investigar en internet y traernos comentarios y aportaciones importantes en la clase del sábado.**

**A continuación, te muestro el contenido a trabajar.**

**Sigue adelante.!!**

**ÍNDICE**

* [Firebase](https://github.com/U-Camp/BOOT-M2-SEM16#firebase)
* [Cloud Firestore](https://github.com/U-Camp/BOOT-M2-SEM16#cloud-firestore)
  + [Creación de cuenta e instancias](https://github.com/U-Camp/BOOT-M2-SEM16#creaci%C3%B3n-de-cuentas-e-instancias)
  + [Crear una base de datos](https://github.com/U-Camp/BOOT-M2-SEM16#crear-una-base-de-datos)
  + [Crear elementos](https://github.com/U-Camp/BOOT-M2-SEM16#crear-elementos)
  + [Leer elementos](https://github.com/U-Camp/BOOT-M2-SEM16#leer-elementos)
  + [Actualizar elementos](https://github.com/U-Camp/BOOT-M2-SEM16#actualizar-elementos)
  + [Borrar elementos](https://github.com/U-Camp/BOOT-M2-SEM16#borrar-elementos)
* [Autenticación (Firebase Auth)](https://github.com/U-Camp/BOOT-M2-SEM16#autenticaci%C3%B3n-firebase-auth)
* [React con servicios de terceros](https://github.com/U-Camp/BOOT-M2-SEM16#react-con-servicios-de-terceros)

**Firebase**

Firebase es una plataforma para el desarrollo de aplicaciones web y móviles.

Entre sus características más importantes se encuentran:

**Integración con la “nube”.** Colaborando con Google Cloud Platform, permite la escalabilidad de tu aplicación de manera segura y eficiente. Conforme vayas obteniendo más usuarios, Firebase te permitirá ser flexible, aumentar el rendimiento de la herramienta y no tener ningún conflicto al crecer.

**Sincronización.** Es rápido. Muy rápido. Debido a su abstracción, contiene diferentes funciones que nos permiten gestionar nuestros datos fácilmente y sin tanta complicación.

**Arquitectura e infraestructura.** Al contar con diferentes productos, puedes hacer desde aplicaciones simples hasta complejas, entrelazándolos.

**“Serverless”.** Cada herramienta te permite acceder a un SDK (un conjunto de herramientas) y su diseño logra que no sea necesario construir un servidor.

Es importante saber que:

* Firebase se fundó en 2012.
* Fue comprado por Google en 2014.
* Su área de especialización ha sido el desarrollo móvil. Sin embargo, puedes utilizarlo para Web sin problemas.
* Puedes utilizarlo con IOS, Android, Web, Unity y C++.

**Productos de Firebase**

Firebase, funciona como un ecosistema de productos. Al integrarlo, tendrás a tu disposición diferentes integraciones. Podríamos dividirlas en compilación, calidad y métricas.

**COMPILACIÓN**

* Cloud Firestore
* Kit de AA (Aprendizaje automático)
* Cloud Functions
* Autenticación
* Hosting
* Cloud Storage
* Realtime Database

**CALIDAD**

* Crashlytics
* Performance Monitoring
* Test Lab

**MÉTRICAS**

* Analytics
* Predictions
* Firebase A/B Testing
* Cloud Messaging
* Remote Config
* Dynamic Links
* Indexación de aplicaciones
* Invites

**Cloud Firestore**

**Creación de cuentas e instancias**

Para comenzar, lo primero que debemos hacer es ir directamente a la página oficial de Firebase ([https://firebase.google.com](https://firebase.google.com/))

Darás clic a comenzar y te abrirá el área de consola (<https://console.firebase.google.com/u/0/>)

Te pedirá crear una cuenta en Google. Si ya lo tienes, sólo accede con tu usuario y contraseña.

Encontrándonos en el panel de control, accederás a la zona de “Proyectos recientes” y dentro, verás un cuadro que dice “Añadir proyecto”.

Al darle clic, te abrirá un área de configuración inicial, donde deberás poner el nombre del proyecto.

Recuerda que no debe ser el mismo nombre “utel-ejemplo-2021”. Deberás usar otro porque este ya está ocupado asignado a mi cuenta. Sólo puede existir un nombre de proyecto globalmente.

Avanza con la creación del proyetco. Te dará una opción sobre "Google Analytics". Deshabilita la opción de Google Analytics para Firebase.

Después, el proyecto empezará a crearse.

Y te abrirá el área de configuración principal y su panel de control.

Con esto realizado, estamos listos para crear nuestra aplicación. Debemos registrar el app con Firebase.

Le darás clic al botón de </>.

Posteriormente, tendrás esta pantalla, en la cual podemos hacer el registro. Realizaremos todos los pasos.

No seleccionaremos la casilla de Firebase Hosting.

Una vez registrado, te aparecerá este siguiente paso. Lo que haremos será crear un proyecto dentro de Visual Studio Code e integraremos el siguiente código mencionado.

Sólo que será a través de una aplicación con Node.js.

Haremos una pausa aquí y nos iremos directamente a la generación de la aplicación.

**Generación de la aplicación**

Lo que haremos ahora será crear un proyecto simple con Node.js, en el cual insertaremos y aplicaremos los servicios respectivos de Firebase. Abrirás la consola. Crearemos las carpetas y archivos:

mkdir utel-firebase-ejemplo

cd utel-firebase-ejemplo

touch index.js

touch index.html

Esta será la estructura de tu proyecto:

|- index.js

|- index.html

Nos situaremos en el archivo HTML y crearemos la base del documento. Añadiremos el SDK para vincular Firebase, que nos dejaron en el área de registro de la aplicación.

Te deberá quedar de esta forma:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Firebase</title>

</head>

<body>

<!-- The core Firebase JS SDK is always required and must be listed first -->

<script src="https://www.gstatic.com/firebasejs/8.2.1/firebase-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: "AIzaSyDzCJOt3GUDffojp2cJHAD12MK6rnmnlrM",

authDomain: "utel-ejemplo-2021-89fbc.firebaseapp.com",

projectId: "utel-ejemplo-2021-89fbc",

storageBucket: "utel-ejemplo-2021-89fbc.appspot.com",

messagingSenderId: "328179447374",

appId: "1:328179447374:web:ad55953238563106de351f"

};

// Initialize Firebase

firebase.initializeApp(firebaseConfig);

</script>

</body>

</html>

Recuerda que los valores del objeto “firebaseConfig” serán diferentes al ejemplo. Tendrás otros valores y está bien. Ahí se incluyen las contraseñas para el acceso a la base de datos.

Por ahora, está bien dejarlo abierto pero, conforme vayamos creciendo nuestra aplicación, deberemos ejecutar prácticas de seguridad.

Ahora, deberemos instalar, a través de la consola, la librería de Firebase, de manera global. Para hacerlo, se agrega el parámetro “-g” que permite disponerlo en cualquier lugar de tu computador.

npm install -g firebase-tools

Listo. Regresemos a nuestro panel de control.

Veamos los diferentes servicios y su uso.

**Servicios**

**Cloud Firestore** es una base de datos flexible y escalable para la programación en servidores, dispositivos móviles y la Web desde Firebase y Google Cloud Platform.

Con este servicio, se mantienen tus datos sincronizados entre apps cliente a través de agentes de escucha en tiempo real y ofrece asistencia sin conexión para dispositivos móviles y la Web, por lo que puedes compilar aplicaciones con capacidad de respuesta que funcionan sin importar la latencia de la red ni la conectividad a Internet.

Cloud Firestore también ofrece una integración sin interrupciones con otros productos de Firebase y Google Cloud Platform, incluido Cloud Functions.

**¿Cómo funciona?**

Cloud Firestore, es una base de datos NoSQL alojada en la nube a la que pueden acceder tus apps para iOS, Android y Web directamente desde los SDK nativos.

Cloud Firestore, también está disponible en los SDK nativos de Node.js, Java, Python y Go, junto con las API de REST y RPC.

A partir del modelo de datos NoSQL de Cloud Firestore, almacenas los datos en documentos que contienen campos que se asignan a valores. Estos documentos se almacenan en colecciones, que son contenedores para tus documentos y que puedes usar para organizar tus datos y compilar consultas. Los documentos admiten varios tipos de datos diferentes, desde strings y números simples, hasta objetos anidados complejos.

También puedes crear subcolecciones dentro de documentos y crear estructuras de datos jerárquicas que se ajustan a escala a medida que tu base de datos crece. El modelo de datos de Cloud Firestore admite cualquier estructura de datos que funcione mejor con tu aplicación.

Además, las consultas de Cloud Firestore son expresivas, eficientes y flexibles. Crea consultas superficiales para recuperar datos en el nivel del documento, sin la necesidad de recuperar la colección completa ni las subcolecciones anidadas. Agrega criterios de orden, filtros y límites a tus consultas o cursores para paginar los resultados.

Para mantener actualizados los datos de tus apps sin tener que recuperar toda la base de datos cada vez que haya una actualización, agrega agentes de escucha en tiempo real. Estos te notifican con una instantánea de los datos cada vez que los datos de tus apps cliente escuchan para detectar cambios y recuperan solo los cambios nuevos.

Protege el acceso a tus datos en Cloud Firestore con Firebase Authentication y las reglas de seguridad de Cloud Firestore para Android, iOS y JavaScript, o con la administración de identidades y accesos (IAM) para los lenguajes en el lado del servidor.

**Crear una base de datos**

Para iniciar, crearemos nuestra base de datos desde el panel de control. Da clic en “Cloud Firestore”.

Y luego, “Crear base de datos”.

Verás el siguiente recuadro. Da “Siguiente”. Por ahora manejaremos la base de datos en modo de producción. Finalmente, clic en el botón “Habilitar”.

Para implementar Firebase puntualmente en nuestro proyecto, haremos los siguientes tres comandos:

firebase login

firebase init

**firebase login.** Te permitirá autenticarte con tu correo en tu computadora con Firebase.

**firebase init.** Te permitirá inicializar Firebase en el proyecto. Selecciona “Firestore” y “Functions”. Luego, ubícate en tu proyecto. Avanza con todas las configuraciones por defecto.

A partir de aquí, vamos a utilizar la documentación de Cloud Firestore, desde el sitio oficial de Firebase.

<https://firebase.google.com/docs/firestore?hl=es>

Te recomendamos enormemente, en caso de ser necesario, revisar la documentación. Mucha de la información escrita aquí está relacionada o es similar a la descrita ahí mismo.

**Crear elementos**

Hay varias formas de escribir datos en Cloud Firestore:

* Configurar los datos de un documento dentro de una colección y especificar explícitamente un identificador de documento.
* Agregar un documento nuevo a una colección. En este caso, Cloud Firestore genera automáticamente el identificador de documento.
* Crear un documento vacío con un identificador generado automáticamente y asignarle datos después.

Explicamos cómo usar los métodos de configurar, agregar o actualizar documentos individuales en Cloud Firestore. Si deseas escribir datos de manera masiva, consulta Transacciones y escrituras en lotes.

db.collection("cities").doc("LA").set({

name: "Los Angeles",

state: "CA",

country: "USA"

})

.then(() => {

console.log("Document successfully written!");

})

.catch((error) => {

console.error("Error writing document: ", error);

});

Cuando usas set() para crear un documento, debes especificar un ID para el documento que vas a crear. Por ejemplo:

db.collection("cities").doc("new-city-id").set(data);

**Leer elementos**

Hay dos maneras de recuperar datos almacenados en Cloud Firestore. Ambos métodos pueden usarse con documentos, colecciones de documentos o resultados de consultas:

* Llamar a un método para obtener los datos.
* Configurar un objeto de escucha para recibir eventos de cambios de datos.

Cuando configuras un objeto de escucha, Cloud Firestore le envía una instantánea inicial de los datos y, luego, otra instantánea cada vez que se modifica el documento.

Generaremos un ejemplo:

var citiesRef = db.collection("cities");

citiesRef.doc("SF").set({

name: "San Francisco", state: "CA", country: "USA",

capital: false, population: 860000,

regions: ["west\_coast", "norcal"] });

citiesRef.doc("LA").set({

name: "Los Angeles", state: "CA", country: "USA",

capital: false, population: 3900000,

regions: ["west\_coast", "socal"] });

citiesRef.doc("DC").set({

name: "Washington, D.C.", state: null, country: "USA",

capital: true, population: 680000,

regions: ["east\_coast"] });

citiesRef.doc("TOK").set({

name: "Tokyo", state: null, country: "Japan",

capital: true, population: 9000000,

regions: ["kanto", "honshu"] });

citiesRef.doc("BJ").set({

name: "Beijing", state: null, country: "China",

capital: true, population: 21500000,

regions: ["jingjinji", "hebei"] });

Y luego, obtendremos el dato:

var citiesRef = db.collection("cities");

var query = citiesRef.where("state", "==", "CA");

**Actualizar elementos**

Para actualizar algunos campos de un documento sin reemplazarlo por completo, usa el método update():

var washingtonRef = db.collection("cities").doc("DC");

return washingtonRef.update({

capital: true

})

.then(() => {

console.log("Document successfully updated!");

})

.catch((error) => {

console.error("Error updating document: ", error);

});

**Borrar elementos**

Para borrar campos específicos de un documento, usa el método FieldValue.delete() cuando lo actualices:

db.collection("cities").doc("DC").delete().then(() => {

console.log("Document successfully deleted!");

}).catch((error) => {

console.error("Error removing document: ", error);

});

**Autenticación (Firebase Auth)**

Puedes usar Firebase Authentication para permitir que los usuarios accedan a la app con uno o más métodos de acceso, que incluyen el acceso con dirección de correo electrónico y contraseña, y proveedores de identidad federada, como Acceso con Google y Acceso con Facebook.

Este instructivo te muestra cómo agregar un acceso con dirección de correo electrónico y contraseña a la app para comenzar con Firebase Authentication.

Crea un formulario que permita a los usuarios nuevos registrarse en la app mediante su dirección de correo electrónico y una contraseña.

Cuando un usuario complete el formulario, valida la dirección de correo electrónico y la contraseña que proporcionó el usuario para después pasarlos al método createUserWithEmailAndPassword:

firebase.auth().createUserWithEmailAndPassword(email, password)

.then((userCredential) => {

// Signed in

var user = userCredential.user;

// ...

})

.catch((error) => {

var errorCode = error.code;

var errorMessage = error.message;

// ..

});

Crea un formulario que permita a los usuarios existentes acceder con su dirección de correo electrónico y una contraseña. Cuando un usuario complete el formulario, llama al método signInWithEmailAndPassword:

firebase.auth().signInWithEmailAndPassword(email, password)

.then((userCredential) => {

// Signed in

var user = userCredential.user;

// ...

})

.catch((error) => {

var errorCode = error.code;

var errorMessage = error.message;

});

**React con servicios de terceros**

Para integrar Firebase con React, es bastante importante preparar el proyecto de esta forma.

Vale la pena utilizar ReactFire para la gestión de datos.

npm install --save reactfire firebase

Posteriormente, agregaremos el siguiente ejemplo, de acuerdo a la documentación, para intregarlo:

import React, { Component } from 'react';

import { render } from 'react-dom';

import 'firebase/firestore';

import { FirebaseAppProvider, useFirestoreDocData, useFirestore } from 'reactfire';

const firebaseConfig = {

/\* Agregar tu configuración. Lo que colocaremos aquí es un ejemplo \*/

apiKey: "\*\*\*\*\*\*\*\*\*\*\*",

authDomain: "\*\*\*\*\*\*\*\*\*\*\*",

databaseURL: "\*\*\*\*\*\*\*\*\*\*\*",

projectId: "\*\*\*\*\*\*\*\*\*\*\*",

storageBucket: "\*\*\*\*\*\*\*\*\*\*\*",

messagingSenderId: "\*\*\*\*\*\*\*\*\*\*\*",

appId: "\*\*\*\*\*\*\*\*\*\*\*"

};

function Burrito() {

// Accedemos a la librearía de Firestore

const burritoRef = useFirestore()

.collection('tryreactfire')

.doc('burrito');

// Suscribirse an los cambios en tiempo real

const { status, data } = useFirestoreDocData(burritoRef);

// Revisamos el status de carga

if (status === 'loading') {

return <p>Fetching burrito flavor...</p>;

}

return <p>The burrito is {data.yummy ? 'good' : 'bad'}!</p>;

}

function App() {

return (

<FirebaseAppProvider firebaseConfig={firebaseConfig}>

<h1>🌯</h1>

<Burrito />

</FirebaseAppProvider>

);

}

render(<App />, document.getElementById('root'));

Puedes ver el ejemplo directamente aquí: <https://stackblitz.com/edit/reactfire-sample-btyhpp?file=index.js>

Este ejemplo fue tomado directamente de la documentación de ReactFire.

**Hasta aquí el contenido de esta semana, ¿pudiste aplicar todos los pasos?, ¿qué te pareció este contenido?, ¿puedes imaginar aplicarlo en tu vida laboral?, espero que todas las respuestas sean afirmativas y que el contenido te sea de mucha utilidad.**

**Muchas gracias por tu atención, nos vemos en clases.**