

Escuela de Informática

Curso: Desarrollo de aplicaciones web

EIF 508

Profesora: Gregorio Villalobos Camacho

Desarrollo de una aplicación web progresiva (PWA)

Carlos Arguedas

Bidkar Solís

II Ciclo, 2019

# 

**Introducción**

El desarrollo de ambientes web es uno de los ambientes de desarrollo de constante cambio, existencias de tecnologías como Angular, React y el mismo JavaScript como definición se encuentran en constantes cambios, y hasta se encuentran entregas cada seis meses con una nueva versión.

Con la llegada del primer Iphone (año 217), Apple realizó un concurso para desarrollar una tecnología que les permita desarrollar aplicaciones para sus celulares móviles. A partir de esta carrera uno de los proyectos género que a partir del buscador Safari, en su versión de IOs se incluyeran ciertos metatags que permitieron emular un servicio web como una aplicación nativa.

Con el pasar de los tiempos con el soporte de Google Chrome para Android como navegador web móvil comenzó el desarrollo de Aplicaciones Web Progresivas (PWA). Y que hoy en día se encuentra disponible en muchas plataformas, y para lo cual Google es el pionero y el que define las reglas para su desarrollo.

Esta tecnología es basada en tecnologías básicas del desarrollo web como lo son: HTML5, CSS y JavaScript. Es la tecnología usada para el desarrollo de la aplicación web para luego utilizar las personalizaciones que da CSS para acomodar según el tamaño de la pantalla.

A continuación, presentaremos un ejemplo básico en el cual se agrega además la parte de persistencia de datos, esto haciendo uso de la tecnología de almacenamiento de Cloud se servicio gratuito Firebase.

**Aplicaciones progresivas web**

Como anteriormente mencionamos las aplicaciones progresivas residen su soporte en el servicio del buscador web instalado en el dispositivo. Por lo tanto, como observamos tiene una gran dependencia del proveedor de buscador web y por lo tanto es el mismo servicio el que da el soporte.

Para lo cual Google crea un estándar de los requerimientos que la aplicación web debe cumplir para poder considerar un sitio web como una aplicación progresiva, a continuación se presentara los requerimientos basado en la aplicación de nombre Mi Comida:

**Manifesto Web App:**

El manifiesto es un archivo de tipo JSON simple, que le informa al navegador sobre su aplicación web y cómo debe comportarse cuando se instala en el dispositivo móvil o escritorio del usuario.

Chrome requiere tener este manifestó para mostrar el mensaje Agregar a la pantalla de inicio. Un archivo de manifiesto incluye información sobre el nombre de la aplicación, los íconos que debe usar, el start\_url en el que debe comenzar cuando se inicia, y más.

Ejemplo aplicado:

{

    "name": "Mi Comida",

    "short\_name": "MiComida",

    "start\_url": "./index.html",

    "display": "standalone",

    "background\_color": "#FFE9D2",

    "theme\_color": "#FFE1C4",

    "orientation": "portrait-primary",

    "icons": [

        {

            "src": "/img/icons/icon-72x72.png",

            "type": "image/png",

            "sizes": "72x72"

        }, {

            "src": "/img/icons/icon-96x96.png",

            "type": "image/png",

            "sizes": "96x96"

        }, {

            "src": "/img/icons/icon-128x128.png",

            "type": "image/png",

            "sizes": "128x128"

        }, {

            "src": "/img/icons/icon-144x144.png",

            "type": "image/png",

            "sizes": "144x144"

        }, {

            "src": "/img/icons/icon-152x152.png",

            "type": "image/png",

            "sizes": "152x152"

        }, {

            "src": "/img/icons/icon-192x192.png",

            "type": "image/png",

            "sizes": "192x192"

        }, {

            "src": "/img/icons/icon-384x384.png",

            "type": "image/png",

            "sizes": "384x384"

        }, {

            "src": "/img/icons/icon-512x512.png",

            "type": "image/png",

            "sizes": "512x512"

        }

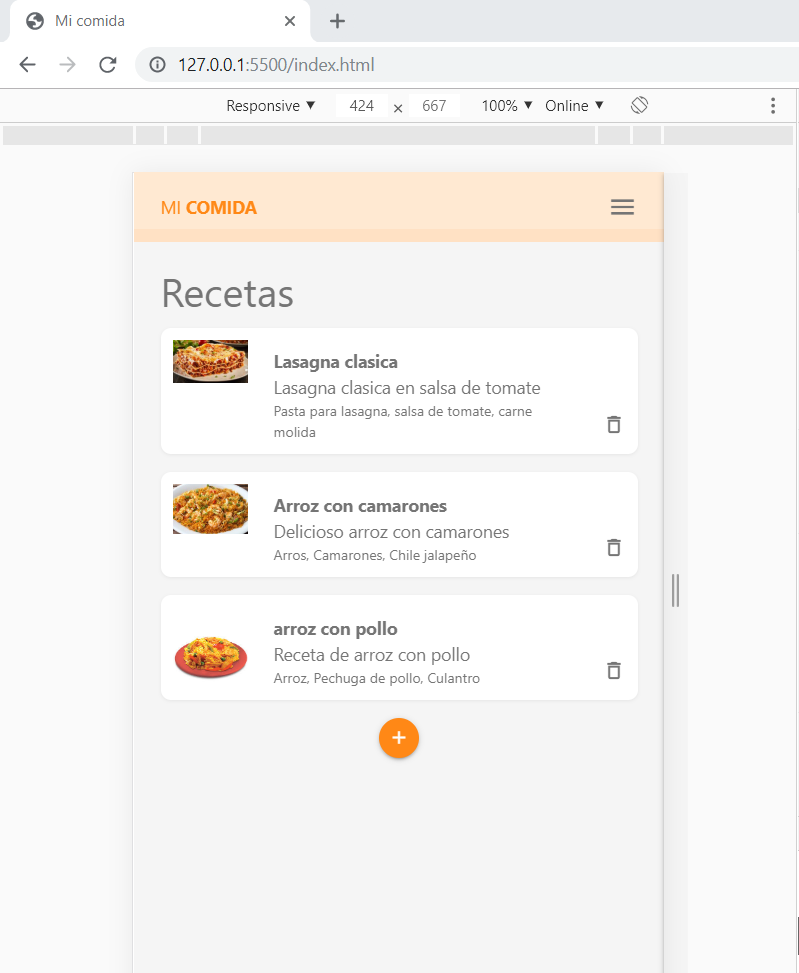
    ]

}

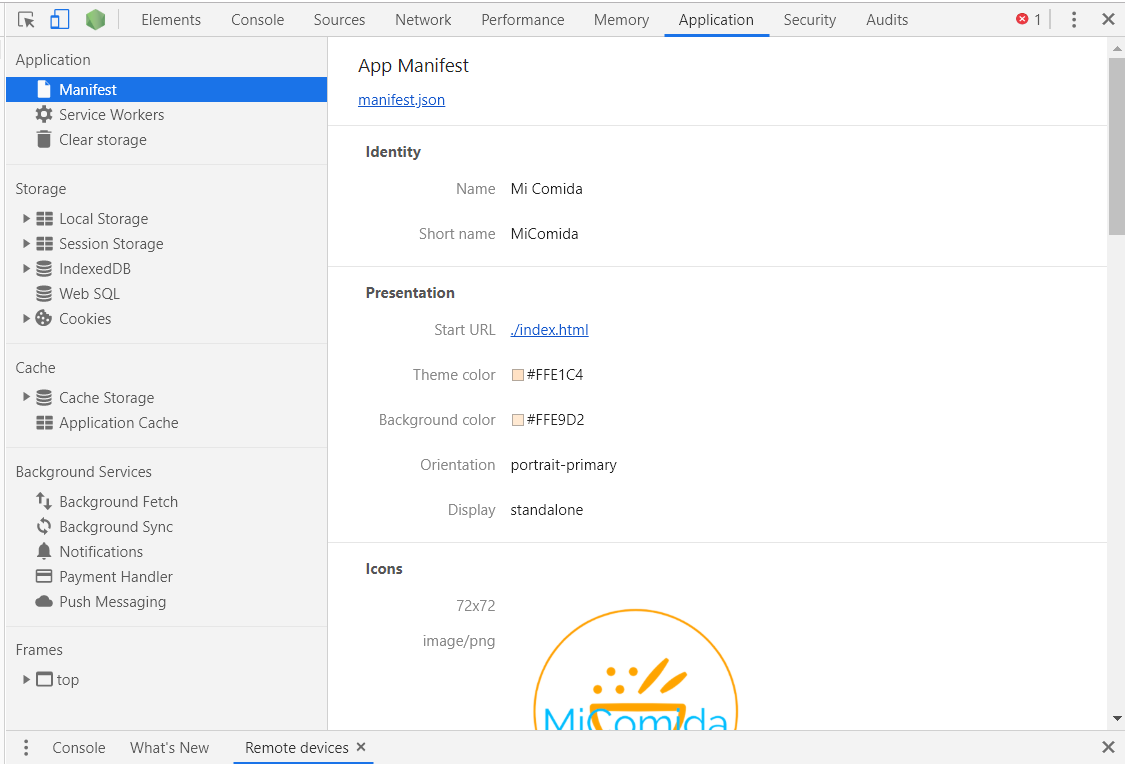
Una vez finalizado el manifestó este debe ser agregado a todas las paginas del aplicativo: <link rel="manifest" href="/manifest.json">. Y en cuando al tamaño de iconos, este como mínimo pide iconos de tamaño 192x192 y 512x512.

Y por último desarrollar la aplicación como un aplicativo web normal y responsivo para los cambios de pantallas en los cuales se van a presentar. Esta herramienta se convierte en una tecnología importante para el desarrollo de negocios que no tienen la capacidad económica para el desarrollo nativo para las plataformas actuales.

A continuación, una muestra de la aplicación:



Y un factor muy interesante para el desarrollo de PWA es que el navegador como tal brinda las herramientas para realizar pruebas y observar los cambios de su trabajo. Para ello buscamos la opción de Aplication en el modo desarrollador del navegador. A continuación, un ejemplo con el manifestó de la aplicación Mi Comida:



**Service Worker**

Este es el segundo requerimiento para el desarrollo de una aplicación progresiva, es un archivo JavaScript que funciona como un tipo de trabajador web. Funcionan por separado del hilo principal del navegador para manejar notificaciones push, sincronizar datos en segundo plano, almacenar en caché o recuperar solicitudes de recursos, interceptar solicitudes de red y recibir actualizaciones centralizadas.

Este servicio se utiliza en un intento de dar a las aplicaciones web progresivas la capacidad de proporcionar el alto rendimiento y la rica experiencia de las aplicaciones móviles nativas, con el bajo espacio de almacenamiento, las actualizaciones en tiempo real y la visibilidad mejorada del motor de búsqueda.

Pasan por un ciclo de vida de tres pasos de registro, instalación y activación. El registro es decirle al navegador la ubicación del trabajador para preparar instalación. La instalación se ejecuta cuando no hay un trabajador de servicio instalado en el navegador para la aplicación web, o si hay una actualización para el trabajador de servicio. La activación se produce cuando todas las páginas de PWAs están cerradas, por lo que no hay conflicto entre la versión anterior y la actualizada

**Evento Fetch**

Y, por último, un método requerido para la aplicación es el método Fetch para escuchar los request y almacenar los necesarios para tenerlos en cache para simular que la aplicación se encuentra en línea mientras regresa el servicio de internet.

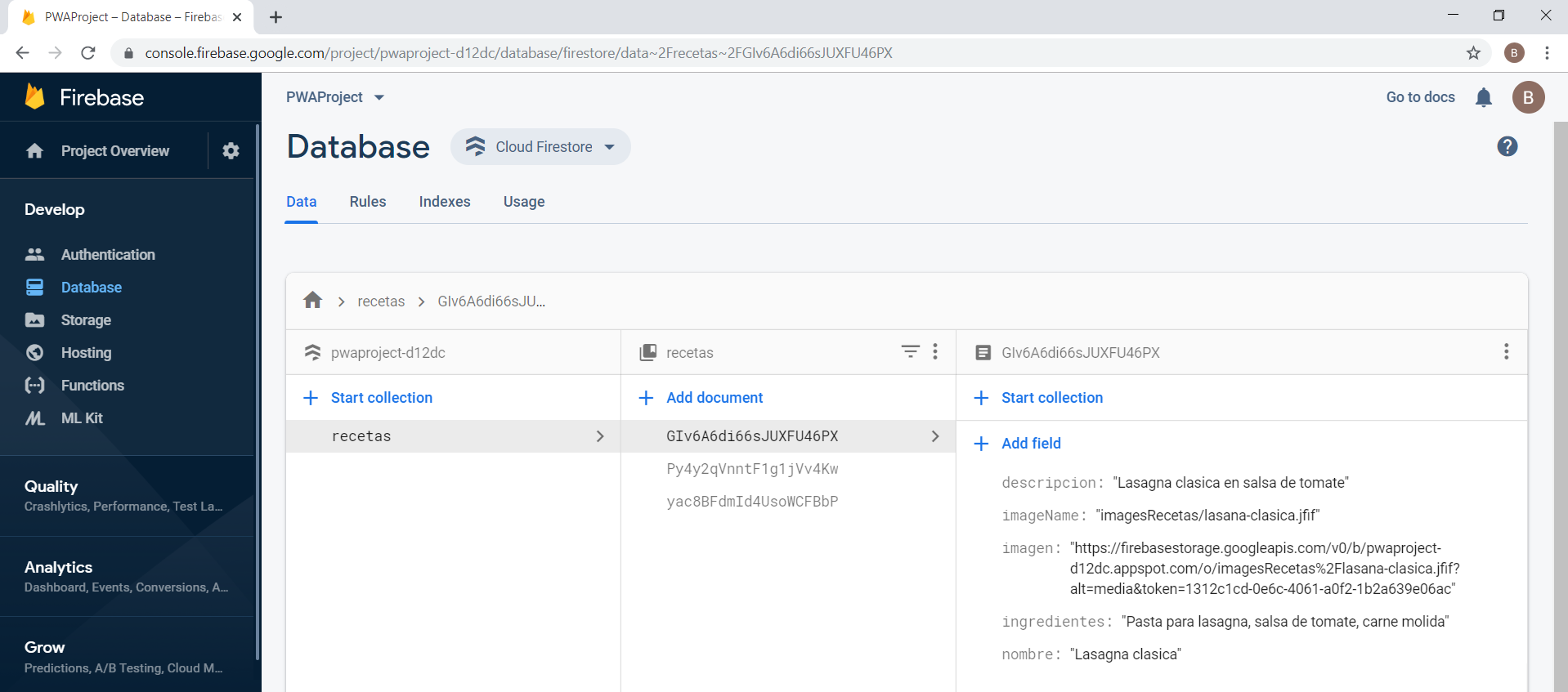
Este listener se programa en el Service Worker, en el cual se almacena en cache los recursos necesarios para eventos offline del servicio web. Se puede realizar un guardado en cache estático y también dinámico. Estático para los servicios básicos, y dinámico para ser flexible y cachear los servicios que el cliente haya accedido de manera dinámica.

**Firebase**

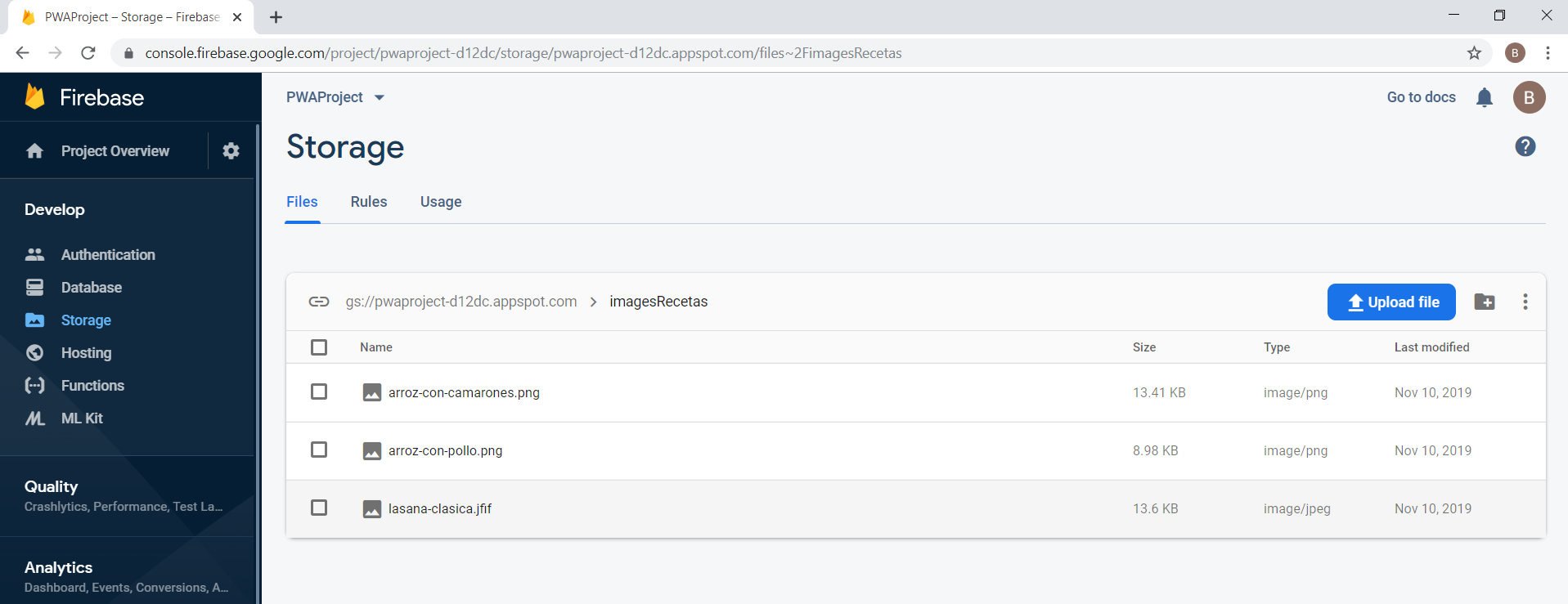
Para la persistencia de datos vamos a usar Firebase, que es un servicio de Cloud de almacenamiento y otros en línea que puede ser usado para pruebas y si se cuenta con la capacidad económica para pagar el servicio, incluso se puede hacer uso del servicio a producción.

En este caso vamos hacer uso de dos servicios, el de base de datos y el de servicio de almacenamiento. En la base de datos se van a manejar la información de las comidas como nombre del plato, descripción, ingredientes y además agregar el url de las imágenes del servicio.

**Servicio de base de datos**



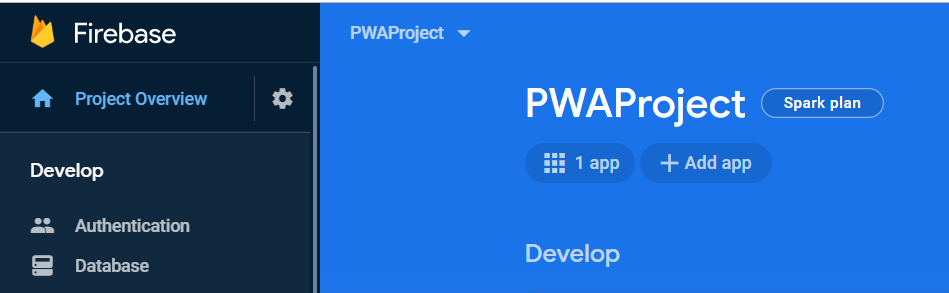
Servicio de almacenamiento para imágenes:



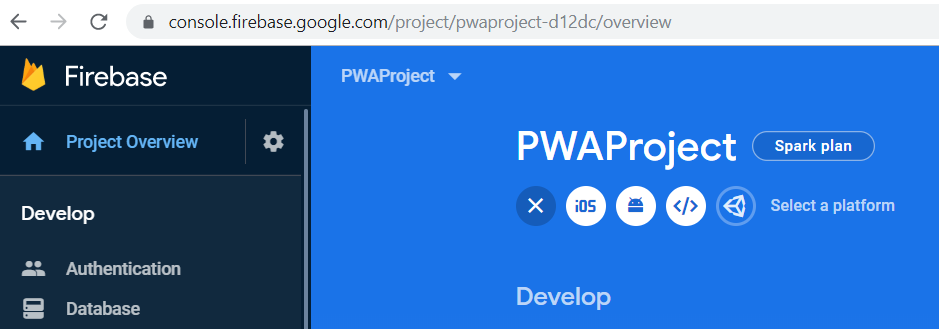
El cual nos da un url para la imagen la cual es la que damos acceso al servicio para poder mostrar la imagen con respecto al servicio.

Control de la base de datos:

Se registra la aplicación para obtener la configuración de la API para trabajar con la base de datos.



Se selecciona el tipo de aplicación, ya sea Android, Web o IOs



Y al registrar se obtiene el siguiente código de configuración:

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

        <script src="https://www.gstatic.com/firebasejs/7.2.3/firebase-app.js"></script>

        <script src="https://www.gstatic.com/firebasejs/7.2.3/firebase-firestore.js"></script>

        <script src="https://www.gstatic.com/firebasejs/7.2.3/firebase-storage.js"></script>

        <script>

            // Your web app's Firebase configuration

            var firebaseConfig = {

                apiKey: "AIzaSyCctOU99ElZ0zjSXE1bBT5woKlj-B\_MFZE",

                authDomain: "pwaproject-d12dc.firebaseapp.com",

                databaseURL: "https://pwaproject-d12dc.firebaseio.com",

                projectId: "pwaproject-d12dc",

                storageBucket: "pwaproject-d12dc.appspot.com",

                messagingSenderId: "388308114930",

                appId: "1:388308114930:web:741f46f593ee553f41df6b"

            };

            // Initialize Firebase

            firebase.initializeApp(firebaseConfig);

            const DataBaseCon = firebase.firestore();

            const Storage = firebase.storage();

        </script>

Y al final creamos el código de la base de datos:

// datos mantenidos fuera de linea

DataBaseCon.enablePersistence().catch(err => {

    if (err.code == 'failed-precondition') { // probablemente muchas ventanas abiertas a la vez

        console.log("La persistencia de datos fallo");

    } else if (err == 'unimplemented') { // falta de soporte del browser

        console.log("Persistencia no esta disponible");

    }

})

// en tiempo real traer datos con cambios nuevos al agregar o eliminar

DataBaseCon.collection('recetas').onSnapshot((snaphot) => { // console.log(snaphot.docChanges());

    snaphot.docChanges().forEach(change => { // console.log(change, change.doc.data(), change.doc.id);

        if (change.type === "added") { // add document data to web page

            renderRecetas(change.doc.data(), change.doc.id);

        } else if (change.type === "removed") { // remove the document data from webpage

            removeRecipe(change.doc.id);

        }

    });

});

// Agregar una nueva receta

const form = document.querySelector('form');

form.addEventListener('submit', evt => {

    evt.preventDefault();

    var urlImage;

    var file = document.getElementById('image');

    console.log(file.files.item(0).name);

    var receta;

    // Create a root reference

    var storageRef = firebase.storage().ref();

    storageRef.child('imagesRecetas/' + file.files.item(0).name).put(file.files.item(0)).then(() => {

        var starsRef = storageRef.child('imagesRecetas/' + file.files.item(0).name);

        console.log(urlImage);

        starsRef.getDownloadURL().then(function (url) {

            this.url = url;

            urlImage = this.url;

            console.log("teste: " + this.url);

            console.log("URL: " + urlImage);

            receta = {

                nombre: form.title.value,

                descripcion: form.description.value,

                ingredientes: form.ingredients.value,

                imagen: urlImage,

                imageName: "imagesRecetas/" + file.files.item(0).name

            };

            console.log(receta);

            DataBaseCon.collection('recetas').add(receta).catch(err => console.log(err));

            form.title.value = '';

            form.description.value = '';

            form.ingredients.value = '';

            form.image.value = '';

        });

    })

})

// Eliminar receta

const recipeContainer = document.querySelector('.recipes');

recipeContainer.addEventListener('click',evt => {

    console.log(evt);

    if(evt.target.tagName === 'I'){

        const id = evt.target.getAttribute('data-id');

        console.log(id);

        var docRef = DataBaseCon.collection('recetas').doc(id);

    docRef.get().then(function(doc) {

        if (doc.exists) {

            console.log("Document data:", doc.data().imageName);

            var imageRef = doc.data().imageName;

            var storageRef = Storage.ref();

            var desertRef = storageRef.child(imageRef);

            // Delete the file

            desertRef.delete();

            DataBaseCon.collection('recetas').doc(id).delete();

        } else {

            // doc.data() will be undefined in this case

            console.log("No such document!");

        }

    }).catch(function(error) {

        console.log("Error getting document:", error);

        DataBaseCon.collection('recetas').doc(id).delete();

    });

    }else{

        console.log("Not correct icon selected");

    }

})

En el cual mediante las conexiones creadas por la API nos permite tener los métodos de interacción con la base de datos.

**Conclusión**

Las aplicaciones progresivas web son una gran alternativa para que los productos y servicios de una empresa con presencia en web, puedan además entrar en el mercado de aplicaciones web. El simular su aplicación de una manera que el cliente pueda acceder de manera directa sin necesidad de ir al buscador brinda muchas oportunidades.

La flexibilidad y el menor costo en el desarrollo web, también hace atractivo la inversión y uso de este tipo de tecnología para cuando no se cuenta con el capital para le desarrollo de aplicaciones nativas para una plataforma en específica.

Es una tecnología interesante, en la cual los desarrolladores web pueden dar una opción flexible, de menor costo y amplio acceso para sus clientes. Esta tecnología permite que las organizaciones inviertan en una plataforma, la web, y estar presentes en móviles sin manejo de múltiples plataformas.

En conclusión, las aplicaciones web progresivas tienen beneficios de costos, presentación y facilidad de realización. Pero tiene la gran dependencia del buscador web que lo implemente, lo cual conlleva un riesgo importante, pero a pesar de ello resulta ser un recurso valioso e importante hoy en día.