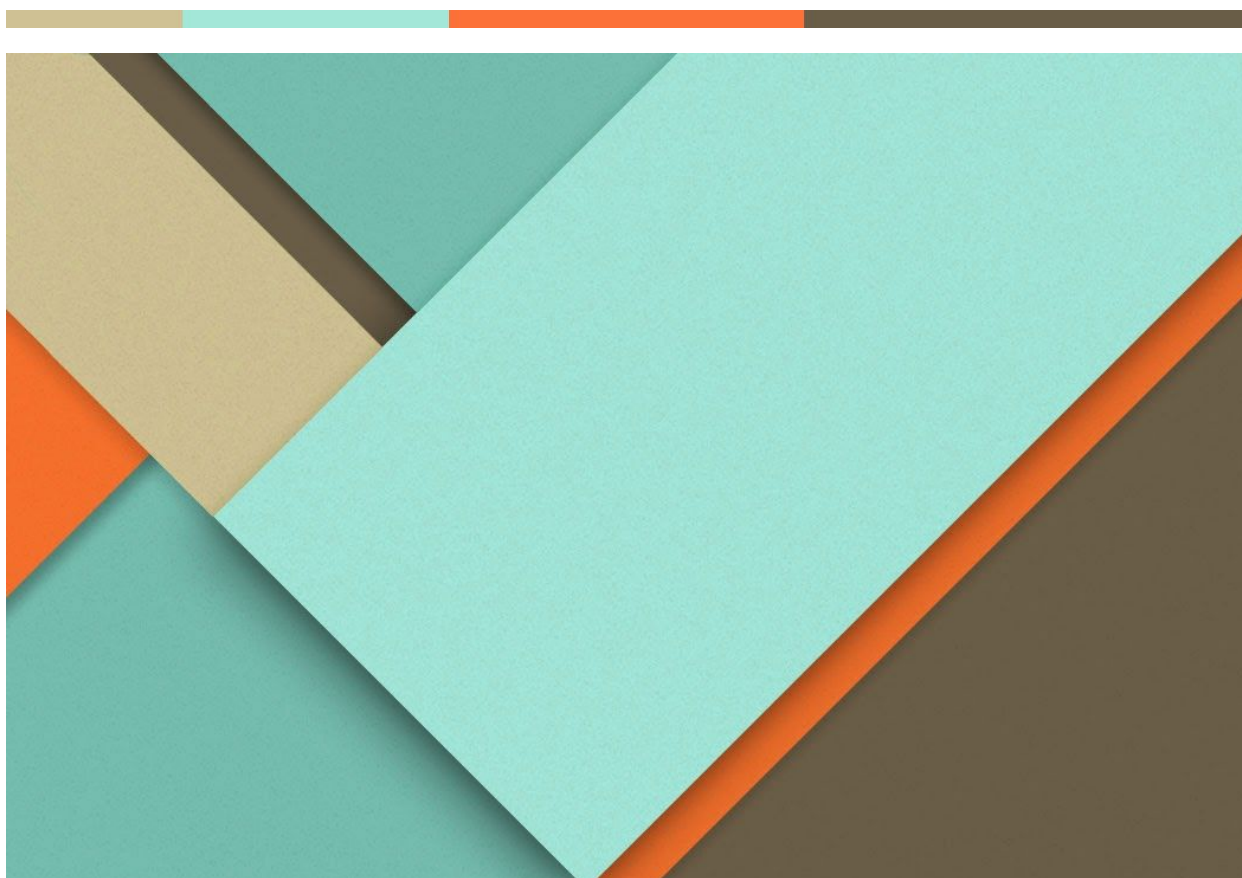


Proyecto de investigación

29/11/2019



José Luis Sánchez Delgado
CIPFP Mislata



+



+



firebase

Índice

1. Marco de la investigación
 - a. Temática elegida.
 - b. Contextualización.
2. Organización del proyecto
 - a. Recursos.
 - b. Temporalización.
3. Aplicación práctica
 - a. Introducción.
 - b. Ciclo de vida.
 - i. Análisis de requerimientos.
 - ii. Diseño.
 - iii. Implementación.
 - iv. Pruebas.
4. Manual de uso
5. Valoración personal del proyecto
6. Fuentes bibliográficas

Este proyecto se encuentra bajo la licencia de “Creative Commons” Reconocimiento-NoComercial-CompartirIgual CC-BY-NC-SA Esta licencia permite a otros modificar a partir de esta obra con fines no comerciales, siempre y cuando le reconozcan la autoría y sus nuevas creaciones estén bajo una licencia con los mismos términos.



1. Marco de la investigación.

Lo principal de este trabajo es formarme y afianzar mis conocimientos con las siguientes tecnología:

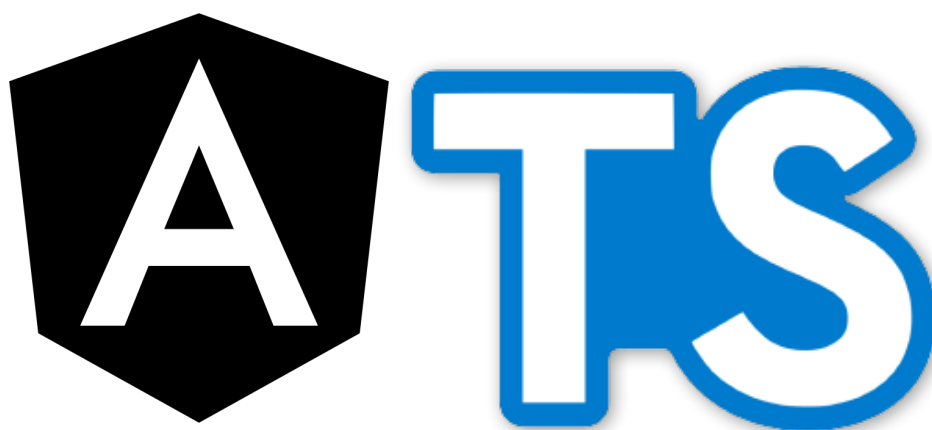
Angular

Firebase

Bootstrap

PWA

He elegido como primera opción **Angular**, ya que es un framework de código abierto desarrollado por **Google**, y me permite construir cualquier tipo de aplicación. Además es una tecnología que he utilizado bastante en mis fcts, y estoy muy familiarizado, a parte de tener conocimientos ya en **TypeScript**.



También había oído sobre **Firestore**, y la facilidad para gestionar una base de datos. Por eso he elegido esta tecnología, ya que era el momento oportuno para poder investigar sobre ella. Por lo tanto, combina bien con la **PWA** que os voy a enseñar, ya que no tengo que administrar ningún servidor, el loguearse es bastante simple, y está soportado por **Google**.



Bootstrap es una biblioteca multiplataforma para diseño de sitios y aplicaciones web. Tiene plantillas de diseño con tipografía, formularios, botones, cuadros, menús de navegación y otros elementos de diseño basado en HTML y CSS, así como extensiones de JavaScript adicionales. Solo se ocupa del desarrollo front-end.

Decidí utilizarlo, ya que me familiarice bastante en las prácticas, y me quita el problema del diseño de la interfaz, centrándome solo en la lógica.



Y finalmente, una tecnología que permite hacer aplicaciones webs progresivas(**PWA**). Esta tecnología está en auge, y se puede utilizar en cualquier lenguaje de front, ya que solo requiere de JavaScript.

Pero primero, ¿Que es una **PWA**?

Las aplicaciones web progresivas están a medio camino entre una aplicación nativa y una aplicación web. Son una evolución natural de las aplicaciones web que difumina la barrera entre la web y las aplicaciones, pudiendo realizar tareas que generalmente solo las aplicaciones nativas podían llevar a cabo. Algunos ejemplos son:

- Notificaciones
- Funcionamiento sin conexión a Internet
- Probar una versión light antes de descargar la App nativa.

Decidí poner en práctica esta tecnología, porque pese a su corta edad, creo que es una futura tecnología a tener en cuenta, dadas sus posibles aplicaciones.



1.a Temática elegida

Para mi proyecto final he decidido crear una PWA sobre una agenda para organizarse, que es un asunto que tengo pendiente.

Ya que es un tema que he decidido investigar, me he basado en aplicaciones como **Todoist** y **Do It Tomorrow**.

Algo sencillo, y que permita realizar el cometido de una agenda, que es desde recordar eventos, hasta apuntarse cosas que te vienen en el momento.

Todo esto irá ligado a que esté sincronizado a una BD y el usuario.

Puesto que una PWA puede funcionar en cualquier navegador.

Esta agenda(PWA), te permitirá en cualquier momento gestionar tus quehaceres y desde cualquier dispositivo.

Lo resumí en algo tan simple, como en tareas, y que cada usuario las gestione como crea conveniente, para apuntar el nombre de esta, su fecha e incluso si necesita detalles.

1.b Contextualización

La temática del proyecto me vino tras meses de fcts, que no tenía forma de recordar eventos importantes que no tenía apuntados por ejemplo en Facebook.

Dado que normalmente no tengo una agenda a mano, y mi mala memoria, creo que es una buena idea hacer una PWA sobre una agenda electrónica.

Además, aproveché para investigar sobre Firebase, y su potencial.

El resultado ha sido una PWA, en el cual yo puedo añadir, actualizar, eliminar y leer mis propias tareas, sin necesidad incluso de instalarme nada, ya sea en un dispositivo móvil o en un ordenador.

2. Organización del proyecto

2.a Recursos

¿Qué lenguajes he necesitado para hacer este proyecto?

Para el apartado de diseño se han utilizado los siguientes lenguajes:

HTML(HyperText Markup Language): Lenguaje de etiquetas, que se utiliza para definir la estructura básica(necesaria) de una página web.

CSS(Cascading Style Sheets): Lenguaje de diseño gráfico para definir y crear la presentación de un documento. Utilizado para establecer el diseño gráfico de los documentos web e interfaces de usuario.

CSS



HTML



JavaScript: (abreviado como JS) es un lenguaje de programación interpretado, dialecto del estándar ECMAScript 6, dado la versión de **Angular** que utilizaremos en el proyecto.

Se suele usar en el lado del cliente, implementado como parte de una web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas.

Dado que ese lenguaje lo utilizaremos básicamente para el apartado de montar la **PWA**, es muy relevante, ya que es el único lenguaje que permite trabajar de cara a las **PWA**.



TypeScript: es un lenguaje de programación libre y de código abierto desarrollado y sostenido por Microsoft.

Puede ser usado para desarrollar aplicaciones JavaScript que se ejecutarán en el lado del cliente o del servidor (Node.js).

TypeScript extiende la sintaxis de JavaScript, por tanto cualquier código JavaScript existente debería funcionar sin problemas. Está pensado para grandes proyectos, los cuales a través de un compilador de TypeScript se traducen a código JavaScript original.

Por otra parte, como extiende sintaxis de JavaScript, lo que añade es un sistema de tipos como tal, reforzando la seguridad.

TypeScript

¿Que es Angular?

Angular es un framework desarrollado en TypeScript, de código abierto y soportado por Google.

Se utiliza para crear y mantener aplicaciones web de una sola página.

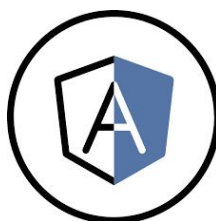
El objetivo es aumentar el uso de las aplicaciones basadas en navegador con capacidad de Modelo Vista Controlador (MVC), para hacer que el desarrollo y las pruebas sea sencillo.

Lo que hace Angular es:

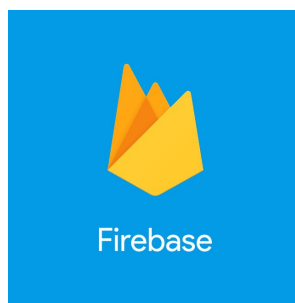
- La biblioteca lee primero el HTML que contiene atributos de las etiquetas personalizadas adicionales(específicos de Angular).
- Después va a las directivas de los atributos personalizados, y une las piezas de entrada o salida de la página a un modelo representado por las variables estándar de JavaScript.

Angular se basa en clases tipo "Componentes", cuyas propiedades son las usadas para hacer la unión de los datos. En las clases tenemos propiedades y métodos.

Angular es el siguiente paso de AngularJS aunque no es compatible con este último.



¿Que es Firebase?



Firestore es una plataforma para el desarrollo de aplicaciones web y aplicaciones móviles, y también es una base de datos NoSQL(No relacional).

Se presentan los siguientes beneficios:

- Sincronizar fácilmente los datos de los proyectos.
- Se usa fácilmente tanto como en plataformas web como en aplicaciones móviles.
- Es compatible con grandes plataformas, como IOS, Android, aplicaciones web, Unity y C++.

Usa la infraestructura de Google.

Crea proyectos sin necesidad de un servidor, lo que facilita mucho el proceso.

Firestore nos proporciona una gran documentación para crear aplicaciones.

Ofrecen soporte gratuito mediante correo electrónico y además sus desarrolladores participan en plataformas como **Github** y **StackOverflow**.

También tienen un canal de Youtube explicando el funcionamiento de varias de sus herramientas.

Gracias a todo esta herramienta, no me tengo que preocupar mucho por la parte del back, ya que Firebase me proporciona todo lo necesario, y de forma gratuita.

También he utilizado la librería AngularFire, para poder utilizar varios aspectos de Firebase en el proyecto.

AngularFire es una librería que se puede descargar desde npm.

Tiene mucha profundidad esta librería con todo lo que puede ofrecer el propio Firebase, pero en el proyecto solo se utilizará básicamente para ofrecer un sistema de login sencillo, y una conexión a nuestra base de datos para poder gestionar de forma sencilla el CRUD.

¿Que es una PWA?

Una **Progressive Web App** o **PWA** une lo mejor de una página web y de una app nativa, ofreciendo grandes beneficios tanto al usuario como a la empresa.

Como su propio nombre indica, una PWA asegura una experiencia móvil perfecta desde cualquier dispositivo y sistema operativo.

Ventajas:

- Carga más rápida.
- No hay que publicarla(Google Play o App Store), ni necesidad de descargar, ni instalar.
- Uso de cache, por lo tanto posible uso offline.
- Perfectamente adaptada para ser progresiva.
- Se puede utilizar desde cualquier navegador.
- Notificaciones push.
- Mejor posicionamiento del SEO(posición global de buscadores sobre la PWA).

Desventajas:

- Limitaciones a la hora de uso en un iOS.
- No se pueden usar todas las funciones nativas.
- Tecnología experimental.

Lo primero que entraría en juego sería declarar el **manifiesto**(Archivo json o webmanifest) en el index.html, ahí le indicamos a la web, que ya somos una PWA, con todo lo que eso conlleva. Y después tocaría el turno del **SW**.

Lo primordial de una PWA, es la implementación de los **SW**(Service Worker), que suele ser un archivo js, totalmente separado de los componentes del proyecto, que sirve para gestionar las peticiones al servidor, ya sean llamadas a una api, o a nuestra propia página, para así poder seguir utilizando la PWA, ya sea offline, o con una caída de la api.

Para utilizarlo en Angular, utilizaremos una librería que nos proporcionan ellos mismos.



Otras tecnologías

Node.js: Solo lo utilizo para usar los módulos que me ofrece, el cual ofrece paquetes para configurar los proyectos.

Entornos de desarrollo

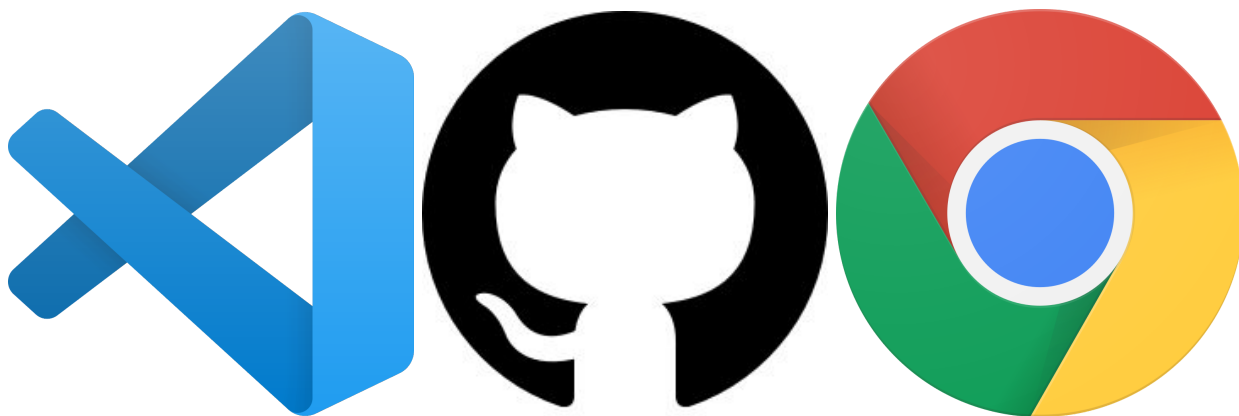
Para hacer este proyecto he trabajado con el **“IDE Visual Studio Code”**.

Permite modificarse con plugins específicos para cada lenguaje que utilices el cual facilita mucho el trabajo.

Es el editor con el que más a gusto trabajo, aunque he probado otros como Atom.

En desarrollo, lo relacionado con el uso de control de versiones con **GIT** lo he gestionado con el terminal que me proporciona el Visual Studio Code.

Por último, para poder debuggear y probar he utilizado las **herramientas de desarrollador** que me proporciona **Google Chrome** y el inspector de dispositivos móviles que te permite inspeccionar elementos HTML como se hace en WEB, pero en un dispositivo móvil.



2.b Temporalización

El desarrollo del proyecto he tenido que separarlo en tres fases donde hago la planificación, la investigación y el desarrollo del propio proyecto.

Las fases serían las siguientes:

1. Planificación de la PWA.

Esta primera fase se basa totalmente en pensar que iba a hacer.

Durante las prácticas estuve pensando una aplicación práctica sobre los conocimientos que iba adquiriendo en la empresa.

Durante el tiempo en las prácticas, repasé **Angular** que me vino muy bien para afianzar estos conocimientos.

Y después me descubrieron **ionic**, y el mundo móvil, ya que sobre todo, la empresa donde estaba, enfoca a las apps de móviles. No me disgustó aprender esta tecnología, pero no lo veía relevante para el proyecto que tenía en mente.

A la par, yo ya sabía que quería utilizar en una parte de back, **Firebase**, ya que había oído muy bien, y me empecé a leer documentación sobre Firebase.

Cuando decidí que iba a hacerlo con Firebase, pensé en cómo le podía sacar el mayor provecho en algo para front.

Entonces en la empresa me pidieron que investigara sobre las **PWA**(progressive web app), he hice varios tutoriales y me leí bastante documentación las siguientes semanas.

Cuando ví el potencia que podía alcanzar una **PWA**, supe que tenía que aprovechar estos conocimientos en el proyecto.

El resto ya se ocupó Angular y las librerías que me puede ofrecer.

2. - Elaboración de la memoria y preparación de la aplicación.

La duración de esta fase me abarcó varias semanas, ya que la memoria hay que hacerla de forma sosegada, ya que tiene cosas muy laboriosas que hacer, como lo que tienes que investigar, elaborar los diagramas, etc.

La aplicación la estuve construyendo mientras hacía la memoria, y así concreté las páginas y las funcionalidades de la PWA.

3. - Creación de la PWA.

Esta fase se me hizo mucho más amena, ya que consiste en poner en práctica todos los conocimientos adquiridos durante las prácticas y aplicarlos a un proyecto de verdad.

Empecé pensando en poner en práctica mis conocimientos sobre **ionic**, y aplicar la metodología de **Mobile First**. Pero a las semanas en la empresa, me dijeron que investigara sobre las **PWA**.

Vi la potencia que podía lograr esta tecnología, y no dude en investigar.

Al tener claro el contexto del proyecto, y con las tecnologías que iba a utilizar, decidí seguir con la metodología de **Mobile First**, ya que el punto fuerte de las PWA es poder usarse en cualquier dispositivo.

El apartado visual lo pude hacer con soltura, ya que en la empresa me dieron diversos consejos para hacer un diseño limpio y agradable.

3. Aplicación práctica

3.a Introducción

Para la introducción del proyecto comentare sobre los primeros pasos, del diseño y los diferentes tipos de componentes que forman la **PWA**.

Comentaré también los aspectos visuales y los requisitos necesarios para poder crear esta **PWA**.

El proyecto consistía en poder tramitar una agenda simple, y que cada usuario pueda crear, eliminar, actualizar y ver sus propias tareas.

Todo esto último gracias a Firebase.

Se necesitan usuarios, para usar la PWA. Ya que cada usuario tendrá sus propias tareas gestionadas, etc.

Por último, las pantallas de la que consta la PWA son: login, y página principal. Para poder entrar en la PWA será necesario haberse registrado.

En la pantalla de login, a parte de acceder, también se puede registrar. Se le hace saber al usuario en todo momento, lo que sucede con cualquier acción.

En la página principal se nos mostrará las tareas que estén asignadas a ese usuario. Y por lo tanto, el usuario podrá crear nuevas tareas, eliminarlas, e incluso modificarlas.

3.b Ciclo de vida

3.b.1 Análisis de requerimientos

Usuarios: necesitare un sistema de login para que el usuario pueda entrar en la PWA. (Lo gestiona Firebase)

Tareas: la columna vertebral de mi PWA, el usuario gestiona sus tareas, pudiendo añadir, eliminar o modificarla.

Componentes y páginas de la PWA: La PWA tendrá un provider donde se conectará a la BD.

Dentro del componente de la página principal, utilizo modales para cada apartado que se necesite.

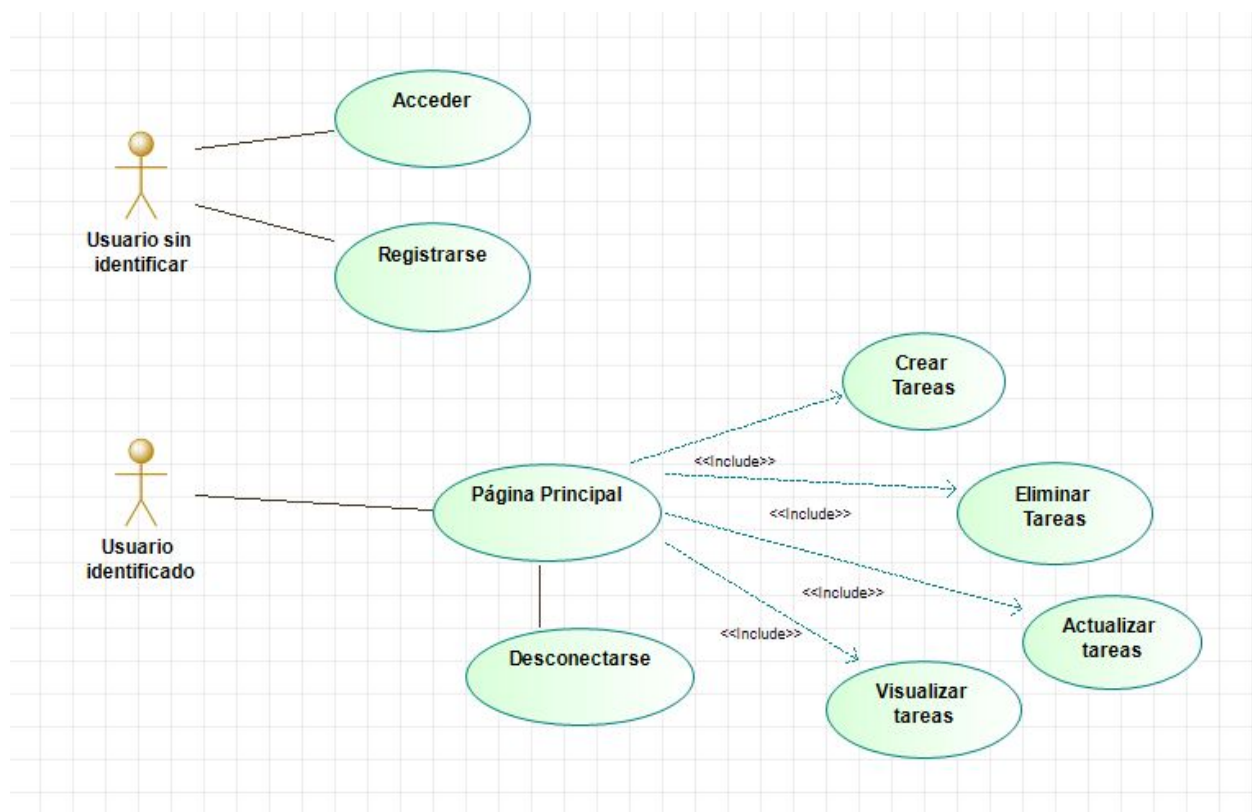
Y dentro del componente también utilizaré el servicio para las peticiones con la BD de Firebase para gestionar la información.

Gestión de PWA: este apartado lo gestionaremos en un archivo que nos genera Angular, llamado `nsgw.config.json`

Requisitos mínimos: El único requisito para poder utilizar la PWA es tener acceso a un navegador.

3.b.2 Diseño

Diagrama de casos de uso(UML)



3.b.3 Implementación

Para empezar el proyecto, primero tenemos que comprobar que tenemos todo lo necesario:

- Tener instalado node.js(Para poder utilizar el comando “npm”), lo descargamos de la página oficial.
- Para comprobar que está correctamente instalado, lanzaremos el comando node -v, para corroborar su versión.

```
C:\Users\kokek>node -v  
v10.15.0
```

- Procedemos a instalar Angular con el siguiente comando.

“npm install -g @angular/cli”

A continuación, ya podemos generar nuestro proyecto en Angular.

Dentro de la carpeta que queramos, ejecutamos el siguiente comando para crear nuestro proyecto de Angular:

ng new “nombre-del-proyecto”

```
C:\Users\kokek\Desktop\Proyectos>ng new proyectoInvestigacion
```

Lo primero que nos preguntará será si deseamos añadir enrutado a nuestro proyecto de Angular, le decimos que sí, ya que queremos que el usuario se mueva libremente entre las páginas.

```
C:\Users\kokek\Desktop\Proyectos>ng new proyectoInvestigacion  
? Would you like to add Angular routing? (y/N) y
```

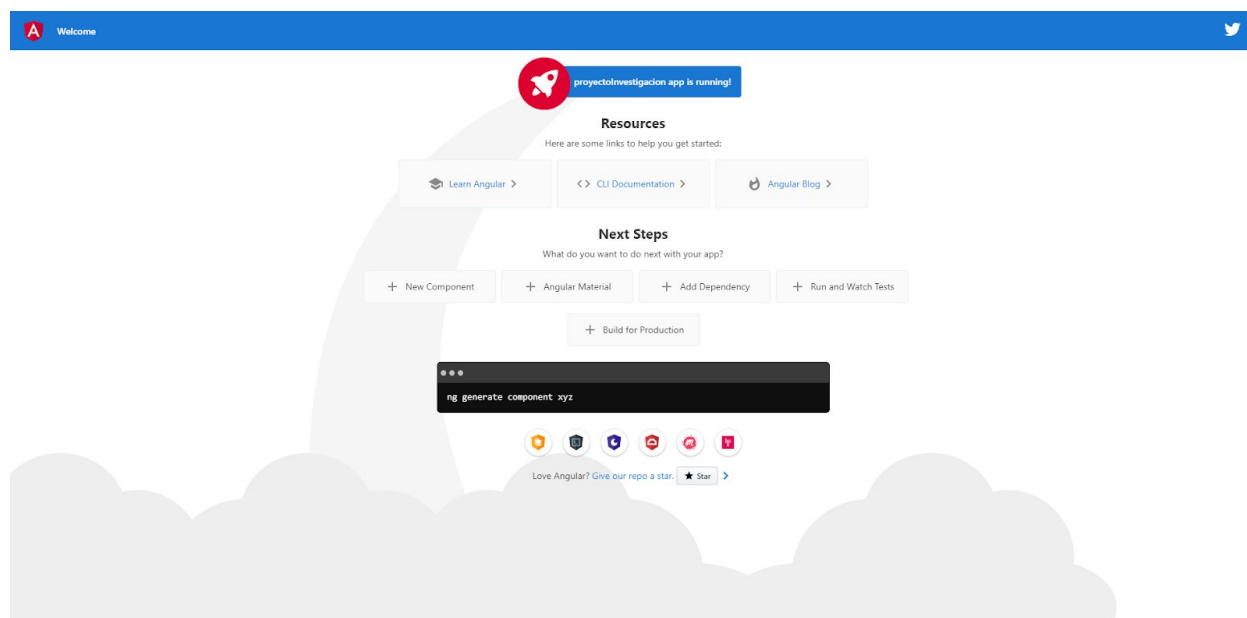

Acto seguido nos preguntará qué formato de estilo querremos utilizar, yo he utilizado **css** ya que no profundizaremos mucho en el asunto del diseño.

```
? Which stylesheet format would you like to use? (Use arrow keys)
> CSS
SCSS [ https://sass-lang.com/documentation/syntax#scss ]
Sass [ https://sass-lang.com/documentation/syntax#the-indented-syntax ]
Less [ http://lesscss.org ]
Stylus [ http://stylus-lang.com ]
```

Hechas estas preguntas, se procederá a crear el proyecto.

Para poder desplegar en local, ejecutaremos el siguiente comando:

`ng serve`



Para poder crear componentes utilizaremos el comando:

`ng generate component "nombre-componente"`

Para poder crear servicios, se utilizará:

`ng generate service "nombre-servicio"`

Convertirla a PWA

Para convertir esta aplicación en una PWA, vamos a recurrir a una librería de angular, y ejecutaremos el siguiente **comando** para añadirla a nuestro proyecto.

```
ng add @angular/pwa
```

Esta librería añade lo siguiente:

- Archivo ngsw-config.json, donde podemos configurar que guardamos y que no en la caché.
- Archivo manifest.webmanifest, donde decimos en el index.html que esto es una PWA.
- Iconos en assets, que utilizamos en el manifest.webmanifest.

Se modifica el proyecto en el index.html para introducir el manifest

Para confirmar que se instala correctamente el SW, deberíamos añadir las siguientes líneas en nuestro main.ts

```
platformBrowserDynamic().bootstrapModule(AppModule)
  .catch(err => console.error(err))
  .then(() => registerServiceWorker());

let registration = undefined;

function registerServiceWorker() {
  if ('serviceWorker' in navigator) {
    navigator.serviceWorker
      .register('/ngsw-worker.js')
```

```
.then(reg => {
  registration = reg;
  swLog('Se ha registrado correctamente', registration);
  registration.onupdatefound = () => checkServiceWorkerStateChange();
})
.catch(e =>
  console.error('Error durante el registro:', e)
);
} else {
  console.warn('SW no es soportado');
}
}

function checkServiceWorkerStateChange() {
  const installingWorker = registration.installing;

  installingWorker.onstatechange = () => {
    switch (installingWorker.state) {
      case 'installed':
        if (navigator.serviceWorker.controller) {
          swLog('Contenido nuevo disponible', installingWorker);
        } else {
          swLog('Contenido disponible offline', installingWorker);
        }
        break;
      case 'redundant':
```

```
        console.error(  
            'Redundante',  
            installingWorker  
        );  
        break;  
    default:  
        swLog(installingWorker.state);  
        break;  
    }  
};  
}  
  
function swLog(eventName, event?) {  
    console.log('Service Worker - ' + eventName);  
    if (event) {  
        console.log(event);  
    }  
}
```

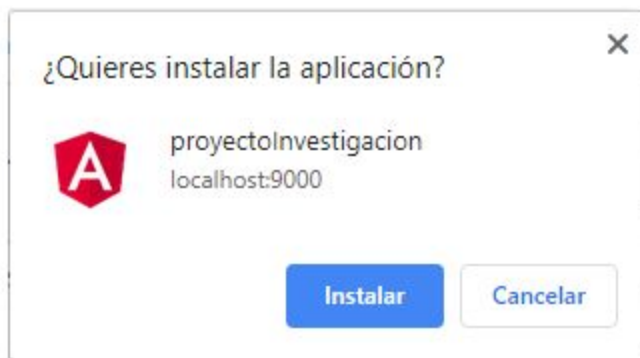
Con todo esto añadido, registramos, comprobamos actualizaciones y mostramos los eventos del SW que se vayan realizando en la consola, cuando ejecutamos la PWA en un navegador.

Service Worker - Se ha registrado correctamente	main-es2015.55de10d....js:1
▶ ServiceWorkerRegistration	main-es2015.55de10d....js:1
Service Worker - Contenido disponible offline	main-es2015.55de10d....js:1
▶ ServiceWorker	main-es2015.55de10d....js:1
Service Worker - activating	main-es2015.55de10d....js:1
Service Worker - activated	main-es2015.55de10d....js:1

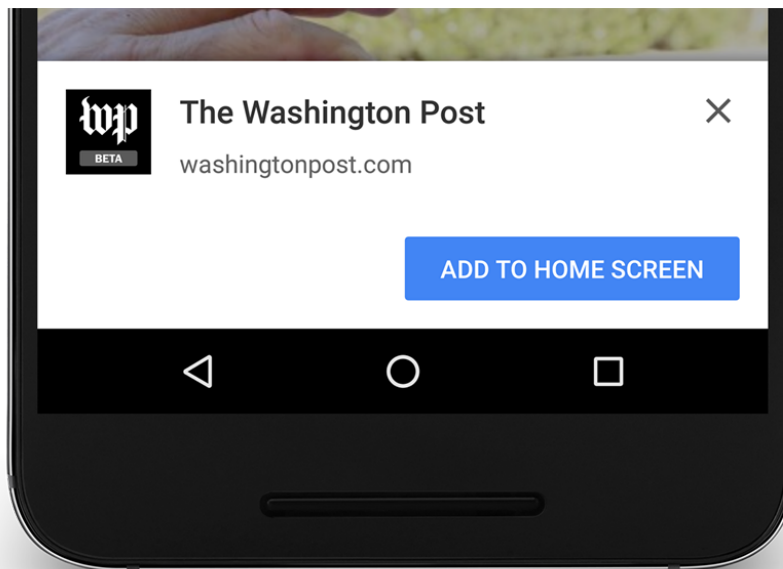
Tras haberse registrado, lo que hayamos decidido que guarde en la caché, volverá a mostrarse una vez la página esté offline.

También se reconocerá, porque en un navegador ya que nos saldría para añadir la página como aplicación de la siguiente forma.

En un navegador en escritorio:



O en un navegador de mòbil:



Por el contrario, si hay problemas con el navegador, o directamente lo lanzamos en local la **PWA**, nos mostrará parecido en la consola.

```
Angular is running in the development mode. Call  
enableProdMode() to enable the production mode.
```

```
✖ A bad HTTP response code (404) was received when fetching the script.  
✖ ▶ Error durante el registro: TypeError: Failed to register ServiceWorker for scope ('http://localhost:4200/') at http://localhost:4200/ngsw-worker.js: A bad HTTP response when fetching the script.
```

```
[WDS] Live Reloading enabled.
```

Por último para añadir correctamente Firebase en mi PWA, debo seguir los siguientes pasos:

- Añadir las siguientes líneas, debajo de mi body, en mi index.html

```
<!-- The core Firebase JS SDK is always required and must be listed first -->
<script src="/__/firebase/7.5.2/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 src="/__/firebase/7.5.2/firebase-analytics.js"></script>

<!-- Initialize Firebase -->
<script src="/__/firebase/init.js"></script>
```

- Necesito la herramienta de líneas de comandos de Firebase, lo instalaré con el siguiente comando: `npm install -g firebase-tools`
- Y también con el siguiente comando: `npm i firebase @angular/fire`
- Acto seguido importamos las librerías en "app.module"

```
import { AngularFireModule } from '@angular/fire';
import { AngularFireStoreModule } from '@angular/fire/firestore';
import { AngularFireStorageModule } from '@angular/fire/storage';
import { AngularFireAuthModule } from '@angular/fire/auth';
```

- Añadir las credenciales del proyecto en environments:

```
export const environment = {
  production: false,
  firebaseConfig: {
    apiKey: "AIzaSyBrpSimTHUrfojDS-BA3GjzKru9FfnuWDM",
    authDomain: "proyectoinvestigacion-4e50b.web.app",
    databaseURL: "https://proyectoinvestigacion-4e50b.firebaseio.com/",
    projectId: "proyectoinvestigacion-4e50b",
    storageBucket: "gs://proyectoinvestigacion-4e50b.appspot.com"
  }
};
```


- Imports en app.module

```
imports: [  
  AngularFireModule.initializeApp(config),  
  AngularFireStoreModule,  
  AngularFireStorageModule,  
  AngularFireAuthModule,
```

- Añadir AngularFireStore a providers
- Una vez lo tengamos implementado, lanzamos el comando **"firebase init"** para enlazar desde el directorio raíz al proyecto local.
- Tras configurar la conexión a nuestro gusto, lanzaremos el comando **"firebase deploy"** desde el directorio raíz del proyecto.
- De esta forma, habremos desplegado nuestra propia PWA, con cualquier URL que nos haya ofrecido por defecto el propio **Firestore** que suelen acabar en "web.app" o en "firebaseapp.com"

Bootstrap en el proyecto.

Para empezar, debemos añadir bootstrap a nuestro proyecto de la siguiente forma, a través de la línea de comandos:

```
kokek@DESKTOP-6LGHISS MINGW64 ~/Desktop/Proyectos/proyectoInvestigacion (master)  
$ npm i bootstrap
```

Luego añadimos la siguiente línea al archivo **angular.json**:

```
"styles": [  
  "node_modules/bootstrap/dist/css/bootstrap.min.css",  
  "src/styles.css"
```

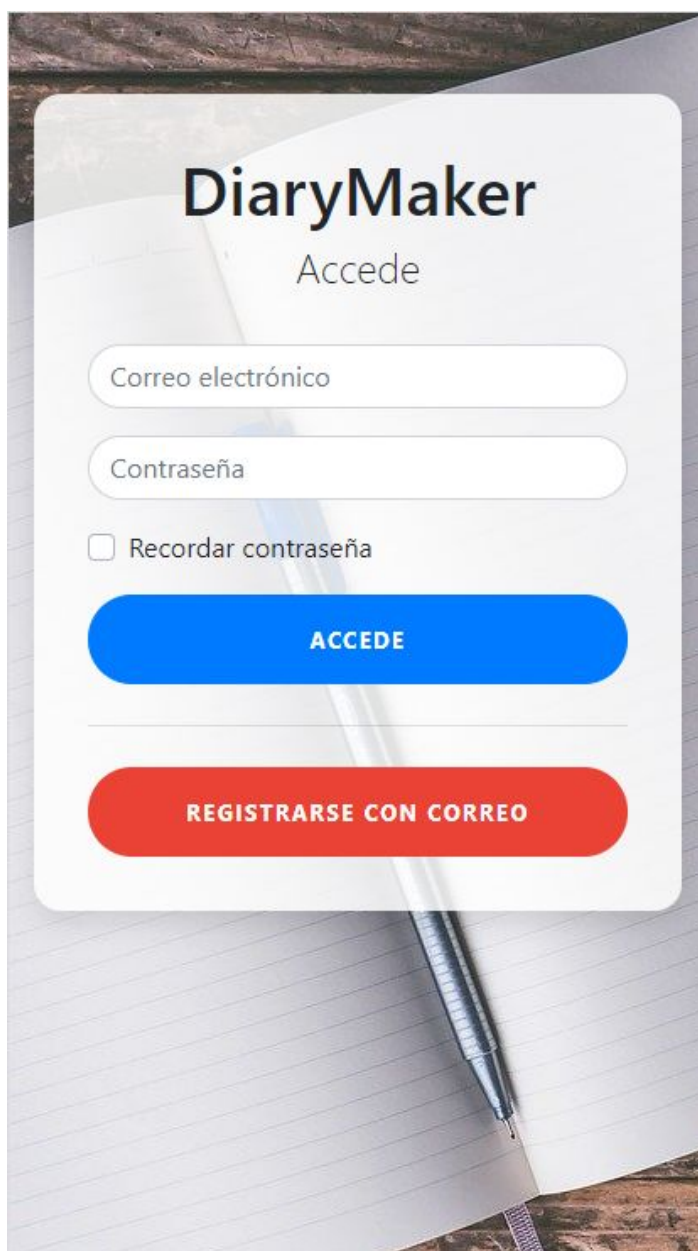

Con esto añadido, ya tendremos integrado bootstrap en nuestro proyecto.

Vamos a añadir también una librería muy especial, llamada **"ng-bootstrap"**

Esta librería nos ofrece un bootstrap específico para utilizar en angular, y que he utilizado en el proyecto para algunos aspectos estéticos, como por ejemplo, las ventanas modales.

4. Manual del usuario

Lo primero que veremos al entrar en nuestra PWA, es una ventana de inicio de sesión.



The image shows a login interface for 'DiaryMaker'. The title 'DiaryMaker' is prominently displayed at the top, followed by the word 'Accede' (Access). Below this, there are two input fields: 'Correo electrónico' (Email) and 'Contraseña' (Password). A checkbox labeled 'Recordar contraseña' (Remember password) is positioned below the password field. Two large buttons are at the bottom: a blue 'ACCEDE' button and a red 'REGISTRARSE CON CORREO' button. The entire interface is overlaid on a background image of an open notebook with a pen resting on it.

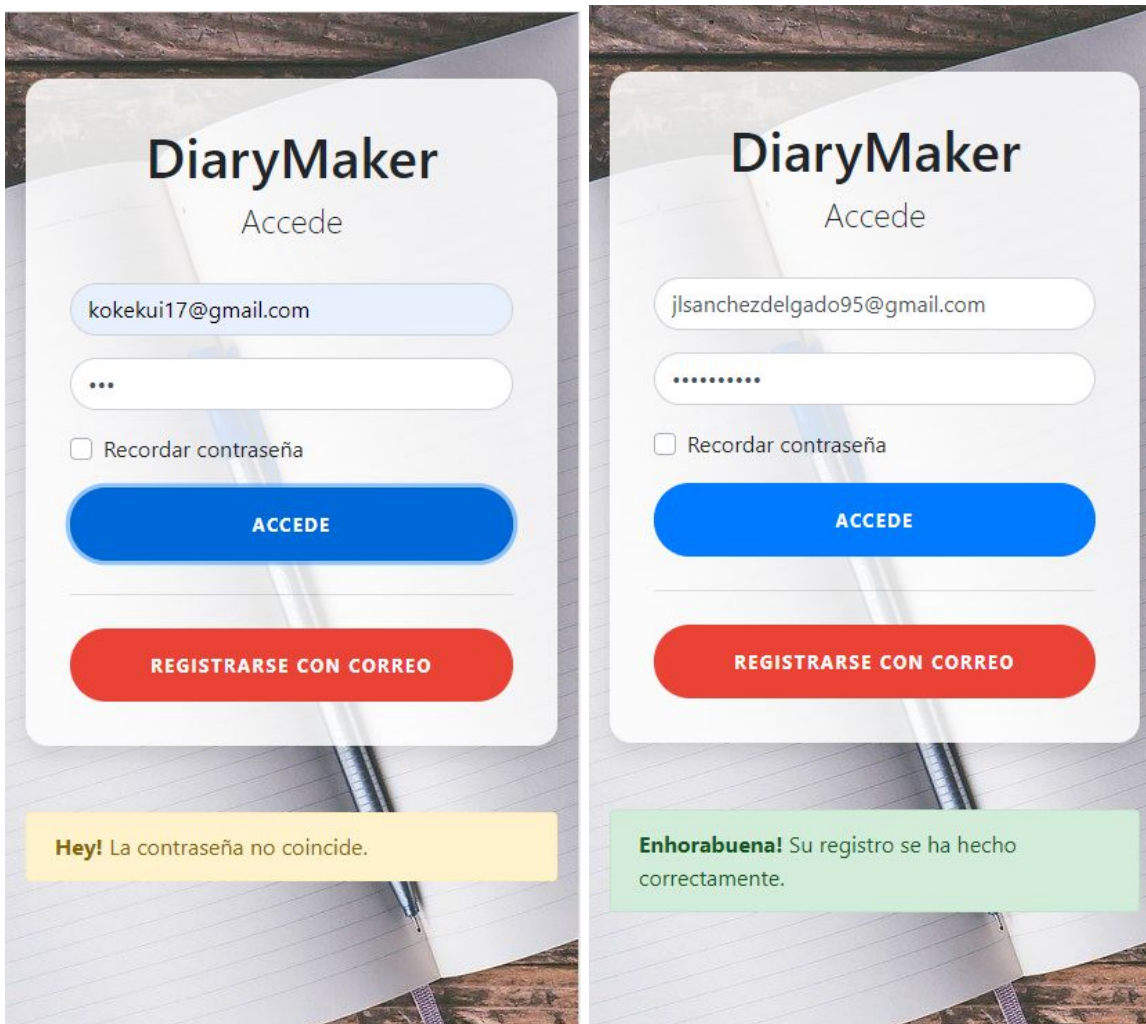
En esta primera página, se hace el esfuerzo de que sea limpia e intuitiva para ofrecer al usuario una experiencia sencilla, y con colores distintivos.

En esta ventana se pueden hacer dos cosas, o acceder con una cuenta ya registrada, al botón de “accede”, o en el caso contrario, registrarte.

En ambos casos se requieren los mismos campos, correo electrónico y contraseña.

En cualquiera de las dos acciones, si hay algún error saltará una alerta advirtiéndolo de los problemas, y por el contrario, si accedes también saltará una alerta diciendo que ha sido un éxito.

Algunos ejemplos de las alertas:



The image displays two side-by-side screenshots of the DiaryMaker application interface, which is overlaid on a background of an open notebook and a pen. Both screens feature the 'DiaryMaker' title and the 'Accede' (Login) button. The left screen shows a login attempt with the email 'kokekui17@gmail.com' and a password field containing three dots. Below the fields is a checkbox for 'Recordar contraseña' (Remember password) and a red 'REGISTRARSE CON CORREO' (Register with email) button. A yellow error message at the bottom states: 'Hey! La contraseña no coincide.' (Hey! The password does not match). The right screen shows a login attempt with the email 'jlsanchezdelgado95@gmail.com' and a password field containing eight dots. It also has the 'Recordar contraseña' checkbox and the red 'REGISTRARSE CON CORREO' button. A green success message at the bottom states: 'Enhorabuena! Su registro se ha hecho correctamente.' (Congratulations! Your registration has been completed correctly).

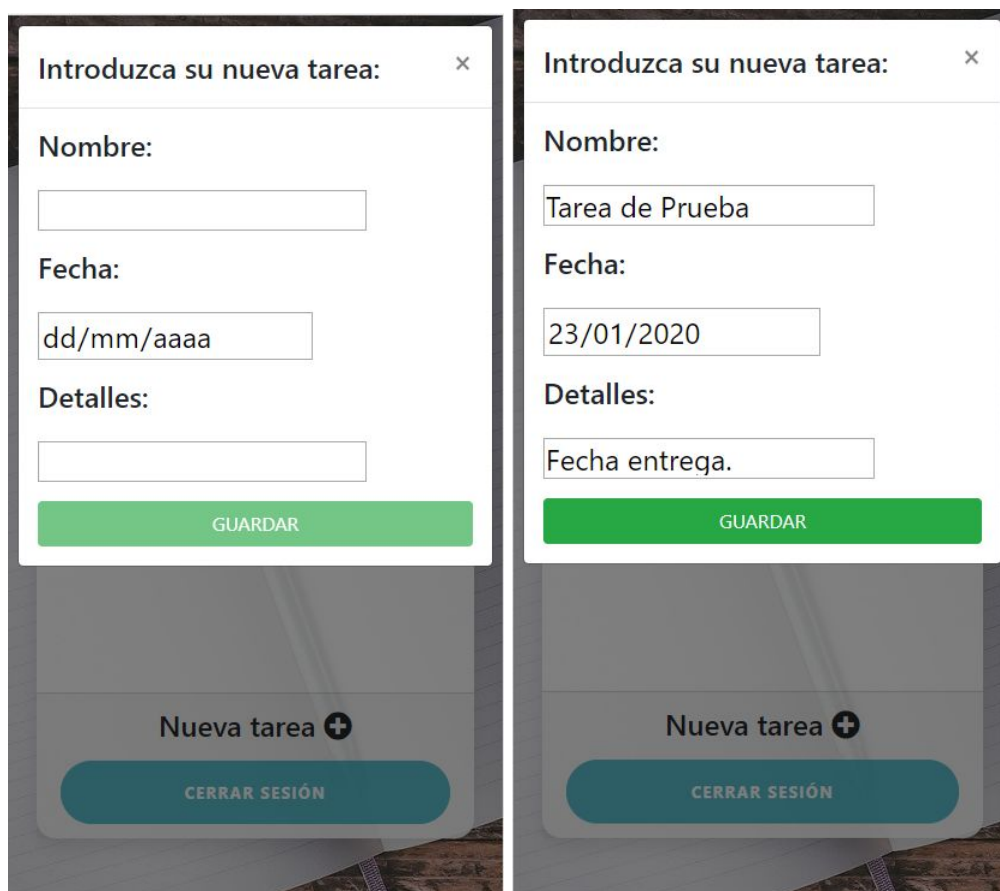
Una vez hayamos accedido correctamente, nos encontraremos en la página principal en el que podremos ver una modal dándonos la bienvenida.

Tendremos una tabla donde se mostrará nuestras tareas, que podremos modificar, eliminar, y también añadir nuevas tareas.

Y por supuesto, un botón para poder cerrar la sesión.



Al haber creado un usuario nuevo, este no tiene tareas asignadas a él. Por lo tanto, vamos a hacer algún ejemplo de cómo se crearía una nueva tarea. Al darle a “Nueva tarea”, se nos mostrará una ventana modal, con un formulario sobre nuestra nueva tarea. Se requieren los campos nombre y fecha, para poder tener una nueva tarea, de lo contrario, el botón de guardar estaría deshabilitado.



Introduzca su nueva tarea: ✕

Nombre:

Fecha:

Detalles:

GUARDAR

Nueva tarea +

CERRAR SESIÓN

Introduzca su nueva tarea: ✕

Nombre:

Fecha:

Detalles:

GUARDAR

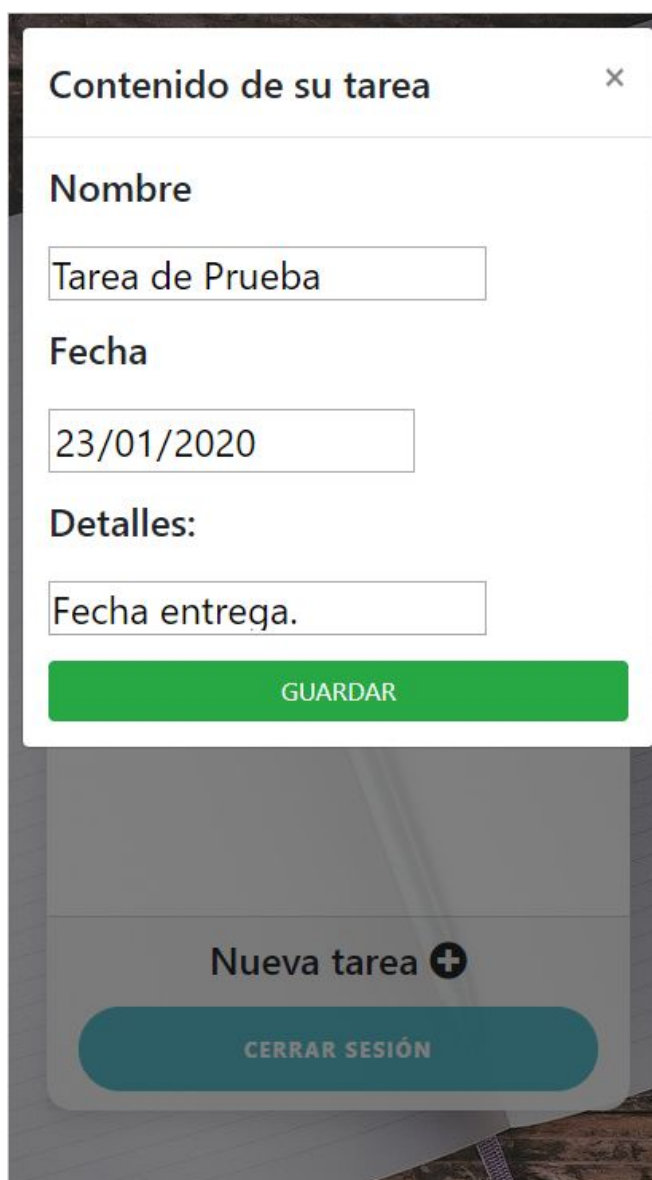
Nueva tarea +

CERRAR SESIÓN

En el caso de que se haya creado correctamente, se nos mostrará la siguiente modal:



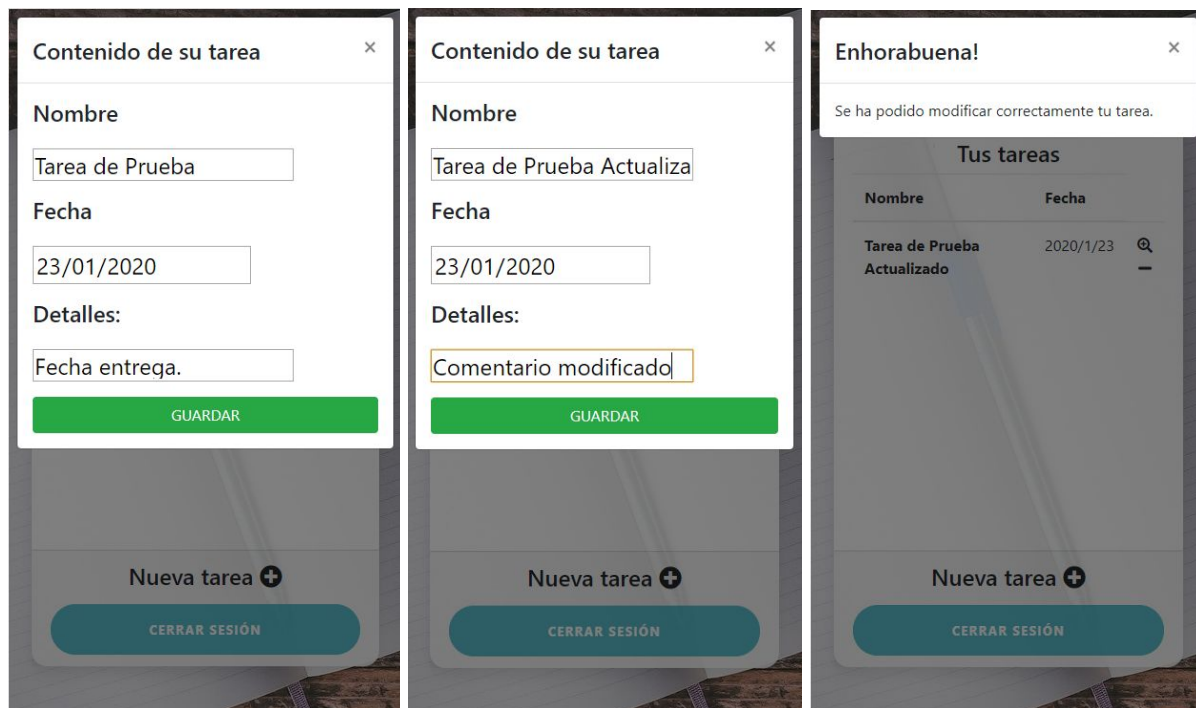
Ya que nos encontramos ante una metodología de **Mobile First** para esta PWA, no se puede mostrar toda la información en la tabla principal, por lo tanto, en la lupa que nos aparecerá en la tabla, podremos ver los detalles de cada tarea.



The screenshot shows a mobile application interface with a modal window titled "Contenido de su tarea" (Content of your task). The modal contains the following fields and buttons:

- Nombre** (Name): A text input field containing "Tarea de Prueba" (Test Task).
- Fecha** (Date): A date input field containing "23/01/2020".
- Detalles:** (Details): A section header.
- Fecha entrega.** (Delivery date): A text input field.
- GUARDAR** (Save): A green button.
- Nueva tarea +** (New task): A button with a plus icon.
- CERRAR SESIÓN** (Log out): A button.

En esta ventana también podemos editar la propia información de la tarea, si cambiáramos cualquier campo, se modificaría correctamente, y saldría esta ventana modal:



The image displays three sequential screenshots of a mobile application interface for task management.

First Screenshot: 'Contenido de su tarea' modal. This screen allows editing task details. It includes fields for 'Nombre' (Task de Prueba), 'Fecha' (23/01/2020), and 'Detalles' (Fecha entrega.). A green 'GUARDAR' button is at the bottom. The background shows a 'Nueva tarea +' button and a 'CERRAR SESIÓN' button.

Second Screenshot: 'Contenido de su tarea' modal. This screen shows the updated task details. The 'Nombre' field now contains 'Tarea de Prueba Actualiza', and the 'Detalles' field contains 'Comentario modificado'. The 'GUARDAR' button remains at the bottom.

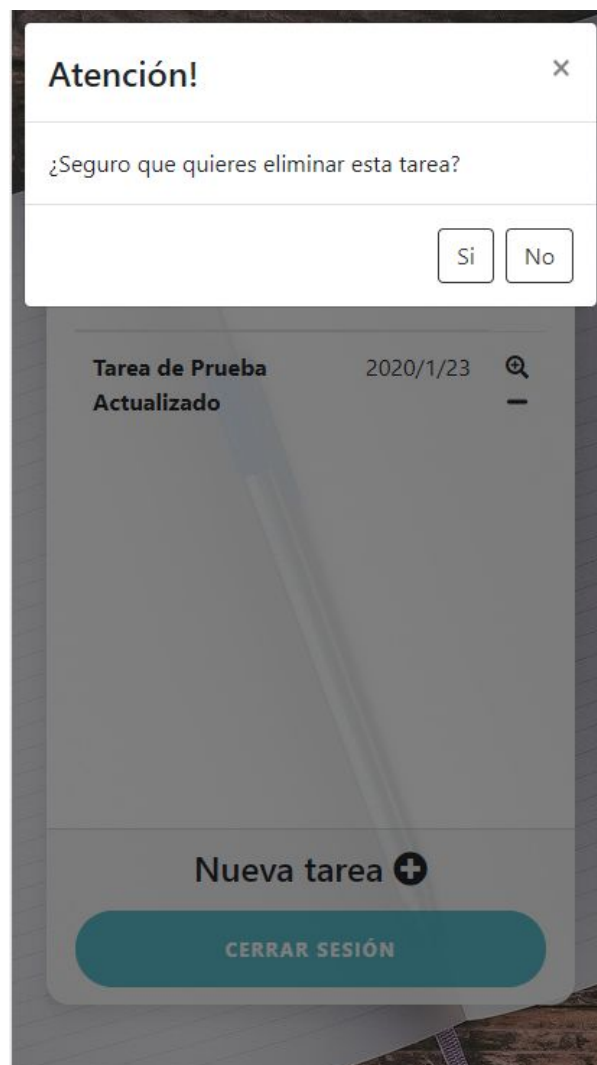
Third Screenshot: 'Enhorabuena!' modal. This screen displays a success message: 'Se ha podido modificar correctamente tu tarea.' Below the message is a table titled 'Tus tareas'.

Nombre	Fecha
Tarea de Prueba Actualizado	2020/1/23

The 'Enhorabuena!' modal also features a 'Nueva tarea +' button and a 'CERRAR SESIÓN' button at the bottom.

Por último tendríamos la opción de eliminar la tarea, sería algo tan simple como darle al símbolo de “-” que tenemos en la vista principal de todas nuestras tareas.

Nos saldría la siguiente ventana modal, para confirmar si queremos eliminar esa tarea:



5. Valoración personal del proyecto

Hacer este proyecto me ha gustado por diversos factores.

Ha sido muy fructífero el poder poner en práctica los conocimientos adquiridos en las prácticas, ya sea con **Angularjs**, **Bootstrap** y la tecnología **PWA**.

Como aprender conocimientos por cuenta propia con Firebase, que he aprendido bastante, gracias a los tutoriales que me han podido brindar.

Pese a que fue muy cargante todo el proceso de pensar en cómo enfocar el proyecto e investigar nuevas tecnologías por cuenta propia, te instiga a seguir mirando más, de cara a mejorar tu propio proyecto, ya que, por un lado o por otro, sabes sobre ese tipo de lenguajes.

Lo que más fatigoso del proyecto, fue investigar sobre **Firebase**, pese a que aporta muchas facilidades, se me hizo un mundo al tocar mi primera BD NoSQL, y demás.

El trabajo de este proyecto ha sido para poner en práctica, tanto los conocimientos adquiridos en las **FCTs** como conocimientos que puedo adquirir por mi cuenta.

Aunque no haya podido plasmar en el proyecto conocimientos como los adquiridos con **ionic**, este lenguaje me ayudó a comprender la metodología **Mobile First**.

Por último, quiero reconocer el nivel de profesionalidad que he podido alcanzar gracias al ciclo.

Ya que he podido observar con mis propios ojos lo bien preparados que salimos de este centro, y el nivel de otros centros.

6. Bibliografía

- Página oficial de Angular:
<https://angular.io/>
- Página oficial de Firebase:
<https://firebase.google.com/>
- Página oficial de Bootstrap:
<https://getbootstrap.com/>
- Página de rxjs:
<https://rxjs-dev.firebaseapp.com/>
- Página oficial ng-bootstrap:
<https://ng-bootstrap.github.io/#/home>
- Tutorial Firebase Authentication:
<https://www.positronx.io/firebase-authentication-in-angular-8-with-angularfire2/>
- Tutorial Angular 8 y Firestore:
<https://medium.com/angular-chile/angular-6-y-firestore-b7f270adcc96>

- PWA vs App:

<https://blog.wemake.pe/pwa-vs-apps-nativas/>

- Pixabay para imágenes:

<https://pixabay.com/es/>