Aplicaciones Distribuidas en Internet

Notificaciones push PWA

Francisco García Mora

DNI: 48780988-N

Grupo de prácticas: Martes 9:00-11:00

ADI

Ingeniería Informática

ÍNDICE

1. Añadir dependencias. 2
2. Crear el servicio. 2
3. Añadir configuración de firebase. 3
4. Servicios auxiliares. 3
5. Modificaciones en el proyecto. 4
- App.module.ts 4
- Angular.json 4
- Angular.json 4

6. Implementación notificaciones push 4

Francisco García Mora 48780988-N
Grupo de Prácticas: Martes 9:00 - 11:00
Fecha de entrega: Lunes 26 de octubre de 2020.

1. Añadir dependencias.

En primer lugar debemos añadir las dependencias de firebase y PWA. Para ello en este tutorial partimos de un proyecto nuevo, este es el primer paso a realizar en cualquier proyecto.

Para añadir las dependencias nos situamos en la carpeta raíz del proyecto y ejecutamos en este orden:

- ng add @angular/fire
- ng add @angular/pwa

2. Crear el servicio.

Ahora crearemos un servicio de mensajería que será el que gestione las notificaciones recibidas:

- ionic g s services/messaging

El código de este servicio será el siguiente:

```
import { Injectable } from '@angular/core';
import { AngularFireMessaging } from '@angular/fire/messaging';
import { tap } from 'rxjs/operators';

@Injectable({
  providedIn: 'root'
})
export class MessagingService {

  token = null;

  constructor(
    private afMessaging: AngularFireMessaging
  ) { }

  requestPermission(){
    return this.afMessaging.requestToken.pipe(
      tap( token => {
        console.log("Store the token: " + token);
      })
    )
  }

  getMessages(){
    return this.afMessaging.messages;
  }

  deleteToken(){
    if(this.token){
      this.afMessaging.deleteToken(this.token);
      this.token = null;
    }
  }

}
```

3. Añadir configuración de firebase.

Para poder trabajar con firebase, una vez tenemos creado el servicio de mensajería, debemos modificar los archivos de enviroment (enviroments.ts y enviroments.prod.ts) para que firebase trabaje con nuestros datos:

```
firebase: {
    apiKey: "[redacted]",
    authDomain: "[redacted]",
    projectId: "[redacted]",
    storageBucket: "[redacted]",
    messagingSenderId: "[redacted]",
    appId: "[redacted]",
    measurementId: "[redacted]"
}
```

*En este archivo debéis especificar vuestros propios datos obtenidos en el proyecto de firebase.

4. Servicios auxiliares.

Tendremos que crear dos servicios auxiliares para configurar adecuadamente nuestro proyecto, uno conectará con firebase para obtener las notificaciones push, y el otro hará que podamos trabajar con más service workers adicionales.

Dicho esto procedemos a crear el archivo /src/firebase-messaging-sw.js que tendrá el siguiente código:

```
importScripts('https://www.gstatic.com/firebasejs/8.2.1/firebase-app.js');
importScripts('https://www.gstatic.com/firebasejs/8.2.1/firebase-analytics.js');
importScripts('https://www.gstatic.com/firebasejs/8.2.1/firebase-messaging.js');

firebase.initializeApp({
    apiKey: "[redacted]",
    authDomain: "[redacted]",
    projectId: "[redacted]",
    storageBucket: "[redacted]",
    messagingSenderId: "[redacted]",
    appId: "[redacted]",
    measurementId: "[redacted]"
});

const messaging = firebase.messaging();
```

El otro archivo a crear es /src/combine-sw.js que tendrá únicamente los imports de los dos service workers existentes, el que ya teníamos en el proyecto y el que acabamos de crear:

```
importScripts('ngsw-worker.js');
importScripts('firebase-messaging-sw.js');
```

5. Modificaciones en el proyecto.

- ## App.module.ts

En primer lugar modificaremos el archivo app.module.ts sustituyendo el serviceWorker existente por este otro, añadiendo además las otras dos líneas siguientes.

```
  ServiceWorkerModule.register('combine-sw.js', { enabled: environment.production }),
  AngularFireModule.initializeApp(environment.firebase),
  AngularFireMessagingModule,
```

- ## Angular.json

En el archivo angular.json agregar estas dos líneas justo después de la línea "src/manifest.webmanifest",

```
  "src/combine-sw.js",
  "src/firebase-messaging-sw.js"
```

- ## Angular.json

En el archivo angular.json agregar al inicio la siguiente línea (el dato se extrae del proyecto en firebase):
`"gcm_sender_id": "XXXXXXXXXXXX"`

6. Implementación notificaciones push

Por último, implementamos en la página que queramos la lógica necesaria para mostrar las notificaciones push, todo este código se extrae de firebase, puede sufrir alguna modificación con cada actualización.

```
import { Component } from '@angular/core';
import { AlertController, ToastController } from '@ionic/angular';
import { MessagingService } from '../services/messaging.service';

@Component({
  selector: 'app-home',
  templateUrl: 'home.page.html',
  styleUrls: ['home.page.scss'],
})
export class HomePage {

  constructor(
    private messagingService: MessagingService,
    private alertCtrl: AlertController,
    private toastCtrl: ToastController,
  ) {
    this.listenForMessages();
  }

  listenForMessages(){
    this.messagingService.getMessages().subscribe( async (msg: any) => {
      console.log("NEW MESSAGE: ", msg);

      const alert = await this.alertCtrl.create({
        header: msg.notification.title,
```

```
        subHeader: msg.notification.body,
        message: msg.data.info,
        buttons: ['OK'],
      });

      await alert.present();
    });
  }

  requestPermission(){
    this.messagingService.requestPermission().subscribe(
      async token => {
        const toast = await this.toastCtrl.create({
          message: 'Got your token',
          duration: 2000
        });
        toast.present();
      },
      async (err) => {
        const alert = await this.alertCtrl.create({
          header: 'Error',
          message: err,
          buttons: ['OK'],
        });
      }
    )
  }

  async deleteToken(){
    this.messagingService.deleteToken();
    const toast = await this.toastCtrl.create({
      message: 'Token removed',
      duration: 2000
    });
    toast.present();
  }

}
```

VÍDEO TUTORAL YOUTUBE: https://youtu.be/_6L58ImsVCU