

UNIT 3

Ionic



Annex **Push notifications**

Client-side Web Development
2nd course – DAW
IES San Vicente 2020/2021
Author: Arturo Bernal Mayordomo

Index

Push Notifications.....	3
Receiving push notifications.....	4
Sending notifications from Firebase.....	5
Creating services.....	6
Example guide.....	9


Push Notifications

Push notifications are messages that usually a server sends to the client application (reminders, general information, messages from other user, events, etc.). These messages can have multiple fields (title, subtitle, image, icon, payload data, ...).

To implement Push notifications, we are going to use [Firebase](#) and the [Capacitor Push Notifications Plugin](#). the first thing to do is create or add an existing project in the [Firebase Console](#). After adding the project, create an Android application.

Agregar un proyecto

Nombre del proyecto

 DWEC 2017

Sugerencia: Los proyectos llevan las apps a distintas plataformas

ID del proyecto

dwec-2017-1507729559860

País/Región

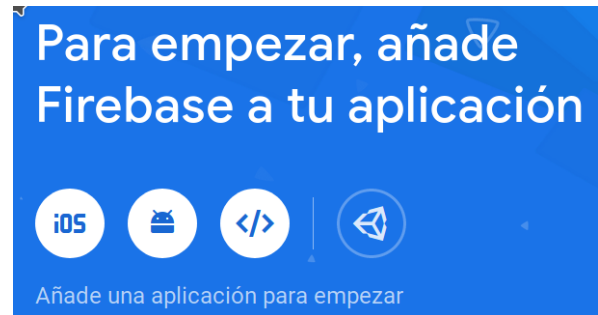
España

En la configuración predeterminada, tus datos de Analytics mejorarán otras funciones de Firebase y otros productos de Google. Puedes controlar cómo se comparten tus datos de Analytics en cualquier momento desde la configuración. [Más información](#)

Si continúas y haces clic en el siguiente botón, reconoces que usas los servicios de Firebase en tu app y aceptas las [condiciones](#) aplicables.

CANCELAR

AGREGAR FIREBASE



The configuration will ask you to write your app ID (package name). See the manifest or build.gradle (app) to check the ID.

[illegible]

Download **google-services.json** file and put it inside the root folder of you Android project (inside **android/app**).

Receiving push notifications

Before receiving any push notification, we have to register the client app with Firebase. To do that we'll need to call the register method. If we register successfully, Firebase will send us a token (it identifies device and app). We'll send that token to our server because that's the thing it'll need to send us notifications.

For example, we can register our device when we log in:

```
...
import {
  Plugins,
  PushNotificationToken
} from '@capacitor/core';
const { PushNotifications } = Plugins;

...
export class LoginPage implements OnInit {
  ...
  firebaseToken = null;
  ...
  ngOnInit() {
    PushNotifications.register();

    // On success, we should be able to receive notifications
    PushNotifications.addListener('registration',
      (token: PushNotificationToken) => {
        this.firebaseToken = token.value;
      }
    );
  }

  login() {
    this.authService.login(this.email, this.password,
      this.firebaseToken).subscribe(...);
  }
}
```

Then, in the app component (globally for all the app), we can listen to the events the fire when a notification is received and the app is running, or when the app is in background (or closed). If the app is not open, we'll receive a notification on our device and when tapping on it, it will open the app.

```
...
import {
  ...
  PushNotification,
  PushNotificationActionPerformed
} from '@capacitor/core';
const { ..., PushNotifications } = Plugins;
...
export class AppComponent {
  ...
  initializeApp() {
    ...

    // Show us the notification payload if the app is open on our device
    PushNotifications.addListener('pushNotificationReceived',
      async (notification: PushNotification) => {
        const toast = await this.toast.create({
          header: notification.title,
```

```

        message: notification.body,
        duration: 3000
    });
    await toast.present();
}
);

// Method called when tapping on a notification
PushNotifications.addListener('pushNotificationActionPerformed',
(notification: PushNotificationActionPerformed) => {
    if (notification.notification.data.prodId) {
        this.nav.navigateRoot(['/products', 'details',
notification.notification.data.prodId, 'comments']);
    }
}
);
}

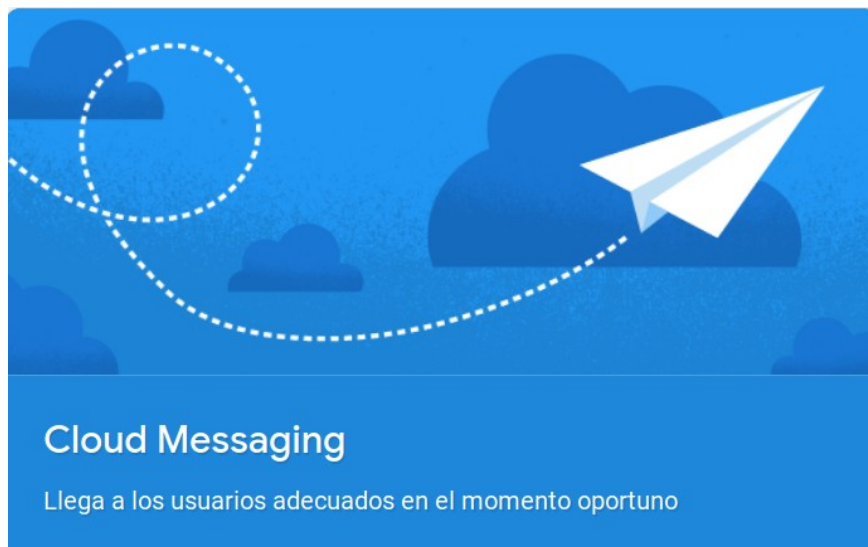
async logout() {
    await this.authService.logout();
    this.nav.navigateRoot(['/auth/login']);
}
}

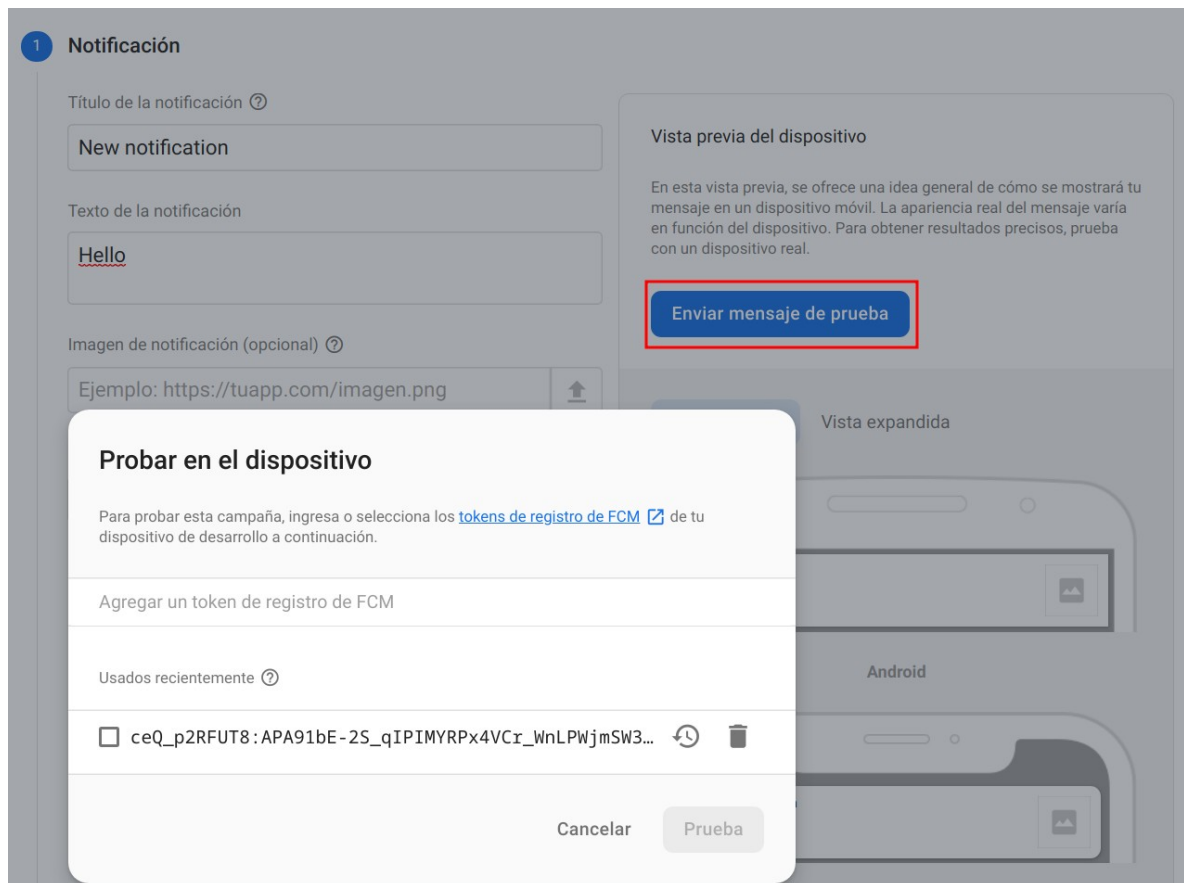
```

When receiving a notification in background, the object is different (there's no title or body, but the data is there. If you want to see what's inside the object, do a console.log first for testing.

Sending notifications from Firebase

You can send a test notification from your Firebase console, by going to the Cloud Messaging section.



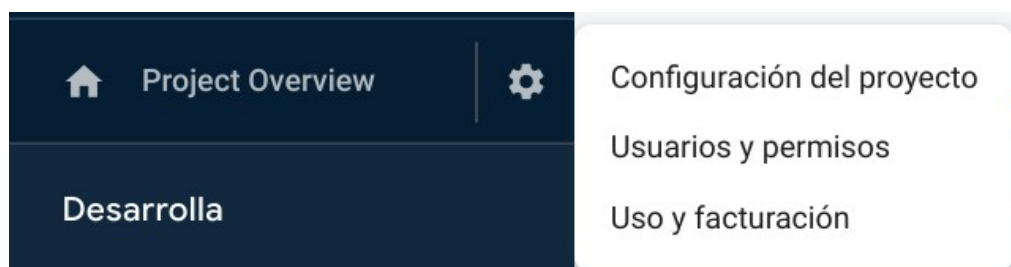


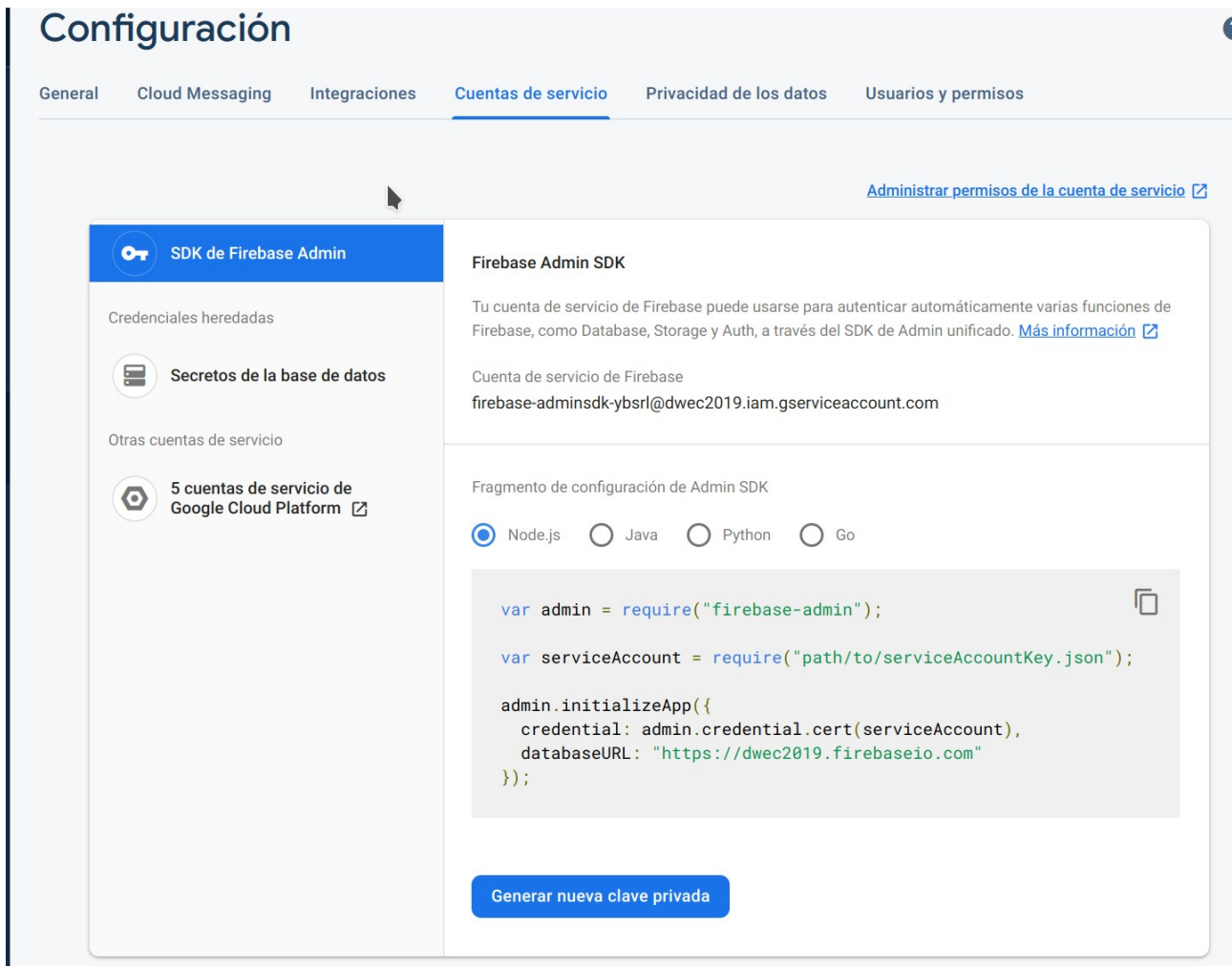
Creating services

For sending notifications in Node.js (or any framework like Nestjs), you must install the Firebase Admin package:

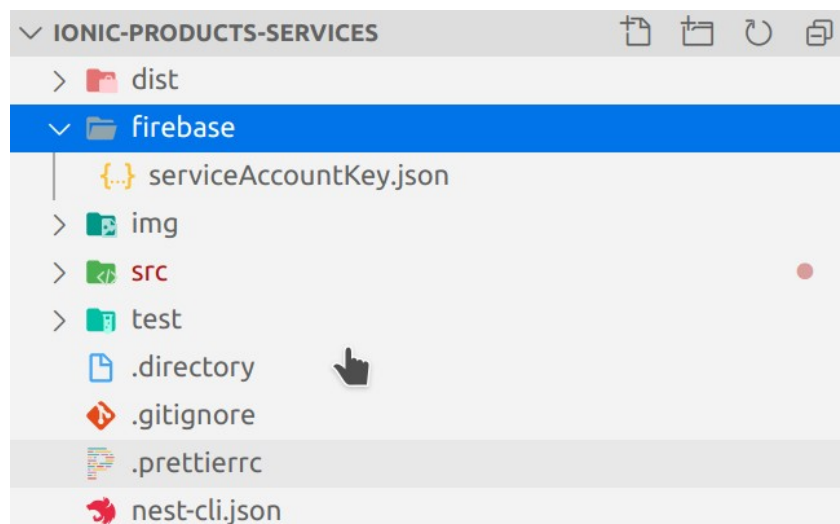
npm install firebase-admin

Now, we are going to generate a private key for our services, so they can connect to Firebase and send push notifications to registered devices. Go to project configuration → Service Accounts:





Generate a new private key and save it inside your services root directory (I've placed it inside a folder called firebase):



Then, we can import it and configure our app (in Nest I've done it inside the main.ts file):

```

import { NestFactory } from '@nestjs/core';
import { AppModule } from './app.module';
import * as express from 'express';
import * as admin from 'firebase-admin';
import { useContainer } from 'class-validator';

// tslint:disable-next-line:no-var-requires
const serviceAccount = require('../firebase/serviceAccountKey.json');

async function bootstrap() {
  const app = await NestFactory.create(AppModule, { cors: true });
  app.use('/img', express.static('img'));
  app.use(express.json({ limit: '10mb' }));
  useContainer(app.select(AppModule), { fallbackOnError: true });

  admin.initializeApp({
    credential: admin.credential.cert(serviceAccount),
    databaseURL: 'https://dwec2019.firebaseio.com',
  });

  await app.listen(4000);
}
bootstrap();

```

I've also created a service to send notifications:

```

import { Injectable } from '@nestjs/common';
import * as admin from 'firebase-admin';

@Injectable()
export class FirebaseService {
  constructor() {}

  async sendMessage(token: string, title: string, body: string, data: any) {
    const message = {
      notification: {
        title,
        body,
      },
      data,
      token,
    };

    try {
      const resp = await admin.messaging().send(message);
      return resp;
    } catch (e) {
      return null;
    }
  }
}

```

And when creating a new comment on a product, it will send a notification to the owner of that product (if he/she has a firebase token):

```

...

@Injectable()
export class ProductsService {
  ...
  async insertComment(commentDto: AddCommentDto) {
    const result = await this.commentRepo.insert(commentDto);
    const prod = await this.productRepo.findOne(commentDto.product,
    {loadRelationIds: true});
    const user = await this.userRepo.findOne(prod.creator);
    if (user.firebaseToken) {
      await this.firebaseService.sendMessage(

```



```

        user.firebaseToken,
        `New comment (${prod.description})`,
        commentDto.text,
        { prodId: ' ' + prod.id }
    );
}
return await this.commentRepo.findOne(result.identifiers[0],
                                     {relations: ['user']});
}
}

```

Example guide

There's an example guide about configuring Android and iOS with capacitor and Push notifications here:

<https://capacitor.ionicframework.com/docs/apis/push-notifications>