Overview

Welcome to the smartest online dating app that actually works and sets you up with perfect matches. Ionic Dating is a free local dating app that enables you to browse through available local singles nearby, check out their profiles, and start the conversation with those who you like.

So, if you are looking for an online dating app to meet new people and find single girls and boys nearby, download Ionic Dating for free on your Android phone or tablet, complete your profile and let our powerful matchmaking engine do the wingman job for you.

We've developed this app to make sure that you can easily get started with a feature-specific dating app. It's smooth, suave, sophisticated and great for creating the kind of atmosphere needed when you want to help people get in the mood and groove for dating.

Professional design and immersive features ensure you are left with an easy way to get people using your new dating platform. This new and improved modern design improves upon many of the features that are now standard within the dating app industry, fitting them for you.

Installing Ionic & Cordova

Ionic apps are created and developed primarily through the Ionic command line utility (the "CLI"), and use Cordova to build/deploy as a native app. This means we need to install a few utilities to get developing. Check the official installation process.

You will need to:

- Get Node and NPM
- Install Ionic CLI and Cordova

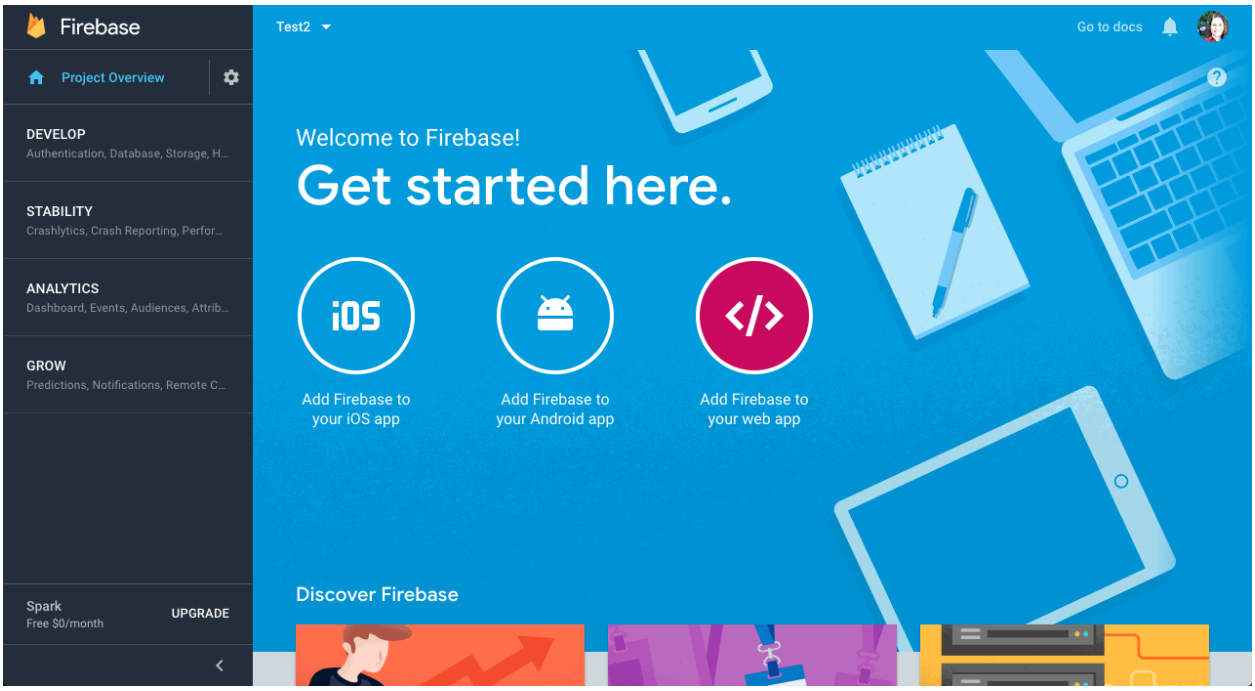
Install the template dependencies. After you have Ionic and Cordova installed, you have to install all the dependencies required by the app. To do this just run this command using a terminal inside your ionic app folder path.

$ npm install

Set up your Ionic Dating App

Creating the Firebase Application

Once AngularFire has been installed (the plugin that we'll use to communicate our Ionic app with Firebase), we need to create a new project in Firebase. To create a project, go to the firebase console, where you'll see the following menu:



Click on "**Add Firebase to your web app**" to see your new Firebase application's credentials. We'll specify these credentials to tell our Ionic application to communicate with our Firebase application.

```
<script src="https://www.gstatic.com/firebasejs/4.9.0/firebase.js"></script>
<script>
  // Initialize Firebase
  // TODO: Replace with your project's customized code snippet
  var config = {
    apiKey: "<API_KEY>",
    authDomain: "<PROJECT_ID>.firebaseapp.com",
    databaseURL: "https://<DATABASE_NAME>.firebaseio.com",
    storageBucket: "<BUCKET>.appspot.com",
    messagingSenderId: "<SENDER_ID>",
  };
  firebase.initializeApp(config);
</script>
```

The next step will be to add our Firebase credentials to our Ionic application. For this we'll go to our Ionic project, which we created in the previous step, and add the following code in the credentials.js file located in src/*app*:

Keep in mind that you have to replace the constants with your own values.

Now we're ready to start implementing the CRUD with Firebase and Ionic.

That way we can use the Facebook app to sign-in our users, instead of having them log in through a browser.

Set up Facebook Authentication

To make things easier, we're going to break down the process into three different parts:

- Step #1: We'll log into our Facebook developer account, create a new app and get the credentials.
- Step #2: We'll go into our Firebase console and enable Facebook Sign-In with the credentials from the first step.
- Step #3: We'll write the code to authorize the user through Facebook and then authenticate that user into our Firebase app.

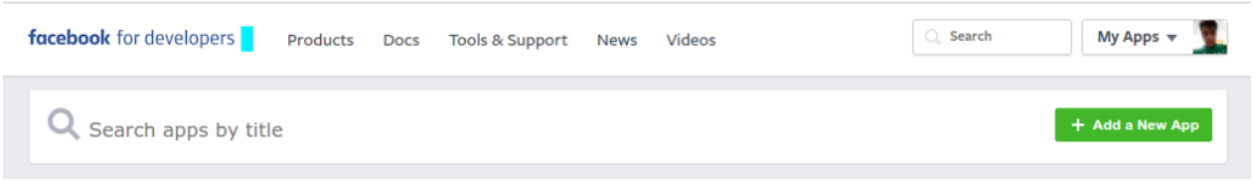
By the way, at the end of this post, I'm going to link a Starter Template that already has Google & Facebook authentication ready to go, all you'd need to do is add your credentials and run npm install.

With that in mind, let's start!

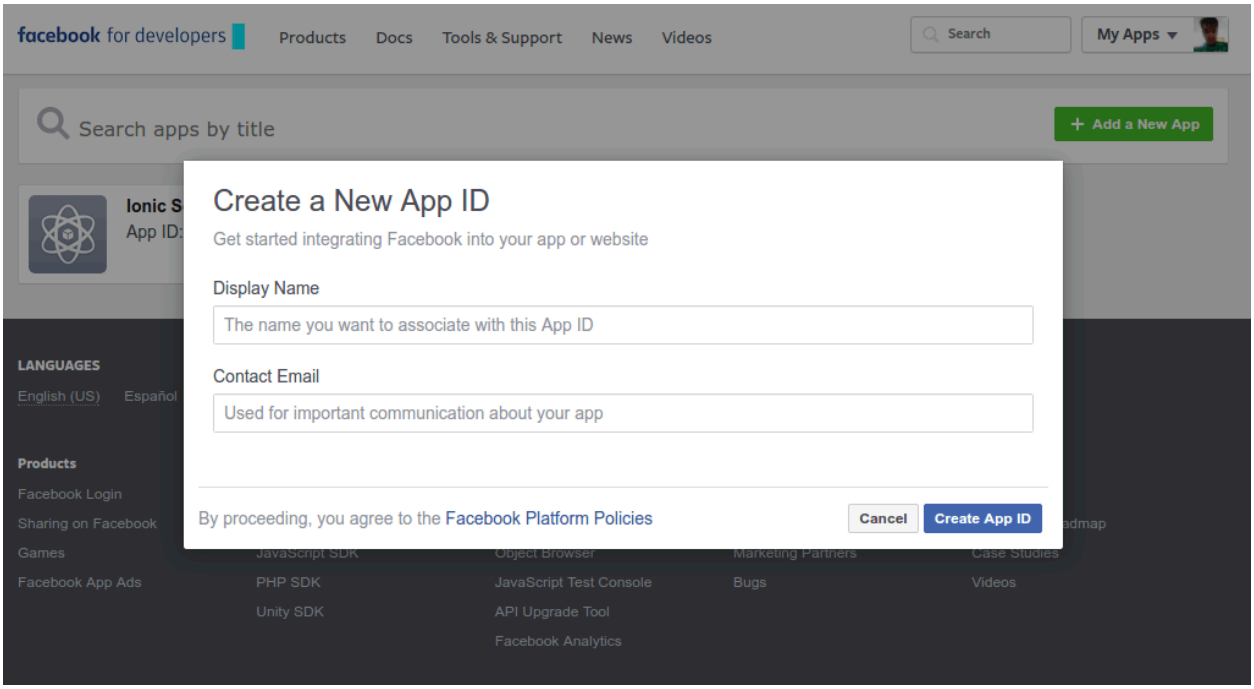
Step #1: Facebook Developer Console

The first thing we need to do is to create a new application in Facebook's developer dashboard, and this app is the one that Facebook will use to ask our users for their permission when we try to log them into our Ionic application.

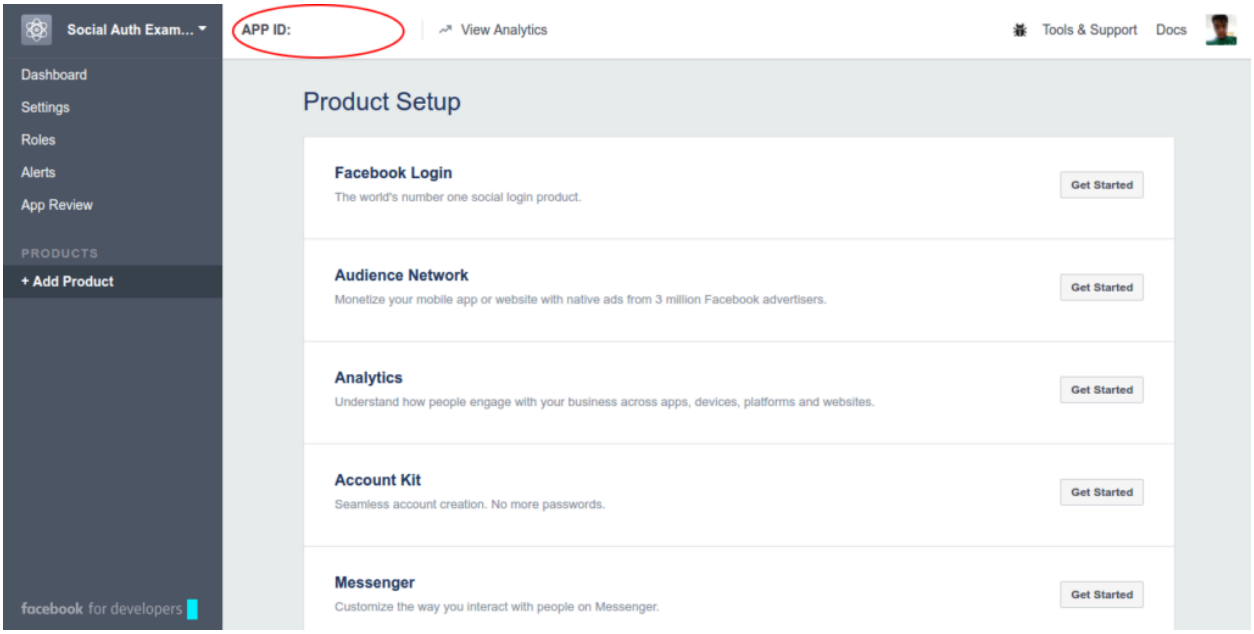
For that you'll need to go to https://developers.facebook.com/apps and create a new app.



Once you click on the button, you'll get a short form pop up, add the name you want for your app and the contact email that will be public to users.



Once we finish creating our app, it will take you to the app's dashboard, where you can see the app's ID, take note of that ID, we'll need it when it's time to install the Facebook plugin.



Install the Cordova Plugin

Now that you created your app on Facebook, we need to install the Cordova plugin in our Ionic app so we can have our app calling the Facebook Sign-In widget.

For that, open your terminal and type (All in the same line):

$ ionic plugin add cordova-plugin-facebook4 --variable APP_ID="123456789" --variable APP_NAME="myApplication"

You'll need to replace the values or APP_ID and APP_NAME for your real credentials. You can find both of those inside your Facebook Developers Dashboard.

It's a bit of a pain to work with Cordova plugins, luckily the great Ionic Team created Ionic Native, which is a wrapper for the Cordova plugins so we can use them in a more "Angular/Ionic" way.

So the next thing we need to do is install the facebook package from Ionic Native, open your terminal again and type:

$ npm install --save @ionic-native/facebook

After we finish installing it, we need to tell our app to use it, that means, we need to import it in our app.module.ts file

```
import { SplashScreen } from '@ionic-native/splash-screen';
    import { StatusBar } from '@ionic-native/status-bar'
    import { Facebook } from '@ionic-native/facebook'

    @NgModule({
      ...,
      providers: [ SplashScreen, StatusBar, Facebook ]
    })
```

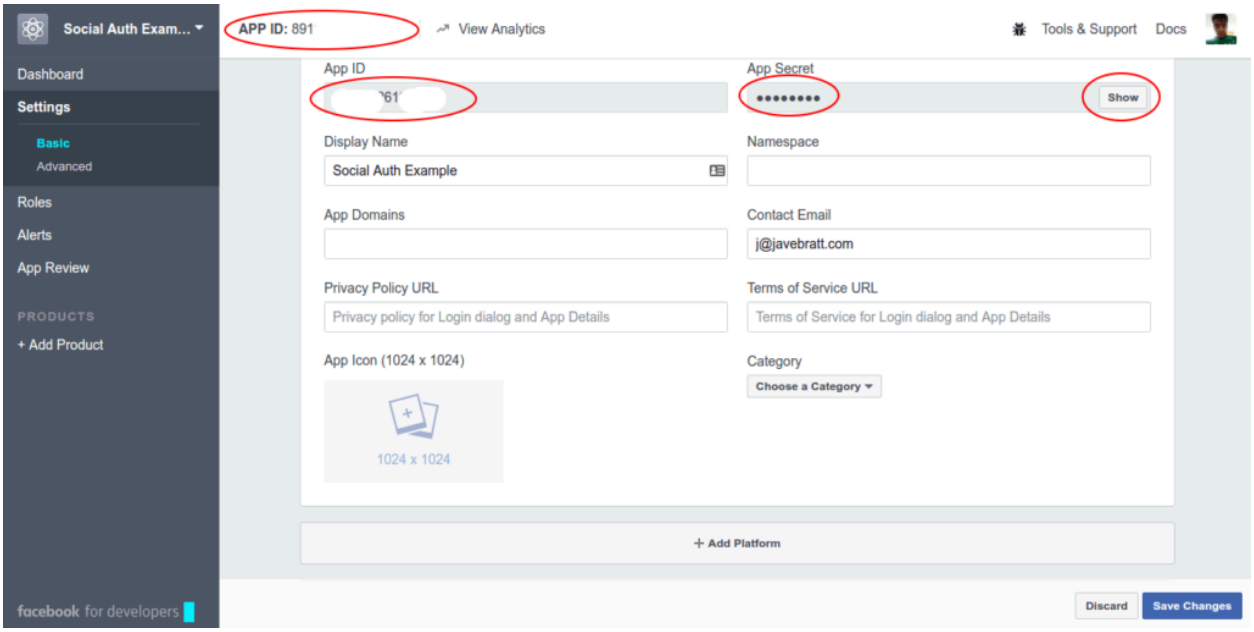
```
export class AppModule {}
```

Add your Platforms to Facebook

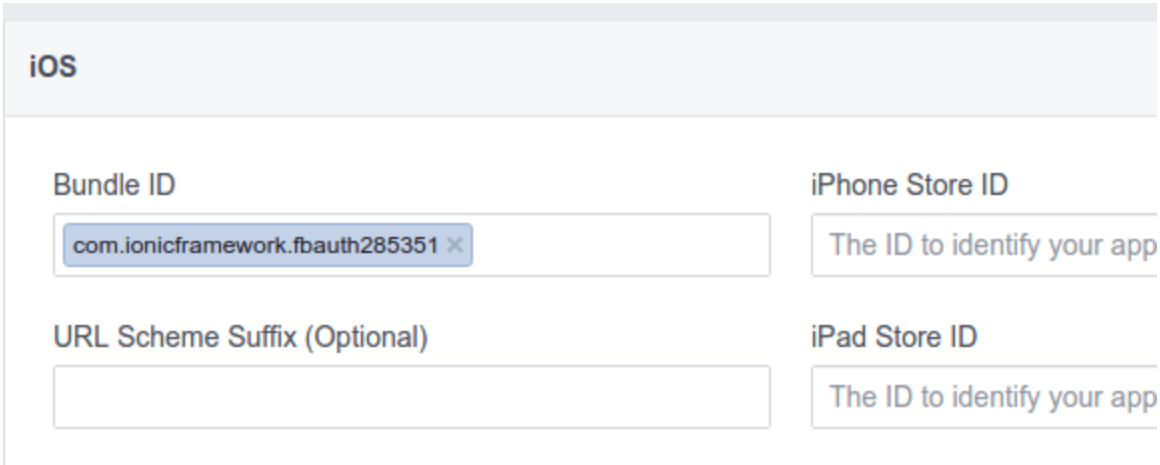
Once everything is set up in our development environment, we need to let Facebook know which platforms we'll be using (if it's just web, iOS, or Android).

In our case, we want to add two platforms, iOS and Android.

To add the platforms, go ahead and inside your Facebook dashboard click on settings, then, right below the app's information you'll see a button that says Add Platform, click it.

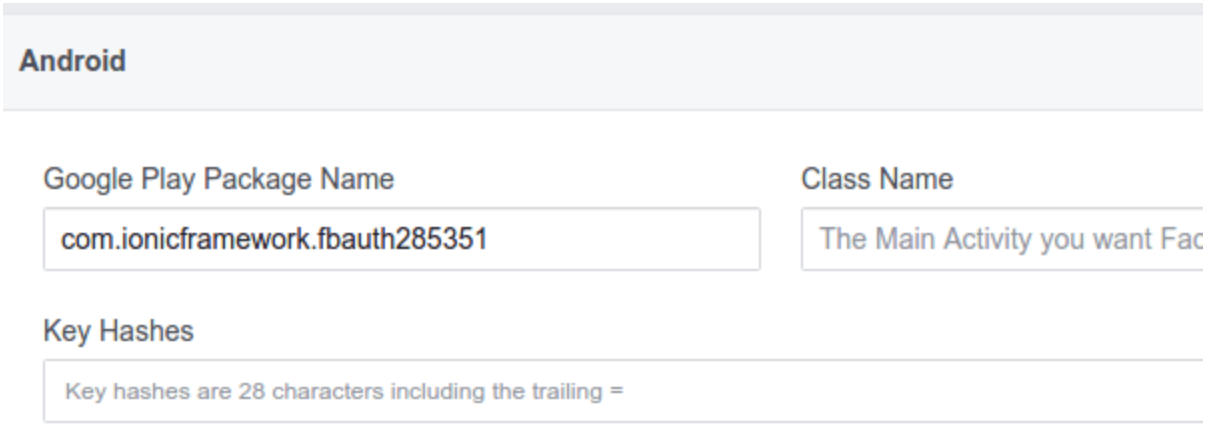


Once you click the button, you'll see several options for the platforms you're creating, let's start with iOS, you'll see a form asking you for some information, right now we just need the Bundle ID.



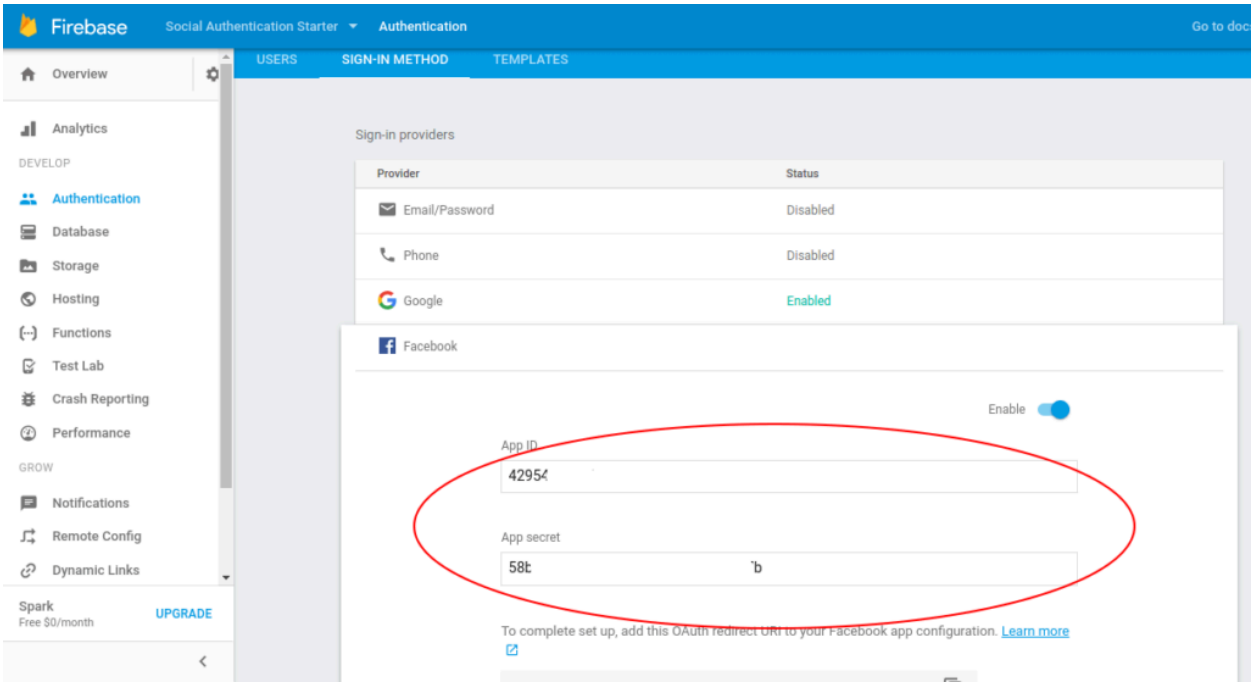
If you don't know where to get the Bundle ID, it's the same as the package name when you create an Ionic app, it's inside your config.xml file:
<widget id="co.ionic.facebook435" version="0.0.1" xmlns="http://www.w3.org/ns/widgets"
    xmlns:cdv="http://cordova.apache.org/ns/1.0">

Please, I beg you, change co.ionic.facebook435 (or what you got there) for something that's more "on brand" with your app or your business.
NOTE: Not kidding, go to Google Play and do a search for "ionicframework" you'll see a couple of hundred apps that didn't change the default package name.
Once you add the Bundle ID, just follow the process to create the app and then do the same for Android, the difference is that instead of Bundle ID, Android calls it "Google Play Package Name."



Step #2: Enable Facebook Sign-In in Firebase.

Now that everything is set up on Facebook's side, we need to go into our Firebase console and enable Facebook authentication for our app.
To enable Facebook, you'll need to go to your Firebase Console and locate the app you're using.
Once you're inside the app's dashboard, you're going to go into Authentication > Sign-In Method > Facebook and are going to click the Enable toggle.



Once you do, it's going to ask you for some information, specifically your Facebook app ID and secret key, you'll find both of those inside your Facebook console, under your app's settings.

Run ionic serve to start a local dev server and watch/compile Sass to CSS. This will start a live-reload server for your project. When changes are made to any HTML, CSS, or JavaScript files, the browser will automatically reload when the files are saved.

Check these developer tips to learn how to test your Ionic app like a Native app on iOS and Android devices or emulators.

Setting up Push Notifications

Step 1: Install Android and/or iOS Tools and SDKs

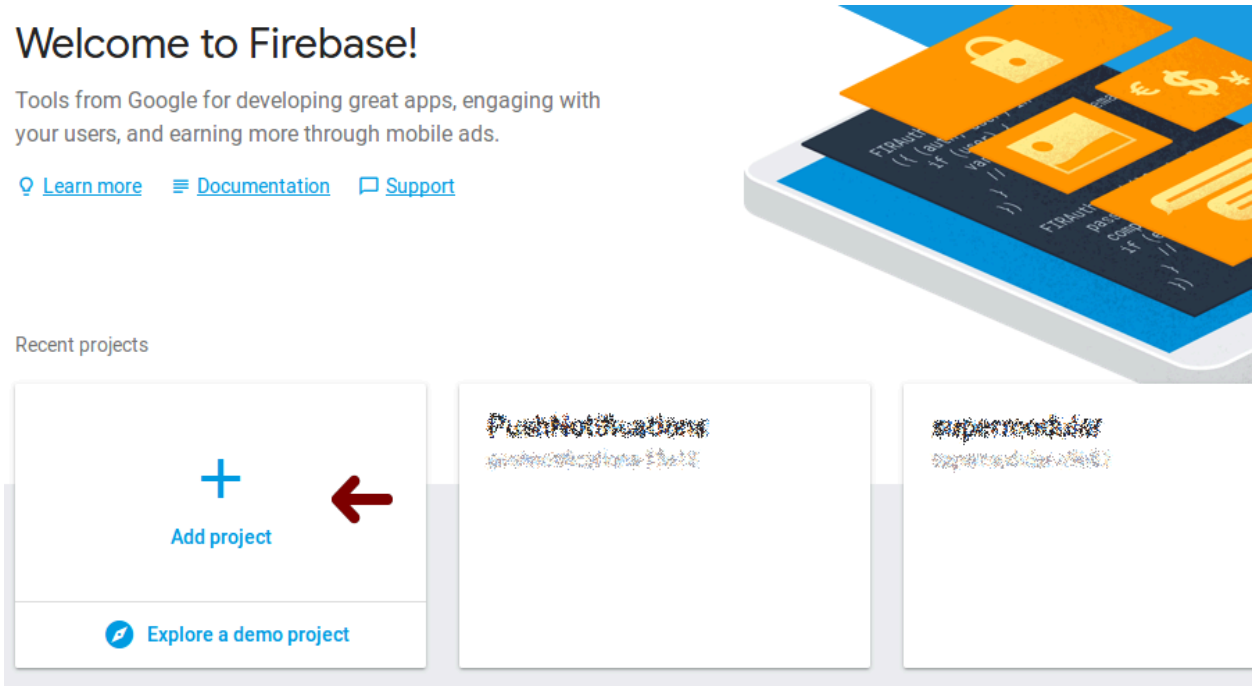
In order to test this application, you should install Android and iOS tools and SDKs on your machine. Since Ionic is based on Cordova, Cordova provides a very useful guide for installing the requirements for Android platform and iOS platform (Mac OS users only).

Step 2: Add platforms

In your terminal, from your app folder, run:
$ ionic cordova platform add android
or
$ ionic cordova platform add ios

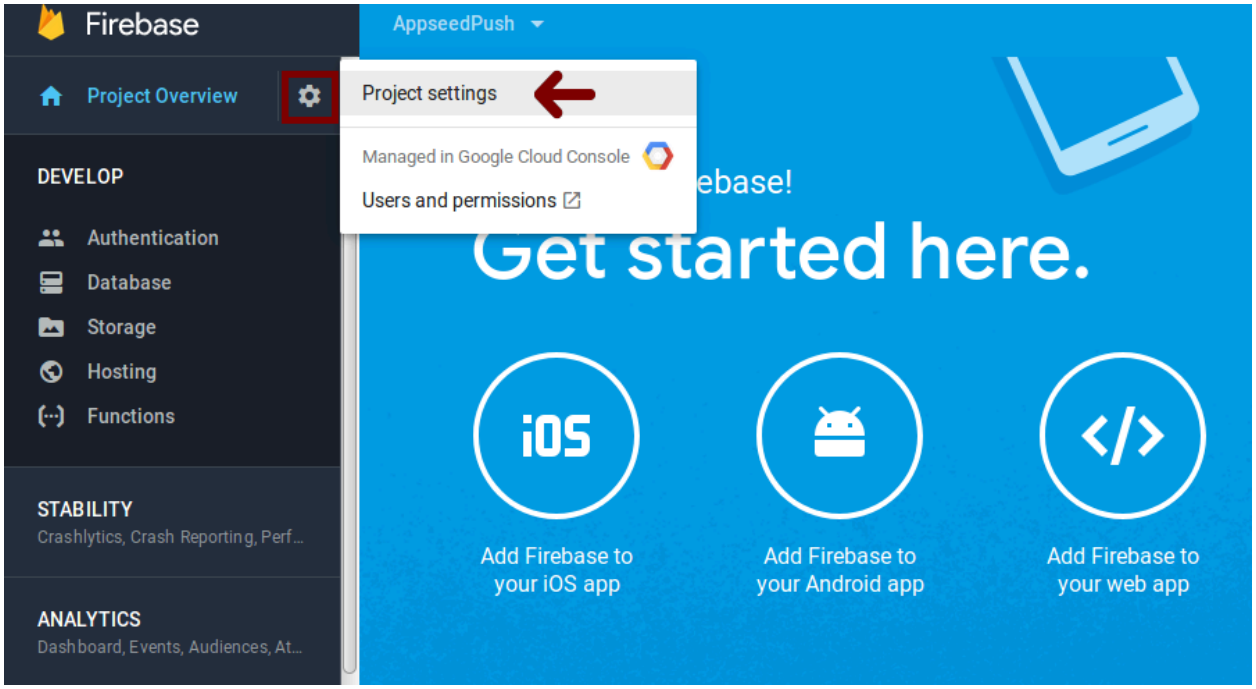
Step 3: Generate a Google Server API Key

The application we are going to build, uses Firebase in order to generate a Google Server API Key. This key allows OneSignal to use Google's web push services for your notifications. Visit https://console.firebase.google.com/ and create a new project.



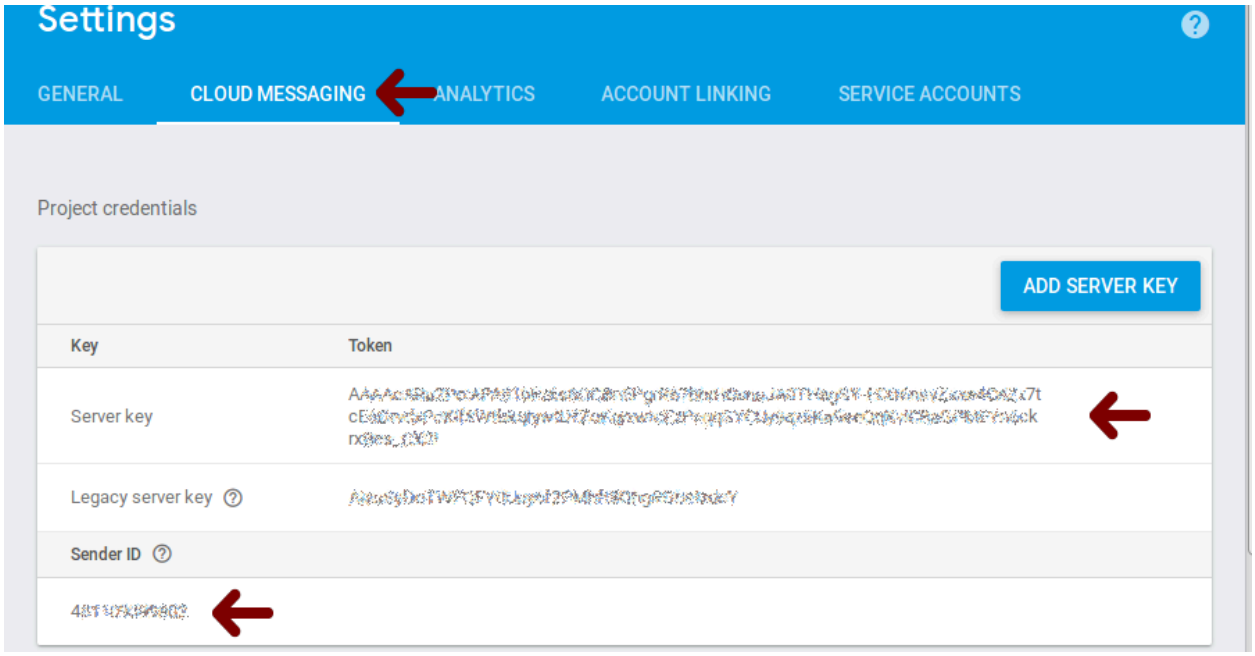
Get your Firebase Cloud Messaging token and Sender ID

Click the Gear icon at the top left and select Project settings.



Select the CLOUD MESSAGING tab.
Make a note of the two values listed:

- You'll need your Server key, also known as the Google Server API key.
- You'll need your Sender ID, also known as the Google Project Number, later as well.

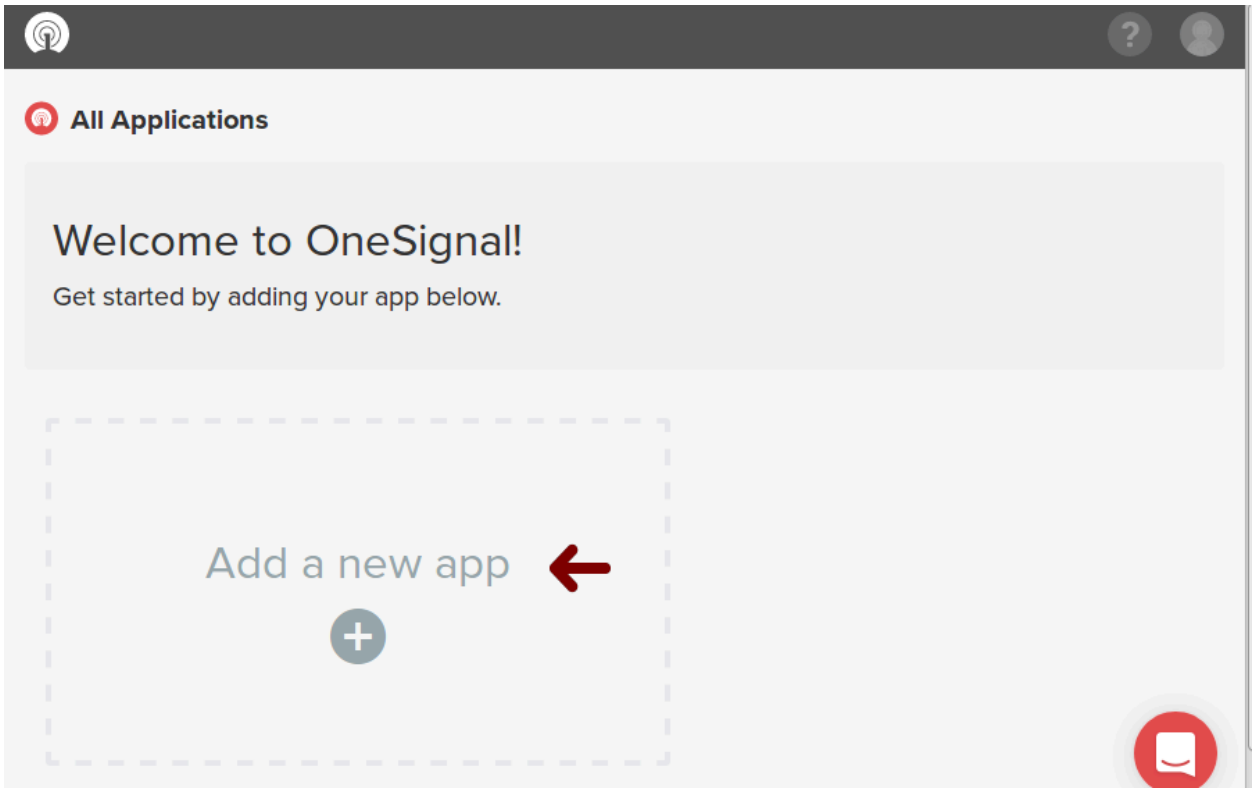


Step 4: Configure OneSignal application

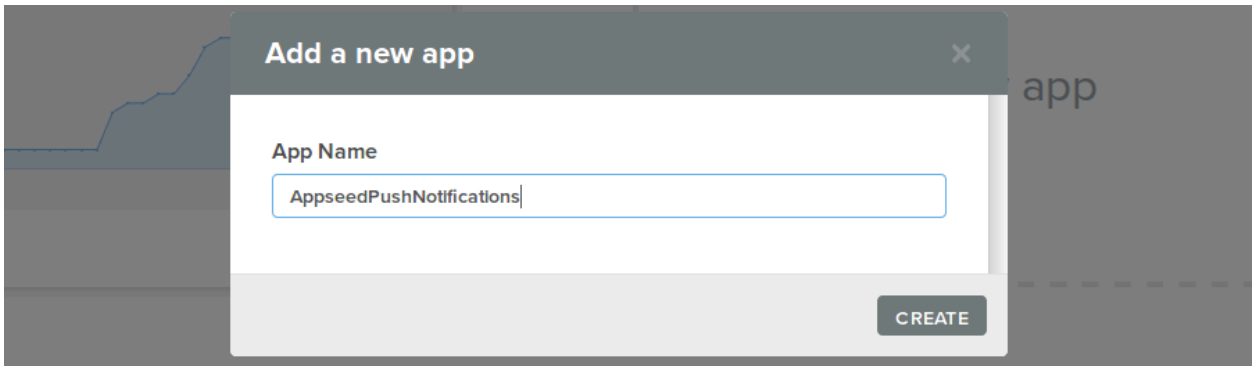
We will use OneSignal to send push notifications. Visit OneSignal and create an account if you haven't already.

Configure OneSignal Android platform settings.

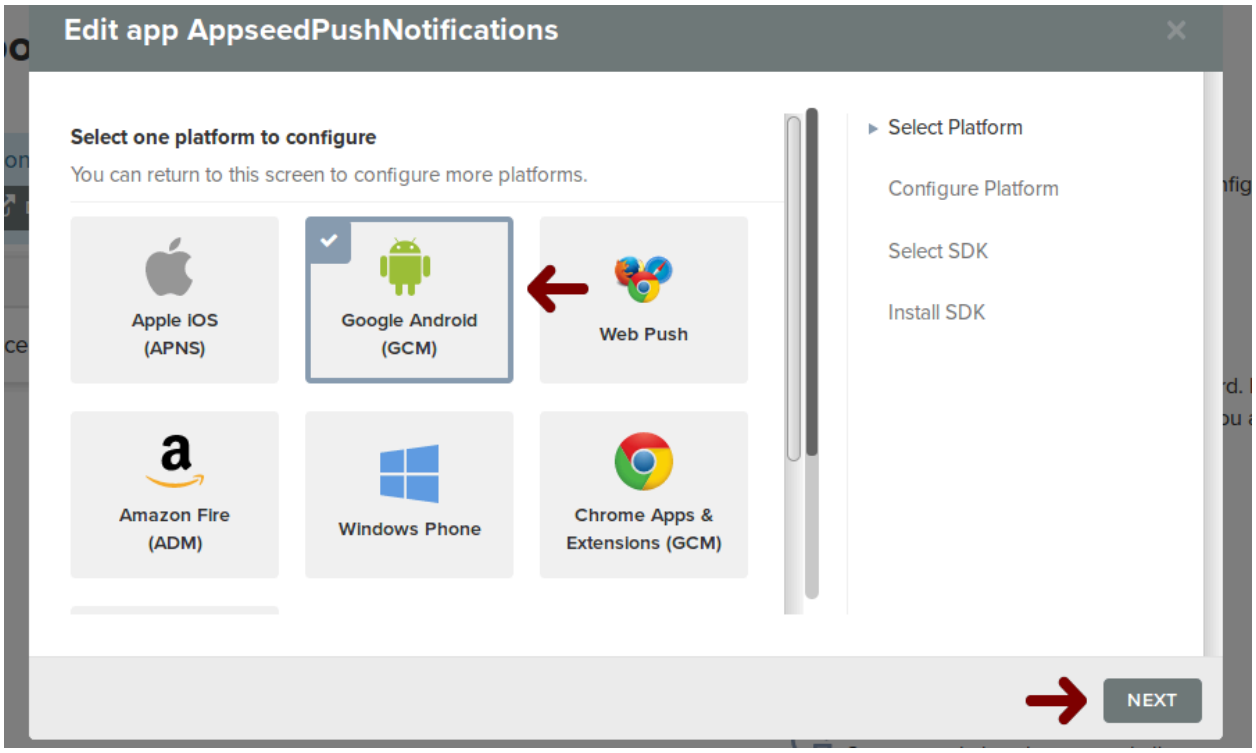
After login to One Signal, click on Add new app.



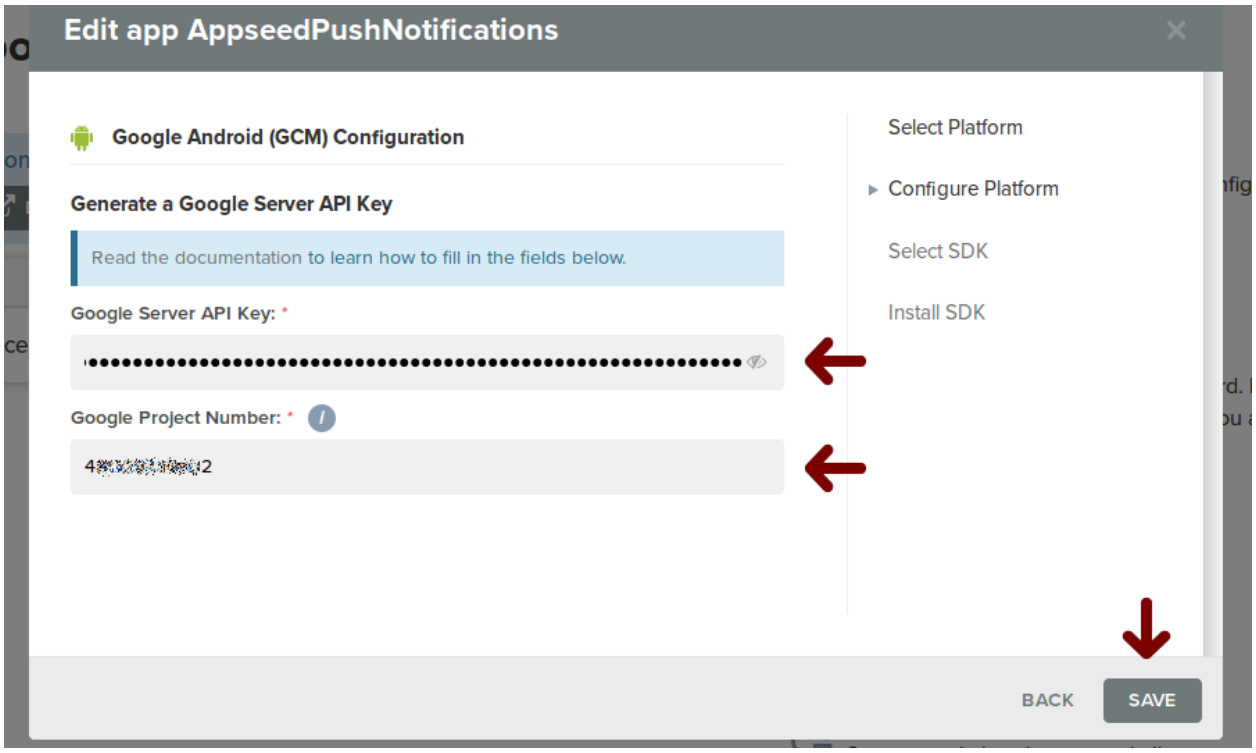
Type a name for your application in the appeared pop up. Here, we use “AppseedPushNotifications” as an App Name.



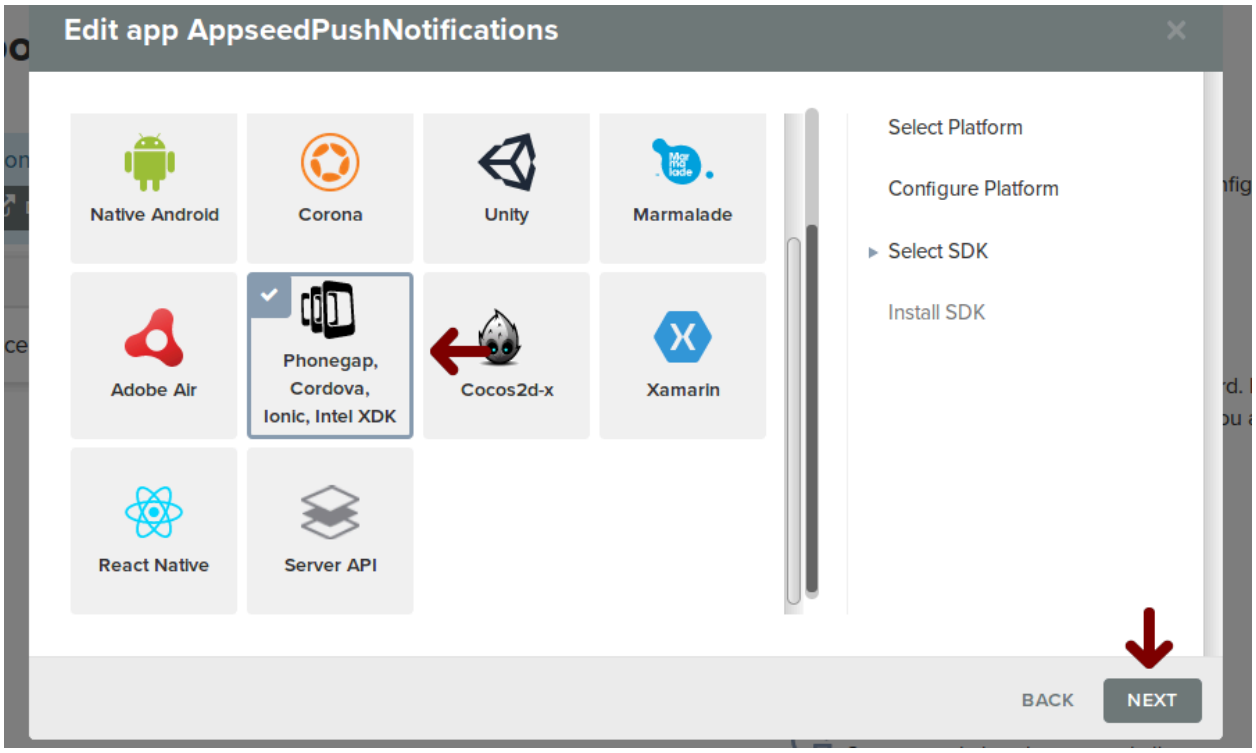
Select the platform that you want to configure with OneSignal push notifications and click NEXT. For this tutorial, select Google Android (GCM).



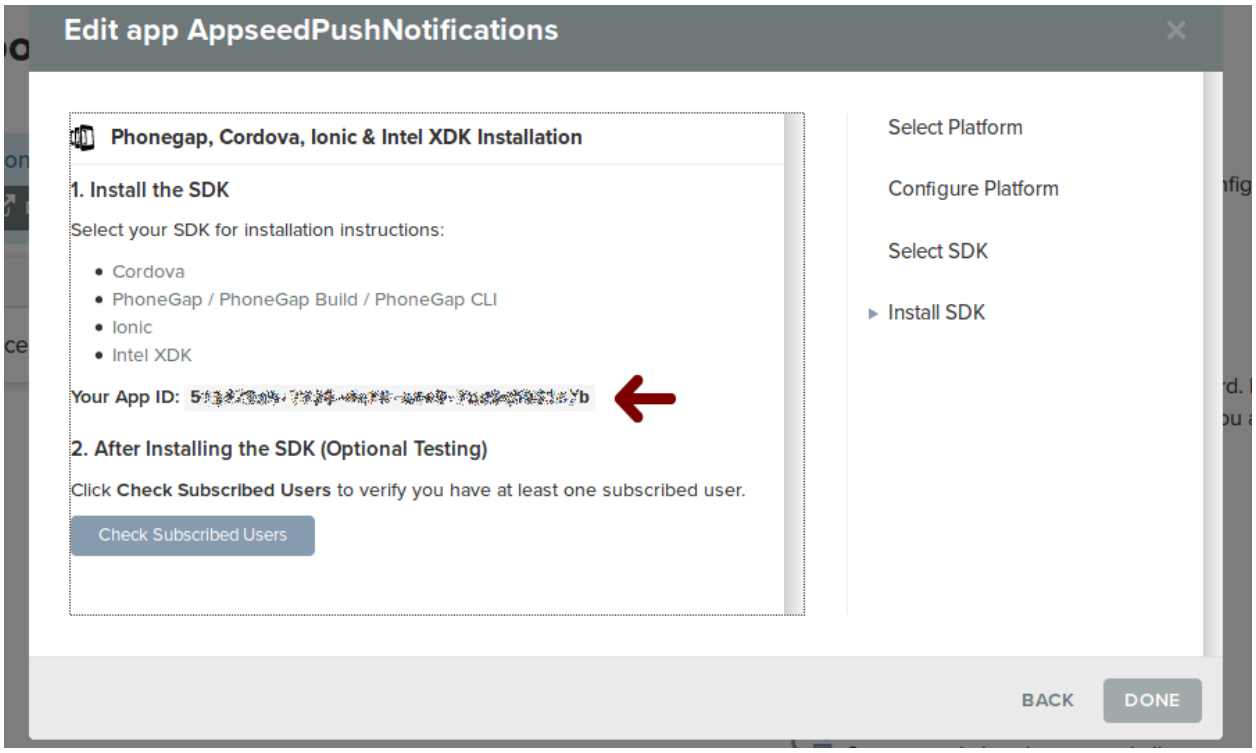
Next, enter your Google Server API Key and Google Project Number and click SAVE. These are the details you got from the Firebase console in the previous step.



Then, select the target SDK for your app. In our case, we are making an Ionic app so select, PhoneGap, Cordova, Ionic, Intel XDK and click NEXT.



As a final step, take a note of the OneSignal App ID, as we will use it in our application later.



Step 5. Implement OneSignal push notifications

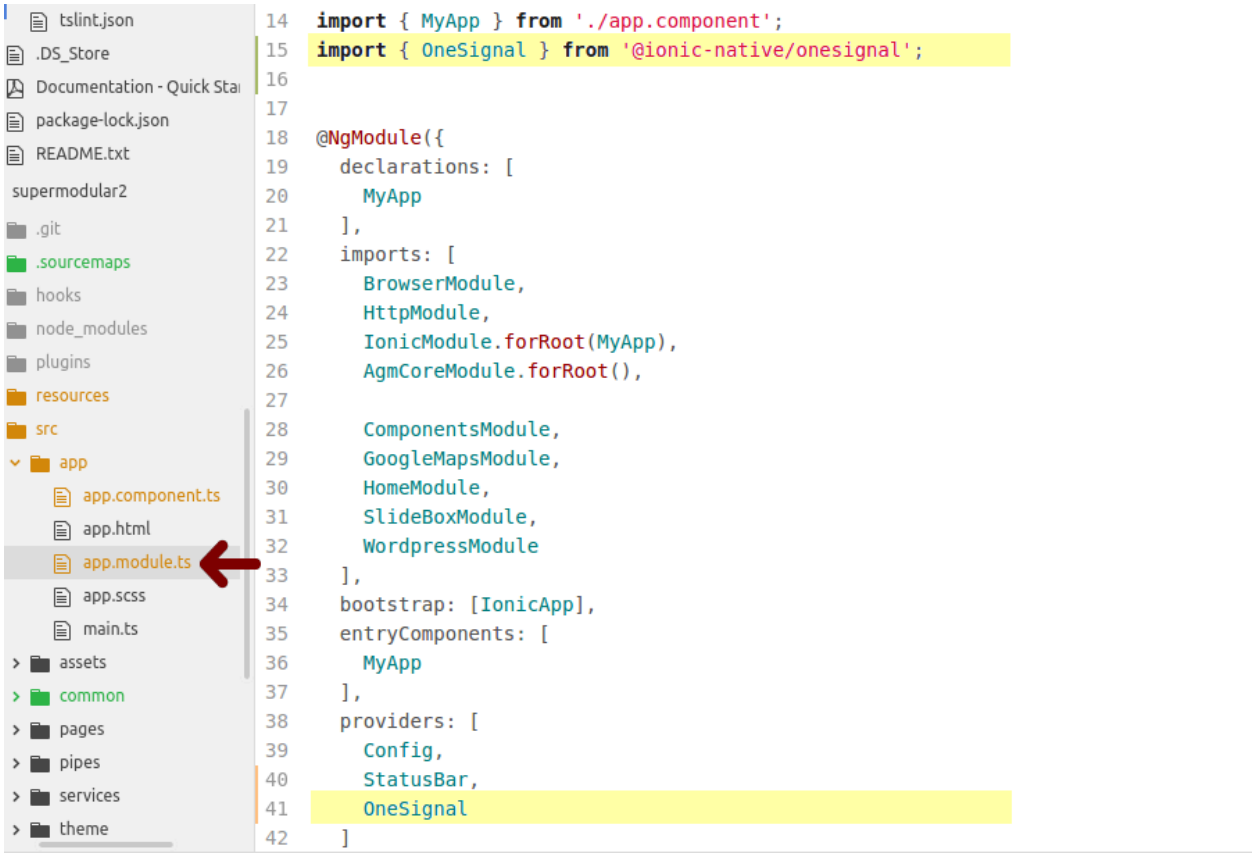
Install Cordova plugin

For this application to work, we need to install OneSignal Cordova plugin.From your terminal, navigate in the app folder and execute the following command:

```
$ ionic cordova plugin add onesignal-cordova-plugin
```

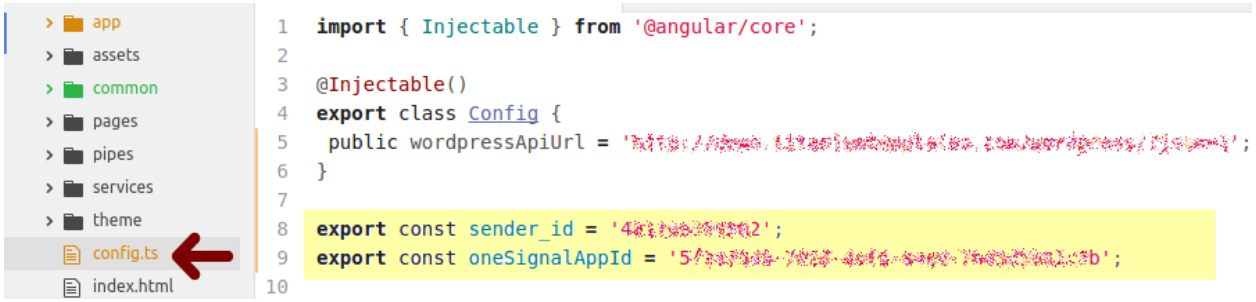
Import Cordova plugin

After installing the plugin, import OneSignal and add it to your app's NgModule in app.module.ts under src/app as shown below.



Add required keys

Next, locate the config.ts file under the src folder. Here, we need to copy the App ID provided by OneSignal earlier and paste it to the oneSignalAppIdvariable. Also copy the Google Project Number provided by Firebase earlier and paste it to the sender_id variable in the config.ts file.



Check Cordova availability

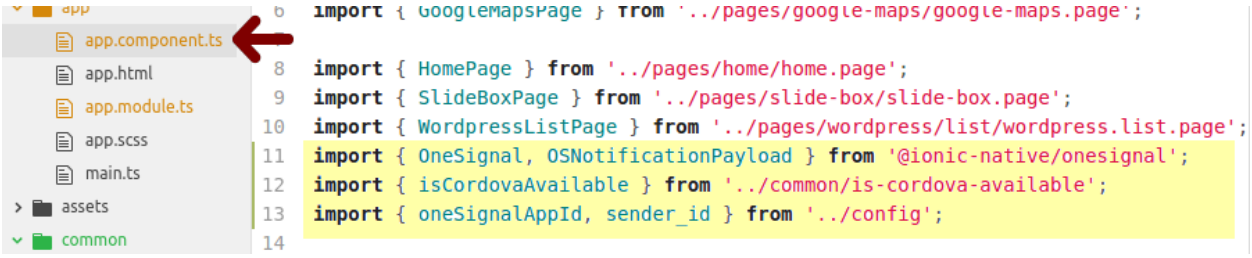
The application uses the OneSignal cordova plugin which we referred to earlier. So, the app cannot run in a web browser. That's why we have to check if the application runs in a real device or an emulator where Cordova is available.

The code below, checks if Cordova is available and returns true or false accordingly. In case Cordova is not available, a message it is also shown prompting the user to use a mobile device.

Under src/ create a folder called common and a file called is-cordova-available.ts and paste the following code.

Initialize One Signal plugin at application startup

In this section, we are going to register the device for receiving push notifications once the app loads. In app.component.ts under src/app add the highlighted section.



Also, add the highlighted line in the constructor in the same file.



Add the following code in app.component.ts file under src/app.



In short, in line 48, we check if the application is executed on a device. The startInit() method, starts the initialization process.

The inFocusDisplaying() method, controls how OneSignal notifications will be shown when one is received while your app is in focus. In this case, it uses Notification as a parameter, which is a native notification displayed while the app is in focus.

The handleNotificationReceived() is a method that runs when a notification is received. It calls the onPushReceived() method which is explained later.

The handleNotificationOpened() is a method that runs when a notification is tapped or when closing an alert notification shown in the app. It calls the onPushOpened() method which is explained later.

The endInit() method must be called to complete the initialization of OneSignal.
Finally, add the following code to handle the received and opened notifications.



Here, we use two methods. onPushReceived() which alerts the user with the notification message once the message is received and onPushOpened()which alerts the user with the notification message once the notification is opened.

OSNotificationPayload contains a payload with the content and settings of the notification the user has received, in this case, only the message of the notification.

Run / Test

Since OneSignal is a Cordova plugin we need to test the application on an emulator or on a real mobile device.

In your terminal, from your project folder type:

```
$ ionic cordova platform add android
```

Customizing your App

New pages

In Ionic 3 there's some more boilerplate compared to ionic 1, but don't panic. The new ionic CLI also has more tools to help you out with this.

Take for example the new generator functions, they provide an easy way to create pages and services for your app. This makes going from a basic app to a full featured app with navigation much easier.

To create a page you can use the following command:

```
# ionic g page <PageName>
ionic g page myPage

√ Create app/pages/my-page/my-page.html
√ Create app/pages/my-page/my-page.ts
√ Create app/pages/my-page/my-page.scss
```

Note: Please refer to ionic documentation for more information about adding pages to your app.

Custom styles

We have talked about this above. Believe me, Sass will be your best friend when it comes to styling, it gives superpowers to your css.

The way we structured and build the styling system for this app template will enable you to quickly modify, add and create new styles to match the look and feel and the use cases

of your app. There's much to learn about Sass to get the most out of it. We believe learning by example is the best way, and you will enjoy that benefit with our app template. Also, you will find useful documentation about Sass here and here.

New layouts

If you want to implement new layouts I would highly recommend you to the first search within the ionic vast list of components to check if there's one that fit your needs.

If you find none, then try to think the way you do when developing for the web, after all, one of the beautiful things and advantages about ionic is that you build stuff using web technologies (HTML, css, js), and such the web is an open platform you will find tons of solutions for already solved issues.

In particular, when I don't find the component I need, the approach I follow is first thinking the structure and elements I may need to represent and build the shell of what I want. I constantly rely on the ionic grid system. And since I discovered it, I've become an expert on Flexbox, and let me tell you, that makes the difference!

Splash screen and app icon

An app icon and splash screen (launch image) are important parts of any app, yet making them used to be incredibly tedious. You needed numerous icons for iOS and Android, and then you had to deal with splash screens and all their different sizes.

To save you the stress of dealing with all that, ionic enables you to generate app icons and splash screens via the Ionic CLI.

With the ionic CLI, all you need is a resource directory and two images. These images can be .png files, Photoshop .psd files, or Illustrator .ai files, named icon.png and splash.png.

With the images in a resources directory, ./resources, the ionic resources command will generate the icons and splash screen images for each platform set up in the project, sending them to Ionic's image resizing and cropping server, so you don't have to install extra dependencies.

If you only need to update one of the resources, or you only want to generate icons and not both, the ionic resources command has two flags that allow you to target each asset, instead of generating both.

ionic resources --icon
ionic resources --splash

If you need any further assistance with this, please go to ionic cordova resources for more information.

Data Integration and Services

In this section, we will not discuss any backend implementation in particular because as we explained above, there are so many options and flavors when it comes to backend implementations that it turns impossible to provide examples for every option.

We will focus instead on the app's side of the problem, how to handle data calls, as this works the same and is independent on the way you implement the backend. We will talk about models and services and how they work together to achieve this.

Models

Domain models are important for defining and enforcing business logic in applications and are especially relevant as apps become larger and more people work on them.

At the same time, it is important that we keep our applications DRY and maintainable by moving logic out of components themselves and into separate classes (models) that can be called upon. A modular approach such as this makes our app's business logic reusable.

To learn more about this, please visit this great post about angular 2 domain models.

Services

As you may know, ionic 3 is implemented on top of angular 5 and it borrows it's best parts. Angular 5 enables you to create multiple reusable data services and inject them in the components that need them.

Refactoring data access to a separate service keeps the component lean and focused on supporting the view. It also makes it easier to unit test the component with a mock service.

To learn more about this, please visit Angular Tutorial: Learn Angular from scratch step by step.

We encourage the use of models in combination with services for handling data all the way from the backend to the presentation flow.

Object-oriented everything!

As we mentioned when we talk about models, they are key to defining and enforcing business logic in applications. Angular 5 embraces this and that's why you will see object-oriented stuff all along the way. Depending on the style guide you followed for developing angular 1 applications, this may sound akin to you or not as angular 1 didn't embrace OOP the way angular 2 does.

Other key ingredients to my previous argument are Typescript and ES6. Angular 5 is written in TypeScript, and TypeScript is a superset of JavaScript, in particular, ES6.

ES6

One of the new features of ES6 are Classes. They are a simple sugar over the prototype-based OO pattern. Having a single convenient declarative form makes class patterns easier to use, and encourages interoperability. You can learn more in this article about ES6 features.

Typescript

TypeScript makes abstractions explicit, and a good design is all about well-defined interfaces. It is much easier to express the idea of an interface in a language that supports them. I won't go in detail here because much it's said in this post about typescript and angular.

But as a conclusion let me tell you that TypeScript takes 95% of the usefulness of a good statically-typed language and brings it to the JavaScript ecosystem. You still feel like you

write ES6: you keep using the same standard library, same third-party libraries, same idioms, and many of the same tools (e.g., Chrome dev tools). It gives you a lot without forcing you out of the JavaScript ecosystem.

Ionic 3 Navigation

This may be the key point that differs more an angular 5 app. Ionic 3 doesn't use angular 5 routing system, instead they created their own navigation to better serve the use cases of mobile apps. They inspired themselves in the way ios handles navigation between pages, pulling pushing views instead of a tree structure most commonly seen on the web.

At first, I was reticent about this approach as I was used to the way ionic 1 worked with tree structures for the routing. After some time playing around with ionic 3 I must admit that this new navigation is much more easy and understandable than the one in ionic 1.

In ionic 1 you were reaching the limits of the framework everytime you wanted to implement sophisticated navigation. I have created complex navigation structures in this app template and I don't get that feeling, in the contrary, complex navigations can be easily implemented with ionic 3 navigation system.

As a final note, in ionic 3, don't think about your pages as specific route paths, but rather views you can display anywhere at any time in your app, for example, a ContactDetail page could be displayed through any number of user navigation flows. It could even link to itself infinitely.

In Ionic 1 you had to say "this page lives at this route" and new flows required new routes. Ionic 3 liberates you from strict paths -> pages.

Ahead of Time

Using this angular 5 technique you can radically improve performance by compiling Ahead of Time (AoT) during a build process.

An ionic application consists largely of components and their HTML templates. Before the browser can render the application, the components and templates must be converted to executable JavaScript by the Angular compiler.

When developing the app you can compile the app in the browser, at runtime, as the application loads, using the Just-in-Time (JiT) compiler. This is the standard development approach, it's great .. but it has shortcomings.

JiT compilation incurs a runtime performance penalty. Views take longer to render because of the in-browser compilation step. The application is bigger because it includes the Angular compiler and a lot of library code that the application won't actually need.

The Ahead-of-Time (AOT) compiler can catch template errors early and improve performance by compiling at build time. If you want to learn more about AoT I would suggest you read these posts here and here.