# UNIT 4

## Ionic

**Part 3**
**Ionic Native. Publish app.**
**3rd party libraries.**

Client-side Web Development
2nd course – DAW
IES San Vicente 2019/2020
Author: Arturo Bernal Mayordomo

## Index

# Capacitor

Capacitor is a new native API created by the Ionic team. It provides the necessary APIs to access native functionality like the camera, clipboard, device, filesystem, networking, push notifications, storage, status bar, motion, haptics, etc.

This library is made to replace Cordova / PhoneGap. It's compatible with Android, iOS, Electron and PWAs. iOs development requires Xcode 10 or above installed (and a Mac), while Android development requires Android Studio.

When you create a new project, it will ask you if you want to enable capacitor, but you can do it afterwards. To enable Capacitor on an Ionic app, run this command:

**ionic integrations enable capacitor**

You can also install Capacitor on a regular Angular app using ng add:

**ng add @capacitor/angular**

You can even use Capacitor anywhere (like plain JavaScript app):

**npm install --save @capacitor/core @capacitor/cli**

After enabling Capacitor, we need to initialize it with the name of our app and a domain (Java style). Example:

**npx cap init ionic-products com.example.products**

Before adding platforms (Android, iOS), we must at least build our app once.

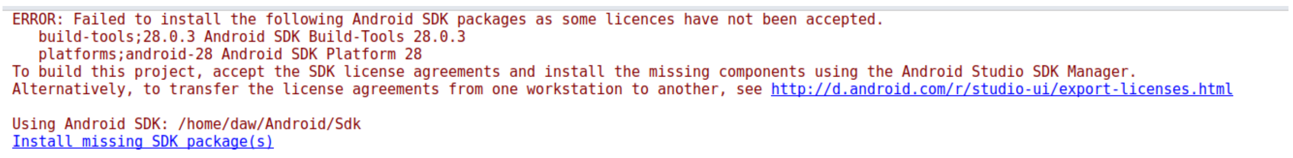
**ionic build**

**npx cap add ios** (add iOS support)

**npx cap add android** (add Android support)

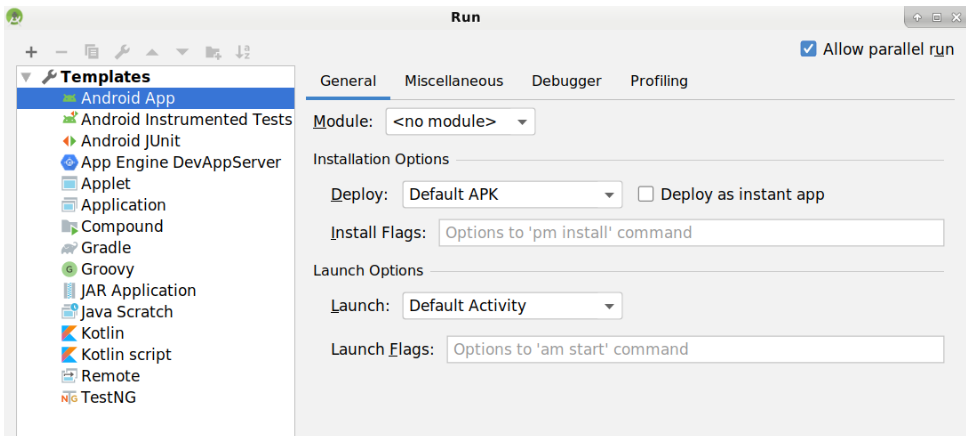
We can open Android studio with this command (npx cap open android) or just open the **android** folder inside the project from Android Studio.

If you get an error in the IDE like this, click the link "Install missing SDK packages" before running the app.



```
ERROR: Failed to install the following Android SDK packages as some licences have not been accepted.
   build-tools;28.0.3 Android SDK Build-Tools 28.0.3
   platforms;android-28 Android SDK Platform 28
To build this project, accept the SDK license agreements and install the missing components using the Android Studio SDK Manager.
Alternatively, to transfer the license agreements from one workstation to another, see http://d.android.com/r/studio-ui/export-licenses.html

Using Android SDK: /home/daw/Android/Sdk
Install missing SDK package(s)
```

After opening the project in Android Studio, you can run it as an Android App (alt+F10).


Run

Allow parallel run [✓]

Module: <no module>

Installation Options

Deploy: Default APK    Deploy as instant app [ ]

Install Flags: Options to 'pm install' command

Launch Options

Launch: Default Activity

Launch Flags: Options to 'am start' command

Every time you update your code (or install any Capacitor plugin) you'll need to run **ionic build** and **npx cap sync**.

## Configuring the Android project

To modify the bundle/app id for your app, edit the top `<manifest>` line in **manifests/AndroidManifest.xml**:

```
<manifest package="com.arturober.capacitor">
```

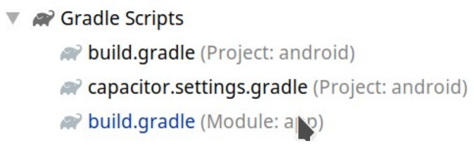
Now, edit **res/values/strings.xml** file and set the name for the app and main Activity, and also package name and custom url (usually the same as the app's id).

```
<?xml version='1.0' encoding='utf-8'?>
<resources>
    <string name="app_name">Ionic capacitor</string>
    <string name="title_activity_main">Ionic capacitor</string>
    <string name="package_name">com.arturober.capacitor</string>
    <string name="custom_url_scheme">com.arturober.capacitor</string>
</resources>
```

By default, the entire initial permissions requested for the latest version of Capacitor with the standard plugins can be found in the android-template's AndroidManifest.xml.

Also, change the **build.gradle** file with the App id:

```
android {
    compileSdkVersion 28
    defaultConfig {
        applicationId "com.arturober.capacitor"
        ...
    }
    ...
}
```


Gradle Scripts
build.gradle (Project: android)
capacitor.settings.gradle (Project: android)
build.gradle (Module: app)

## Enabling requests to HTTP

If your server is not using HTTPS, you must enable **cleartext** traffic. GO to your **AndroidManifest.xml** file and add this to the application tag:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest ...>

    <application
        ...
        android:networkSecurityConfig="@xml/network_security_config">

        ...
    </application>
    ...
</manifest>
```

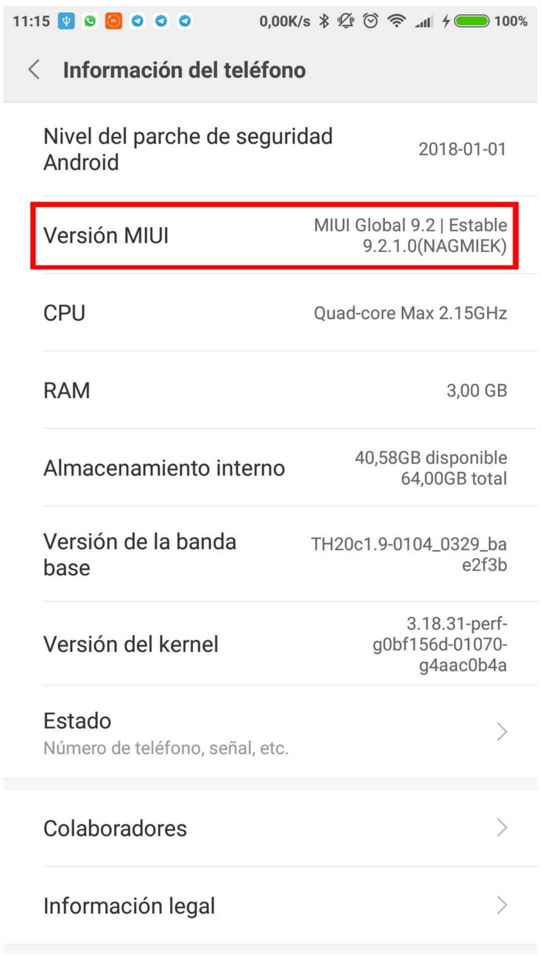
Create a file called **network_security_config.xml** inside **res/xml** directory with this content:

```
<?xml version="1.0" encoding="utf-8"?>
<network-security-config>
    <!--Set application-wide security config using base-config tag.-->
    <base-config cleartextTrafficPermitted="true"/>
</network-security-config>
```

## Configuring your Android device

To install (and run) applications from your computer in your connected Android device, you'll need to activate the "**Developer Options**". Usually this is done by tapping several times on your Android version.

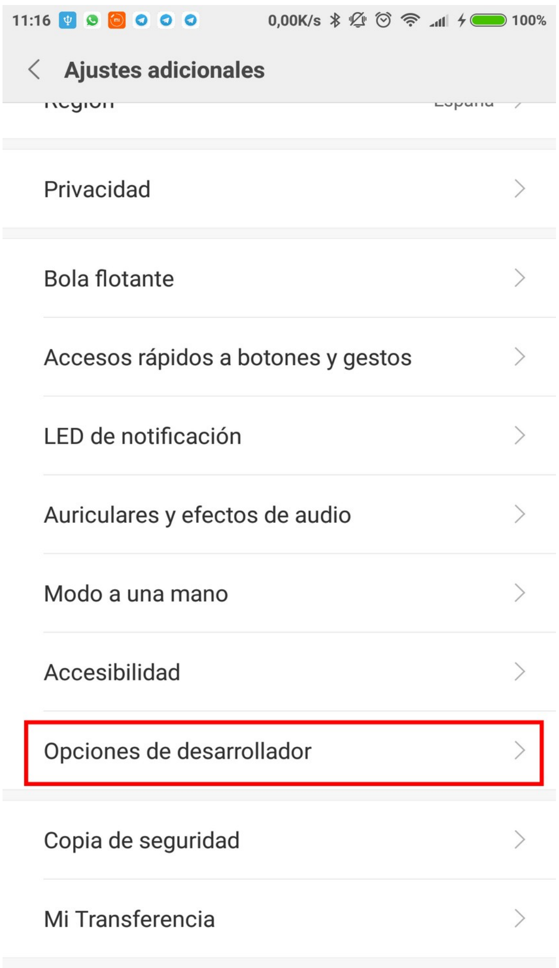
After enabling Developer options, you can access them by going back to the main configuration options and searching from there.



11:15 0,00K/s 100%

Información del teléfono

| | |
|---|---|
| Nivel del parche de seguridad Android | 2018-01-01 |
| Versión MIUI | MIUI Global 9.2 \| Estable 9.2.1.0(NAGMIEK) |
| CPU | Quad-core Max 2.15GHz |
| RAM | 3,00 GB |
| Almacenamiento interno | 40,58GB disponible 64,00GB total |
| Versión de la banda base | TH20c1.9-0104_0329_bae2f3b |
| Versión del kernel | 3.18.31-perf-g0bf156d-01070-g4aac0b4a |
| Estado Número de teléfono, señal, etc. | |
| Colaboradores | |
| Información legal | |



11:16 0,00K/s 100%

Ajustes adicionales

Región España

Privacidad

Bola flotante

Accesos rápidos a botones y gestos

LED de notificación

Auriculares y efectos de audio

Modo a una mano

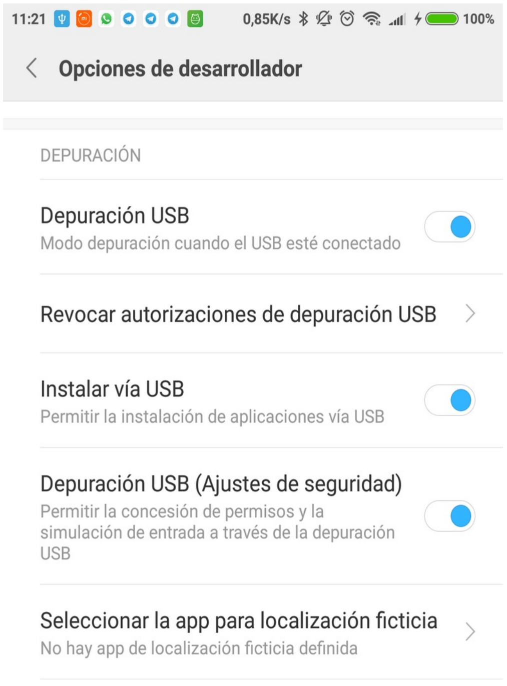
Accesibilidad

Opciones de desarrollador

Copia de seguridad

Mi Transferencia

Inside the Developer Options section, you'll need to enable the **USB debugging** option and the possibility to install applications from USB.


11:21 0,85K/s 100%

Opciones de desarrollador

DEPURACIÓN

Depuración USB
Modo depuración cuando el USB esté conectado

Revocar autorizaciones de depuración USB

Instalar vía USB
Permitir la instalación de aplicaciones vía USB

Depuración USB (Ajustes de seguridad)
Permitir la concesión de permisos y la simulación de entrada a través de la depuración USB

Seleccionar la app para localización ficticia
No hay app de localización ficticia definida

# Plugins

Capacitor supports most Cordova plugins, but there are many plugins already written for Capacitor (which is much more recent). In fact, Capacitor already has built-in support for some plugins so you won't need to install anything to access the Camera, sensors, etc.

## Splash Screen

When your app is starting, it shows an splash screen image, this is handled by the Splash Screen plugin. By default, the Splash Screen is set to automatically hide after a certain amount of time (3 seconds). However, your app should boot much faster than this.

You can follow these guides to know how to change the app icon and splash screen in Android and iOS:

https://www.joshmorony.com/adding-icons-splash-screens-launch-images-to-capacitor-projects/

https://www.joshmorony.com/creating-a-dynamic-universal-splash-screen-for-capacitor-android/

## Status bar

The Status Bar Plugin allows you to show / hide the top status bar and set the style of it. Just when the app is loaded, is a good place to set the style for the bar.

```
...
import { Plugins, StatusBarStyle } from '@capacitor/core';
const { SplashScreen, StatusBar } = Plugins;

...
  constructor(private platform: Platform) {
    this.initializeApp();
  }

  initializeApp() {
    this.platform.ready().then(() => {
      SplashScreen.hide();
      StatusBar.setBackgroundColor({color: '#3880ff'});
      StatusBar.setStyle({style: StatusBarStyle.Dark});
    });
  }
}
```

## Haptics / Vibration

The Haptics Plugin provides physical feedback to the user through touch or vibration.

In order to use vibration, we need to ask for the permission in the

**AndroidManifest.xml** file:

```
<uses-permission android:name="android.permission.VIBRATE" />

...
import { Plugins } from '@capacitor/core';
const { Haptics } = Plugins;

...
export class VibrationPage implements OnInit {
  ...
  vibrate() {
    Haptics.vibrate();
  }
}
```

## Camera

With the Camera Plugin you can get a photo (or image) from the device's camera or gallery (depending on the chosen option).

They should be already present, but check if the necessary permissions for saving photos or getting photos from the gallery are set:

```
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />

import { Component, OnInit, NgZone } from '@angular/core';
import { Plugins, CameraResultType, CameraSource } from '@capacitor/core';
const { Camera } = Plugins;

@Component({
  selector: 'app-camera',
  templateUrl: './camera.page.html',
  styleUrls: ['./camera.page.scss'],
})
export class CameraPage implements OnInit {
  image: string;

  constructor(private ngZone: NgZone) { }

  ngOnInit() {
  }


  async takePhoto() {
    const photo = await Camera.getPhoto({
      source: CameraSource.Camera,
      quality: 90,
      // allowEditing: true,
      resultType: CameraResultType.DataUrl // Base64 (url encoded)
    });

    this.ngZone.run(() => this.image = photo.dataUrl);
  }

  async pickFromGallery() {
    const photo = await Camera.getPhoto({
      source: CameraSource.Photos,
      resultType: CameraResultType.DataUrl
    });

    this.image = photo.dataUrl;
  }
}
```

## Clipboard

The Clipboard Plugin allows the app to access the device clipboard and copy (write) or paste (read) content.

```
...
import { Plugins } from '@capacitor/core';
import { ToastController } from '@ionic/angular';
const { Clipboard } = Plugins;

...
export class ClipboardPage implements OnInit {
  text = '';

  constructor(private toastCtrl: ToastController) { }

  ngOnInit() {
  }

  async copy() {
    Clipboard.write({
      string: this.text
    });

    const toast = await this.toastCtrl.create({
      message: 'Text copied',
      duration: 1500,
      position: 'middle',
    });
    toast.present();
  }

  async paste() {
    const result = await Clipboard.read({
      type: 'string'
    });

    this.text = result.value;
  }

}
```

## Device

Using the Device Plugin you can get general information from the device, like the manufacturer, operating system and version, battery level, etc.

```
...
import { Plugins, DeviceInfo } from '@capacitor/core';
const { Device } = Plugins;
...
export class DevicePage implements OnInit {
  info: DeviceInfo;

  constructor() { }

  async ngOnInit() {
    this.info = await Device.getInfo();
  }
}
```

## Filesystem

We can read, write, copy, delete files (and directories) in the device using the Filesystem Plugin.

```
...
import { FilesystemDirectory, Plugins, FilesystemEncoding } from
'@capacitor/core';
const { Filesystem } = Plugins;
...
export class FilesystemPage implements OnInit {
  files: string[] = [];
  constructor() {}

  async ngOnInit() {
    await Filesystem.writeFile({
      path: 'secrets/text.txt',
      data: 'This is a test',
      directory: FilesystemDirectory.Documents,
      encoding: FilesystemEncoding.UTF8
    });

    try {
      const result = await Filesystem.readdir({
        path: 'secrets',
        directory: FilesystemDirectory.Documents,
      });
      this.files = result.files;
    } catch (e) {
      console.error('Unable to read dir', e);
    }
  }
}
```

## Geolocation

The Geolocation Plugin gets you current location using the GPS (and if not available, uses other sources like WiFi).

Check that you have the following permissions in AndroidManifest.xml (the last line is optional, only if you want that your app requires a GPS):

```
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-feature android:name="android.hardware.location.gps" />
```

If you use Mapbox to display a map, include this in the polyfill.ts file:

**(window as any).global = window;**

This is how we get the current location:

```
...
import { MapComponent } from 'ngx-mapbox-gl';
import { Plugins } from '@capacitor/core';
const { Geolocation } = Plugins;
...
export class GeolocationPage implements OnInit, AfterViewInit {
  lat = 0;
  lng = 0;
  @ViewChild(MapComponent) mapComp: MapComponent;
  constructor() { }

  async ngOnInit() {
    const coordinates = await Geolocation.getCurrentPosition();
    this.lat = coordinates.coords.latitude;
    this.lng = coordinates.coords.longitude;
  }

  ngAfterViewInit() {
    this.mapComp.load.subscribe(
      () => {
        this.mapComp.mapInstance.resize(); // Necessary for full height
      }
    );
  }
}
```

```
<ion-content>
  <mgl-map
    [style]="'mapbox://styles/mapbox/streets-v11'"
    [zoom]="[14]"
    [center]="[lng, lat]"
  >
    <mgl-marker #marker [lngLat]="[lng, lat]"></mgl-marker>
    <mgl-popup [marker]="marker">
      You are here
    </mgl-popup>
  </mgl-map>
</ion-content>
```

## Keyboard

You can display or hide the keyboard anytime with the Keyboard Plugin. You can also listen to keyboard events.

```
...
import { Plugins } from '@capacitor/core';
const { Keyboard } = Plugins;
...
export class KeyboardPage implements OnInit {
  visible = false;

  constructor() { }

  ngOnInit() {
    Keyboard.addListener('keyboardDidShow', info => this.visible = true);
    Keyboard.addListener('keyboardDidHide', () => this.visible = false);
  }

  showKeyboard() {
    Keyboard.show();
  }

  hideKeyboard() {
    Keyboard.hide();
  }
}
```

## Local Notifications

Local notifications are useful to remind a user about anything in the notification bar. As opposed to push notifications (sent from the server), these notifications generate when the app is running.

You can create notifications with sound, icons, etc. If the app isn't in the foreground when the user taps the notification, it will be opened.

```
...
import { Plugins } from '@capacitor/core';
const { LocalNotifications } = Plugins;
...
export class LocalNotificationsPage implements OnInit {
  message = '';
  triggered = false;
  scheduled = false;

  constructor(private ngZone: NgZone) {}

  ngOnInit() {
    // Not implemented yet! (Capacitor 1.4)
    // https://github.com/ionic-team/capacitor/issues/2352
    LocalNotifications.addListener('localNotificationReceived', notif => {
      this.ngZone.run(() => {
        this.triggered = true;
        this.scheduled = false;
      });
    });
  }

  async createNotification() {
    await LocalNotifications.schedule({
      notifications: [
        {
          id: 1, // Unique id
          title: this.message,
          body: 'This is a local notification generated by Capacitor',
          schedule: { at: new Date(new Date().getTime() + 10000) },
          sound: null
        }
      ]
    });
    this.scheduled = true;
    this.triggered = false;
  }

  async cancelNotification() {
    try {
      await LocalNotifications.cancel(await LocalNotifications.getPending());
    } finally {
      this.scheduled = false;
    }
  }
}
```

## Modals

If we want to show native dialogs (alert, prompts, confirm) or action sheet, we can use the Modal Plugin.

```
...
import { Plugins, ActionSheetOptionStyle } from '@capacitor/core';
const { Modals } = Plugins;
...
export class ModalsPage implements OnInit {
  confirm = false;
  name = '';
  option = -1;

  constructor() {}
  ngOnInit() {}

  async showAlert() {
    await Modals.alert({
      title: 'Hello',
      message: 'This is an alert'
    });
  }

  async showConfirm() {
    const result = await Modals.confirm({
      title: 'Confirm',
      message: 'Are you going to develop your next app with Ionic?'
    });

    this.confirm = result.value;
  }

  async showPrompt() {
    const result = await Modals.prompt({
      title: 'Hello',
      message: 'What\'s your name?'
    });

    if (!result.cancelled) {
      this.name = result.value;
    }
  }

  async showActions() {
    const result = await Modals.showActions({
      title: 'Photo Options',
      message: 'Select an option to perform',
      options: [
        {
          title: 'Upload'
        },
        {
          title: 'Share'
        },
        {
          title: 'Remove',
          style: ActionSheetOptionStyle.Destructive
        },
        {
          title: 'Cancel',
          style: ActionSheetOptionStyle.Cancel
        }
      ]
    });

    this.option = result.index;
  }
}
```

## Motion

The Motion plugin lets you track the device accelerometer and the orientation (compass).

```
...
import { Plugins, PluginListenerHandle, MotionOrientationEventResult } from
'@capacitor/core';
const { Motion } = Plugins;
...
export class MotionPage implements OnInit, OnDestroy {
  accelListener: PluginListenerHandle;
  orientListener: PluginListenerHandle;
  acceleration = { x: 0, y: 0, z: 0};
  orientation = {alpha: 0, beta: 0, gamma: 0};

  constructor() { }

  ngOnInit() {
    this.accelListener = Motion.addListener('accel', event => {
      this.acceleration = event.acceleration;
    });
    this.orientListener = Motion.addListener('orientation', event => {
      this.orientation.alpha = event.alpha;
      this.orientation.beta = event.beta;
      this.orientation.gamma = event.gamma;
    });
  }

  ngOnDestroy() {
    this.accelListener.remove();
    this.orientListener.remove();
  }
}
```

## Network status

You can check every time the device is connected or disconnected from the internet, and even which type of connection is using (Wi-fi, 4G, …) with the Network Plugin.

Check that you have the permission in the AndroidManifest.xml file:

```
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
```

```
import { Component, OnInit, OnDestroy, NgZone } from '@angular/core';
import { Plugins, PluginListenerHandle, NetworkStatus } from '@capacitor/core';
const { Network } = Plugins;
...
export class NetworkPage implements OnInit, OnDestroy {
  type = 'none';
  connected = false;
  connHandler: PluginListenerHandle;

  constructor(private zone: NgZone) { }

  async ngOnInit() {
    this.setConnection(await Network.getStatus());

    this.connHandler = Network.addListener('networkStatusChange', status => {
      this.setConnection(status)
    });
  }

  ngOnDestroy() {
    this.connHandler.remove();
  }

  setConnection(status: NetworkStatus) {
    this.zone.run(() => {
      this.connected = status.connected;
      this.type = status.connectionType;
    });
  }
}
```

## Share

The Share plugin allows the user to share some content in any sharing apps installed on the device (email, whatsapp, twitter, etc.).

```
import { Component, OnInit } from '@angular/core';
import { Plugins } from '@capacitor/core';
const { Share } = Plugins;
...
export class SharePage {
  message = '';

  constructor() { }

  share() {
    Share.share({
      dialogTitle: 'Share with others!',
      text: this.message
    });
  }
}
```

## Storage

In mobile devices, Local Storage can be periodically cleared, so it's better to use native storage. The Storage Plugin provides key/value storage using UserDefaults on iOs and SharedPreferences on Android (and localStorage when running as a web app). More advanced data storage options like SQlite can be used with other plugins.

Data is stored in string format, however we can store JSON objects if we serialize them using JSON.stringify (and JSON.parse to get the object back).

```
...
import { ToastController } from '@ionic/angular';
import { Plugins } from '@capacitor/core';
const { Storage } = Plugins;
...
export class StoragePage implements OnInit {
  name = '';

  constructor(private toastCtrl: ToastController) { }

  async ngOnInit() {
    const {value} = await Storage.get({key: 'name'});
    if (value) {
      this.name = value;
    }
  }

  async save() {
    await Storage.set({key: 'name', value: this.name});

    const toast = await this.toastCtrl.create({
      message: 'Name saved!',
      duration: 1500,
      position: 'middle',
    });
    toast.present();
  }
```

## Toast

Instead of Ionic's toast component, we can show native Toasts using the Toast Plugin. These native toast notifications are more limited in options.

```
...
import { Plugins } from '@capacitor/core';
const { Toast } = Plugins;
...
export class ToastPage implements OnInit {
...
  async showToast() {
    await Toast.show({
      text: 'I\'m a toast notification',
      duration: 'short'
    });
  }

}
```

## App

The App Plugin manages high level app state and events. You can exit the app from here, open other apps (using url), exit the app, manage back button (Android) and manage general app events.

```
...
import { Plugins, AppUrlOpen, PluginListenerHandle } from '@capacitor/core';
const { App, Modals } = Plugins;
...
export class AppPage implements OnInit, OnDestroy {
  backHandler: PluginListenerHandle;

  constructor() {}

  ngOnInit() {
    this.backHandler = App.addListener('backButton', async (data: AppUrlOpen) =>
    {
      const resp = await Modals.confirm({
        title: 'Close app',
        message: 'Do you really want to exit?',
        okButtonTitle: 'Yes, please',
        cancelButtonTitle: 'Never'
      });

      if (resp.value) {
        App.exitApp();
      }
    });
  }

  ngOnDestroy() {
    this.backHandler.remove();
  }
}
```

## Navigation

This is not an official Capacitor plugin. This Navigation plugin lets you open the native navigation app (car, bicycle, walking, etc…) to go to some location.

**npm install --save @proteansoftware/capacitor-start-navigation**

**npx cap sync**

There's an extra step in Android. Registering the plugin in the main Activity (app → java → io.ionic.starter → MainActivity). You may have to reopen Android Studio first:

```java
...
import com.servicesight.capacitor.startnavigation.StartNavigationPlugin;

public class MainActivity extends BridgeActivity {
  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    // Initializes the Bridge
    this.init(savedInstanceState, new ArrayList<Class<? extends Plugin>>() {{
      // Additional plugins you've installed go here
      // Ex: add(TotallyAwesomePlugin.class);
      add(StartNavigationPlugin.class);
    }});
  }
}
```

```typescript
...
import { Plugins } from '@capacitor/core';
import { Result } from 'ngx-mapbox-gl/lib/control/geocoder-control.directive';
const { Geolocation, StartNavigationPlugin } = Plugins;

...
export class DrivingDirectionsPage implements OnInit, AfterViewInit {
  lat = 0;
  lng = 0;
  @ViewChild(MapComponent) mapComp: MapComponent;

  ...
  startNavigation() {
    StartNavigationPlugin.launchMapsApp({
      latitude: this.lat,
      longitude: this.lng,
      name: 'Directions example',
    });
  }
}
```

## Google Login

The Google login plugin is not officially included in Capacitor, but there are some third-party plugins already made. On the Capacitor documentation (community plugins) you can find this plugin:

https://github.com/CodetrixStudio/CapacitorGoogleAuth

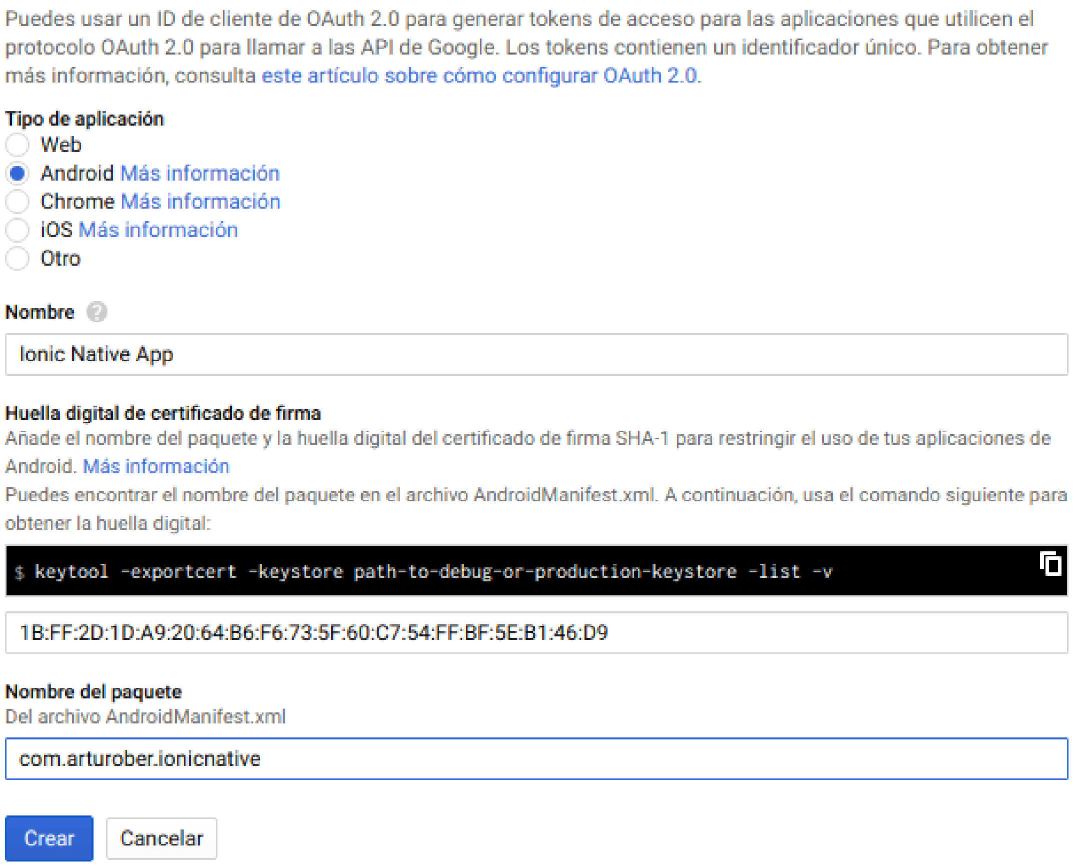
First of all, you should generate a fingerprint of your app signing certificate. Use the keystore created for publishing the app or the debug keystore (we'll use the debug one in this example → Password: android). Instructions here.

**Linux →** `keytool -list -v -alias androiddebugkey -keystore ~/.android/debug.keystore`

**Windows** → `keytool -list -v -alias androiddebugkey -keystore %USERPROFILE%\.android\debug.keystore`

After that, go to the **config.xml** file and change the widget id (default 'io.ionic.starter'), and give the app a unique name (usually the reverse company domain and the name of the app) like '**com.arturober.ionicnative**'.

Go to the Google developers console and create new **OAuth** credentials. Select Android (also create a web credential if you haven't done so already), and fill the necessary data with an app name (can be anything), the SHA1 fingerprint generated before, and the config.xml widget's id.



Puedes usar un ID de cliente de OAuth 2.0 para generar tokens de acceso para las aplicaciones que utilicen el protocolo OAuth 2.0 para llamar a las API de Google. Los tokens contienen un identificador único. Para obtener más información, consulta este artículo sobre cómo configurar OAuth 2.0.

Tipo de aplicación
- ○ Web
- ● Android Más información
- ○ Chrome Más información
- ○ iOS Más información
- ○ Otro

Nombre
Ionic Native App

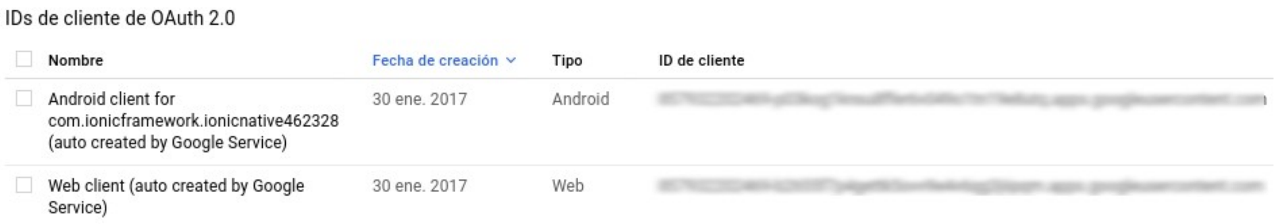
Huella digital de certificado de firma
Añade el nombre del paquete y la huella digital del certificado de firma SHA-1 para restringir el uso de tus aplicaciones de Android. Más información
Puedes encontrar el nombre del paquete en el archivo AndroidManifest.xml. A continuación, usa el comando siguiente para obtener la huella digital:

```
$ keytool -exportcert -keystore path-to-debug-or-production-keystore -list -v
```

1B:FF:2D:1D:A9:20:64:B6:F6:73:5F:60:C7:54:FF:BF:5E:B1:46:D9

Nombre del paquete
Del archivo AndroidManifest.xml
com.arturober.ionicnative

You'll need the client ID generated with the credentials.



IDs de cliente de OAuth 2.0

| Nombre | Fecha de creación | Tipo | ID de cliente |
|---|---|---|---|
| Android client for com.ionicframework.ionicnative462328 (auto created by Google Service) | 30 ene. 2017 | Android | [illegible] |
| Web client (auto created by Google Service) | 30 ene. 2017 | Web | [illegible] |

Now we can proceed to install the plugin.

**npm i --save @codetrix-studio/capacitor-google-auth**

**npx cap sync**

## Web instructions

Add this line in your **index.html** file:

```
<meta name="google-signin-client_id" content="{your web client id here}">
```

import the plugin in your **polyfill.ts** file:

```
import "@codetrix-studio/capacitor-google-auth";
```

## Android instructions

Inside the **strings.xml** file:

```
<resources>
    ...
    <string name="server_client_id">web client id</string>
</resources>
```

Register plugin inside your **MainActivity.onCreate**:

```
...
import com.codetrixstudio.capacitor.GoogleAuth.GoogleAuth;

public class MainActivity extends BridgeActivity {
  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    // Initializes the Bridge
    this.init(savedInstanceState, new ArrayList<Class<? extends Plugin>>() {{
      ...
      add(GoogleAuth.class);
    }});
  }
}
```

Add this configuration in **capacitor.config.json** file:

```
{
  ...
  "plugins": {
    "GoogleAuth": {
      "scopes": ["profile", "email"],
      "serverClientId": "web client id"
    }
  }
}
```

We can use the plugin now:

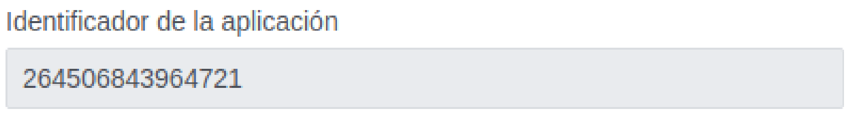
```
import { Component, OnInit } from '@angular/core';
import { Plugins } from '@capacitor/core';
const {GoogleAuth} = Plugins;

...
export class GoogleLoginPage implements OnInit {
  user = null;

  constructor() { }
  ngOnInit() {
  }

  async login() {
    try {
      this.user = await Plugins.GoogleAuth.signIn();
    } catch (err) {
      console.error(err);
    }
  }
}
```

## Facebook Login

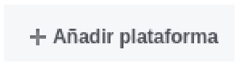
This plugin is also not included in Capacitor. Allows the app to access the native Facebook application (if not installed it will open a browser and let the user log in there), and among other things, perform authentication getting a token.

https://github.com/rdlabo/capacitor-facebook-login

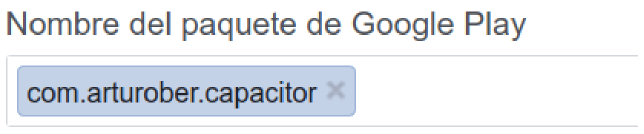
To configure this plugin to work with your app, you'll need to have a project created in the Facebook developers page. After creating the new project, you'll need the Application's ID (Go to configuration → Basic information). Also set the Apps package name (<widget> id in config.xml).


Identificador de la aplicación
264506843964721

After that we'll go to the project's configuration (basic information) and add the desired platform (Android, iOS). In this case, we'll choose Android.



Finally, we just need to include the app's package name in the Android configuration (app ID).


Nombre del paquete de Google Play
com.arturober.capacitor

We can now install the plugin

**npm i @rdlabo/capacitor-facebook-login**

**npx cap sync**

In Android (syncronize Gradle first or reopen Android Studio after adding a plugin), we need to register this plugin in the Main Activity.

```
...
import jp.rdlabo.capacitor.plugin.facebook.FacebookLogin;
```

```
public class MainActivity extends BridgeActivity {
  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    // Initializes the Bridge
    this.init(savedInstanceState, new ArrayList<Class<? extends Plugin>>() {{
      ...
      add(FacebookLogin.class);
    }});
  }
}
```

Then, we must include these activities inside **AndroidManifest.xml** (inside manifest/application):

```
<manifest ...>
    <application
         ...>
       ...
        <meta-data android:name="com.facebook.sdk.ApplicationId"
            android:value="@string/facebook_app_id"/>

        <activity
            android:name="com.facebook.FacebookActivity"
            android:configChanges="keyboard|keyboardHidden|screenLayout|screenSize|
orientation"
            android:label="@string/app_name" />

        <activity
            android:name="com.facebook.CustomTabActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.VIEW" />
                <category android:name="android.intent.category.DEFAULT" />
                <category android:name="android.intent.category.BROWSABLE" />
                <data android:scheme="@string/fb_login_protocol_scheme" />
            </intent-filter>
        </activity>
       ...
    </application>
    ...
</manifest>
```

And finally add this in the **strings.xml** file (replace [APP_ID] with your ID):

```
<resources>
    ...
    <string name="facebook_app_id">[APP_ID]</string>
    <string name="fb_login_protocol_scheme">fb[APP_ID]</string>
</resources>
```

We're now ready to go.

```
...
import { Plugins } from '@capacitor/core';
import { FacebookLoginResponse } from '@rdlabo/capacitor-facebook-login';
const { FacebookLogin } = Plugins;

...
export class FacebookLoginPage implements OnInit {
  accessToken = '';
  constructor() {}

  async ngOnInit() {
    const resp = await FacebookLogin.getCurrentAccessToken() as
FacebookLoginResponse;
    if (resp.accessToken) {
      this.accessToken = resp.accessToken.token;
    }
  }

  async login() {
    const resp = await FacebookLogin.login({ permissions: ['email'] }) as
FacebookLoginResponse;
    if (resp.accessToken) {
      this.accessToken = resp.accessToken.token;
    }
  }

  async logout() {
    await FacebookLogin.logout();
    this.accessToken = '';
  }
}
```

## Barcode Scanner (Cordova plugin)

Current Capacitor QR/barcode scanner plugins are not very mature, and for this example we're going to use a Cordova plugin.

https://github.com/enesyalcin/cordova-plugin-qrscanner

**npm i phonegap-plugin-barcodescanner**

**npx cap sync** (it will find a Cordova plugin and add to the project)

It's always better to use Capacitor plugins as some Cordova plugins have problems with Java libraries, etc. With this plugin, there's a duplicate library, and we need to add this to the **build.gradle** (module:app) file.

```
android {
    ...
    configurations {
        compile.exclude group: 'com.google.zxing'
    }
}
```

Ionic Native integration doesn't work either when using this plugin with Capacitor, so we will use in "JavaScript" mode and not in Angular mode. We need to access the global cordova object from window and need to tell TypeScript to treat window global object as "any" type.

```
…

declare let window: any;
...
export class BarcodeScannerPage {
  data = '';

  constructor(private ngZone: NgZone) { }
  async scan() {
    // this.data = (await this.bcScanner.scan()).text;
    window.cordova.plugins.barcodeScanner.scan(
      result => this.ngZone.run(() => this.data = result),
      err => console.error(err),
      {
        showTorchButton: true,
        prompt: 'Scan your code',
        // formats: "QR_CODE",
        resultDisplayDuration: 0
      }
    );
  }
}
```

## Flashlight (Cordova plugin)

Currently, there's no plugin in Capacitor to turn on the camera's flashlight. We're goin to install the Cordova plugin instead.

**npm i cordova-plugin-flashlight**

**npx cap sync**

We'll need to do the same with the window object as in the barcode plugin.

```
import { Component, OnInit, OnDestroy, NgZone } from '@angular/core';
declare let window: any;
...
export class FlashlightPage implements OnDestroy {
  on = false;

  constructor(private ngZone: NgZone) { }

  async toggleFlash() {
    await window.plugins.flashlight.toggle(() => {
      this.ngZone.run(() => this.on = window.plugins.flashlight.isSwitchedOn());
    });
  }

  ngOnDestroy() {
    window.plugins.flashlight.switchOff();
  }
}
```

## SQLite

SQLite is an  embedded SQL database engine. It's an small library that works with disk files directly and doesn't have a server process running. It supports multiple tables, indices, triggers, and views.

We're going to install this package that includes an SQLite plugin (and others):

https://www.npmjs.com/package/@jeepq/capacitor

**npm install @jeepq/capacitor@latest**

After installing the package, we're going to register the necessary Plugin in Android's project Main Activity.

```java
...
import com.jeep.plugins.capacitor.CapacitorSQLite;

public class MainActivity extends BridgeActivity {
  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    // Initializes the Bridge
    this.init(savedInstanceState, new ArrayList<Class<? extends Plugin>>() {{
      ...
      add(CapacitorSQLite.class);
    }});
  }
}
```

We can use this plugin now in Ionic:

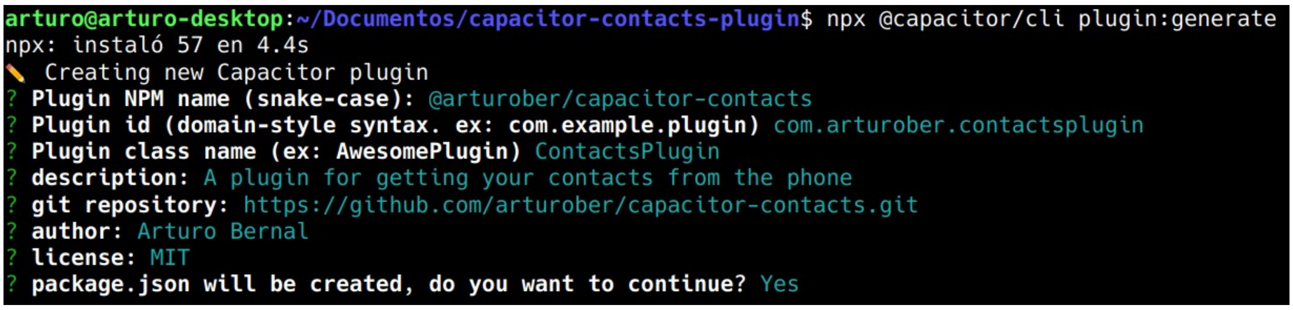
```typescript
import { Component, OnInit, OnDestroy, ViewChild } from '@angular/core';
import { Plugins } from '@capacitor/core';
import { NgForm } from '@angular/forms';
const { CapacitorSQLite } = Plugins;

@Component({
  selector: 'app-sqlite',
  templateUrl: './sqlite.page.html',
  styleUrls: ['./sqlite.page.scss'],
})
export class SqlitePage implements OnInit, OnDestroy {
  @ViewChild('personForm') personForm: NgForm;
  open = false;
  persons: { id?: number, name: string, age: number }[] = [];
  person: { id?: number, name: string, age: number } = {
    name: '',
    age: null
  };

  constructor() { }

  async ngOnInit() {
    const opened = await CapacitorSQLite.open({database: 'testsqlite'});
    if (!opened.result) { return; }

    this.open = true;
    await CapacitorSQLite.run({statement: `CREATE TABLE IF NOT EXISTS person (
      id integer primary key,
      name text not null,
      age integer not null)`});

    const result = await CapacitorSQLite.query({
      statement: 'SELECT * FROM person'
    });
    this.persons = result.values;
  }

  async ngOnDestroy() {
    if (this.open) {
      await CapacitorSQLite.close({database: 'testsqlite'});
    }
  }

  async add() {
    if (!this.open) { return; }

    const addRes = await CapacitorSQLite.run({
      statement: 'INSERT INTO person (name, age) VALUES (?,?)',
      values: [this.person.name, this.person.age]
    });
    const idRes = await CapacitorSQLite.query({
      statement: 'SELECT last_insert_rowid()'
  });

    this.person.id = +Object.values(idRes.values[0])[0];
    this.persons.push(this.person);
    this.person = { name: '', age: null };
    this.personForm.resetForm();
  }

  async remove(person, index: number) {
    if (!this.open) { return; }

    const delRes = await CapacitorSQLite.run({
      statement: 'DELETE FROM person WHERE id = ?',
      values: [person.id]
    });

    if (delRes.changes > 0) {
      this.persons.splice(index, 1);
    }
  }
}
```

## Creating a Capacitor plugin

Creating  a Capacitor plugin is not very difficult if you know how to develop with Android / iOS, because that is what you are going to do.

You must create a new project using this command:

**npx @capacitor/cli plugin:generate**



```
arturo@arturo-desktop:~/Documentos/capacitor-contacts-plugin$ npx @capacitor/cli plugin:generate
npx: instaló 57 en 4.4s
✏ Creating new Capacitor plugin
? Plugin NPM name (snake-case): @arturober/capacitor-contacts
? Plugin id (domain-style syntax. ex: com.example.plugin) com.arturober.contactsplugin
? Plugin class name (ex: AwesomePlugin) ContactsPlugin
? description: A plugin for getting your contacts from the phone
? git repository: https://github.com/arturober/capacitor-contacts.git
? author: Arturo Bernal
? license: MIT
? package.json will be created, do you want to continue? Yes
```

The android folder contains the Android plugin (ios → iOS, src → web). You'll need to open the android folder with Android Studio to develop this plugin. More info about creating plugins: https://capacitor.ionicframework.com/docs/plugins

For this example, I'm going to create a plugin that retrieves all contacts in the phone (name and phones). You can check the source code here:

https://github.com/arturober/capacitor-contacts

You can publish this plugin in NPM by running:

**npm publish –access=public** (private NPM repositories aren't free)

After changing anything, don't forget to change the version before publishing again. For example: **npm version patch** → https://docs.npmjs.com/cli/version

You can now proceed with the installation on any project:

**npm install @arturober/capacitor-contacts@latest**

Request necessary permissions:

```
<uses-permission android:name="android.permission.READ_CONTACTS" />
```

Add the plugin in your activity:

```
...
import com.arturober.contactsplugin.ContactsPlugin;

public class MainActivity extends BridgeActivity {
  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    // Initializes the Bridge
    this.init(savedInstanceState, new ArrayList<Class<? extends Plugin>>() {{
      ...
      add(ContactsPlugin.class);
    }});
  }
}
```

Get results and show them:

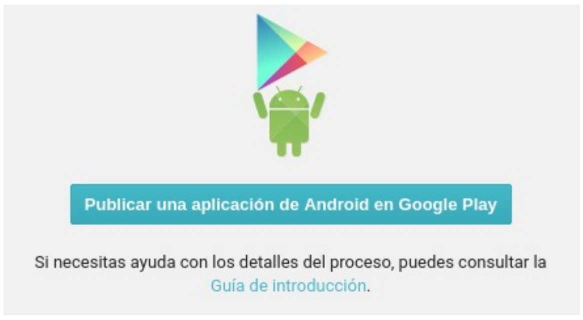
```
import { Component, OnInit } from '@angular/core';
import { Plugins } from '@capacitor/core';
import { ContactsResult } from '@arturober/capacitor-contacts';
const { ContactsPlugin } = Plugins;

@Component({
  selector: 'app-contacts',
  templateUrl: './contacts.page.html',
  styleUrls: ['./contacts.page.scss']
})
export class ContactsPage implements OnInit {
  result: ContactsResult;

  constructor() {}

  async ngOnInit() {
    this.result = await ContactsPlugin.getContacts() as ContactsResult;
  }
}
```

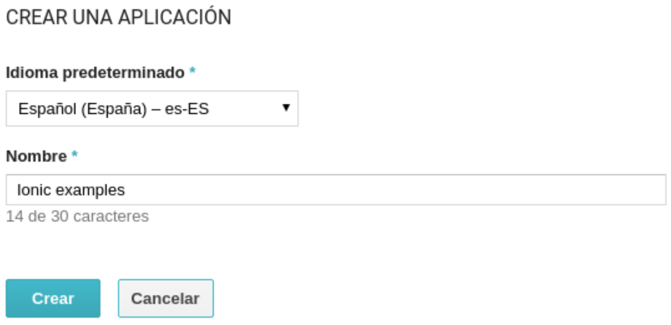
```
<ion-content class="ion-padding">
  <ion-list *ngIf="result">
    <ion-item *ngFor="let contact of result.contacts">
      <ion-label>
        <h3>{{contact.name}}</h3>
        <p>{{contact.phones?.join(', ')}}</p>
      </ion-label>
    </ion-item>
  </ion-list>
</ion-content>
```

## Publishing your app in Google Play

Before publishing your app, you'll need to create a Google Play account. You can do that going here: https://play.google.com/apps/publish/signup/. There's a unique fee you'll need to pay of $25 (unique payment).

Once you have registered and payed, publishing an app is very easy, first of all, go to "your applications" section or click on the button that says "Publish an Android application in Google Play".


Publicar una aplicación de Android en Google Play
Si necesitas ayuda con los detalles del proceso, puedes consultar la Guía de introducción.


CREAR UNA APLICACIÓN

Idioma predeterminado *
Español (España) – es-ES

Nombre *
Ionic examples
14 de 30 caracteres

You can publish an application in alpha or beta (for testing purposes, it will only be available to the people you choose), or in production (public). There are some limitations to public apps like for example, if your application sends user data, it must be through **SSL** (https), so your server should be configured to work that way.

Build your app for production with the –prod flag (if it's for production, of course):

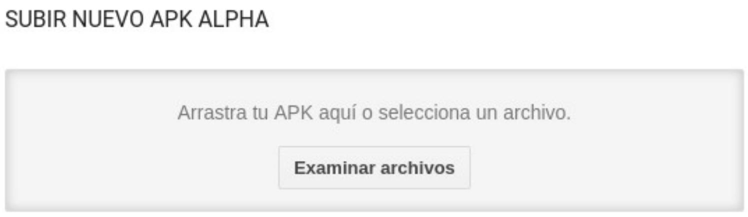
**ionic build –prod**

**npx cap sync**

Then, you'll have to create a keystore to sign your app before uploading it. You can chek instructions here:

https://developer.android.com/studio/publish/app-signing?hl=es_419

After signing the app, upload it to Google Play:



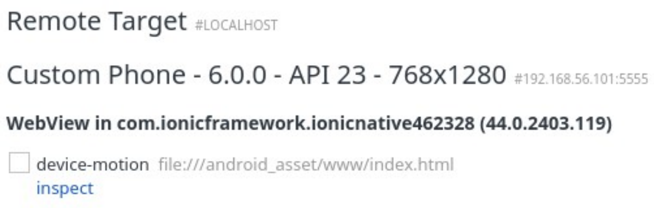
Before publishing your app and becoming available to others, you must complete the necessary information (description, icon, screenshots, etc.). Each time you want to upload a new version, you should first increment the version (build.gradle file, versionCode and versionName), compile the app and sign it again.

# Native app debug

## Chrome

We can debug an app running in our device or emulator from Chrome (must be connected to the computer). Here we can only debug the web side, but that's enough for the majority of cases, we can watch what's printed on the console, etc.

Go to this URL in Chrome → **chrome://inspect/#devices**

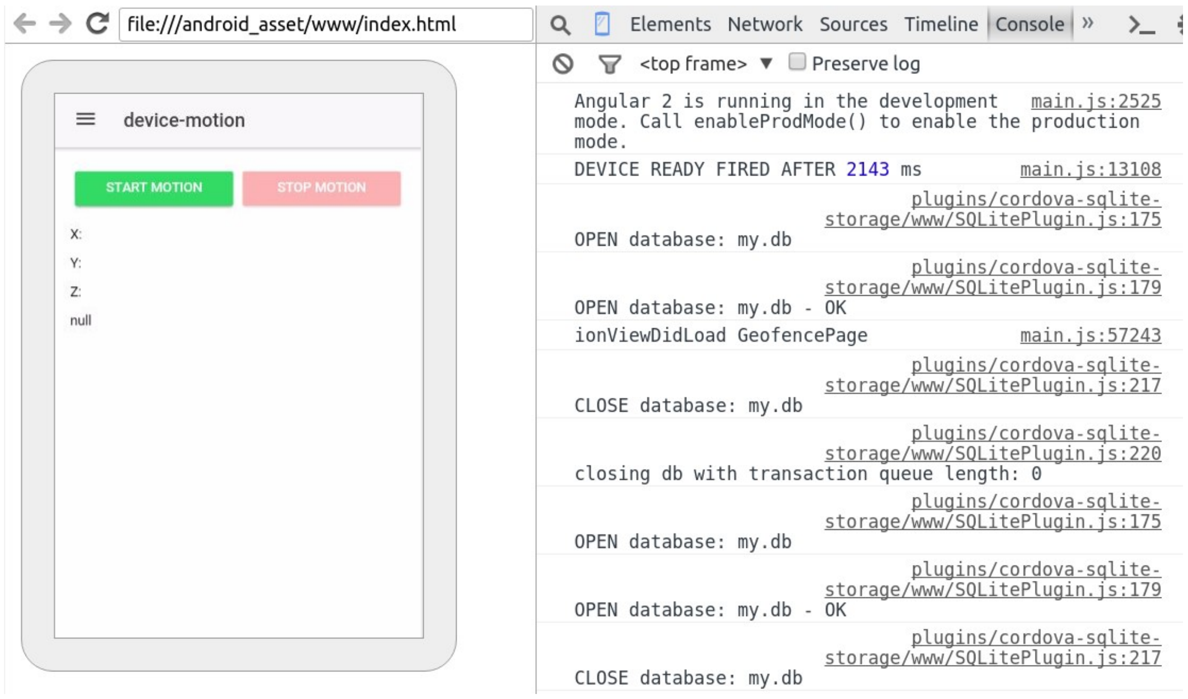


Remote Target #LOCALHOST

Custom Phone - 6.0.0 - API 23 - 768x1280 #192.168.56.101:5555

**WebView in com.ionicframework.ionicnative462328 (44.0.2403.119)**

device-motion file:///android_asset/www/index.html
inspect



```
Angular 2 is running in the development    main.js:2525
mode. Call enableProdMode() to enable the production
mode.
DEVICE READY FIRED AFTER 2143 ms          main.js:13108
                                   plugins/cordova-sqlite-
                          storage/www/SQLitePlugin.js:175
OPEN database: my.db
                                   plugins/cordova-sqlite-
                          storage/www/SQLitePlugin.js:179
OPEN database: my.db - OK
ionViewDidLoad GeofencePage               main.js:57243
                                   plugins/cordova-sqlite-
                          storage/www/SQLitePlugin.js:217
CLOSE database: my.db
                                   plugins/cordova-sqlite-
                          storage/www/SQLitePlugin.js:220
closing db with transaction queue length: 0
                                   plugins/cordova-sqlite-
                          storage/www/SQLitePlugin.js:175
OPEN database: my.db
                                   plugins/cordova-sqlite-
                          storage/www/SQLitePlugin.js:179
OPEN database: my.db - OK
                                   plugins/cordova-sqlite-
                          storage/www/SQLitePlugin.js:217
CLOSE database: my.db
```

a
d
b

We can also get the Android console output using either Android Studio or from the command line using **adb logcat** command. Execute **adb logcat --help** for more info. For example, if we want to print only error messages we shoud execute: **adb logcat *:E**.

## Android Studio

You can follow this instructions to open the project in Android Studio, launch and debug the application from there.