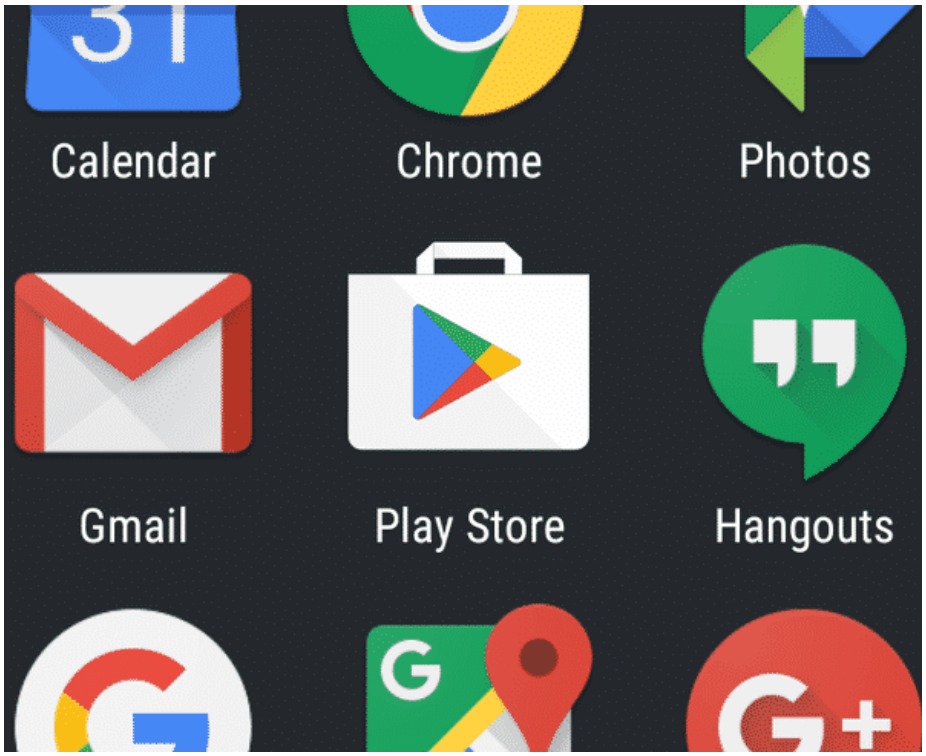

Figure 1:

Build and publish an Ngx-Rocket Cordova Ionic app in Play Store

How to build an Android app in Android Studio and publish it in the Google Play Store.

Convert this guide

```
$ pandoc -o output.docx -f markdown -t docx README.md   ## docx
$ pandoc -o output.pdf README.md                        ## pdf
```

APK file

APK refers to the Android Package Kit (also Android Application Package) which is used in the Android operating system for the distribution and installation

of mobile apps. For instance, you tend to use .exe file for installing software in your PC, similarly, .apk files are used to install Android application on your phone.

Signed APK file

As for the developer, an unsigned APK file is developed mainly for local testing purposes. Furthermore, these APK files can be made publicly available. However, Google Play Store does not accept these files as unsigned APK files are not secure. Here, an unsigned APK is more like a zipped file that can get unzipped easily without any security. So, in case if the unwanted individuals unzip the files and unzip the APK file, they can have access to the code because of which you may lose your authority to the files.

Now, talking about generating signed APK files, it is secured by a Keystore credential made by the developer and includes a password for the security purpose. Therefore, Signed APK cannot be easily unzipped and mainly used for production purposes. In conclusion, if you are generating a signed APK file, it is more secure and also acceptable in Google Play Store.

Guide to Generate a signed APK using Android Studio

First, clone the project, enter the correct branch and install it, npm install, etc. It's important to fix that the project should be a ngx-rocket project cordova-based with ionic.

To build Android apps you will need Android SDK. Remember to declare the paths of the sdk in your ~/.bashrc:

```
export ANDROID_SDK_ROOT=$HOME/Android/Sdk
export PATH=$PATH:$ANDROID_SDK_ROOT/tools/bin
export PATH=$PATH:$ANDROID_SDK_ROOT/platform-tools
export PATH=$PATH:$ANDROID_SDK_ROOT/emulator
export PATH=$PATH:$ANDROID_SDK_ROOT/build-tools
```

Open file `config.xml` and change the id in the `widget` tag to something like "com.yourcompany.yourappname". If you are updating an existing app, increment the version of the app in this file (else Google Play will not accept the build).

Requirements

```
$ sudo add-apt-repository ppa:cwchien/gradle    ## add a gradle ppa repository
$ sudo apt update                               ## update apt repositories to get the new pp
$ sudo apt install gradle zipalign apksigner    ## install all dependencies to build androic
$ npm i -g cordova                              ## install cordova globally
$ npm run cordova requirements                  ## check if all requirements are satisfied
```

Verify cordova versions, should be at least the following:

```
"dependencies": {
    "cordova-android": "^9.0.0",
    "cordova-custom-config": "^5.1.0",
    "cordova-ios": "^6.1.0",
    "cordova-plugin-device": "^2.0.3",
    "cordova-plugin-ionic-webview": "^5.0.0",
    "cordova-plugin-ionic-keyboard": "^2.2.0",
    "cordova-plugin-splashscreen": "^6.0.0",
    "cordova-plugin-statusbar": "^2.4.3",
    "cordova-plugin-whitelist": "^1.3.4"
},
"devDependencies": {
    "cordova": "^10.0.0"
}
```

After this, just do the following commands:

```
$ mkdir www                                    ## create www dir, where the build files wil
$ npm run cordova platform add android         ## add android platform
$ npm run cordova:prepare android              ## prepare build
```

If you are creating your first app and have something like pusher notifications,
you will need to generate a Firebase app.

```
Firebase part
Add procedure:
    Go to https://firebase.google.com/
    Click on "GO TO CONSOLE"
    Click on "Add Project"
    Project name: Enter: sample-app
    Click "Create Project" [Takes about 10 seconds or so...]
    Click "Continue"
    Click in "Project settings" in the left top, click "Add Firebase to your Android app" (A
    Enter package name for the android app (Same as the id in config.xml if you have this fi
    Click on download google-services.json
    Move this file to /platforms/android/app

Remove procedure:
    Go to https://firebase.google.com/
    Click on "GO TO CONSOLE"
    Settings -> Project Settings -> Delete this App
    Settings -> Project Settings -> Delete Project
    Enter project ID and press delete
```

Finally, build the app:

```
$ npm run cordova:build android                ## build android
```

If build succeed, lets finish it:

```
## Enter the folder that have the .apk
$ cd platforms/android/app/build/outputs/apk/release/

## If you are in Windows, do this, else just go ahead
Copy file app-release-unsigned.apk
Go to folder C:\Program Files\Java\jdk1.8.0_231\bin
Delete app-release-unsigned.apk if file exists.
Delete my-release-key.keystore if file exists.
Paste file app-release-unsigned.apk
## Windows part ends

## If you are just updating the app, you can skip the next two commands and go right to jars
## Generate a signing key in JKS format and then convert it to PKCS12
$ keytool -genkey -v -keystore my-release-key.keystore -alias alias_name -keyalg RSA -keysiz
$ keytool -importkeystore -srckeystore my-release-key.keystore -destkeystore my-release-key.

## Once this last command has been ran and its prompts have been answered a file called my-r

###
### WARNING: Save this file and keep it somewhere safe. If it is
### lost the Google Play Store will not accept updates for this app!
###

## If you are in Windows do this, else just go ahead
Copy file app-release-unsigned.apk
Copy file my-release-key.keystore
Go to folder C:\Program Files (x86)\Android\android-sdk\build-tools\28.0.3
Delete app-release-unsigned.apk if file exists.
Delete HelloWorld.apk if file exists.
Delete my-release-key.keystore if file exists.
Delete Welever.apk if file exists.
Delete projecty-signed.apk if file exists.
Paste file app-release-unsigned.apk
Paste file my-release-key.keystore
## Windows part ends

## Optimize the APK
$ zipalign -v 4 app-release-unsigned.apk projecty-unsigned.apk

## Sign the app with JAR Signature Schema V1 and APK Signature Schema V2
$ apksigner sign --verbose --ks my-release-key.keystore --ks-key-alias alias_name --out proj

## Verify signature (ignore warnings)
$ apksigner verify --verbose projecty-signed.apk
```

```
## This generates a final release binary called projecty-signed.apk that can be accepted in
```

Publishing it in the Play Store

Preparing to release

Preparing your application for release is a multi-step process that involves the following tasks:

- At a minimum you need to remove Log calls and remove the android:debuggable attribute from your manifest file. You should also provide values for the android:versionCode and android:versionName attributes, which are located in the element. You may also have to configure several other settings to meet Google Play requirements or accommodate whatever method you're using to release your application.
- If you are using Gradle build files, you can use the release build type to set your build settings for the published version of your app.
- You can use the Gradle build files with the release build type to build and sign a release version of your application. See Building and Running from Android Studio.
- Before you distribute your application, you should thoroughly test the release version on at least one target handset device and one target tablet device.
- You need to be sure that all application resources such as multimedia files and graphics are updated and included with your application or staged on the proper production servers.
- If your application depends on external servers or services, you need to be sure they are secure and production ready.

You may have to perform several other tasks as part of the preparation process. For example, you will need to get a private key for signing your application. You will also need to create an icon for your application, and you may want to prepare an End User License Agreement (EULA) to protect your person, organization, and intellectual property.

Releasing your application on Google Play is a simple process that involves three basic steps:

- Preparing promotional materials.

  To fully leverage the marketing and publicity capabilities of Google Play, you need to create promotional materials for your application, such as screenshots, videos, graphics, and promotional text.

- Configuring options and uploading assets.

  Google Play lets you target your application to a worldwide pool of users and devices. By configuring various Google Play settings, you can choose the countries you want to reach, the listing languages you want to use, and the price you want to charge in each country. You can also configure listing details such as the application type, category, and content rating. When you are done configuring options you can upload your promotional materials and your application as a draft (unpublished) application.

- Publishing the release version of your application.

  If you are satisfied that your publishing settings are correctly configured and your uploaded application is ready to be released to the public, you can simply click Publish in the Play Console and within minutes your application will be live and available for download around the world.

The big release day

Requirements

For creating your own Google Play Developer Account, your age should be at least above 18 years old. If you pass this age criterion, you can use your Google account to sign up for a Developer Account. After then, you will have to accept the Developer Distribution Agreement and pay the one-time registration fee of $25. For Registration fee payment, you can use MasterCard, Visa, American Express, Discover (the U.S. only), Visa Electron (Outside of the U.S. only). Finally, you will be required to complete your account details.

> Signup here: https://play.google.com/apps/publish/signup
>
> Login here: https://play.google.com/apps/publish

After login, everything is intuitive. Will have a big blue button “Create app” which will guide you to publish your first app.

References

Positive stud - What is an APK File? Difference Between Building an Android APK and Generating Signed APK file. (Aug 11, 2020)

Google - Publish your app

Positive stud - How to publish an Android App on Google Play Store? (Aug 14, 2020)

Apksigner - Google

Leonardo Zanotti