



FLUTTER

Desplegar aplicaciones en las AppStores

Objetivo

Este documento tiene como finalidad ayudarte y guiarte en el proceso de generación y despliegue de tu aplicación de Flutter y subirla a la Google PlayStore y la Apple AppStore. Esta guía no reemplaza la guía oficial de flutter, es una ayuda

Fernando Herrera

Flutter – Preparar el despliegue:

Cambiar el ícono de la aplicación:

Se pueden crear manualmente los íconos uno a uno si se desea, pero utilizaremos este paquete:

Noten que la instalación es abajo del `dev_dependencies`

Enlace del paquete:

https://pub.dev/packages/flutter_launcher_icons#-readme-tab-

```
dev_dependencies:  
  flutter_test:  
    sdk: flutter  
  
  flutter_launcher_icons:
```

Al mismo nivel que `dev_dependencies`, configurar el paquete

```
flutter_icons:  
  android: "launcher_icon"  
  ios: true  
  image_path: "assets/icon/movie-icon.png"  
  adaptive_icon_background: "assets/icon/movie-icon.png"
```

Luego ejecutar estos comandos para crearlos

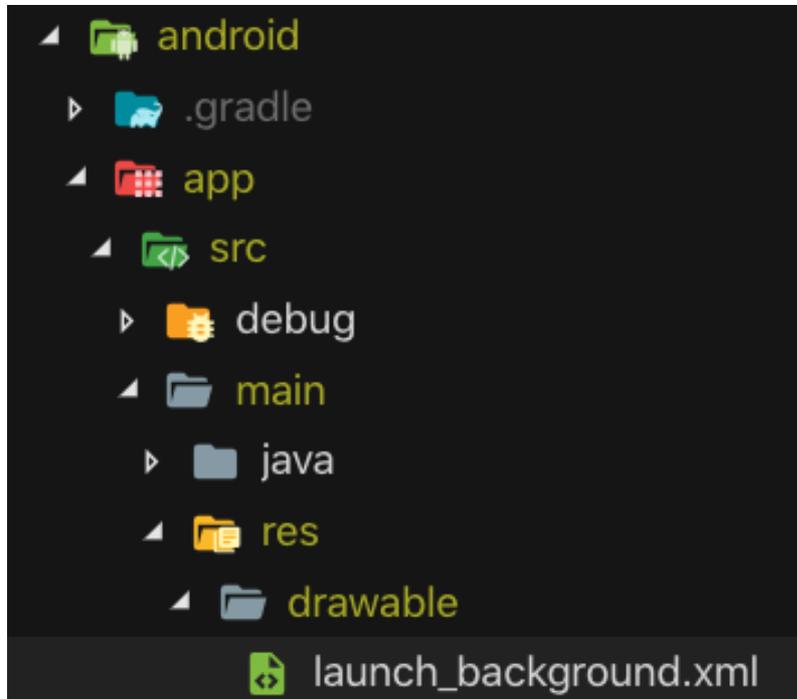
```
flutter packages get  
flutter packages pub run flutter_launcher_icons:main
```

Al hacer el build normal, se debería de apreciar el nuevo ícono

Generar Splash Screen

Android:

Abrir el archivo lunch_background.xml

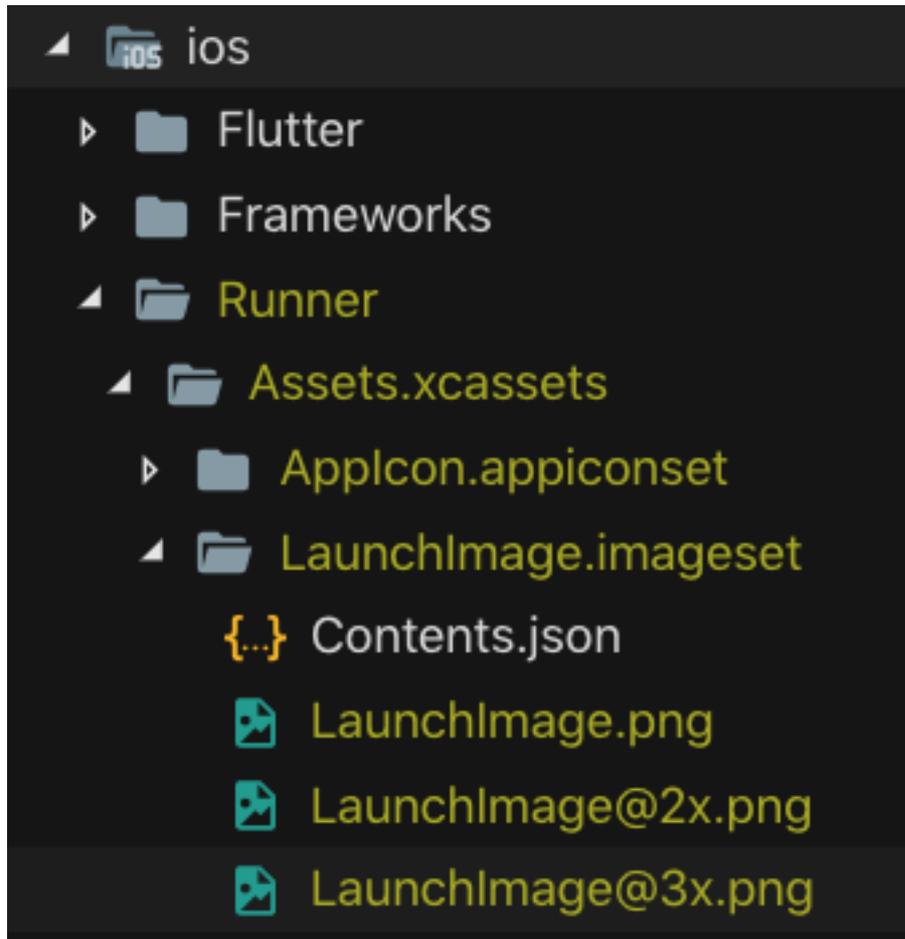


Realizar la siguiente configuración

```
<?xml version="1.0" encoding="utf-8"?>
<!-- Modify this file to customize your launch splash screen -->
<layer-list xmlns:android="http://schemas.android.com/apk/res/android">
  <item android:drawable="@android:color/black" />

  <!-- You can insert your own image assets here -->
  <item>
    <bitmap
      android:gravity="center"
      android:src="@mipmap/ic_launcher" />
  </item>
</layer-list>
```

Reemplazar estas 3 imágenes por las imágenes que deseas usar en un splash screen



Dimensiones de las imágenes:

Portrait:

1x = 768 x 1024

2x = 1536 x 2048

3x = 2304 x 3072

Landscape

1x = 1024 x 768

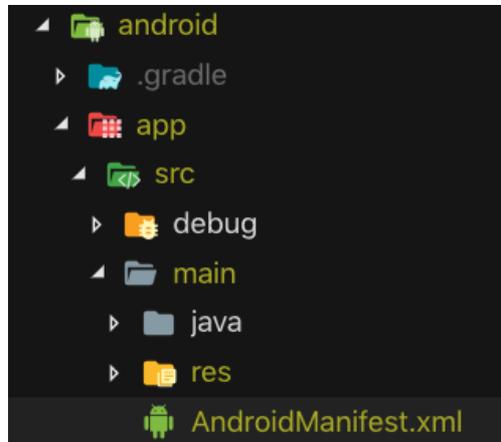
2x = 2048 x 1536

3x = 3072 x 2304

Ajustar el nombre de la aplicación

Android

Abrir el AndroidManifest.xml en la siguiente dirección

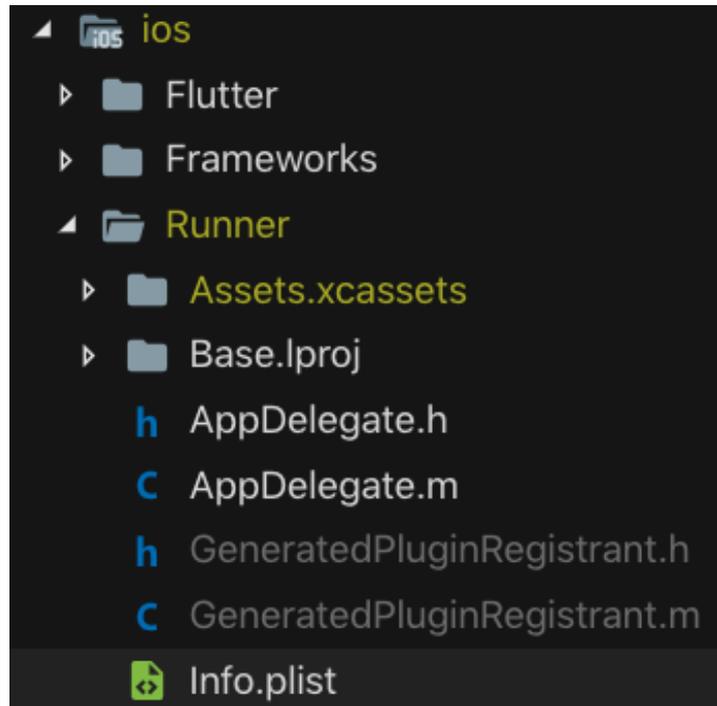


Dentro de este archivo pueden cambiar información relacionada a los accesos que esta app necesitará, pero lo que andamos buscando es el label de nuestra aplicación

```
<application  
    android:label="Películas"
```

IOS:

Abrir el archivo de Info.plist en la siguiente dirección



Aquí estamos buscando esta llave y cambiarla a tu gusto:

```
<key>CFBundleName</key>  
<string>Películas</string>
```



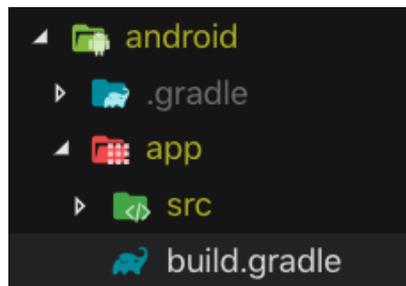
Nota:

La guía oficial se encuentra aquí:

<https://flutter.dev/docs/deployment/android>

Seguir los siguientes pasos:

1) Buscar y abrir el archivo build.gradle, el que se encuentra dentro de la carpeta app, (Cuidado, que hay varios archivos llamados igual)



2) En este archivo estamos buscando el ID de la aplicación

```
defaultConfig {  
    applicationId "com.example.peliculas"
```

Nota: este APPID debe ser ÚNICO a nivel mundial, adicionalmente también en el archivo AndroidManifest.xml, se encuentra el mismo ID, ambos deben de ser iguales

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"  
    package="com.example.peliculas">
```

3) Se puede manejar manualmente la versión cambiando esta línea:

```
versionCode flutterVersionCode.toInteger()
```

Por esta otra

```
versionCode 1
```

IMPORTANTE:

Este número es un entero, el orden como cambie, depende de ti.

La instrucción

```
versionName flutterVersionName
```

La cambiamos por la versión visible que los usuarios observarán en la tienda

Ejemplo:

```
versionName "1.0.0"
```

Nota:

Cada vez que se quiera desplegar una nueva versión de tu app a la tienda, DEBES incrementar el versioncode al menos en 1

4) Especificar la versión mínima para usar tu app (Android)

Esta lista les dará la idea de el minSdkVersion para usar tu aplicación, por si acaso necesitas una característica propia de alguna versión de Android

SDK_INT value	Build.VERSION_CODES	Human Version Name
1	BASE	Android 1.0 (no codename)
2	BASE_1_1	Android 1.1 Petit Four
3	CUPCAKE	Android 1.5 Cupcake
4	DONUT	Android 1.6 Donut
5	ECLAIR	Android 2.0 Eclair
6	ECLAIR_0_1	Android 2.0.1 Eclair
7	ECLAIR_MR1	Android 2.1 Eclair
8	FROYO	Android 2.2 Froyo
9	GINGERBREAD	Android 2.3 Gingerbread
10	GINGERBREAD_MR1	Android 2.3.3 Gingerbread
11	HONEYCOMB	Android 3.0 Honeycomb
12	HONEYCOMB_MR1	Android 3.1 Honeycomb
13	HONEYCOMB_MR2	Android 3.2 Honeycomb
14	ICE_CREAM_SANDWICH	Android 4.0 Ice Cream Sandwich
15	ICE_CREAM_SANDWICH_MR1	Android 4.0.3 Ice Cream Sandwich
16	JELLY_BEAN	Android 4.1 Jellybean
17	JELLY_BEAN_MR1	Android 4.2 Jellybean
18	JELLY_BEAN_MR2	Android 4.3 Jellybean
19	KITKAT	Android 4.4 KitKat
20	KITKAT_WATCH	Android 4.4 KitKat Watch
21	LOLLIPOP	Android 5.0 Lollipop
22	LOLLIPOP_MR1	Android 5.1 Lollipop
23	M	Android 6.0 Marshmallow
24	N	Android 7.0 Nougat
25	N_MR1	Android 7.1.1 Nougat
26	O	Android 8.0 Oreo
27	O_MR1	Android 8 Oreo MR1
28	P	Android Pie
10000	CUR_DEVELOPMENT	Current Development Version

5) Firmar la aplicación:

Documentación de la creación de la firma

<https://flutter.dev/docs/deployment/android#create-a-keystore>

```
keytool -genkey -v -keystore ~/key.jks -keyalg RSA -keysize 2048 -validity 10000 -alias key
```

Nota:

Este comando pedirá una contraseña y repetir la contraseña, (guarden eso en un lugar seguro)

Luego pide información personal, no es tan importante, pero deben de colocar sus datos reales.

Al finalizar, hay que escribir **yes**

Confirmar con una nueva contraseña (puede ser la misma de antes) y al finalizar, les dirá donde quedará guardado tu **key.jks**

Ejemplo:

```
Re-enter new password:  
[Storing /Users/strider/key.jks]  
StridersMac:peliculas striders$
```

6) Referir la llave desde tu APP:

Referencia:

<https://flutter.dev/docs/deployment/android#reference-the-keystore-from-the-app>

Crear el archivo: `<app dir>/android/key.properties`

IMPORTANTE:

Si estamos usando un sistema de control de versiones, este archivo NO DEBE de estar incluido en el repositorio, así que hay que agregarlo al `.gitignore`:

6.1) Abrir el archivo `.gitignore` y añadir este archivo `key.properties`

6.2) Copiar el contenido de la documentación, para llenar el `key.properties`

El `StoreFile`, es el path relativo desde el ROOT del sistema operativo o disco C

7) Configurar la firma en gradle:

Seguir estos pasos:

<https://flutter.dev/docs/deployment/android#configure-signing-in-gradle>

8) Generar la versión de producción de nuestra app:

Ejecutar el siguiente comando en la raíz de nuestro proyecto

```
flutter build apk --release
```

Nota:

Este proceso puede demorar un rato, dependiendo de la capacidad de tu computadora

Referencia

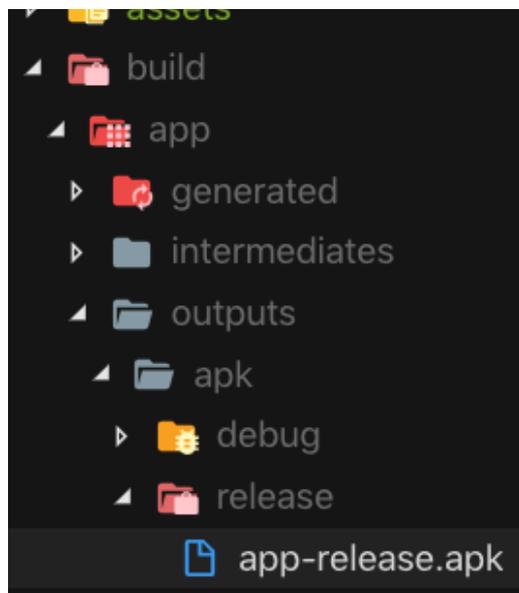
<https://flutter.dev/docs/deployment/android#building-a-release-apk>

Al finalizar, debería de decirles el PATH donde se acaba de generar el APK:

```
Running Gradle task 'assembleRelease'... Done
Built build/app/outputs/apk/release/app-release.apk (5.7MB).
StridersMac:peliculas striders$
```

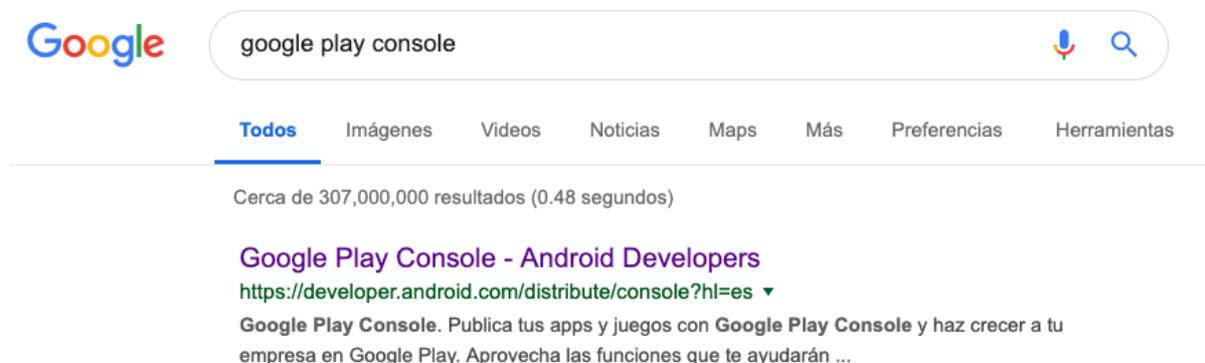
Nota:

El archivo se encuentra en una carpeta en la raíz del proyecto (NO en la carpeta Android)



9) Publicar el APK en Google PlayStore

Buscar en Google: Google Play Console



O navegar y registrarse aquí

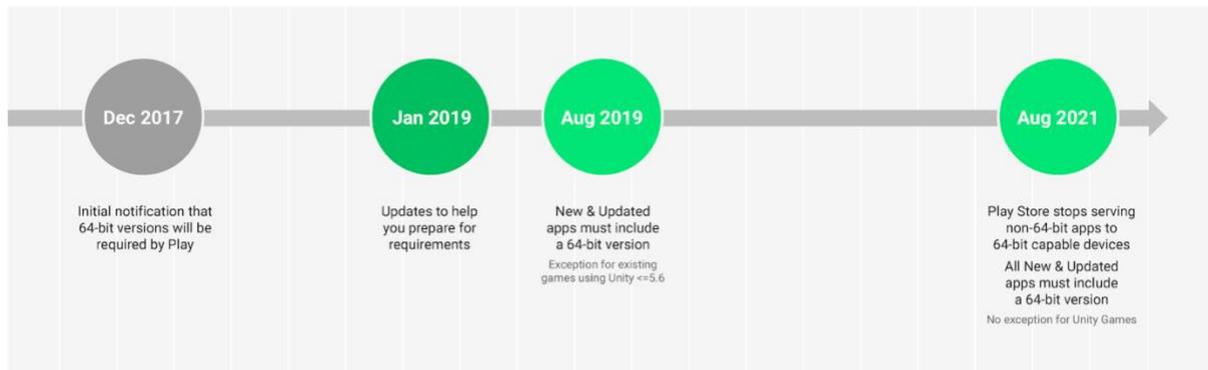
<https://play.google.com/apps/publish>

IMPORTANTE:

Para desplegar tu aplicación en la Google PlayStore, debes de realizar un único pago de 30 dólares (precio puede variar), para poder desplegar tus aplicaciones en la Google PlayStore.

10) Generar versión de 64bits:

64-bit requirement timeline for Google Play



Please note this requirement does not apply to:

- APKs or app bundles explicitly targeting Wear OS or Android TV.
- APKs or app bundles that are not distributed to devices running Android 9 Pie or later.

Referencia:

<https://developer.android.com/distribute/best-practices/develop/64-bit>

Nota:

Se puede abrir el APK, y dentro de la carpeta lib, buscar estos archivos, si se tienen ambos, la app funciona con arm64 y no tenemos que hacer nada.

armeabi-v7a

arm64-v8a

Abrir el archivo ./Android/app/build.gradle

Hay que añadir que compile para dispositivos de 32 bits y 64 bits, para que nuestra app funcione correctamente en dichos dispositivos.

Hay que insertar estas líneas después de la referencia a flutter (ver siguiente página)

```

flutter {
  source '../..'
}

afterEvaluate {
  mergeReleaseJniLibFolders.doLast {
    def archTypes = ["arm-release", "arm64-release"]
    archTypes.forEach { item ->
      copy {
        from zipTree("$flutterRoot/bin/cache/artifacts/engine/android-$item/flutter.jar")
        include 'lib/*/libflutter.so'
        into "$buildDir/intermediates/jniLibs/release/"
        eachFile {
          it.path = it.path.replaceFirst("lib/", "")
        }
      }
    }
  }
}
}
}

```

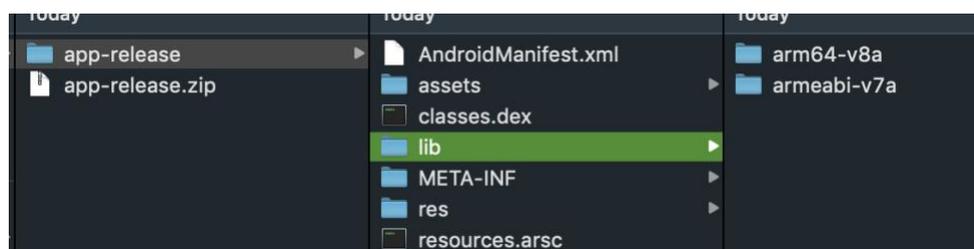
Ver el archivo completo en mi Gist:

<https://gist.github.com/Klerith/53147980b8f6dc3ef5bbf3ef6b7472f6>

Luego de realizar los cambios, volver a realizar el release

```
flutter build apk --release
```

debería de tener tanto el arm64 como el armeabi-v7a, esto quiere decir que funcionará para 32 y 64 bits nuestra aplicación.



Desplegar Aplicaciones

IOS

Documentación oficial:

<https://flutter.dev/docs/deployment/ios>

IMPORTANTE:

Para desplegar aplicaciones en la Apple AppStore, es necesario pagar 100 dólares ANUALES, a diferencia de Google PlayStore que el servicio es de 30 dólares de por vida, Apple es una membresía de 100 dólares

Diferentes membresías de Apple aquí:

<https://developer.apple.com/support/compare-memberships/>

Enrolarse al programa de developer de Apple

<https://developer.apple.com/programs/>

Paso 1: Tener la membresía. Sin la membresía no se puede continuar, es necesario también tener una computadora MAC, no es opcional.

Paso 2: Ir a la página de developer de Apple, para generar un AppID

<https://developer.apple.com/account/ios/identifier/bundle>

Ref: <https://developer.apple.com/account/ios/identifier/bundle>

Paso 3: Crear la aplicación en AppStore connect

NOTA:

Al crear la aplicación en AppStore Connect, puede demorar hasta una hora en parecer, usualmente son 10 minutos

Ref: <https://flutter.dev/docs/deployment/ios#create-an-application-record-on-app-store-connect>

Ejemplo de llenado de la pantalla

Nota:

El Nombre de la aplicación debe de ser único

New App

Platforms ?
 iOS tvOS

Name ?
Flutter Peliculas DEV

Primary Language ?
Spanish (Mexico) ▾

Bundle ID ?
Flutter Peliculas DEV - com.fernandoherrera.pelisflutter-dev ▾

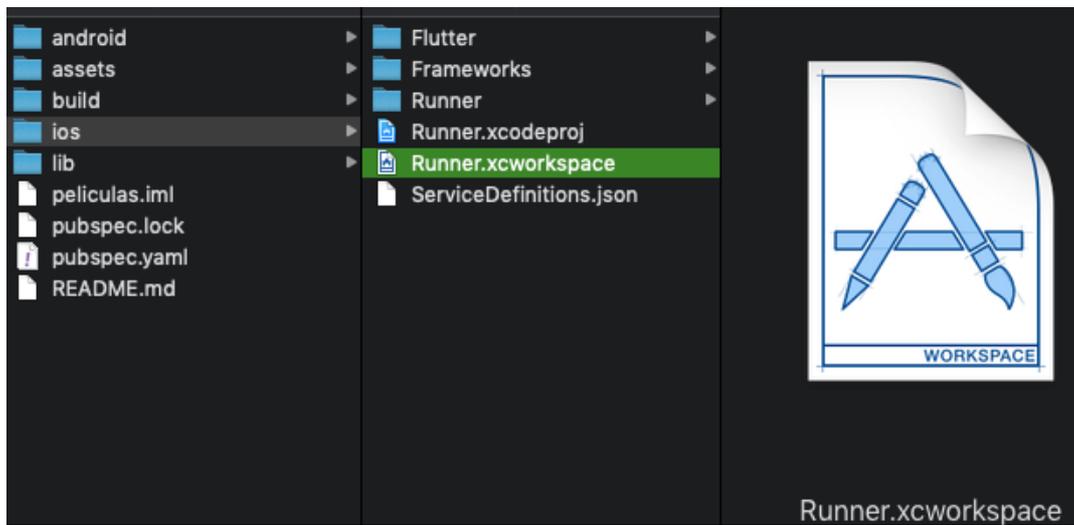
SKU ?
udemy-curso-flutter-dev

User Access ?
 Limited Access Full Access

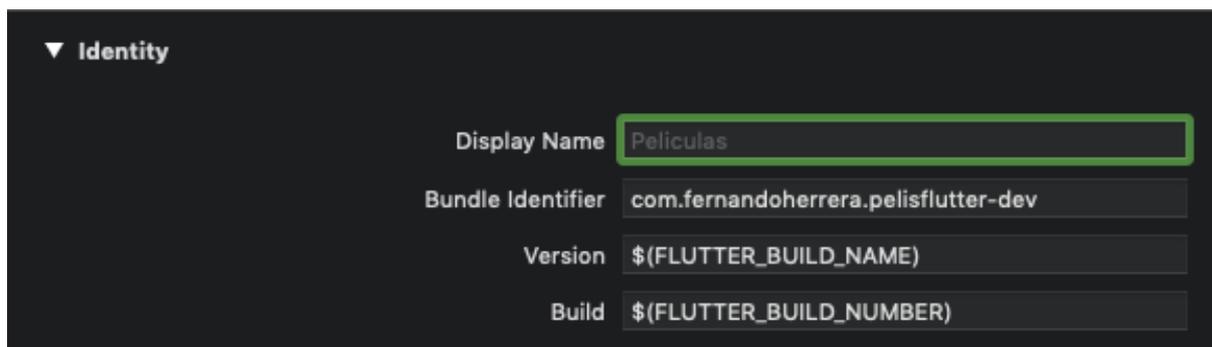
Cancel **Create**

Paso 4: XCODE

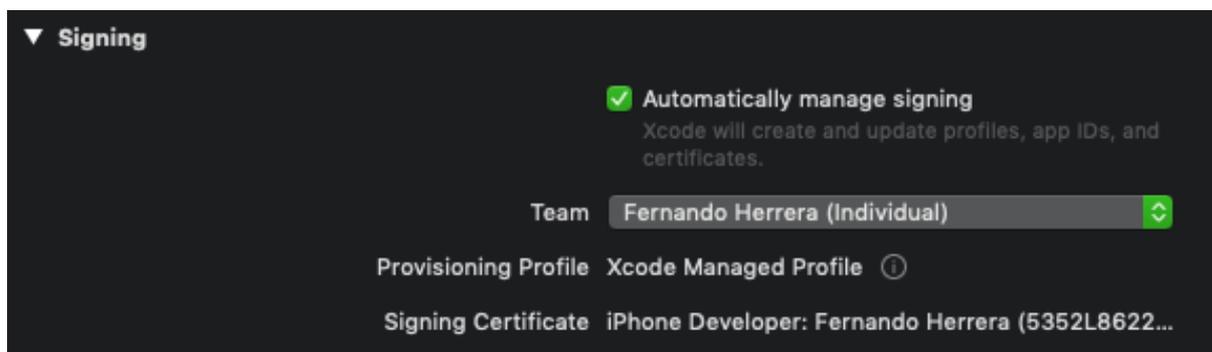
Abrir el proyecto, dentro de la carpeta IOS, buscar el archivo Runner.xcworkspace



Paso 5: cambiar el bundle identifier, por el que creamos en la página de developer de Apple



Paso 6: Para el firmado, seleccionar forma automática y seleccionar tu cuenta de developer de Apple



Paso 7: Crear el build archive (El APK firmado por decirlo así), este es el archivo que vamos a subir a la Apple AppStore

flutter build ios --release

Nota:

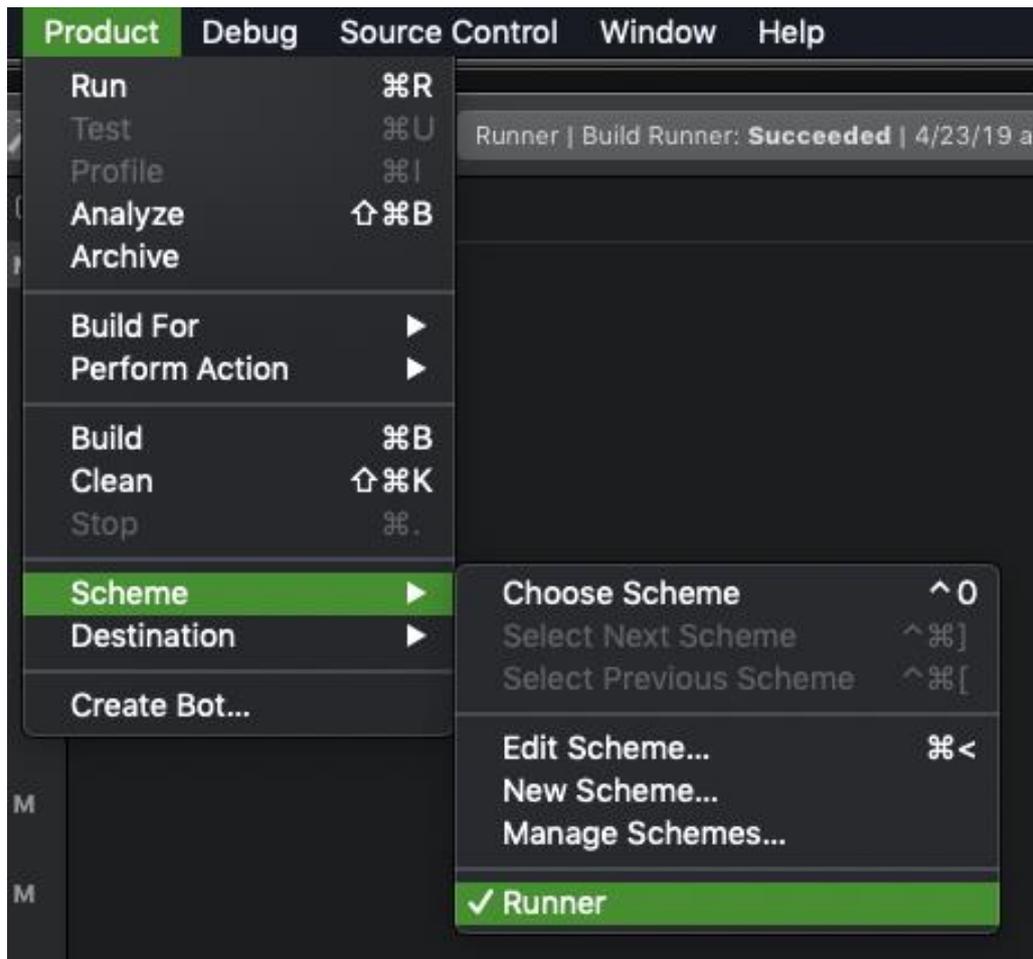
Este proceso puede demorar dependiendo de tu computadora

```
Automatically signing iOS for device deployment using specified development team in Xcode project: RVKJA4NFPL
Running Xcode build...
  └─Building Dart code...                39.2s
  └─Generating dSYM file...              1.1s
  └─Stripping debug symbols...           0.6s
  └─Assembling Flutter resources...      1.2s
  └─Compiling, linking and signing...    8.5s
Xcode build done.                        52.2s
Built /Users/strider/Desktop/dev/apps/peliculas/build/ios/iphoneos/Runner.app.
```

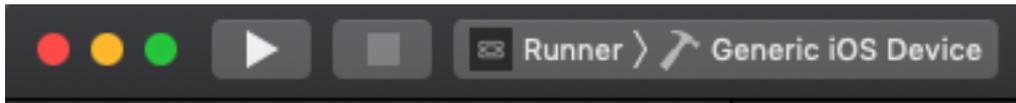
Paso 8: Reiniciar XCODE. Es sugerido reiniciar XCode después del build, para que tome todos los nuevos cambios hechos en el BUILD, y luego re abrir el proyecto en XCODE.

Paso 9: Configuraciones OBLIGATORIAS antes del build

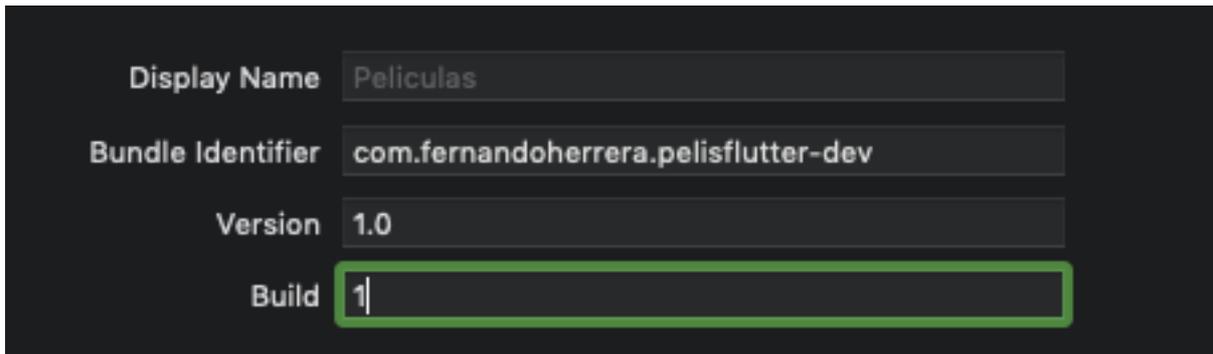
Product > Scheme > Runner. Y también asegúrate de tener seleccionado el dispositivo genérico: (ver imagen 2)



Dispositivo genérico, esta opción DEBE de estar seleccionada



Paso 10: Establecer la versión de tu aplicación, recuerden esta versión tiene que ser diferente en cada subida a la AppStore Connect

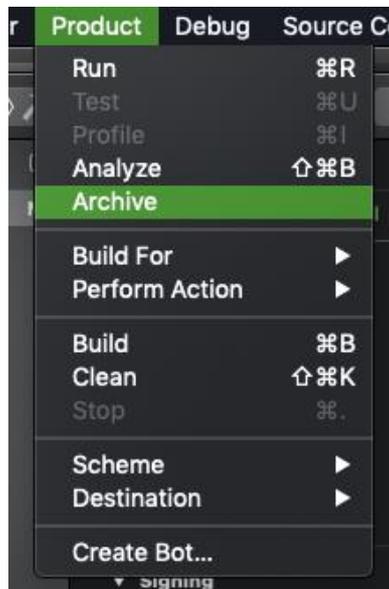


Nota:

La versión es la que lo usuarios verán

El Build es el numero entero que DEBE ser autoincrementado con cada despliegue.

Paso 11: Generar el Archive:

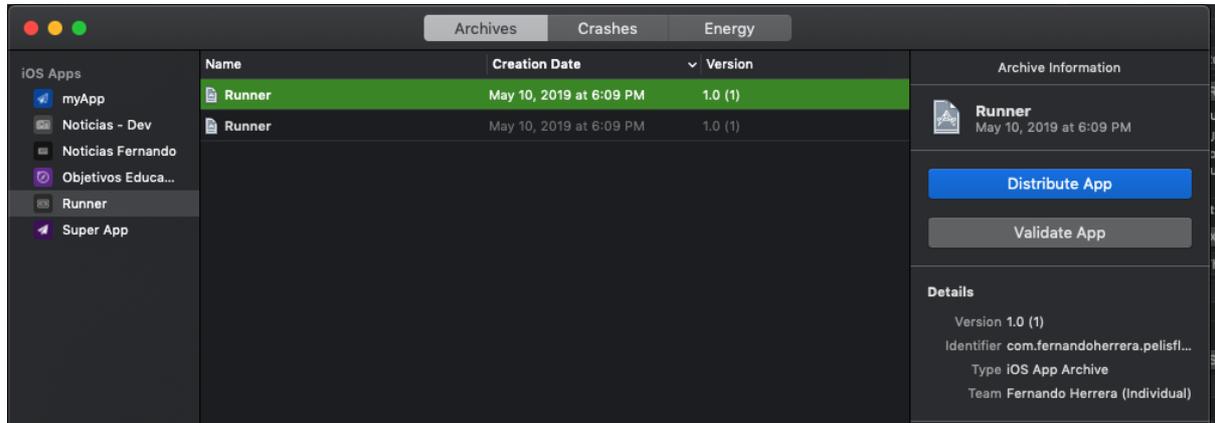


Nota:

Es posible que pida ingresar tu cuenta de developer de Apple y password

Paso 12: Subir la aplicación

El paso anterior, nos deja en esta pantalla:



Presionen el botón azul para distribuir la aplicación (subir la app a AppStore Connect)

Nota:

El botón gris de Validar la App, pueden hacerlo si quieren confirmar que todo este correcto antes de subirlo.

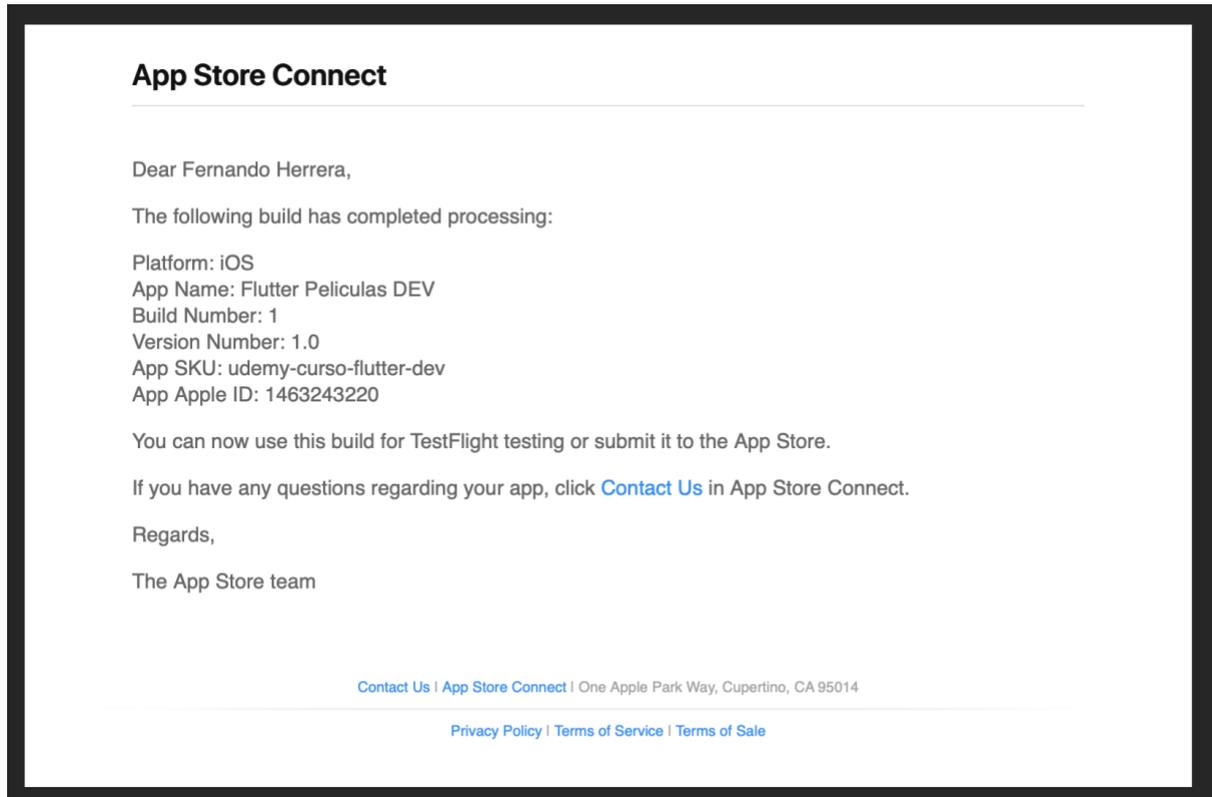
Nota:

Este error se me presentó cuando hice la guía, lo resolví generando nuevamente los íconos y realice nuevamente el build... luego de eso me funcionó.

ERROR ITMS-90717: "Invalid App Store Icon. The App Store Icon in the asset catalog in 'Runner.app' can't be transparent nor contain an alpha channel."

Paso 13: Invitar a Fernando a un café!

Paso 14: Hay que esperar a que se procese la APP en la AppStore Connect, serán notificados mediante correo electrónico, o bien el proceso usualmente demora menos de 10 minutos, el correo luce algo así:



Felicidades, su aplicación esta en la Apple AppStore, ahora tienen que llenar la ficha en AppStore connect y enviarla a revisión!