🚀 Guía de Despliegue - Napphy Services

Esta guía te ayudará a preparar y desplegar la aplicación Napphy Services en Android e iOS.

📋 Tabla de Contenidos

Preparación Pre-Despliegue

1. Checklist General

- [ ] Todas las funcionalidades probadas
- [ ] Firebase configurado en modo producción
- [ ] Reglas de seguridad de Firebase actualizadas
- [ ] API keys configuradas
- [ ] Certificados generados
- [ ] Íconos y splash screens configurados
- [ ] Versión y build number actualizados
- [ ] Documentación completa

2. Actualizar Versión

pubspec.yaml:

```
version: 1.0.0+1
# Formato: versión+buildNumber
# Ejemplo: 1.0.0+1, 1.0.1+2, 1.1.0+3
```

3. Configurar Variables de Producción

Asegúrate de que Firebase esté configurado en modo producción:

- Firestore: Cambia reglas de “test mode” a reglas de producción
- Storage: Verifica límites de tamaño y tipo de archivo
- Authentication: Habilita proveedores necesarios
- Messaging: Verifica configuración de notificaciones

Despliegue Android

Paso 1: Configurar Firma de la App

1.1 Generar Keystore

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

Importante: Guarda la contraseña en un lugar seguro.

1.2 Configurar key.properties

Crea el archivo `android/key.properties`:

```
storePassword=TU_STORE_PASSWORD
keyPassword=TU_KEY_PASSWORD
keyAlias=napphy-key
storeFile=/ruta/a/napphy-keystore.jks
```

⚠️ **NUNCA** subas este archivo a Git. Agrégalo a `.gitignore`.

1.3 Configurar build.gradle

android/app/build.gradle:

```
def keystoreProperties = new Properties()
def keystorePropertiesFile = rootProject.file('key.properties')
if (keystorePropertiesFile.exists()) {
    keystoreProperties.load(new FileInputStream(keystorePropertiesFile))
}

android {
    ...

    defaultConfig {
        applicationId "com.napphy.services"
        minSdkVersion 21
        targetSdkVersion 33
        versionCode 1
        versionName "1.0.0"
    }

    signingConfigs {
        release {
            keyAlias keystoreProperties['keyAlias']
            keyPassword keystoreProperties['keyPassword']
            storeFile keystoreProperties['storeFile'] ? file(keystoreProperties['store
File']) : null
            storePassword keystoreProperties['storePassword']
        }
    }

    buildTypes {
        release {
            signingConfig signingConfigs.release
            minifyEnabled true
            shrinkResources true
            proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'pr
oguard-rules.pro'
        }
    }
}
```

Paso 2: Configurar ProGuard (Ofuscación)

android/app/proguard-rules.pro:

```
# Flutter
-keep class io.flutter.app.** { *; }
-keep class io.flutter.plugin.** { *; }
-keep class io.flutter.util.** { *; }
-keep class io.flutter.view.** { *; }
-keep class io.flutter.** { *; }
-keep class io.flutter.plugins.** { *; }

# Firebase
-keep class com.google.firebase.** { *; }
-keep class com.google.android.gms.** { *; }

# Modelo de datos
-keep class com.napphy.services.models.** { *; }
```

Paso 3: Configurar Íconos y Splash Screen

3.1 Ícono de la App

Coloca el ícono en:
- `android/app/src/main/res/mipmap-hdpi/ic_launcher.png` (72x72)
- `android/app/src/main/res/mipmap-mdpi/ic_launcher.png` (48x48)
- `android/app/src/main/res/mipmap-xhdpi/ic_launcher.png` (96x96)
- `android/app/src/main/res/mipmap-xxhdpi/ic_launcher.png` (144x144)
- `android/app/src/main/res/mipmap-xxxhdpi/ic_launcher.png` (192x192)

O usa el paquete `flutter_launcher_icons`:

```yaml
# pubspec.yaml
dev_dependencies:
  flutter_launcher_icons: ^0.13.1

flutter_launcher_icons:
  android: true
  ios: true
  image_path: "assets/icon/app_icon.png"
```

Ejecuta:

```
flutter pub run flutter_launcher_icons
```

Paso 4: Construir APK/App Bundle

APK (para distribución directa o pruebas)

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

Output: `build/app/outputs/flutter-apk/app-release.apk`

App Bundle (para Google Play Store)

```
flutter build appbundle --release
```

Output: `build/app/outputs/bundle/release/app-release.aab`

Paso 5: Publicar en Google Play Store

1. **Crear cuenta de desarrollador**
   - Ve a Google Play Console (https://play.google.com/console)
   - Paga la tarifa única de $25

2. **Crear una nueva aplicación**
   - Click en “Crear aplicación”
   - Nombre: “Napphy Services”
   - Idioma: Español
   - Tipo: App

3. **Completar información de la tienda**
   - Descripción corta (80 caracteres)

- Descripción completa (4000 caracteres)
- Screenshots (mínimo 2)
- Ícono de la app (512x512)
- Gráfico destacado (1024x500)
- Categoría: “Crianza” o “Estilo de vida”
- Clasificación de contenido

4. **Subir el App Bundle**
   - Ve a “Producción” → “Crear nueva versión”
   - Sube el archivo `.aab`
   - Completa notas de la versión
   - Revisa y publica

5. **Esperar revisión**
   - Google revisa en 1-3 días
   - Recibirás notificación por email

Despliegue iOS

Paso 1: Configuración en Xcode

1.1 Abrir proyecto en Xcode

```
open ios/Runner.xcworkspace
```

1.2 Configurar Signing

1. Selecciona el target “Runner”
2. Ve a “Signing & Capabilities”
3. Selecciona tu Team (Apple Developer Account)
4. Bundle Identifier: `com.napphy.services`
5. Automáticamente se generará el Provisioning Profile

Paso 2: Configurar Capabilities

Habilita las siguientes capabilities:

1. **Push Notifications**
2. **Background Modes**
   - Remote notifications
   - Background fetch
3. **Sign in with Apple** (si aplica)

Paso 3: Configurar Íconos y Assets

3.1 App Icon

1. En Xcode, abre `Runner/Assets.xcassets/AppIcon.appiconset`
2. Arrastra las imágenes del ícono en los tamaños correctos:
   - 20pt (40x40, 60x60)
   - 29pt (58x58, 87x87)
   - 40pt (80x80, 120x120)

- 60pt (120x120, 180x180)
- 1024pt (1024x1024)

3.2 Launch Screen

Configura el splash screen en `Runner/Assets.xcassets/LaunchImage.imageset`

Paso 4: Configurar Info.plist

ios/Runner/Info.plist:

```
<key>CFBundleDisplayName</key>
<string>Napphy Services</string>

<key>CFBundleShortVersionString</key>
<string>1.0.0</string>

<key>CFBundleVersion</key>
<string>1</string>

<!-- Permisos -->
<key>NSLocationWhenInUseUsageDescription</key>
<string>Necesitamos tu ubicación para encontrar niñeras cercanas</string>

<key>NSCameraUsageDescription</key>
<string>Necesitamos acceso a la cámara para tu foto de perfil</string>

<key>NSPhotoLibraryUsageDescription</key>
<string>Necesitamos acceso a tus fotos para tu foto de perfil</string>
```

Paso 5: Construir IPA

Opción 1: Desde Flutter

```
flutter build ios --release
```

Opción 2: Desde Xcode

1. En Xcode, selecciona “Any iOS Device (arm64)”
2. Product → Archive
3. Espera a que termine el archivado
4. Se abrirá el Organizer

Paso 6: Publicar en App Store

1. **Crear cuenta de Apple Developer**
   - Ve a Apple Developer (https://developer.apple.com/)
   - Paga $99/año

2. **Crear App ID**
   - Ve a Certificates, Identifiers & Profiles
   - Crea un nuevo App ID: `com.napphy.services`
   - Habilita capabilities necesarias

3. **Crear App en App Store Connect**
   - Ve a App Store Connect (https://appstoreconnect.apple.com/)
   - Click en “My Apps” → “+”
   - Nombre: “Napphy Services”

   - Bundle ID: `com.napphy.services`
   - SKU: Un código único

4. **Completar información de la tienda**
   - Descripción (hasta 4000 caracteres)
   - Palabras clave
   - Screenshots (6.5”, 5.5”)
   - Ícono de la app (1024x1024)
   - Categoría: “Lifestyle” o “Productivity”
   - Clasificación de contenido

5. **Subir el build desde Xcode**
   - En el Organizer, selecciona el archive
   - Click en “Distribute App”
   - Selecciona “App Store Connect”
   - Selecciona “Upload”
   - Espera a que se procese (5-15 minutos)

6. **Enviar para revisión**
   - En App Store Connect, selecciona el build
   - Completa toda la información requerida
   - Click en “Submit for Review”

7. **Esperar aprobación**
   - Apple revisa en 24-48 horas
   - Recibirás notificación por email

Variables de Entorno

Producción vs Desarrollo

Para mantener configuraciones separadas, considera usar flavors:

Android Flavors

android/app/build.gradle:

```
android {
    ...
    flavorDimensions "environment"
    productFlavors {
        dev {
            dimension "environment"
            applicationIdSuffix ".dev"
            versionNameSuffix "-dev"
        }
        prod {
            dimension "environment"
        }
    }
}
```

iOS Schemes

En Xcode, duplica el scheme “Runner” y crea:
- Runner-Dev
- Runner-Prod

Archivo de Configuración

lib/config/environment.dart:

```
class Environment {
  static const bool isProduction = bool.fromEnvironment('PRODUCTION', defaultValue: fa
lse);

  static String get apiUrl => isProduction
    ? 'https://api.napphy.com'
    : 'https://dev-api.napphy.com';

  static String get firebaseProject => isProduction
    ? 'napphy-prod'
    : 'napphy-dev';
}
```

Ejecutar con:

```
flutter run --dart-define=PRODUCTION=true
```

Testing en Producción

1. Internal Testing (Beta)

Android - Internal Testing

1. En Google Play Console → “Pruebas internas”
2. Sube un App Bundle
3. Agrega testers por email
4. Comparte el link de prueba

iOS - TestFlight

1. En App Store Connect → “TestFlight”
2. El build aparecerá automáticamente después de subir
3. Agrega “Internal Testers” (tu equipo)
4. O agrega “External Testers” (usuarios beta)
5. Comparte el link de TestFlight

2. Staged Rollout

Android

1. En Google Play Console → “Producción”
2. Al publicar, selecciona “Rollout escalonado”
3. Comienza con 10%, luego 25%, 50%, 100%

iOS

1. En App Store Connect → “App Store”
2. Selecciona “Phased Release”
3. Se distribuirá gradualmente durante 7 días

Monitoreo y Analytics

1. Firebase Analytics

Ya configurado. Monitorea:
- Número de usuarios activos
- Retención de usuarios
- Eventos personalizados
- Flujos de usuario

2. Firebase Crashlytics

Para reportes de crashes, agrega:

```
# pubspec.yaml
dependencies:
  firebase_crashlytics: ^3.4.8
```

main.dart:

```
import 'package:firebase_crashlytics/firebase_crashlytics.dart';

void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp();

  FlutterError.onError = FirebaseCrashlytics.instance.recordFlutterFatalError;

  runApp(MyApp());
}
```

3. Firebase Performance Monitoring

```
dependencies:
  firebase_performance: ^0.9.3+8
```

4. Remote Config

Para cambiar configuración sin actualizar la app:

```
dependencies:
  firebase_remote_config: ^4.3.8
```

Checklist Final de Despliegue

Antes de Publicar

- [ ] Todas las funcionalidades funcionan correctamente
- [ ] Tests pasan (unitarios, integración, UI)
- [ ] No hay console.log() o print() innecesarios
- [ ] Íconos y splash screens configurados
- [ ] Screenshots de la tienda tomados
- [ ] Descripción y metadata completados
- [ ] Política de privacidad publicada (URL)
- [ ] Términos y condiciones publicados (URL)
- [ ] Firebase en modo producción
- [ ] Reglas de seguridad actualizadas
- [ ] Certificados y keys generados
- [ ] Version y build number actualizados

Después de Publicar

- [ ] Monitorear crashes en Crashlytics
- [ ] Monitorear analytics
- [ ] Responder reviews de usuarios
- [ ] Preparar actualizaciones basadas en feedback
- [ ] Mantener Firebase y dependencias actualizadas

Actualizaciones Futuras

Versionado Semántico

- **MAJOR** (1.0.0): Cambios incompatibles
- **MINOR** (0.1.0): Nueva funcionalidad compatible
- **PATCH** (0.0.1): Correcciones de bugs

Proceso de Actualización

1. Actualizar versión en `pubspec.yaml`
2. Actualizar CHANGELOG.md
3. Construir nueva versión
4. Probar en beta
5. Publicar en producción
6. Crear tag en Git: `git tag v1.0.1`

Recursos Útiles

- Flutter Deployment (https://docs.flutter.dev/deployment)
- Google Play Console (https://play.google.com/console)
- App Store Connect (https://appstoreconnect.apple.com/)

- Firebase Console (https://console.firebase.google.com/)

¡Tu app está lista para conquistar el mundo! 🚀