

CURSO DESARROLLO DE APLICACIONES MÓVILES ANDROID TRAINEE

Módulo 4.

Desarrollo de aplicaciones móviles Android Kotlin



Temario

Unidad 4

a) Release

b) Principios básicos de diseño Orientado a Objetos



Gradle

Gradle es una herramienta de automatización de construcción (Build automation) diseñada para ser flexible en la construcción de cualquier tipo de software.

Gradle favorece el uso de convenciones por sobre configuraciones y generalmente entrega valores por defecto para ajustes y propiedades

Características de Gradle

Sus características principales tiene relación principalmente con la velocidad y flexibilidad en la construcción. Además, otras características son:

- Alto rendimiento: Gradle evita el trabajo innecesario ejecutando sólo las tareas necesarias que han cambiado su entrada o salida. Además se puede activar un caché para la construcción utilizando la salida de construcciones previas y ejecuta tareas en paralelo.

Características de Gradle

- Corre sobre JVM: Gradle se ejecuta sobre la JVM y se debe tener instalado un JDK para que funcione. No tiene dependencias externas, por lo que cualquier proyecto Java o Android(incluso con Kotlin) puede ocupar Gradle. Además es fácil de correr en multiplataforma.

Características de Gradle

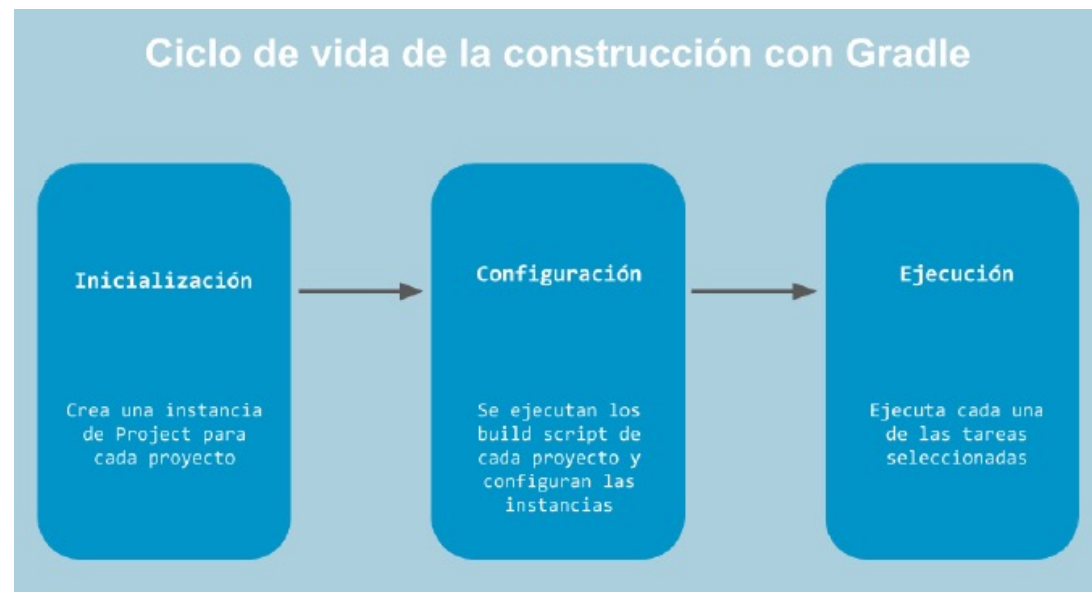
- Convenciones: Gradle toma parte de la filosofía de Maven y hacer que los tipos comunes de proyectos, como los proyectos Java, sean fáciles de construir mediante la implementación de convenciones. Se puede aplicar los complementos apropiados y tener instrucciones de compilación acotadas para muchos proyectos.

Características de Gradle

- Extensibilidad: Más aún, las tareas comunes definidas por convención pueden ser extendidas y personalizadas entregando más flexibilidad a la construcción.
- Soporte para IDE: Varios IDEs permiten importar los archivos Gradle para compilar y permiten interactuar con ellos de manera sencilla. Entre estos, se encuentran Android Studio, IntelliJ IDEA, Eclipse, NetBeans y Visual Studio.

Ciclo de vida de la construcción

La construcción usando Gradle está dividida en 3 etapas en las cuales se inicializa, configura y ejecuta tareas específicas.



Inicialización

Gradle soporta construcciones para uno o varios proyectos. Durante la fase de inicialización determina qué proyectos van a tomar parte de la construcción y crea una instancia de la clase Project para cada uno de estos proyectos

Archivo de configuración

Además del script de construcción, Gradle utiliza un archivo de configuración, que tiene el nombre por default de `s`. Este archivo indica si es una construcción de un proyecto único o se conforma por varios proyectos, y además indica cuales son los proyectos que componen la construcción.

En el caso de un proyecto único, `settings.gradle` solo hace referencia al nombre de un proyecto

```
include 'app'
```

Archivo de configuración

En el caso de multi-proyecto, el archivo indica cada uno de ellos

```
include 'project1', 'project2:child', 'project3:child1'
```

Dependiendo de las funcionalidades y la estructura del código se puede necesitar más de un proyecto, por ejemplo para agrupar código para un módulo específico que interactúe con código común dentro de la app.

Configuración y ejecución

Para un proyecto único, el flujo después de la fase de inicialización es simple. El script de construcción es ejecutado contra la instancia de Project previamente creada. Luego Gradle busca por las tareas con nombre igual a los pasados como argumento. Si las tareas con esos nombres existe, entonces son ejecutadas en forma independiente en mismo orden entregado.

Android Gradle plugin

El sistema de construcción que utiliza Android Studio está basado en Gradle el Android Gradle Plugin entrega funcionalidades que son específicas para la construcción de Android. Tanto el plugin como Gradle pueden correr en forma independiente de Android Studio

Archivo build.gradle

El archivo build.gradle es el utilizado para compilar y empaquetar los proyectos, que es el nombre asignado por convención.

Android Gradle plugin

En un proyecto Android hay 2 archivos build.gradle que son creados por defecto para cada nuevo proyecto.

1. El build.gradle de la raíz del proyecto es para agregar las configuraciones comunes a todos los módulos o subproyectos.
2. El build.gradle dentro de cada subproyecto o módulo, que tiene la configuración particular.

Tipos de Compilacion (Build Types)

Cada variante de compilación representa una versión diferente de la app que se puede compilar y surgen del uso de conjuntos de reglas específicas de Gradle que combinan configuración, códigos y recursos configurados.

El archivo build.gradle del módulo contiene las variantes de compilación. Al crear un nuevo proyecto, el plugin de Gradle crea automáticamente los tipos de compilación para depuración o lanzamiento. Para depuración crea una clave de depuración que permite depurar la app en dispositivos.

Tipos de Compilación (Build Types)

```
android {  
    compileSdkVersion 29  
    defaultConfig { ... }  
    buildTypes {  
        release {  
            minifyEnabled false  
            proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'  
        }  
    }  
}
```


Tipos de productos

La compilación de tipos de productos es similar a las variantes de compilación y representa varias versiones de la misma app, como por ejemplo, las versiones pagadas y gratuitas. Se puede personalizar los tipos de producto para utilizar código y recursos diferentes y al mismo tiempo reutilizar partes comunes a las distintas versiones. Los tipos de productos son opcionales y deben ser definidos en el archivo build.gradle

Tipos de productos

Un build.gradle que tenga dos tipos de producto: una versión demo y una versión full es de la siguiente manera

```
android {  
    ...  
    defaultConfig {...}  
    buildTypes {  
        debug{...}  
        release{...}  
    }  
  
    flavorDimensions "version"  
    productFlavors {  
        demo {  
            dimension "version"  
            applicationIdSuffix ".demo"  
            versionNameSuffix "-demo"  
        }  
    }  
}
```

```
}  
full {  
    dimension "version"  
    applicationIdSuffix ".full"  
    versionNameSuffix "-full"  
}  
}  
}
```

Tipos de productos

flavorDimensions permite combinar configuraciones de varios tipos de producto, por ejemplo, para crear productos del tipo demo y full mezcladas con niveles de API mínima segmentadas. Se define una o más dimensiones y se indica a qué dimensión corresponde cada flavor.

```
android {  
    ...  
    buildTypes {  
        debug {...}  
        release {...}  
    }  
}
```

```
flavorDimensions "api", "mode"  
productFlavors {  
    demo {  
        dimension "mode"  
        applicationIdSuffix ".demo"  
        versionNameSuffix "-demo"  
    }  
}
```

```
full {  
    dimension "mode"  
    applicationIdSuffix ".full"  
    versionNameSuffix "-full"  
}
```

```
minApi24 {  
    dimension "api"  
    minSdkVersion 24  
    versionNameSuffix "-minApi24"  
    ...  
}
```

```
minApi21 {  
    dimension "api"  
    minSdkVersion 21  
    versionNameSuffix "-minApi21"  
    ...  
}  
...  
}
```

Variables de compilación

Una variable de compilación es el producto cruzado entre los tipos de compilación y tipo de producto, y es la configuración que se utiliza para compilar el APK. Dado lo anterior, las variables de compilación no son configuradas directamente si no que mediante la configuración de los tipos de compilación y producto.

Dependencias

El sistema de compilación administra las dependencias locales y remotas en forma automática. Basta con definir las dependencias en el build.gradle para que sean sincronizadas desde los repositorios indicados. Las dependencias son agregadas al archivo build.gradle como una tarea independiente usando dependencies

Dependencias

```
apply plugin: 'com.android.application'

android { ... }
dependencies {
    // Dependencia a módulo local llamado mylibrary
    implementation project(":mylibrary")

    // Dependencia de binarios alojados localmente en el directorio libs
    implementation fileTree(dir: 'libs', include: ['*.jar'])

    // Dependencia a un binario remoto
    implementation 'cl.desafiolatam.android:cool-app:4.2'
}
```

Firmas

El sistema de compilación permite definir configuraciones de firma para firmar los APK automáticamente durante el proceso de compilación. Para la versión de depuración, el proceso de compilación firma el APK con una clave y un certificado predeterminados, evitando la solicitud de contraseña durante la compilación. Para la versión de lanzamiento, se debe definir explícitamente una configuración de firma y credenciales.

ProGuard

Durante la compilación se puede especificar un conjunto de reglas ProGuard diferentes para cada variable de compilación. Proguard es una herramienta de línea de comandos que encoge, optimiza y ofusca código

Con cada compilación, ProGuard crea los siguientes archivos:

- dump.txt: Describe la estructura interna de todos los archivos de clase del APK.

ProGuard

- mapping.txt: Proporciona una traducción entre la clase original y la oculta, el método y los nombres de campos. Se sobrescribe cada vez que se construye el proyecto.
- seeds.txt: Indica las clases y los miembros que no se ocultaron.
- usage.txt: Indica el código que se quitó del APK.

Estos archivos se guardan en:

```
<module-name>/build/outputs/mapping/release/
```

ProGuard

- mapping.txt: Proporciona una traducción entre la clase original y la oculta, el método y los nombres de campos. Se sobrescribe cada vez que se construye el proyecto.
- seeds.txt: Indica las clases y los miembros que no se ocultaron.
- usage.txt: Indica el código que se quitó del APK.

Estos archivos se guardan en:

```
<module-name>/build/outputs/mapping/release/
```

Compatibilidad con varios APK

El sistema de compilación permite compilar automáticamente diferentes APK que contengan solo el código y los recursos necesarios para una determinada densidad de pantalla, personalizando el instalador y disminuyendo su tamaño

Proceso de compilación

En el proceso de compilación intervienen herramientas y procesos que convierten el proyecto en un APK.

Los pasos que sigue el proceso son los siguientes:

1. Los compiladores convierten:

- El código fuente en archivos DEX que incluyen el código en bytes que se ejecuta en el
- Dispositivo Los recursos en recursos compilados

2. El empaquetador combina los archivos DEX y los recursos compilados en un solo APK

Proceso de compilación

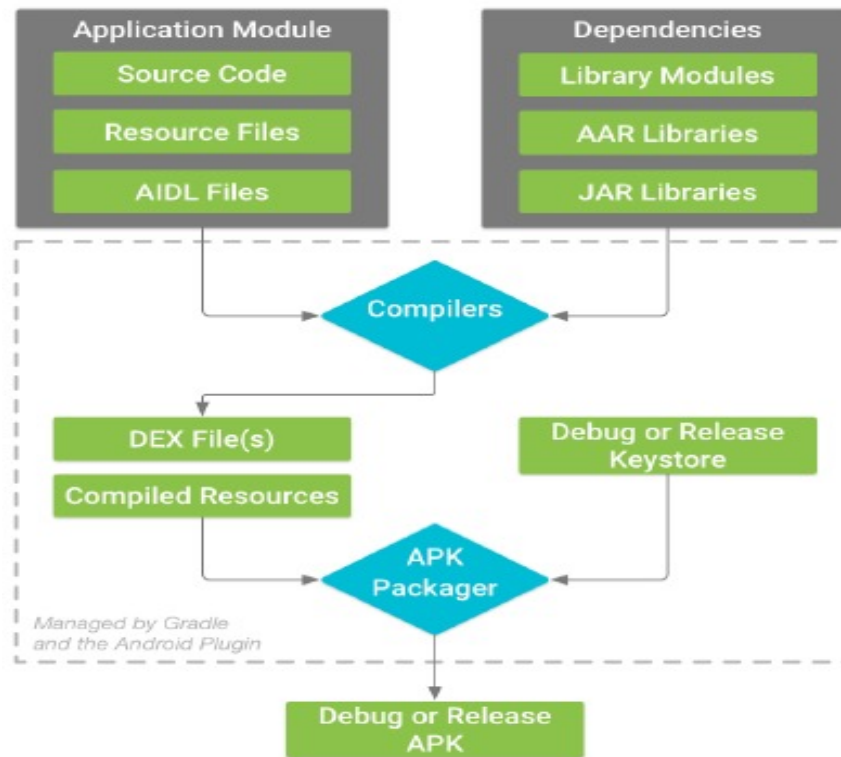
3. Para que este APK pueda ser instalado y utilizado debe ser previamente firmado para 1 de estos propósitos:

- Versión de depuración, para hacer pruebas
- Versión de producción o lanzamiento

4. Antes de generar el APK final, el empaquetador usa zipalign para optimizar la utilización de memoria al ejecutarse en el dispositivo

El resultado de este proceso es un APK de depuración o producción listo para ser instalado en los dispositivos.

Proceso de compilación



Preparando la publicación de la App

Luego de diseñar, codificar y probar una app se llega al paso final que es ponerla a disposición de los usuarios. Para publicar la app se siguen 2 pasos:

1. Preparar la app de lanzamiento, compilando una versión de lanzamiento que los usuarios puedan descargar e instalar en sus dispositivos Android.
2. Realizar el lanzamiento de la app para los usuarios, donde se publicita, vende y distribuye la versión de lanzamiento para los usuarios.

Preparando la publicación de la App



Preparando la publicación de la App

De las opciones disponibles para el lanzamiento de la app, utilizar la tienda oficial de Android es la más utilizada, están disponibles Amazon Appstore y APPGallery de Huawei entre otras como tiendas alternativas (y complementarias) para hacer el lanzamiento masivo a los usuarios. Si se quiere entregar a un grupo de usuarios más reducido es posible también hacer el lanzamiento compartiendo directamente el APK.

Preparación de la App de lanzamiento

De En esta etapa se construye y prueba la app antes de hacer el lanzamiento.



Reunir materiales y recursos

La app utiliza varios elementos secundarios y cómo mínimo se debe proporcionar la clave criptográfica y el ícono de la app.

Clave Criptográfica

Cada app para lanzamiento debe ser firmada digitalmente con un certificado que pertenece al desarrollador de la app, manejado con una clave privada. Luego, el sistema puede identificar el autor de una app y establecer sistemas de confianza con otras apps.

Iconos de la App

Cada app debe contar un ícono de aplicación y que cumpla con las pautas definidas para íconos. El ícono ayuda a los usuarios a identificar la app en la pantalla de inicio, en el launcher de aplicaciones o en otras secciones. Además, los servicios de publicación muestran ese ícono en sus búsquedas

Licencia

Es una buena práctica incluir un contrato de licencia para el usuario final (EULA) puede ayudar a proteger al desarrollador, organización y propiedad intelectual. Algunos ejemplos de estas políticas de privacidad las podemos ver en Facebook o Instagram o podemos usar alguna política estándar.

Configurar reléase App

Hay ciertas recomendaciones, que si bien son opcionales optimizan la app final

Elegir un buen nombre de paquete

El nombre del paquete es usado como identificador en la app y se usa para toda la vida útil de la app sin poder cambiarlo, por lo que es importante elegir un buen nombre de paquete

Este nombre es definido en el manifiesto usando el atributo package

Configurar reléase App

Limpiar el proyecto

Eliminar todos los archivos que no sean utilizados por la app en los directorios res, lib, assets para empaquetar solamente lo necesario por la app. El proyecto puede ser analizado por Android Studio usando Analyze > Inspect Code, que indica posibles mejoras al código, entre esas, encontrar los archivos no referenciados

Configurar reléase App

Actualizar el manifiesto

Se debe comprobar que la configuración del manifiesto y los elementos de compilación sean los correctos.

Elemento

La aplicación debe declarar exclusivamente los permisos mínimos necesarios para su funcionamiento, evitando incluir permisos que no serán utilizados.

Configurar reléase App

Atributos android:icon y android:label

Estos valores se encuentran en el elemento <application> y son utilizados en la app como los valores por defecto a utilizarse en toda la app.

Atributos android:versionCode y android:versionName

Estos valores son del elemento <manifest> y son utilizados en el control de versiones de la app por parte de las tiendas de aplicaciones.

Configurar reléase App

Atributos android:minSdkVersion y android:targetSdkVersion

Se encuentran en el elemento <uses-sdk> para indicar las versiones adecuadas para la app.

Actualizar las rutas a los servidores

Si la app accede a servidores remotos hay que verificar que las URLs utilizadas por la app correspondan a las de producción y no otros ambientes Una forma de manejar esto es inyectando variables al manifiesto.

Configurar reléase App

Compilar la release app

Con la configuración terminada, se puede compilar un APK de release, firmado y optimizado. El JDK incluye las herramientas para firmar el archivo APK: Keytool para administrar claves y certificados y Jarsigner para firmar y verificar la firma e integridad de los JAR firmados. Android Studio permite construir el APK firmado usando Gradle, que también puede ser ocupado para automatizar el proceso completo de compilación.

Configurar reléase App

Probar release app

Algunas pruebas básicas por hacer:

- Probar la app en al menos un dispositivo físico. Si bien los emuladores sirven mucho para el desarrollo, la real fluidez la veremos en un dispositivo físico.
- Probar la app en al menos una pantalla de teléfono y una pantalla de tablet para verificar que los elementos visuales tengan el tamaño correcto.
- Verificar el rendimiento de la app y la utilización de la batería sean aceptables

Firma de la App

Android exige que todos los APK sean firmados digitalmente con un certificado antes de poder instalarse.

Certificados y keystores

Un certificado de clave pública, también conocido como certificado digital o un certificado de identidad, contiene la clave pública de un par de claves públicas y privadas, y otros metadatos que identifican al propietario de la clave como el nombre, ubicación y compañía. El propietario del certificado guarda la clave privada correspondiente.

Firma de la App

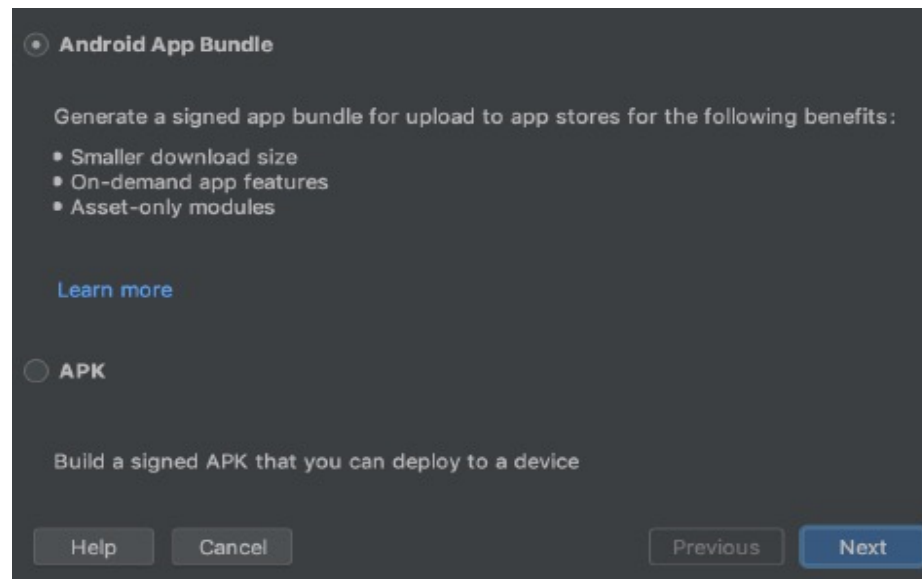
Android Al firmar un APK, la herramienta de firma adjunta el certificado de clave pública. El certificado de clave pública sirve como una "huella digital" que asocia de manera exclusiva el APK con la clave privada correspondiente. Esto permite a Android verificar que cualquier actualización futura a esa app sea auténtica y provenga del autor original. La clave que se usa para la creación de este certificado se llama clave de firma de apps.

Firma de la App

Un keystore es un campo binario que contiene una o más claves privadas. Por esto es importante conservar la clave privada, ya que las apps deben ser firmadas con la misma clave y certificado durante su vida útil para que los usuarios puedan instalar versiones nuevas como actualizaciones de la misma app.

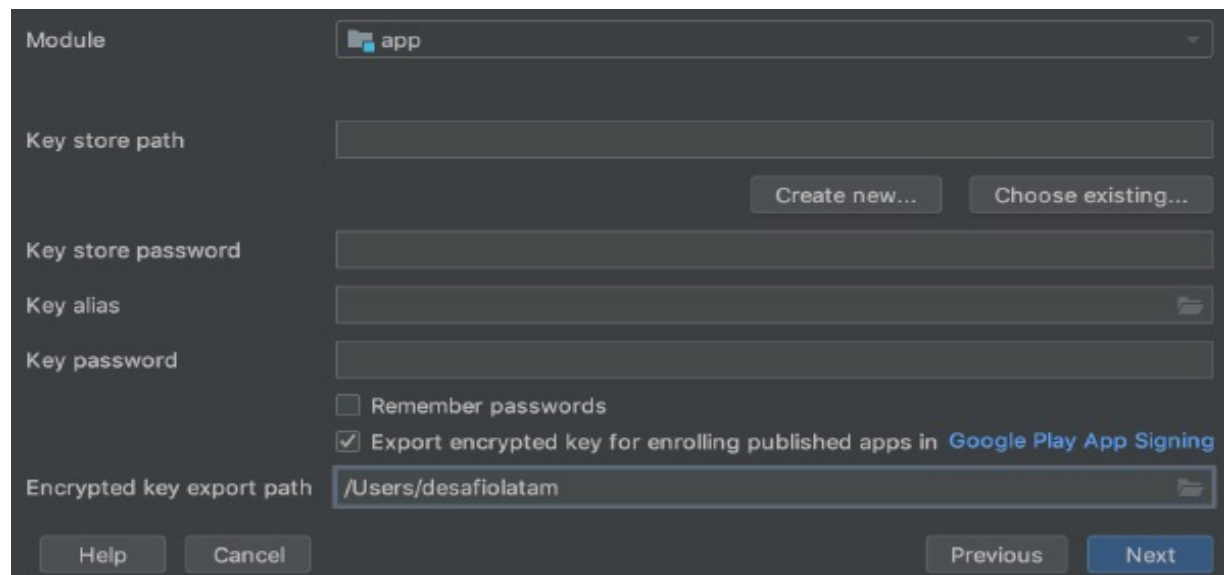
Firmar la reléase App

1. Seleccionar desde el menú Build -> Generate Signed Bundle or APK
2. En el cuadro de diálogo Generate Signed Bundle or APK se debe elegir una Android App Bundle o un APK



Firmar la reléase App

3. Si no se cuenta con un certificado, se puede crear uno seleccionando debajo del campo Key store path, la opción Create new.



The screenshot shows the 'Key Store' dialog box in Android Studio. The 'Module' dropdown is set to 'app'. The 'Key store path' field is empty, with 'Create new...' and 'Choose existing...' buttons to its right. The 'Key store password' field is empty. The 'Key alias' field is empty with a folder icon on the right. The 'Key password' field is empty. There are two checkboxes: 'Remember passwords' (unchecked) and 'Export encrypted key for enrolling published apps in Google Play App Signing' (checked). The 'Encrypted key export path' field contains '/Users/desafiolatam' with a folder icon on the right. At the bottom are 'Help', 'Cancel', 'Previous', and 'Next' buttons.

Firmar la reléase App

4. En el diálogo New Key Store se debe proporcionar la información para identificar al dueño

Key store path:

Password: Confirm:

Key

Alias:

Password: Confirm:

Validity (years):

Certificate

First and Last Name:

Organizational Unit:

Organization:

City or Locality:

State or Province:

Country Code (XX):

Cancel OK

Firmar la reléase App

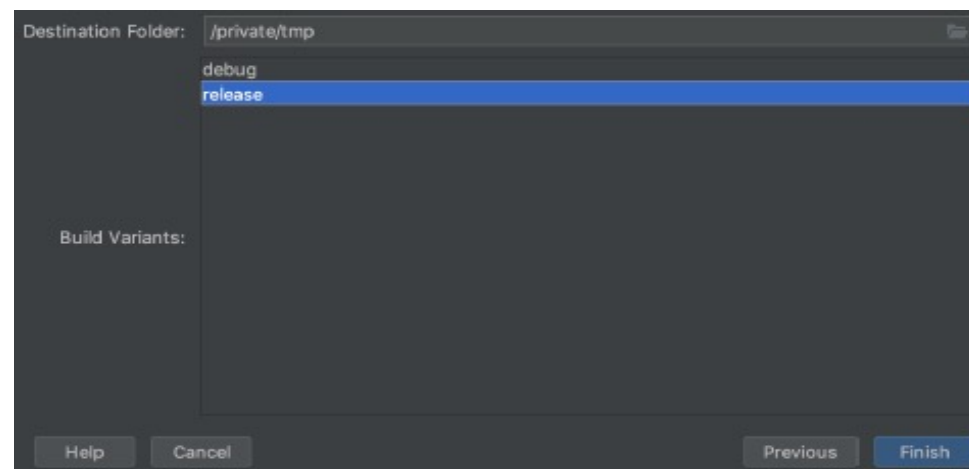
- Key store path: ubicación en la que se va a crear la Key Store
- Password: la contraseña de la Key Store
- Alias: Nombre de identificación para la key (clave)
- Password: Contraseña segura para la key que debe ser distinta a la de la Key Store. Si bien las contraseñas pueden ser iguales, se recomienda que sean distintas

Firmar la reléase App

- Validity: Cantidad de años que dura la key que debe ser al menos 25 años
- Certificate: Datos personales que no se muestran en la app pero se incluyen como parte del APK
- Country Code: corresponde al código de 2 letras definidos en la ISO 3166 como código de país. En el caso de Chile es cl

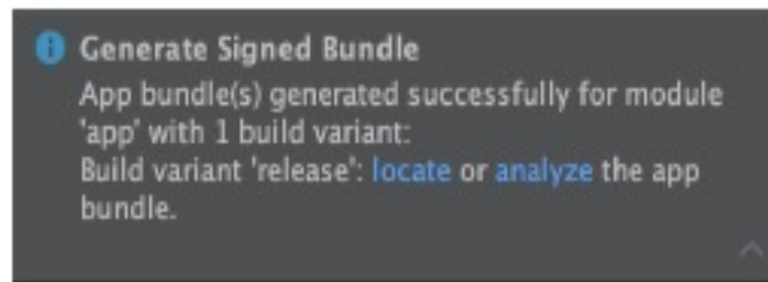
Firmar la reléase App

5. Luego de crear la Key Store, la siguiente parte es seleccionar la Build Variant adecuada. En este caso release, aunque es posible seleccionar más de una Build Variant.



Firmar la reléase App

6. Al finalizar el proceso, se ejecuta la construcción con Gradle que al finalizar muestra un mensaje indicando la creación del Bundle



Firma de App de Google Play

Con la firma de apps de Google Play, Google gestiona y protege la clave de firma de la app y la utiliza para firmar los APK. Para usar la firma de apps de Google Play se utilizan 2 claves:

- La firma de apps
- La firma de subida

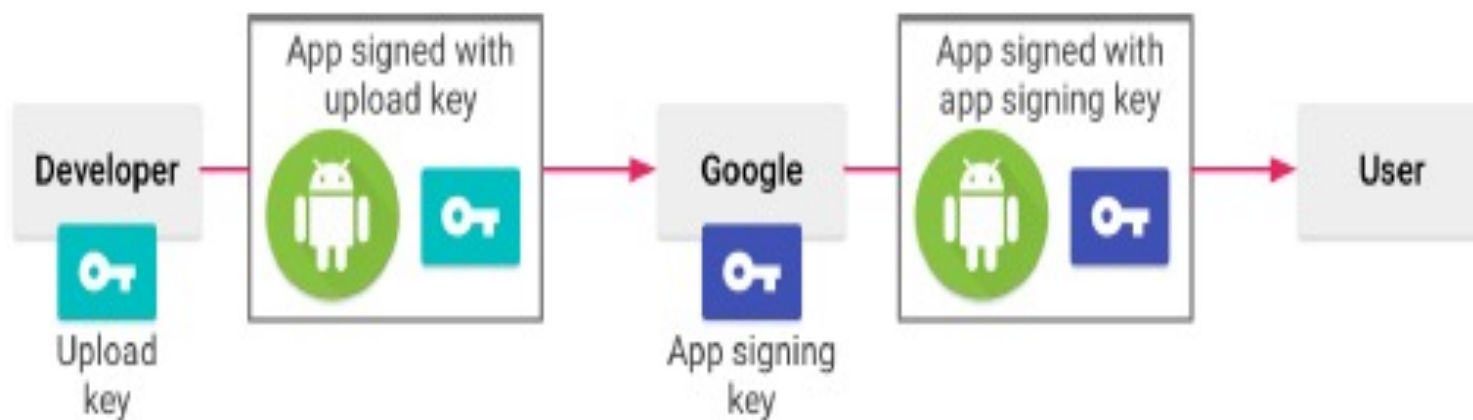
Firma de App de Google Play

Al usar la firma de apps de Google Play se encripta y exporta la clave de firma de apps usando Play Encrypt Private Key (PEPK) de Google Play, para subirla a la infraestructura de Google. Al momento de la publicación, la app es firmada con la clave de subida y subida a Google Play. Luego, usando el certificado de subida para verificar la identidad del autor se firma el APK con la clave de firma de apps para su distribución

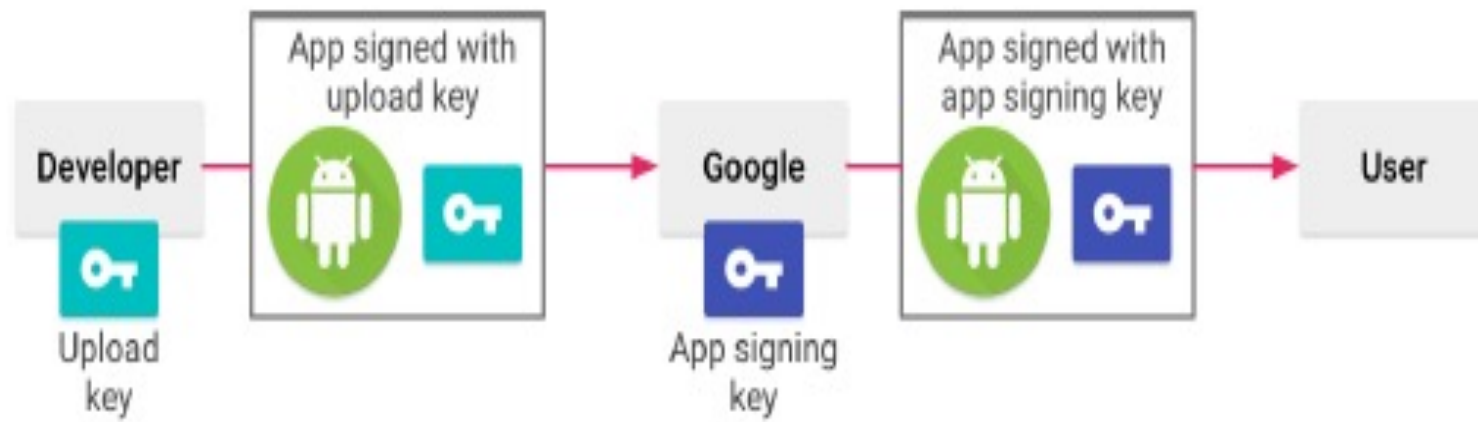
Firma de App de Google Play

Si la clave de subida está comprometida o se pierde es posible anularla y crear una nueva. Como la clave de firma de apps está en los servidores de Google se seguirá usando y la app puede seguir siendo actualizada

Firma de App de Google Play



Firma de App de Google Play



Firma de App de Google Play

Para utilizar esta opción, al generar un nuevo signed bundle de release se debe seleccionar la opción de exportar las claves encriptadas para usar Google Play App Signing.

Module: app

Key store path: [Empty] [Create new...] [Choose existing...]

Key store password: [Empty]

Key alias: [Empty]

Key password: [Empty]

☐ Remember passwords

☒ Export encrypted key for enrolling published apps in Google Play App Signing

Encrypted key export path: /Users/desafiolatam

[Help] [Cancel] [Previous] [Next]

Otras formas de distribución para Apps

Android brinda protección a los usuarios contra descargas e instalaciones inadvertidas de apps desde otras ubicaciones que no sean Google Play (confiable).

Para poder usar otras formas de distribución distintas a la tienda oficial es necesario configurar el dispositivo Android para que permita la instalación desde fuentes desconocidas.

Otras formas de distribución para Apps

Activar fuentes desconocidas

Se debe activar la opción para permitir fuentes desconocidas desde el menú de Ajustes -> Seguridad. Estas instalaciones se bloquean hasta que el usuario lo permita activando explícitamente fuentes desconocidas en Ajustes > Seguridad.

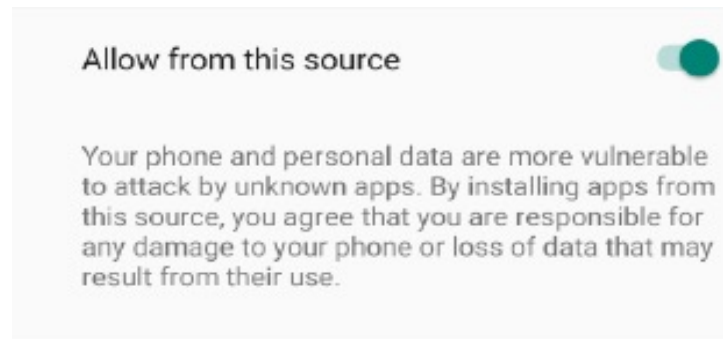
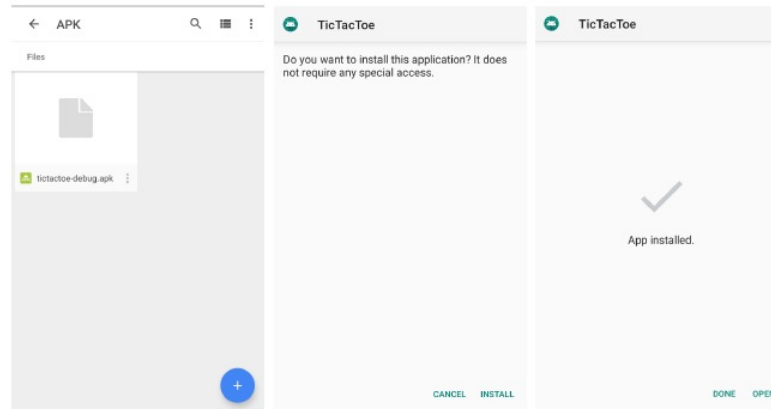


Imagen 20. Activar fuente desconocida.

Otras formas de distribución para Apps

Distribuir usando Google Drive

Por razones de seguridad ya no es posible enviar archivos de extensión APK por correo usando Gmail, sin embargo, cuando el APK está listo, se puede compartir desde Google Drive. Al seleccionar desde el dispositivo Android el APK compartido será instalado



Otras formas de distribución para Apps

Distribuir usando un servidor

Al igual que distribuirlo usando Google Drive, el APK se puede alojar en un servidor para que los usuarios puedan acceder desde un dispositivo Android para la instalación con la misma pantalla de confirmación y postinstalación que al instalar de Google Drive

Crear una aplicación en la consola

Se debe crear una aplicación en la consola que tiene la información asociada a la app y el instalador que debe utilizar. Para esto se deben seguir los siguiente pasos:

- Ir a la Play Console
- Crear una aplicación usando el botón CREA UNA APLICACIÓN
- Rellenar el título y el idioma predeterminado para luego crear la app

Crear una aplicación en la consola

Crea una aplicación

Idioma predeterminado ^{*}

Español (Latinoamérica) – es-419 [⌵]

Título ^{*}

0/50

CANCELAR **CREAR**



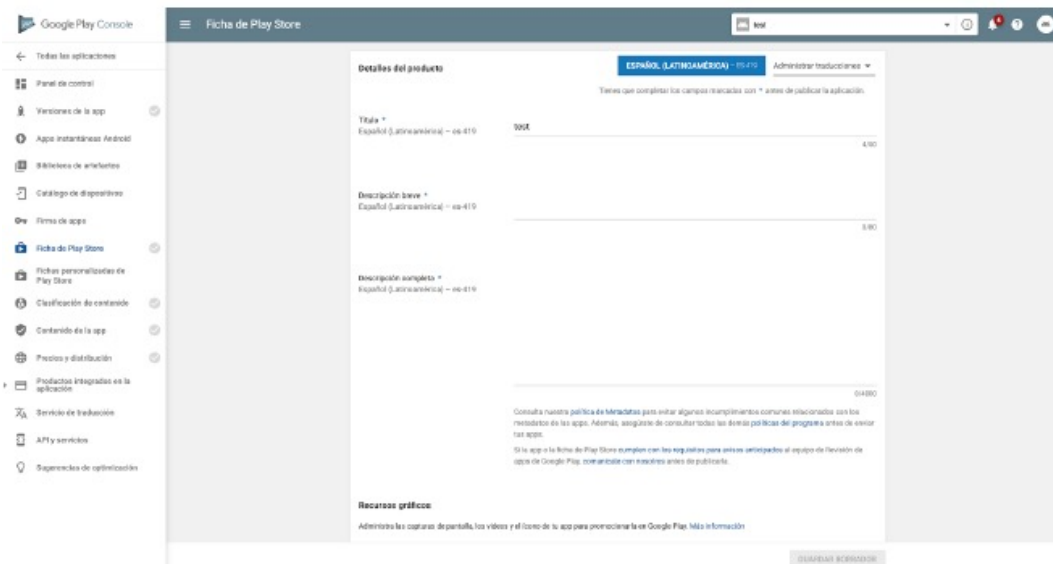
OTEC
EDUCACIÓN
CONTINUA

Partner
Laboratoria

**TALENTO
DIGITAL**
INTELIGENCIA
HUMANA

Crear una aplicación en la consola

La redirección es hacia la configuración de la app, donde en la parte izquierda se indica los componentes obligatorios que deben ser completados antes de realizar el lanzamiento.



Clasificación de contenido

En la tienda se clasifican las apps en distintas categorías por público objetivo, contenido de la app y si es de pago o gratis. Al crear una nueva app se debe entregar información para determinar los criterios de clasificación.

Al seleccionar desde el menú Contenido de la app se puede realizar la categorización contestando una serie de preguntas de contenido y precio

Clasificación de contenido



Nueva Política de Familias de Google Play

Realizamos modificaciones en la [Política de Familias de Google Play](#). Revisa la política y completa la sección "Público objetivo y contenido" en la página de contenido de la app antes del 1 de septiembre de 2019. Google revisará tu app en función de la información que proporciones. Las declaraciones que hayas hecho anteriormente sobre la idoneidad de tu app con respecto al uso por parte de menores no se tendrán en cuenta.

[MÁS INFORMACIÓN](#)

Público objetivo y contenido

Cuéntanos cuál es el público objetivo de tu app y comparte más información sobre el contenido. Esto nos ayuda a garantizar que las apps que estén diseñadas para niños sean seguras y adecuadas.

 Cuéntanos si tu app contiene anuncios antes de comenzar la sección "Público objetivo y contenido"

INICIAR

Clasificación de contenido

La clasificación del tipo de app se calcula en forma automática luego de contestar una serie de preguntas orientadas a verificar si cumple con las políticas del programa para desarrolladores que deben ser respetadas para que la app pueda ser publicada.

Entre los tópicos consultados se encuentra el uso de la violencia, miedo y humor crudo

Clasificación de contenido

Completa el cuestionario para que podamos calcular la calificación de tu aplicación.

 **JUEGO**
La aplicación es un juego... [Editar categoría](#)

VIOLENCIA CERRAR ✓

¿El juego incluye inferencias, referencias o representaciones relacionadas con la violencia? *
Ten en cuenta que esta pregunta no se refiere al contenido generado por los usuarios.

☐ Sí ☒ No

IEDO

¿El juego incluye imágenes o sonidos que podrían asustar o infundir miedo? *
Ten en cuenta que esta pregunta no se refiere al contenido generado por los usuarios.

☐ Sí ☐ No

SEXUALIDAD

JUEGOS DE APUESTAS REALES O SIMULADAS, O PAGOS EN EFECTIVO

IDIOMA

SUSTANCIA CONTROLADA

HUMOR CRUDO

VARIOS

[CALCULAR CLASIFICACIÓN](#) [GUARDAR CUESTIONARIO](#) 

Precios y distribución

En la siguiente parte se indica si es una app de pago o gratis además de configurar los países en los que estará disponible la app.

The screenshot shows the Google Play Store app configuration interface. At the top, there are icons for Google Play admin, Wear OS by Google, and Android Auto. Below these, the 'La aplicación es' section has two buttons: 'PAGADA' (disabled) and 'GRATIS' (selected). A note below states: 'Para publicar aplicaciones pagadas, tienes que configurar una cuenta del comerciante. Más información'. The 'Disponibilidad de la app' section shows a message: 'En este momento, tu app no está disponible en Play Store.'. The 'Países' section shows 'Países no disponibles' as 148 and 'Países disponibles' as 1 (Alfa y Beta sincronizadas con producción). There is an 'ADMINISTRAR PAISES' button. The 'Contiene anuncios' section has a question: '¿Tu aplicación tiene anuncios? Además, consulta nuestra Política de anuncios para evitar incumplimientos comunes. Si es así, los usuarios podrán ver la etiqueta "anuncios" en la aplicación en Play Store. Más información'. There are two radio buttons: 'Si, tiene anuncios' (selected) and 'No, no tiene anuncios'.

Países no disponibles	Países disponibles
148	1
y resto del mundo	Alfa y Beta sincronizadas con producción

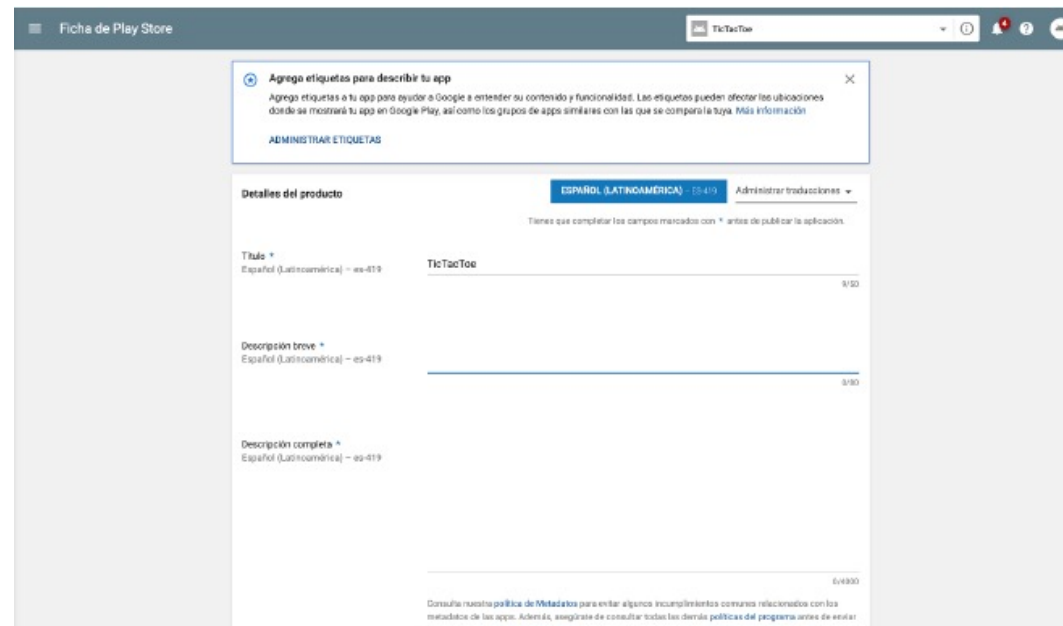


OTEC
EDUCACIÓN
CONTINUA

Partner
Laboratoria

TALENTO
DIGITAL
INTELIGENCIA
HUMANA

Ficha de la Play Store



The screenshot shows the 'Ficha de Play Store' (Play Store Listing) for an app named 'TicTacToe'. The interface is in Spanish (Latinoamérica). At the top, there's a notification box titled 'Agrega etiquetas para describir tu app' (Add tags to describe your app), which explains that tags help Google understand the app's content and functionality, and can affect its visibility in search results. Below this, the 'Detalles del producto' (Product details) section is visible. It includes a language selector set to 'ESPAÑOL (LATINOAMÉRICA)' and a link to 'Administrar traducciones'. A note states: 'Tienes que completar los campos marcados con * antes de publicar la aplicación.' (You must complete the fields marked with * before publishing the application). The form fields are: 'Título *' (Title) with the value 'TicTacToe' and a character count of 8/30; 'Descripción breve *' (Short description) with a character count of 0/100; and 'Descripción completa *' (Full description) with a character count of 0/4000. At the bottom, there is a link to 'Consulta nuestra política de Metadatos' (Check our metadata policy) for more information on common compliance issues.

Ficha de Play Store

TicTacToe

Agrega etiquetas para describir tu app
Agrega etiquetas a tu app para ayudar a Google a entender su contenido y funcionalidad. Las etiquetas pueden afectar las ubicaciones donde se mostrará tu app en Google Play, así como los grupos de apps similares con las que se compara la tuya. [Más información](#)
[ADMINISTRAR ETIQUETAS](#)

Detalles del producto **ESPAÑOL (LATINOAMÉRICA)** 0/419 [Administrar traducciones](#)

Tienes que completar los campos marcados con * antes de publicar la aplicación.

Título *
Español (Latinoamérica) – es-419
TicTacToe
8/30

Descripción breve *
Español (Latinoamérica) – es-419
0/100

Descripción completa *
Español (Latinoamérica) – es-419
0/4000

[Consulta nuestra política de Metadatos](#) para evitar algunos incumplimientos comunes relacionados con los metadatos de las apps. Además, asegúrate de consultar todas las demás [políticas del programa](#) antes de enviar tu app.

Detalle de producto

La mayoría de los detalles del producto son obligatorios y corresponden a la información básica que se muestra en la tienda de aplicaciones

- Título
- Descripción breve
- Descripción completa

Detalle de producto

Los recursos gráficos serán mostrados al usuario dentro de la tienda de aplicaciones y corresponden a íconos, capturas de pantalla y videos promocionales. La documentación oficial ofrece guías actualizadas sobre formatos, restricciones y configuraciones, que quedan disponibles para todas las pistas

- Ícono de alta resolución
- Capturas de pantalla
- Gráfico de la función
- Video promocional

Detalle de producto

Los recursos gráficos serán mostrados al usuario dentro de la tienda de aplicaciones y corresponden a íconos, capturas de pantalla y videos promocionales. La documentación oficial ofrece guías actualizadas sobre formatos, restricciones y configuraciones, que quedan disponibles para todas las pistas

- Ícono de alta resolución
- Capturas de pantalla
- Gráfico de la función
- Video promocional

Detalle de producto

La versión de prueba interna se usa para distribuir rápidamente la app para pruebas internas y controles de calidad usando los testers internos (se puede tener hasta 100 testers) Entre otras características, se encuentran:

- Rapidez: Distribuir apps es mucho más rápido desde la pista de prueba interna que desde las pistas abiertas o cerradas. Al publicar un APK o app bundle actualizado en la pista de prueba interna, estará disponible para los verificadores en pocos minutos.

Detalle de producto

- Flexibilidad: Las pruebas internas se pueden ajustar para admitir distintas etapas de prueba. Esto incluye pruebas internas, verificaciones de calidad y depuraciones posteriores al lanzamiento.
- Seguridad: Con la pista de prueba interna, tu app de prueba se distribuye a los usuarios a través de Play Store.

Administrador de verificadores

La administración de verificadores o testers permite inscribir a una lista de usuarios que pueden acceder a la app de prueba interna usando la URL para participar

Administrar verificadores

Prueba Interna ^

Elige cómo ejecutar el programa de prueba. [Más información](#)



100 verificadores internos por app

Tu programa de prueba interna puede tener un máximo de 100 verificadores a la vez. Podrán participar los primeros 100 que se registren.

Usuarios

Nombre de la lista		Usuarios	
<input checked="" type="checkbox"/>	internos	1	EDITAR

[CREAR LISTA NUEVA](#)

Canal de comentarios

Dirección de correo electrónico o URL

URL para participar

<https://play.google.com/apps/interna/test/4698282699905858618>



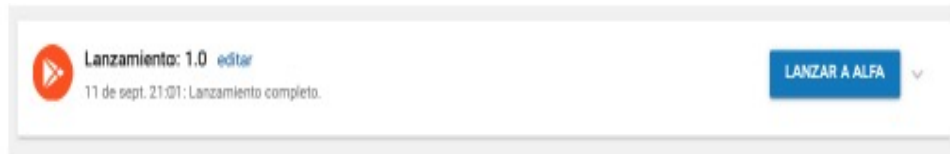
QUITAR VERIFICADORES

GUARDADO

Pista cerrada (Alpha)

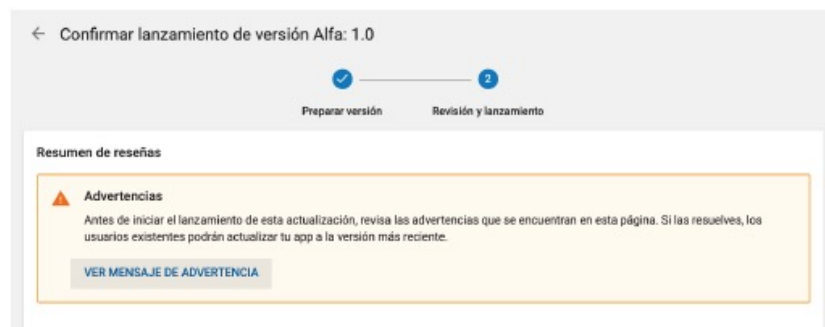
Los grupos de prueba cerrados deben ser pequeños y probar las versiones más inestables y experimentales de tu app y se puede ejecutar una prueba cerrada con direcciones de correo electrónico individuales.

Para esto se debe lanzar a ALFA (Alpha) la app previamente en la pista de prueba interna



Pista cerrada (Alpha)

En el primer paso se prepara la versión que indica la promoción desde la prueba interna. Luego, en el paso indica que existen advertencias que deben ser solucionadas antes de hacer el lanzamiento.



Pista cerrada (Alpha)

En este caso particular de la app TicTacToe no se ha configurado los verificadores, como indica al presionar VER MENSAJE DE ADVERTENCIA

1 mensaje

1 mensaje para la actualización



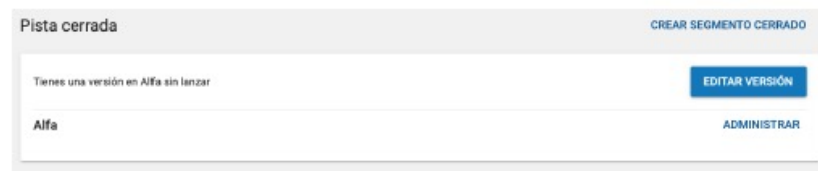
Advertencia

Como no especificaste los verificadores, esta versión no estará disponible para ningún usuario. Sugerencia: Configura tu segmento de prueba para garantizar que la versión esté disponible para los verificadores.

CERRAR

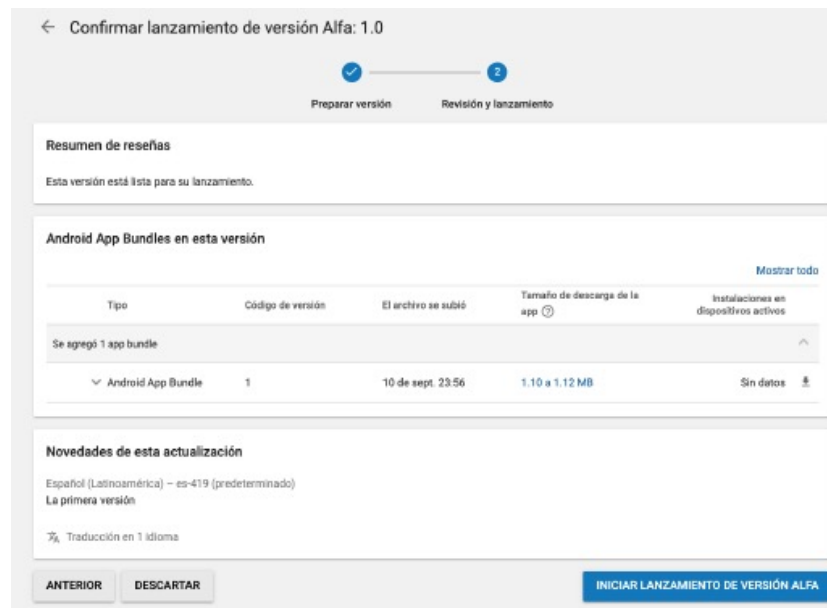
Pista cerrada (Alpha)

No existe un segmento de prueba cerrado configurado por lo que la app no será distribuida a algún usuario



Pista cerrada (Alpha)

Luego de crear el segmento, la app aún no sido lanzada. Al editar la versión, la advertencia ha sido solucionada y permite iniciar el lanzamiento de versión alfa



Pista abierta (beta)

Después de haber probado una versión cerrada, el lanzamiento abierto puede incluir una gama más amplia de usuarios para pruebas, antes de que la app entre en producción. La prueba abierta facilita un link que se puede compartir directamente en un foro, a través de una cuenta de correo colectiva o algún otro medio, con la finalidad de conseguir un beta testing masivo.

Pista de producción

La versión de producción está disponible para todos los usuarios en los países que fueron configurados y es accesible en la tienda de aplicaciones. La publicación en producción supone una espera de al menos 2 hrs para que sea efectiva, por lo que la actualización no es inmediata.

Flujo de release

El flujo propuesto permite promover la app desde una prueba interna hasta la puesta en producción para probar nuevas funcionalidades y evaluar tanto el desempeño como la estabilidad previa a la puesta en producción



Pistas separadas por cantidad de usuarios

Ventajas:

- Disminuye el riesgo de afectar a los usuarios finales con bugs que pueden ser descubiertos en las etapas iniciales
- Probar funcionalidades con grupos de early users y comprobar hipótesis de comportamiento
- Administrar a quiénes y cuándo se debe actualizar la app

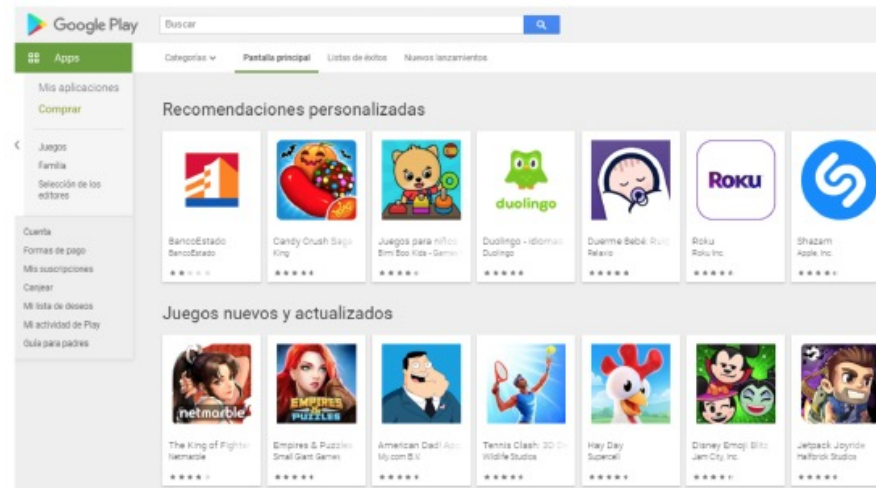
Cuidados:

Considerar en la planificación de lanzamiento los tiempos de publicación por cada pista

Google Play

Es una plataforma de distribución digital de aplicaciones móviles para los dispositivos con sistema operativo Android, así como una tienda en línea desarrollada y operada por Google. Esta plataforma permite a los usuarios navegar y descargar aplicaciones (desarrolladas mediante Android SDK), juegos, música, libros, revistas y películas. También se pueden adquirir dispositivos móviles como ordenadores Chromebook, teléfonos inteligentes Nexus, Google Chromecast, entre otros productos de la empresa.

Google Play



Pruebas de desarrollo en la consola

- Prueba interna: te permite distribuir tu app de forma rápida para realizar pruebas internas y verificaciones de control de calidad.
- Pruebas Abiertas: te permiten realizar una verificación con un grupo más amplio y publicar la versión de prueba de tu app en Google Play. Si realizas una prueba abierta, cualquier usuario podrá unirse al programa y enviarte comentarios privados. Antes de elegir esta opción, asegúrate de que tu app y su ficha de Play Store estén listas para tener visibilidad en Google Play.

Nota: Ya no es posible crear pruebas Alfa abiertas ni Beta cerradas. Seguirás teniendo acceso a aquellas que ya estén activas.

Pruebas de desarrollo en la consola

Pruebas Cerradas: Te permite realizar pruebas de las versiones previas al lanzamiento de tu app con un grupo de verificadores más extenso. Cuando hayas realizado pruebas con un grupo reducido de empleados o usuarios de confianza, podrás expandir la prueba a una versión abierta. En la página Versiones de la app, encontrarás un segmento Alfa disponible como prueba cerrada inicial. Si es necesario, también puedes crear segmentos cerrados adicionales y colocarles nombres. Si quieres probar una app que ya publicaste anteriormente, solo los usuarios del grupo de prueba recibirán una actualización de la versión cerrada.

Pruebas de desarrollo en la consola

Hay cuatro tipos de ambientes de lanzamiento que admite la Consola de Google Play:

- Pruebas Internas: este ambiente es relativamente nuevo y se utiliza principalmente para que los desarrollos prueben incluso durante la construcción de la aplicación. Este ambiente pretende facilitar la colaboración entre desarrolladores.

Pruebas de desarrollo en la consola

- Alfa: en alfa debemos crear una versión para realizar rápidamente las primeras pruebas de la aplicación. Este canal le permite distribuir una acción a un pequeño conjunto de usuarios sin pasar por una revisión completa de Google. Los usuarios recibirán advertencias de que la acción no se ha sometido a una revisión de Google para el cumplimiento de la política. Una vez que hayamos probado con un grupo más pequeño de usuarios de confianza, podemos ampliar las pruebas a una versión beta cerrada.

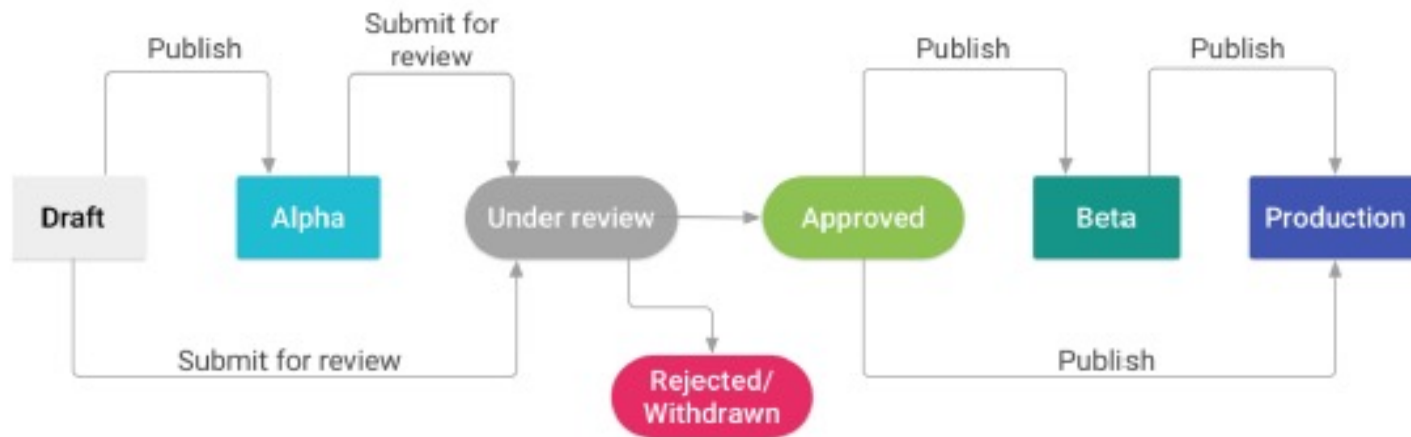
Pruebas de desarrollo en la consola

- Beta: este canal o ambiente, permite distribuir una acción a un conjunto limitado de usuarios después de pasar una revisión completa de Google. Debe usar este canal para dar acceso a los usuarios fuera de su organización a las primeras versiones de la acción. Se informará a los usuarios que la acción es una versión preliminar. Dado que la versión beta cerrada ha pasado la revisión de Google, puede optar por hacerla pública en cualquier momento sin otra revisión por parte de Google.

Pruebas de desarrollo en la consola

- Producción: este es el ambiente definitivo que disponibiliza tu aplicación para todo tipo de usuarios, en grupos, pueblos, ciudades y países. Para realizar el lanzamiento en producción es necesario que pasemos por todos los ambientes de prueba para asegurarnos de que los errores e inconvenientes hayan sido minimizados, asegurando la supervivencia de nuestra aplicación en la tienda de google play.

Pruebas de desarrollo en la consola



Analítica usando Crashlytics

La estabilidad en una app es un factor relevante a la hora de conquistar a los usuarios. Las apps con bugs hacen usuarios infelices que pueden desde dar una mal review hasta desinstalar la app.

Cuando un usuario descontento desinstala una app, es muy complicado que la pueda volver a instalar.

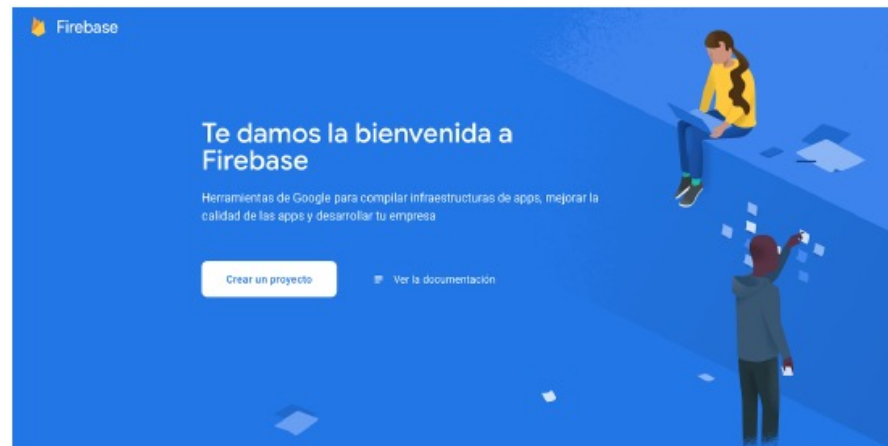
Dependiendo de la app y la cantidad de usuarios los bugs pueden convertirse en un dolor de cabeza, ocupando tiempo, personas y recursos en revisar, priorizar y solucionar los errores.

Requisitos

La Para instalar Firebase Crashlytics se debe tener como requisito una cuenta para usar la consola de Firebase y habilitar Google Analytics para el proyecto

Crea un proyecto con Firebase

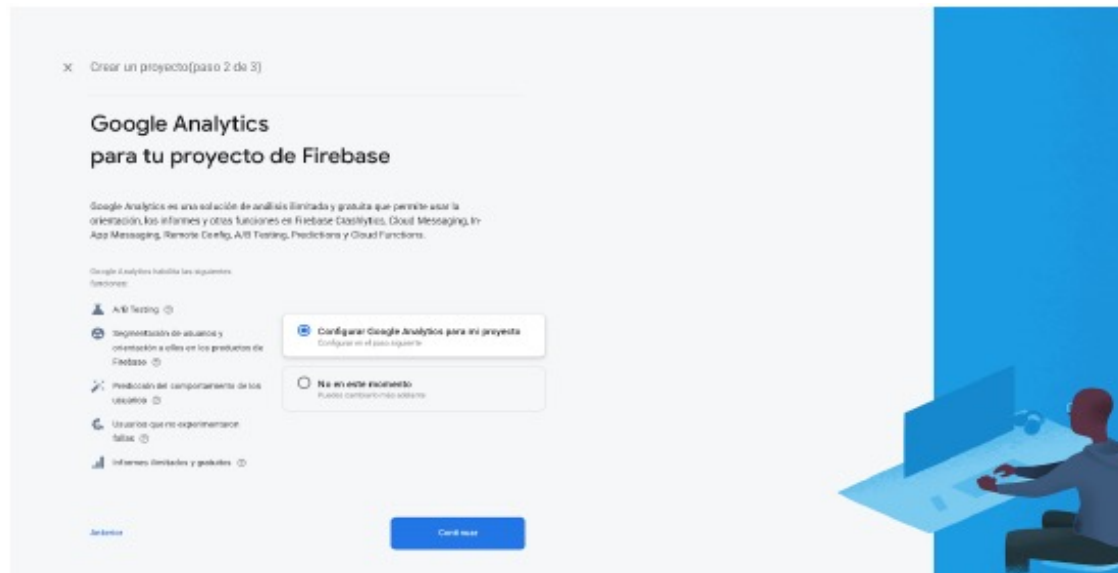
Primero se debe tener un proyecto Firebase. Para crearlo se accede a la página oficial de la consola de Firebase



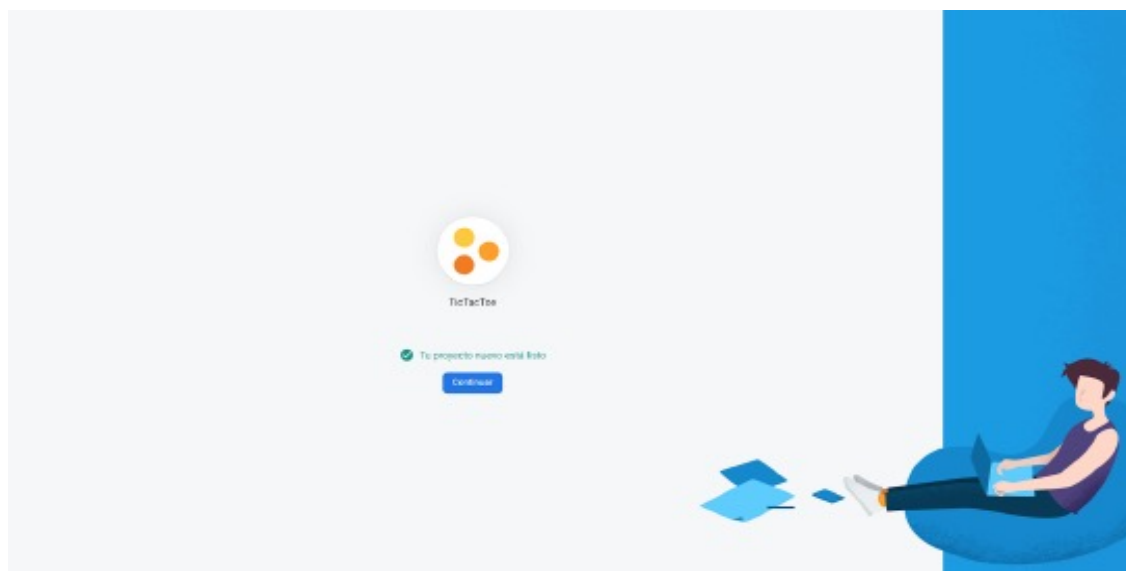
Crea un proyecto con Firebase



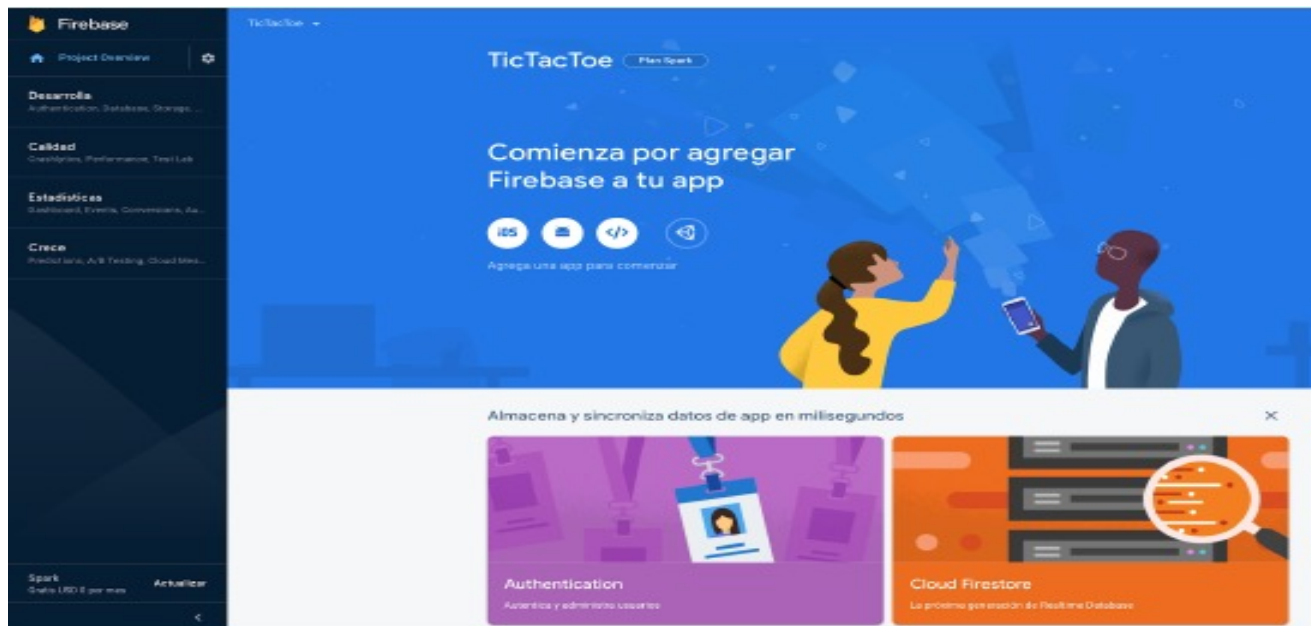
Crea un proyecto con Firebase



Crea un proyecto con Firebase



Crea un proyecto con Firebase



Habilitar Google Analytics

Google Analytics también ofrece información importante al desarrollador, pero con un propósito de verificar la cantidad e instalaciones, información de que pantalla pasan más los usuarios, entre otros. Es una información complementaria a la que podemos tener usando Crashlytics. Para poder utilizar Crashlytics es necesario configurar Google Analytics en la consola de Firebase. Para esto, desde el menú izquierdo, en el apartado Calidad > Crashlytics se agrega un nuevo proyecto.



Habilitar Google Analytics

La configuración de Google Analytics pide aceptar condiciones para protección de datos, mediciones de datos en general y las condiciones de Google Analytics. Una vez confirmado que se está de acuerdo con los documentos, la habilitación es bastante simple

Habilitar Google Analytics

× Configurar Google Analytics

Elige o crea una cuenta de Google Analytics ⓘ

TicTacToe

Ubicación de Analytics ⓘ

Chile

Configuración de uso compartido de datos y Condiciones de Google Analytics

☒ Usar la configuración predeterminada para compartir datos de Google Analytics [Learn more](#)

☒ Compartir tus datos de Analytics con Google para mejorar nuestros productos y servicios

☒ Compartir tus datos de Analytics con Google para habilitar las comparativas

☒ Compartir tus datos de estadísticas con Google para habilitar la asistencia técnica

☒ Compartir tus datos de Analytics con los especialistas en Cuentas de Google

☒ Acepto las [Condiciones de la Protección Conjunta de Datos por parte del Servicio de Medición de Datos y Google](#), y confirmo que estoy sujeto a la [Política de Consentimiento de Usuarios Finales de la UE](#). Esto es obligatorio cuando se comparten datos de Google Analytics para mejorar los productos y servicios de Google. [Más información](#)

☒ Acepto las [Condiciones de Google Analytics](#).

Se creará una cuenta nueva de Google Analytics, se transferirá tu propiedad a ella y se volverá a vincular con tu proyecto de Firebase. Este vínculo permitirá el flujo de datos entre los productos. Ten en cuenta que los datos que se exportan de tu propiedad de Google Analytics a Firebase están sujetos a las Condiciones del Servicio de Firebase, mientras que los datos de Firebase que se importan a Google Analytics están sujetos a las Condiciones del Servicio de Google Analytics. [Obten más información](#)

Habilitar Google Analytics

Configurar Crashlytics

Desde el home de firebase es accesible Crashlytics para su configuración



Configurar Crashlytics

Al seleccionar la opción se debe iniciar la configuración de Crashlytics



La solución de generación de informes de fallas
más poderosa y ligera

[Más información](#)

Configurar Crashlytics

Configurar Crashlytics

A continuación se debe ejecutar 3 pasos



La solución de generación de informes de fallas más poderosa y ligera

[Más información](#)

1 ¿Eres un usuario de Fabric que quiere migrar una app de Crashlytics? ⓘ

☒ No, quiero configurar una app de Firebase nueva

Selecciona esta opción si quieres configurar Crashlytics para una app nueva.

☐ Sí, quiero migrar mi app de Fabric a Firebase

Selecciona esta opción si eres un usuario existente de Fabric y quieres migrar a la app de Fabric Crashlytics app. Los datos de fallas aparecerán en los paneles de Fabric y de Firebase.

[Siguiente](#)

2 Instala el SDK

3 Compila y ejecuta tu app

Instalar JDK de Crashlytics

Se debe modificar a el script build.gradle a nivel de proyecto para agregar el plugin de Crashlytics

```
buildscript {
    repositories {
        // ...
        // Add the following repositories:
        maven {
            url 'https://maven.fabric.io/public'
        }
    }

    dependencies {
        // ...

        // Check for v3.1.2 or higher
        classpath 'com.google.gms:google-services:4.3.2' // Google Services plugin

        // Add dependency
        classpath 'io.fabric.tools:gradle:1.31.0' // Crashlytics plugin
    }
}
```

Instalar JDK de Crashlytics

Además, a nivel de módulo se debe aplicar el plugin y agregar la dependencia a Crashlytics

```
apply plugin: 'com.android.application'
apply plugin: 'io.fabric'

dependencies {
    // ...

    // (Recommended) Add Analytics
    implementation 'com.google.firebase:firebase-analytics:17.2.0'

    // Add dependency
    implementation 'com.crashlytics.sdk.android:crashlytics:2.10.1'
}
```


Instalar JDK de Crashlytics

Después de sincronizar el proyecto queda listo para compilar y ejecutar



La solución de generación de informes de fallas más poderosa y ligera

[Más información](#)

- ✓ Configura una app de Firebase nueva
- ✓ Instala el SDK
- 3 **Compila y ejecuta tu app**
Esperaremos a que tu app se comunique con nuestros servidores.

Instalar JDK de Crashlytics

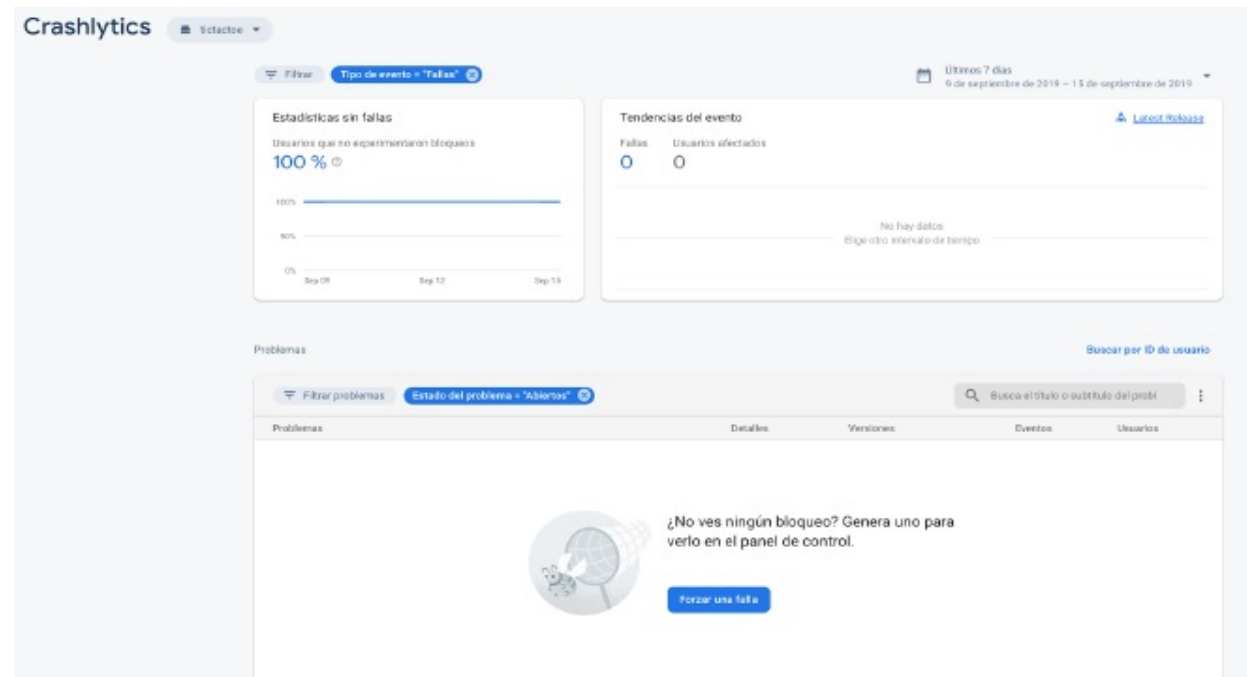
Con este paso se completa la inclusión y configuración de Crashlytics en la app. Para finalizar el proceso, basta con desinstalar la app y volver a instalarla con estas modificaciones para que se comunice con el servidor y quede habilitado Firebase Crashlytics



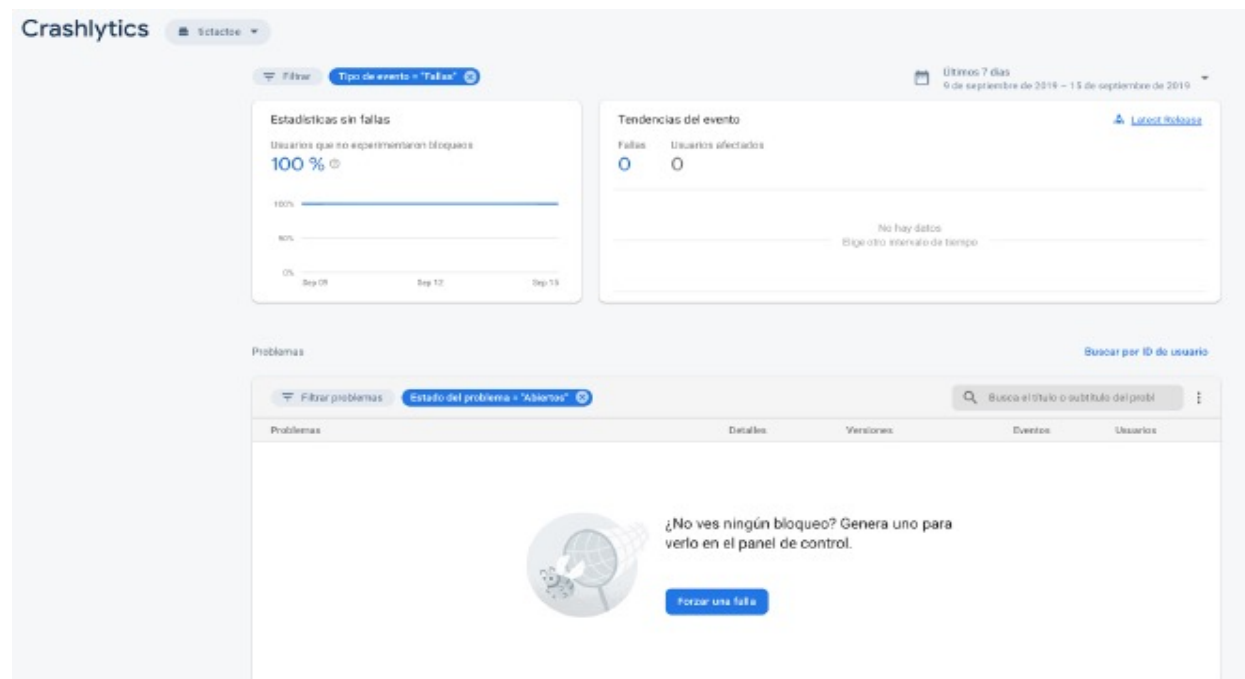
Installation successful!

[Ir al panel de Crashlytics](#)

Panel de Crashlytics



Panel de Crashlytics



Inicializar Crashlytics

La inicialización de Crashlytics debe hacerse al comienzo de la ejecución de la app de modo que quede disponible lo antes posible para recopilar los problemas. Se realiza llamando de la siguiente forma al builder de Crashlytics.

```
private void setupCrashlytics() {  
    Crashlytics crashlyticsKit = new Crashlytics.Builder()  
        .core(new CrashlyticsCore.Builder().build())  
        .build();  
  
    Fabric.with(this, crashlyticsKit);  
}
```

Personalizar los reportes

Para entregar más información dentro de los reportes, Crashlytics entrega 4 mecanismos de logging custom keys, custom logs, identificación de usuarios, y manejo de excepciones no fatales.

Claves personalizadas (custom keys)

Asocian pares clave/valor arbitrarios que aportan estados específicos de la app cuando ocurre un problema. Existen 5 métodos para asignar el valor a una clave

Personalizar los reportes

```
Crashlytics.setString(key, "foo" /* string value */);  
  
Crashlytics.setBool(key, true /* boolean value */);  
  
Crashlytics.setDouble(key, 1.0 /* double value */);  
  
Crashlytics.setFloat(key, 1.0f /* float value */);  
  
Crashlytics.setInt(key, 1 /* int value */);
```

Personalizar los reportes

Se pueden entregar hasta 64 pares que entreguen información extra sobre el estado de la app, por ejemplo

```
Crashlytics.setInt("current_level", 3);  
Crashlytics.setString("last_UI_action", "logged_in");
```


Personalizar los reportes

Se pueden entregar hasta 64 pares que entreguen información extra sobre el estado de la app, por ejemplo

```
Crashlytics.setInt("current_level", 3);  
Crashlytics.setString("last_UI_action", "logged_in");
```