



II – CodeLab Game Service

Contenido



01 Visión general del Demo

02 Configuración de AG y ambiente

03 Codeando



01

Visión general del Demo

Visión general del Demo

Categoría		Descripción		Ejemplo	
Funcionalidad		Características principales del sistema		Ejemplos de uso	
Seguridad		Mecanismos de protección de datos		Ejemplos de ataques y defensas	
Performance		Análisis de rendimiento		Ejemplos de optimización	
Compatibilidad		Soporte de plataformas y dispositivos		Ejemplos de integración	
Escalabilidad		Capacidad de crecimiento		Ejemplos de expansión	
Mantenimiento		Procedimientos de actualización		Ejemplos de parches	
Integración		Conectores con otros sistemas		Ejemplos de APIs	
Documentación		Guías de usuario y administrador		Ejemplos de manuales	
Soporte		Canales de asistencia al cliente		Ejemplos de tickets	
Licenciamiento		Modelos de precios y condiciones		Ejemplos de contratos	
Marketing		Estrategias de promoción		Ejemplos de campañas	
Legal		Aspectos regulatorios		Ejemplos de cláusulas	
Financiero		Análisis de costos y beneficios		Ejemplos de presupuestos	
Recursos Humanos		Estructura organizativa		Ejemplos de organigramas	
Tecnología		Herramientas y equipos		Ejemplos de configuraciones	
Comunicación		Protocolos de intercambio de información		Ejemplos de formatos	
Logística		Procesos de distribución		Ejemplos de rutas	
Operaciones		Flujos de trabajo		Ejemplos de diagramas	
Investigación y Desarrollo		Proyectos de innovación		Ejemplos de prototipos	
Ejecución		Implementación de planes		Ejemplos de cronogramas	
Evaluación		Métricas de éxito		Ejemplos de reportes	
Cierre		Resumen de hallazgos		Ejemplos de conclusiones	

A stylized wireframe illustration of a human hand, palm facing forward, rendered in a light blue color. The hand is positioned on the left side of the slide. Overlaid on the hand and extending outwards are several concentric circular lines and segments, resembling a futuristic interface or a data visualization. The background is a solid dark blue.

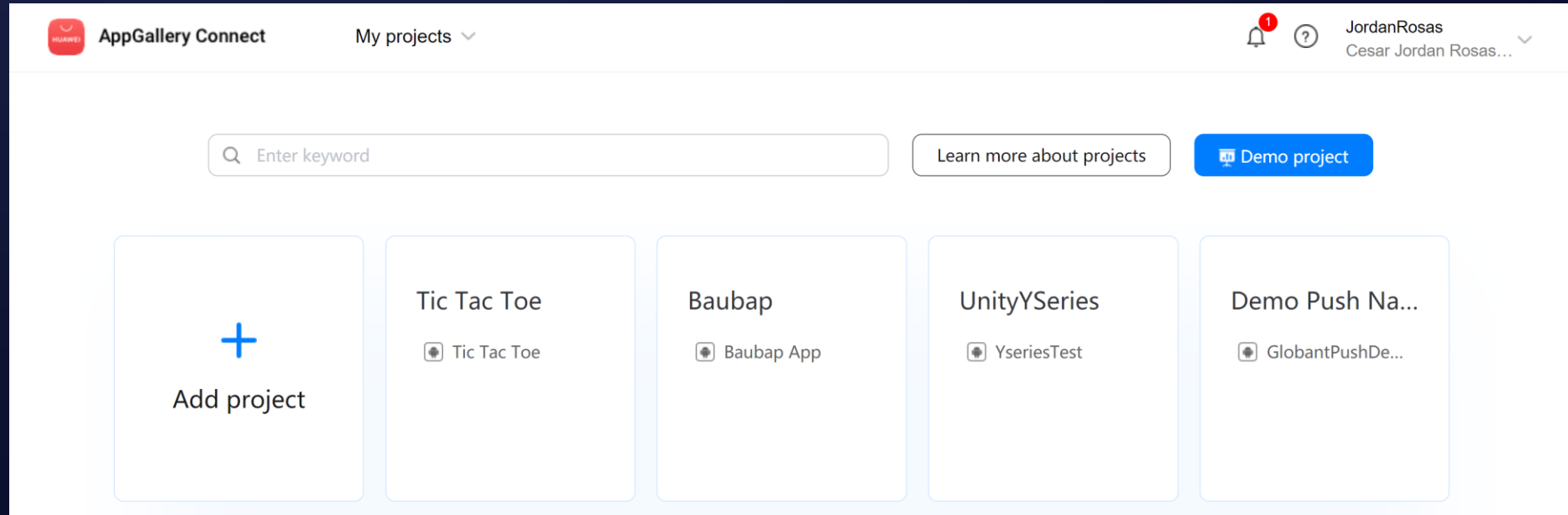
02

Configuración del AppGallery Connect y
el ambiente de desarrollo

Visitar la pagina oficial de Huawei Developer




<https://developer.huawei.com/>

Crear un nuevo proyecto llamado Tic Tac Toe





Crear una aplicación y agregarla al proyecto

Platform: **Android**
Device: **Mobile phone**
App category: **Game**



 **AppGallery Connect** My apps ▾  

App status ▾

 Search

 Demo project

New app

App	Status	Operation
 Tic Tac Toe	 Draft	Release

Generar un Fingerprint del certificado de firma de la aplicación

Tomar en cuenta que el JDK debe estar instalado en la computadora
Debe haber creado el certificado de firma

- 1.- Abrir una interface de línea de comandos, ir al path donde esta alojado el certificado
- 2.- Correr el comando: `keytool -list -v -keystore <keystore-file>`
- 3.- Obtener el certificado Fingerprint SHA-256

```
C:\Program Files\Java\jdk\bin>keytool -list -v -keystore C:\TestApp.jks
Enter keystore password:
Keystore type: jks
Keystore provider: SUN

Your keystore contains 1 entry

Alias name: key0
Creation date: Dec 23, 2019
Entry type: PrivateKeyEntry
Certificate chain length: 1
Certificate[1]:
Owner: CN=HUAWEI HMS
Issuer: CN=HUAWEI HMS
Serial number: 222d2080
Valid from: Mon Dec 23 14:45:27 CST 2019 until: Fri Dec 16 14:45:27 CST 2044
Certificate fingerprints:
    MD5: BF:DC:B8:D6:7F:C0:39:59:E9:D5:B3:86:EB:C5:80:A5
    SHA1: 53:08:E7:18:21:70:E3:E7:4C:FC:A3:67:F5:B7:E9:3D:69:40:13:A4
    SHA256: [REDACTED]
Signature algorithm name: SHA256withRSA
Subject Public Key Algorithm: 2048-bit RSA key
Version: 3
```


Configurar el certificado de firma Fingerprint


1.- Abrir el [AppGallery Connect](#) y click en My Projects

2.- Ir a [Project settings > General information](#). En la parte de [App information](#) click en el icono:  junto a la etiqueta [SHA-256 certificate fingerprint](#) y pegar el certificado obtenido

App information

Add SDK

Download the latest configuration file (you may need to apply the latest configuration file to your app if you have modified your project or app information).

 agconnect-services.json

Package name:


com.huawei.hms.framework

App ID:

102537808


API key:


api-key-102537808




App secret:

secret-102537808



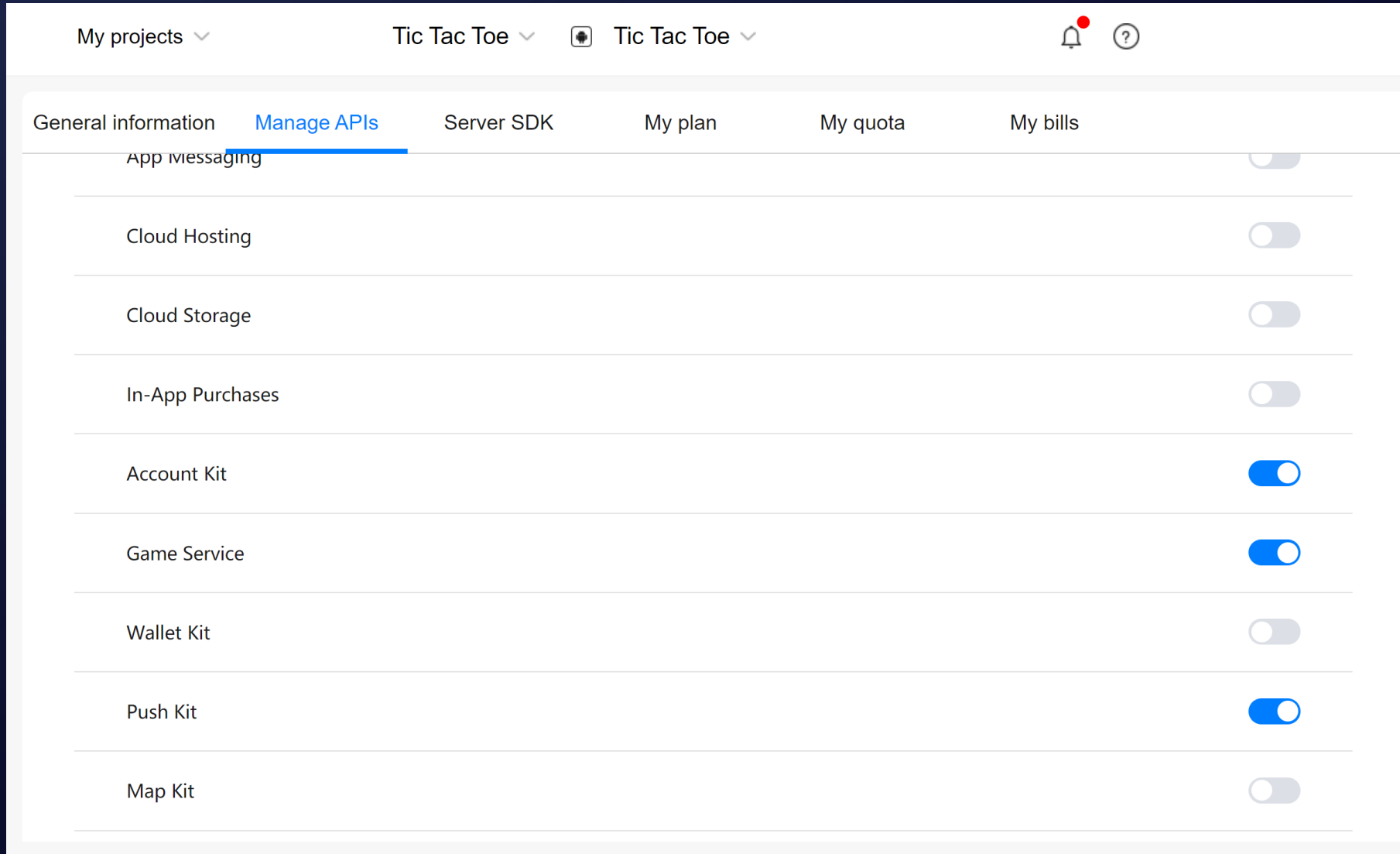
SHA-256 certificate fingerprint: 

Enter an SHA-256 certificate fingerprint.



Delete app

Habilitar los servicios requeridos



The screenshot shows the Google Cloud console interface for a project named "Tic Tac Toe". The "Manage APIs" tab is selected, displaying a list of services with their respective toggle switches. The services listed are:

Service	Status
App Messaging	Disabled
Cloud Hosting	Disabled
Cloud Storage	Disabled
In-App Purchases	Disabled
Account Kit	Enabled
Game Service	Enabled
Wallet Kit	Disabled
Push Kit	Enabled
Map Kit	Disabled

Configurar Achievements (Logros)

1.- Abrir el [AppGalleryConnect](#) y click en My Apps

2.- Ir a la aplicación y seleccionar la pestaña [Operate, Achievements > Create](#)

						Review records	Obtain resources	Create
No.	Name	ID	Earners	Status	Operation			
1	Gana una partida	9310BC3AFB2163D1768B E6186EE185BFA82506...	0	Testable	View Edit Reset progress Delete			
2	Gana tres partidas	B111821C14386C08CD66 4F66E8E3738BC9AC8B...	0	Testable	View Edit Reset progress Delete			
3	Gana cinco partidas	F641E28683780090698D5 6B13BE3BC6F5DF246F...	0	Testable	View Edit Reset progress Delete			
4	Gana 5 partidas seguidas	98FF61FBBF2B246002EE 87DF33452C6C97BF18...	0	Testable	View Edit Reset progress Delete			

Descargar y compilar código base

1.-Descargar el código base del repositorio:

https://github.com/jordanrsas/HMS_TicTacToe

2.-Abrir el archivo `build.gradle` del directorio `app` del proyecto Android Studio y agregar los `signingConfigs` de acuerdo al certificado de firma creado.

3.- Agregar el tag `signingConfig` a los `buildTypes` del proyecto

```
6  android {
7      compileSdkVersion 30
8      buildToolsVersion "30.0.2"
9
10     defaultConfig {...}
11
12     signingConfigs {
13         release {
14             storeFile file('./HmsDemos.jks')
15             keyAlias 'hmsheyscodelab'
16             keyPassword 'keyPassword'
17             storePassword 'storePassword'
18         }
19     }
20
21     buildTypes {
22         release {
23             signingConfig signingConfigs.release
24             minifyEnabled false
25             proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'
26         }
27         debug {
28             signingConfig signingConfigs.release
29         }
30     }
31
32     compileOptions {...}
33     kotlinOptions {jvmTarget = '1.8'}
34 }
```

Descargar y compilar código base

4.-Agregar los permisos de internet en el archivo AndroidManifest.xml

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

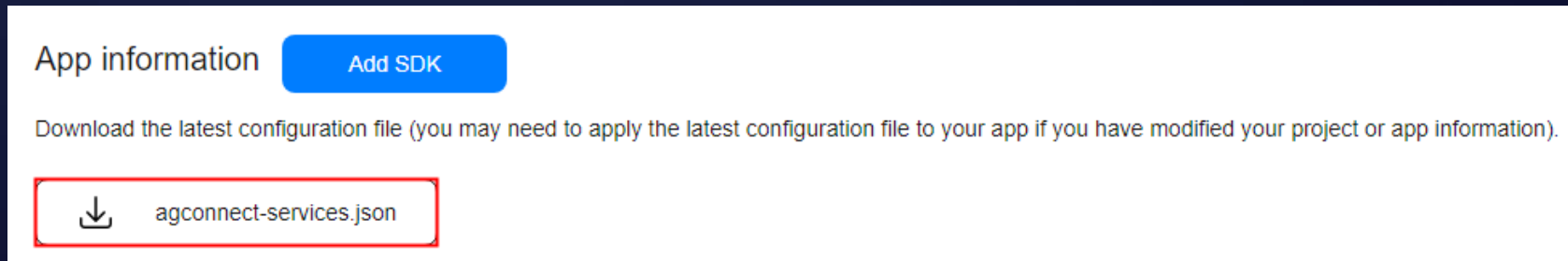
```
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
```

5.-Compilar e instalar la aplicación



Integrar HMS Core SDK

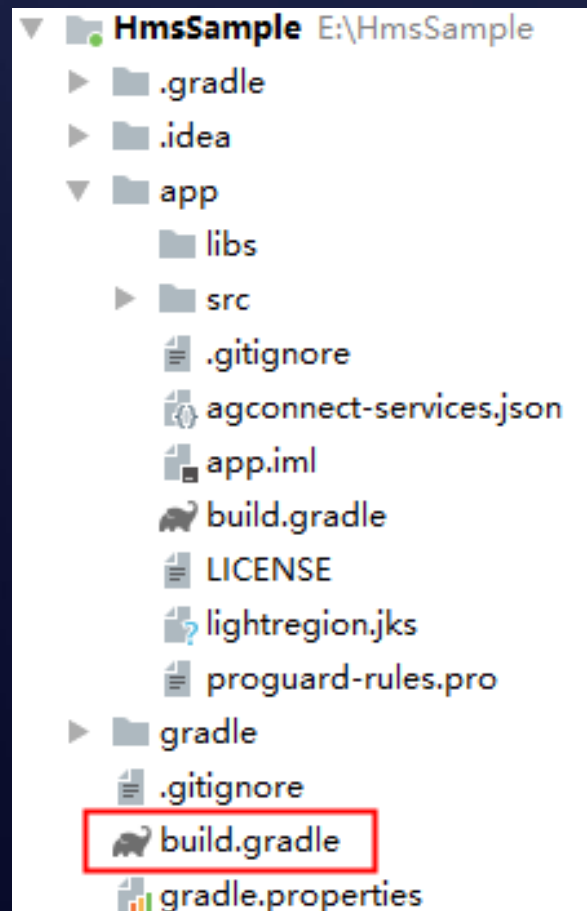
- 1.- Abrir el [AppGalleryConnect](#) y click en My projects
- 2.- Buscar y seleccionar la aplicación
- 3.- Ir a [Project settings > General information](#). En el área de [App information](#) descargar el archivo [agconnect-services.json](#).



- 2.- Copiar el archive [agconnect-services.json](#) en la app del proyecto de Android Studio

Configurar la dirección del Repositorio Maven para el HMS Core SDK

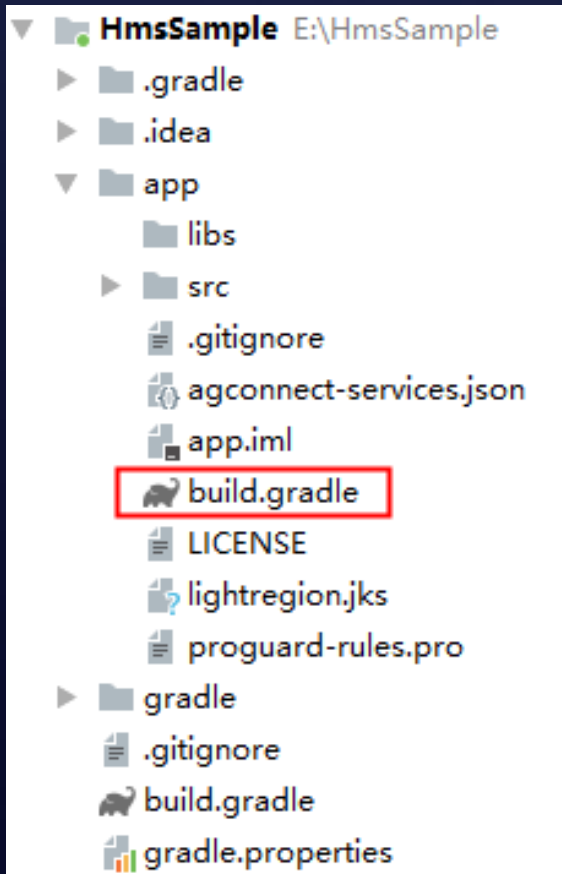
- 1.- Abrir el archivo `build.gradle` del directorio raíz del proyecto Android Studio
- 2.- Agregar el `AppGallery Connect` plug-in y el repositorio `Maven`



```
1
2  buildscript {
3      ext.kotlin_version = "1.3.72"
4      repositories {
5          google()
6          jcenter()
7          maven {url 'https://developer.huawei.com/repo/'}
8      }
9  }
10 dependencies {
11     classpath 'com.huawei.agconnect:agcp:1.4.1.300'
12 }
13
14
15
16 allprojects {
17     repositories {
18         google()
19         jcenter()
20         maven {url 'https://developer.huawei.com/repo/'}
21     }
22 }
23
24 task clean(type: Delete) {
25     delete rootProject.buildDir
26 }
```

Agregar las dependencias

- 1.- Abrir el archivo `build.gradle` del directorio `app` del proyecto Android Studio
- 2.- Agregar las build dependencies en la sección dependencies



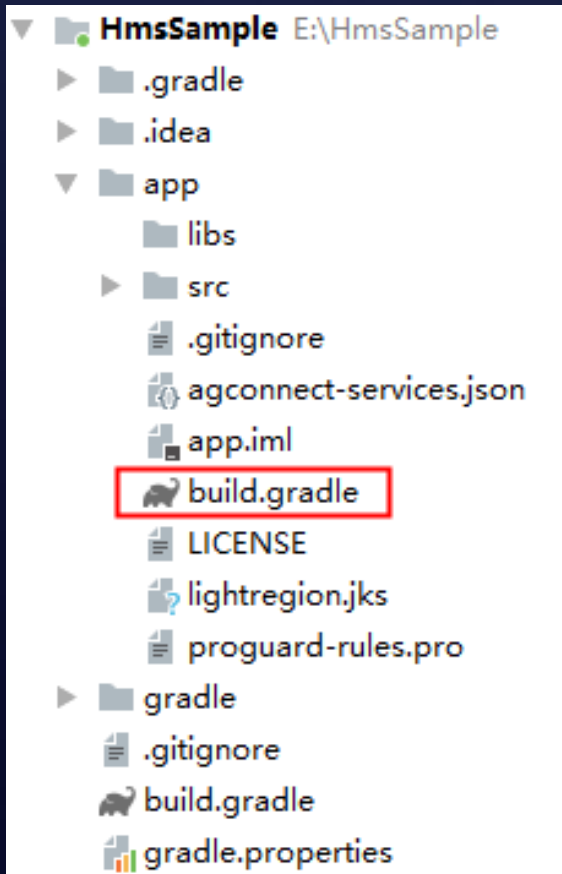
```
dependencies {  
    implementation 'com.huawei.hms:hwid:{version}'  
    implementation 'com.huawei.hms:ibase:{version}'  
    implementation 'com.huawei.hms:game:{version}'  
}
```

- 3.- Agregar `apply plugin: 'com.android.application'` en el header de archivo

```
apply plugin: 'com.huawei.agconnect'
```


Definir la configuración multi-lenguaje

- 1.- Abrir el archivo `build.gradle` del directorio `app` del proyecto Android Studio
- 2.- Ir a `android > defaultConfig`, y agregar la etiqueta `resConfigs`, configurar los idiomas como se muestra



```
android {  
    defaultConfig {  
        ...  
        resConfigs "en", "zh-rCN", "Other languages supported by your app"  
    }  
}
```

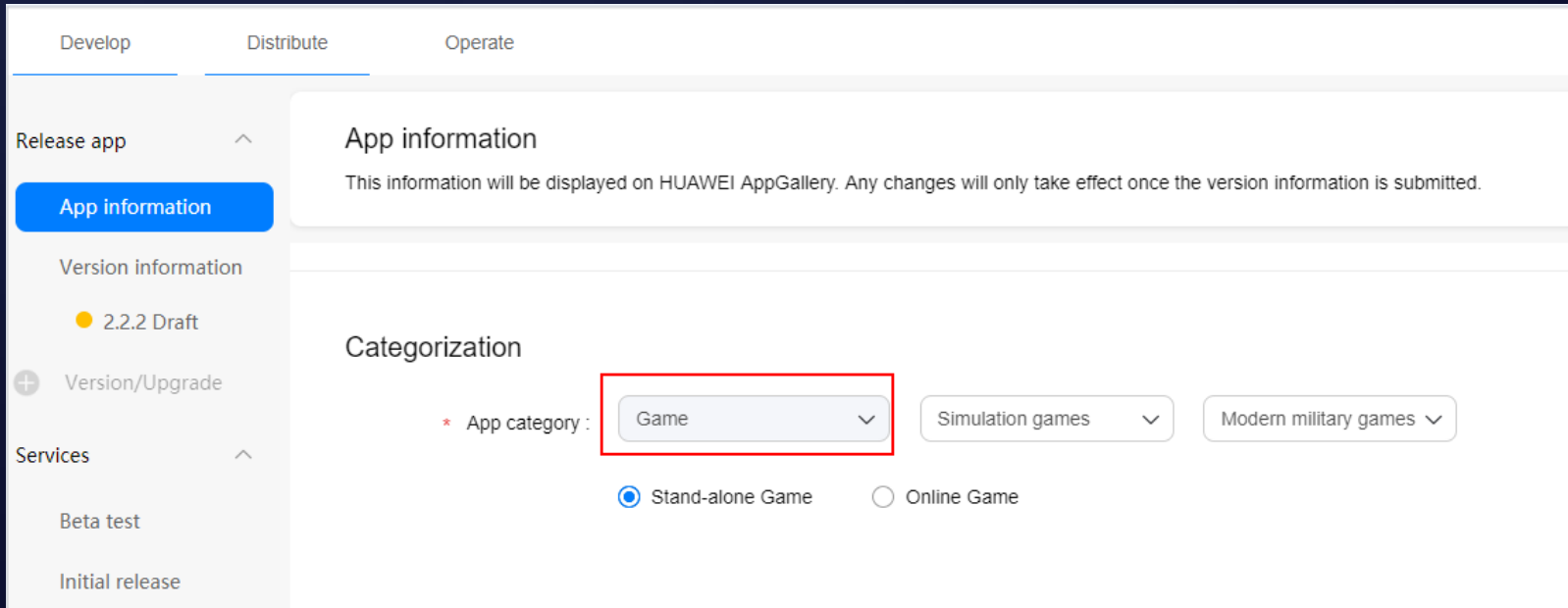
Por default la aplicación soportara todos los idiomas proporcionados por el HMS Core SDK. Si la aplicación usas todos estos idiomas, no es necesaria ninguna configuración.

Si la aplicación sólo es funcional únicamente con algunos idiomas, siga estas instrucciones.

Revisión de configuraciones

1.- Revisión de la categoría de la aplicación

- Login en el [AppGallery Connect](#) y dar click en [My apps](#).
- Click en tu app.
- Ir a [Distribute > Release app > App information](#). Asegurarse que la categoria del juego esta definida en [Game](#)



The screenshot displays the 'App information' page in the AppGallery Connect interface. The top navigation bar includes 'Develop', 'Distribute', and 'Operate' tabs. The left sidebar shows a 'Release app' section with a blue 'App information' button, and a 'Version information' section showing '2.2.2 Draft'. The main content area is titled 'App information' and includes a note: 'This information will be displayed on HUAWEI AppGallery. Any changes will only take effect once the version information is submitted.' Below this, the 'Categorization' section is visible. It features a red box around the 'App category' dropdown, which is currently set to 'Game'. To the right of this dropdown are two more dropdowns: 'Simulation games' and 'Modern military games'. At the bottom of the 'Categorization' section, there are two radio buttons: 'Stand-alone Game' (which is selected) and 'Online Game'.

Revisión de configuraciones


2.- Revisión de la configuración del certificado de firma

- Asegúrese de que se haya creado el certificado de firma para su aplicación y de que se haya generado la huella digital del certificado de firma correspondiente
- Asegúrese de que se hayan habilitado los servicios necesarios.
- Asegúrese de que el certificado de firma que se utiliza para ejecutar y compilar la aplicación sea coherente con la huella digital del certificado de firma que se ha configurado en [AppGallery Connect](#). De lo contrario, no podrá acceder a [HMS Core \(APK\)](#).

App information

Add SDK

Download the latest configuration file (you may need to apply the latest configuration file to your app if you have modified your project or app information).

 agconnect-services.json

Package name: com.huawei.hms2023.0801

App ID: 102537808

API key: [redacted]

App secret: [redacted]

SHA-256 certificate fingerprint: ✓

Delete app

My projects Tic Tac Toe Tic Tac Toe 🔔 ?

General information Manage APIs Server SDK My plan My quota My bills

App messaging ☐

Cloud Hosting ☐

Cloud Storage ☐

In-App Purchases ☐

Account Kit ☒

Game Service ☒

Wallet Kit ☐

Push Kit ☒

Map Kit ☐

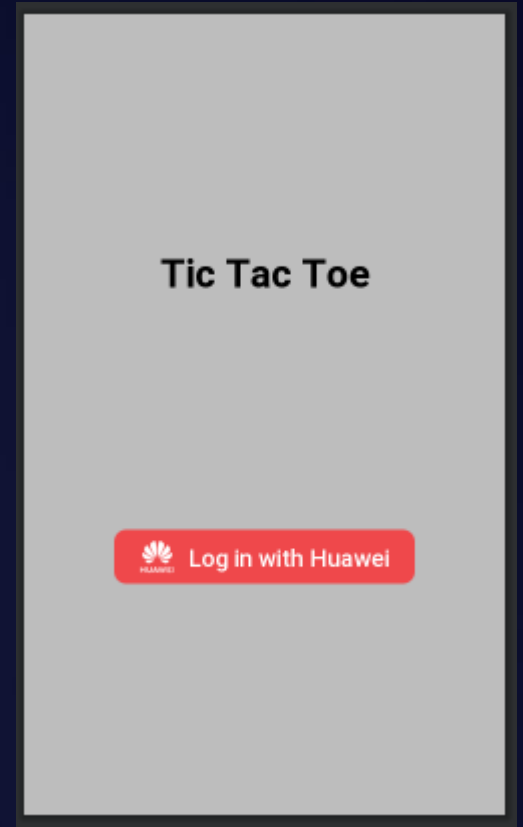


Código

Integrar SignIn mediante el botón Huawei

1.- Agregar un elemento `HuaweildAuthButton` al layout `fragment_first.xml`

```
<com.huawei.hms.support.hwid.ui.HuaweildAuthButton
    android:id="@+id/buttonHuaweiLogin"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    app:hwid_button_theme="hwid_button_theme_full_title"
    app:hwid_color_policy="hwid_color_policy_red"
    app:hwid_corner_radius="hwid_corner_radius_medium" />
```



2.- En la actividad principal `MainActivity.kt` crear el método `signIn()` e instanciar un objeto `HuaweildAuthParams`

```
val authParams: HuaweildAuthParams =
    HuaweildAuthParamsHelper(HuaweildAuthParams.DEFAULT_AUTH_REQUEST_PARAM)
        .setIdToken()
        .createParams()
```

Integrar SignIn mediante el botón Huawei

3.- Llamar el servicio de inicio de sesión mediante un `startActivityResult`

```
val service: HuaweiIdAuthService = HuaweiIdAuthManager.getService(activity, authParams)
startActivityResult(service.signInIntent, HMS_REQUEST_CODE)
```

2.- Cachar la respuesta sobrescribiendo el método `onActivityResult`

```
override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
    super.onActivityResult(requestCode, resultCode, data)
    if (requestCode == HMS_REQUEST_CODE) {
        val authHuaweiIdTask = HuaweiIdAuthManager.parseAuthResultFromIntent(data)
        if (authHuaweiIdTask.isSuccessful) {
            val huaweiAccount = authHuaweiIdTask.result
            Log.i(TAG, "idToken:${huaweiAccount.idToken}")
            findNavController().navigate(R.id.action_FirstFragment_to_SecondFragment)
        } else {
            authHuaweiIdTask.exception.printStackTrace(TAG, "SignIn failed")
        }
    }
}
```

Integrar Silent Sign In

1.- En la actividad principal `MainActivity.kt` crear el método `authSilentSignIn()` e Instanciar un objeto `HuaweildAuthParams`

```
val authParams =  
HuaweildAuthParamsHelper(HuaweildAuthParams.DEFAULT_AUTH_REQUEST_PARAM).createParams()
```

2.- Crear el servicio `HuaweildAuthService` agregando los Listeners de Success y Fail

```
val service : HuaweildAuthService = HuaweildAuthManager.getService(activity, authParams)  
service.silentSignIn().apply {  
    addOnSuccessListener { authHuaweild ->  
        Log.i(TAG, "User already logged, Display name: ${authHuaweild.displayName}")  
        findNavController().navigate(R.id.action_FirstFragment_to_SecondFragment)  
    }  
  
    addOnFailureListener { exception ->  
        exception.printLog(TAG, "Silent sign in fail")  
    }  
}
```

Integrar Sign Out

- 1.- En la actividad principal `MainActivity.kt` crear el método `signOut()`
- 2.- Usando la instancia del `HuaweildAuthService` invocamos el método `signOut()` y le agregamos los listeners `OnSuccess` y `OnFail` para cazar las respuestas

```
service?.signOut()?.apply {  
    addOnSuccessListener {  
        Log.i(TAG, "signOut Success")  
        nav_host_fragment.findNavController()  
            .navigate(R.id.action_SecondFragment_to_FirstFragment)  
    }  
  
    addOnFailureListener {  
        Log.i(TAG, "signOut fail")  
    }  
}
```


Lanzamiento del Juego

1.- En la clase `TicTacToeApplication` la cual hereda de `Application()` agregar el siguiente código al método `onCreate()` para registra la función de escucha devolución de llamada de la actividad

```
HuaweiMobileServicesUtil.setApplication(this)
```

Antes de la compilación, asegúrese de que el nombre de la clase de aplicación, por ejemplo:

```
android: name = ".TicTacToe"
```

se haya configurado en el nodo `application` en el `AndroidManifest.xml` de la aplicación.

2.- En la actividad principal llamar `JosApps.getJosAppsClient` para inicializar el objeto `JosAppsClient` y `JosAppsClient.init` para inicializar el juego.

```
private fun initializeGame() {  
    val appsClient: JosAppsClient = JosApps.getJosAppsClient(this)  
    appsClient.init()  
    Log.i(TAG, "init success")  
}
```

Obtener datos del jugador

La aplicación llama a `Games.getPlayersClient` para solicitar la inicialización de la instancia `PlayersClient` y llama al método `getCurrentPlayer()` para obtener información sobre el jugador actual.

```
val playersClient: PlayersClient = Games.getPlayersClient(this)
playersClient.currentPlayer.apply {
    addSuccessListener { player ->
        val playerId = player.playerId
        val playerName = player.displayName
    }

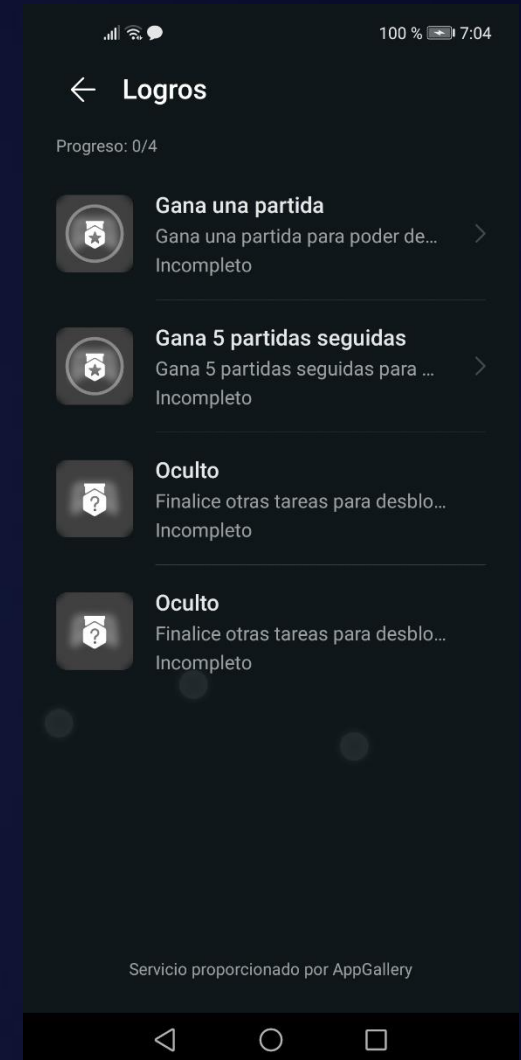
    addOnFailureListener { exception ->
        exception.printStackTrace(TAG, "getPlayerInfo failed")
    }
}
```

Mostrar lista de Logros mediante Huawei AppAssistant(Achievements)

- 1.-La aplicación llama a `Games.getAchievementClient` para solicitar la inicialización de la instancia `AchievementsClient` y llama al método `getShowAchievementListIntent()` para solicitar la pantalla de logros.
- 2.-HMS SDK devuelve de forma asincrónica el objeto `Intent` a la aplicación, que se utiliza para saltar a la página de logros.
- 3.- La aplicación debe llamar al método `startActivityForResult` (`Intent`, `int`) de la página para mostrar la página de logros a Huawei `AppAssistant`.
- 4.- Huawei `AppAssistant` muestra la página de logros del jugador.

Mostrar lista de Logros mediante Huawei AppAssistant

```
val achievementsClient: AchievementsClient =  
    Games.getAchievementsClient(this)  
  
achievementsClient.showAchievementListIntent.apply {  
    addOnSuccessListener { intent ->  
        intent?.let {  
            try {  
                startActivityForResult(it, 234)  
            } catch (ex: Exception) {  
                Log.e(TAG, "Achievement Activity is Invalid")  
            }  
        } ?: Log.w(TAG, "get achievements list intent = null")  
    }  
    addOnFailureListener { exception ->  
        Log.e(TAG, "get achievements list intent fail")  
    }  
}
```



Desbloquear un logro

1.-El jugador ha completado un Logro

2.-La aplicación invoca al método `reachWithResult` del `AchievementsClient` para solicitar desbloquear un Logro designado.

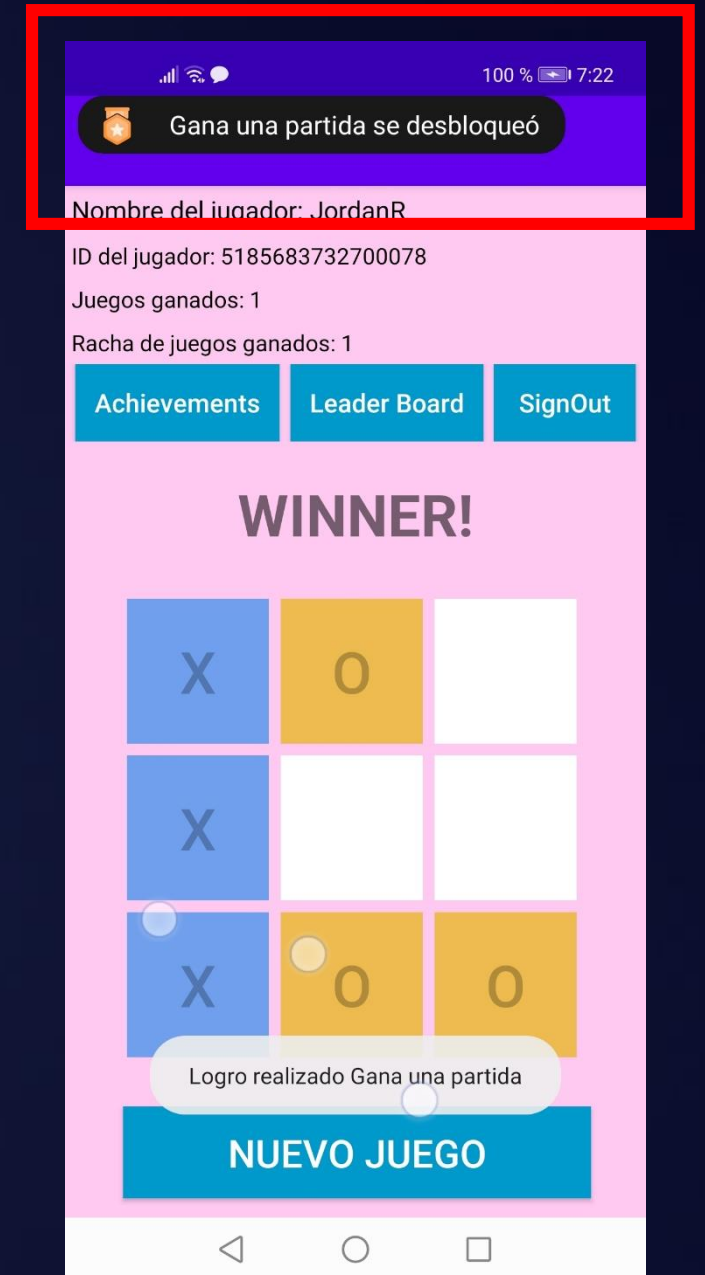
Si no hay conexión de red o la aplicación no necesita obtener el resultado de la operación de inmediato, puede adoptar el método `reach`.

*Cuando se utiliza el método `reach`, los datos no se pueden enviar al servidor de Huawei de inmediato. Por lo tanto, puede haber un retraso o desviación.

3.- Cuando se utiliza `reachWithResult`, HMS SDK devolverá el resultado de la operación de forma asíncrona.

Desbloquear un logro

```
val achievementsClient: AchievementsClient =  
    Games.getAchievementsClient(this)  
  
achievementsClient.reachWithResult(achievement.id).apply {  
    addOnSuccessListener {  
        Log.i(TAG, "Achievement reached, ID: ${achievement.id}")  
    }  
  
    addOnFailureListener { exception ->  
        Log.e(TAG, "Reach achievement fail")  
    }  
}
```

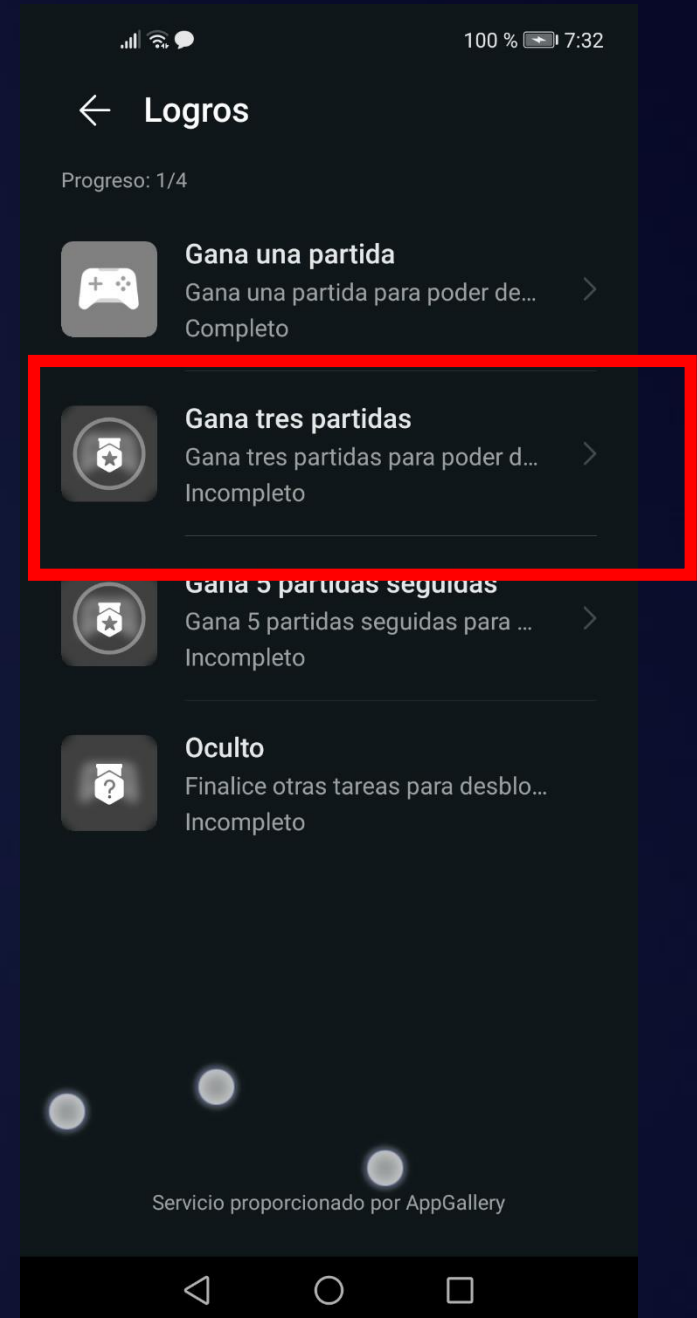


Revelar un logro oculto

1. El jugador activa un logro oculto al jugar.
2. La aplicación llama al método `visualize` o `visualizeWithResult` del `AchievementsClient` para solicitar que se revele el logro oculto.
3. Cuando se utiliza `visualizeWithResult`, HMS SDK devolverá el resultado de la operación de forma asincrónica.

```
val achievementsClient: AchievementsClient =  
    Games.getAchievementsClient(this)
```

```
achievementsClient.visualizeWithResult(achievementId).apply {  
    addSuccessListener {  
        Log.i(TAG, "Achievement with ID: $achievementId reveled")  
    }  
  
    addOnFailureListener { exception ->  
        Log.e(TAG, "Achievement reveled fail")  
    }  
}
```



Mostrar tabla de clasificación mediante Huawei AppAssistant

1. Cuando un jugador activa la exploración de la tabla de clasificación, el juego obtiene el `Intent` de la pantalla de lista de la tabla de clasificación de HUAWEI AppAssistant.

Para obtener el objeto `Intent` de todas las tablas de clasificación, llame al método `RankingsClient.getTotalRankingsIntent`.

Para obtener el objeto `Intent` de una tabla de clasificación específica en todos los marcos de tiempo, llame al método `RankingsClient.getRankingIntent` (String rankingId).

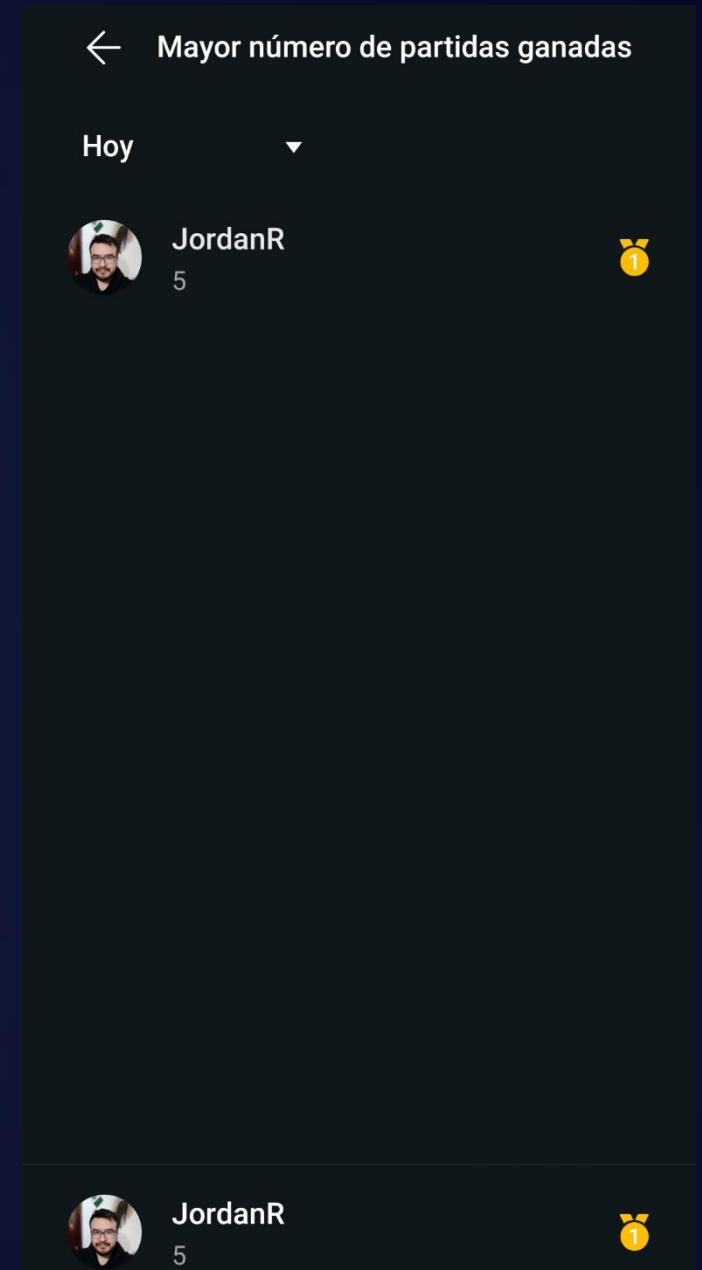
Para obtener el objeto `Intent` de una tabla de clasificación específica en un período de tiempo específico, llame al método `RankingsClient.getRankingIntent` (String rankingId, int timeDimension).

Mostrar tabla de clasificación mediante Huawei AppAssistant

2. Llame al método `startActivityForResult` para abrir la pantalla de la lista de clasificación de HUAWEI AppAssistant.

```
val rankingsClient: RankingClient = Games.getRankingsClient(this)
```

```
rankingsClient.totalRankingsIntent.apply {  
    addOnSuccessListener { intent ->  
        intent?.let {  
            try {  
                startActivityForResult(it, 134)  
            } catch (ex: Exception) {  
                Log.e(TAG, "Ranking Activity is Invalid")  
            }  
        }  
    }  
}?: Log.w(TAG, "get ranking client activity = null")  
}
```



Enviar puntuación de clasificación

Para enviar síncronamente la puntuación del jugador actual sin una unidad personalizada usamos el método `submitScoreWithResult`

Este método envía solicitudes de forma sincrónica. Es decir, las solicitudes se envían inmediatamente al servidor de juegos de Huawei después de que un cliente de la aplicación llama a este método y el resultado de la ejecución se devuelve inmediatamente.

```
rankingsClient.submitScoreWithResult(LEADER_BOARD_ID, score.toLong()).apply {  
    onSuccessListener {  
        Log.i(TAG, "Success submit score, player Id: ${it.playerId}, ranking Id: ${it.rankingId}" )  
    }  
  
    onFailureListener { exception ->  
        Log.e(TAG, "Sumit player score fail")  
    }  
}
```

Obtener la puntuación de clasificación del jugador actual

Obtiene la puntuación de un jugador en una tabla de clasificación específica en un período de tiempo específico usamos el método `getCurrentPlayerRankingScore(String rankingId, int timeDimension)`

```
rankingsClient.getCurrentPlayerRankingScore(LEADER_BOARD_ID, TIME_FRAME).apply {  
    addOnSuccessListener { rankingScore ->  
        rankingScore?.let {  
            Log.i(TAG, "Actual player ranking: ${it.displayRank}")  
        } ?: Log.w(TAG, "ranking score null")  
    }  
  
    addOnFailureListener { exception ->  
        Log.e(TAG, "Get current player ranking score")  
    }  
}
```

TIME_FRAME

Periodo de tiempo de la tabla de clasificacion

0 – Diario

1 – Semanal

3 – Todo el tiempo

An abstract graphic on the left side of the slide. It features a series of concentric circles and arcs, some of which are filled with a pattern of small, repeating chevron shapes. In the center of these circles is a large, stylized arrow pointing to the right, composed of several overlapping, slightly offset chevron shapes. The entire graphic is rendered in shades of teal and light blue against a dark blue background.

Preguntas y respuestas

An abstract graphic on the left side of the slide. It features a series of concentric circles and arcs, some of which are filled with a pattern of small, repeating chevron shapes. In the center of this graphic is a large, stylized arrow pointing to the right, composed of several overlapping, semi-transparent layers of the same chevron pattern. The entire graphic is rendered in shades of teal and light blue against the dark blue background.

Gracias