Universidad Nacional de La Matanza

Remote Config

Intro

- Durante la vida de las aplicaciones, es posible que necesitemos subir un cambio inmediatamente a los usuarios (ej: crash, adaptarse a nueva legislación, etc)
- La naturaleza de las apps mobile hace que sea imposible actualizarlas de forma instantánea
- Sumado también a que muchos usuarios tal vez nunca lleguen a tener la última versión de nuestra app
- Por lo tanto necesitamos de algún mecanismo que nos permita **modificar el comportamiento dinámicamente**, sin necesidad de actualizar.

Intro

Firebase

Remote Config

Casos de Uso

- AB Testing (rediseños, reimplementaciones, bug fixing, etc)
- Lanzamientos progresivos (soft lanch, hard launch)
- Habilitar/deshabilitar una funcionalidad -> Feature flags!

¿Cómo funciona?

1 - La aplicación solicita la configuración a los servidores de Firebase

2 - Firebase evalúa una serie de **condiciones** y devuelve la configuración adecuada

3 - La aplicación utiliza la configuración

* *Si se produce un error que impide obtener la configuración del servidor de Firebase, el dispositivo cuenta con una configuración local de backup*

Parámetros y Condiciones

- Las configuraciones de Firebase (“*parámetros*”) vienen en forma de pares de **clave-valor**
  Ej: [“color”=”gris”], [“timeout”=10], [“sonidos”=false]
- Para determinar qué valor de configuración debe retornar, Firebase evalúa una serie de condiciones (Ej: si es Android o iOS, versión de la aplicación, % de usuarios, etc)
- La primera condición que se cumpla, determina el valor del parámetro
- Las condiciones pueden referenciar User Properties y Eventos registrados por Firebase Analytics

Parámetros y Condiciones

Integración

Integración

- Integrar Firebase modificando los archivos build.gradle y agregando el google-services.json, de acuerdo a lo visto en clases anteriores

Integración

En el archivo build.gradle a **nivel app**

```
dependencies {

    // Dependencia de Firebase
    implementation platform('com.google.firebase:firebase-bom:27.1.0')

    // Dependencia de Remote Config
    implementation 'com.google.firebase:firebase-config-ktx'
    ...

    // Otras dependencias
    ...
}
```

En el código accedemos a Firebase Remote Config de la siguiente manera:

```
Firebase.remoteConfig
```

Integración

Tenemos 3 pasos:

1 - Asignar *settings* a Firebase Remote Config

2 - Asignar valores por defecto para las configuraciones (en caso de error, timeout, etc.)

3 - Buscar las configuraciones remotas

4 - Usar las configuraciones

Integración: 1 - Asignar settings a Firebase Remote Config

`minimumFetchIntervalInSeconds`: Es el tiempo que tiene que transcurrir entre actualizaciones de la configuración remota (*"cada cuanto mi app puede ir a buscar la configuración a Firebase"*)

`fetchTimeoutInSeconds`: Es el tiempo que nuestra aplicación va a esperar a que Firebase le envíe una configuración

```
val settings = remoteConfigSettings {
    minimumFetchIntervalInSeconds = 3600 // 3600 = 1 hora, por default es 12 horas !!!
    fetchTimeoutInSeconds = 10
}
Firebase.remoteConfig.setConfigSettingsAsync(settings)
```

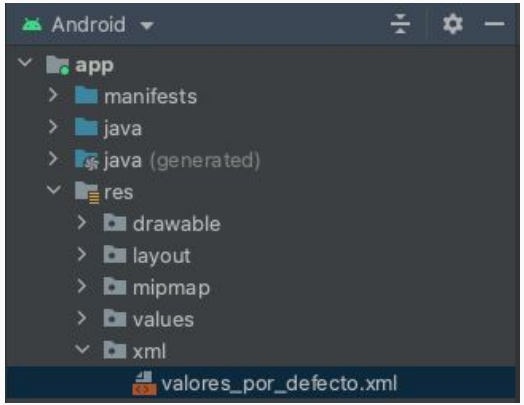
Integración: 2 - Asignar valores por defecto para las configuraciones

Tenemos 2 formas:

- Por **código**:

```
val defaults = mapOf(
    "color" to "rojo",
    "tiempo" to 123,
    "nueva_feature" to false
)
Firebase.remoteConfig.setDefaultsAsync(defaults)
```

- Por **XML**: Creamos un archivo dentro de app/res/xml/[*nombre_archivo_defaults*].xml



valores_por_defecto.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<defaultsMap>
    <entry>
        <key>nueva_feature</key>
        <value>false</value>
    </entry>
    ...
</defaultsMap>
```

```
Firebase.remoteConfig.setDefaultsAsync(R.xml.valores_por_defecto)
```

Integración: 3 - Buscar las configuraciones remotas

La forma de buscar las configuraciones es a través del método *fetchAndActivate()*:

```
Firebase.remoteConfig.fetchAndActivate()
    .addOnCompleteListener { task ->
        // Podemos validar si se completo OK llamando a task.isSuccessful
    }
```

Siempre se recomienda que este código se llame lo más temprano posible en nuestra aplicación !!

Integración: 4 - Usar las configuraciones

Utilizo las funciones get...() para obtener y usar las configuraciones

```
val color = Firebase.remoteConfig.getString("color")
val tiempo = Firebase.remoteConfig.getLong("tiempo")
val nuevaFeatureHabilitada = Firebase.remoteConfig.getBoolean("nueva_feature")
```

Bibliografia

- Documentación Firebase Remote Config

¿Preguntas?