|  |  |
| --- | --- |
| Una imagen de una carretera de curvas con árboles  Trabajo 02  Servicios cognitivos | Descripción breve  El presente documento explica la creación de una aplicación Android que permite obtener información de una imagen proporcionada por el usuario usando inteligencia artificial (servicios cognitivos).  Elias Ruiz González  Introducción al desarrollo de aplicaciones móviles para Android |

Contenido

[1. Introducción 4](#_Toc49733874)

[2. Requerimientos 4](#_Toc49733875)

[3. Análisis y Diseño 5](#_Toc49733876)

[3.1. Casos de uso 5](#_Toc49733877)

[3.1.1. Principal 6](#_Toc49733878)

[3.1.2. Tomar foto 6](#_Toc49733879)

[3.1.3. Seleccionar imagen 6](#_Toc49733880)

[3.1.4. Ver imagen 6](#_Toc49733881)

[3.1.5. Login 6](#_Toc49733882)

[3.1.6. Analizar 6](#_Toc49733883)

[3.1.7. Almacenar imagen 6](#_Toc49733884)

[3.1.8. Consultar imagen 6](#_Toc49733885)

[3.1.9. Almacenar información 7](#_Toc49733886)

[3.1.10. Ver últimas fotos 7](#_Toc49733887)

[3.2. Arquitectura 7](#_Toc49733888)

[3.3. Diseño 7](#_Toc49733889)

[3.3.1. Aplicación 7](#_Toc49733890)

[3.3.2. Servicio de login 8](#_Toc49733891)

[3.3.3. Servicio de almacenamiento de imagen 8](#_Toc49733892)

[3.3.4. Servicio de consulta cognitiva de imagen 8](#_Toc49733893)

[3.3.5. Servicio de almacenamiento de información 8](#_Toc49733894)

[3.4. Sobre FireBase y Azure 8](#_Toc49733895)

[3.4.1. Evolución de desarrollo de aplicaciones 8](#_Toc49733896)

[3.4.2. Desarrollo tradicional de aplicaciones web 8](#_Toc49733897)

[3.4.3. Desarrollo intermedio de aplicaciones web/móvil 9](#_Toc49733898)

[3.4.4. Desarrollo actual de aplicaciones web/móvil 9](#_Toc49733899)

[3.4.5. Sobre Firebase 11](#_Toc49733900)

[4. Implementación 11](#_Toc49733901)

[4.1. Diagrama de Flujo de aplicación 11](#_Toc49733902)

[4.1.1. Diagrama de flujo de actividad principal 11](#_Toc49733903)

[4.1.2. Interfaz de usuario de Actividad Principal 12](#_Toc49733904)

[4.1.3. Diagrama de flujo de actividad Analizar 13](#_Toc49733905)

[4.1.4. Interfaz de usuario de Actividad Analizar 14](#_Toc49733906)

[4.2. Inicialización 14](#_Toc49733907)

[4.2.1. Creación de proyecto vacío 14](#_Toc49733908)

[4.2.2. Adición de librería de materiales 14](#_Toc49733909)

[4.2.3. Ajuste de teclado virtual 14](#_Toc49733910)

[4.2.4. Métodos de log no funcionan correctamente 15](#_Toc49733911)

[4.2.5. Errores de librería de compatibilidad 15](#_Toc49733912)

[4.2.6. ContraintLayout vs LinearLayout 16](#_Toc49733913)

[4.2.7. Sobre el menú de navegación 16](#_Toc49733914)

[4.2.8. Iconos 18](#_Toc49733915)

[4.3. Clases auxiliares 19](#_Toc49733916)

[4.3.1. Estructura general de una clase 19](#_Toc49733917)

[4.3.2. Clase de lógica de negocio 19](#_Toc49733918)

[4.4. Conversión 19](#_Toc49733919)

[4.5. Control de versiones 20](#_Toc49733920)

[4.5.1. Selección de plataforma de control de versiones 20](#_Toc49733921)

[4.5.2. Configuración del control de versiones 21](#_Toc49733922)

[4.6. Tomar fotos desde cámara 21](#_Toc49733923)

[4.6.1. Configurar permisos 21](#_Toc49733924)

[4.6.2. Metodos de acceso a la camara 22](#_Toc49733925)

[4.6.3. Uso de la cámara 22](#_Toc49733926)

[4.6.4. Ajuste de tamaño de imagen a vista 22](#_Toc49733927)

[4.7. Uso de fragmentos sin interfaz de usuario. TODO 22](#_Toc49733928)

[4.8. Ayuda en visibilidad de código 23](#_Toc49733929)

[4.9. Firebase 23](#_Toc49733930)

[4.9.1. Creación de cuenta 23](#_Toc49733931)

[4.9.2. Acceso a la consola 23](#_Toc49733932)

[4.9.3. Configuración de Firebase en proyecto 23](#_Toc49733933)

[4.9.4. Base de datos en Firebase (Realtime Database) 27](#_Toc49733934)

[4.9.5. Almacenamiento en Firebase (Storage) 31](#_Toc49733935)

[4.9.6. Subir archivo en Storage de Firebase 34](#_Toc49733936)

[4.9.7. Configurar autenticación en Firebase (login) 35](#_Toc49733937)

[4.9.8. Lectura de información de base de datos de FireBase (RealTime Database) 35](#_Toc49733938)

[4.10. Barra de progreso 38](#_Toc49733939)

[4.11. Servicios Cognitivos de Azure 39](#_Toc49733940)

[4.11.1. Creación de cuenta Azure 39](#_Toc49733941)

[4.11.2. Activación de servicios cognitivos en Azure 39](#_Toc49733942)

[4.11.3. APIs disponibles para “Computer Vision” y “Face” 40](#_Toc49733943)

[4.11.4. Claves y punto de conexión 41](#_Toc49733944)

[4.11.5. Prueba de servicios 42](#_Toc49733945)

[4.11.6. Computer Vision - Análisis de imágenes disponibles: selección 43](#_Toc49733946)

[4.11.7. Face – Análisis de rostros disponibles: selección 44](#_Toc49733947)

[4.11.8. Esquema de llamadas a APIs de Cognitive Services de Azure 45](#_Toc49733948)

[4.12. Selección de librería para acceso a APIs REST de servicios cognitivos 45](#_Toc49733949)

[4.12.1. Selección de librería 45](#_Toc49733950)

[4.12.2. Código de ejemplo 45](#_Toc49733951)

[4.13. Sobre ReciclerView y el patrón viewHolder 46](#_Toc49733952)

[4.13.1. Diversidad de implementaciones 46](#_Toc49733953)

[4.13.2. Implementación realizada 47](#_Toc49733954)

[4.13.3. Empleo de cardview 47](#_Toc49733955)

[4.14. Imágenes de aplicación en funcionamiento 49](#_Toc49733956)

[5. Bibliografía 50](#_Toc49733957)

Trabajo 02

# Introducción

El presente documento explica los pasos seguidos desde la identificación de requerimientos hasta la implementación, de una aplicación que permite demostrar los conocimientos adquiridos en las lecciones avanzadas del curso “Introducción al desarrollo de aplicaciones móviles para Android”.

En particular se centra en los siguientes conocimientos:

* Selección de una imagen almacenada localmente
* Subir y recuperar imagen a un servidor de archivos
* Consultar información de imagen a servidor cognitivo
* Subir información cognitiva a base de datos en tiempo real
* Recuperar ultimas búsquedas de base de datos en tiempo real
* Visualizar ultimas búsquedas

De manera adicional se incluirán los siguientes conocimientos:

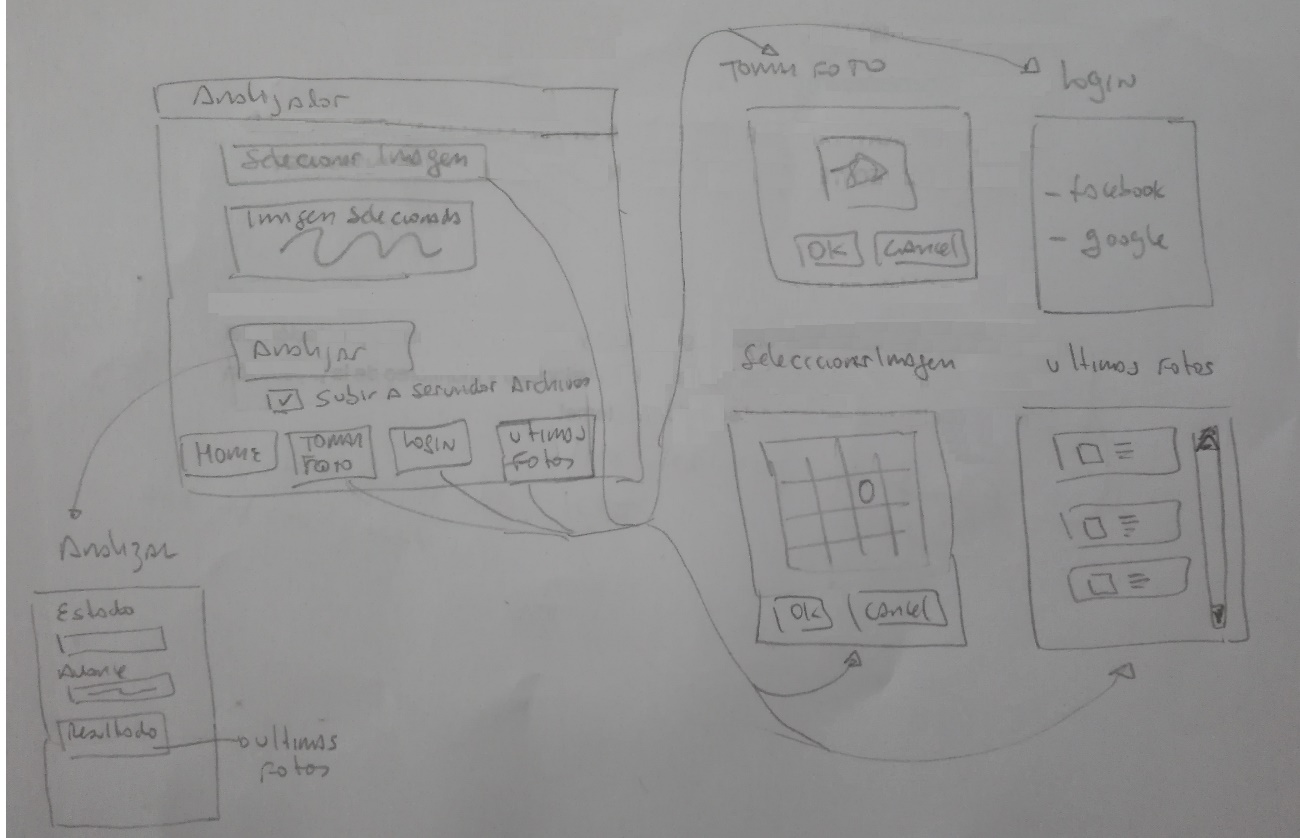
* Captura y selección de una imagen desde la cámara del dispositivo
* Manejo de login por redes sociales
* Manejo de menús de aplicación
* Manejo de logs y analítica
* Inclusión del nombre del autor en la visualización de las últimas búsquedas
* Empleo de archivo .xml para almacenar los mensajes propios de la interfaz de usuario

Y se considerarán los conocimientos ya adquiridos:

* Manejo de más de una actividad (“activity”)
* Incremento de complejidad de la lógica básica de aplicación
* Control de versiones/publicación de contenido usando GitHub

# Requerimientos

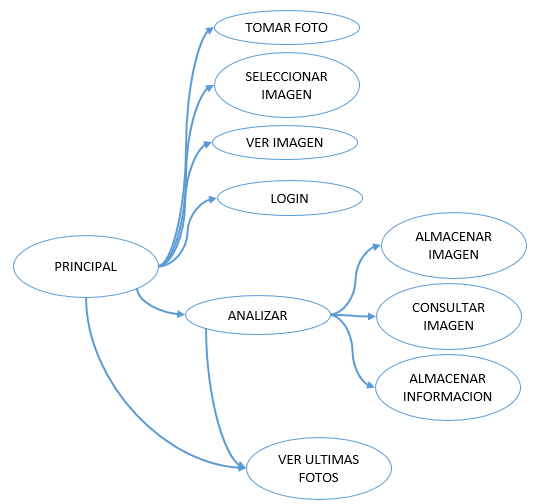
1. Se desea elaborar una aplicación Android que permita al usuario obtener información de una imagen que seleccione.
2. La imagen puede seleccionarse del sistema de archivos del dispositivo o capturarse usando la cámara del mismo
3. La imagen debe almacenarse en un servidor de archivos en la nube y recuperar un enlace a la misma.
4. Con dicha imagen se debe consultar a un servicio cognitivo y obtener información asociada.
5. Dicha información se almacena en una base de datos en tiempo real, la cual devuelve una lista de últimas búsquedas realizadas
6. El sistema realiza la visualización de dichas búsquedas.



# Análisis y Diseño

## Casos de uso

Los casos de uso identificados se muestran en el gráfico siguiente, y su descripción se realiza en los acápites subsiguientes.



### Principal

Muestra el formulario principal de la aplicación. Contiene el botón “Seleccionar Imagen”, la imagen seleccionada, el botón “Analizar” y la opción de “subir a servidor de archivos”.

También contiene el menú principal de la aplicación, el cual contiene, entre otros, enlaces a “Tomar Foto”, “Login” y “Ultimas fotos”.

Al presionar en cada botón u opción de menú, se activan los casos de uso (y formularios) asociados.

### Tomar foto

Abre el formulario que permite la captura de imágenes utilizando la cámara del dispositivo.

La imagen se almacena en la carpeta predeterminada de imágenes (usualmente la carpeta DCIM\).

### Seleccionar imagen

Abre el formulario de selección de imagen almacenada en el dispositivo.

Devuelve la ubicación de la misma.

### Ver imagen

Visualiza la imagen seleccionada por el usuario (en el formulario principal).

### Login

Es una opción de la aplicación.

Permite al usuario de la aplicación autenticarse utilizando redes sociales. Las credenciales obtenidas se almacenan internamente para ser proporcionadas en los procedimientos que sean necesarios.

### Analizar

Es el proceso principal de la aplicación.

El proceso llama en forma consecutiva a los casos de uso “Almacenar imagen”, “Consultar imagen” y “Almacenar información”, realizando las acciones respectivas.

### Almacenar imagen

Según la opción de “subir a servidor de archivos”, se encarga de almacenar la imagen seleccionada en el servidor de archivos apropiado.

### Consultar imagen

Utilizando la imagen seleccionada (dirección remota del servidor de archivos o imagen local) realiza la consulta al servicio cognitivo, espera el resultado y lo devuelve.

### Almacenar información

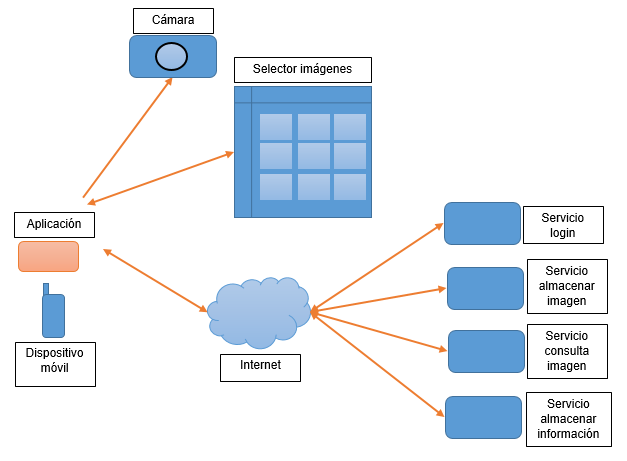
El resultado del servicio cognitivo es almacenado en una base de datos en tiempo real para su consulta posterior.

### Ver últimas fotos

Abre un formulario “Ultimas Fotos” que permite mostrar las últimas consultas realizadas. Dicha cantidad se configura en la aplicación.

El proceso realiza previamente la consulta a la base de datos en tiempo real para obtener la información necesaria.

## Arquitectura



El sistema es una aplicación que se ejecuta en un dispositivo móvil (programable) que tiene acceso a recursos internos (cámara y selector de imágenes) y dispone de una conexión a internet que permite conectarse a servicios externos (login, almacenamiento de imágenes, servicios cognitivos de consulta/análisis de imágenes y almacenamiento de información).

## Diseño

### Aplicación

La aplicación se ejecutará sobre el sistema operativo Android, y se desarrollará utilizando el entorno Android Studio.

El nombre de la aplicación es “Analizador” y su correspondiente paquete es “pe.pucp.analizador”.

Se ha elegido la mínima versión de los dispositivos Android con los que se cuentan en casa, la cual corresponde a Android 4.4.2 (Api 19 KitKat) del dispositivo Samsung Galaxy Tab de 7 pulgadas y modelo SM-t230nt.

El software Android Studio se utiliza en su versión 4.0.1 (build 24/jun/2020).

El lenguaje de programación es Java.

Se ha podido configurar el emulador para depuración (no se cuentan con privilegios para instalar el software HAXM), sin embargo la depuración es lenta. Se ha utilizado el dispositivo Pixel 3, con el Android 7.1.1 (API 25) armeabi-v7a que es la versión que el emulador permite cargar de manera exitosa.

### Servicio de login

Se utilizará el servicio Firebase Auth de Firebase. Se creará una cuenta personal para tal efecto.

### Servicio de almacenamiento de imagen

Se utilizará el servicio Firebase Storage de Firebase. Para tal efecto se utilizará la cuenta personal creada en un acápite anterior.

### Servicio de consulta cognitiva de imagen

Se utilizará el servicio de Computer visión de Microsoft Azure. Se utilizará la cuenta personal para tal efecto (ya se contaba con una cuenta activa).

### Servicio de almacenamiento de información

Se utilizará el servicio Realtime Database de FireBase. Para tal efecto se utilizará la cuenta personal creada en un acápite anterior.

## Sobre FireBase y Azure

### Evolución de desarrollo de aplicaciones

El actual escenario de desarrollo de aplicaciones web ha venido evolucionando desde sus inicios. En <https://www.youtube.com/watch?v=-ci7EwXaIJg> puede encontrarse una reseña útil para el presente trabajo, la cual se resume a continuación.

### Desarrollo tradicional de aplicaciones web

El desarrollo web tradicional utiliza lo que se denomina el “Old Stack”: HMTL, CSS, JQuery para programar la interfaz de usuario, PHP para programar páginas web y MySql en la base de datos.

Todos estos programas se instalaban en equipos ubicados en las mismas instalaciones de los usuarios, requiriendo recursos de hardware (servidores), personal especializado, sistemas auxiliares (refrigeración, instalaciones eléctricas trifásicas, subestaciones de energía, etc.)



### Desarrollo intermedio de aplicaciones web/móvil

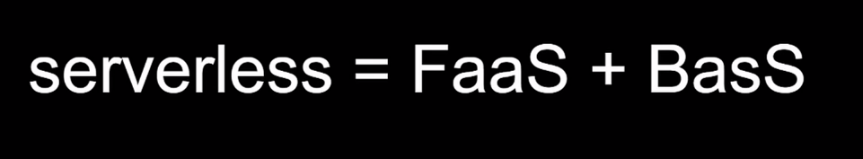
Pero los continuos desarrollos cambiaron la forma cómo se realiza el desarrollo de páginas web. Se iniciaron los procesos de virtualización del hardware o “cloud”, donde aparecieron empresas que compraban servidores de alta capacidad y vendían servicios de máquinas virtuales donde los clientes las podían crear y modificar según sus requerimientos en cuestión de minutos, eliminando la necesidad de contar con hardware y reduciéndolos costos asociados en forma significativa. En esta etapa se introduce el desarrollo para dispositivos móviles.

### Desarrollo actual de aplicaciones web/móvil

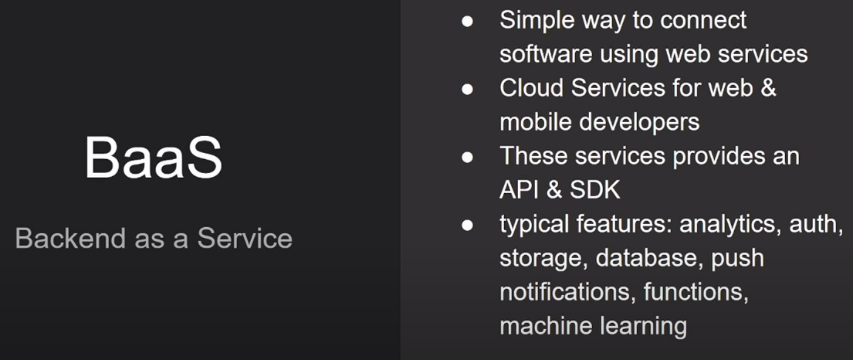
Los desarrollos han seguido evolucionando y actualmente se dispone de una virtualización de las mismas aplicaciones a través de servicios Serveless (sin servidor), donde empresas desarrollan servicios comunes a las aplicaciones (login, bases de datos, etc) y los ponen a disposición de desarrolladores para crear aplicaciones.

Estas aplicaciones ahora pueden ser desarrollas en forma separada de la interfaz de usuario (aplicaciones web, aplicaciones móviles) y de los procesos internos (que se ejecutan en el servidor).

Serverless se divide en dos aspectos: FaaS (Functions as a Service) y BaaS (Backend as a service).



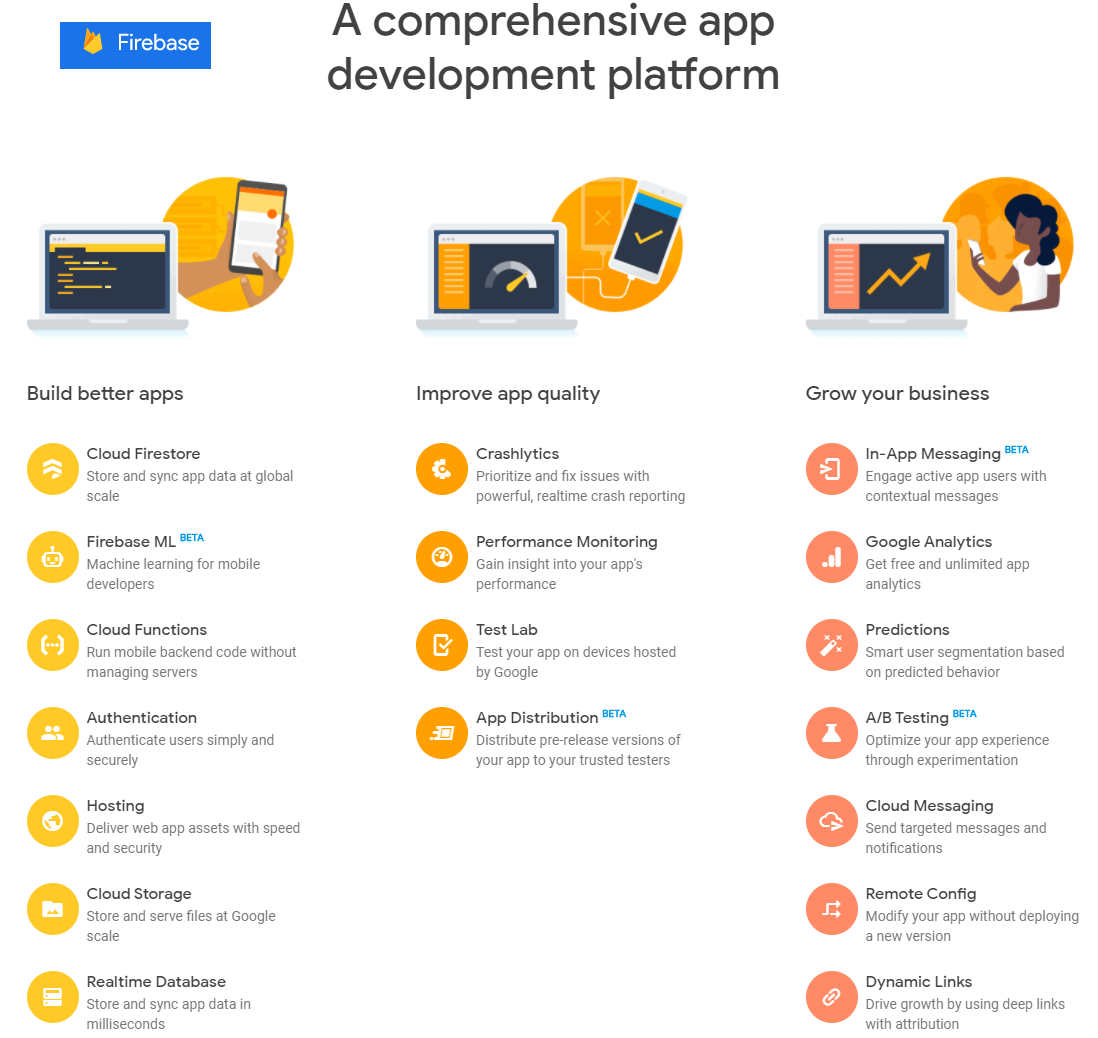


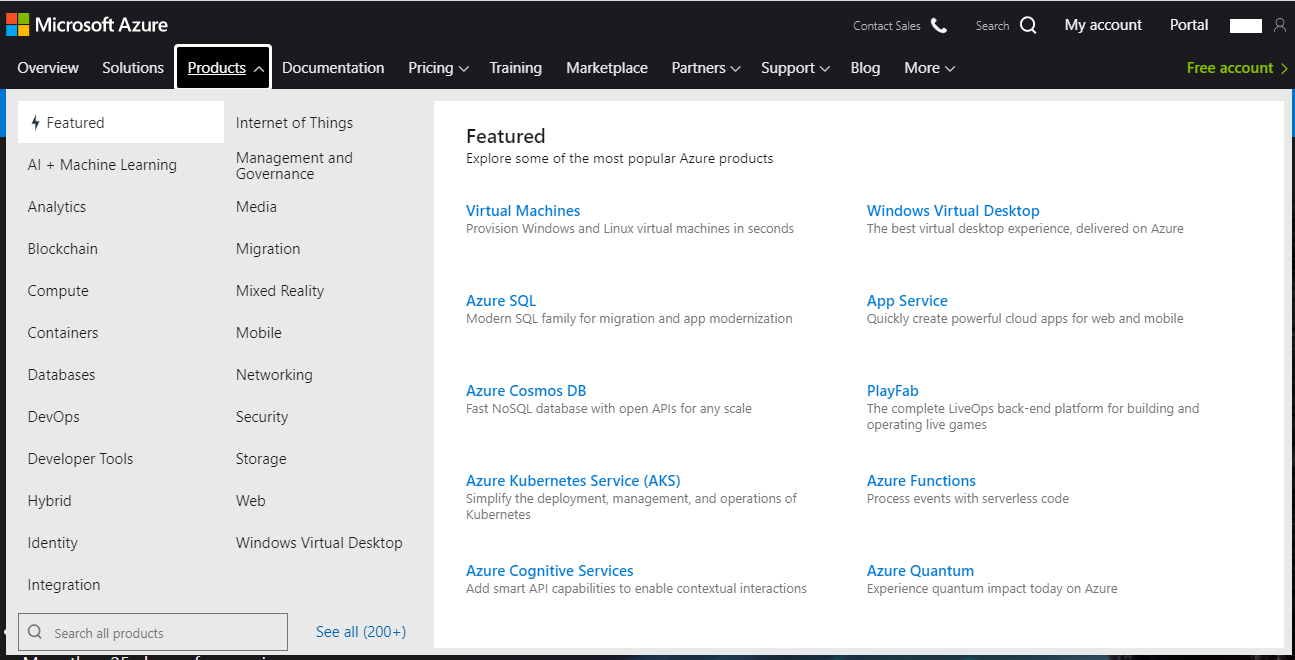


De esta forma el desarrollo actual de aplicaciones web (en mejora y evolución continuos pero apuntando al mismo objetivo: serverless) requiere que una aplicación deje de estar conformada por todos los componentes integrados en la misma aplicación, sino que se encuentren distribuidos en funciones o servicios especializados, tales como Cloudinary (imágenes), Auth0 (autenticación), etc. que el desarrollador ahora los utiliza a través de servicios web (mediante protocolos REST API) según los necesita, sin necesidad de conocer el detalle de cómo se realiza, permitiéndole enfocarse en el objetivo principal de su aplicación.



Precisamente, dos de estos proveedores son FireBase[[1]](#footnote-1) y Azure[[2]](#footnote-2), los cuales proporcionan múltiples servicios de software, de manera tal que el desarrollador los puede utilizar según lo requiera. De eta forma, en la presente aplicación se utilizan algunos de los servicios que proporcionan según las necesidades identificadas.





### Sobre Firebase

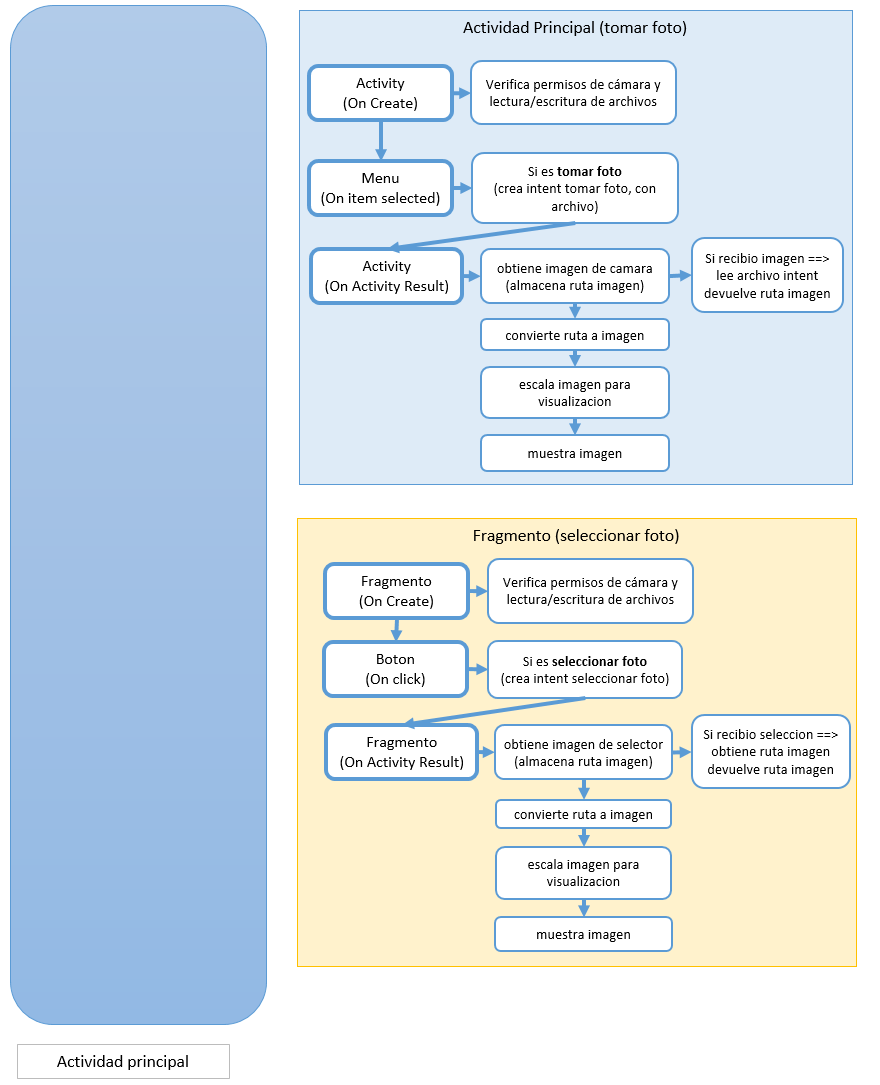
En (Kumar, 2018) y (Moroney, 2017) puede encontrarse información detallada sobre Firebase.

# Implementación

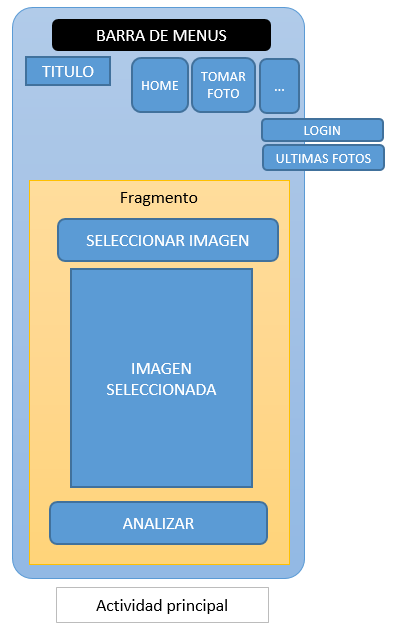
## Diagrama de Flujo de aplicación

### Diagrama de flujo de actividad principal

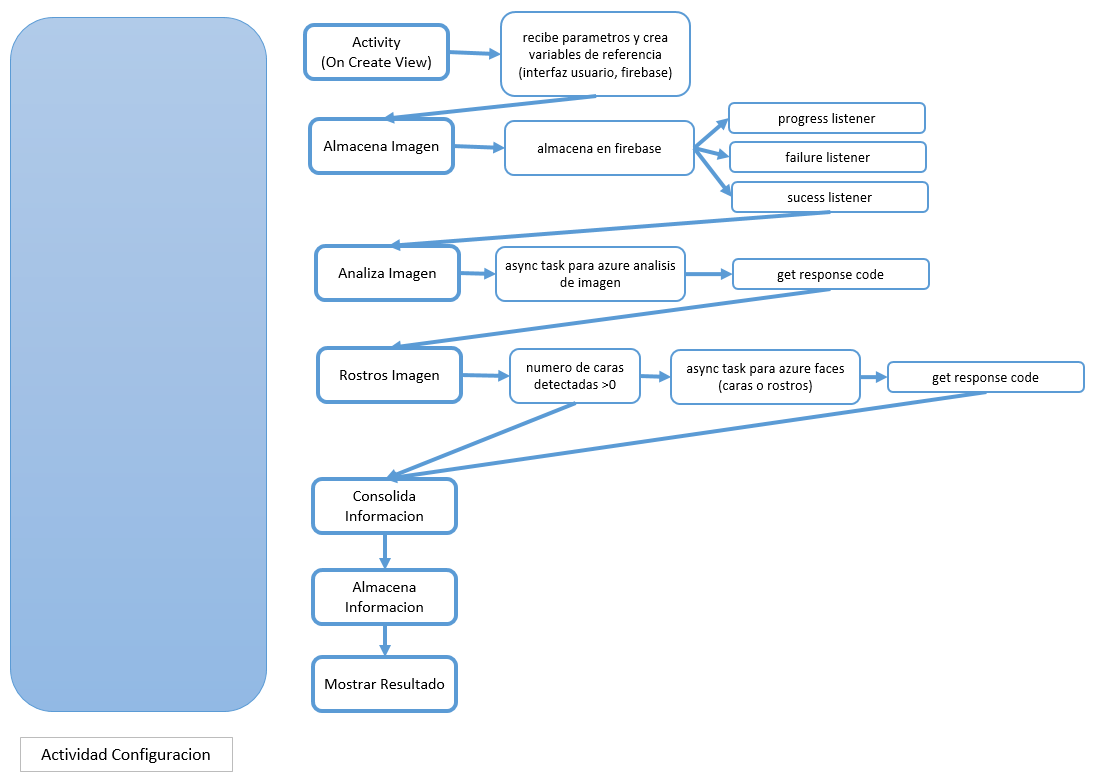
Es importante recordar que, en general, las librerías de webservices utilizan hilos de ejecución asíncronos, es decir siguen un flujo de ejecución alterno al de la actividad principal. Si bien este es el procedimiento establecido para las aplicaciones Android y así evitar que un proceso pueda detener a la aplicación e incluso todo el sistema, es necesario comprender su funcionamiento para evaluar la forma adecuada de hacer uso de este mecanismo. De esta forma, la aplicación inicia la ventana y, cuando se activan los webservices, se espera que la librería reciba los datos del servicio web en el evento “OnResponse” (según cada caso el nombre del evento puede cambiar). Es dentro de este evento que el flujo del programa continúa.



### Interfaz de usuario de Actividad Principal



### Diagrama de flujo de actividad Analizar



### Interfaz de usuario de Actividad Analizar



## Inicialización

### Creación de proyecto vacío

Crear el proyecto con actividad en blanco, estableciendo las opciones nombre de aplicación, proyecto y API.

### Adición de librería de materiales

Se modifica el archivo build.gradle del proyecto

dependencies {…

implementation **'com.google.android.material:material:1.1.0'**

…}

Se modifica archivo styles.xml

<**style name="AppTheme" parent="Theme.MaterialComponents.Light"**>  
 *<!-- Customize your theme here. -->* <**item name="colorPrimary"**>@color/colorPrimary</**item**>  
 <**item name="colorPrimaryDark"**>@color/colorPrimaryDark</**item**>  
 <**item name="colorAccent"**>@color/colorAccent</**item**>  
</**style**>

### Ajuste de teclado virtual

Se modifica el archivo AndroidManifest.xml

<**activity android:name=".MainActivity" android:windowSoftInputMode="adjustResize"**>

### Métodos de log no funcionan correctamente

En algunos casos los métodos de depuración (Log.d) no muestran información en el Android Studio.

Se ha solucionado utilizando la función “Log.wtf”

Log.*wtf*(**"url"**,url);

### Errores de librería de compatibilidad

Ejecutando el programa, sin incorporar aun el código subsiguiente, se obtiene un error de compatibilidad, tal como se aprecia a continuación:

…

2020-08-15 14:33:42.846 4808-4808/pe.pucp.analizador I/art: Rejecting re-init on previously-failed class java.lang.Class<androidx.core.view.ViewCompat$2>: java.lang.NoClassDefFoundError: Failed resolution of: Landroid/view/View$OnUnhandledKeyEventListener;

2020-08-15 14:33:42.846 4808-4808/pe.pucp.analizador I/art: at void androidx.core.view.ViewCompat.setBackground(android.view.View, android.graphics.drawable.Drawable) (ViewCompat.java:2559)

…

2020-08-15 14:33:42.847 4808-4808/pe.pucp.analizador I/art: at void androidx.appcompat.app.AppCompatActivity.setContentView(int) (AppCompatActivity.java:161)

2020-08-15 14:33:42.847 4808-4808/pe.pucp.analizador I/art: at void pe.pucp.convertisador.MainActivity.onCreate(android.os.Bundle) (MainActivity.java:47)

…

Donde la línea 47 del archivo MainActivity.java, coloreada en verde, se muestra a continuación:

…

*//oncreate se encarga de inicializar la base de datos*@Override  
**protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_main***);  
  
…

Este problema se debe a la librería de compatibilidad de Android, la cual se debe actualizar a una versión más reciente. En particular en “[https://stackoverflow.com/questions/51782548/androidxappcompat-iart-error-android-view-viewonunhandledkeyeventlistener/52954286#52954286](https://stackoverflow.com/questions/51782548/androidxappcompat-iart-error-android-view-viewonunhandledkeyeventlistener/52954286%2352954286)” se recomienda lo siguiente:

Reemplazar:

implementation **'androidx.appcompat:appcompat:1.1.0'**

por:

implementation **'androidx.appcompat:appcompat:1.3.0-alpha01'**

o:

implementation **'androidx.appcompat:appcompat:1.3.0-alpha02'**

Se observa en la ejecución que ya no aparece el error encontrado.

### ContraintLayout vs LinearLayout

Cuando se crea el proyecto inicial, el layout de la actividad principal (activity\_mail.xml) utiliza ContraintLayout, reemplazando el tradicional LinearLayout o RelativeLayout, para posicionar los elementos de la interfaz de usuario.

Según <https://stackoverflow.com/questions/37321448/differences-between-constraintlayout-and-relativelayout/37324703#37324703>, ConstraintLayout pretende corregir ciertos problemas presentados por LinearLayout y también acelerar la visualización de las actividades, para lo cual adiciona una librería adicional (aprox 150KBytes adicionales).

No obstante, las mediciones señaladas en la referencia permiten observar que la diferencia es muy pequeña (alrededor de 3 mseg sobre 120 mseg = 2.5%), además de la complejidad adicional de escritura y configuración de los xmls respectivos (por la combinación de diferentes opciones de layouts en uno solo).

En tal sentido, se mantendrá el estilo tradicional de layouts, utilizando LinearLayout/RelativeLayout.

### Sobre el menú de navegación

En la actualidad los menús de navegación tienen la forma señalada en <https://developer.android.com/guide/navigation/navigation-ui>, dentro de las cuales dicho menú de navegación se puede ubicar en la parte superior (top app bar), al costado en forma deslizante (navigation drawer) o en la parte inferior (bottom navigation).

Considerando que las imágenes tendrán un tamaño que se irá incrementando con el tiempo, la aplicación utilizará el menú ubicado en la parte superior (top app bar).

Por otro lado, la implementación del sistema de navegacion puede realizarse mediante “activities” o mediante “fragments”. En <https://stackoverflow.com/questions/32435389/navigationdrawer-with-activities-vs-navigationdrawer-with-fragments> puede encontrarse una discusión útil para decidir.

En tal sentido, internamente el sistema de menús se implementará mediante fragmentos (mejor performance y en un posible uso futuro sería fácilmente reutilizado en otras vistas).

Ejemplos de configuración de la aplicación puede encontrase en <https://developer.android.com/guide/navigation/navigation-getting-started>

De esta manera se adicionarán los siguientes elementos en el archivo gradle del proyecto:

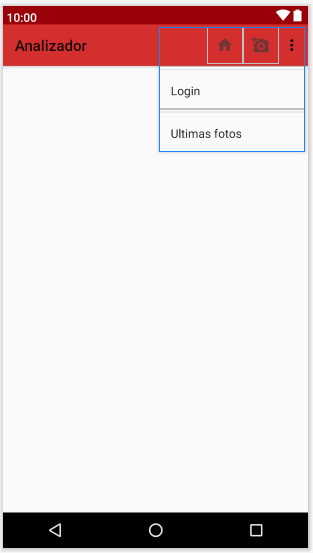
dependencies **{**…*//navigation* **def** nav\_version = **"2.3.0"** implementation **"androidx.navigation:navigation-fragment:**$nav\_version**"** implementation **"androidx.navigation:navigation-ui:**$nav\_version**"** *// navigation Feature module Support* implementation **"androidx.navigation:navigation-dynamic-features-fragment:**$nav\_version**"**…

*// navigation Testing Navigation* androidTestImplementation **"androidx.navigation:navigation-testing:**$nav\_version**"  
  
}**

Asimismo, los elementos del menu se definen utilizando textos, iconos y especificaciones funcionales que permiten mostrar una mejor interfaz de usuario. El método utilizado puede encontrarse en la página 467 y siguientes de (Meier & Lake, 2018).

*<?***xml version="1.0" encoding="utf-8"***?>*<**menu xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"**>  
 <**item  
 android:id="@+id/action\_home"  
 android:title="Home"  
 android:icon="@drawable/ic\_action\_home"  
 app:showAsAction="ifRoom|withText"**/>  
 <**item  
 android:id="@+id/action\_tomarfoto"  
 android:title="Tomar foto"  
 android:icon="@drawable/ic\_action\_tomarfoto"  
 app:showAsAction="ifRoom|withText"**/>  
 <**item  
 android:id="@+id/action\_login"  
 android:title="Login"  
 app:showAsAction="never"**/>  
 <**item  
 android:id="@+id/action\_ultimasfotos"  
 android:title="Ultimas fotos"  
 android:icon="@drawable/ic\_action\_ultimasfotos"  
 app:showAsAction="ifRoom|withText"**/>  
</**menu**>

Una imagen referencial de menú se muestra a continuación



### Iconos

Se ha configurado el icono de la aplicación según la siguiente figura



Asimismo se han creado iconos adicionales para los elementos del menú (ver carpeta drawables) tales como:

|  |  |  |
| --- | --- | --- |
|  |  |  |
| ic\_action\_home | ic\_action\_tomarfoto | ic\_action\_ultimasfotos |

El método utilizado puede encontrase en la página 467 y siguientes de (Meier & Lake, 2018)

## Clases auxiliares

### Estructura general de una clase

Las clases siguen la convención de nombre-archivo.

Asimismo la estructura de las clases incluirá secciones para propiedades, constructores (Java no utiliza destructores) y métodos, según la plantilla siguiente:

public Class Nombre {

//propiedades

…

//constructor

--constructor nativo

--constructores auxiliares

…

//métodos

…

}

### Clase de lógica de negocio

El Sistema utilizara una clase auxiliar “Cambiador” que encapsulará la lógica de conexión a la base de datos.

## Conversión

La estrategia utilizada en la implementación es utilizar objetos de tipo “TipoCambio” para todas las conversiones. Dichos objetos ya han sido llenados por las funciones respectivas.

En general existen dos formas de conversión, la primera se refiere a cálculos mediante expresiones complejas, como la conversión de temperatura de grados Celsius a Fahrenheit, y la segunda que se refiere a expresiones simples (lineales: y= a.x ) como la conversión de temperaturas, distancias y monedas.

De esta forma, el evento “onclick” del botón de conversión recupera el objeto “TipoCambio” seleccionado del spinner (*variable =* *(cast de tipo)spiner. getSelectedItem()*) y evalúa si se trata de expresiones complejas o simples (*switch – case-break-default*). El evento calcula en forma específica las primeras, en tanto que utiliza una forma común para las demás.

## Control de versiones

### Selección de plataforma de control de versiones

El proyecto utilizará el control de versiones utilizando herramientas libres. En particular se utiliza GitHub, considerando que el programador ya dispone de una cuenta en dicha plataforma.

### Configuración del control de versiones

Los pasos a seguir para utilizar el control de versión se basan en las indicaciones contenidas en <https://academiaandroid.com/uso-de-github-desde-android-studio/>, y se resumen a continuación:

1. Disponer una cuenta GitHub.
   1. Crear una cuenta gratuita o utilizar una ya existente.
   2. En este proyecto se ha utilizado la cuenta existente del programador.
2. Activar la cuenta de GitHub en Android Studio.
3. Activar el control de versiones usando GitHub en el proyecto específico en Android Studio.
4. Utilizar el control de versiones.

El proyecto se encuentra disponible en la siguiente url:

<https://github.com/ergio2000/Analizador>

Finalmente, recordar que, al momento de terminar parte del código operativo, realizar una acción commit (VCS/Commit), y luego el push respectivo (VCS/Git/Push).

## Tomar fotos desde cámara

El acceso a la cámara en Android es relativamente sencillo, y sus pasos se explican a continuación.

### Configurar permisos

En <https://stackoverflow.com/questions/14421694/taking-pictures-with-camera-on-android-programmatically> se puede observar los permisos necesarios a configurar en al archivo AndoirdManifest.xml

…

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

<uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE"/>

…

<application

…

Es interesante considerar que, a partir del API 23, según <https://stackoverflow.com/questions/43042725/revoked-permission-android-permission-camera>, el permiso de acceso a camara no es necesario. En tal sentido es necesario adicionar cierto código para detectar la versión de API existente.

### Metodos de acceso a la camara

Según <https://guides.codepath.com/android/Accessing-the-Camera-and-Stored-Media> existen dos formas de acceder a la cámara:

* Forma sencilla: utilizando un intent.
* Forma difícil: usando el API de la cámara e integrándolo en la aplicación.

Para nuestra aplicación utilizaremos la forma sencilla, es decir se utilizará un intent.

Una referencia de la evolución del api de google para el manejo de la cámara puede encontrarse en <https://www.youtube.com/watch?v=Bi4QjMfSOE0>

|  |  |
| --- | --- |
|  |  |
| Camera API | Camera API 2 |

### Uso de la cámara

Como referencia se puede consultar el api de la cámara en <https://developer.android.com/guide/topics/media/camera>, y ejemplos prácticos en <https://developer.android.com/training/camera/photobasics> y <https://stackoverrun.com/es/q/5467251>.

### Ajuste de tamaño de imagen a vista

Una vez seleccionada la imagen a visualizar, es necesario escalar la imagen a la vista que se desea mostrar. Un ejemplo se encuentra en <https://stackoverflow.com/questions/8232608/fit-image-into-imageview-keep-aspect-ratio-and-then-resize-imageview-to-image-d#:~:text=try%20adding%20android%3AscaleType%3D%22fitXY%22%20to%20your%20ImageView%20.&text=this%20can%20all%20be%20done,the%20size%20of%20the%20image.>.

## Uso de fragmentos sin interfaz de usuario. TODO

La captura de imágenes y de selección de imágenes son procedimientos que pueden ser utilizados en diversas partes de la misma aplicación. En tal sentido es razonable colocar el código asociado en otros módulos para facilitar su rehúso.

Según la forma de funcionamiento de la captura de imágenes y selección, internamente es necesaria una llamada mediante un “intent” y capturar el evento “onActivityResult”, en tal sentido es razonable que dicho código se encuentre dentro de un fragmento de la actividad llamante.

Una particularidad es que en estos caso, no se requiere mostrar nada al usuario porque el propio “intent” se encarga de realizar tal efecto. En tal sentido es razonable utilizar un fragmento sin interfaz de usuario.

Un ejemplo de cómo crear un fragmento sin interfaz de usuario se puede encontrar en <https://datacadamia.com/android/fragment>.

## Ayuda en visibilidad de código

Los IDEs proporcionan al programador herramientas para ayudarlo en su trabajo. Una de ellas es la agrupación de código de un mismo módulo en bloques que sean de fácil lectura.

En particular, Visual Studio incorpora la directiva “#region … #endregion” para delimitar zonas que pueden ser “colapsadas” y así simplificar la lectura de código.

En <https://stackoverflow.com/questions/2344524/java-equivalent-to-region-in-c-sharp> se ha encontrado que Android Studio implementa una funcionalidad similar utilizando la forma “//región … //endregion”.

## Firebase

### Creación de cuenta

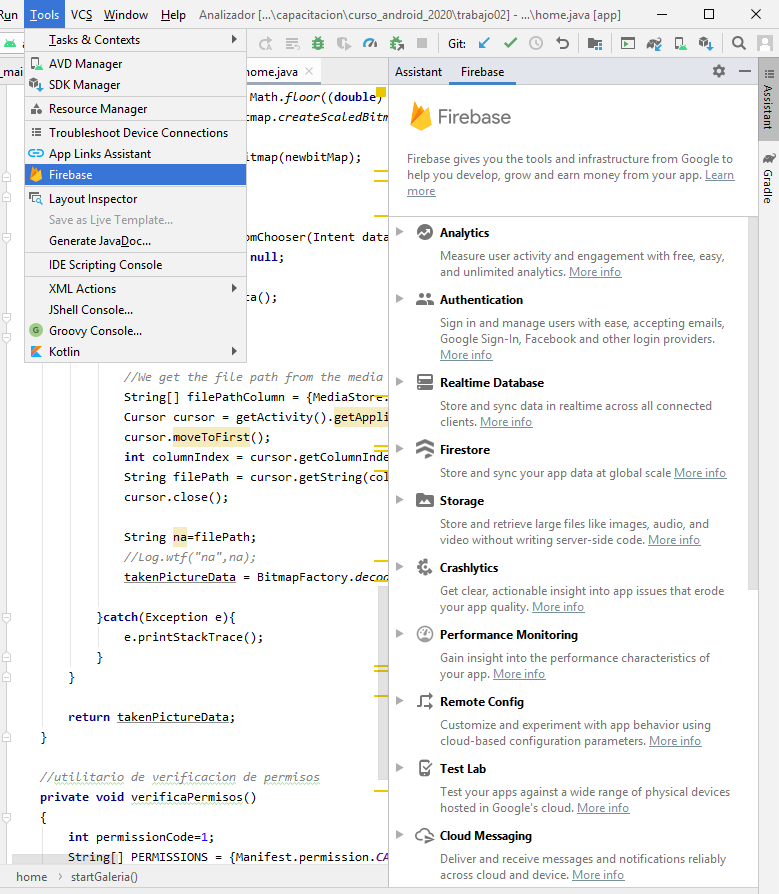
Desde el navegador Google Chrome, con una cuenta de Google ya existe, se ingresa a la página de firebase (<https://firebase.google.com/?hl=es>) y automáticamente se reconoce a dicha cuenta dentro del plan gratuito (Plan Spark).

### Acceso a la consola

Desde el navegador Google Chrome, dirigirse a la dirección <https://console.firebase.google.com/?hl=es>.

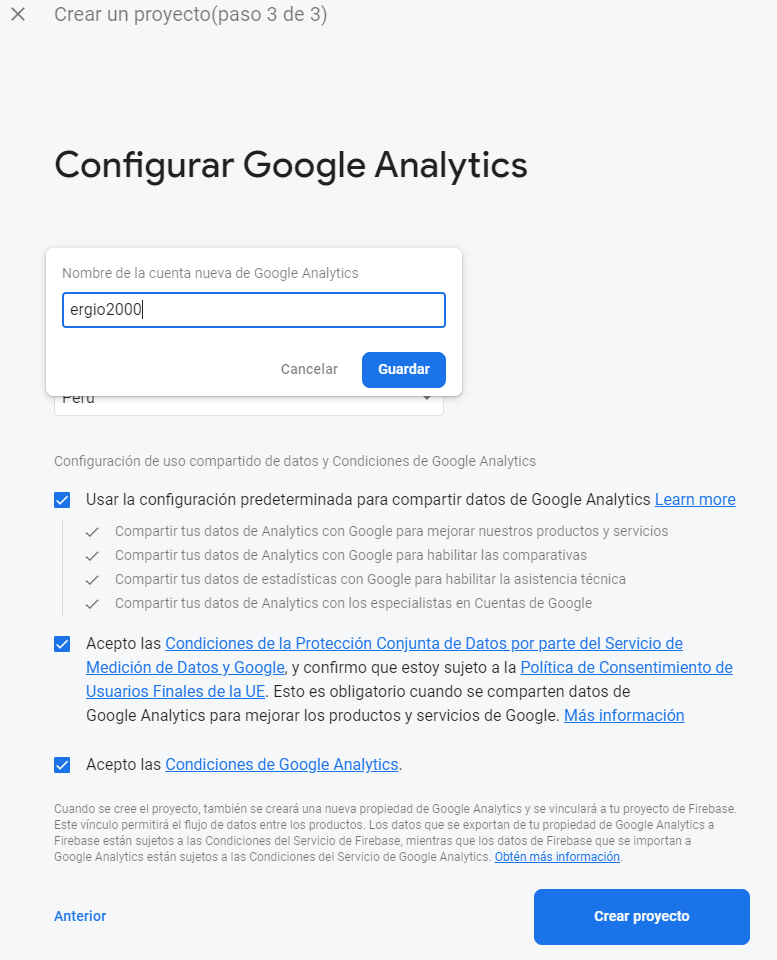
### Configuración de Firebase en proyecto

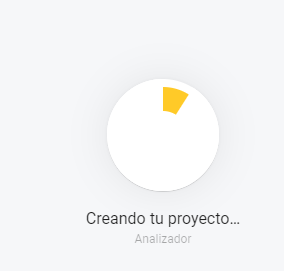
La instalación puede realizarse en dos formas: mediante la página web de Firebase, y mediante Android Studio. Para la presente aplicación se utilizará el segundo método. Como referencia puede observarse las pantallas asociadas siguientes.



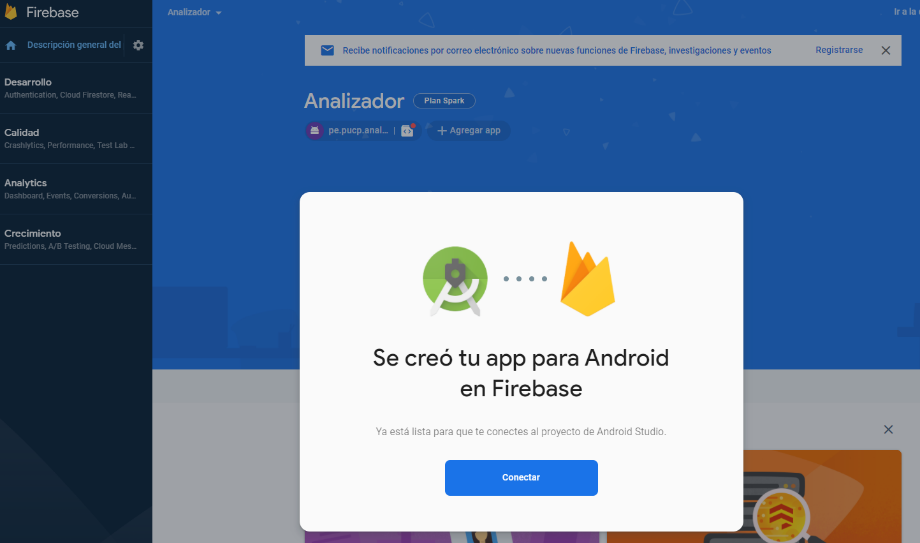


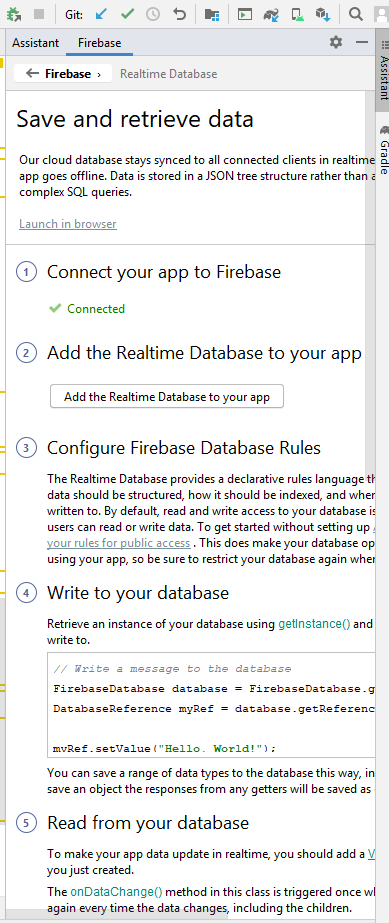




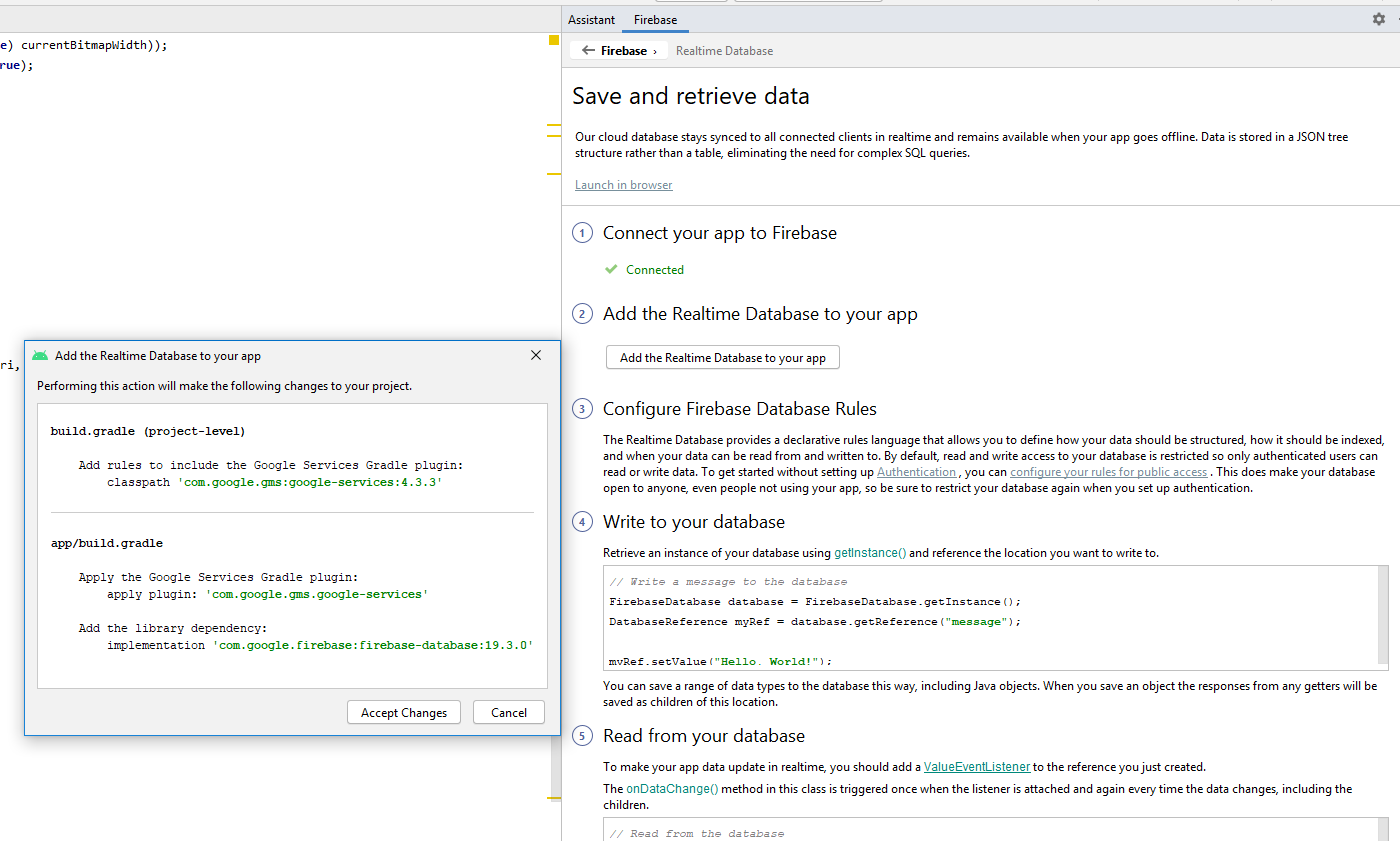


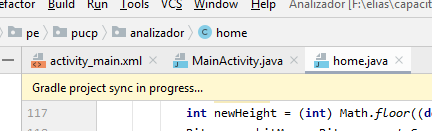




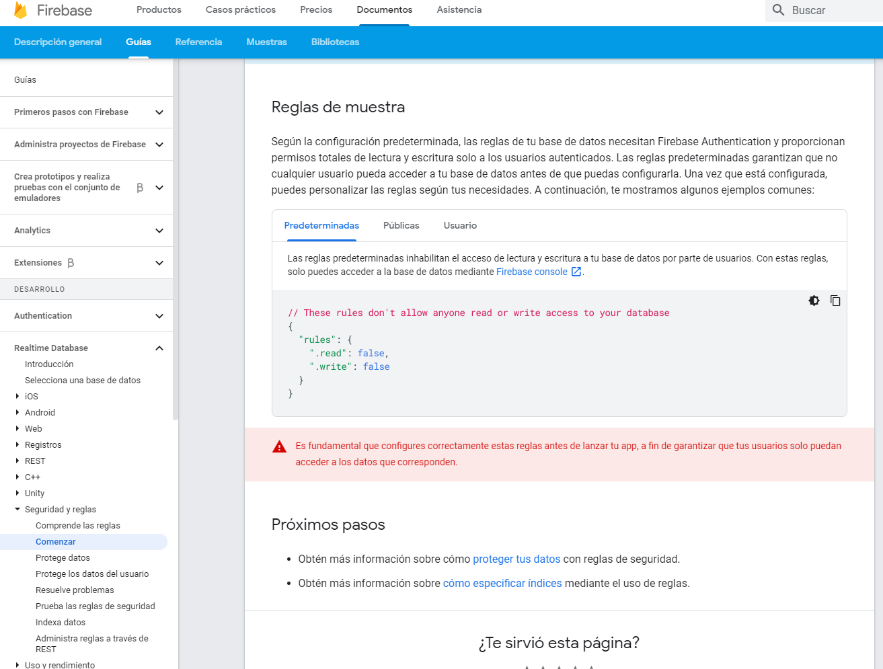


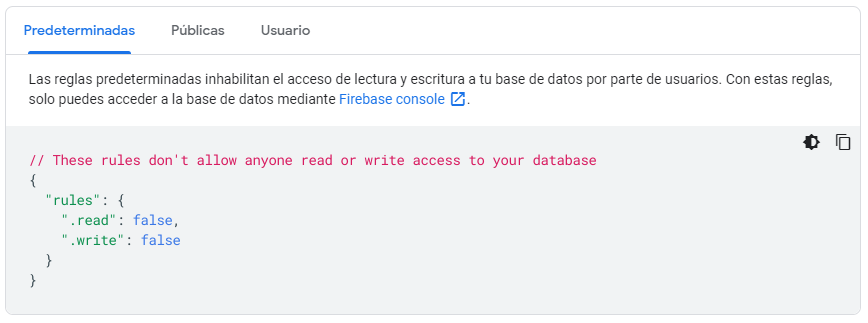
### Base de datos en Firebase (Realtime Database)

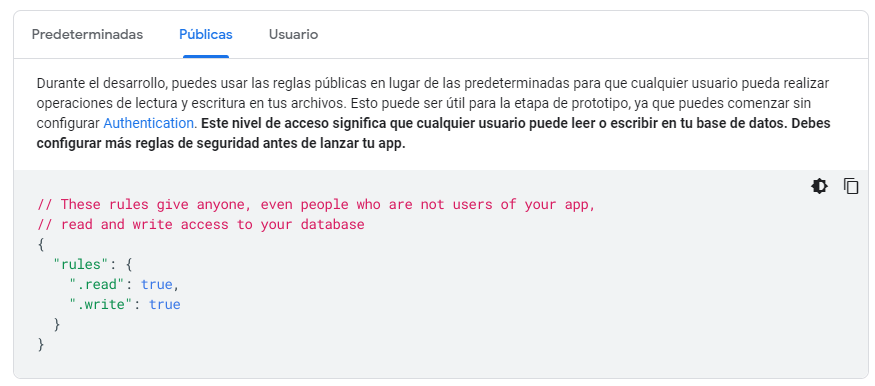




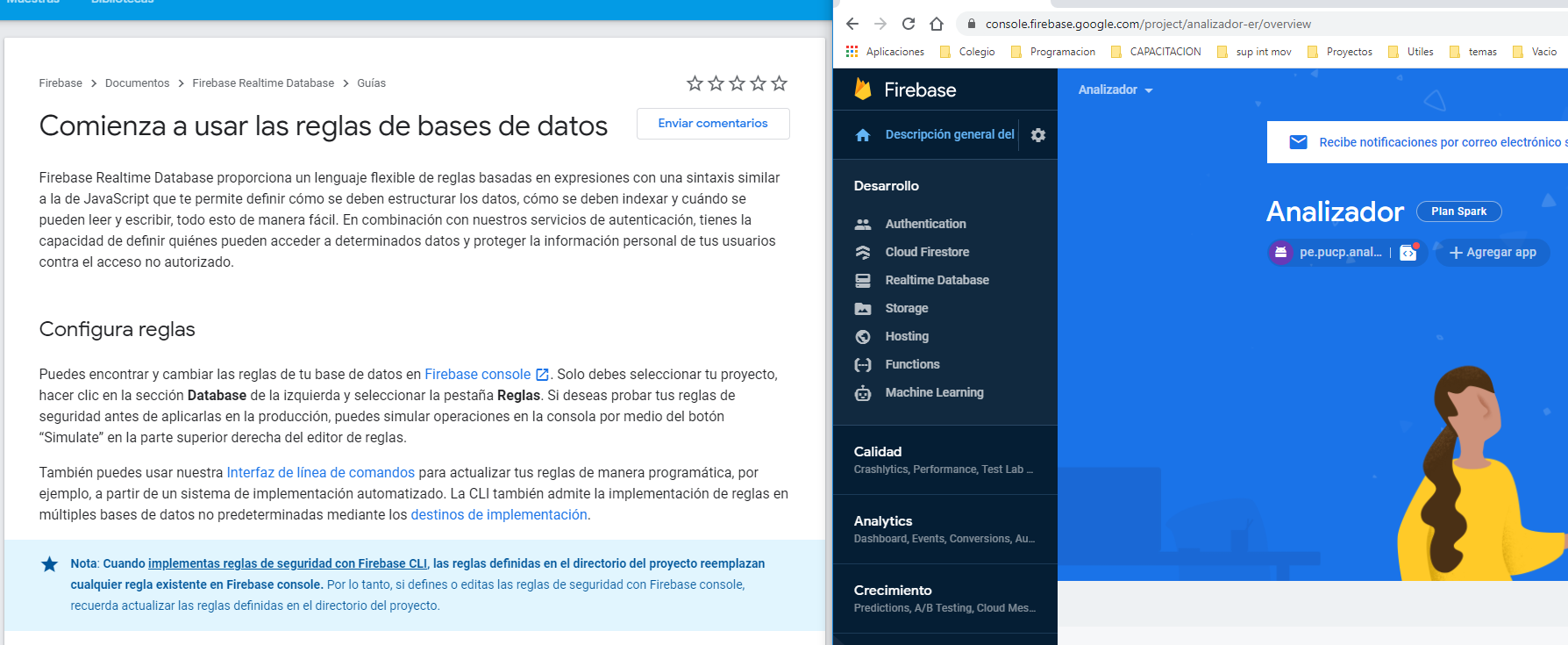
A continuación se solicita configurar las reglas de acceso a la base de datos. El comportamiento por defecto es que solo los usuarios autenticados tengan acceso, sin embargo es posible configurar el acceso público, según se muestra a continuación.

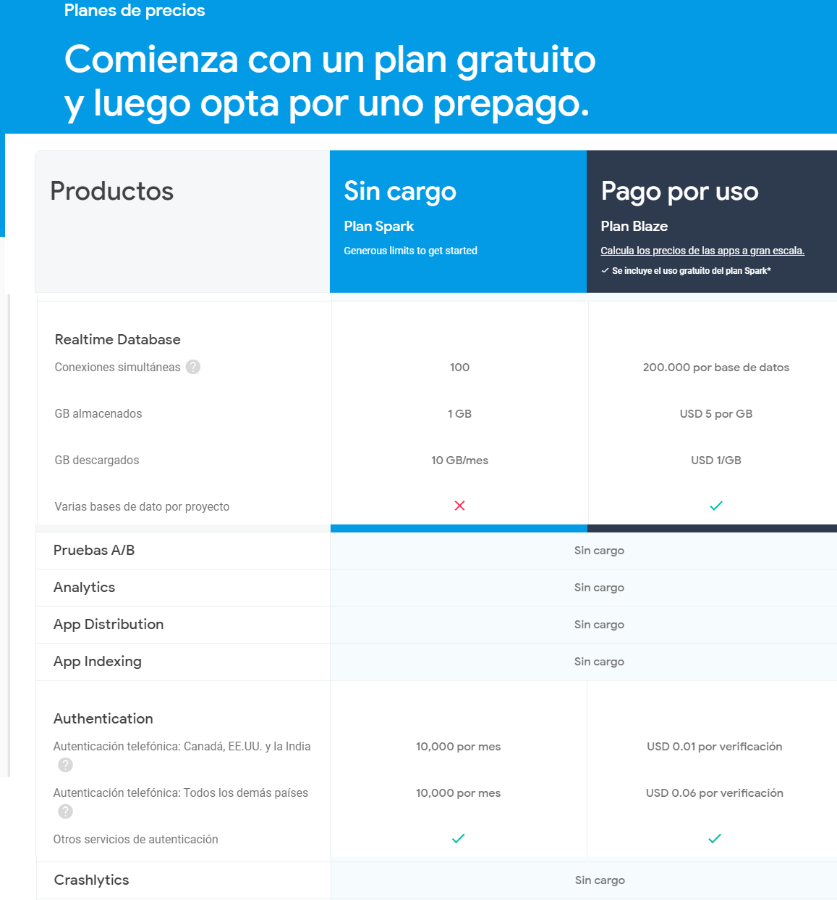


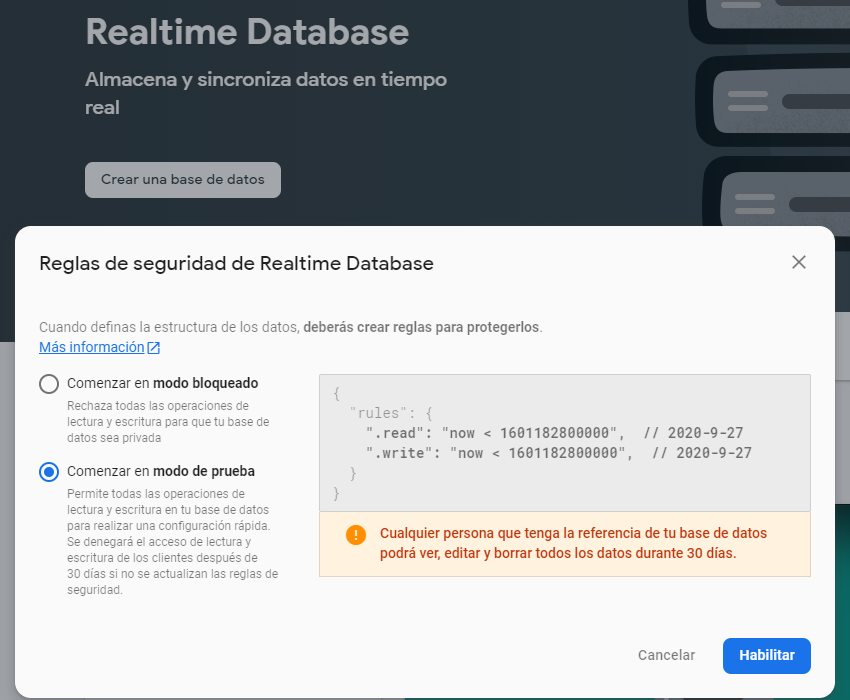










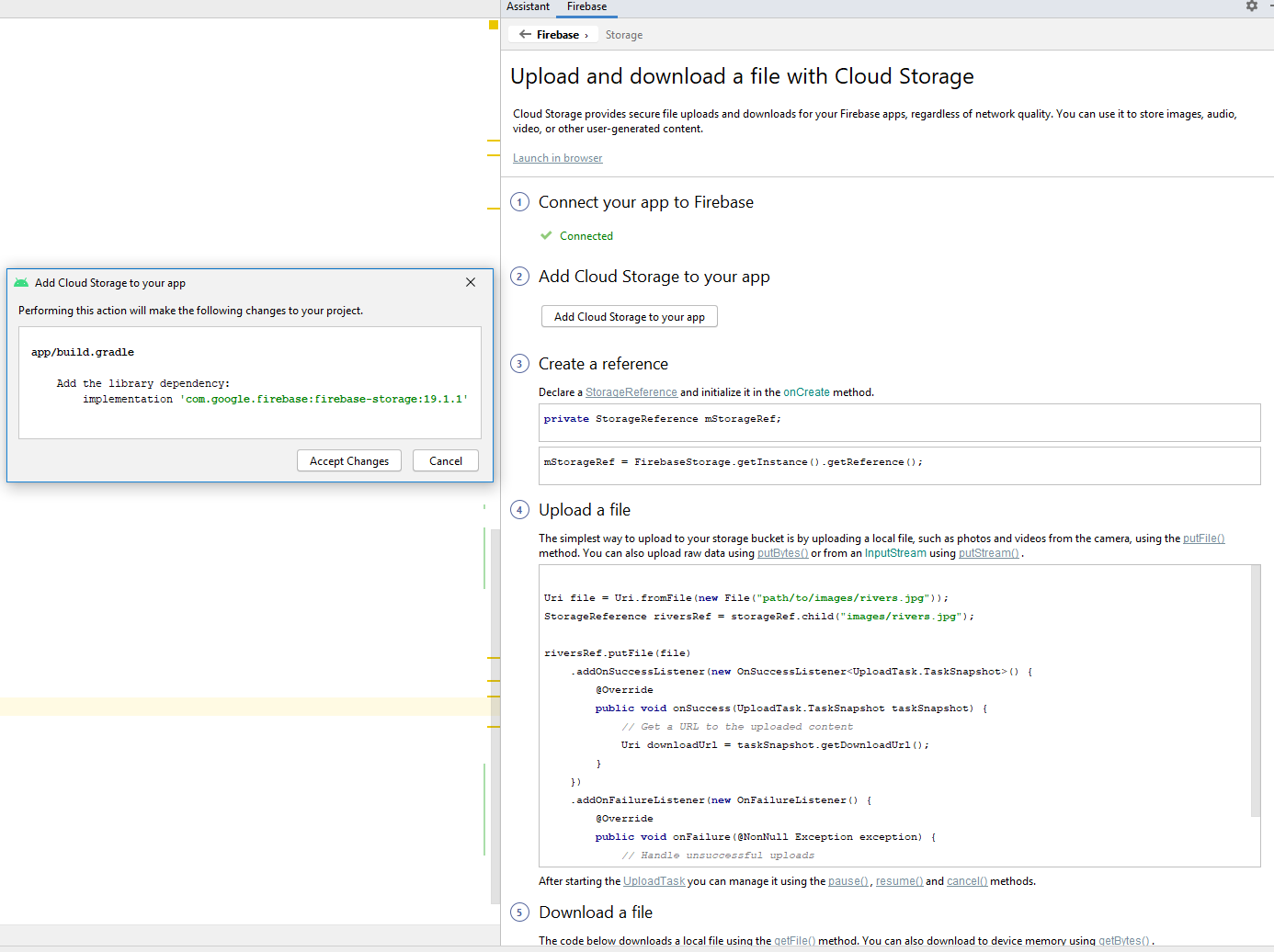




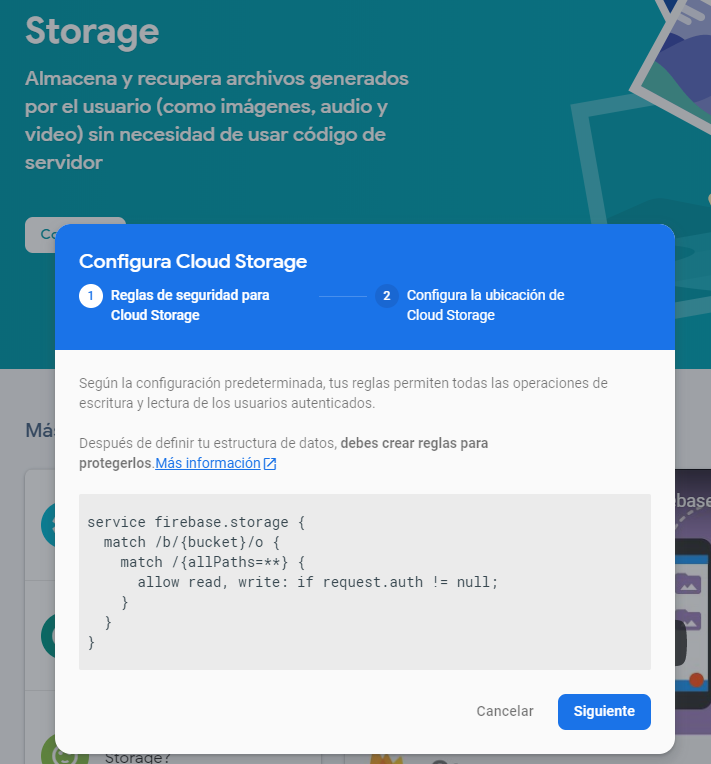


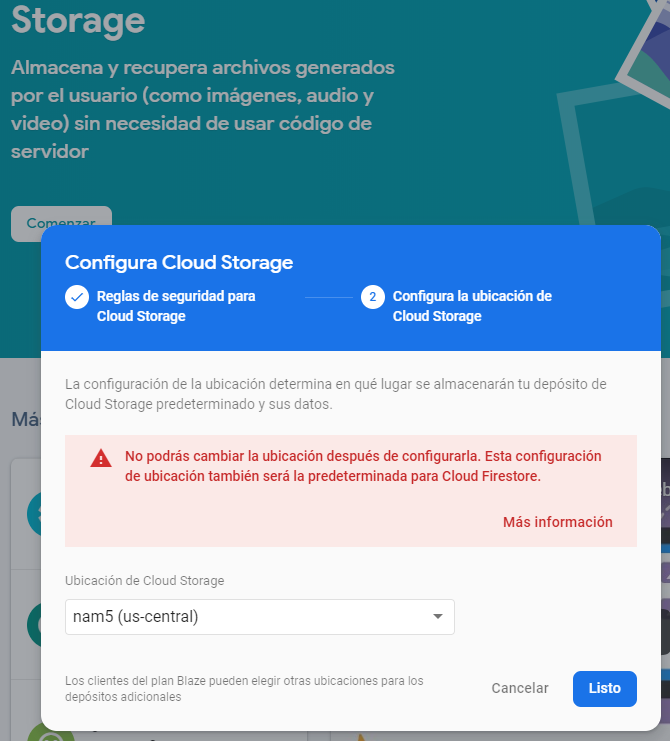
### Almacenamiento en Firebase (Storage)

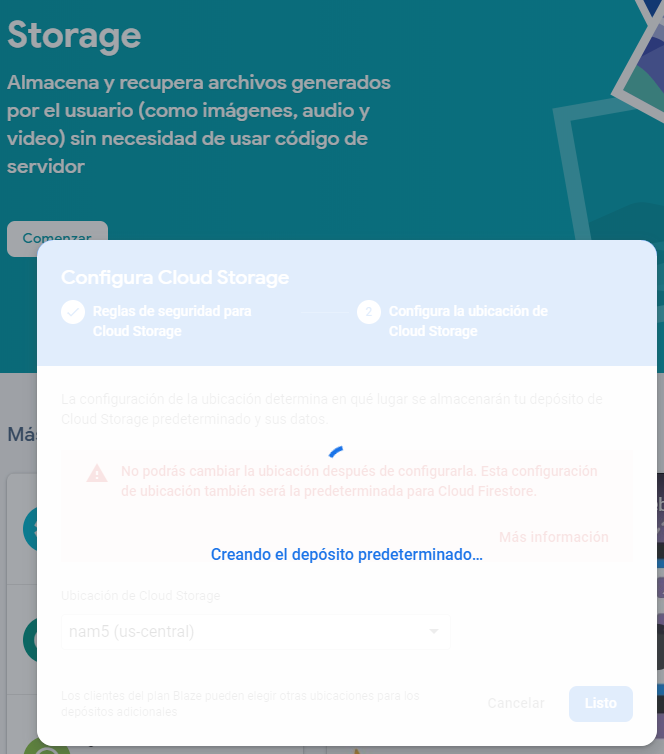
Firebase dispone de un servicio de almacenamiento en la nube. Información detallada púede encontrase en la página 153 de (Kumar, 2018). Dicho servicio se configura fácilmente desde Android Studio, según se aprecia a continuación.



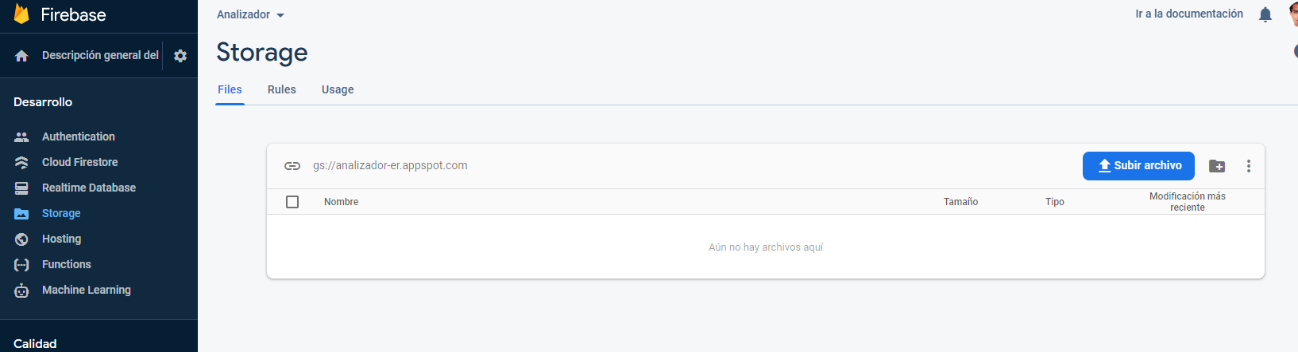
Luego, en la consola de Firebase se configura el Storage.



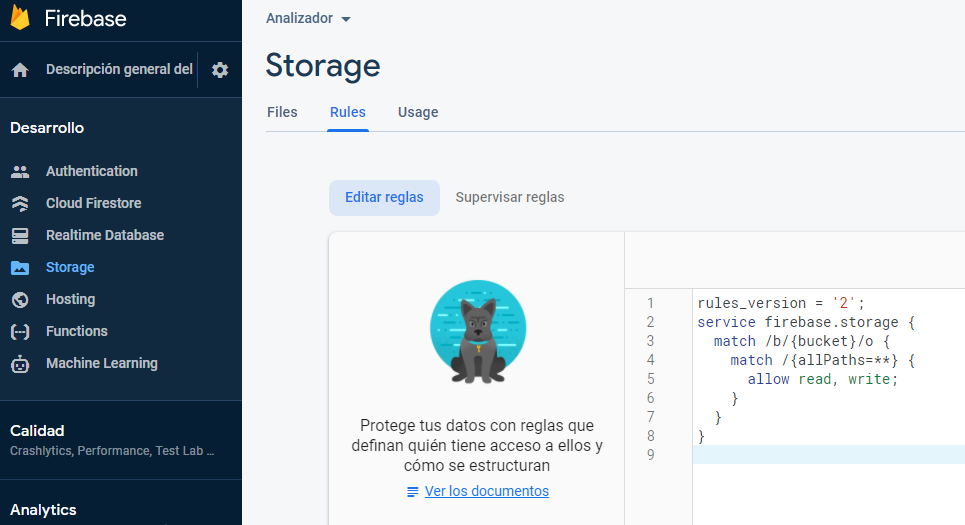




Se crea el servicio



Luego permitir que todos puedan actualizar el almacenamiento



### Subir archivo en Storage de Firebase

El procedimiento es relativamente sencillo, tal como puede encontrarse en la página 79 de (Moroney, 2017), según se transcribe a continuación.

UploadTask uploadTask = imageRef.putBytes(imageData);

uploadTask.addOnFailureListener(new OnFailureListener() {

@Override

public void onFailure(@NonNull Exception e) {

String ex = e.getLocalizedMessage();

}

}).addOnSuccessListener(new OnSuccessListener<UploadTask.TaskSnapshot>() {

@Override

public void onSuccess(UploadTask.TaskSnapshot taskSnapshot) {

String url = taskSnapshot.getDownloadUrl().toString();

}

});

No obstante, el código tradicionalmente utilizado para conocer la url de acceso al archivo almacenado no es la tradicional, sino que requiere tener en cuenta el concepto de “tareas”, según se aprecia en <https://stackoverflow.com/questions/50467814/tasksnapshot-getdownloadurl-is-deprecated#:~:text=Please%20note%20that%20it%20returns,you%20do%20any%20other%20Task.&text=ImageUrl%20%3D%20taskSnapshot.,()%3B%20this%20method%20is%20deprecated.>

*…*StorageReference riversRef = **mStorageRef**.child(**"images/"**+filelocal.getName());  
*//sube archivo local en el almacenamiento remoto*riversRef.putFile(file)  
 .addOnProgressListener(**new** OnProgressListener<UploadTask.TaskSnapshot>() {  
 @Override  
 **public void** onProgress(@NonNull UploadTask.TaskSnapshot taskSnapshot) {

…  
}  
 })  
 .addOnSuccessListener(**new** OnSuccessListener<UploadTask.TaskSnapshot>() {  
 @Override  
 **public void** onSuccess(UploadTask.TaskSnapshot taskSnapshot) {  
…

Task<Uri> downUrl=taskSnapshot.getMetadata().getReference().getDownloadUrl();  
 **while**(!downUrl.isComplete());  
 Log.*i*(**"url:"**,downUrl.getResult().toString());  
…  
 }  
 })  
 .addOnFailureListener(**new** OnFailureListener() {  
 @Override  
 **public void** onFailure(@NonNull Exception e) {

…  
 }  
 });

### Configurar autenticación en Firebase (login)

De manera similar a los servicios anteriores, utilizando la consola se configuran los métodos de login. En particular se utilizaran “Email” y “Google”.

### Lectura de información de base de datos de FireBase (RealTime Database)

Firebase utiliza objetos JSON para almacenar información en la base de datos, siendo un procedimiento simple de realizar.

No obstante, la lectura de dichos objetos no es tan trivial.

Como la aplicación ha definido el objeto a almacenar, este puede ser utilizado para recuperarlo desde la base de datos manteniendo su forma de objeto (es decir, no es necesario acceder propiedad por propiedad). Ejemplos de estos mecanismos se encuentran en <https://es.stackoverflow.com/questions/186473/expected-a-list-while-deserializing-but-got-a-class-java-util-hashmap-firebase>, <https://www.it-swarm.dev/es/java/databaseexception-no-se-puede-convertir-un-objeto-de-tipo-java.lang.string-para-escribir/834903214/>, o en <https://www.it-swarm.dev/es/java/como-convertir-datos-de-firebase-objeto-java-.../1053016412/>.

El punto importante para esta técnica es la correcta definición del objeto a almacenar/recuperar.

* Tipos de datos comunes (String, int, double, etc.) son aceptados fácilmente, sin embargo es posible también manejar Listas/Arrays de objetos
* Todo objeto debe tener definidas propiedades, sea como miembros públicos, o como miembros privados y funciones “getters” y “setters”
* Todo objeto debe tener un constructor sin parámetros ( *“public objecto(){}”* ), incluso los objetos miembros de listas/arrays.

Un ejemplo de objeto compuesto es el utilizado en la aplicación

**imagenModel.java**

**package** pe.pucp.analizador;  
  
**import** java.util.ArrayList;  
**import** java.util.List;  
**import** java.util.UUID;  
  
**public class** imagenModel {  
 *//propiedades* **public** String **Nombre**; *//uuid* **public** String **RutaLocal**;  
 **public** String **RutaRemota**;  
  
 **public int numCaras**;  
 **public** List<CaraModel> **Caras**;  
  
 **public int numObjetos**;  
 **public** List<ObjetoModel> **Objetos**;  
  
 **public** String **Descripcion**;  
  
 *//constructor* **public** imagenModel()  
 {  
 **Nombre**= UUID.*randomUUID*().toString();  
 **RutaLocal**=**""**;  
 **RutaRemota**=**""**;  
 **numCaras**=0;  
 **Caras**= **new** ArrayList<CaraModel>();  
 **numObjetos**=0;  
 **Objetos**= **new** ArrayList<ObjetoModel>();  
 **Descripcion**=**""**;  
 }

**CaraModel.java**

**package** pe.pucp.analizador;  
  
**public class** CaraModel {  
 *//propiedades* **public double age**;  
 **public** String **gender**;  
  
 **public** String **smile**; *//SI/NO* **public** String **glasses**;  
 **public** String **emotion**;  
  
  
 **public int left**;  
 **public int top**;  
 **public int width**;  
 **public int height**;  
  
  
 *//constructor* **public** CaraModel()  
 {  
 **age**=0;  
 **gender**=**""**;  
  
 **smile**=**""**;  
 **glasses**=**""**;  
 **emotion**=**""**;  
  
 setRectangle(0,0,0,0);  
 }  
  
 **public** CaraModel(**double** pAge, String pGender)  
 {  
 **age**=pAge;  
 **gender**=pGender;  
  
 **smile**=**""**;  
 **glasses**=**""**;  
 **emotion**=**""**;  
  
 setRectangle(0,0,0,0);  
 }  
  
 *//metodos* **public void** setRectangle(**int** pLeft, **int** pTop, **int** pWidth, **int** pHeight)  
 {  
 **left**=pLeft;  
 **top**=pTop;  
 **width**=pWidth;  
 **height**=pHeight;  
 }  
  
}

Y la forma cómo leer de la base de datos se resume a continuación.

Activity/Frament

…

@Override  
**public** View onCreateView(LayoutInflater inflater, ViewGroup container,  
 Bundle savedInstanceState) {  
 View view = inflater.inflate(R.layout.***fragment\_imagenmodel\_list***, container, **false**);  
  
 *// Set the adapter* **if** (view **instanceof** RecyclerView) {  
 Context context = view.getContext();  
 **final** RecyclerView recyclerView = (RecyclerView) view;  
 **if** (**mColumnCount** <= 1) {  
 recyclerView.setLayoutManager(**new** LinearLayoutManager(context));  
 } **else** {  
 recyclerView.setLayoutManager(**new** GridLayoutManager(context, **mColumnCount**));  
 }  
  
 *//lista de base de datos* **final** List<imagenModel> list = **new** ArrayList<imagenModel>();  
  
 *//obtiene lista de imagenes* **final** FirebaseDatabase database = FirebaseDatabase.*getInstance*();  
 **final** DatabaseReference myRef = database.getReference( getString(R.string.***firebaseRTDB\_name***));  
 myRef.limitToLast(20).addListenerForSingleValueEvent(  
 **new** ValueEventListener()  
 {  
 @Override  
 **public void** onDataChange(DataSnapshot snapshot) {  
**try**{  
 **for** (DataSnapshot child: snapshot.getChildren()) {  
 list.add(child.getValue(imagenModel.**class**));  
 }  
 }**catch** (Exception e){e.printStackTrace();}  
  
 *//establece adaptador* recyclerView.setAdapter(**new** imagenModelRecyclerViewAdapter(list));  
  
 }  
  
 @Override  
 **public void** onCancelled(DatabaseError databaseError) {}  
 }  
 );  
  
 }  
 **return** view;  
}

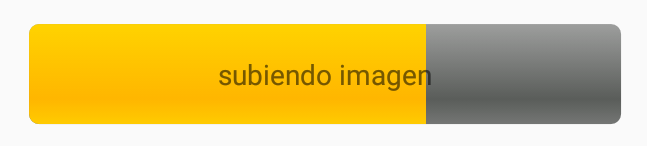
…

## Barra de progreso

Android dispone de una barra de progreso que permite visualizar el porcentaje de avance un proceso. Según <https://stackoverflow.com/questions/18410984/android-displaying-text-in-center-of-progress-bar>, una técnica que mejora la visualización consiste en añadir un texto que muestre el nombre del proceso que se está evaluando.

Un ejemplo de layout que sigue dicha técnica se muestra a continuación.

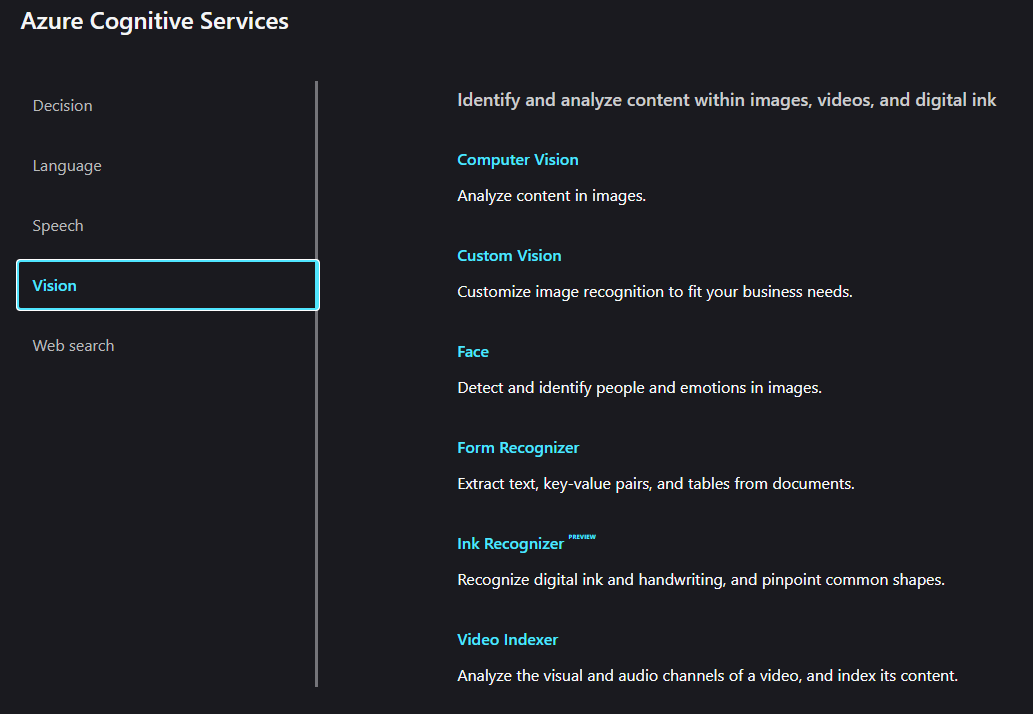
<**RelativeLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="8dp"  
 android:layout\_marginTop="8dp"  
 android:layout\_marginEnd="8dp"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent"** >  
 <**ProgressBar  
 android:id="@+id/progress\_bar"  
 style="@android:style/Widget.ProgressBar.Horizontal"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:max="100"  
 android:minWidth="50dp"  
 android:minHeight="50dp"  
 android:progress="1"** />  
  
 <**TextView  
 android:id="@+id/progressBarinsideText"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_alignParentLeft="true"  
 android:layout\_alignParentRight="true"  
  
 android:layout\_centerVertical="true"  
 android:text="mensaje"  
 android:textAlignment="center"** />  
</**RelativeLayout**>



## Servicios Cognitivos de Azure

Microsoft, a través de su producto Azure, ofrece un conjunto de servicios cognitivos para ser utilizados por las aplicaciones.

Dentro de los servicios cognitivos, Vision proporciona los siguientes (<https://azure.microsoft.com/en-us/services/cognitive-services/#features>).



De ellos resultan de interés “Computer Vision” y “Face”.

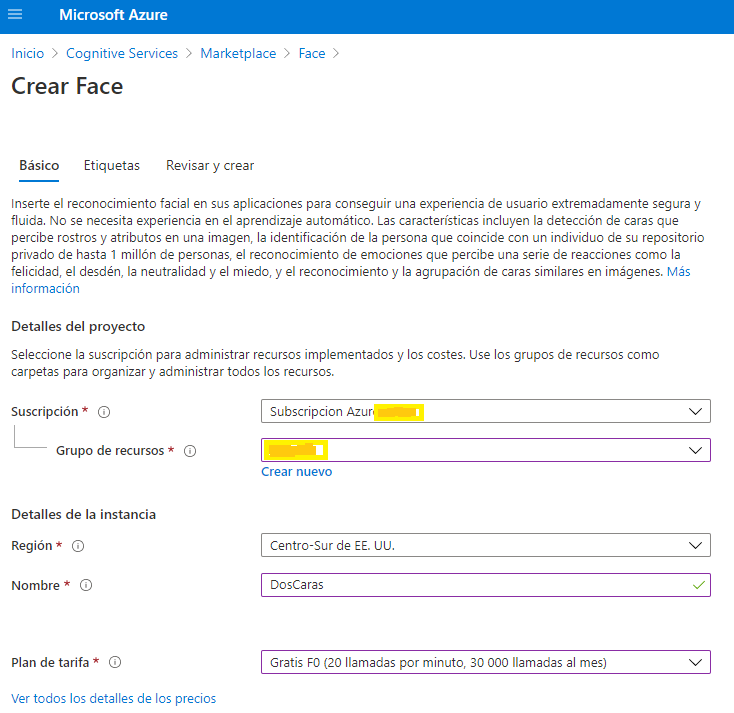
### Creación de cuenta Azure

Para el empleo de los servicios cognitivos de Azure, es necesario tener una cuenta en Microsoft Azure. En este caso la cuenta que ya se disponía con anterioridad será utilizada.

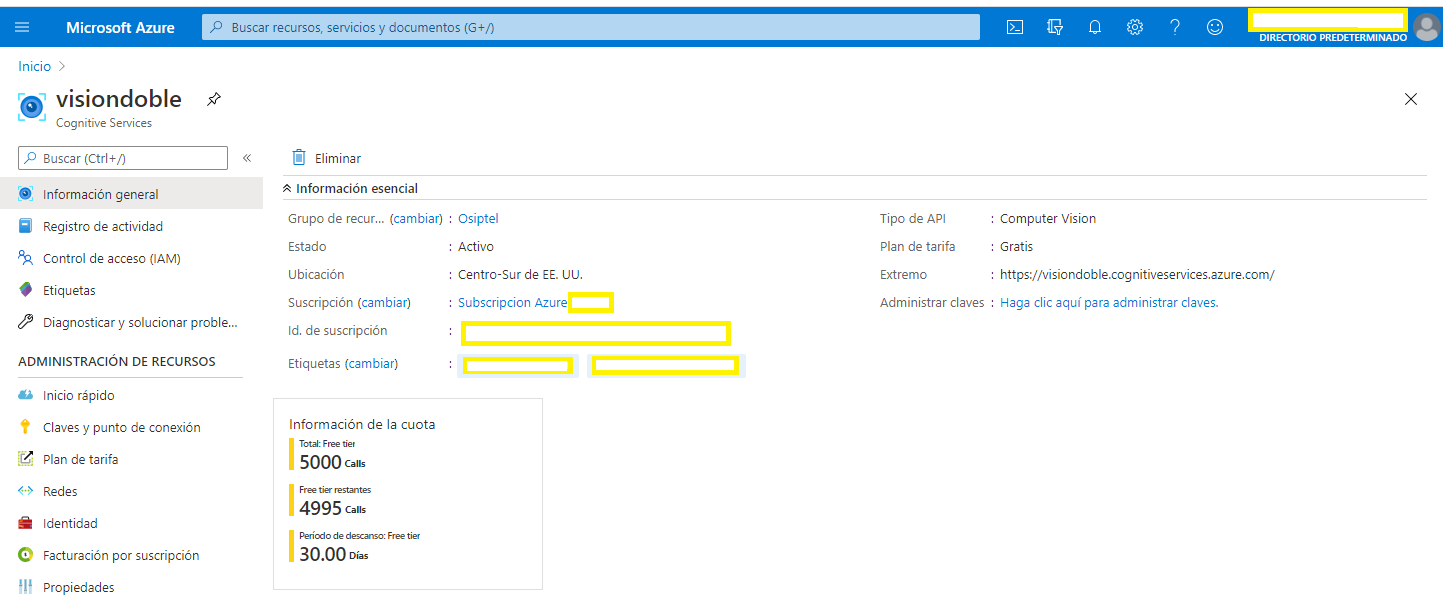
### Activación de servicios cognitivos en Azure

De manera similar a Firebase, donde su administración se realiza mediante la “Consola”, en este caso Azure permite su administración mediante el “Portal”.

Dentro de Portal de Azure, se utiliza el grupo de recursos Osiptel (ya existente) para almacenar los servicios cognitivos a utilizar. Se muestra un ejemplo de creación del servicio “Face” denominado “DosCaras”.



Siguiendo un procedimiento similar, se agrega un recurso cognitivo “visiondoble”, dentro de un plan gratuito, tal como se aprecia a continuación.



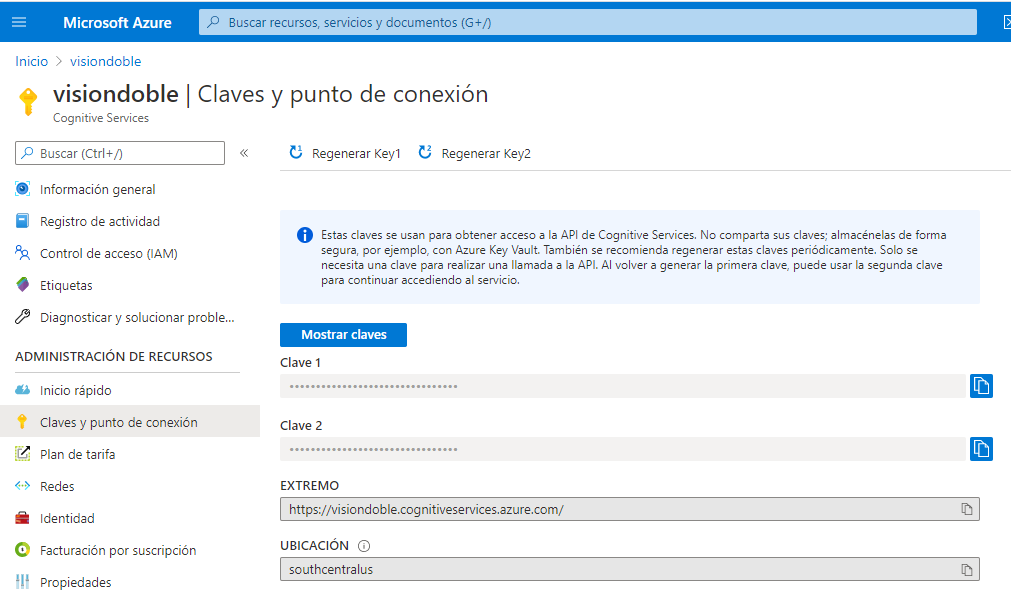
### APIs disponibles para “Computer Vision” y “Face”

A continuación se muestran las APIs disponibles en cada caso.

|  |  |
| --- | --- |
| Face | Computer Vision |
|  |  |

### Claves y punto de conexión

Una vez activado el servicio, el portal permite explorarlo. En particular la sección “Claves y puntos de conexión” del menú izquierdo permite tener la información necesaria para realizar la conexión a través de programación. Se muestra un ejemplo para el servicio cognitivos de “Computer Vision” creado denominado “visiondoble”.



### Prueba de servicios

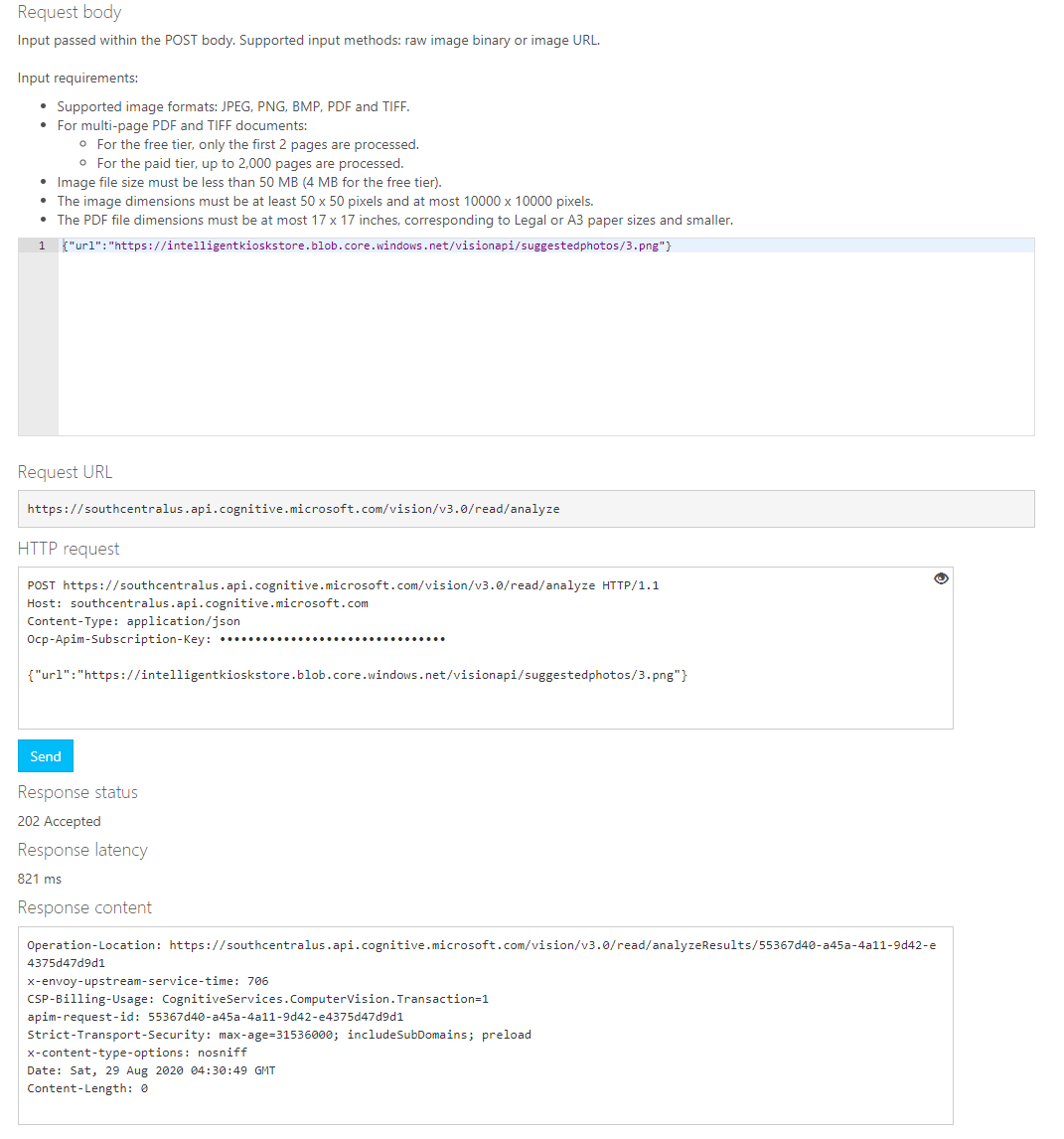
El portal permite realizar la prueba de los servicios directamente en la misma plataforma utilizando la “Consola de API”.



Para tal efecto se accede a la consola mediante el enlace en la misma página y luego se selecciona la región donde fue creada y se coloca la clave 1 del API.

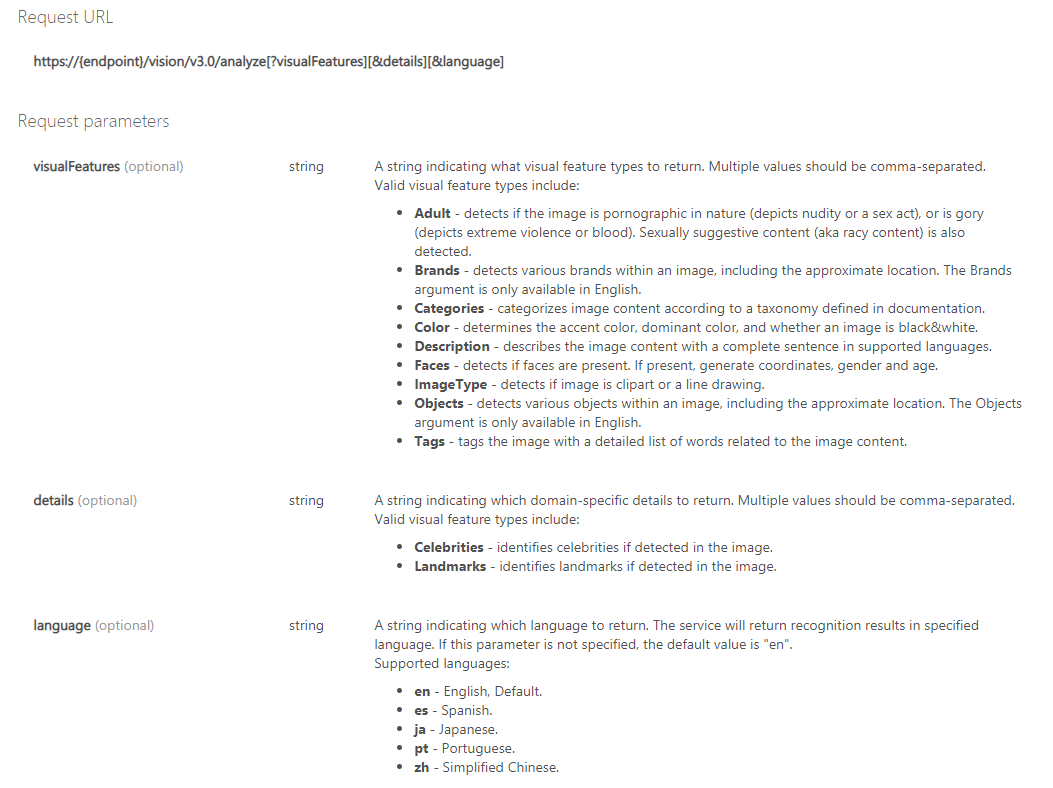
|  |  |
| --- | --- |
|  |  |

Luego se presiona el botón “Send” en la parte inferior (el sistema proporciona una imagen de ejemplo) y se obtiene el resultado del análisis.



### Computer Vision - Análisis de imágenes disponibles: selección

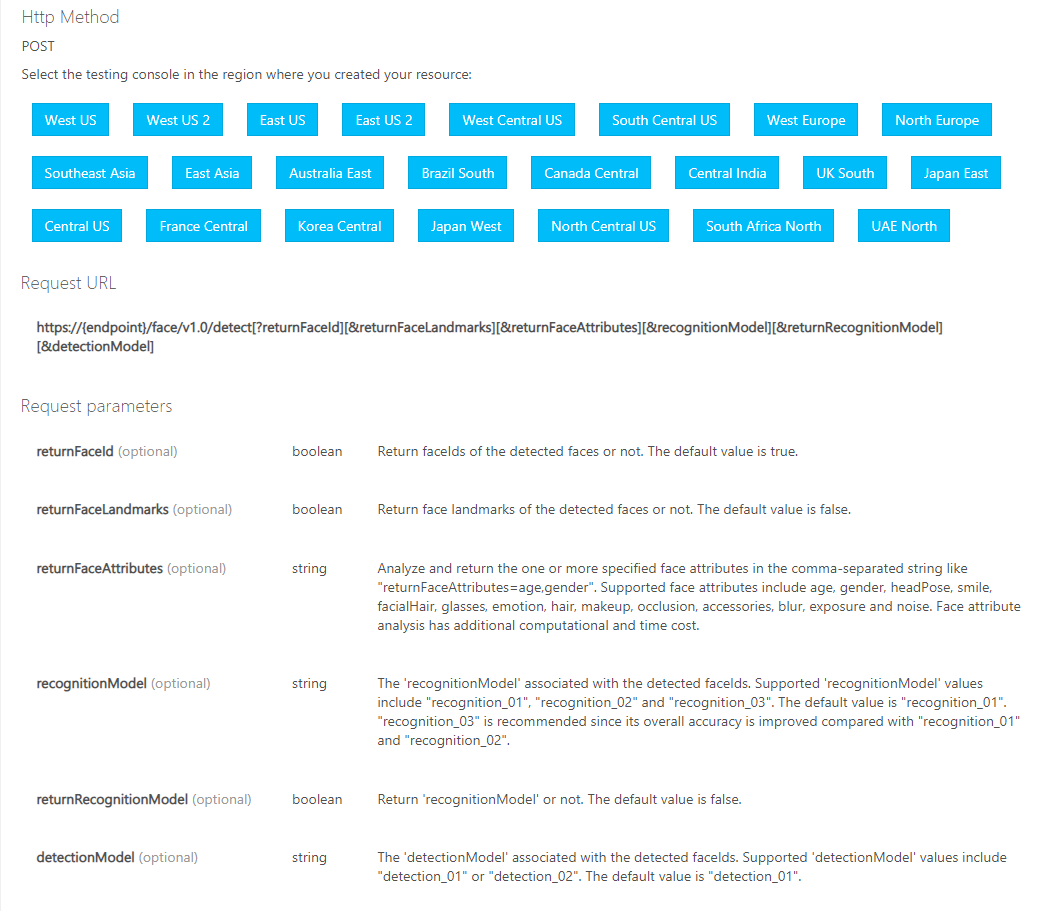
El API de permite realizar los siguientes análisis.



De este conjunto de análisis se seleccionará: Description (captions/text), Objects (rectangle,object), Faces (age,gender,faceRectangle).

### Face – Análisis de rostros disponibles: selección

El API de Face incluye el método “Detect”, el cual permite detectar, adicionalmente a aquella incluida en el API de Computer Vision (faceRectangle, gender y age), otro tipo de información de interés (smile, glasses, emotion), tal como se muestra a continuación.



### Esquema de llamadas a APIs de Cognitive Services de Azure

De acuerdo al análisis efectuado, la aplicación primro realizará una llamada al API “Computer Vision”, y, en caso se detecten rostros, se llamará al API “Faces” para obtener información adicional.

## Selección de librería para acceso a APIs REST de servicios cognitivos

### Selección de librería

Se ha utilizado inicialmente la librería okhttp, sin embargo según <https://square.github.io/okhttp/> dicha libreía sólo se puede utilizar a partir del API 21 de Android, y la aplicación debe funcionar desde el API 19. En tal sentido no es posible el uso de dicha librería en esta aplicación.

De esta forma se debe utilizar los procesos más sencillos disponibles en Android, es decir, creando un hilo separado de ejecución donde se realice el llamado al servicio web a través de conexiones TCP sencillas, y manejando internamente el llamado y posteriormente los eventos de recepción y error.

### Código de ejemplo

Un ejemplo de esta forma de trabajo puede encontrarse en <https://code.tutsplus.com/es/tutorials/android-from-scratch-using-rest-apis--cms-27117>

*//crea tarea asincrona para manejo de consulta a servicio web REST*AsyncTask.*execute*(**new** Runnable()  
{  
 @Override  
 **public void** run()  
 {  
**try** {  
 *// Create URL* URL githubEndpoint = **new** URL(**"https://api.github.com/"**);  
 *// Create connection* HttpsURLConnection myConnection =  
 (HttpsURLConnection) githubEndpoint.openConnection();  
 *//añade request headers* myConnection.setRequestProperty(**"User-Agent"**, **"my-rest-app-v0.1"**);  
myConnection.setRequestProperty(**"Accept"**,**"application/vnd.github.v3+json"**);  
  
 *//lee respuestas* **if** (myConnection.getResponseCode() == 200) {  
 *// Success  
  
 //lee respuesta en forma binaria* InputStream responseBody = myConnection.getInputStream();

*//formatea respuesta en formato texto utf-8, usual de REST* InputStreamReader responseBodyReader =  
 **new** InputStreamReader(responseBody, **"UTF-8"**);

*//formatea respuesta en formato json* JsonReader jsonReader = **new** JsonReader(responseBodyReader);  
  
 *//obtiene un elemento del json recibido* jsonReader.beginObject(); *// Start processing the JSON object* **while** (jsonReader.hasNext()) { *// Loop through all keys* String key = jsonReader.nextName(); *// Fetch the next key* **if** (key.equals(**"organization\_url"**)) { *// Check if desired key  
 // Fetch the value as a String* String value = jsonReader.nextString();

*// utiliza valor* Log.*wtf*(**"json"**, value);  
**break**; *// Break out of the loop* } **else** {  
 jsonReader.skipValue(); *// Skip values of other keys* }  
 }  
 *//cierra json reader* jsonReader.close();  
 *//cierra conexion a servicio web* myConnection.disconnect();  
  
 } **else** {  
 *// Error handling code goes here* Log.*e*(**"error ResponseCode:"**, String.*valueOf*( myConnection.getResponseCode() ) );  
 }  
 }  
 **catch** (Exception e) {  
 e.printStackTrace();  
 }  
 }  
});

Dicho código corresponde a un método GET, sin embargo, los servicios cognitivos de Azufre se invocan utilizando el método POST. En tal sentido, se debe adecuar dicho código de ejemplo. Una forma se encuentra en <https://www.baeldung.com/httpurlconnection-post>.

## Sobre ReciclerView y el patrón viewHolder

### Diversidad de implementaciones

Existen mucha información sobre implementaciones de estos artefactos, incluso existen utilitarios que buscan implementaciones sencillas. Sin embargo su implementación no es trivial.

Luego de muchas pruebas e investigaciones, en <https://medium.com/swlh/create-recyclerview-in-android-fragment-c0f0b151125f> se ha podido conseguir un método funcional. Este método permite entender el patrón para el caso de una actividad donde existe un control de navegación basado en fragmentos, donde la navegación reemplaza el fragmento activo por el siguiente según la opción de menú seleccionado.

El código de ejemplo utilizado en el curso no resultó aplicable por el empleo de un API de Android antiguo (API 19), y las librerías utilizarías y materiales de layout empleados no eran soportados en forma correcta por el IDE Android Studio.

Adicionalmente, la aplicación tiene una forma diferente a las anteriores y, por tanto, se tuvo que modificar para poder implementar el patrón.

### Implementación realizada

La forma de implementación se ha incluido en el proyecto de la siguiente forma:

1. Crear una clase con la información a visualizar. En este caso se denomina imagenModel, la cual contiene, entre otros, la url de la imagen y la información necesaria para mostrar el resultado (mediante la sobreescritura del método ToString())
2. Adicionar una nueva actividad vacía, colocada en un nivel inferior al paquete de la aplicación (para ordenar el código)
3. Adicionar un nuevo fragmento de tipo lista, donde se configuran las propiedades paquete (mismo paquete de la actividad vacía), tipo de objeto (clase imagenModel), y renombrar los demás nombres de propiedades para reemplazar “ítem” por “imagen”.
4. Notar que Android Studio agrega un paquete denominado “dummy” que inserta objetos aleatorios sólo para visualización
5. En el layout de la nueva actividad se adiciona un fragment que haga referencia al nuevo fragmento tipo lista creado, configurando las propiedades android name (con el nombre del fragmento nuevo creado) y tools layout (con el nombre del layout del nuevo fragmento)

Otras modificaciones corresponden a dar la mejor forma visual al archivo de layout que contiene el elemento de la lista, y reemplazar el paquete “dummy” con la conexión a la base de datos para obtener la información real a mostrar.

### Empleo de cardview

Una modificación particularmente problemática es el empleo del cardview en los elementos de la lista.

Luego de incluir la dependencia de cardview (librería materials de Google) en el archivo gradle del proyecto, su empleo ocasiona que la actividad se cierre.

El problema se origina en el empleo de la librería de compatibilidad para versiones antiguas de Android, librería que no es perfecta del todo. En <https://stackoverflow.com/questions/60182380/materialcardview-crashes-with-material1-1-0> se puede apreciar una configuración que funciona en forma efectiva. El método consiste en definir el tema e incluir la librería res-auto, como se aprecia a continuación.

…

<**com.google.android.material.card.MaterialCardView xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_margin="8dp"  
  
 android:theme="@style/Theme.MaterialComponents.Light"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 app:cardElevation="8dp"** >

…

El efecto deseado se aprecia a continuación.



## Imágenes de aplicación en funcionamiento

A continuación se muestran ejemplos de la aplicación en funcionamiento.

|  |  |  |
| --- | --- | --- |
|  |  |  |

|  |  |  |
| --- | --- | --- |
|  |  |  |

|  |  |  |
| --- | --- | --- |
|  |  |  |

|  |  |  |
| --- | --- | --- |
|  |  |  |

# Bibliografía

Kumar, A. (2018). *Mastering Firebase for Android Development.* Birmingham: Packt Publishing.

Meier, R., & Lake, I. (2018). *Professional Android.* Indianapolis: Wrox.

Moroney, L. (2017). *The Definitive Guide to Firebase.* Seattle: Apress.

1. Ver <https://firebase.google.com/?gclid=CjwKCAjwyo36BRAXEiwA24CwGYHjsG_ZZYntJ8IgTU1QAq4bNJTHDywwg1rGNAbP4BvD2kXUnlXsfRoCnhAQAvD_BwE> [↑](#footnote-ref-1)
2. <https://azure.microsoft.com/en-us/> [↑](#footnote-ref-2)