



POLITÉCNICA

Firebase

ANDRÉS MICÓ MILLÁN

Máster universitario en desarrollo de aplicaciones móviles y servicios para dispositivos móviles

E.T.S. de Ingeniería de Sistemas Informáticos | Programación en Android

Tabla de contenido

1. Introducción	2
2. Funcionalidades	3
3. Implementación	4
3.1. Android Studio & Firebase	4
3.2. Autenticación	5
3.2.1. Email & Password	5
3.2.2. Número de Teléfono	6
3.2.3. Facebook	7
3.2.4. GoogleID	8
3.2.5. Twitter	9
3.3. Notificaciones PUSH	9
3.3.1. Consola Firebase	10
3.3.2. Propio dispositivo	10
3.3.3. BackEnd AdHoc	11
3.4. Base de Datos	11
3.5. Almacenamiento	12
3.6. Admob	12
3.7. Crashlytics	13
3.8. InviteLinks	13
4. Conclusiones	13

1. Introducción

La aplicación para Android presentada en esta memoria, está optimizada para dispositivos de 5 o más pulgadas funcionando a partir de la API 19 o 4.4 (KitKat). La aplicación muestra gran parte de las funcionalidades y opciones de desarrollo que nos ofrece la herramienta de Google Firebase.

Firebase es una plataforma para el desarrollo de aplicaciones móviles, que ofrece una gran cantidad de servicios para nuestro desarrollo en Android, iOS, Web, C++, Unity. Es por ello, que se podría equiparar a la herramienta Microsoft Azure o AWS de Amazon, ya que todas son plataformas en la nube que dan servicios y una infraestructura para los proyectos.

Con esta plataforma adquirimos una serie de ventajas como, por ejemplo, prescindir de un Webservice, es decir, no necesitaremos elaborar un BackEnd propio en nuestro servidor, haciendo todas las conexiones con la base de datos, ya que nos ofrece de una manera muy sencilla la posibilidad de construir un Webservice con arquitectura API-Rest, además de poder configurar prácticamente todas las prestaciones del servidor. Por otro lado, cuenta con APIs de autenticación de usuarios, lo que nos ahorra una gran cantidad de trabajo en comparación a crear nuestro propio sistema de Login. Así mismo cuenta con notificaciones PUSH usando la tecnología FCM, casi transparente para el programador.

No obstante, como pequeño inconveniente podríamos mencionar el precio de licencia de Firebase, que nada tiene que ver con el de otras plataformas en la nube pero, aun así, sigue siendo caro si queremos incorporar las Analytics y las Predictions.

En la siguiente figura se puede apreciar gran parte de las funcionalidades que Firebase nos ofrece para incorporarlas a nuestros proyectos.

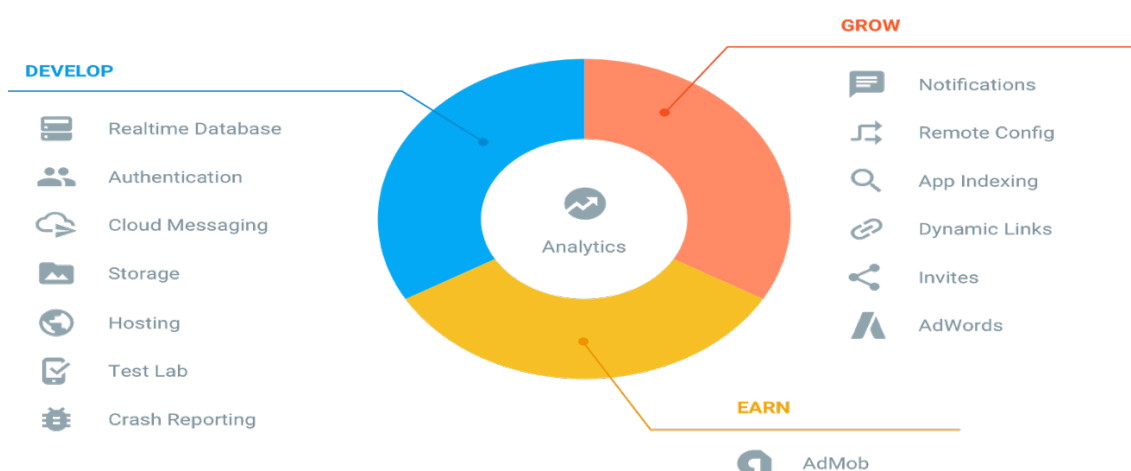


Figura 1: Firebase

2. Funcionalidades

En este apartado comentaremos las funcionalidades que se han implementado en la aplicación FirebaseApp:

- Autenticación de usuarios en la App de diferente forma
 - Email y contraseña.
 - Número de teléfono.
 - Google SingIn.
 - Facebook Login.
 - Twitter Login.
- Notificaciones PUSH, implementada de diferente forma
 - A través de la consola de Firebase.
 - A través de la aplicación.
 - A través de un BackEnd personalizado.
- Base de datos en tiempo real
 - Manejo de peticiones,
 - Actualización y recuperación de los datos en tiempo real.
- Almacenamiento en Cloud.
 - Recuperación de archivos almacenados.
 - Subida de archivos de nuestro dispositivo.
- Integración con AdMob, plataforma de monetización
 - Banners publicitarios de terceros.
 - Publicidad intersticial.
 - Videos recompensados.
- Crashlytics
- Links de invitación.

3. Implementación

En este apartado se explicará de una forma más detallada como se ha implementado las diferentes funcionalidades que ofrece la aplicación.

3.1. Android Studio & Firebase

La primera tarea que debemos de realizar para empezar a utilizar la herramienta, es enlazar nuestra aplicación. Para ello tenemos dos opciones, haciéndolo manualmente desde la consola, indicando el nombre del paquete y una clave SHA-1 específica para esa aplicación, o podemos hacerlo automáticamente desde Android Studio, que se encargará de generar las claves y configurar un proyecto vacío.

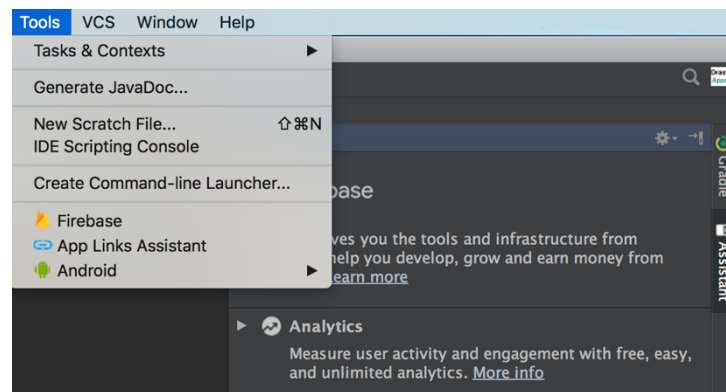


Figura 2: Tools Android Studio

Una vez enlazada la aplicación, en nuestra consola de Firebase nos aparecerá el proyecto creado y listo para configurarlo. Por defecto, aparece un dashboard con los usuarios activos diarios y mensuales, así como los fallos. Esta vista de inicio se puede modificar para ver de un primer vistazo aquella información que nos resulte más relevante.

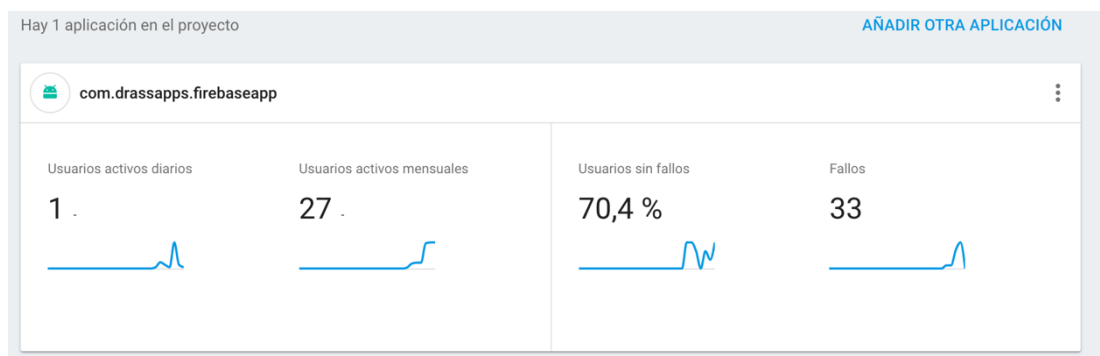


Figura 3: Dashboard principal Firebase

3.2. Autenticación

Una gran funcionalidad que incorpora Firebase es la posibilidad de crear un Login en nuestra aplicación totalmente gratis y con distintas vías de acceso, haciendo que nuestra app sea accesible a más usuarios.

3.2.1. Email & Password

La autenticación por defecto que nos propone Firebase es la utilización a través de un email como clave primaria, por lo tanto, única y no nula que representará el nombre de usuario, y una contraseña asociada a ese email.

La primera tarea que tenemos que hacer es añadir las dependencias y plugin de google services, para a continuación poder utilizar los métodos de la API de Firebase-Auth.

```
// Autenticación con Firebase (Email & Telefono)
compile 'com.google.firebase:firebase-auth:11.6.0'
compile 'com.google.firebase:firebase-core:11.0.4'
```

Figura 4: Dependencias Email & Password

Nuestra labor en la actividad consiste en pedir al usuario un email válido y una contraseña para a continuación, utilizar el método ***createUserWithEmailAndPassword*** que recibirá dos cadenas de texto. Al ejecutarse el método si todo ha ido bien, se generará un nuevo usuario en nuestro proyecto, con un UID único que permite a Firebase gestionar los usuarios. Para mejorar la experiencia del usuario, podemos añadir un ***Task<AuthResult>*** que nos devolverá el resultado del registro y dependiendo si ha sido completado o no, actualizar la UI y mostrar al usuario sus datos, o un error en el registro.

Adicionalmente, Firebase ofrece diversos métodos asociados a esta clase, pudiendo enviar un email de validación del email del usuario, de recuperación de contraseñas o cambio del correo electrónico, que al tener ya el UID registrado, podemos editarlo, todas ellas plantillas configurables. Es importante mencionar, que para que este método de autenticación del usuario funcione, debemos de habilitarlo desde la consola.



Figura 5: Métodos de inicio de sesión

3.2.2. Número de Teléfono

El segundo método de autenticación que nos propone Firebase es utilizar un número teléfono como nombre de usuario, cabe destacar que esta funcionalidad no está incluida en la licencia gratuita. Al ser una clase propia de Firebase-Auth ya está incluida en las dependencias del apartado 3.1.1 por lo que podemos utilizar sus métodos de forma inmediata.

La finalidad de esta autenticación es que el usuario indique un número de teléfono, para a continuación recibir un SMS con un código de verificación que tendrá que poner en la aplicación. Una vez enviado el código a Firebase, comparará si el código corresponde con el que ha enviado a ese número de teléfono y si es así se registrará un nuevo usuario.

Las peticiones de los SMS no se sobrescriben, es decir, si un usuario pide muchas veces un código de verificación recibirá el mismo siempre para ese número de teléfono ya que los códigos tienen un tiempo de vida especificado por el desarrollador.

```
// Enviamos el código de registro al numero dicho por el usuario
private void enviarCodigoRegistro(String numero){
    PhoneAuthProvider.getInstance().verifyPhoneNumber(
        numero,                // Número de teléfono a verificar
        60,                     // Tiempo de vida del código
        TimeUnit.SECONDS,       // Medida del tiempo de vida
        this,                   // Activity (para el callback)
        mCallbacks);           // OnVerificationStateChangedCallbacks
}
```

Figura 6: Envía código registro

Al igual que ocurre con el email & password, debemos de habilitar dicho método y contiene métodos de reenvío, recuperación de contraseña o de modificación de datos. En la figura que se presenta a continuación se puede ver el método que verifica y registra al usuario en la aplicación de Firebase.

```
// Verificamos el código introducido por el usuario
private void verifyPhoneNumberWithCode(String verificationId, String code) {
    PhoneAuthCredential credential = PhoneAuthProvider.getCredential(verificationId, code);
    signInWithPhoneAuthCredential(credential);
}
```

Figura 7: Envía código registro

3.2.3. Facebook

Dejando a un lado las autenticaciones de Firebase, también tenemos la posibilidad de vincular nuestra cuenta de Facebook a la aplicación, y una vez vinculada, nuestras credenciales de la cuenta de Facebook serán las mismas para nuestra aplicación. Este método de verificación conlleva una serie más de pasos.

En primer lugar, tras añadir las dependencias necesarias para poder utilizar la API y el SDK de Facebook, debemos de crear una nueva aplicación en la consola de Facebook Developers, que nos provee de una serie de APIs para integrarla con nuestra aplicación.

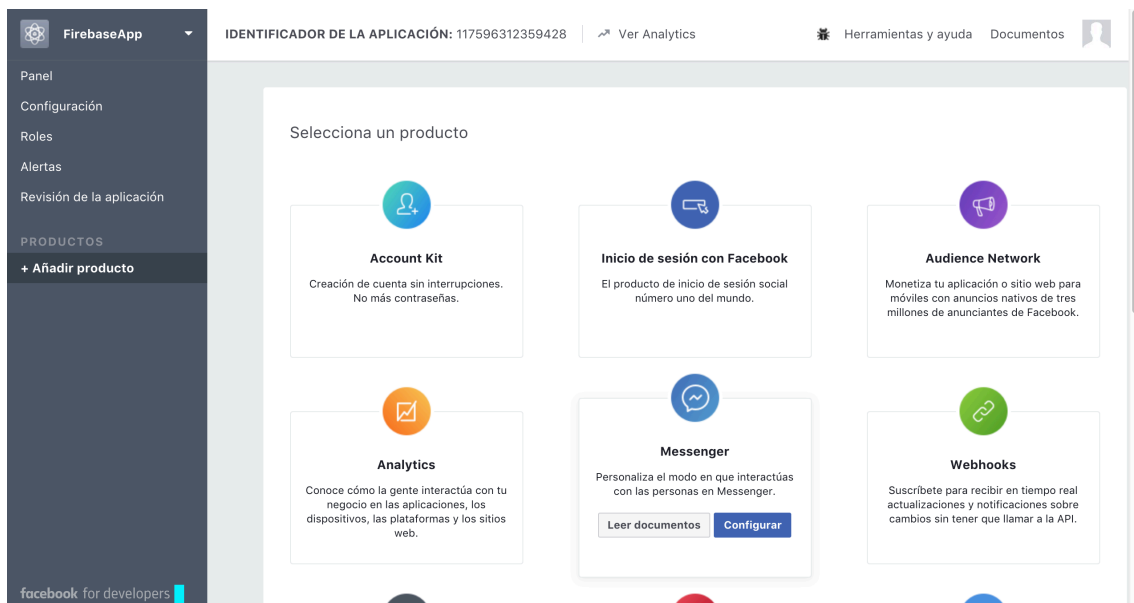




Figura 8: Consola Facebook Developer

Una vez creada la aplicación, debemos establecer el servicio de Login, para ello necesitaremos un par de claves HASH, una para desarrollo, y otra para producción cuando esté lanzada en los markets. Este par de claves HASH se pueden conseguir ejecutando la siguiente sentencia en un terminal Linux/Mac.

```
keytool -exportcert -alias androiddebugkey -keystore ~/.android/debug.keystore | openssl sha1 -binary | openssl base64
```

Tras configurar las claves y vincular el nombre del paquete, (com.drassapps.firebaseapp) en nuestro caso, obtendremos dos claves únicas de dicha aplicación de Facebook, una será la necesaria para inicializar el SDK en el código Java y otra para vincular a la autenticación de Firebase. Además debemos de dar permiso de autenticación y uso del SDK en el archivo manifest de nuestro proyecto en Android Studio.

 Facebook

Habilitar 

App ID


1499383623464404

App Secret

8c8c2446135dcf4360e5388504a65ddc

Para completar la configuración, añade este URI de redirección de OAuth a la configuración de tu aplicación de Facebook. [Más información](#)

https://fir-app-ae63e.firebaseio.com/_/auth/handler



CANCELAR

GUARDAR

Figura 9: Facebook Auth

Una vez realizada la configuración solo nos queda utilizar los métodos necesarios para realizar la autenticación:

1. Inicializar SDK
2. Si la sesión está iniciada, recoger el usuario de Firebase y mostrarlo.
3. Si no está iniciada, habilitamos el botón de registrarse con Facebook.
4. Al pulsar, aparecerá un OAuth de Facebook para introducir nuestras credenciales
5. Recuperamos los datos de la vista de autenticación y los manejamos.

Cabe destacar que hay que habilitar en Firebase la API de OAuth para nuestra aplicación, de tal forma que sea capaz de generar métodos que puedan recibir y gestionar la devolución de un cliente de autenticación externo.

Credenciales

Credenciales

Pantalla de autorización de OAuth

Verificación de dominio

Crear credenciales

Eliminar

Usa una de estas credenciales para acceder a esta API, o bien crea nuevas credenciales en la parte superior. [Consulta la documentación de la API](#) para obtener más información.

IDs de cliente de OAuth 2.0




<input type="checkbox"/>	Nombre	Fecha de creación	Tipo	ID de cliente	
<input type="checkbox"/>	Android client for com.drassapps.firebaseio.com (auto created by Google Service)	9 dic. 2017	Android		 
<input type="checkbox"/>	Web client (auto created by Google Service)	4 dic. 2017	Web		 

Figura 10: Credenciales de la API de Google

3.2.4. GoogleID

Al igual que ocurre con Facebook, Firebase también nos ofrece la posibilidad de autenticarnos con nuestra cuenta de Google. Al ser las dos de la misma empresa, ofrecen un SDK muy sencillo y fácil de configurar, simplemente necesitaremos lanzar una actividad especial, que se parece al OAuth de Facebook, y recogeremos los datos que el OAuth nos ha proporcionado, para a continuación registrarnos con la dirección de correo.

3.2.5. Twitter

El quinto método de autenticación para nuestra aplicación es a través de la cuenta de Twitter, donde los pasos a realizar son muy similares a la autenticación por Facebook. Cabe destacar, que la consola de Twitter developers no requiere de un par de claves HASH.

A continuación, vamos a ver el método llamado en la actividad para realizar la autenticación de la sesión de Twitter que recibe la vista tras lanzar el OAuth.

```
// Al pulsar el botón se abre el OAuth de Facebook para logearnos con nuestra cuenta, si la
// autenticación de Facebook es correcta, nos devuelve un token que utilizaremos para
// el registro en el Firebase
private void handleTwitterSession(TwitterSession session) {

    // Obtenemos las credenciales del usuario registrado a través del OAuth de Twitter
    AuthCredential credential = TwitterAuthProvider.getCredential(
        session.getAuthToken().token,
        session.getAuthToken().secret);

    mAuth.signInWithCredential(credential)
        .addOnCompleteListener(this, (task) -> {
            if (task.isSuccessful()) {
                // Registro correcto, actualizamos la UI
                Log.d(TAG, "signInWithCredential:success");
                FirebaseUser user = mAuth.getCurrentUser();
                updateUI(user);
            } else {
                // Si falla el login damos un mensaje al usuario
                Log.w(TAG, "signInWithCredential:failure", task.getException());
                setSnackBar(mLayout, "Error en el registro");
                updateUI(null);
            }
        });
}
```

Figura 11: Crea usuario con Sesión de Twitter

3.3. Notificaciones PUSH

Una tecnología emergente y utilizada prácticamente en todas las aplicaciones, son las notificaciones PUSH que a diferencia de las notificaciones normales, estas se envían desde un servidor. Nos permiten tener al usuario informado en todo momento,

además de mejorar su experiencia de usuario, también son útiles para realizar campañas de marketing específicas.

En este proyecto se ha tenido en cuenta 3 formas distintas de enviar una notificación al usuario.

3.3.1. Consola Firebase

La primera forma y más sencilla es a través de la propia consola de Firebase, que nos ofrece una vista con una serie de campos a completar como, por ejemplo, título de la notificación, el mensaje, a que usuarios queremos que les llegue, a que hora... Como es evidente esta forma de enviar notificaciones, no puede ser automáticas cuando ocurre un evento, si no que un administrador se tiene que encargar de crearlas, costando tiempo y dinero a la empresa, además de ser un menú complejo para alguien que no es del sector de la informática.

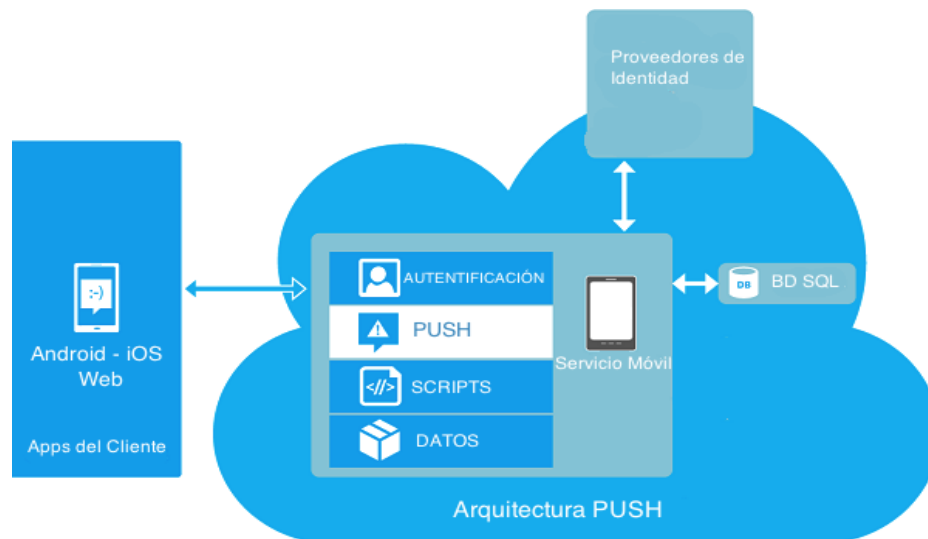


Figura 12: Arquitectura PUSH

3.3.2. Propio dispositivo

La segunda forma de enviar una notificación que se ha pensado es, enviar se a nosotros mismos una notificación de forma inmediata. Si bien carece de sentido, puede ser útil para agendas programando notificaciones para nosotros mismos en un determinado momento. Para implementar esta clase es necesario un Backend que reciba un mínimo de parámetros para crear la notificación.

Es por ello que la clase pide al usuario un título y un mensaje, y los envía junto con el token de notificación del dispositivo a un Backend desarrollado para este proyecto. El Backend recibe los datos y se comunica con el servidor de Firebase que es quien tiene

la capacidad de enviar las notificaciones al dispositivo. En la siguiente imagen se puede ver cómo se crea la notificación que recibirá el usuario.

```
public void recibirPush(RemoteMessage remoteMessage){
    // Crea y muestra una notificación enviada a través de FCM
    NotificationCompat.Builder builder = new NotificationCompat.Builder(this)
        .setSmallIcon(R.drawable.firebase)
        .setContentTitle(remoteMessage.getNotification().getTitle())
        .setContentText(remoteMessage.getNotification().getBody());
    NotificationManager manager = (NotificationManager) getSystemService(NOTIFICATION_SERVICE);
    manager.notify(0, builder.build());
}
```

Figura 13: Notificación PUSH

3.3.3. BackEnd AdHoc

La tercera forma de enviar notificaciones push que se ha implementado es a través de un panel de administrador. Por lo tanto, se ha creado una interfaz de usuario para el BackEnd que gestiona las notificaciones al propio dispositivo. Este panel se asemeja al de la consola de Firebase, que tiene sentido cuando una empresa ya tiene un BackEnd propio, es por ello por lo que nuestro trabajo es integrar nuestro servicio en el suyo, no hacerles cambiar la forma de trabajar. A continuación, se puede observar la función que recoge la información de la interfaz y la envía a los servidores de Firebase.

```
public function enviarNotificacionUI(Request $request){
    // KEY de la aplicación de Firebase
    $API_ACCESS_KEY = "AAAAqJxe81s:APA91bHeGAoqUM71Djs7KujS7BigJ8rucSDSsk6oqSy2PYuaIPASjEiiu8a7_16vRRqR3ctDbeluyIOTojr8u_p1PtP9c0qBP0RMmHX1CfVgbQpWMIKSXL1e1R1fTWbNpHnAxhafezI";

    // Cabeceras para la petición curl
    $headers = array(
        'Authorization: key=' . $API_ACCESS_KEY,
        'Content-Type: application/json'
    );

    // Recogemos los valores introducidos por el administrador
    $titulo = $request->input('titulo');
    $mensaje = $request->input('descripcion');

    // Recogemos los tokens de nuestra BD
    $tokens = DB::table('Datos')->select('Token')->get();

    foreach($tokens as $token)
    { echo $token->Token; }

    // Inicializamos la petición al servidor de Firebase y establecemos los valores
    $ch = curl_init();
    curl_setopt($ch, CURLOPT_URL, 'https://fcm.googleapis.com/fcm/send');
    curl_setopt($ch, CURLOPT_POST, 1);
    curl_setopt($ch, CURLOPT_HTTPHEADER, $headers);
    curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, false);
    curl_setopt($ch, CURLOPT_POSTFIELDS, "
    {
        \"notification\":
        {
            \"title\": \"$titulo\",
            \"text\": \"$mensaje\"
        },
        \"to\" : \"$token\"
    }
    ");
    curl_exec($ch);
    curl_close($ch);
}
```

Figura 14: Función enviarNotificacionUI

3.4. Base de Datos

Firestore ofrece de forma gratuita una base de datos en tiempo real. La BD tiene un formato JSON y su principal característica es que no necesitamos tener un método encargado de hacer consultas constantes a la BD ya que al ser en tiempo real, al modificar un elemento desde la aplicación o desde la propia consola, el usuario percibirá los cambios automáticamente.

En el código solo necesitaremos un Listener que hace referencia al objeto de la BD creado y comprueba si se han cambiado los datos, si se modifican podemos gestionar este evento, como por ejemplo, mostrando los cambios en la UI.

3.5. Almacenamiento

Otra herramienta que Firebase pone a nuestra disposición es el almacenamiento, que no hay que confundirlo con la BD pues la BD en tiempo real solo almacena datos como cadenas de texto, números, booleanos... Mientras que en el almacenamiento se guardan todos los archivos como documentos, imágenes, música...

En la clase implementada podemos subir una imagen al almacenamiento, así como descargar una imagen ubicada en Firebase.

3.6. AdMob

Además de las herramientas que aportan un gran valor a la aplicación, también tenemos herramientas que permiten ganar dinero al desarrollador mostrando anuncios de terceros a cambio de dinero por tasa de visualización, y por tasas de banners de publicidad pulsados.

En el proyecto se han implementado tres tipos diferentes de publicidad:

1. Los banners publicitarios que son fijos y no impiden el buen desarrollo de la aplicación. Se suelen ubicar en la parte inferior de la interfaz, donde menos molesta al usuario.
2. Los banners intersticiales que son aquellos de pantalla completa que debemos esperar un tiempo o simplemente pulsar el botón de cerrar en la parte superior, se suelen incluir en cambios de pantalla dentro de la aplicación.
3. Los videos recompensados, que son videos donde el usuario debe esperar a que termine para recibir una recompensa, en forma de vida de juego o monedas. Y al mismo tiempo el desarrollador gana dinero por que vean el video.

3.7. Crashlytics

Firebase ofrece una herramienta para tener al desarrollador al tanto de los errores que ocurren en la aplicación. Esta herramienta se llama crashlytics una abreviatura de crash (fallo) y Analytics (análisis), por lo que podemos decir que es una herramienta que nos ofrece un análisis de los fallos.

Es de gran ayuda, ya que nos envía automáticamente un email al correo asociado a la cuenta de Firebase indicándonos en que clase y en que línea se ha producido el crash, además que facilita, simplifica y agiliza la hora de depuración en dispositivos móviles reales.

3.8. Invite Links

Por último, nos encontramos con los invite links, que no es otra cosa que unos links de invitación personalizados que nos permiten enviar nuestra aplicación bajo dos supuestos.

Si no tenemos la aplicación descargada, nos enviará a nuestra aplicación en GooglePlay, para que podamos descargar la aplicación. Si ya tenemos la aplicación descargada, podemos enviar links específicos de tal forma que cuando el usuario la habrá accederá a una vista de la aplicación sin tener que recorrer las demás vistas.

Evidentemente, también sirve para recomendar la aplicación a nuestros contactos, familiares, amigos... y que de esta forma se haga famosa.

4. Conclusiones

El código presentado está documentado de forma completa, indicando la funcionalidad de cada método, así como la integración con Firebase, explicando los pasos seguidos y para que se utiliza. Para el acceso a Firebase y al BackEnd desarrollado se han habilitado unas cuentas especiales.

Usuario para acceder a Firebase en modo propietario, pero solo con permisos de lectura:

1. Acceder a Google con el siguiente usuario:
Usuario: ***drassappsfirebaseapp@gmail.com***
Contraseña: ***CniHwmzi2up6vBz***
2. Acceder a la dirección: <https://console.firebase.google.com/?pli=1>
3. Podremos ver el proyecto creado FirebaseApp

Usuario para acceder al BackEnd para probar la funcionalidad de envío de notificaciones

1. Acceder a la dirección: <http://drassapps.es/public>

Usuario: **Andres**

Contraseña: **CniHwmzi2up6vBz**