

Firestore

Desarrollo de aplicaciones móviles

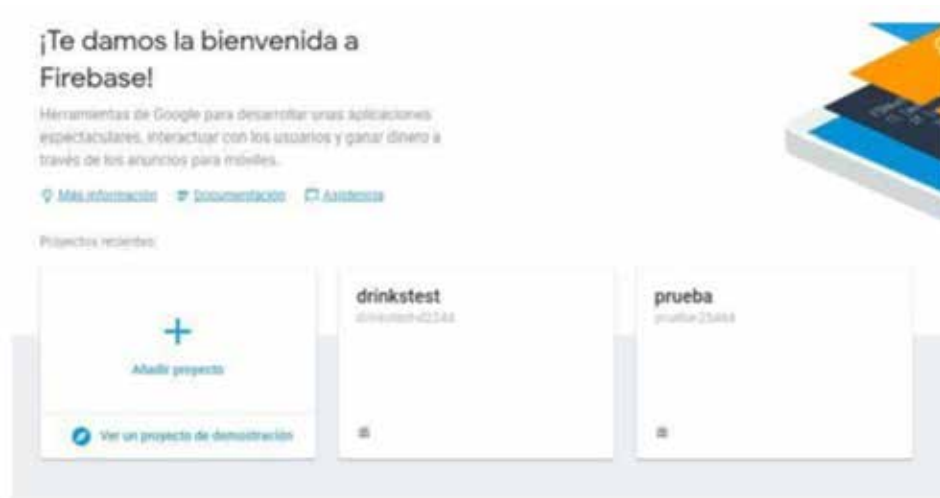
Servicio Nacional de Aprendizaje - SENA

Firestore

A continuación, veremos un ejemplo para incorporar en nuestro proyecto todos estos servicios.

Prerrequisitos: Tener un proyecto Android con un activity y su respectivo layout.

1. Ahora agregaremos Firebase a nuestro proyecto, para esto debemos entrar a la consola de Firebase (si no tiene cuenta favor registrarse) : <https://console.firebase.google.com/>

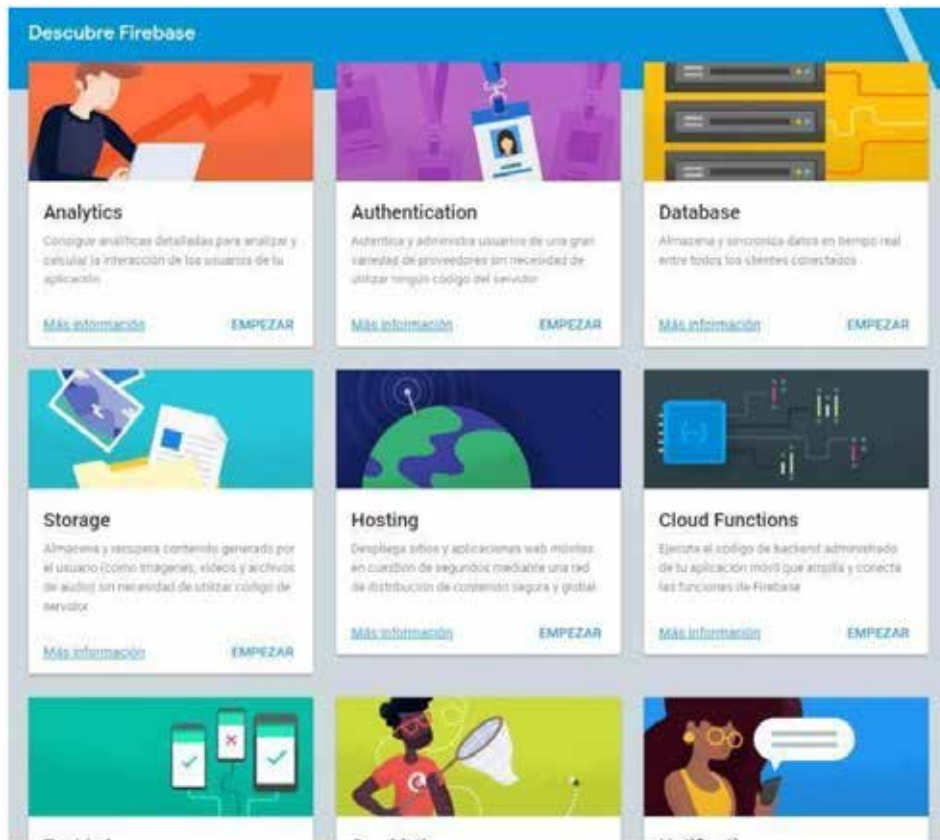


Vamos a seleccionar la opción **Añadir proyecto:**

A continuación, se muestra un formulario donde debemos poner el nombre de nuestro proyecto, en el caso de la guía será "sena2017" y país Colombia, seguimos con el botón Crear Proyecto:



Podemos ver todos los servicios que incluye Firebase:



Seleccionamos la opción “Añade Firebase a tu aplicación de Android”



Nombre del paquete de Android ?

com.yourapp.android

Apodo de la aplicación (opcional) ?

Freemium Android App




Y el apodo es opcional y puede ser cualquier texto:



A continuación descargamos el archivo Google-services.json en el botón indicado

Instrucciones para Android Studio Alternativas: [Unity](#) [C++](#)

1. [Descargar google-services.json](#)
2. Cambia a la vista **Proyecto** de Android Studio para ver el directorio "root" de tu proyecto.
3. Mueve el archivo **google-services.json** que acabas de descargar al directorio "root" del módulo de tu aplicación de Android.

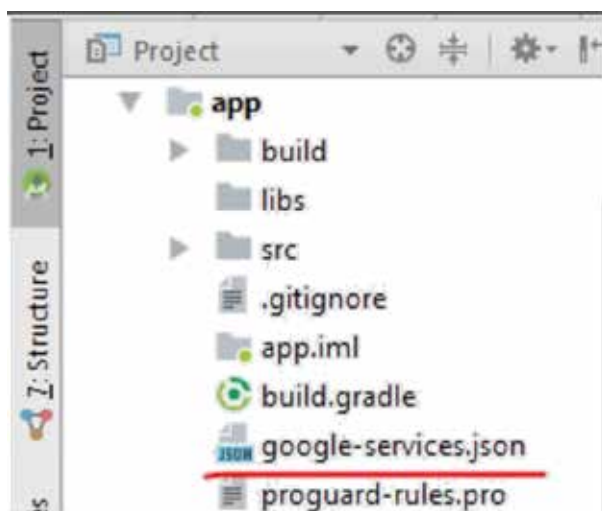


¿Ya has añadido las dependencias?
[Saltar a la consola](#)

CONTINUAR

Cuando lo hayamos descargado damos continuar y finalizar.

Nos queda agregar el archivo descargado dentro de la raíz de la carpeta app:



Con esto podemos sincronizar o actualizar el proyecto y ya habremos agregado Firebase. Observe que ya nos encontramos en el proyecto y podemos agregar los diferentes servicios:



2. Agreguemos analítica a nuestra app: Declaramos una variable `FirebaseAnalytics` en nuestra actividad: /

```
public class FireActivity extends AppCompatActivity {
    private FirebaseAnalytics mFirebaseAnalytics;
```

En el `onCreate()`, inicializamos la variable:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_fire);

    mFirebaseAnalytics = FirebaseAnalytics.getInstance(this);
}
```

Con el objeto de Analítica inicializado, podemos registrar eventos con el método `logEvent`, creamos un botón de ejemplo y a continuación agregamos el registro del evento cuando un usuario le da click al botón:

```

<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="ekmilo.sena20172.FireActivity">

    <Button
        android:id="@+id/button" android:text="click"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

</android.support.constraint.ConstraintLayout>

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_fire);
    Button click = findViewById(R.id.button);
    mFirebaseAnalytics = FirebaseAnalytics.getInstance(this);

    Bundle bundle = new Bundle();
    bundle.putString(FirebaseAnalytics.Param.ITEM_ID, String.valueOf(R.id.button));
    bundle.putString(FirebaseAnalytics.Param.ITEM_NAME, "click");
    bundle.putString(FirebaseAnalytics.Param.CONTENT_TYPE, "button");
    mFirebaseAnalytics.logEvent(FirebaseAnalytics.Event.SELECT_CONTENT, bundle);
}

```

Tras correrla y darle click al botón podemos ver en la consola de Firebase en nuestro proyecto:



En un plazo de 24 horas:

3. Agreguemos ahora notificaciones o mensajería Cloud:

Agregamos la dependencia para mensajería de Firebase en el gradle de nuestra aplicación:

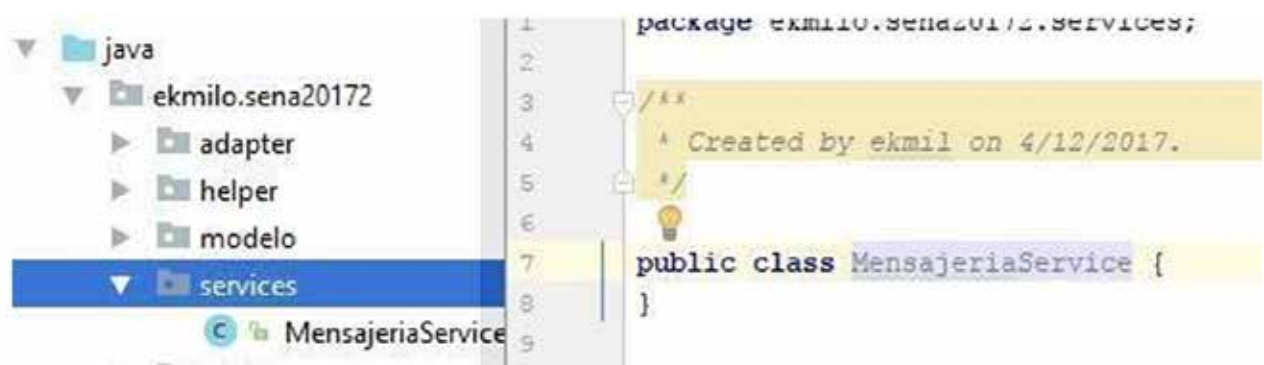
```
dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation "org.jetbrains.kotlin:kotlin-stdlib-jre7:$kotlin_version"
    implementation 'com.android.support:appcompat-v7:26.1.0'
    implementation 'com.android.support.constraint:constraint-layout:1.0.2'
    implementation 'com.squareup.retrofit2:retrofit:2.0.2'
    implementation 'com.squareup.retrofit2:converter-gson:2.0.2'
    implementation 'com.android.support:recyclerview-v7:26.1.0'
    implementation 'com.jakewharton:butterknife:8.1.0'

    implementation 'com.google.firebase:firebase-core:11.0.4'
    implementation 'com.google.firebase:firebase-messaging:11.0.4'

    testImplementation 'junit:junit:4.12'
}
```

Crearemos un servicio (Los servicios los veremos en otra guía con mayor detalle, por ahora tener en cuenta que un servicio es un componente de la aplicación que puede ejecutar operaciones de larga ejecución en background y no provee una interfaz de usuario, pese a que el usuario cambie de aplicación. Adicionalmente, otros componentes pueden conectarse con el servicio e interactuar realizando comunicaciones, por ejemplo un servicio puede manejar transacciones en la red o reproducir música.)

Para crear el servicio creamos una clase en el paquete servicios:



Extendemos del Servicio de Firebase sobrescribiendo el método `onMessageReceived`:


```
public class MensajeriaService extends FirebaseMessagingService {
    private static final String TAG = "FCM Service";
    @Override
    public void onMessageReceived(RemoteMessage remoteMessage) {
        // TODO: Handle FCM messages here.
        // If the application is in the foreground handle both data and notification messages here.
        // Also if you intend on generating your own notifications as a result of a received FCM
        // message, here is where that should be initiated.
        Log.d(TAG, msg: "From: " + remoteMessage.getFrom());
        Log.d(TAG, msg: "Notification Message Body: " + remoteMessage.getNotification().getBody());
    }
}
```

Agregamos el servicio a nuestro manifest:

```
<application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportRtl="true"
    android:theme="@style/AppTheme">
    <activity android:name=".MainActivity" />
    <activity android:name=".Main2Activity" />
    <activity android:name=".CursorActivity" />
    <activity android:name=".AsyncTaskActivity"></activity>
    <activity android:name=".RetrofitActivity">
    </activity>
    <activity android:name=".FireActivity">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
    <service android:name=".services.MensajeriaService">
        <intent-filter>
            <action android:name="com.google.firebase.MESSAGING_EVENT"/>
        </intent-filter>
    </service>
</application>
```

Agregamos otro servicio que herede del servicio FirebaseInstanceIdService, sobrescribiendo el método onTokenRefresh.

```

public class FirebaseIdService extends FirebaseInstanceIdService {
    private static final String TAG = "FirebaseIdService";
    @Override
    public void onTokenRefresh() {
        // Get updated InstanceID token.
        String refreshedToken = FirebaseInstanceId.getInstance().getToken();
        Log.d(TAG, "Refreshed token: " + refreshedToken);

        // TODO: Implement this method to send any registration to your app's servers.
        sendRegistrationToServer(refreshedToken);
    }
    /**
     * Persist token to third-party servers.
     *
     * Modify this method to associate the user's FCM InstanceID token with any server-side account
     * maintained by your application.
     *
     * @param token The new token.
     */
    private void sendRegistrationToServer(String token) {
        // Add custom implementation, as needed.
    }
}

```

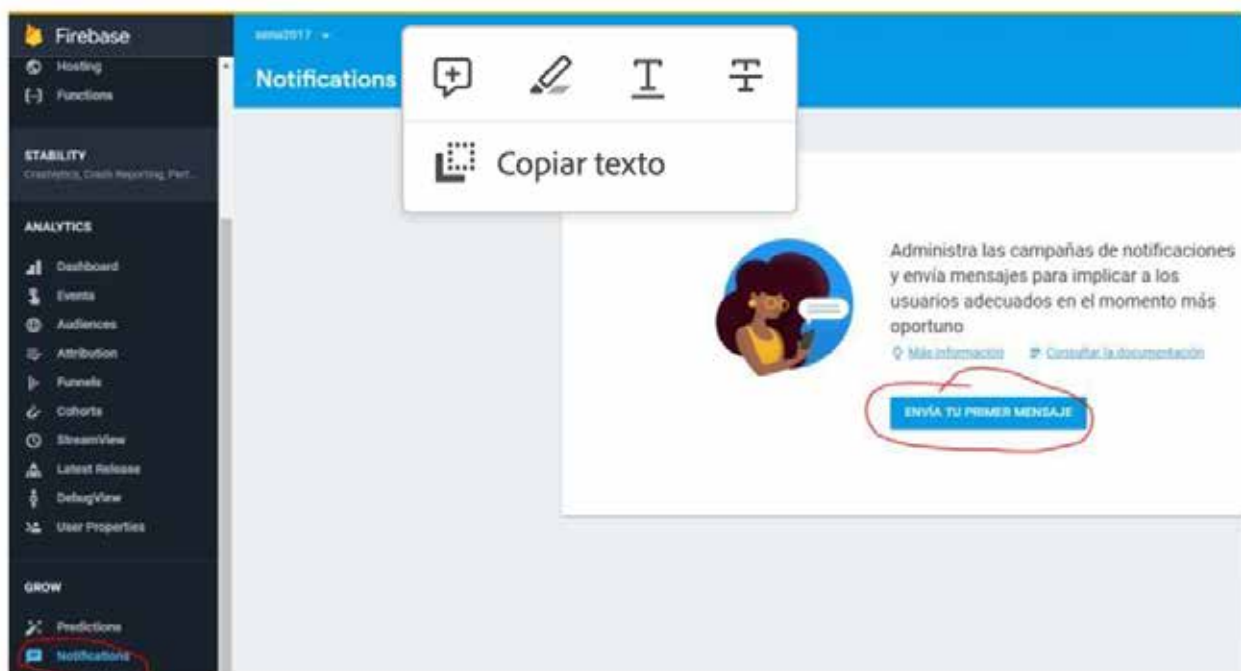
Agregamos este servicio también al manifest de nuestro proyecto: :

```

</service>
<service android:name=".services.FirebaseIdService">
    <intent-filter>
        <action android:name="com.google.firebase.INSTANCE_ID_EVENT"/>
    </intent-filter>
</service>
</application>

```

Actualizamos la aplicación en el dispositivo móvil e incluso podemos cerrarla, y desde la consola de Firebase podemos enviar un mensaje:



Texto del mensaje

Hola mundo

Etiqueta del mensaje (opcional) 📌

alias

Fecha de entrega 🕒

Enviar aho... ▼

Destino

☒ Segmento de usuarios ☐ Tema ☐ Un único dispositivo

Dirigir al usuario si...

Aplicación ekmilo.sena20172 ▼ Y

No se pueden añadir declaraciones adicionales. Se han seleccionado todas las aplicaciones.

📊 Eventos de conversión 📌 ▼

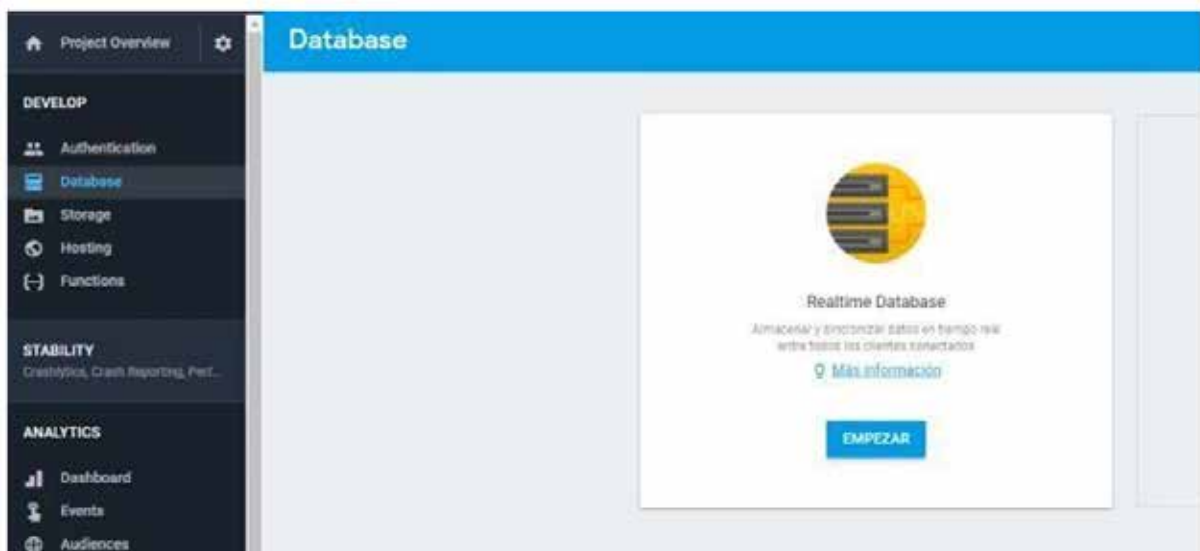
Opciones avanzadas ▼

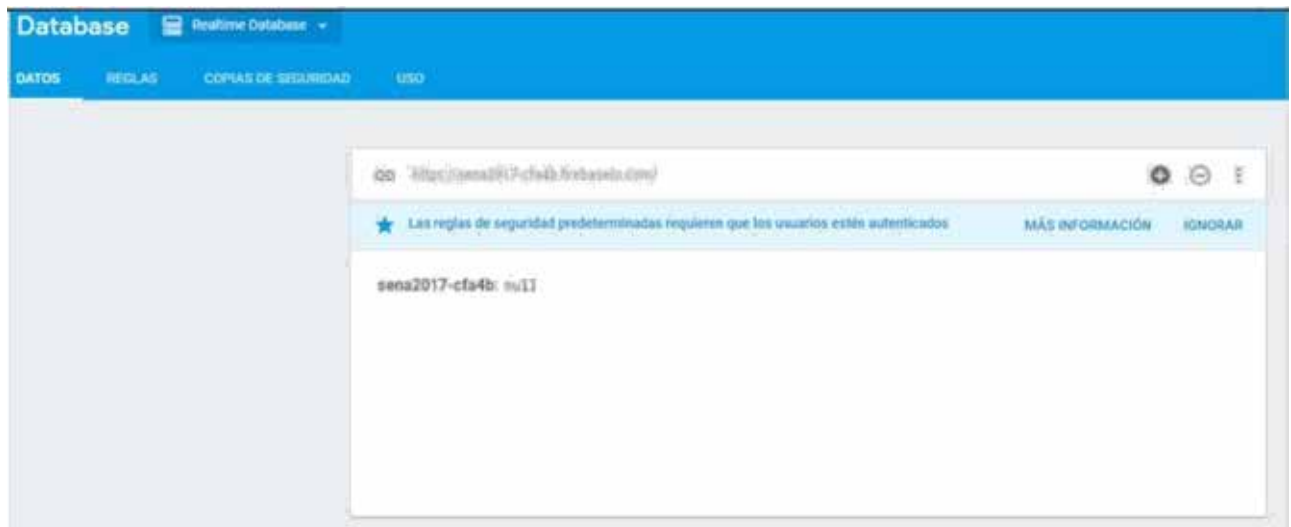
GUARDAR COMO BORRADOR **ENVIAR MENSAJE**

Llegando a nuestro dispositivo:



4. Agregando Bases de datos en tiempo real: Firebase Realtime Database te permite compilar aplicaciones ricas y colaborativas, ya que permite el acceso seguro a la base de datos directamente desde el código del cliente. Los datos persisten de forma local. Además, incluso cuando no hay conexión, los eventos en tiempo real se siguen activando, lo que proporciona al usuario final una experiencia adaptable. Cuando el dispositivo vuelve a conectarse, Realtime Database sincroniza los cambios de los datos locales con las actualizaciones remotas que ocurrieron mientras el cliente estuvo sin conexión, lo que combina los conflictos de forma automática.

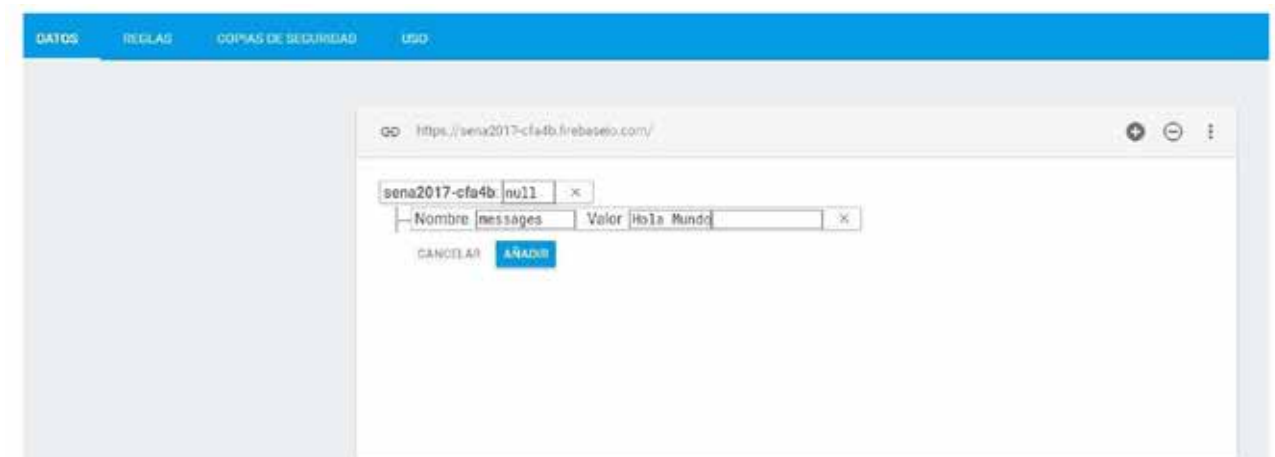




Primero actualizamos las reglas para que un usuario pueda leer o escribir sobre la base de datos sin que esté autenticado:



En la pestaña de datos,



Dando añadir



Agregamos ahora la dependencia para manejo de base de datos en nuestro gradle:



Ahora en nuestra actividad, agregamos un TextView en el layout para leer los datos de internet



En la actividad, agregamos las variables necesarias, en el onCreate: las inicializamos y en el Listener del botón agregamos la lectura con Firebase:

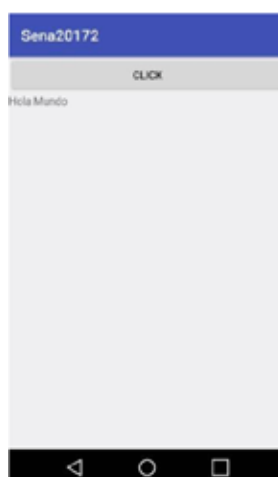
```
public class FireActivity extends AppCompatActivity {
    private FirebaseAnalytics mFirebaseAnalytics;
    private DatabaseReference myRef;
    private FirebaseDatabase database;

    @Override
```

En el onCreate

```
Bundle bundle = new Bundle();
bundle.putString(FirebaseAnalytics.Param.ITEM_ID, String.valueOf(R.id.button));
bundle.putString(FirebaseAnalytics.Param.ITEM_NAME, "click");
bundle.putString(FirebaseAnalytics.Param.CONTENT_TYPE, "button");
mFirebaseAnalytics.logEvent(FirebaseAnalytics.Event.SELECT_CONTENT, bundle);
database = FirebaseDatabase.getInstance();
myRef = database.getReference().child("messages");
click.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        myRef.addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(DataSnapshot dataSnapshot) {
                String value = dataSnapshot.getValue(String.class);
                text.setText(value);
            }
            @Override
            public void onCancelled(DatabaseError error) {
                // Failed to read value
                Log.w("tag:", "msg: "Failed to read value.", error.toException());
            }
        });
    }
});
```

Generando cuando damos click al botón:



Hay muchos otros servicios que puedes revisar e incorporar en tus proyectos, para mayor información:

<https://firebase.google.com/docs/guides/?hl=es-419>