



INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO



Application Development for Mobile Devices

Chat Firebase

González López Emiliano
Patiño Cipriano Jesús Abraham
Trinidad León Roberto Antonio
Vásquez Blanco Carlos Alberto

19/junio/2020

Contenido

Firestore	3
Desarrollo.....	4
Creación del proyecto	4
Dependencias	6
Definir diseño	6
Gestiona la autenticación del usuario	7
Paso 1: Gestiona el inicio de sesión del usuario	7
Paso 2: Gestiona el cierre de sesión del usuario	9
Creación de un modelo para los mensajes	10
Publica un mensaje de chat.....	11
Muestra los mensajes de chat.....	12
Pruebas	14

Firestore

Firestore es la nueva y mejorada plataforma de desarrollo móvil en la nube de Google. Se trata de una plataforma disponible para diferentes plataformas (Android, iOS, web), con lo que de esta forma presentan una alternativa seria a otras opciones para ahorro de tiempo en el desarrollo como Xamarin.

¿En qué consiste Firestore? Es la evolución de una plataforma que ha ido mejorando desde que Google la compró en 2014 y luego la continuó mejorando con la compra del equipo de Firebase.

En muchas ocasiones nos planteamos cómo poder acceder a un servicio web para poder tener nuestra aplicación trabajando con datos en la nube. Por ello surgió Firestore, para proveer una API para guardar y sincronizar datos en la nube en tiempo real.

Sus características fundamentales están divididas en varios grupos, las cuales podemos agrupar en:

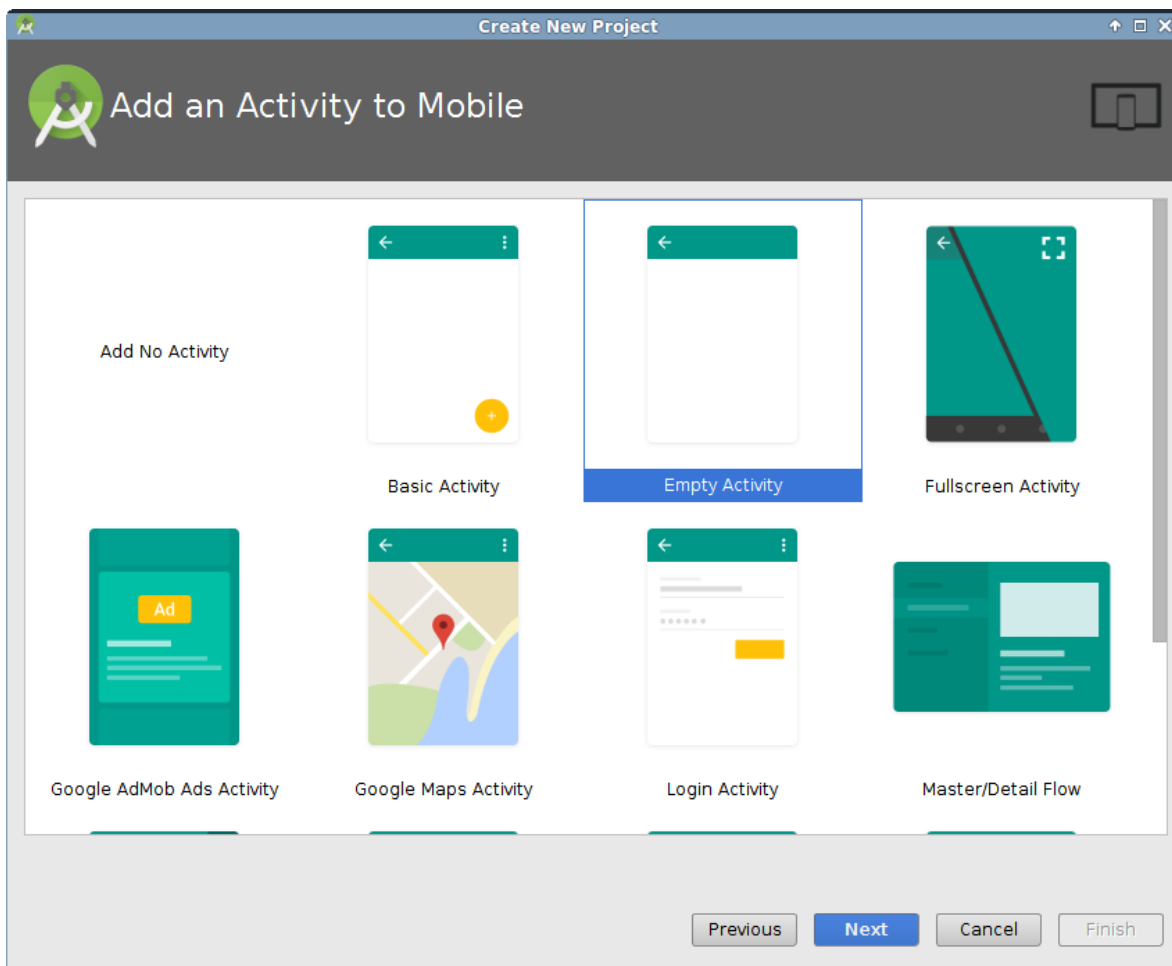
- **Análíticas:** Provee una solución gratuita para tener todo tipo de medidas (hasta 500 tipos de eventos), para gestionarlo todo desde un único panel.
- **Desarrollo:** Permite construir mejores apps, permitiendo delegar determinadas operaciones en Firestore, para poder ahorrar tiempo, evitar bugs y obtener un aceptable nivel de calidad. Entre sus características destacan el almacenamiento, testeo, configuración remota, mensajería en la nube o autenticación, entre otras.
- **Crecimiento:** Permite gestionar los usuarios de las aplicaciones, pudiendo además captar nuevos. Para ello dispondremos de funcionalidades como las de invitaciones, indexación o notificaciones.
- **Monetización:** Permite ganar dinero gracias a AdMob.

Con Firestore, crear aplicaciones de redes sociales en tiempo real es un trabajo sencillo. Y lo mejor de todo: no se tiene que escribir una sola línea de código del lado del servidor, ya que todo el servicio es provisto por la API de firestore.

Desarrollo

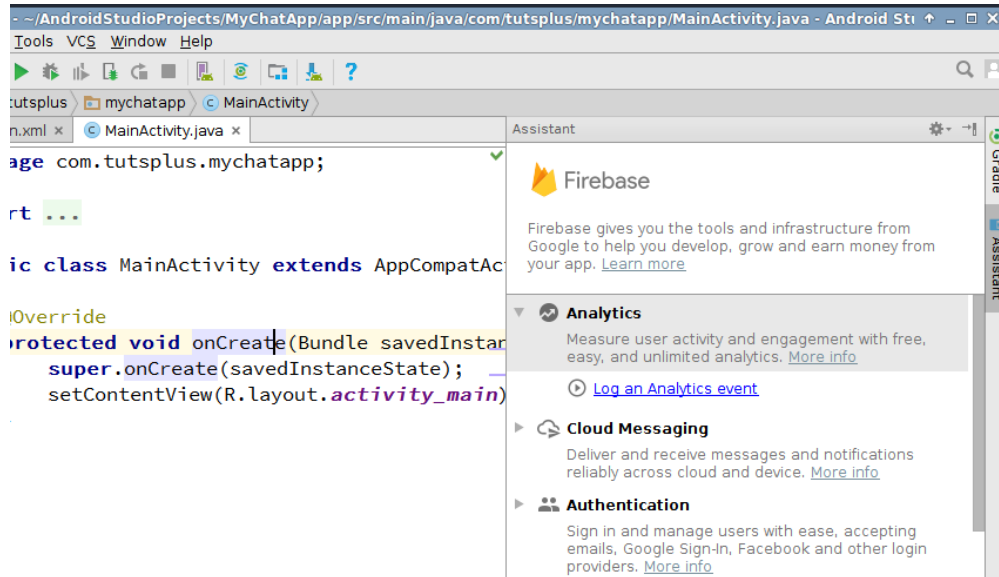
Creación del proyecto.

Ejecuta Android Studio y crea un nuevo proyecto con una actividad vacía llamada MainActivity.

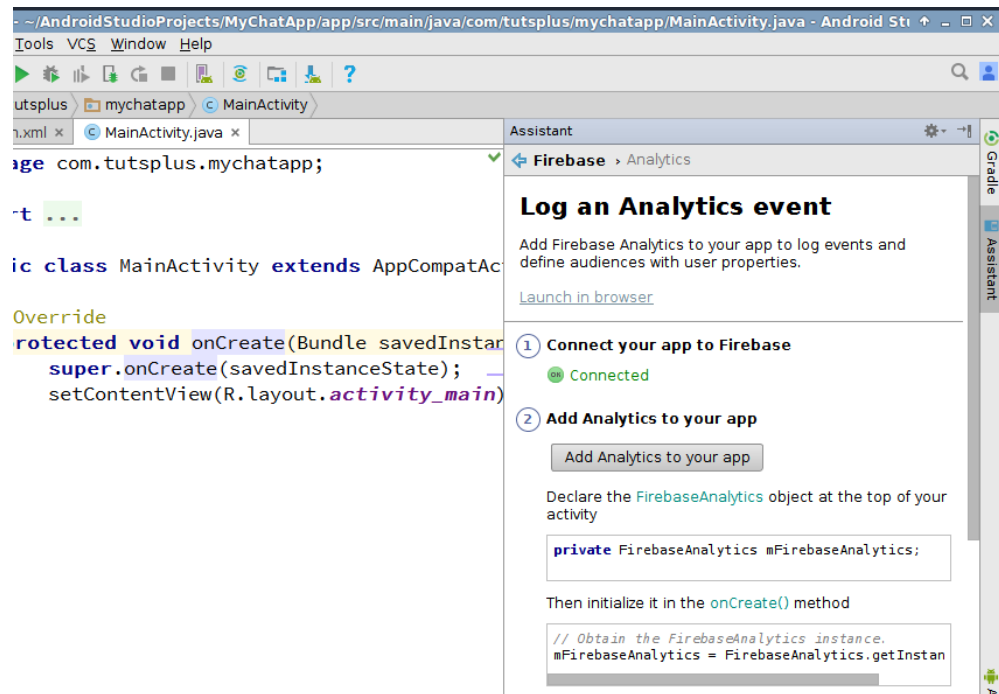


Para configurar el proyecto para que use la plataforma Firebase, abre la ventana del Asistente de Firebase haciendo clic en Tools > Firebase (Herramientas > Firebase).

Al usar la plataforma Firebase por lo general es una buena idea añadir Firebase Analytics al proyecto. Por lo tanto, dentro de la ventana del Asistente de Firebase ve a la sección Analytics y presiona la opción Log an Analytics event (Registrar un evento de Analytics).



A continuación, presiona el botón Connect to Firebase (Conectar con Firebase) y asegúrate de que la opción Create new Firebase project (Crear nuevo proyecto de Firebase) esté seleccionada. Una vez que la conexión haya sido establecida presiona el botón Add Analytics to your app (Añadir Analytics a tu aplicación).



Dependencias

Vamos a usar dos bibliotecas en este proyecto: Firebase UI y la biblioteca de soporte para diseño de Android. Por lo tanto, abre el archivo build.gradle del módulo app y agrega las siguientes dependencias compile:

```
1 compile 'com.android.support.design:23.4.0'
2 compile 'com.firebaseui:firebase-ui:0.6.0'
```

Y procedemos a sincronizar el proyecto.

Definir diseño

El archivo activity_main.xml, que ya está vinculado con MainActivity, define los contenidos de la pantalla de inicio de la aplicación. En otras palabras, representará a la sala de chat.

Al igual que la mayoría del resto de las aplicaciones de grupos de chat disponibles hoy en día, nuestra aplicación tendrá los siguientes elementos en la interfaz de usuario:

- Una lista que muestre todos los mensajes del grupo de chat en orden cronológico
- Un campo de entrada en el que el usuario pueda escribir un mensaje nuevo
- Un botón que el usuario pueda presionar para publicar el mensaje

Por lo tanto, activity_main.xml debe tener un ListView, un EditText y un FloatingActionButton. Después de colocarlos dentro de un widget RelativeLayout.

Ahora que el diseño de la pantalla de inicio está listo podemos continuar creando un diseño para los mensajes del chat, que son elementos contenidos en un ListView. Comienza creando un nuevo archivo de diseño XML llamado message.xml, cuyo elemento raíz sea RelativeLayout.

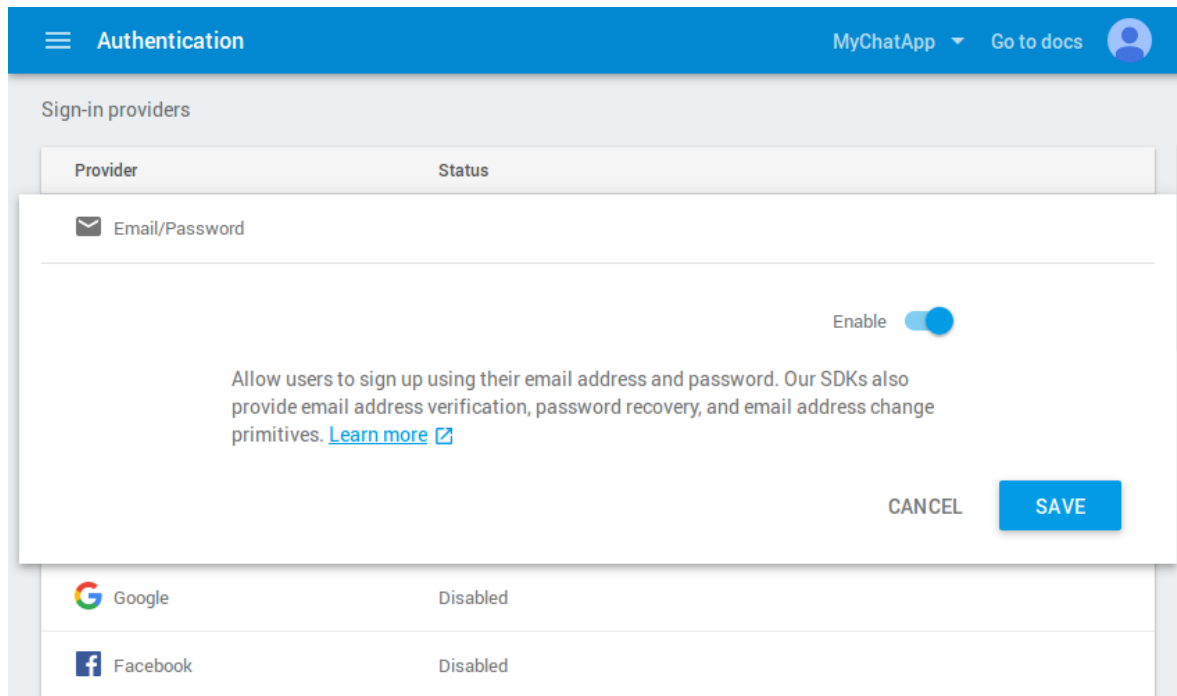
El diseño debe tener widgets TextView para mostrar el texto del mensaje del chat, la hora a la que fue enviado y su autor.

Gestiona la autenticación del usuario

Permitir que los usuarios publiquen mensajes de forma anónima en la sala de chat sería una muy mala idea. Esto podría ocasionar spam, problemas de seguridad y una experiencia de chat poco deseable para los usuarios. Por lo tanto, vamos a configurar nuestra aplicación para que solamente los usuarios registrados puedan leer y publicar mensajes.

Comienza yendo a la sección Auth (Autorización) de la Consola de Firebase y habilitando Email/Password (Correo/Contraseña) como proveedor de inicio de sesión.

Al ser un servicio provisto por Google todo esto será realmente sencillo y solo tendremos que invocar el método de inicio de sesión.



Paso 1: Gestiona el inicio de sesión del usuario

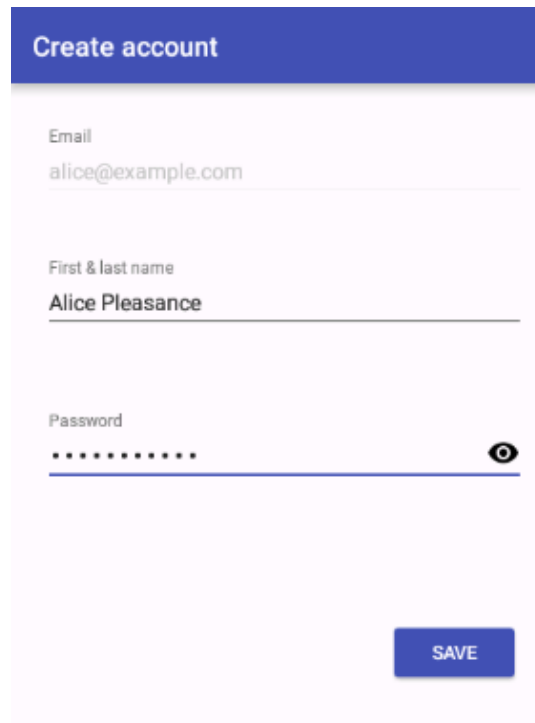
Tan pronto como la aplicación arranque, ésta debe verificar si el usuario ha iniciado sesión. Si es así, la aplicación debe continuar y mostrar el contenido de la sala de chat. De lo contrario, ésta debe redirigir al usuario ya sea a una pantalla de inicio de sesión o a una de registro. Con FirebaseUI, la creación de esas pantallas requiere mucho menos código del que podrías imaginar.

Dentro del método `onCreate()` de `MainActivity`, revisa si el usuario ya inició sesión verificando si el objeto `FirebaseUser` actual no es `null`. Si es `null`, debes crear y configurar un objeto `Intent` que abre una actividad de inicio de sesión. Para hacer esto usa la clase `SignInIntentBuilder`. Una vez que el intent esté listo, debes ejecutar la actividad de inicio de sesión usando el método `startActivityForResult()`.

El código que tendremos que ocupar para la autenticación es el siguiente:

```
if(FirebaseAuth.getInstance().getCurrentUser() == null) {  
    // Start sign in/sign up activity  
    startActivityForResult(  
        AuthUI.getInstance()  
        .createSignInIntentBuilder()  
        .build(),  
        SIGN_IN_REQUEST_CODE  
    );  
} else {  
    // User is already signed in. Therefore, display  
    // a welcome Toast  
    Toast.makeText(this,  
        "Welcome " + FirebaseAuth.getInstance()  
        .getCurrentUser()  
        .getDisplayName(),  
        Toast.LENGTH_LONG)  
        .show();  
  
    // Load chat room contents  
    displayChatMessages();  
}
```

Como puedes ver en el código anterior, si el usuario ya inició sesión primero mostramos un Toast que saludará al usuario, y posteriormente invocamos al método llamado `displayChatMessages`. Por ahora solamente crea el esqueleto del método. Vamos a añadirle código posteriormente.



Paso 2: Gestiona el cierre de sesión del usuario

De forma predeterminada FirebaseUI usa el Bloqueo Inteligente para Contraseñas. Por lo tanto, una vez que los usuarios inicien sesión éstos permanecerán en ella incluso si reinician la aplicación. Para permitir que los usuarios cierren sesión agregaremos la opción correspondiente al menú de desbordamiento de MainActivity.

Crea un nuevo archivo de recursos de menú llamado main_menu.xml y agrégale un elemento item único cuyo atributo title tenga el valor Sign out (Cerrar sesión).

A continuación, sobrescribe el método onOptionsItemSelected() para gestionar los eventos clic en el elemento del menú. Dentro del método puedes invocar a la función signOut() de la clase AuthUI para cerrar la sesión del usuario. Dado que la operación de cierre de sesión se ejecuta de forma asíncrona, también vamos a añadirle un OnCompleteListener.

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    if(item.getItemId() == R.id.menu_sign_out) {
        AuthUI.getInstance().signOut(this)
            .addOnCompleteListener(new OnCompleteListener<Void>() {
                @Override
                public void onComplete(@NonNull Task<Void> task) {
                    Toast.makeText(MainActivity.this,
                        "You have been signed out.",
                        Toast.LENGTH_LONG)
```

```

        .show();

        // Close activity
        finish();
    }
});
}
return true;
}

```

Creación de un modelo para los mensajes

Para almacenar los mensajes de chat en la base de datos de Firebase en tiempo real, debes crear un modelo para ellos. El diseño del mensaje de chat, que creamos anteriormente en este tutorial, tiene tres vistas. Para poder llenar esas vistas, el modelo también debe tener al menos tres campos.

Crea una nueva clase de Java llamada `ChatMessage.java` y agrégale tres variables miembros: `messageText`, `messageUser` y `messageTime`. Además, agrega un constructor para inicializar esas variables.

Para que el modelo sea compatible con FirebaseUI, también debes añadirle un constructor predeterminado junto con funciones `get` y `set` para todas las variables miembro.

```

public class ChatMessage {

    private String messageText;
    private String messageUser;
    private long messageTime;

    public ChatMessage(String messageText, String messageUser) {
        this.messageText = messageText;
        this.messageUser = messageUser;

        // Initialize to current time
        messageTime = new Date().getTime();
    }
}

```

Publica un mensaje de chat

Ahora que el modelo está listo, podemos añadir fácilmente nuevos mensajes de chat a la base de datos de Firebase en tiempo real.

Para publicar un mensaje nuevo el usuario presionará el FloatingActionButton. Por lo tanto debes añadirle un OnClickListener.

Dentro del listener, primero debes obtener un objeto DatabaseReference usando el método `getReference()` de la clase `FirebaseDatabase`. Después puedes hacer llamadas a los métodos `push()` y `setValue()` para añadir nuevas instancias de la clase `ChatMessage` a la base de datos en tiempo real.

Desde luego, las instancias `ChatMessage` deben ser inicializadas usando los contenidos del `EditText` y el nombre para mostrar del usuario registrado actualmente.

En consecuencia, agrega el siguiente código al método `onCreate()`:

```
FloatingActionButton fab =
    (FloatingActionButton)findViewById(R.id.fab);

fab.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        EditText input = (EditText)findViewById(R.id.input);

        // Read the input field and push a new instance
        // of ChatMessage to the Firebase database
        FirebaseDatabase.getInstance()
            .getReference()
            .push()
            .setValue(new ChatMessage(input.getText().toString(),
                FirebaseAuth.getInstance()
                    .getCurrentUser()
                    .getDisplayName())
            );

        // Clear the input
        input.setText("");
    }
});
```

Los datos de la base de datos de Firebase en tiempo real siempre se almacenan como pares clave-valor. Sin embargo, si observas el código anterior podrás ver que estamos invocando a `setValue()` sin especificar una clave. Esto se permite solamente debido a que la llamada al método `setValue()` está precedida por una llamada al método `push()`, quien genera una nueva clave automáticamente.

Muestra los mensajes de chat

FirestoreUI tiene una clase muy útil llamada `FirestoreListAdapter`, que reduce dramáticamente el esfuerzo requerido para llenar un `ListView` usando datos que se encuentran presentes en la base de datos de Firestore en tiempo real. Vamos a usarla ahora para recuperar y mostrar todos los objetos `ChatMessage` que estén presentes en la base de datos.

Añade un objeto `FirestoreListAdapter` como una nueva variable miembro de la clase `MainActivity`.

```
private void displayChatMessages() {

    Log.d(TAG, "+++++");
    reference = FirebaseDatabase.getInstance().getReference();
    Log.d(TAG, "CREADO EL QUERY "+ reference.toString());
    FirebaseFirestoreOptions<ChatMessage> options = new FirebaseFirestoreOptions.Builder<ChatMessage>()
        .setQuery(reference, ChatMessage.class)
        .setLayout(R.layout.message)
        .build();

    //The error said the constructor expected FirebaseFirestoreOptions - here you create them:

    Log.d(TAG, "CREADO OPCIONES ");
    //Finally you pass them to the constructor here:
    ListView listOfMessages = (ListView)findViewById(R.id.list_of_messages);

    adapter = new FirebaseFirestoreListAdapter<ChatMessage>(options){
        @Override
        protected void populateView(View v, ChatMessage model, int position) {
            Log.d(TAG, "INICIO POBLACION");
            // Get references to the views of message.xml
            TextView messageText = (TextView)v.findViewById(R.id.message_text);
            TextView messageUser = (TextView)v.findViewById(R.id.message_user);
            TextView messageTime = (TextView)v.findViewById(R.id.message_time);

            // Set their text
            messageText.setText(model.getMessageText());
            messageUser.setText(model.getMessageUser());

            // Format the date before showing it
            messageTime.setText(DateFormat.format("dd-MM-yyyy (HH:mm:ss)",
                model.getMessageTime()));
        }
    };
}
```

```

        Log.d(TAG, "++++++");
        Log.d(TAG, "Mensaje: " + model.getMessageText());
        Log.d(TAG, "Mensaje: " + model.getMessageUser());
        Log.d(TAG, "Mensaje: " + model.getMessageTime());
    }
};

listOfMessages.setAdapter(adapter);
}

```

También debemos agregar los siguientes métodos para poder tener una correcta actualización de los mensajes.

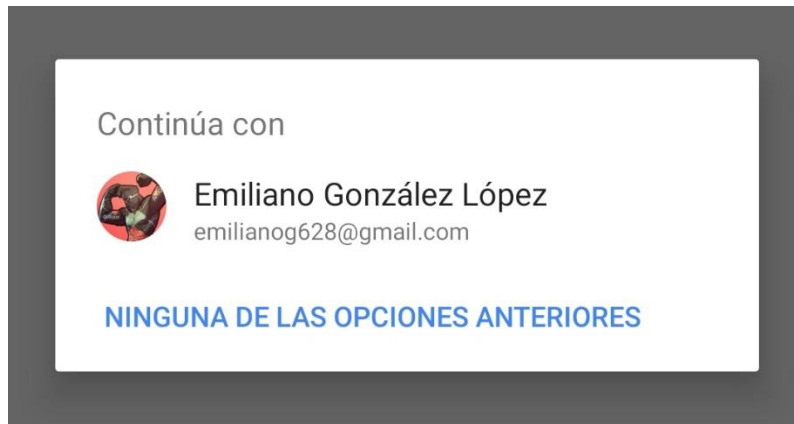
```

@Override
protected void onStart() {
    super.onStart();
    adapter.startListening();
}

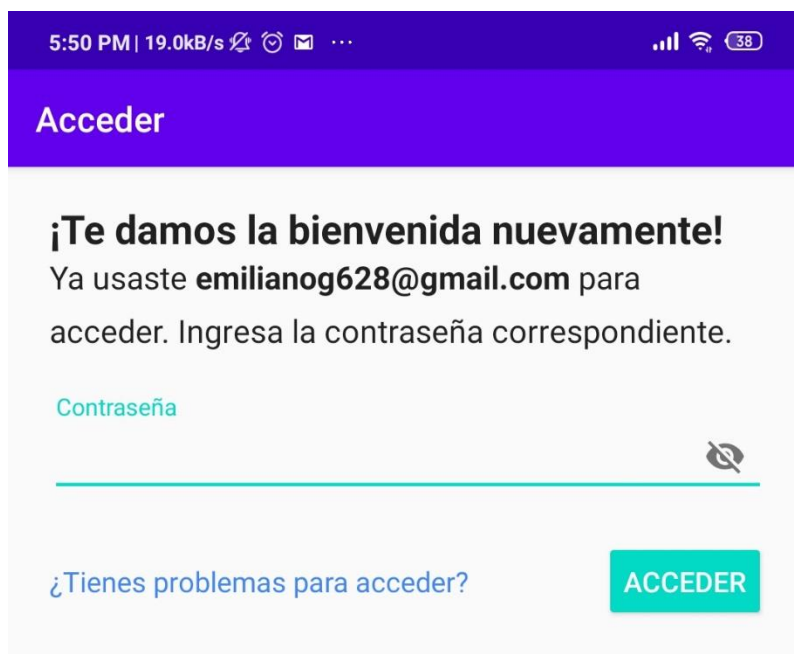
@Override
protected void onStop() {
    super.onStop();
    adapter.stopListening();
}

```

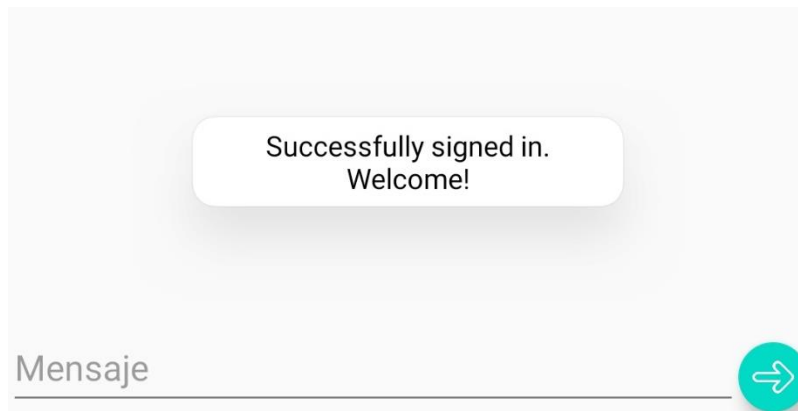
Pruebas



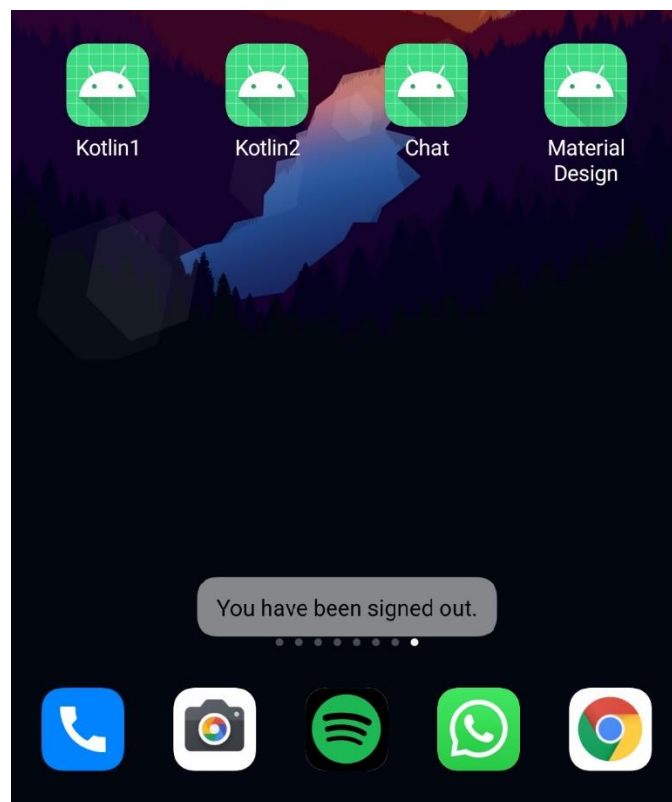
Para comenzar la pagina de inició sugiere ingresar con alguna de las cuentas registradas en el celular, en este caso aparece la cuenta personal de correo.



Si ya tenemos una cuenta registrada nos recordara y solo tendremos que ingresar la contraseña.



Podemos observar cómo es que se muestra el mensaje al iniciar



También podemos ver el mensaje al salir, y como nos saca de la aplicación.

Crash Bandicoot	18-06-2020 (18:31:40)
hola	
Carlos Vásquez	18-06-2020 (19:40:34)
hola	
Carlos Vásquez	18-06-2020 (19:40:43)
hola	
Carlos Vásquez	18-06-2020 (20:10:02)
apenas me cargaron los mensajesxd	

Mensaje



Finalmente vemos los mensajes recuperados de la base de datos de Firebase

```
-MA8IW8A0u_XPQQw1Y71
  |-- messageText: "hola"
  |-- messageTime: 159252310029
  |-- messageUser: "Crash Bandicoot"
-MA90Hf9cfIFvYPfGtBr
  |-- messageText: "hola"
  |-- messageTime: 159252723486
  |-- messageUser: "Carlos Vásquez"
-MA90Jrfdm1lv6MHKw38
  |-- messageText: "hola"
  |-- messageTime: 159252724386
  |-- messageUser: "Carlos Vásquez"
-MA9719h3HpbhcB5ytRC
  |-- messageText: "apenas me cargaron los mensajes"
  |-- messageTime: 159252900224
  |-- messageUser: "Carlos Vásquez"
```

Podemos observar los mismos mensajes en Firebase