Forma

Descripción generada automáticamente con confianza media

Firebase

CRUD de un app Android con Firebase de Google

Diego Hernandez Cote

2022

# Configuración inicial

## Firebase. Agregar un nuevo proyecto

Después de iniciar sesión en la página de Firebase, debemos acceder a la consola de nuestra cuenta:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Se accederá a la opción de **Agregar proyecto**, donde se nos pedirá completar tres pasos:

* Nombre del proyecto
* Habilitación de Google Analytics
* Configurar Google Analytics

Posteriormente, se generará el proyecto en Firebase, apareciendo el siguiente panel:

Interfaz de usuario gráfica, Sitio web

Descripción generada automáticamente

## Base de datos: Inicialización

En el panel lateral izquierdo, seleccionamos la opción **Todos los productos** y en la lista accedemos a la opción **Realtime Database**

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

A continuación, se nos preguntará la ubicación del almacenamiento de información, a la fecha, los servidores mas cercanos son los de **Estados Unidos**. Posteriormente, seleccionaremos la opción **Comenzar modo de prueba**. AL finalizar accederemos al siguiente panel:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

## Android Studio. Proyecto CRUDFirebase

Crearemos un nuevo proyecto en Android Studio llamado [CRUDFirebase](https://github.com/QuetzalDeveloper/CRUDFirebase), que por el momento se compondrá de una sola activity en blanco.

Nos dirigiremos al archivo Java y copiamos la primera línea o el **paquete de la aplicación**. Guardamos esta línea para el paso siguiente.

Interfaz de usuario gráfica

Descripción generada automáticamente

También debemos obtener la firma SHA-1 de depuración de la aplicación. Como primer paso, realizaremos una depuración (icono del bicho); posteriormente accedemos a la pestaña  **Gradle** a la derecha del editor, desplegaremos la opción **task**, abrimos el menú  **android** y deremos dos click a la configuración **signingReport**. Se abrirá la consola en la parte inferiory nos dará la información del SHA -1; lo guardamos para el paso siguiente.

Texto

Descripción generada automáticamente

Finalmente, debemos agregar el permiso de internet en el **manifiesto** de la aplicación.

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

## Firebase. Agregar Firebase a app Android

Regresamos a la pagina principal del proyecto o **Project Overview**  o **Descripción general**; una vez en la página, seleccionamos el icono de Android para agregar Firebase a la aplicación Android.

Interfaz de usuario gráfica, Sitio web

Descripción generada automáticamente

Una vez seleccionado, deberemos llenar los formularios que se nos presentan:

1. Registrar app
   1. **Nombre del paquete Android.** Es la línea que copiamos anteriormente
   2. **Sobrenombre de la app**. Se puede omitir. Es un alias de la aplicación
   3. **Certificado de firma SHA-1 de depuración.** Es opcional, sin embargo, es siempre recomendado colocarlo. También se obtuvo anteriormente.
2. Descargar archivo de configuración
   1. Descargamos el archivo de configuración desde el botón azul y seguimos las instrucciones que nos muestran en pantalla.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

1. Agregar el SDK de Firebase
   1. Simplemente seguimos las instrucciones de la sección.
2. Terminar

Una vez terminado el proceso. Nos enviará a la página principal del proyecto.

# Insertar

Como primer paso, creamos un contenedor Java **Persona** como interfaz de los datos:

public class Persona {  
 private String uid, nombre, apellidos, correo, password;  
  
 public Persona() {  
 }  
  
 public String getUid() {  
 return uid;  
 }  
  
 public String getPassword() {  
 return password;  
 }  
  
 public void setPassword(String password) {  
 this.password = password;  
 }  
  
 public void setUid(String uid) {  
 this.uid = uid;  
 }  
  
 public String getNombre() {  
 return nombre;  
 }  
  
 public void setNombre(String nombre) {  
 this.nombre = nombre;  
 }  
  
 public String getApellidos() {  
 return apellidos;  
 }  
  
 public void setApellidos(String apellidos) {  
 this.apellidos = apellidos;  
 }  
  
 public String getCorreo() {  
 return correo;  
 }  
  
 public void setCorreo(String correo) {  
 this.correo = correo;  
 }  
  
 @Override  
 public String toString() {  
 return nombre;  
 }  
}

Creamos la siguiente activity; omitiremos el tomar los datos de los **TextInputEditText**.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Agregaremos la librería para acceder a la base de datos Firebase en el **Gradle/Module:app**

implementation 'com.google.firebase:firebase-database'

En el archivo Java de la ventana de inserción, declaramos las siguientes variables:

FirebaseDatabase firebaseDatabase;  
DatabaseReference databaseReference;

Creamos un método que será invocado desde **onCreate** para inicializar las variables de Firebase:

protected void onCreate(Bundle savedInstanceState) {  
InicializarFirebase();  
}

…

private void InicializarFirebase(){  
 FirebaseApp.*initializeApp*(this);  
 firebaseDatabase = FirebaseDatabase.*getInstance*();  
 databaseReference = firebaseDatabase.getReference();  
}

Se ha creado un método con el nombre **Gurdar** donde se recupera la información del formulario; posteriormente creamos una instancia a la clase **Persona** y enviamos los datos del formulario al objeto.

Persona p = new Persona();  
p.setUid(UUID.*randomUUID*().toString());  
p.setNombre(name);  
p.setApellidos(last);  
p.setCorreo(mail);  
p.setPassword(contra);

Continuando en el mismo método, realizamos la inserción con el siguiente comando:

databaseReference.child("Persona").child(p.getUid()).setValue(p);

* databaseReference**.** Es la variable que hará referencia a la base de datos en Firebase, contiene los permisos y parámetros de acceso a la BD NoSQL.
* child("Persona"). Es la asignación de la entidad donde se almacenará la información; en un modelo relacional, podríamos compararlo con una tabla en la BD.
* child(p.getUid()). Asignamos a una variable como el nodo de los datos que almacenaremos; en un modelo relacional, podríamos compararlo con una clave primaria. En este caso, usaremos la librería UUID que ofrece una serie alfanumérica única cada vez que es invocado. En caso de tener una variable que pueda actuar como llave, puede sustituirse de forma directa.
* setValue(p). Enviamos el resto de los atributos a la base de datos.

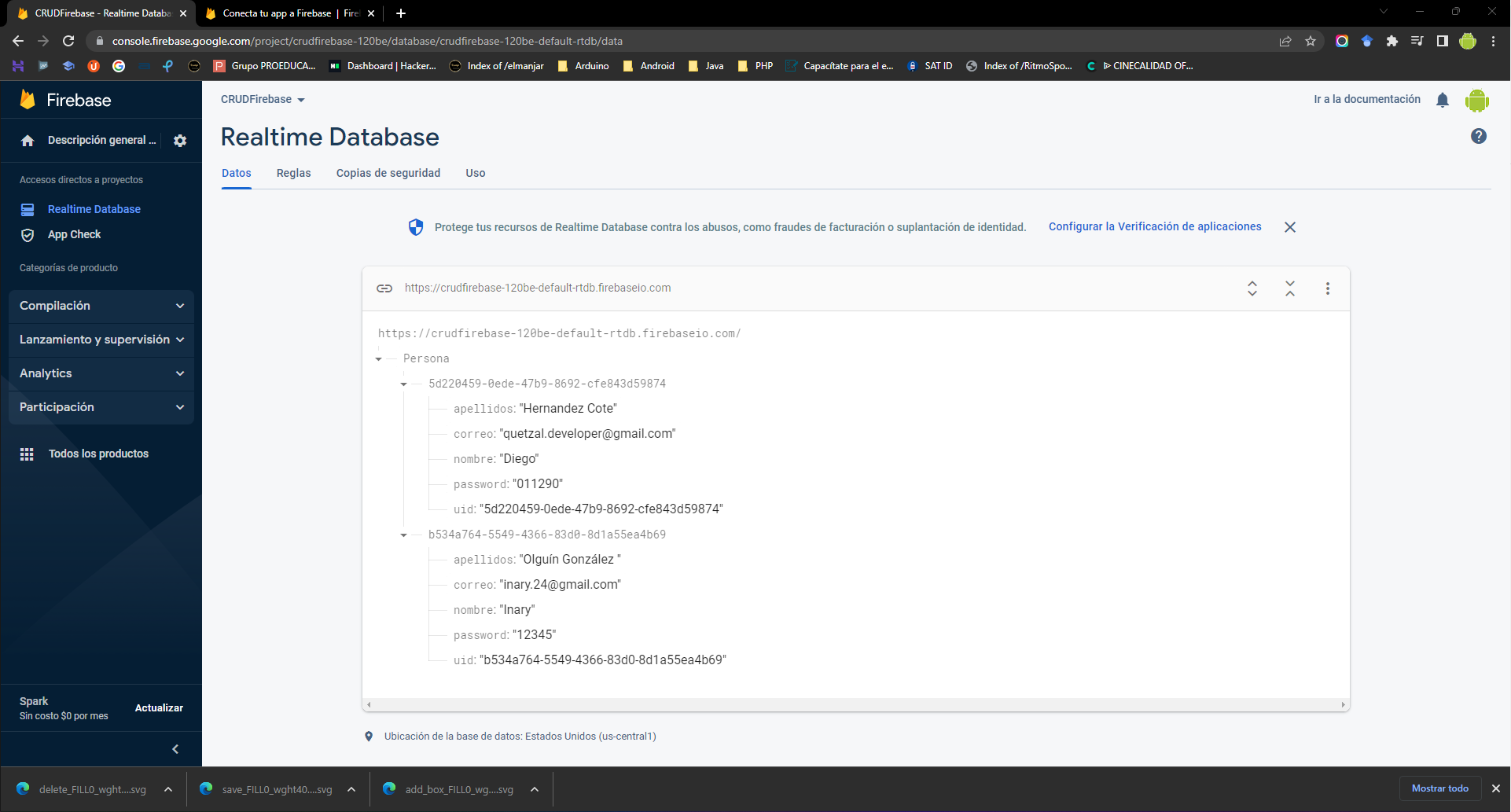
Finalmente colocamos un **Toast** y cerramos la ventana para finalizar la inserción de datos.

Si la programación y conexión ha sido correcta, los datos se guardarán en Firebase como se muestra a continuación:

Una captura de pantalla de una computadora

Descripción generada automáticamente

Recordemos que Firebase es un gestor de bases de datos no relacional bajo el paradigma NoSQL, por lo que no se usan tablas, si no que se almacena en forma de árboles y nodos. Insertamos otro dato desde la aplicación, en este caso desde un dispositivo físico para verificar el funcionamiento:



# Consultar

Para esta sección, se ha creado una **Activity** llamada inicio, la cúal será el **layout** principal de la aplicación y donde se enlistarán los datos almacenados en **Firebase**. El **XML** se compone de un simple **TextView**y un **ListView**. Iniciaremos declarado una lista y un adapter de la clase **Persona** que se encargarán de almacenar y vincular los datos al **ListView** respectivamente; también agregamos las variables de la base de datos de **Firebase**.

private List<Persona> personaList = new ArrayList<Persona>();  
ArrayAdapter<Persona> personaArrayAdapter;  
FirebaseDatabase firebaseDatabase;  
DatabaseReference databaseReference;

Creamos un método que será invocado desde **onCreate** para inicializar las variables de Firebase:

protected void onCreate(Bundle savedInstanceState) {

…  
InicializarFirebase();  
}

…

private void InicializarFirebase(){  
 FirebaseApp.*initializeApp*(this);  
 firebaseDatabase = FirebaseDatabase.*getInstance*();  
 databaseReference = firebaseDatabase.getReference();  
}

Creamos un nuevo método que también será invocado desde **onCreate** para obtener y listar los datos.

protected void onCreate(Bundle savedInstanceState) {

…  
listarDatos();  
}

El método se compone de la siguiente forma:

private void listarDatos(){  
 databaseReference.child("Persona").addValueEventListener(new ValueEventListener() {  
 @Override  
 public void onDataChange(@NonNull DataSnapshot snapshot) {  
 personaList.clear();  
 for(DataSnapshot dataSnapshot : snapshot.getChildren()){  
 Persona p = dataSnapshot.getValue(Persona.class);  
 personaList.add(p);  
 personaArrayAdapter = new ArrayAdapter<Persona>(getApplicationContext(), android.R.layout.*simple\_list\_item\_1*, personaList);  
 lista.setAdapter(personaArrayAdapter);  
 }  
 }  
  
 @Override  
 public void onCancelled(@NonNull DatabaseError error) {  
  
 }  
 });  
}

Si se ha programado correctamente, arrojaría este resultado:

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

## Persistencia de datos

La persistencia de datos se aplica para conservar los cambios de forma local cuando han sido modificados y no se tenga una conexión a la base de datos; una vez que se tiene la conexión, la aplicación detectará los cambios y modificara los registros a como sea necesario. Para hacer una persistencia de datos local, debemos modificar la función **InicializarFirebase()** de la siguiente manera:

private void InicializarFirebase(){  
 FirebaseApp.*initializeApp*(this);  
 firebaseDatabase = FirebaseDatabase.*getInstance*();  
 firebaseDatabase.setPersistenceEnabled(true);  
 databaseReference = firebaseDatabase.getReference();  
}

Sin embargo, el uso de esta persistencia es incorrecta dada la documentación de Android con Firebase; la persistencia debe ser llamada antes de la referencia y debe ejecutarse una sola vez. Como primer paso, eliminamos la línea que colocamos anteriormente, después crearemos una nueva clase **Java** con el nombre **MyFirebaseApp** y agregamos el siguiente código:

public class MyFirebaseApp extends android.app.Application{  
  
 @Override  
 public void onCreate(){  
 super.onCreate();  
 FirebaseDatabase.*getInstance*().setPersistenceEnabled(true);  
 }

}

Posteriormente accederemos al **Manifiest** y agregamos una etiqueta **name** dentro de la etiqueta **application** en el área de las propiedades principales direccionando a la clase generada.

<application  
 android:name=".MyFirebaseApp"  
 android:allowBackup="true"  
 android:dataExtractionRules="@xml/data\_extraction\_rules"

…

El objetivo de hacer la segunda forma de persistencia permite que este presente en toda la aplicación, evitando la generación múltiple de esta propiedad al llamarla en mas de una o todas las clases del proyecto

# Actualizar

Creamos una nueva **Activity** con el mismo formulario de insertar. En la clase principal, agregamos el evento **setOnItemClickListener** a la lista y enviamos los datos del contacto a la nueva clase.

lista.setOnItemClickListener(new AdapterView.OnItemClickListener() {  
 @Override  
 public void onItemClick(AdapterView<?> parent, View view, int position, long id) {  
 Intent i= new Intent(getApplicationContext(), Actualizar.class);  
 i.putExtra("correo", personaList.get(position).getCorreo());  
 i.putExtra("nombre", personaList.get(position).getNombre());  
 i.putExtra("apellidos", personaList.get(position).getApellidos());  
 i.putExtra("password", personaList.get(position).getPassword());  
 startActivity(i);  
 }  
});

En la activity **Actualizar** capturamos los datos y los enviamos a los **TextView** del **XML**.

protected void onCreate(Bundle savedInstanceState) {  
 …

uid = getIntent().getStringExtra("uid");  
 correo.setText(getIntent().getStringExtra("correo"));  
 nombre.setText(getIntent().getStringExtra("nombre"));  
 apellidos.setText(getIntent().getStringExtra("apellidos"));  
 pass.setText(getIntent().getStringExtra("password"));  
*…*  
}

Así mismo, también agregamos las variables de la base de datos de **Firebase**.

FirebaseDatabase firebaseDatabase;  
DatabaseReference databaseReference;

Creamos un método que será invocado desde **onCreate** para inicializar las variables de Firebase:

protected void onCreate(Bundle savedInstanceState) {

…  
InicializarFirebase();  
}

…

private void InicializarFirebase(){  
 FirebaseApp.*initializeApp*(this);  
 firebaseDatabase = FirebaseDatabase.*getInstance*();  
 databaseReference = firebaseDatabase.getReference();  
}

Se ha creado un método con el nombre **Gurdar** donde se recupera la información del formulario; posteriormente creamos una instancia a la clase **Persona** y enviamos los datos del formulario al objeto.

Persona p = new Persona();  
p.setUid(uid);  
p.setNombre(name);  
p.setApellidos(last);  
p.setCorreo(mail);  
p.setPassword(contra);

Continuando en el mismo método, realizamos la inserción con el siguiente comando:

databaseReference.child("Persona").child(p.getUid()).setValue(p);

Finalmente colocamos un **Toast** y cerramos la ventana para finalizar la actualización de datos.

¿Qué diferencia hay entre la línea de inserción y la línea de actualización si son idénticas?; la respuesta radica en que Firebase automáticamente detecta que el identificador **uid** existe en la base de datos, de esta forma, si la llave ya existe en la BD los atributos se actualizan, de lo contrario, se insertan.

# Eliminar

Para este caso, agregamos el evento **setOnItemLongClick** a la lista de la activity **principal**, de tal forma que al mantener presionado un elemento de la lista, la información se borre de la base de datos.

…

lista.setOnItemLongClickListener(new AdapterView.OnItemLongClickListener() {  
 @Override  
 public boolean onItemLongClick(AdapterView<?> parent, View view, int position, long id) {  
 Eliminar(position);  
 return true;  
 }  
});

…

Crearemos el método **Eliminar** donde programaremos un **AlertDialog**. Si el usuario selecciona la opción positiva obtendremos el **UUID** del elemento seleccionado y lo eliminamos de la base de datos con la siguiente instrucción:

databaseReference.child("Persona").child(personaList.get(position).getUid()).removeValue();

El bloque complete quedaría de la siguiente forma:

private void Eliminar(int position){  
 final AlertDialog.Builder d = new AlertDialog.Builder(inicio.this);  
 d.setTitle("Eliminar");  
 d.setMessage("Se eliminara a "+personaList.get(position).toString()+" de la base de datos");  
 d.setCancelable(false);  
 d.setPositiveButton("Eliminar", new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
 databaseReference.child("Persona").child(personaList.get(position).getUid()).removeValue();  
 }  
 });  
 d.setNegativeButton("Cancelar", new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
  
 }  
 });  
 d.show();  
}

# Procesamiento de consultas

La migración entre bases de datos SQL y NoSQL puede ser complicada al momento de hacer consultas. A continuación, ejemplificamos y mostramos alternativas que se pueden aplicar al realizar consultas más específicas.

## Ordenar datos

Android y Firebase poseen 3 funciones para ordenar datos en una consulta:

* orderByChild(). Ordena de acuerdo con el valor de un atributo
* orderByKey(). Ordena el resultado de a cuerdo a la llave primaria
* orderByValue(). Ordena según el valor de cada nodo

## Filtrar datos

Android y Firebase poseen 5 funciones para filtrar datos en una consulta:

* limitToFirst(). Nos permite limitar la cantidad de registros que se desean leer
* limitToLast(). Equivalente al anterior, pero retorna los últimos resultados
* startAt(). Establece el valor donde se desea empezar a consultar los datos
* endAt(). Establece el valor donde se desea terminar de consultar los datos
* equalTo(). Retorna el resultado que es igual al valor que le damos como parámetro

## Ejemplos

SQL: **SELECT** \* **FROM** grupos **WHERE** nombre = "Diego" **LIMIT** 1;

Firebase:

databaseReference.child("grupos").orderByChild("nombre").equalTo("Diego").limitToFirst(1)

SQL: **SELECT** \* **FROM** Persona **WHERE** email = " quetzal.developer@gmail.com ";

Firebase:

databaseReference.child("Persona").orderByChild("email").equalTo("quetzal.developer@gmail.com")

SQL: **SELECT** \* **FROM** Persona **LIMIT** 5;

Firebase:

databaseReference.child("Persona").limitToFirst(5)

SQL: **SELECT** \* **FROM** Persona **WHERE** nombre **LIKE** 'R%';

Firebase:

databaseReference.child("Persona").orderByChild(“nombre”).startAt(“R”)

SQL: **SELECT** \* **FROM** Persona **WHERE** edad < 20;

Firebase:

databaseReference.child("Persona").orderByChild(“edad”).endAt(19)

SQL: **SELECT** \* **FROM** Persona **WHERE** edad>=25 **AND** edad<=50

Firebase:

databaseReference.child("Persona").orderByChild(“edad”).startAt(25).endAt(19)