



INSTITUTO TECNOLÓGICO SUPERIOR DE JEREZ

Ingeniería en Sistemas Computacionales

8vo Semestre

Alumno:

Daniel Alejandro de la Rosa Castañeda

NC:16070126

Materia:

Programación Móvil

Nombre del trabajo:

***AlReporte de trabajo de segunda
oportunidad***

Docente:

Dr. Jorge Manjarrez

Jerez de García Salinas a 4 de junio del 2020



Reporte

En este reporte, se hace énfasis en los tres primeros temas de la materia de programación móvil:

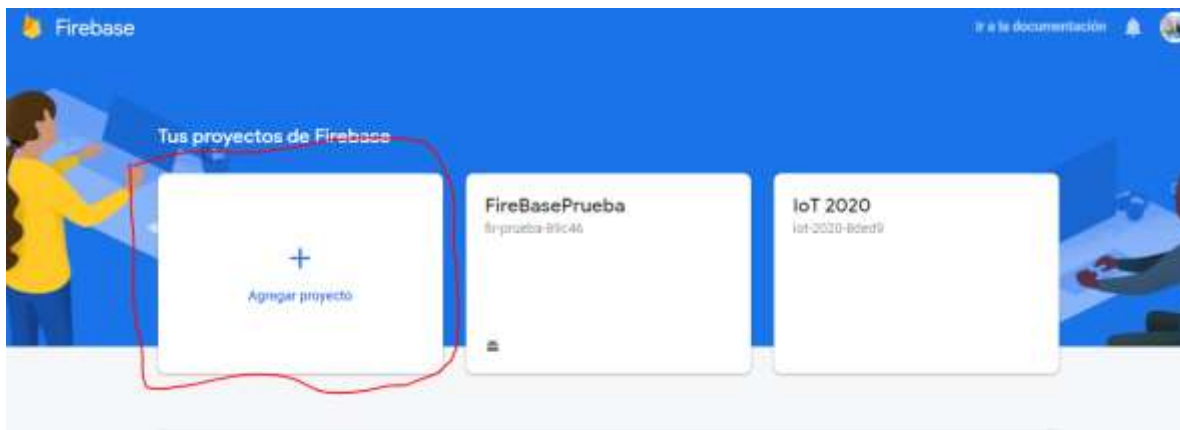
1. Tema 1 Fundamentos: Interfaz de usuario, navegación, eventos
2. Tema 2 Acceso a Datos: Local, Firebase
3. Tema 3 Sensores y conectividad

Retomando una aplicación ya realizada (lector de sensores), ahora, para que el usuario sea el que decida de cuales sensores se les tendrá que realizar lectura de datos.

Procedimiento

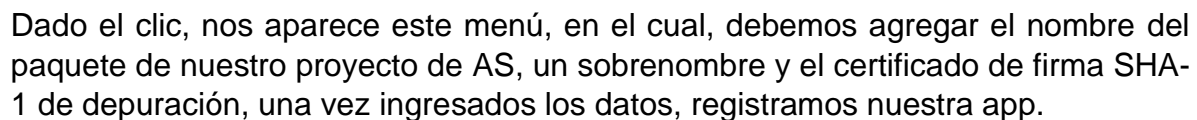
Tomando de base la aplicación antes mencionada, solo se modificaron ciertos aspectos de esta, siendo ellos el poder hacer la elección de los sensores, purificar un poco más el código, añadir una barra de menú, la cual, contiene todos los botones de la interfaz principal, haciéndola más estética a la vista del usuario y por último, arreglando el ciclo de vida de la aplicación, ya que en la versión anterior, no se manejaba adecuadamente.

Lo primero que se tiene que hacer, es configurar firebase. Para ello entramos a su página oficial, y si estamos logeados con una cuenta de google solo tendremos que dar en crear proyecto nuevo.





Una vez finalizado el proceso de creación, tendremos que darle clic en el botón donde está la mascota de Android para enlazar nuestro proyecto.

[illegible]



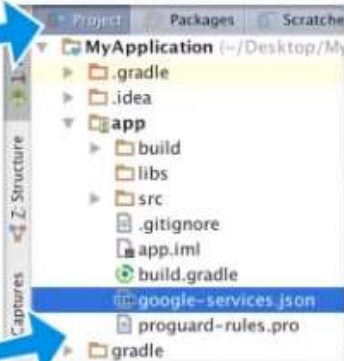
Después, se tendrá que descargar el archivo de configuración JSON para agregarlo a nuestro AS.

2 Descargar archivo de configuración Instrucciones para Android Studio a continuación | [Unity](#) [C++](#)

[Descargar google-services.json](#)

Cambia a la vista **Proyecto** de Android Studio para ver el directorio raíz de tu proyecto.

Coloca el archivo `google-services.json` que acabas de descargar en el directorio raíz del módulo de tu app para Android.



Anterior [Siguiente](#)

Una vez añadido el JSON, dentro del gradle poner el siguiente código (tanto en el gradle de app como en el de proyecto):

Archivo `build.gradle` de nivel de proyecto (`<project>/build.gradle`):

```
buildscript {
    dependencies {
        // Add this line
        classpath 'com.google.gms:google-services:4.0.0'
    }
}
```

Archivo `build.gradle` de nivel de app (`<project>/<app-module>/build.gradle`):

```
dependencies {
    // Add this line
    compile 'com.google.firebase:firebase-core:16.0.0'
}

// Add to the bottom of the file
apply plugin: 'com.google.gms.google-services'
```

Incluye Analytics en la configuración predeterminada ?



Y sincronizamos.

Se necesita inicializar el Firebase, para eso, se hace un método que inicializa las variables necesarias para usar Firebase en Android.

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

Para funcionar correctamente, es necesario que, en el manifest, se añada el permiso de internet:

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

ESTRUCTURA DE LA APLICACIÓN

La aplicación cuenta con tres activities: la activity que controla cuales serán los sensores que se van a leer, la activity donde se muestran todas las lecturas y se realizan las mismas y otra donde se muestran las lecturas en forma de un gráfico.

Activity_principal

Esta activity, consta de seis checkBox y un botón, el cual, al ser presionado, manda la lista de los checkBox seleccionados al otro activity, para realizar la lectura de estos. Si ningún checkBox está seleccionado, se mandará una alerta, que haga mención a que mínimo, debe de haber uno seleccionado.





El código que maneja esta activity, es simple, ya que solo consta de las validaciones de los checkBox para mandar una lista, con los nombres de los sensores seleccionados (el código está comentado sobre lo que hacen ciertas partes de código):

```
public class PrincipalActivity extends AppCompatActivity {
    CheckBox ac,grav,lum,mag,pas,prox;
    ArrayList<String> lista;
    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_principal);
        // se enlazan Los componentes (check box) con el codigo JAVA
        ac=findViewById(R.id.CB_Acelerometro);
        grav=findViewById(R.id.CB_Gravedad);
        lum=findViewById(R.id.CB_Luminosidad);
        mag=findViewById(R.id.CB_Magnetico);
        pas=findViewById(R.id.CB_Pasos);
        prox=findViewById(R.id.CB_Proximidad);
        lista=new ArrayList<>();
    }
    //Metodo onClick, que se acciona al presionar el boton
    public void onClick(View view){
        //compara si no hay ningun checkBox seleccionado, si esto es verdad
        //manda una alerta haciendo mencion a eso
        if(!ac.isChecked() && !grav.isChecked() && !lum.isChecked() &&
!mag.isChecked() && !pas.isChecked() && !prox.isChecked()){
            Toast.makeText(getApplicationContext(),"Error, Minimo debe de
haber algun sensor seleccionado",Toast.LENGTH_LONG).show();
            //sino, a traves de varios If, se comprueba si esta seleccionado
        }else{
            //depende de cuales sean Los check box seleccionados, seran
            Los datos mandados
            if(ac.isChecked()){
                lista.add("Acelerometro");
            }if(grav.isChecked()){
                lista.add("Gravedad");
            }if(lum.isChecked()){
                lista.add("Luminosidad");
            }if(mag.isChecked()){
                lista.add("Magnetico");
            }if(pas.isChecked()){
                lista.add("Pasos");
            }if(prox.isChecked()){
                lista.add("Proximidad");
            }
            //se crea el intent, haciendo referencia a La activity que se
            quiere abrir
            Intent intent=new Intent(this,MainActivity.class);
            //se le pasan La lista de datos
            intent.putExtra("Lista", lista);
        }
    }
}
```




```
        //se inicia la otra actividad  
        startActivity(intent);  
    }  
}
```

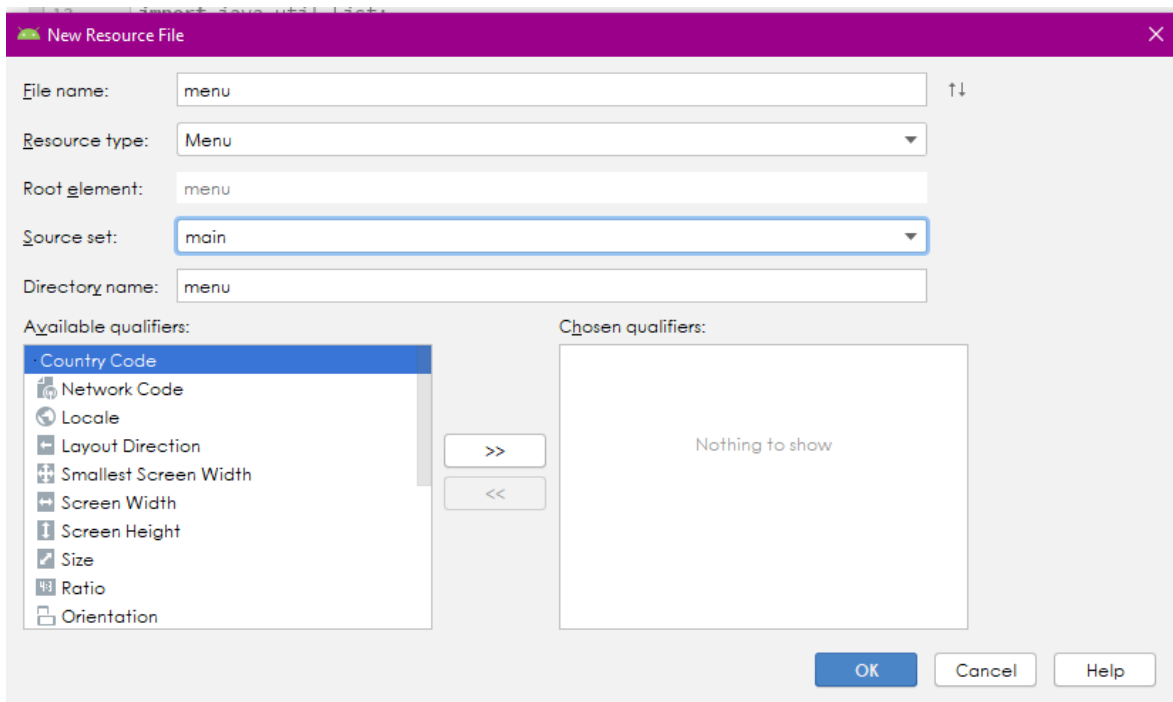
Main_activity

La siguiente activity, consta de muchas partes, tanto de código, como de interfaz. En la interfaz de usuario, se tiene una listView que contiene todas las lecturas, una caja de texto para filtrar esas lecturas y una barra de menú, que contiene los botones de:

- Realizar lectura
- Detener lectura
- Actualizar lista
- Limpiar lista
- Generar graficas



Este componente como tal, no se encuentra dentro de AndroidStudio, por lo cual se tiene que diseñar. Para esto, dentro de la carpeta res, creamos un Android Resource file, el cual debe de estar configurado de la siguiente manera: en primer lugar, un nombre, el que sea, en este caso, menú; el tipo de recurso que será Menú, el source set se queda igual y el nombre de directorio se cambia a menú (no existe ese directorio, por lo cual, se creará automáticamente):



Una vez creado, es solamente el añadir ítems, los cuales tendrán configuraciones como id, título, showAction y en nuestro caso, un icono, el cual se agregó anteriormente, dentro de la carpeta de drawable, nuevo ImageAsset y se agrega, ya se del mismo AS o de nuestro sistema. En este caso, son 5 ítems del menú y este es el código XML:

```
<?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/icon_add"
          android:icon="@drawable/ic_add"
          android:title="Edit done"
          app:showAsAction="always"
        ></item>

    <item android:id="@+id/icon_stop"
          android:icon="@drawable/ic_stop"
          android:title="Edit done"
          app:showAsAction="always"
        ></item>

    <item android:id="@+id/icon_clean"
          android:icon="@drawable/ic_clean2"
          android:title="Edit done"
          app:showAsAction="always"
        ></item>

    <item android:id="@+id/icon_update"
          android:icon="@drawable/ic_update"
          android:title="Edit done"
        ></item>
```




```
        app:showAsAction="always"
    ></item>
    <item android:id="@+id/icon_chart"
        android:icon="@drawable/ic_chart"
        android:title="Edit done"
        app:showAsAction="always"
    ></item>
```

</menu>

Así es el resultado:



Una vez definido el diseño, se pasa a la codificación.

En el método por defecto (onCreate), hay demasiados eventos y/o código, que van desde recibir la lista de datos de la otra interfaz, hasta un keyListener para el filtro de la caja de texto (el código está comentado sobre lo que hacen ciertas partes de código):

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    //se crea un objeto de tipo Bundle, que recibe los datos de
    //la otra actividad
    Bundle parametros=this.getIntent().getExtras();
    //se guardan en una lista
    listaDatosEntreActivities= parametros.getStringArrayList("Lista");
    //se enlaza la listview con el código
    listView=findViewById(R.id.ListViewObjetos);
    //mandamos llamar el método de inicializarFirebase, el cual, inicia
    //el servicio de Firebase
    inicializarFirebase();
    //Se traen los datos de Firebase dependiendo la "tabla"
    //con el método listaDatos
    listaDatos("Gravedad");
    listaDatos("Acelerometro");
    listaDatos("Luminosidad");
    listaDatos("Magnetico");
    listaDatos("Pasos");
    listaDatos("Proximidad");
```



```
//esta bandera se utilizara en otro metodo
bandera=1;
//se enlaza la caja de texto con codigo
cajaSearch=findViewById(R.id.txt_busqueda);
//se le añade un KeyListener para filtrar en la lista
cajaSearch.addTextChangedListener(new TextWatcher() {
    @Override
    public void beforeTextChanged(CharSequence s, int start, int count,
int after) {

    }
    //se usa este metodo, ya que es el necesario
    //por que realiza el filtro cada vez que el texto cambia
    @Override
    public void onTextChanged(CharSequence s, int start, int before, int
count) {
        arrayAdapter.getFilter().filter(s);
    }

    @Override
    public void afterTextChanged(Editable s) {

    }
});
}
```

Para añadir la funcionalidad del menuBar y los botones se usan los siguientes métodos (el código está comentado sobre lo que hacen ciertas partes de código):

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    //infla el menu para ponerlo en la interfaz
    getMenuInflater().inflate(R.menu.menu_main,menu);
    return super.onCreateOptionsMenu(menu);
}

public boolean onOptionsItemSelected(@NonNull MenuItem item) {
    //es un switch, el cual, selecciona el boton
    // que se ha presionado y realiza ciertas acciones
    switch (item.getItemId()){
        case R.id.icon_add:
            Toast.makeText(getApplicationContext(),"Comenzo la lectura de
datos",Toast.LENGTH_LONG).show();
            agregar();
            break;
        case R.id.icon_stop:
            detener();
            Toast.makeText(getApplicationContext(),"Se detuvo la lectura de
```



```
datos", Toast.LENGTH_LONG).show();
    break;
    case R.id.icon_clean:
        Toast.makeText(getApplicationContext(), "Lista
limpia", Toast.LENGTH_LONG).show();
        limpiar();
        break;
    case R.id.icon_update:
        actualizar();
        Toast.makeText(getApplicationContext(), "Se actualizo la lista de
datos", Toast.LENGTH_LONG).show();
        break;
    case R.id.icon_chart:
        Toast.makeText(getApplicationContext(), "Se abrirá la
gráfica", Toast.LENGTH_LONG).show();
        abrirGraf();
        break;
    default:
        break;
}
return true;
}
```

agregar ()

el método agregar, como su nombre lo indica, añade a la base de datos Firebase, las lecturas, basándose en el filtro de los sensores traídos desde la otra activity.

Este método, registra el Listener del sensor, y a través del metodo onSensorChanged, es donde se realizan las lecturas de datos de sensores. (el código está comentado sobre lo que hacen ciertas partes de código):

```
public void agregar(){
    //se crea un sensor manager
    sensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);
    List<Sensor> listaSensores;
    //se recorre la lista a traves de un for
    for(int i=0; i< listaDatosEntreActities.size();i++){
        //si en la lista esta este sensor, se le registra el Listener
        if(listaDatosEntreActities.get(i).equalsIgnoreCase("Acelerometro")){
            listaSensores =
            sensorManager.getSensorList(Sensor.TYPE_ACCELEROMETER);
            if (!listaSensores.isEmpty()) {
                Sensor acelerometerSensor = listaSensores.get(0);
                sensorManager.registerListener((SensorEventListener) this,
                acelerometerSensor, SensorManager.SENSOR_DELAY_NORMAL);}
            //si en la lista esta este sensor, se le registra el Listener
        }if(listaDatosEntreActities.get(i).equalsIgnoreCase("Gravedad")){
```



```
        listaSensores =
        sensorManager.getSensorList(Sensor.TYPE_GRAVITY);
        if (!listaSensores.isEmpty()) {
            Sensor gravedad = listaSensores.get(0);
            sensorManager.registerListener((SensorEventListener) this,
            gravedad, SensorManager.SENSOR_DELAY_NORMAL);}
        //si en la lista esta este sensor, se le registra el Listener
    }if(listaDatosEntreActities.get(i).equalsIgnoreCase("Luminosidad")){
        listaSensores = sensorManager.getSensorList(Sensor.TYPE_LIGHT);
        if (!listaSensores.isEmpty()) {
            Sensor lightSensor = listaSensores.get(0);
            sensorManager.registerListener((SensorEventListener) this,
            lightSensor, SensorManager.SENSOR_DELAY_NORMAL);}
        //si en la lista esta este sensor, se le registra el Listener
    }if(listaDatosEntreActities.get(i).equalsIgnoreCase("Magnetico")){
        listaSensores =
        sensorManager.getSensorList(Sensor.TYPE_MAGNETIC_FIELD);
        if (!listaSensores.isEmpty()) {
            Sensor magneticSensor = listaSensores.get(0);
            sensorManager.registerListener((SensorEventListener) this,
            magneticSensor, SensorManager.SENSOR_DELAY_NORMAL);}
        }if(listaDatosEntreActities.get(i).equalsIgnoreCase("Pasos")){
            listaSensores =
            sensorManager.getSensorList(Sensor.TYPE_STEP_COUNTER);
            if (!listaSensores.isEmpty()) {
                Sensor pasos = listaSensores.get(0);
                sensorManager.registerListener((SensorEventListener) this,
                pasos, SensorManager.SENSOR_DELAY_NORMAL);}
            //si en la lista esta este sensor, se le registra el Listener
        }if(listaDatosEntreActities.get(i).equalsIgnoreCase("Proximidad")){
            listaSensores =
            sensorManager.getSensorList(Sensor.TYPE_PROXIMITY);
            if (!listaSensores.isEmpty()) {
                Sensor proximitySensor = listaSensores.get(0);
                sensorManager.registerListener((SensorEventListener) this,
                proximitySensor, SensorManager.SENSOR_DELAY_NORMAL);}
        }
    }
}
```

```
@Override
public void onSensorChanged(SensorEvent event) {
    //la bandera (que maneja el ciclo de vida de la app)
    //está en 1
    if(bandera==1) {
        //se sincroniza
        synchronized (this) {
            //en un switch que verifica si el sensor tiene evento o no
            String x, y, z;
            date = new Date();
        }
    }
}
```



```
hourFormat = new SimpleDateFormat("HH:mm:ss");
dateFormat = new SimpleDateFormat("dd/MM/yyyy");
formateador = new DecimalFormat("#.00");
switch (event.sensor.getType()) {
    case Sensor.TYPE_ACCELEROMETER:
        //se toman los datos del sensor
        x = formateador.format(event.values[0]);
        y = formateador.format(event.values[1]);
        z = formateador.format(event.values[2]);
        //se guardan en una clase llamada SensorTres,
        //la cual nada mas almacena temporalmente los datos
        sensorTres = new
SensorTres(UUID.randomUUID().toString(), x, y, z, hourFormat.format(date),
dateFormat.format(date));
        //se añade el dato en la base de datos en la "tabla
correspondiente"

        databaseReference.child("Acelerometro").child(sensorTres.getId()).setValue(s
ensorTres);

        break;
    case Sensor.TYPE_GRAVITY:
        //de la misma manera que el case anterior, así funcionan
cada uno

        x = formateador.format(event.values[0]);
        y = formateador.format(event.values[1]);
        z = formateador.format(event.values[2]);
        sensorTres = new
SensorTres(UUID.randomUUID().toString(), x, y, z, hourFormat.format(date),
dateFormat.format(date));

        databaseReference.child("Gravedad").child(sensorTres.getId()).setValue(sens
orTres);

        break;
    case Sensor.TYPE_MAGNETIC_FIELD:
        x = formateador.format(event.values[0]);
        y = formateador.format(event.values[1]);
        z = formateador.format(event.values[2]);
        sensorTres = new
SensorTres(UUID.randomUUID().toString(), x, y, z, hourFormat.format(date),
dateFormat.format(date));

        databaseReference.child("Magnetico").child(sensorTres.getId()).setValue(sens
orTres);

        break;
    case Sensor.TYPE_PROXIMITY:
        x = formateador.format(event.values[0]);
        sensorUno = new SensorUno(UUID.randomUUID().toString(),
x, hourFormat.format(date), dateFormat.format(date));

        databaseReference.child("Proximidad").child(sensorUno.getId()).setValue(sens
orUno);

        break;
```



```
        case Sensor.TYPE_LIGHT:
            x = formateador.format(event.values[0]);
            sensorUno = new SensorUno(UUID.randomUUID().toString(),
            x, hourFormat.format(date), dateFormat.format(date));

            databaseReference.child("Luminosidad").child(sensorUno.getId()).setValue(sensorUno);

            break;
        case Sensor.TYPE_STEP_COUNTER:
            x = formateador.format(event.values[0]);
            sensorUno = new SensorUno(UUID.randomUUID().toString(),
            x, hourFormat.format(date), dateFormat.format(date));

            databaseReference.child("Pasos").child(sensorUno.getId()).setValue(sensorUno);

            break;
    }
}
```

detener ()

el metodo detener, manda a llamar una función del sensor manager llamada unregisterListener, el cual, hace que deje de tener el Listener e interrumpe la lectura

```
public void detener(){
    sensorManager.unregisterListener(this);
}
```

limpiar ()

el método limpiar, como su nombre lo indica, limpia la listView:

```
public void limpiar(){
    listaSensores.clear();
    arrayAdapter=new
    ArrayAdapter<String>(MainActivity.this,android.R.layout.simple_list_item_1,
    listaSensores);
    listView.setAdapter(arrayAdapter);
}
```

actualizar ()

este método, actualiza los datos en la ListView

```
public void actualizar(){
    listaSensores.clear();
```




```
arrayAdapter=new  
ArrayAdapter<String>(MainActivity.this,android.R.layout.simple_list_item_1,  
listaSensores);  
listView.setAdapter(arrayAdapter);  
listaDatos("Gravedad");  
listaDatos("Acelerometro");  
listaDatos("Luminosidad");  
listaDatos("Magnetico");  
listaDatos("Pasos");  
listaDatos("Proximidad");  
}
```

abrirGraf ()

el método en cuestión, crea un intent, el cual hace referencia a la activity de Grafica y le manda las cantidades de lecturas de cada sensor:

```
public void abrirGraf(){  
    Intent i=new Intent(this,Grafica.class);  
    i.putExtra("datos",datosGrafica);  
    startActivity(i);  
}
```

Ahora, hay varios métodos más, que ayudan a la funcionalidad de este activity, entre ellos, están los métodos del ciclo de vida: onResume, onPause, onStop, onStart, onDestroy. Estos metodos, mueven la bandera entre 0 y 1, esto con el fin de que, si la app se cierra o se destruye, deje de registrar datos, ya que si lo sigue haciendo, consume recursos del celular y de wifi, y si vuelve, se puedan registrar datos dándole de nuevo al botón de agregar.

```
@Override  
protected void onResume() {  
    super.onResume();  
    bandera=1;  
}
```

```
@Override  
protected void onPause() {  
    super.onPause();  
    bandera=0;  
}
```

```
@Override  
protected void onStop() {  
    super.onStop();  
    bandera=0;  
}
```

```
@Override
```



```
protected void onStart() {
    super.onStart();
    bandera=1;
}

@Override
protected void onDestroy() {
    super.onDestroy();
    bandera=0;
}
```

listaDatos ()

en este método, se hace la petición a la base de datos para retornar los datos de la misma y puedan ser mostrados en la listView, además de tener un contador en cada uno de los if, para saber cuantas lecturas hay de cada sensor y pasar esos datos a la activity de graficas

```
private void listaDatos(final String sensor) {
    //se añade un listener para iterar sobre una de las "tablas"
    databaseReference.child(sensor).addValueEventListener(new
    ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
            //se itera en esa tabla
            for (DataSnapshot obj: dataSnapshot.getChildren()){
                //se compara si la "tabla" es la de Gravedad
                if(sensor.equalsIgnoreCase("Gravedad")){
                    //se obtiene el dato y se guarda en la clase SensorTres
                    SensorTres s=obj.getValue(SensorTres.class);
                    //se añade un item a la lista del adaptador
                    listaSensores.add(s.toStringGravedad());
                    // se crea el adaptador del listview
                    arrayAdapter=new
                    ArrayAdapter<String>(MainActivity.this,android.R.layout.simple_list_item_1,
                    listaSensores);

                    // se añade el adaptador a la lista
                    listView.setAdapter(arrayAdapter);
                    // se añade uno mas al contador, para saber cuantos

                    datos

                    //hay de este sensor y poderse graficar
                    datosGrafica[1]++;
                    //se hace lo mismo con todos los sensores
                }if(sensor.equalsIgnoreCase("Acelerometro")){
                    SensorTres s=obj.getValue(SensorTres.class);
                    listaSensores.add(s.toStringAcele());
                    arrayAdapter=new
                    ArrayAdapter<String>(MainActivity.this,android.R.layout.simple_list_item_1,
```



```
listaSensores);

        listView.setAdapter(arrayAdapter);
        datosGrafica[0]++;
    }if(sensor.equalsIgnoreCase("Luminosidad")){
        SensorUno s=obj.getValue(SensorUno.class);
        listaSensores.add(s.toStringLuminosidad());
        arrayAdapter=new
ArrayAdapter<String>(MainActivity.this,android.R.layout.simple_list_item_1,
listaSensores);

        listView.setAdapter(arrayAdapter);
        datosGrafica[2]++;
    }if(sensor.equalsIgnoreCase("Magnetico")){
        SensorTres s=obj.getValue(SensorTres.class);
        listaSensores.add(s.toStringMagnetico());
        arrayAdapter=new
ArrayAdapter<String>(MainActivity.this,android.R.layout.simple_list_item_1,
listaSensores);

        listView.setAdapter(arrayAdapter);
        datosGrafica[3]++;
    }if(sensor.equalsIgnoreCase("Pasos")){
        SensorUno s=obj.getValue(SensorUno.class);
        listaSensores.add(s.toStringPasos());
        arrayAdapter=new
ArrayAdapter<String>(MainActivity.this,android.R.layout.simple_list_item_1,
listaSensores);

        listView.setAdapter(arrayAdapter);
        datosGrafica[4]++;
    }if(sensor.equalsIgnoreCase("Proximidad")){
        SensorUno s=obj.getValue(SensorUno.class);
        listaSensores.add(s.toStringProximidad());
        arrayAdapter=new
ArrayAdapter<String>(MainActivity.this,android.R.layout.simple_list_item_1,
listaSensores);

        listView.setAdapter(arrayAdapter);
        datosGrafica[5]++;
    }
    }
}

@Override
public void onCancelled(@NonNull DatabaseError databaseError) {

}

});
}
```

a continuación, se muestran las clases que ayudan a acomodar los datos y a guardarlos momentáneamente: SensorTres y SensorUno



SensorTres

```
public class SensorTres {
    private String id;
    private String x;
    private String y;
    private String z;
    private String hora;
    private String fecha;

    public SensorTres() {
    }

    public SensorTres(String id, String x, String y, String z, String hora,
String fecha) {
        this.id = id;
        this.x = x;
        this.y = y;
        this.z = z;
        this.hora = hora;
        this.fecha = fecha;
    }

    public String getId() {
        return id;
    }

    public void setId(String id) {
        this.id = id;
    }

    public String getX() {
        return x;
    }

    public void setX(String x) {
        this.x = x;
    }

    public String getY() {
        return y;
    }

    public void setY(String y) {
        this.y = y;
    }

    public String getZ() {
        return z;
    }

    public void setZ(String z) {
        this.z = z;
    }
}
```



```
}

public String getHora() {
    return hora;
}

public void setHora(String hora) {
    this.hora = hora;
}

public String getFecha() {
    return fecha;
}

public void setFecha(String fecha) {
    this.fecha = fecha;
}

@Override
public String toString() {
    return "";
}

public String toStringAcele(){
    return "Tipo de sensor: Acelerómetro \n" +
        "Valor en X: "+getX()+" m/s^2\n"+
        "Valor en Y: "+getY()+" m/s^2\n"+
        "Valor en Z: "+getZ()+" m/s^2\n"+
        "Hora de la captura: "+getHora()+"\n" +
        "Fecha de la lectura: "+getFecha();
}

public String toStringGiro(){
    return "Tipo de sensor: Giróscopio \n" +
        "Valor en X: "+getX()+" rad/s\n"+
        "Valor en Y: "+getY()+" rad/s\n"+
        "Valor en Z: "+getZ()+" rad/s\n"+
        "Hora de la captura: "+getHora()+"\n" +
        "Fecha de la lectura: "+getFecha();
}

public String toStringGravedad(){
    return "Tipo de sensor: Gravedad \n" +
        "Valor en X: "+getX()+" rad/s\n"+
        "Valor en Y: "+getY()+" rad/s\n"+
        "Valor en Z: "+getZ()+" rad/s\n"+
        "Hora de la captura: "+getHora()+"\n" +
        "Fecha de la lectura: "+getFecha();
}

public String toStringMagnetico(){
    return "Tipo de sensor: Magnético \n" +
        "Valor en X: "+getX()+" rad/s\n"+
```



```
        "Valor en Y: "+getY()+" rad/s\n"+  
        "Valor en Z: "+getZ()+" rad/s\n"+  
        "Hora de la captura: "+getHora()+"\n" +  
        "Fecha de la lectura: "+getFecha();  
    }  
}
```

SensorUno

```
public class SensorUno {  
    private String id;  
    private String x;  
  
    private String hora;  
    private String fecha;  
  
    public SensorUno() {  
    }  
  
    public SensorUno(String id, String x, String hora, String fecha) {  
        this.id = id;  
        this.x = x;  
  
        this.hora = hora;  
        this.fecha = fecha;  
    }  
  
    public String getId() {  
        return id;  
    }  
  
    public void setId(String id) {  
        this.id = id;  
    }  
  
    public String getX() {  
        return x;  
    }  
  
    public void setX(String x) {  
        this.x = x;  
    }  
  
    public String getHora() {  
        return hora;  
    }  
  
    public void setHora(String hora) {  
        this.hora = hora;  
    }  
}
```




```
public String getFecha() {
    return fecha;
}

public void setFecha(String fecha) {
    this.fecha = fecha;
}

@Override
public String toString() {
    return "";
}

public String toStringHumedad(){
    return "Tipo de sensor: Humedad \n" +
        "Valor de la Humedad: "+getX()+" %\n"+
        "Hora de la captura: "+getHora()+"\n" +
        "Fecha de la lectura: "+getFecha();
}

public String toStringLuminosidad(){
    return "Tipo de sensor: Luminosidad \n" +
        "Valor de la Luminosidad: "+getX()+" lx\n"+
        "Hora de la captura: "+getHora()+"\n" +
        "Fecha de la lectura: "+getFecha();
}

public String toStringPasos(){
    return "Tipo de sensor: Pasos \n" +
        "Pasos desde la ultima lectura: "+getX()+"\n"+
        "Hora de la captura: "+getHora()+"\n" +
        "Fecha de la lectura: "+getFecha();
}

public String toStringProximidad(){
    return "Tipo de sensor: Proximidad \n" +
        "Distancia: "+getX()+" mts\n"+
        "Hora de la captura: "+getHora()+"\n" +
        "Fecha de la lectura: "+getFecha();
}
}
```

a pesar de ya estar funcional esta parte, al momento de regresar con el botón back del dispositivo, y volvemos al otro activity, si seleccionamos otros sensores, no hace el cambio y mantiene los sensores anteriores, por ende, es necesario añadir el siguiente metodo:



```
//metodo para controlar el evento de back del celular
@Override
public void onBackPressed() {
    // si no se ha presionado entra aqui
    if(conter==0){
        //avisa si es que se quiere salir y aumenta el contador
        Toast.makeText(getApplicationContext(),"Presione nuevamente para
salir",Toast.LENGTH_LONG).show();
        conter++;
        //sino
    }else{
        // se crea un intent para abrir la activity anterior
        Intent i=new Intent(this,PrincipalActivity.class);
        startActivity(i);
    }
    //es necesario añadir un contador, el cual contara 3000 milisegundos en
    intervalos de 1000
    //esto con el fin de que si no se volvio a dar back antes de 3 segundos
    // se vuelva a reiniciar el contador
    new CountDownTimer(3000,1000) {
        @Override
        public void onTick(long millisUntilFinished) { }
        @Override
        public void onFinish() {
            // al finalizar el conteo, el contador regresa a 0
            conter=0;
        }
    }.start();
}
```

La última funcionalidad extra es una gráfica de barras que muestra todas las lecturas realizadas hasta el momento.

Primeramente, se tiene que importar una librería de un repositorio de github, ya que, por defecto, Android no cuenta con clases para graficas. Para poder añadirlo, se agregan ciertos valores en el Gradle para que estas puedan ser reconocidas por Android Studio.

Una vez hecho esto, solo es cuestión de empezar a codificar siguiendo algunos tips y tutoriales de ese mismo repositorio. La clase para graficar, quedaría de la siguiente manera:

```
package com.example.firebaseio;

import android.content.Intent;
import android.graphics.Color;
import android.os.Bundle;
import android.os.CountDownTimer;
import android.util.Log;
import android.widget.Toast;
```



```
import com.github.mikephil.charting.charts.BarChart;
import com.github.mikephil.charting.charts.Chart;
import com.github.mikephil.charting.components.Legend;
import com.github.mikephil.charting.components.LegendEntry;
import com.github.mikephil.charting.components.XAxis;
import com.github.mikephil.charting.components.YAxis;
import com.github.mikephil.charting.data.BarData;
import com.github.mikephil.charting.data.BarDataSet;
import com.github.mikephil.charting.data.BarEntry;
import com.github.mikephil.charting.data.DataSet;
import com.github.mikephil.charting.formatter.IndexAxisValueFormatter;

import java.lang.reflect.Array;
import java.util.ArrayList;
import java.util.Arrays;

import androidx.appcompat.app.AppCompatActivity;

public class Grafica extends AppCompatActivity {
    private int conter=0;
    private int datos[];
    private BarChart barChart;
    private String
    []meses={"Acelerometro","Gravedad","Luminosidad","Magnetismo","Pasos","Proxi
    midad"};

    private
    int[] colores={Color.RED,Color.BLUE,Color.GREEN,Color.BLACK,Color.YELLOW,Colo
    r.DKGRAY};
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        datos=getIntent().getExtras().getIntArray("datos");
        super.onCreate(savedInstanceState);
        setContentView(R.layout.graficas);

        barChart=(BarChart)findViewById(R.id.Barchart);
        createCharts();
    }

    private Chart getSameChart(Chart chart, String description, int
    textColor, int backgroundColor, int animateY){
        chart.getDescription().setText(description);
        chart.getDescription().setTextSize(15);
        chart.setBackgroundColor(backgroundColor);
        chart.animateY(animateY);

        return chart;
    }

    private void legend(Chart chart){
        Legend legend=chart.getLegend();
        legend.setForm(Legend.LegendForm.CIRCLE);
    }
}
```



```
legend.setHorizontalAlignment(legend.HorizontalAlignment.CENTER);
    ArrayList<LegendEntry> entries=new ArrayList<>();
    for (int i=0;i<meses.length;i++){
        LegendEntry entry=new LegendEntry();
        entry.formColor=colores[i];
        entry.label=meses[i];
        entries.add(entry);
    }
    legend.setCustom(entries);
}

private ArrayList<BarEntry>getBarEntries(){
    ArrayList<BarEntry>entries=new ArrayList<>();
    for (int i=0;i<datos.length;i++){
        entries.add(new BarEntry(i,datos[i]));
    }
    return entries;
}

private void axisX(XAxis axis){
    axis.setGranularityEnabled(true);
    axis.setPosition(XAxis.XAxisPosition.BOTTOM);
    axis.setValueFormatter(new IndexAxisValueFormatter(meses));
}

private void axisLeft(YAxis axis){
    axis.setSpaceTop(30);
    axis.setAxisMinimum(0);
}

private void axisRight(YAxis axis){
    axis.setEnabled(false);
}

public void createCharts(){

    barChart=(BarChart)getSameChart(barChart,"",Color.RED,Color.GRAY,3000);
    Log.i("m", Arrays.toString(datos));
    barChart.setDrawGridBackground(true);
    barChart.setDrawBarShadow(true);
    barChart.setData(getBarData());
    barChart.invalidate();
    axisX(barChart.getXAxis());
    axisLeft(barChart.getAxisLeft());
    axisRight(barChart.getAxisRight());
}

private DataSet getData(DataSet dataSet){
    dataSet.setColors(colores);
    dataSet.setValueTextColor(Color.WHITE);
    dataSet.setValueTextSize(10);
    return dataSet;
}
```



```
private BarData getBarData(){
    BarDataSet barDataSet=(BarDataSet)getData(new
BarDataSet(getBarEntries(),""));
    barDataSet.setBarShadowColor(Color.GRAY);
    BarData barData=new BarData(barDataSet);
    barData.setBarWidth(0.45f);
    return barData;
}

}
```

Resultados

Se muestran a continuación, varias capturas de pantalla, con la aplicación corriendo en el dispositivo:



Figura 1. Interfaz de selección de sensores



Figura 2. Selección de algunos sensores



Figura 3. Interfaz principal



Figura 4. Comienza lectura de datos

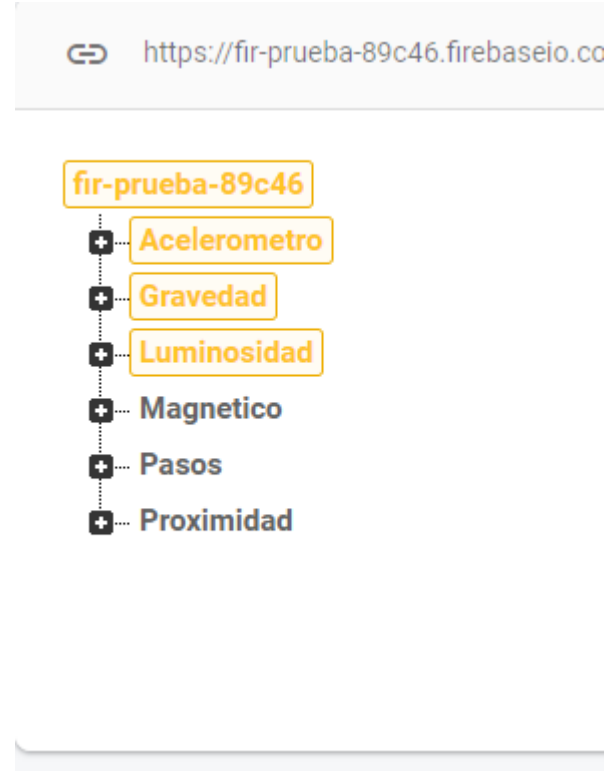


Figura 5. Cuando un dato se inserta, la tabla se pone de color amarillo



Figura 6. se detiene la lectura de datos



Figura 7. Se limpia la lista



Figura 8. Se actualiza la lista

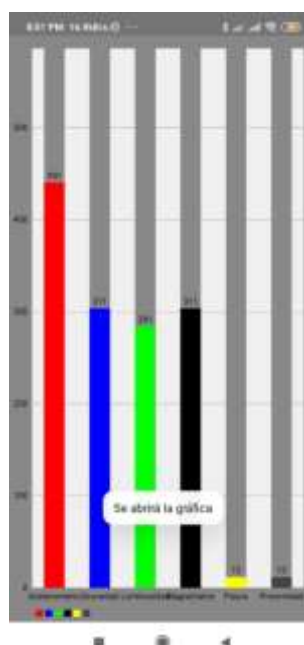


Figura 9. Interfaz de la gráfica de barras



Lector de Sensores con Firebase

04/06/2020

Lista de lecturas

Tipo de sensor: Luminosidad
Valor de la Luminosidad: 83.00 lx
Hora de la captura: 16:58:01
Fecha de la lectura: 04/06/2020
Tipo de sensor: Luminosidad
Valor de la Luminosidad: 66.00 lx
Hora de la captura: 16:42:14
Fecha de la lectura: 04/06/2020
Tipo de sensor: Luminosidad
Valor de la Luminosidad: 120.00 lx
Hora de la captura: 15:40:08
Fecha de la lectura: 04/06/2020
Tipo de sensor: Luminosidad
Valor de la Luminosidad: 66.00 lx
Hora de la captura: 16:42:14
Fecha de la lectura: 04/06/2020
Tipo de sensor: Luminosidad
Valor de la Luminosidad: 25.00 lx

Figura 10. Filtro funcionando