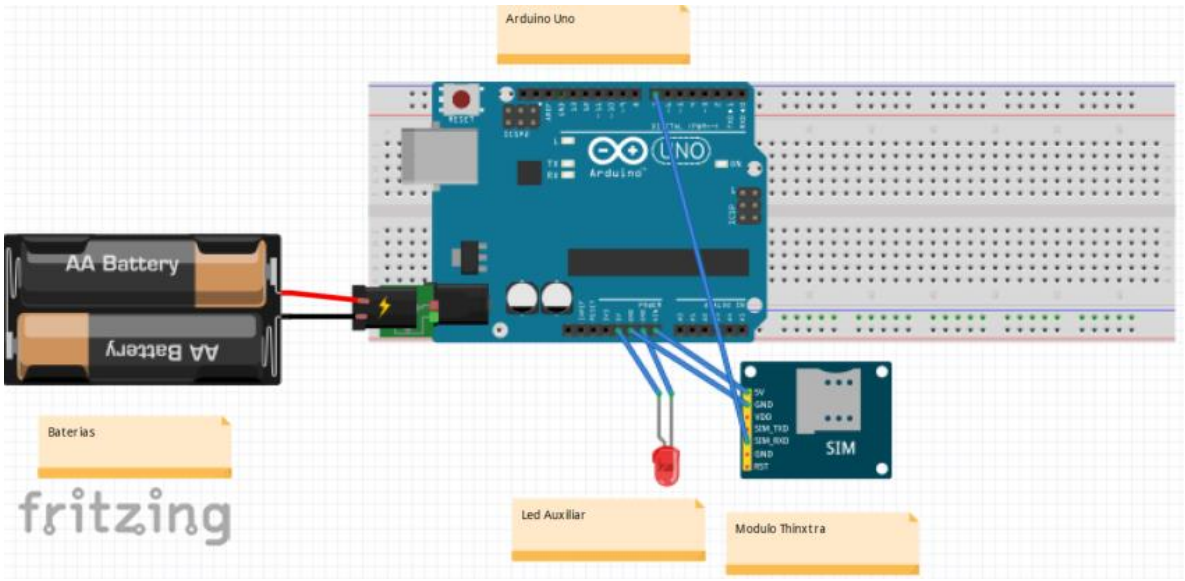
Manual Técnico

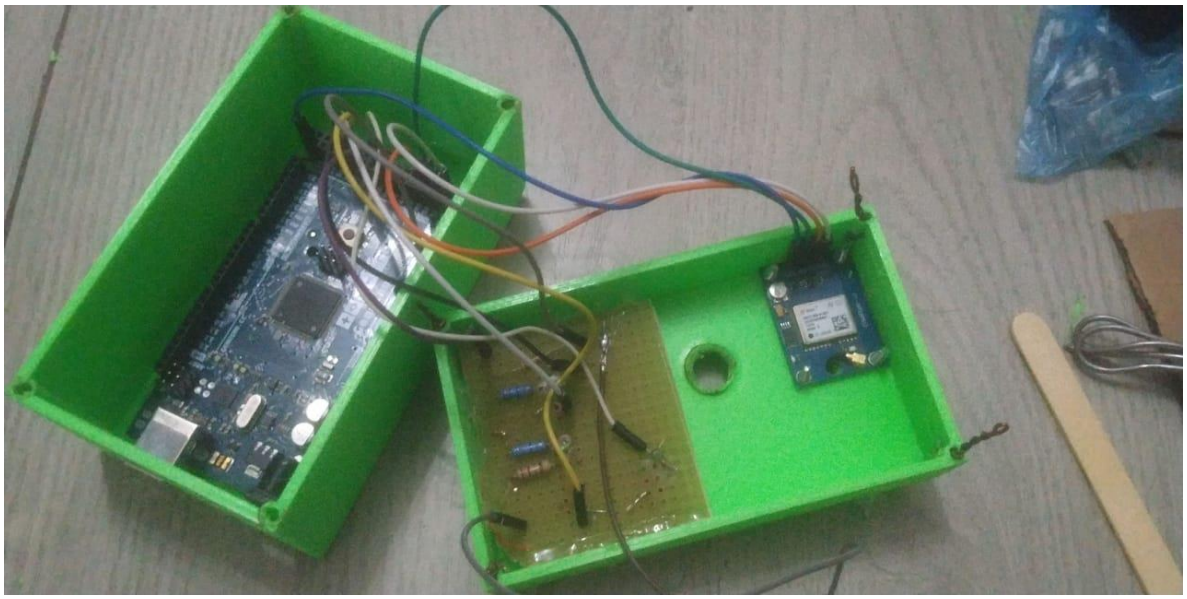
Requisitos

- Arduino Mega
- Dispositivo Thinxtra_DevKit
- Sensor gps gy-gps6mv2
- Pulsador, resistencia,led
- Software Android studio
- Backend Sigfox online
- Backend Firebase online

Diseño de Circuito

Para realizar el esquema se utilizo el software fritzing del prototipo





Diseño de base de datos

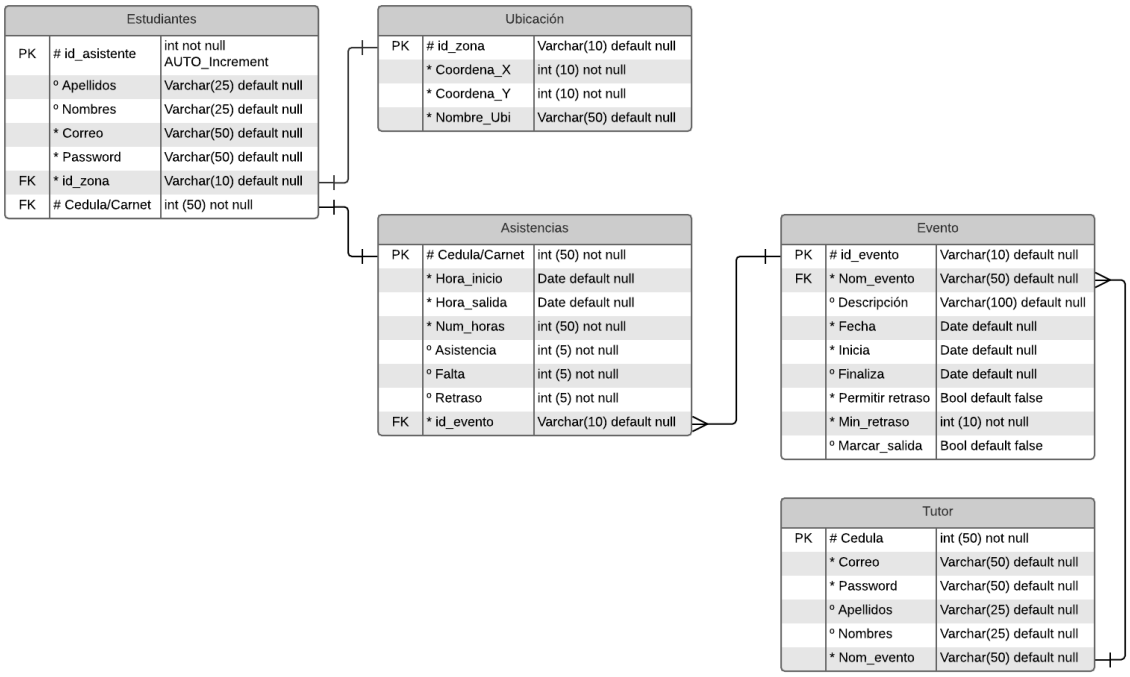


Ilustración 1.- Diagrama Entidad-Relación

Diseño de base de datos No relacional

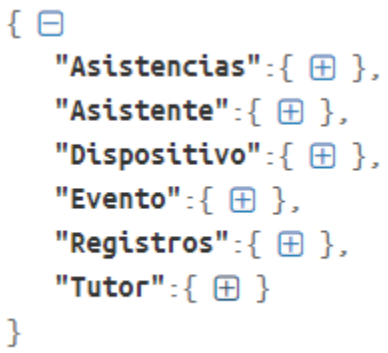


Ilustración 2.- Visión General de la base de datos no relacional en Firebase

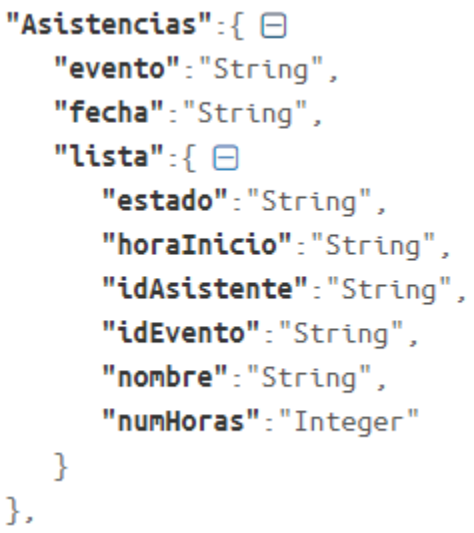


Ilustración 3.- Tabla Asistencia

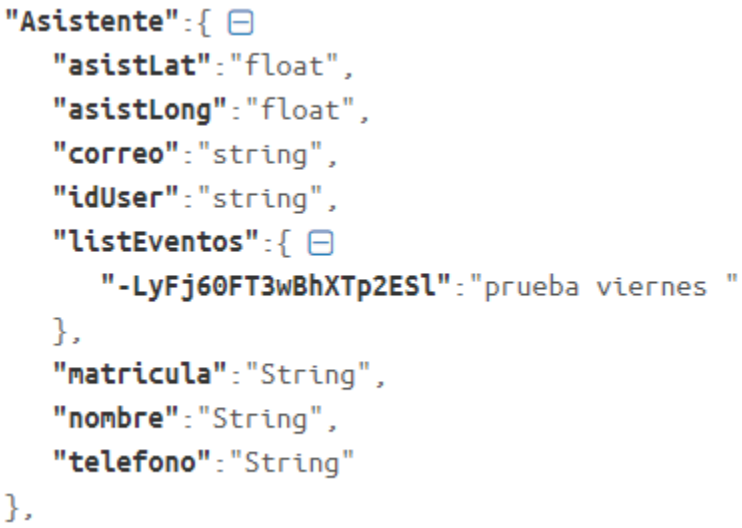


Ilustración 4.-Tabla Asistente

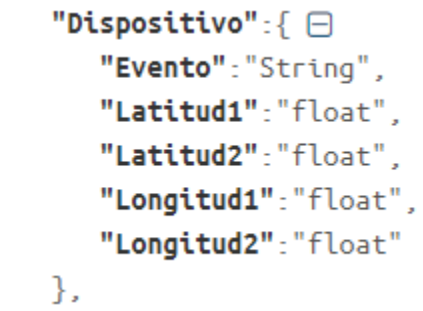


Ilustración 5.- Tabla Dispositivo

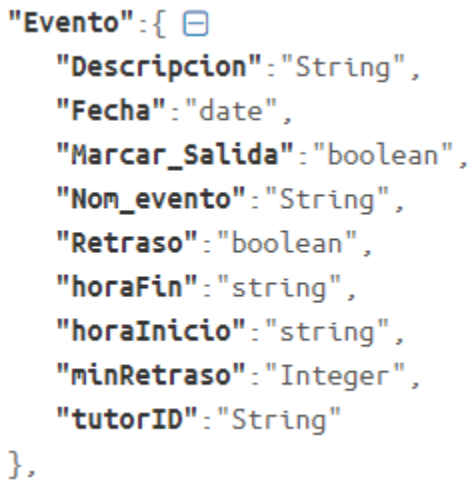


Ilustración 6.- Tabla Evento

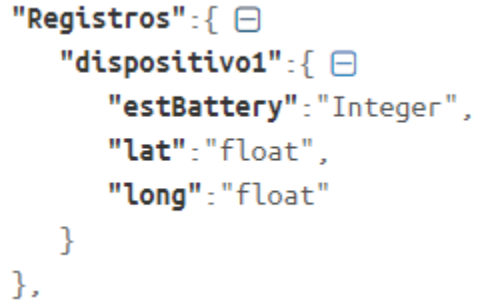


Ilustración 7.- Tabla Registro

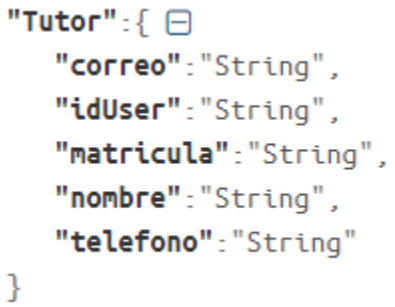


Ilustración 8.- Tabla tutor

Programación de módulo arduino para transmisión de señal a través de la red de Sigfox

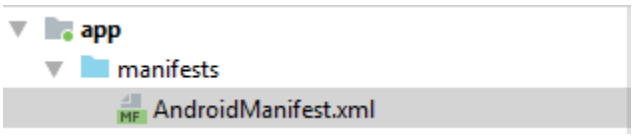
```
#include <SoftwareSerial.h>
#include <TinyGPS.h>

// Pines para los LEDs, boton, entrada analogica de estado de pila.
#define LEDVERDE 2
#define LEDAMARILLO 3
#define LEDROJO 4
#define ANALOGPILA 0
#define BOTON 2

//Variables para conexion del gps
TinyGPS gps;
SoftwareSerial serialgps(50, 51); // pin 50 Tx 51 Rx
unsigned long chars;
unsigned short sentences, failed_checksum;

struct gpscoord {
  float a_latitude;  // 4 bytes
  float a_longitude; // 4 bytes
};

int analogValor = 0;  // 2 bytes
float voltaje = 0;  // 4 bytes
int porcentaje_bateria;  // 2 bytes
int ledDelay = 800;  // 2 bytes

float maximo = 1.6;
float medio = 1.4;
float minimo = 0.3;


void setup() {
  //Inicializamos los led como salida.
  pinMode(LEDVERDE, OUTPUT);
  pinMode(LEDAMARILLO, OUTPUT);
```

```
  pinMode(LEDROJO, OUTPUT);
  pinMode(BOTON, INPUT_PULLUP);

  //Inicializamos los pines Seriales
  Serial.begin(9600);
  serialgps.begin(9600);
  Serial.println("");
  Serial.println(" --- Buscando Señal --- ");
  Serial.println("");

}

void loop() {

  //VIendo disponiblidad del puerto Serial del GPS.
  while (serialgps.available()) {

    //Lee el estado del boton
    int estado = digitalRead(BOTON);
    //Leyendo la información que viene del GPS.
    int c = serialgps.read();

    //Va agrupando toda la información del del GPS, hasta que sea una
linea.
    if (gps.encode(c)) {

      //Obtenemos la latitud y longitud del dispositivo.
      float latitude, longitude;
      gps.f_get_position(&latitude, &longitude);

      // se pasa los datos a la estructura
      gpscoord coords = {latitude, longitude};

      if (estado == LOW) {
        // enviamos por sigfox
        bool answer = sigfoxSend(&coords, sizeof(gpscoord));

        Serial.print("latitud: ");
        Serial.print(latitude, 5);
        Serial.print("\tlongitud: ");
        Serial.println(longitude, 5);

        gps.stats(&chars, &sentences, &failed_checksum);

        delay(100);

        // Leemos valor de la entrada analógica
        analogValor = analogRead(ANALOGPILA);

        // Obtenemos el voltaje
        voltaje = 0.0048 * analogValor;
        porcentaje_bateria = map (analogValor, 0, 1024, 0, 100);
        Serial.print("Voltaje: ");
        Serial.println(voltaje);
        Serial.print("Porcentaje: ");
        Serial.print(porcentaje_bateria);
        Serial.println("%");
```

```
        delay(1000);
      }

      //Segun el estado de la bateria se encenderan los leds
      if (voltaje >= maximo)
      {
        digitalWrite(LEDVERDE, HIGH);
        delay(ledDelay);
        digitalWrite(LEDVERDE, LOW);
      }
      else if (voltaje < maximo && voltaje > medio)
      {
        digitalWrite(LEDAMARILLO, HIGH);
        delay(ledDelay);
        digitalWrite(LEDAMARILLO, LOW);
      }
      else if (voltaje < medio && voltaje > minimo)
      {
        digitalWrite(LEDROJO, HIGH);
        delay(ledDelay);
        digitalWrite(LEDROJO, LOW);
      }

      // Apagamos todos los LEDs
      digitalWrite(LEDVERDE, LOW);
      digitalWrite(LEDAMARILLO, LOW);
      digitalWrite(LEDROJO, LOW);
    }
  }

}
//Metodo que requiere del data que es son la coordenadas y su longitud
//La data es transformada en mensajes hexadecimales para ser enviada a
traves del puerto serial.
bool sigfoxSend(const void* data, uint8_t len) {
  uint8_t* bytes = (uint8_t*)data;
  Serial.println("AT$RC");
  Serial.print("AT$SF=");
  for (uint8_t i = len - 1; i < len; --i) {
    if (bytes[i] < 16) {
      Serial.print("0");
    }
    Serial.print(bytes[i], HEX);
  }
  Serial.print('\r');
}
```

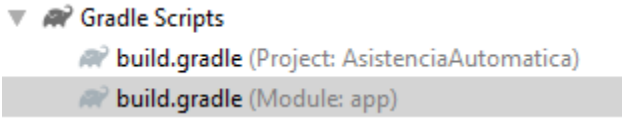
Desarrollo de Aplicación Móvil Utilizando Software Android Studio

Es importante definir cuál o cuáles son los permisos que se le asigna a la aplicación para que pueda ejecutarse correctamente
En el manifiesto de android añadiremos los permisos declarados a continuación que permitirá el uso de nuestros sensores incluidos en nuestro dispositivo, como gps o wifi, además de los permisos de lectura de datos en el smartphone



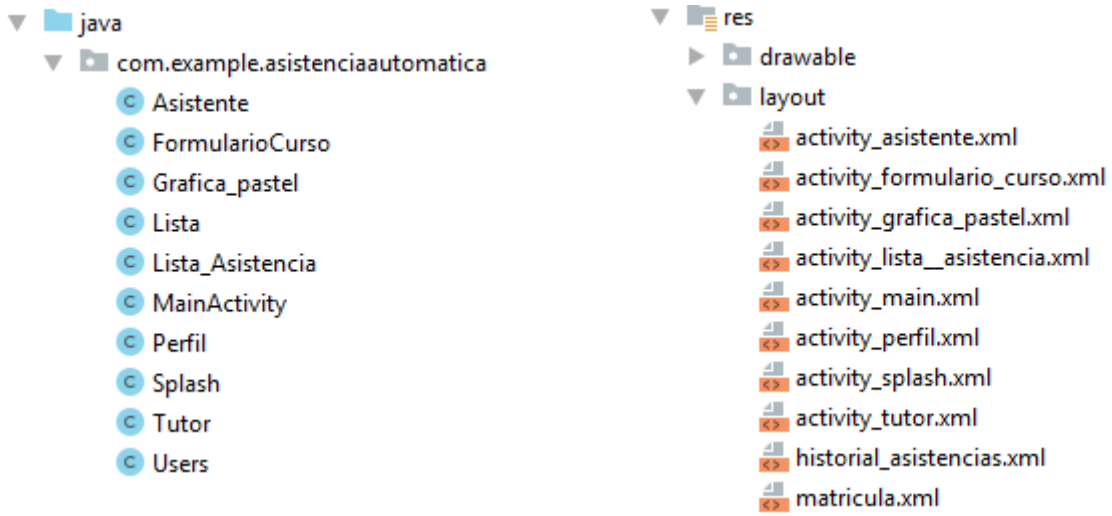

```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"
/>
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"
/>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"
/>
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"
/>
```

Ahora se agregara las dependencias necesarias para que nuestro aplicativo funcione con los objetos externos (como firebase) y se pueda obtener la data que se necesite o la función necesaria de dicha dependencia

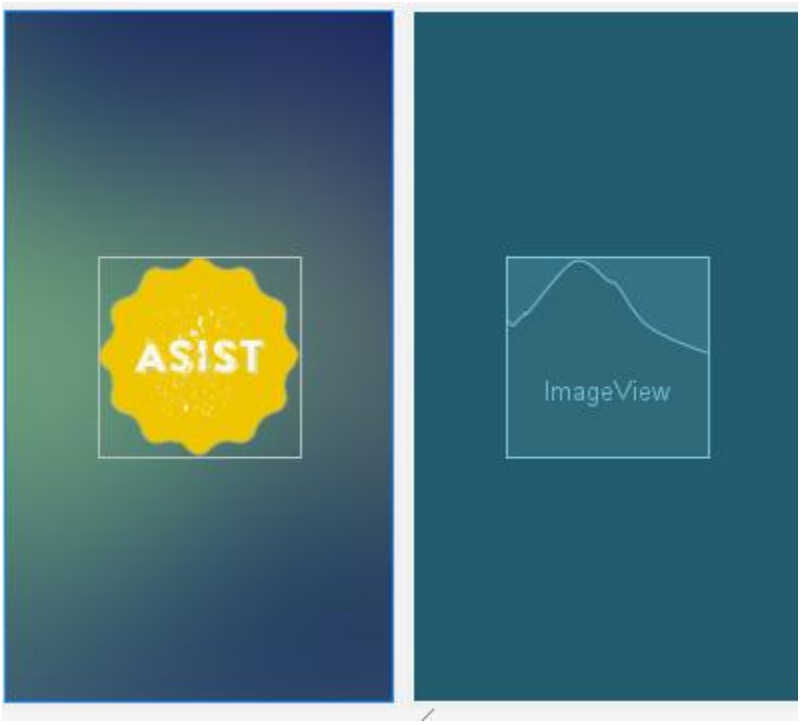



```
dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'androidx.appcompat:appcompat:1.1.0'
    implementation 'androidx.constraintlayout:constraintlayout:1.1.3'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'androidx.test.ext:junit:1.1.1'
    androidTestImplementation 'androidx.test.espresso:espresso-
core:3.2.0'
    implementation 'com.google.firebase:firebase-analytics:17.2.1'
    implementation 'com.google.firebase:firebase-auth:19.2.0'
    implementation 'com.google.firebase:firebase-core:17.2.1'
    implementation 'com.google.android.gms:play-services-auth:17.0.0'
    implementation 'com.squareup.picasso:picasso:2.71828'
    implementation 'com.google.firebase:firebase-database:19.2.0'
    implementation 'com.google.android.gms:play-services-maps:17.0.0'
    implementation 'com.google.android.gms:play-services-location:17.0.0'
    implementation 'androidx.recyclerview:recyclerview:1.1.0'
    implementation 'com.github.PhilJay:MPAndroidChart:v3.0.3'


}
```

Una vez terminado de configurar los recursos necesarios se empieza a crear nuestra aplicación



La **actividad splash** es una ventana de inicio donde se muestra el logo de la aplicación como se muestra a continuación

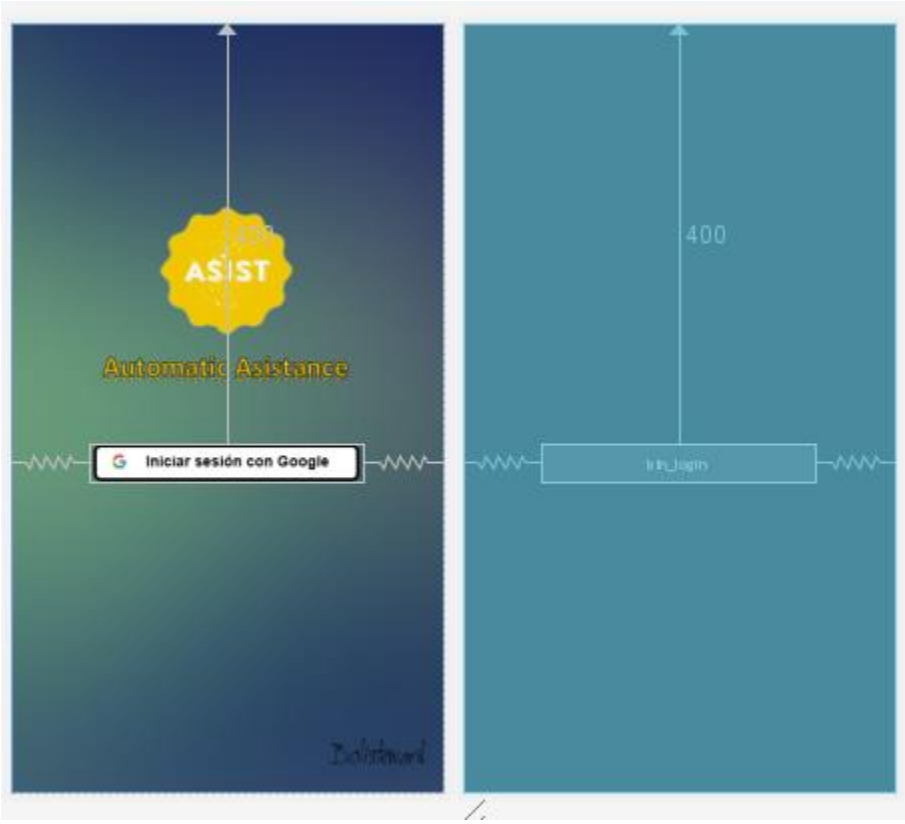


En la **clase Splash** se configura dentro de la función onCreate el tiempo de duración del splash

```
new Handler().postDelayed(() -> {
    Intent intent = new Intent(Splash.this, MainActivity.class);
    startActivity(intent);
},2000);
```

Nota.- se deberá cambiar dentro del manifiesto el intent filter a la actividad correspondiente para el inicio de la aplicación

La actividad **Main Activity** tendrá la función de permitir al usuario obtener los recursos de la aplicación iniciando sesión con su cuenta de google a través de un botón



La clase para **Main Activity** tendrá:

Variables privadas para su posterior instancia

```
private static final int GOOGLE_SIGN_IN = 123;
private FirebaseAuth mAuth;
private GoogleSignInClient mGoogleSignInClient;
```

Función iniciarSesion se encargará de intanciar el inicio de sesión con la cuenta de google de un usuario

```
public void iniciarSesion(View view){
    Intent signInIntent = mGoogleSignInClient.getSignInIntent();
    startActivityForResult(signInIntent, GOOGLE_SIGN_IN);
}
```

La función cerrar sesión permite finalizar la conexión con la cuenta del usuario que se ha logoneado, además también se encargara de cerrar la instancia en caso de un usuario y contraseña erróneo

```
private void cerrarSesion() {
    mGoogleSignInClient.signOut().addOnCompleteListener(this,task ->
```

```
updateUI(null));
}
```

La función `firebaseAuthWithGoogle` permite autentificar con la cuenta de google que ha iniciado

```
private void firebaseAuthWithGoogle(GoogleSignInAccount account) {
    Log.d("TAG", "firebaseAuthWithGoogle: "+account.getId());
    AuthCredential credential =
GoogleAuthProvider.getCredential(account.getIdToken(), null);
    mAuth.signInWithCredential(credential).addOnCompleteListener(this,
task -> {
        if (task.isSuccessful()) {
            FirebaseUser user = mAuth.getCurrentUser();
            MainActivity.this.updateUI(user);
        } else {
            Toast.makeText(getApplicationContext(), "Error de inicio de
sesion con firebase", Toast.LENGTH_SHORT).show();
            System.out.println("error");
            MainActivity.this.updateUI(null);
        }
    });
}
```

Función `updateUI` permite obtener los datos de usuario como nombre de cuenta, correo, teléfono y otros.

```
private void updateUI(FirebaseUser user) {
    if (user != null){

        Bundle info_user = new Bundle();

        info_user.putString("user_name", user.getDisplayName());
        info_user.putString("user_email", user.getEmail());
        info_user.putString("user_photo",
String.valueOf(user.getPhotoUrl()));
        info_user.putString("user_id", user.getUid());

        if (user.getPhoneNumber() !=null){
            info_user.putString("user_phone", user.getPhoneNumber());
            System.out.println("Si tiene numero celular");
        }else {
            info_user.putString("user_phone", "Sin numero");

            System.out.println("Sin numero");
        }

        finish();

        Intent intent = new Intent(MainActivity.this, Perfil.class);
        intent.putExtra("info_user", info_user);
```

```
        startActivity(intent);

    }else {
        Toast.makeText(getApplicationContext(),"Aun no se ha registrado
en google.", Toast.LENGTH_SHORT).show();
        System.out.println("Sin registrarse");
    }
}
```

La función sobrescrita permitirá obtener información sobre el estado de inicio de sesión o si obtuvo un error al iniciar sesión que será mostrado por un log en consola

```
@Override
protected void onActivityResult(int requestCode, int resultCode,
@Nullable Intent data) {
    super.onActivityResult(requestCode, resultCode, data);

    if (requestCode == GOOGLE_SIGN_IN){
        Task<GoogleSignInAccount> task =
GoogleSignIn.getSignedInAccountFromIntent(data);
        try {
            GoogleSignInAccount account =
task.getResult(ApiException.class);
            if (account != null) firebaseAuthWithGoogle (account);
        }catch (ApiException e){
            Log.w("TAG", "Fallo el inicio de sesion con Google.",e);
        }
    }
}
```

Y para que las funciones anteriores se ejecuten correctamente se implementara en la función onCreate con los siguientes comandos:

```
mAuth =  FirebaseAuth.getInstance();
GoogleSignInOptions gso = new GoogleSignInOptions.Builder(
        GoogleSignInOptions.DEFAULT_SIGN_IN).requestIdToken(getString(
        R.string.default_web_client_id)).requestEmail().build();
mGoogleSignInClient = GoogleSignIn.getClient(this, gso);

Intent intent = getIntent();
String msg = intent.getStringExtra("msg");

if(msg != null){
    if(msg.equals("cerrarSesion")){
        cerrarSesion();
    }
}
```

La **actividad perfil** permite elegir el tipo de usuario que se ha logoneado ya sea tutor o asistente que será evaluado en la base de datos

La **clase Perfil** tiene dos funciones que son iniciar actividad asistente o iniciar actividad tutor dependiendo del tipo de usuario que ha accedido

```
public void startAsistente(View view){
    Intent intent = new Intent(this, Asistente.class);
    intent.putExtra("info_user", info_user );
    startActivity(intent);
}

public void startTutor(View view){
    Intent intent = new Intent(this, Tutor.class);
    intent.putExtra("info_user", info_user );
    startActivity(intent);

}
```

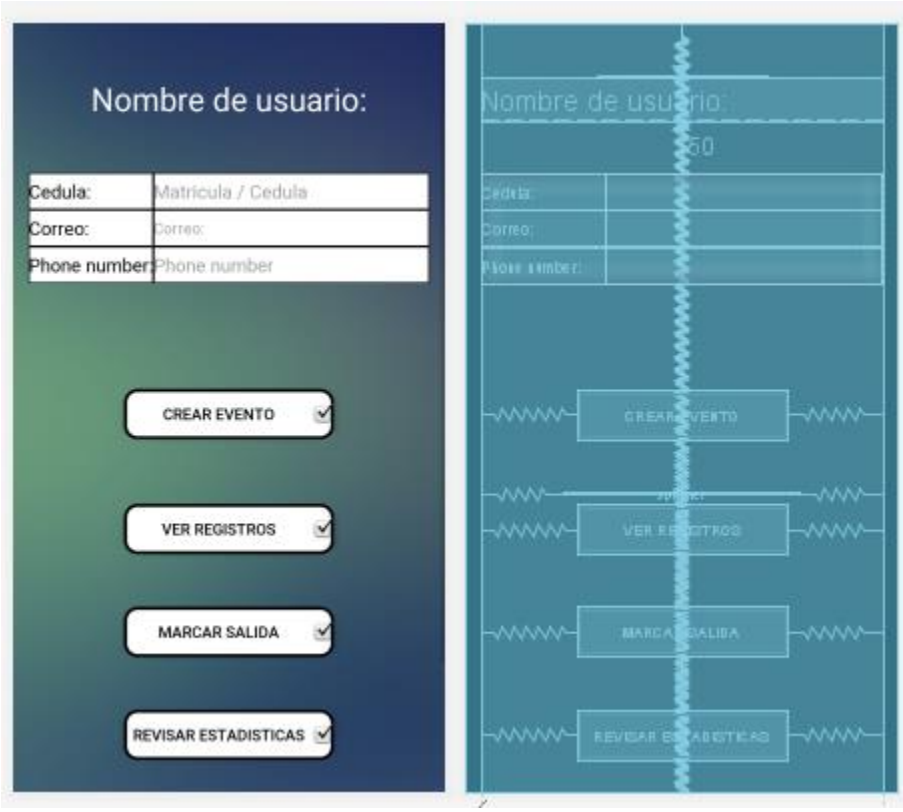
En la función onCreate por defecto se implementara la siguiente línea para obtener la información del usuario proporcionada por la actividad de inicio de sesión

```
info_user = getIntent().getBundleExtra("info_user");
```

Nota: se define los siguientes objetos

```
private static final String TAG = "Perfil";
private Bundle info_user;
```

La **actividad tutor** es el que tendrá los privilegios para la creación de eventos y ajustar la zona en la que se dará el evento, verificar registros y asistencias, además de estadísticas de asistencia de eventos creados



La **clase Tutor**
Variables de entorno

```
private static final String TAG = "Tutor";
private static final int ERROR_DIALOG_REQUEST = 9001;
```

La función permite iniciar la instancia de conexión con la base de datos

```
private void iniciarBaseDeDatos(){
    db_reference = FirebaseDatabase.getInstance().getReference();

}
```

El método leerBaseDatos recorre los asistentes en la base de datos para identificar si el usuario ya ha iniciado sesión previamente para directamente cargar la información en la interfaz gráfica, o si es un usuario nuevo. Este método implementa el botón de Crear Eventos enviando como información extra en el llamado de la activity el userId.

```
private void leerBaseDatos(){
    DatabaseReference db_tutor = db_reference.child("Tutor");

    db_tutor.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
            nuevoTutor = true;
            info_user = getIntent().getBundleExtra("info_user");
```

```
            if (info_user!=null) {
                for (DataSnapshot snapshot : dataSnapshot.getChildren())
{
                    HashMap<String, String> data = (HashMap<String,
String>) snapshot.getValue();
                    if (data != null) {
                        userId = data.get("idUser");
System.out.println(info_user.getString("user_id"));


                        if
(userId.equals(info_user.getString("user_id"))) {
                            nuevoTutor = false;
                            System.out.println(nuevoTutor);
                            presentarDatos();
                            break;
                        }
                        System.out.println("ok");
                    }
                }
            }
            System.out.println(nuevoTutor);
            System.out.println(userId);
            if (nuevoTutor){
                newTutor();
            }
            btn_CrearEvento.setOnClickListener(v -> {
                Intent intent = new Intent(Tutor.this,
FormularioCurso.class);
                intent.putExtra("tutorID", userId);
                startActivity(intent);
            });

        }
        @Override
        public void onCancelled(@NonNull DatabaseError error) {
            Log.e(TAG, "Error!", error.toException());
            System.out.println(error.getMessage());
        }
    });
}
```

Se recorre la sección Evento de la base de datos para agregar todos los nombres de los eventos existentes en el spinner View, y se implementa su acción al ser accedido por el usuario para implementar los métodos de visualización de asistencia, de grafico estadístico y poder marcar la salida del evento.

```
private void leerEventos(){
    DatabaseReference db_evento = db_reference.child("Evento");


    db_evento.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
            eventos = new ArrayList<String>();
```

```
            eventos.add("Seleccione un Evento");
            info_user = getIntent().getBundleExtra("info_user");

            for (DataSnapshot snapshot : dataSnapshot.getChildren()) {

                HashMap<String, String> data = (HashMap<String, String>)
snapshot.getValue();

                if (data!= null) {
                    if
(data.get("tutorID").equals(info_user.getString("user_id"))){
                        eventos.add(data.get("Nom_evento"));

                    }
                }
            }


            ArrayAdapter<String> adapter = new
ArrayAdapter<String>(getApplicationContext(),android.R.layout.simple_spin
ner_item, eventos);

adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_
item );
            spinner.setAdapter(adapter);

            AdapterView.OnItemSelectedListener eventSelected = new
AdapterView.OnItemSelectedListener() {

                @Override
                public void onItemSelected(AdapterView<?> spinner, View
container, int position, long id) {
                    name_evento =
spinner.getItemAtPosition(position).toString();
                    if (position!=0) {
                        Toast.makeText(Tutor.this,"Ha seleccionado el
evento: " + name_evento, Toast.LENGTH_SHORT).show();
                        DatabaseReference db_eventoInfo =
db_reference.child("Evento");
                        db_eventoInfo.addValueEventListener(new
ValueEventListener() {
                            @Override
                            public void onDataChange(@NonNull
DataSnapshot dataSnapshot) {
                                for (DataSnapshot snapshot :
dataSnapshot.getChildren()) {

                                    HashMap<String, String> dataEvento =
(HashMap<String, String>) snapshot.getValue();
                                    if (dataEvento!= null &&
dataEvento.get("Nom_evento").equals(name_evento)) {
                                        info_evento = dataEvento;
                                        break;
                                    }
                                }

                                verAsistencias(name_evento);
```

```
                                marcarSalida(name_evento);
                                verEstadisticas(name_evento);
                            }

                            @Override
                            public void onCancelled(@NonNull
DatabaseError databaseError) {
                                Log.e(TAG, "Error!",
databaseError.toException());
                            }
                        });
                    }

                }

                @Override
                public void onNothingSelected(AdapterView<?> arg0) {
                    // TODO Auto-generated method stub
                }
            };
            spinner.setOnItemSelectedListener(eventSelected);

        }
        @Override
        public void onCancelled(@NonNull DatabaseError error) {
            Log.e(TAG, "Error!", error.toException());
        }
    });
}
```

El método verAsistencias requiere del parámetro @evento e implementa la acción del botón verRegistros en el cual se crea un objeto Intent para llamar la Activity Lista_Asistencia y envía el nombre del evento seleccionado, caso contrario si evento esta vacío mostrara un mensaje pidiendo seleccionar un evento.

```
private void verAsistencias(String evento){
    btn_verRegistros.setOnClickListener(v -> {
        if (evento!=null) {
            Intent intent = new Intent(Tutor.this,
Lista_Asistencia.class);
            intent.putExtra("evento", evento);
            startActivity(intent);
        }else{
            Toast.makeText(Tutor.this,"Seleccione un evento o curso
primero." + name_evento, Toast.LENGTH_SHORT).show();
        }
    });
}
```

El método verEstadisticas requiere del parámetro evento e implementa la acción del botón btn_estadisticas en el cual se crea un objeto Intent para llamar la Activity Grafica_pastel y envía el nombre del evento seleccionado, caso contrario si evento esta vacío mostrara un mensaje pidiendo seleccionar un evento.

```
private void verEstadisticas(String evento){
    btn_estadisticas.setOnClickListener(v -> {
```

```
        if (evento!=null) {
            Intent intent = new Intent(Tutor.this, Grafica_pastel.class);
            intent.putExtra("evento", evento);
            startActivity(intent);
        }else{
            Toast.makeText(Tutor.this,"Seleccione un evento o curso
primero." + name_evento, Toast.LENGTH_SHORT).show();
        }
    });
}
```

El método marcarSalida requiere del parámetro evento e implementa la acción del botón btn_marcaSalida en el cual se valida el nombre del evento seleccionado del spinner con la base de datos y se verifica que la opción de marcar salida haya sido seleccionado, caso contrario mostrara un mensaje diciendo que la acción es innecesaria.

```
private void marcarSalida(String evento){
    btn_marcaSalida.setOnClickListener(v -> {

        if (evento!=null && !evento.equals("Seleccione un Evento") &&
info_evento!=null) {
            DatabaseReference db_mSalida = db_reference.child("Evento");

            if (info_evento.get("Nom_evento").equals(evento)) {
                if (info_evento.get("Marcar_Salida").equals("true")) {
                    horaFinAsistentes(evento);
                } else {
                    Toast.makeText(Tutor.this, "El evento seleccionado se
cierra automaticamente a la hora estipulada en la creacion del evento." +
name_evento, Toast.LENGTH_SHORT).show();
                }

            }
        }else{
            Toast.makeText(Tutor.this,"Seleccione un evento o curso
primero.", Toast.LENGTH_SHORT).show();
        }
    });

}
```

El método horaFinAsistentes toma como parámetro el nombre del evento obtenido del spinner para colocar la hora de finalización del evento en todos los asistentes de dicho evento con el numero de horas totales a partir de su respectiva hora de asistencia.

```
private void horaFinAsistentes(String evento){
    Calendar calendar = Calendar.getInstance();
    int horas=calendar.get(Calendar.HOUR_OF_DAY);
    int minutos=calendar.get(Calendar.MINUTE);
    String horaFin= horas+":"+minutos;

    DatabaseReference db_horaFin = db_reference.child("Asistencias");

    db_horaFin.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {

            for (DataSnapshot snapshot : dataSnapshot.getChildren()) {
```

```
                HashMap<String, String> dataLista = (HashMap<String,
String>) snapshot.getValue();

                if (dataLista!= null) {
                    if (dataLista.get("evento").equals(evento)){

                        DatabaseReference ref =
db_horaFin.child(snapshot.getKey()).child("lista");
                        ref.addValueEventListener(new
ValueEventListener() {
                            @Override
                            public void onDataChange(@NonNull
DataSnapshot dataSnapshot) {
                                for (DataSnapshot snapshot :
dataSnapshot.getChildren()) {
                                    HashMap<String, String> dataUser =
(HashMap<String, String>) snapshot.getValue();

                                    if (snapshot.getKey()!=null &&
dataUser!=null) {

ref.child(snapshot.getKey()).child("horaFin").setValue(horaFin);
                                        String[] horaInicio =
dataUser.get("horaInicio").split(":");

                                        if
(Integer.parseInt(horaInicio[0]) == horas) {
                                            int minTotal = minutos-
Integer.parseInt(horaInicio[1]);
                                            String horaFinAsist =
0+"."+minTotal+"h";

ref.child(snapshot.getKey()).child("numHoras").setValue(horaFinAsist);
                                        }else{
                                            int horasPresente = horas -
Integer.parseInt(horaInicio[0]);

                                            if (minutos >
Integer.parseInt(horaInicio[1])) {
                                                int minTotal = minutos -
Integer.parseInt(horaInicio[1]);
                                                String horaFinAsist =
horasPresente+"."+minTotal+"h";

ref.child(snapshot.getKey()).child("numHoras").setValue(horaFinAsist);
                                            }else {
                                                int minTotal = 60 +
minutos - Integer.parseInt(horaInicio[1]);
                                                String horaFinAsist =
(horasPresente-1)+"."+minTotal+"h";

ref.child(snapshot.getKey()).child("numHoras").setValue(horaFinAsist);
                                            }
                                        }
                                    }
                                }
```

```
                                Toast.makeText(Tutor.this, "Hora de
salida: " + horaFin, Toast.LENGTH_SHORT).show();
                            }

                            @Override
                            public void onCancelled(@NonNull
DatabaseError databaseError) {
                                Log.e(TAG, "Error!",
databaseError.toException());
                            }
                        });

                    }
                }
            }
        }

        @Override
        public void onCancelled(@NonNull DatabaseError databaseError) {
            Log.e(TAG, "Error!", databaseError.toException());
        }
    });
}
```

El método newTutor permite subir los datos obtenidos de la cuenta de google con el que el usuario inicia sesión, se pide al usuario que ingrese el número de cedula mediante un cuadro de dialogo, se sube toda la informacion obtenida a la base de datos en Firebase.

```
private void newTutor() {
    info_user = getIntent().getBundleExtra("info_user");
    if (info_user != null) {
        txt_nombre.setText(info_user.getString("user_name"));
        txt_cellphone.setText(info_user.getString("user_phone"));
        txt_correo.setText(info_user.getString("user_email"));
        userId = info_user.getString("user_id");
        String photo = info_user.getString("user_photo");
        Picasso.get().load(photo).resize(300,
300).error(R.drawable.usuario).into(img_foto);


        tutor = new Users(info_user.getString("user_name"),
info_user.getString("user_email"), info_user.getString("user_phone"),
info_user.getString("user_id"));

        DatabaseReference db_upload = db_reference.child("Tutor");

        db_upload.child(userId).setValue(tutor);

        createCustomDialog().show();
    }
}
```

Cuando el ususario ya existe se implementa el metodo presentarDatos, el cual permite tomar los datos del usuario de la base de datos y cargarlos en los respectivos TextView's del archivo tutor.xml.

```
private void presentarDatos(){
    info_user = getIntent().getBundleExtra("info_user");

    if (info_user != null) {
        userId = info_user.getString("user_id");
        String photo = info_user.getString("user_photo");
        Picasso.get().load(photo).resize(300,
300).error(R.drawable.usuario).into(img_foto);
    }
    DatabaseReference db_asist =
db_reference.child("Tutor").child(userId);
    db_asist.addListenerForSingleValueEvent(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
            Users usr = dataSnapshot.getValue(Users.class);
            if (usr!=null) {
                txt_nombre.setText(usr.getNombre());
                txt_correo.setText(usr.getCorreo());
                txt_cellphone.setText(usr.getTelefono());
                txt_cedula.setText(usr.getMatricula());
            }
        }

        @Override
        public void onCancelled(@NonNull DatabaseError databaseError) {
            Log.e(TAG, "Error!", databaseError.toException());
        }
    });

}
```

Se crea un cuadro de dialogo de tipo AlertDialog el cual utiliza el archivo matricula.xml como interfaz gráfica, para pedir el número de cedula o matricula. Se obtiene el dato ingresado y es subido directamente a la base de datos del usuario registrado.

```
private AlertDialog createCustomDialog() {
    final AlertDialog alertDialog;
    final AlertDialog.Builder builder = new AlertDialog.Builder(this);
    // Get the layout inflater
    LayoutInflater inflater = getLayoutInflater();
    // Inflar y establecer el layout para el dialogo
    // Pasar nulo como vista principal porque va en el diseño del diálogo
    View v = inflater.inflate(R.layout.matricula, null);
    //builder.setView(inflater.inflate(R.layout.dialog_signin, null))
    EditText edtMatricula = v.findViewById(R.id.edtMatricula);
    Button btn_aceptar = v.findViewById(R.id.btn_aceptar);
    builder.setView(v);
    alertDialog = builder.create();
    // Add action buttons
    btn_aceptar.setOnClickListener(
            v1 -> {

                txt_cedula.setText(edtMatricula.getText().toString());
                //System.out.println("el numero de matricula es
"+edtMatricula.getText().toString());
                //System.out.println(userId);
```

```
                DatabaseReference db_upload =
FirebaseDatabase.getInstance().getReference().child("Tutor").child(userId
);

db_upload.child("matricula").setValue(edtMatricula.getText().toString());

                alertDialog.dismiss();
            }

    );
    return alertDialog;
}
```

Se cierra la sesión de la cuenta google con la cual ingreso el usuario y es enviado directamente a la MainActivity

```
public void cerrarSesion(View view){
    FirebaseAuth.getInstance().signOut();
    finish();
    Intent intent = new Intent(this, MainActivity.class);
    intent.putExtra("msg", "cerrarSesion");
    startActivity(intent);
}
```

Verifica si el servicio de google servicie esta activo para el correcto funcionamiento de las API's de google utilizadas como geo localización, googleAccount.

```
private boolean isServiceOk(){
    Log.d(TAG, "isServiceOk: checking google service version");

    int available =
GoogleApiAvailability.getInstance().isGooglePlayServicesAvailable(Tutor.t
his);

    if (available == ConnectionResult.SUCCESS){
        //Everything is fine and the user can make map request
        return true;
    } else
if(GoogleApiAvailability.getInstance().isUserResolvableError(available)){
        //an error ocured but we can resolt it
        Log.d(TAG, "isServiceOk: an error occures but we can fix it");
        Dialog dialog =
GoogleApiAvailability.getInstance().getErrorDialog(Tutor.this, available,
ERROR_DIALOG_REQUEST);
        dialog.show();
    }else{
        Toast.makeText(this, "You can't make map request",
Toast.LENGTH_SHORT).show();
    }
    return false;
}
```

Dentro de la función onCreate de la misma Clase se agrega:

```
if (isServiceOk()) {
    iniciarBaseDeDatos();
    leerBaseDatos();
```

```
        leerEventos();
        marcarSalida(name_evento);
        verAsistencias(name_evento);
        verEstadisticas(name_evento);

    }
```

La clase **User** se creo para dar soporte a la información del usuario obtenida al iniciar sesión

```
public class Users {
    private String nombre;
    private String correo;
    private String telefono;
    private String latitud;
    private String longitud;
    private Lista idlista;
    private String photo;
    private String idUser;
    private String matricula;

    public Users() {
    }

    public Users(String nombre, String correo, String telefono, String idUser) {
        this.nombre = nombre;
        this.correo = correo;
        this.telefono = telefono;
        this.idUser = idUser;
    }

    public String getNombre() { return nombre; }

    public String getCorreo() { return correo; }

    public String getTelefono() { return telefono; }

    public String getMatricula() { return matricula; }

}
```

La actividad lista_asistencia mostrara todos los usuarios que se han registrado en el evento asi como su estado

El método leerAsistencia recorre la sesión Asistencias de la base de datos para determinar que lista de asistencia pertenece el nombre del evento, una vez verificada se recorre esta lista y se muestran en la respectiva interfaz gráfica a través del método monstrarRegistrosPorPantalla que toma como parámetro un objeto tipo DataSnapshot.

```
private void leerAsistencia() {
    db_reference.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
            for (DataSnapshot snapshot : dataSnapshot.getChildren()) {

                HashMap<String, String> data = (HashMap<String, String>)
snapshot.getValue();
                if (data!= null &&
data.get("evento").equals(name_evento)) {
                    DatabaseReference db_lista =
db_reference.child(snapshot.getKey()).child("lista");
                    db_lista.addValueEventListener(new
ValueEventListener() {
                        @Override
                        public void onDataChange(@NonNull DataSnapshot
dataSnapshot) {
                            for (DataSnapshot snapshot :
dataSnapshot.getChildren()) {
                                mostrarRegistrosPorPantalla(snapshot);
                            }
                        }

                        @Override
```

```
                        public void onCancelled(@NonNull DatabaseError
databaseError) {
                            Log.e(TAG, "Error!",
databaseError.toException());
                        }
                    });
                    break;
                }
            }
        }

        @Override
        public void onCancelled(@NonNull DatabaseError error) {
            Log.e(TAG, "Error!", error.toException());
        }
    });
}
```

El método mostrarRegistrosPorPantalla tomo como parámetro un DataSanpshot para obtener la información del estudiante y desglosarla para presentarla en los respectivos componentes tipo View del archivo Lista_Asistencias.xml, se configura el tamaño del texto.

```
private void mostrarRegistrosPorPantalla(DataSnapshot snapshot){
        LinearLayout contNombre = findViewById(R.id.ContenedorNombre);
        LinearLayout contEstado= findViewById(R.id.ContenedorEstado);
        LinearLayout contHoraInicio= findViewById(R.id.ContenedorHora);
        LinearLayout contNumHoras=
findViewById(R.id.ContenedorCantHoras);

        String Name =
String.valueOf(snapshot.child("nombre").getValue());
        String estado =
String.valueOf(snapshot.child("estado").getValue());
        String horas =
String.valueOf(snapshot.child("horaInicio").getValue());
        String cantHoras =
String.valueOf(snapshot.child("numHoras").getValue());


        TextView userName = new TextView(getApplicationContext());
        userName.setText(Name);
        userName.setTextSize(15);
        userName.setTextColor(Color.WHITE);
        userName.setPadding(15,0,0,0);
        contNombre.addView(userName);

        TextView Estado = new TextView(getApplicationContext());
        Estado.setText(estado);
        Estado.setTextSize(15);
        Estado.setTextColor(Color.WHITE);
        Estado.setPadding(15,0,0,0);
        contEstado.addView(Estado);

        TextView horaI = new TextView(getApplicationContext());
        horaI.setText(horas);
        horaI.setTextSize(15);
        horaI.setTextColor(Color.WHITE);
```

```
        horaI.setPadding(15,0,0,0);
        contHoraInicio.addView(horaI);

        TextView numHoras = new TextView(getApplicationContext());
        numHoras.setText(cantHoras);
        numHoras.setTextSize(15);
        numHoras.setTextColor(Color.WHITE);
        numHoras.setPadding(15,0,0,0);
        contNumHoras.addView(numHoras);
    }


}
```

La actividad asistente permitirá al mismo asistir a un evento programado, además puede revisar su estado e historial de asistencia

Para la clase Asistente se implemento los siguientes métodos:

El método newAsist permite subir los datos obtenidos de la cuenta de google con el que el usuario inicia sesion, implemente el metodo getLocationPermission() para obtener la ubicacion y pide al usuario que ingrese el numero de matricula mediante un cuadro de dialogo.

```
private void newAsist() {
        info_user = getIntent().getBundleExtra("info_user");
        if (info_user != null) {
            txt_nombre.setText(info_user.getString("user_name"));
```

```
            txt_phone.setText(info_user.getString("user_phone"));
            txt_correo.setText(info_user.getString("user_email"));
            userId = info_user.getString("user_id");
            String photo = info_user.getString("user_photo");
            Picasso.get().load(photo).resize(300,
300).error(R.drawable.usuario).into(img_foto);


            asistente = new Users(info_user.getString("user_name"),
info_user.getString("user_email"), info_user.getString("user_phone"),
info_user.getString("user_id"));

            DatabaseReference db_upload =
db_reference.child("Asistente");

            db_upload.child(userId).setValue(asistente);

            if (isServiceOk()) {
                getLocationPermission();

            }
            createCustomDialog().show();
        }
    }
```

Se crea un cuadro de dialogo de tipo AlertDialog el cual utiliza el archivo matricula.xml como interfaz grafica Se obtiene el dato ingresado y es subido directamente a la base de datos del usuario registrado.

```
    private AlertDialog createCustomDialog() {
        final AlertDialog alertDialog;
        final AlertDialog.Builder builder = new
AlertDialog.Builder(this);
        // Get the layout inflater
        LayoutInflater inflater = getLayoutInflater();
        // Inflar y establecer el layout para el dialogo
        // Pasar nulo como vista principal porque va en el diseño del
diálogo
        View v = inflater.inflate(R.layout.matricula, null);
        //builder.setView(inflater.inflate(R.layout.dialog_signin, null))
        EditText edtMatricula = v.findViewById(R.id.edtMatricula);
        Button btn_aceptar = v.findViewById(R.id.btn_aceptar);
        builder.setView(v);
        alertDialog = builder.create();
        // Add action buttons
        btn_aceptar.setOnClickListener(
                v1 -> {


txt_matricula.setText(edtMatricula.getText().toString());

                    DatabaseReference db_upload =
FirebaseDatabase.getInstance().getReference().child("Asistente").child(us
erId);

db_upload.child("matricula").setValue(edtMatricula.getText().toString());
```

```
                    alertDialog.dismiss();
                }

        );
        return alertDialog;
    }
```

Cuando el ususario ya existe se implementa el metodo presentarDatos, el cual permite tomar los datos del usuario de la base de datos y cargarlos en los respectivos TextView's del archivo asistente.xml. Unicamente la ubicacion se actualiza.

```
    private void presentarDatos(){
        info_user = getIntent().getBundleExtra("info_user");

        if (info_user != null) {
            userId = info_user.getString("user_id");
            String photo = info_user.getString("user_photo");
            Picasso.get().load(photo).resize(300,
300).error(R.drawable.usuario).into(img_foto);

            if (isServiceOk()) {
                getLocationPermission();
            }
        }
        DatabaseReference db_asist =
db_reference.child("Asistente").child(userId);
        db_asist.addListenerForSingleValueEvent(new ValueEventListener()
{
            @Override
            public void onDataChange(@NonNull DataSnapshot dataSnapshot)
{
                Users usr = dataSnapshot.getValue(Users.class);
                if (usr!=null) {
                    txt_nombre.setText(usr.getNombre());
                    txt_correo.setText(usr.getCorreo());
                    txt_phone.setText(usr.getTelefono());
                    txt_matricula.setText(usr.getMatricula());
                }
            }

            @Override
            public void onCancelled(@NonNull DatabaseError databaseError)
{
                Log.e(TAG, "Error!", databaseError.toException());
            }
        });

    }
```

Se recorre la base de datos en firebase en la sesión Asistente para determinar si el usuario que ingresa es nuevo o ya ha ingresado anteriormente. Segun el caso, se llamara al respectivo metodo.

```
    private void leerBaseDatos(){
        DatabaseReference asistente = db_reference.child("Asistente");

        asistente.addValueEventListener(new ValueEventListener() {
```

```
            @Override
            public void onDataChange(@NonNull DataSnapshot dataSnapshot)
{
                nuevoAsist = true;
                info_user = getIntent().getBundleExtra("info_user");

                if (info_user!=null) {
                    for (DataSnapshot snapshot :
dataSnapshot.getChildren()) {
                        HashMap<String, String> data = (HashMap<String,
String>) snapshot.getValue();

                        if (data != null) {
                            String userId = data.get("idUser");

                            assert userId != null;
                            if
(userId.equals(info_user.getString("user_id"))) {
                                nuevoAsist = false;
                                presentarDatos();

                                break;
                            }
                        }
                    }
                }
                if (nuevoAsist){
                    newAsist();
                }
            }
            @Override
            public void onCancelled(@NonNull DatabaseError error) {
                Log.e(TAG, "Error!", error.toException());
                System.out.println(error.getMessage());
            }
        });
    }
```

El método marcarSalida() implemente la acción del botón de marcar salida del estudiante, en el se verifica el evento que ha seleccionado de la lista en el spinner y luego se compara y verifican la hora del asistente con respecto a la hora de inicio y finalizacion del evento para obtener la cantidad de horas asistidas.

```
    private void marcarSalida(){
        btn_salida.setOnClickListener(v -> {
            if (name_evento != null  && info_evento!=null &&
!name_evento.equals("Seleccione un Evento")) {
                String[] fecha_evento =
info_evento.get("Fecha").split("/");

                Calendar calendar = Calendar.getInstance();
                anio=calendar.get(Calendar.YEAR);
                mes=calendar.get(Calendar.MONTH)+1;
                dia=calendar.get(Calendar.DAY_OF_MONTH);
                int horas = calendar.get(Calendar.HOUR_OF_DAY);
                int minutos = calendar.get(Calendar.MINUTE);
```

```
                if (Conectividad()) {
                    if (Integer.parseInt(fecha_evento[0]) == anio && mes
==Integer.parseInt(fecha_evento[1]) && dia
==Integer.parseInt(fecha_evento[2])) {
                        Boolean presente = verifica_Asistencia();
                        if (presente) {
                            DatabaseReference db_mSalida =
db_reference.child("Asistencias");

                            db_mSalida.addValueEventListener(new
ValueEventListener() {
                                @Override
                                public void onDataChange(@NonNull
DataSnapshot dataSnapshot) {

                                    String id_lista = null;

                                    for (DataSnapshot snapshot :
dataSnapshot.getChildren()) {

                                        HashMap<String, String> data =
(HashMap<String, String>) snapshot.getValue();

                                        if (data != null) {
                                            if
(data.get("evento").equals(name_evento)) {

                                                id_lista =
snapshot.getKey();
                                                break;
                                            }
                                        }
                                    }

                                    if (id_lista!=null){
                                        DatabaseReference db_lista =
db_mSalida.child(id_lista).child("lista").child(userId);

                                        String horaFin = horas + ":" +
minutos;


db_lista.addValueEventListener(new ValueEventListener() {
                                            @Override
                                            public void
onDataChange(@NonNull DataSnapshot dataSnapshot) {

                                                HashMap<String, String>
dataUser = (HashMap<String, String>) dataSnapshot.getValue();

System.out.println(dataUser);

                                                String[] horaInicio =
dataUser.get("horaInicio").split(":");
                                                horaFinE =
info_evento.get("horaFin").split(":");
```

```
                                                    if
(Integer.parseInt(horaFinE[0])== horas  && minutos <=
Integer.parseInt(horaFinE[1])) {


db_lista.child("horaFin").setValue(horaFin);

                                                        if
(Integer.parseInt(horaInicio[0]) == horas) {
                                                            int minTotal =
minutos-Integer.parseInt(horaInicio[1]);
                                                            String
horaFinAsist = 0+"."+minTotal+"h";

System.out.println("aki1");

db_lista.child("numHoras").setValue(horaFinAsist);

Toast.makeText(Asistente.this, "Hora de salida: " + horaFin+" cant. horas
presente: "+horaFinAsist, Toast.LENGTH_SHORT).show();
                                                        } else {
                                                            int horasPresente
= horas - Integer.parseInt(horaInicio[0]);
                                                            if (minutos >
Integer.parseInt(horaInicio[1])) {
                                                                int minTotal
= minutos - Integer.parseInt(horaInicio[1]);
                                                                String
horaFinAsist = horasPresente+"."+minTotal+"h";

System.out.println("aki2");

db_lista.child("numHoras").setValue(horaFinAsist);

Toast.makeText(Asistente.this, "Hora de salida: " + horaFin+" cant. horas
presente: "+horaFinAsist, Toast.LENGTH_SHORT).show();
                                                            } else {
                                                                int minTotal
= 60 + minutos - Integer.parseInt(horaInicio[1]);
                                                                String
horaFinAsist = horasPresente+"."+minTotal+"h";

System.out.println("aki3");

db_lista.child("numHoras").setValue(horaFinAsist);

Toast.makeText(Asistente.this, "Hora de salida: " + horaFin+" cant. horas
presente: "+horaFinAsist, Toast.LENGTH_SHORT).show();
                                                            }
                                                        }
                                                    }else if
(Integer.parseInt(horaFinE[0]) > horas) {

db_lista.child("horaFin").setValue(horaFin);

                                                        if
```

```
(Integer.parseInt(horaInicio[0]) == horas) {
                                                        int minTotal =
minutos-Integer.parseInt(horaInicio[1]);
                                                        String
horaFinAsist = 0+"."+minTotal+"h";

System.out.println("aki4");

db_lista.child("numHoras").setValue(horaFinAsist);

Toast.makeText(Asistente.this, "Hora de salida: " + horaFin+" cant. horas
presente: "+horaFinAsist, Toast.LENGTH_SHORT).show();
                                                    } else {
                                                        int horasPresente
= horas - Integer.parseInt(horaInicio[0]);
                                                        if (minutos >
Integer.parseInt(horaInicio[1])) {
                                                            int minTotal
= minutos - Integer.parseInt(horaInicio[1]);
                                                            String
horaFinAsist = horasPresente+"."+minTotal+"h";

System.out.println("aki5");

db_lista.child("numHoras").setValue(horaFinAsist);

Toast.makeText(Asistente.this, "Hora de salida: " + horaFin+" cant. horas
presente: "+horaFinAsist, Toast.LENGTH_SHORT).show();
                                                        } else {
                                                            int minTotal
= 60 + minutos - Integer.parseInt(horaInicio[1]);
                                                            String
horaFinAsist = (horasPresente-1)+"."+minTotal+"h";

System.out.println("aki6");

db_lista.child("numHoras").setValue(horaFinAsist);

Toast.makeText(Asistente.this, "Hora de salida: " + horaFin+" cant. horas
presente: "+horaFinAsist, Toast.LENGTH_SHORT).show();
                                                        }
                                                    }
                                                }else{
                                                    int minTotal = 0;
                                                    int horaTotal =
Integer.parseInt(horaFinE[0]) - Integer.parseInt(horaInicio[0]);
                                                    if
(Integer.parseInt(horaFinE[1]) > Integer.parseInt(horaInicio[1])){
                                                        minTotal =
Integer.parseInt(horaFinE[1]) - Integer.parseInt(horaInicio[1]);

                                                    }else {
                                                        if (
Integer.parseInt(horaFinE[1]) < Integer.parseInt(horaInicio[1])){
                                                            minTotal = 60
+ Integer.parseInt(horaFinE[1])  - Integer.parseInt(horaInicio[1]);
                                                            horaTotal =
```

```
horaTotal-1;

                                                                    }
                                                                }
                                                                String horaFinAsist =
horaTotal+"."+minTotal+"h";

db_lista.child("numHoras").setValue(horaFinAsist);

db_lista.child("horaFin").setValue(info_evento.get("horaFin"));

Toast.makeText(Asistente.this, "Hora de salida: " +
info_evento.get("horaFin")+" cant. horas presente: "+horaFinAsist,
Toast.LENGTH_SHORT).show();
                                                        }
                                                    }

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

                                                        Log.e(TAG, "Error!",
databaseError.toException());
                                                    }
                                                });
                                            }
                                        }

                                        @Override
                                        public void onCancelled(@NonNull
DatabaseError databaseError) {
                                            Log.e(TAG, "Error!",
databaseError.toException());
                                        }
                                    });
                                } else {
                                    Toast.makeText(Asistente.this, "Se encuentra
fuera de la zona del evento: " + name_evento, Toast.LENGTH_SHORT).show();
                                }
                            } else {
                                Toast.makeText(Asistente.this, "Aun no empieza el
evento o el evento ya finalizo. Contactese con al tutor o administrador
del " +
                                        "evento para mayor informacion.",
Toast.LENGTH_LONG).show();
                            }
                        }else{
                            Toast.makeText(Asistente.this, "No dispone de
conexion a Internet.", Toast.LENGTH_LONG).show();
                        }
                    } else {
                        Toast.makeText(Asistente.this, "Seleccione un evento o
curso primero." + name_evento, Toast.LENGTH_SHORT).show();
                    }
                });

            }
```

El metodo Asistir verifica el evento existente y extrae las coordenadas de la zona del evento y las compara con las del estudiante para validar la asistencia. Ademas, se realiza la respectiva validacion de la fecha y hora del evento.

```
    private void Asistir(){
        btn_Asistir.setOnClickListener(v -> {
            Calendar calendar = Calendar.getInstance();
            anio=calendar.get(Calendar.YEAR);
            mes=calendar.get(Calendar.MONTH)+1;
            dia=calendar.get(Calendar.DAY_OF_MONTH);
            horas=calendar.get(Calendar.HOUR_OF_DAY);
            minutos=calendar.get(Calendar.MINUTE);
            horaActual= horas+":"+minutos;

            if (name_evento!=null && !name_evento.equals("Seleccione un
Evento")) {
                if (info_evento!= null) {

                    String[] fecha_evento =
info_evento.get("Fecha").split("/");
                    String[] hora_evento =
info_evento.get("horaInicio").split(":");
                    String[] hora_finEvento =
info_evento.get("horaFin").split(":");
                    int minRetrado =
Integer.parseInt(info_evento.get("minRetraso"));
                    Boolean retraso =
Boolean.parseBoolean(info_evento.get("Retraso"));

                    if (Conectividad()) {
                        if (Integer.parseInt(fecha_evento[0]) == anio &&
Integer.parseInt(fecha_evento[1]) == mes &&
Integer.parseInt(fecha_evento[2]) == dia) {

                            if (retraso) {
                                if (horas ==
Integer.parseInt(hora_evento[0]) && minutos <=
(Integer.parseInt(hora_evento[1]) + minRetrado)
                                        && minutos >=
Integer.parseInt(hora_evento[1])) {
                                    subirAsistencia(false);

                                } else if (horas >=
Integer.parseInt(hora_evento[0]) && minutos >
(Integer.parseInt(hora_evento[1]) + minRetrado)) {
                                    if (horas ==
Integer.parseInt(hora_finEvento[0]) && minutos <=
Integer.parseInt(hora_finEvento[1])) {
                                        subirAsistencia(true);

                                    } else if (horas <
Integer.parseInt(hora_finEvento[0])) {
                                        subirAsistencia(true);

                                    } else {
                                        Toast.makeText(Asistente.this,
```

```
"Es posible que el evento ya haya finalizado. \n Contactese con al tutor
o administrador del " +
                                                "evento para mayor
informacion.", Toast.LENGTH_SHORT).show();

                                    }
                                } else {
                                    Toast.makeText(Asistente.this, "Aun
no empieza el evento. Contactese con al tutor o administrador del " +
                                                    "evento para mayor
informacion.", Toast.LENGTH_SHORT).show();

                                }
                            } else {
                                if (horas >=
Integer.parseInt(hora_evento[0])) {
                                    if (horas >
Integer.parseInt(hora_finEvento[0])) {
                                        Toast.makeText(Asistente.this,
"Es posible que el evento ya haya finalizado. \n Contactese con al tutor
o administrador del " +
                                                "evento para mayor
informacion.", Toast.LENGTH_SHORT).show();

                                    } else if (horas ==
Integer.parseInt(hora_finEvento[0]) && minutos <=
Integer.parseInt(hora_finEvento[1])) {
                                        subirAsistencia(false);

                                    } else if (horas <
Integer.parseInt(hora_finEvento[0])) {
                                        subirAsistencia(false);

                                    } else {
                                        Toast.makeText(Asistente.this,
"Es posible que el evento ya haya finalizado. \n Contactese con al tutor
o administrador del " +
                                                "evento para mayor
informacion.", Toast.LENGTH_SHORT).show();

                                    }
                                } else {
                                    Toast.makeText(Asistente.this, "Aun
no empieza el evento o el evento ya finalizo. \n Contactese con al tutor
o administrador del " +
                                                    "evento para mayor
informacion.", Toast.LENGTH_SHORT).show();
                                }
                            }

                        } else {
                            Toast.makeText(Asistente.this, "Aun no
empieza el evento o el evento ya finalizo. Contactese con al tutor o
administrador del " +
                                    "evento para mayor informacion.",
Toast.LENGTH_LONG).show();
                        }
```

```
                    }else{
                        Toast.makeText(Asistente.this, "No dispone de
conexion a Internet.", Toast.LENGTH_LONG).show();
                    }
                }
            }else{
                Toast.makeText(Asistente.this, "Escoja el evento/curso
primero",Toast.LENGTH_SHORT).show();
            }
        });
    }
```

El método subirAsistencia permite subir la asistencia del usuario a la base de datos en firebase, en las respectivas ramas de Asistente y Asistencias, tomando como dato previo el parametro tipo Boolean de atrasado.

```
    private void subirAsistencia(Boolean atrasado) {
        boolean present = verifica_Asistencia();
        boolean atraso = atrasado;
        DatabaseReference db_listaAsistencia =
db_reference.child("Asistencias");
        DatabaseReference db_dataAsistente =
db_reference.child("Asistente").child(userId).child("listEventos");

        db_dataAsistente.child(idEvento).setValue(name_evento);
        System.out.println(present+"-"+atrasado);

        db_listaAsistencia.addValueEventListener(new ValueEventListener()
{
            @Override
            public void onDataChange(@NonNull DataSnapshot dataSnapshot)
{
                HashMap<String, String> info_lista = null;
                for (DataSnapshot snapshot : dataSnapshot.getChildren())
{
                    info_lista = (HashMap<String, String>)
snapshot.getValue();
                    if (info_lista!= null) {
                        if (info_lista.get("evento").equals(name_evento))
{
                            idLista = snapshot.getKey();
                        }else{
                            System.out.println("no lista");
                        }
                    }
                }
                System.out.println(idLista);

                if (atraso) {
                    Lista lista = new Lista(userId,
txt_nombre.getText().toString(), horaActual, "Atrasado", idEvento,0);

db_listaAsistencia.child(idLista).child("lista").child(userId).setValue(l
ista);
                    Toast.makeText(Asistente.this, "Asistencia confirmada
al evento: "+name_evento+" como: Atrasado", Toast.LENGTH_SHORT).show();
                }else{
```

```
                    Lista lista = new Lista(userId,
txt_nombre.getText().toString(), horaActual, "Presente", idEvento,0);

db_listaAsistencia.child(idLista).child("lista").child(userId).setValue(l
ista);
                    Toast.makeText(Asistente.this, "Asistencia confirmada
al evento: "+name_evento+" como: Presente", Toast.LENGTH_SHORT).show();
                }
                System.out.println("Data Asistente subido");

            }

            @Override
            public void onCancelled(@NonNull DatabaseError databaseError)
{
                Log.e(TAG, "Error!", databaseError.toException());
            }
        });



    }
```

El metodo verifica_Asistencia() permite verificar si el estudiante se encuentra dentro de la zona de asistencia del evento, caso contrario enviara un mensaje de aviso. Regresa como valor un valor tipo Boolean con la respuesta de la validacion.

```
    private boolean verifica_Asistencia(){
        boolean presente =false;
        Double user_lat =
Double.valueOf(txt_Latitud.getText().toString());
        Double user_long =
Double.valueOf(txt_Longitud.getText().toString());
        Double Dps_Lat1 = Double.valueOf(disp_Lat1);
        Double Dps_Lat2 = Double.valueOf(disp_Lat2);
        Double Dps_Long1 = Double.valueOf(disp_Long1);
        Double Dps_Long2 = Double.valueOf(disp_Long2);

        System.out.println(disp_Lat1+"-"+disp_Lat2);
        System.out.println(user_lat);
        System.out.println(disp_Long1+"-"+disp_Long2);
        System.out.println(user_long);

        if (Dps_Lat1 >=Dps_Lat2) {
            if ((Dps_Long1>=Dps_Long2) && (user_lat<=Dps_Lat1) &&
(user_lat>=Dps_Lat2) && (user_long<=Dps_Long1) && (user_long
>=Dps_Long2)){
                presente = true;
            }
            if ((Dps_Long1<=Dps_Long2) && (user_lat<=Dps_Lat1) &&
(user_lat>=Dps_Lat2) && (user_long>=Dps_Long1) && (user_long
<=Dps_Long2)) {
                presente = true;
            }
            if (!presente) {
                Toast.makeText(getApplicationContext(), "Se encuentra
fuera del rango", Toast.LENGTH_SHORT).show();
```

```
            }
        }
        if (Dps_Lat1 <=Dps_Lat2) {
            if ((Dps_Long1>=Dps_Long2) && (user_lat>=Dps_Lat1) &&
(user_lat<=Dps_Lat2) && (user_long<=Dps_Long1) && (user_long
>=Dps_Long2)){
                presente = true;
            }
            if ((Dps_Long1<=Dps_Long2) && (user_lat>=Dps_Lat1) &&
(user_lat<=Dps_Lat2) && (user_long>=Dps_Long1) && (user_long
<=Dps_Long2)) {
                presente = true;
            }
            if (!presente) {
                Toast.makeText(getApplicationContext(), "Se encuentra
fuera del rango", Toast.LENGTH_SHORT).show();
            }
        }

        return presente;
    }
```

Se recorre la seccion Evento de la base de datos para agregar todos los nombres de los eventos existentes en el spinner View, y se implementa su accion al ser accedido por el usuario para obtener las corrdenas del evento seleccionado a traves del metodo leerDispositivo

```
    private void leerEventos(){
        DatabaseReference db_evento = db_reference.child("Evento");

        db_evento.addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot dataSnapshot)
{
                eventos = new ArrayList<String>();
                eventos.add("Seleccione un Evento");

                for (DataSnapshot snapshot : dataSnapshot.getChildren())
{

                    HashMap<String, String> data = (HashMap<String,
String>) snapshot.getValue();

                    if (data!= null) {
                        eventos.add(data.get("Nom_evento"));
                    }
                }

                ArrayAdapter<String> adapter = new
ArrayAdapter<String>(getApplicationContext(),android.R.layout.simple_spin
ner_item, eventos);

adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_
item );
                spinner.setAdapter(adapter);

                AdapterView.OnItemSelectedListener eventSelected = new
```

```
AdapterView.OnItemSelectedListener() {

                    @Override
                    public void onItemSelected(AdapterView<?> spinner,
View container, int position, long id) {
                        name_evento =
spinner.getItemAtPosition(position).toString();
                        if (position!=0) {
                            Toast.makeText(Asistente.this,"Ha
seleccionado el evento: " + name_evento, Toast.LENGTH_SHORT).show();
                            DatabaseReference db_eventoAsistir =
db_reference.child("Evento");

                            if (name_evento!=null &&
!name_evento.equals("Seleccione un Evento")) {

db_eventoAsistir.addValueEventListener(new ValueEventListener() {
                                    @Override
                                    public void onDataChange(@NonNull
DataSnapshot dataSnapshot) {
                                        for (DataSnapshot snapshot :
dataSnapshot.getChildren()) {

                                            HashMap<String, String>
dataEvento = (HashMap<String, String>) snapshot.getValue();
                                            if (dataEvento!= null &&
dataEvento.get("Nom_evento").equals(name_evento)) {
                                                info_evento = dataEvento;
                                                idEvento =
snapshot.getKey();

                                                break;
                                            }
                                        }
                                        leerDispositivo(name_evento);
                                    }

                                    @Override
                                    public void onCancelled(@NonNull
DatabaseError databaseError) {
                                        Log.e(TAG, "Error!",
databaseError.toException());
                                    }
                                });
                            }
                        }

                    }

                    @Override
                    public void onNothingSelected(AdapterView<?> arg0) {
                        // TODO Auto-generated method stub
                    }
                };
                spinner.setOnItemSelectedListener(eventSelected);

            }
            @Override
```

```
            public void onCancelled(@NonNull DatabaseError error) {
                Log.e(TAG, "Error!", error.toException());
            }
        });
    }
```

Se recorre la sesion Eventos de la base de datos y se compara con el @parametro ingresado curso para extraer las coordenadas de dicho evento.

```
     private void leerDispositivo(String curso){
        DatabaseReference db_dispositivo =
db_reference.child("Dispositivo");
        db_dispositivo.addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot dataSnapshot)
{

                for (DataSnapshot snapshot : dataSnapshot.getChildren())
{
                    HashMap<String, String> data = (HashMap<String,
String>) snapshot.getValue();
                    if (data!=null) {
                        if (data.get("Evento").equals(curso)) {
                            disp_Lat1 = data.get("Latitud1");
                            disp_Long1 = data.get("Longitud1");
                            disp_Lat2 = data.get("Latitud2");
                            disp_Long2 = data.get("Longitud2");
                            break;
                        }
                    }
                }
                Asistir();
                marcarSalida();
            }
            @Override
            public void onCancelled(@NonNull DatabaseError error) {
                Log.e(TAG, "Error!", error.toException());
                System.out.println(error.getMessage());
            }
        });
    }
```

Devuelve un valor tipo Bool indicando si hay o no conectividad del dispositivo con alguna red de internet.

```
    private boolean Conectividad(){
        ConnectivityManager connectivityManager = (ConnectivityManager)
getSystemService(Context.CONNECTIVITY_SERVICE);
        NetworkInfo networkInfo =
connectivityManager.getActiveNetworkInfo();

        if (networkInfo != null && networkInfo.isConnected()) {
            return true;
        } else {
            return  false;
        }
```

```
    }
```

El método Asistir permite obtener las coordenadas de latitud y longitud del dispositivo en ese instante y los sobre-escribe en el txt_Latitud y txt_longitud de la interfaz, si se produce un error mandara una ioException o un mensaje de que la localizacion no se encuentra o es nula.

```
    private void getDeviceLocation(){
        Log.d(TAG, "getDeviceLocation: getting device current location");
        mFusedLocationProviderClient =
LocationServices.getFusedLocationProviderClient(this);
        try{
            if(mLocationPermissionGaranted){

                final Task location =
mFusedLocationProviderClient.getLastLocation();

                location.addOnCompleteListener(task -> {
                    if(task.isSuccessful()){
                        Log.d(TAG, "onComplete: found location!");
                        Location currentLocation = (Location)
task.getResult();
                        if (currentLocation !=null) {

txt_Latitud.setText(String.valueOf(currentLocation.getLatitude()));

txt_Longitud.setText(String.valueOf(currentLocation.getLongitude()));
                            DatabaseReference db_upload =
FirebaseDatabase.getInstance().getReference().child("Asistente").child(us
erId);

db_upload.child("asistLat").setValue(String.valueOf(currentLocation.getLa
titude()));

db_upload.child("asistLong").setValue(String.valueOf(currentLocation.getL
ongitude()));
                        }

                    }else{
                        Log.d(TAG, "onComplete: current location is
null");
                        Toast.makeText(Asistente.this, "unable to get
current location", Toast.LENGTH_SHORT).show();
                    }
                });
            }
        }catch (SecurityException e){
            Log.e(TAG, "getDeviceLocation: SecurityException: " +
e.getMessage() );
        }

    }
```

El método getLocationPermission verifica que los permisos y privilegios hayan sido aceptado, de no ser asi llama al metodo onResquestPermissionsResults para solicitarlos.

```
    private void getLocationPermission(){
```

```
        Log.d(TAG, "getLocationPermission: getting location
permission.");
        String[] permissions = {Manifest.permission.ACCESS_FINE_LOCATION,
Manifest.permission.ACCESS_COARSE_LOCATION};
        if
(ContextCompat.checkSelfPermission(this.getApplicationContext(),
                FINE_LOCATION)== PackageManager.PERMISSION_GRANTED){
            if
(ContextCompat.checkSelfPermission(this.getApplicationContext(),
                    COURSE_LOCATION)==
PackageManager.PERMISSION_GRANTED){
                mLocationPermissionGaranted = true;
                getDeviceLocation();
            }else{
                ActivityCompat.requestPermissions(this, permissions,
LOCATION_PERMISSION_REQUEST_CODE);
            }
        }else{
            ActivityCompat.requestPermissions(this, permissions,
LOCATION_PERMISSION_REQUEST_CODE);
        }
    }
```

El método onRequestPermissionResult permite solicitar al usuario los permisos de poder utilizar su ubicacion otorgandole a la app los privilegios para obtener las datos de latitud y longitud. De no ser asi, mostrara un mensaje de falla o no concedido.

```
    @Override
    public void onRequestPermissionsResult(int requestCode, @NonNull
String[] permissions, @NonNull int[] grantResults) {
        mLocationPermissionGaranted = false;
        Log.d(TAG, "onRequestPermissionsResult: called.");
        switch ( requestCode){
            case LOCATION_PERMISSION_REQUEST_CODE:{
                if (grantResults.length >0){
                    for (int i =0; i < grantResults.length; i++) {
                        if (grantResults[i] !=
PackageManager.PERMISSION_GRANTED) {
                            mLocationPermissionGaranted = false;
                            Log.d(TAG, "onRequestPermissionsResult:
failed.");
                            return;
                        }
                    }
                    mLocationPermissionGaranted =true;
                    Log.d(TAG, "onRequestPermissionsResult: permission
granted.");
                    //obtener la localizacion
                    getDeviceLocation();
                }
            }
        }
    }



}
```

El formulario Curso permitirá crear el evento asi como asignar la zona donde se lo realizará

La clase FormularioCurso contiene:

Se recorre la sesion Eventos de la base de datos para extrar todo el nombre e id de eventos registrados para validar que sean unicos. Se implementa la accion de los botones de fecha, horaInicio y horaFin que muestran un Date y Time Picker Dialog para la sellecion de la fehca y hora del evento. De igual forma para la accion de los

botones de ubicacion1 y ubicacion 2 para seleccionar la zona del evento.

```
    private void leerEventos(){
        DatabaseReference db_evento = db_reference.child("Evento");

        db_evento.addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot dataSnapshot)
{

                for (DataSnapshot snapshot : dataSnapshot.getChildren())
{

                    HashMap<String, String> data = (HashMap<String,
String>) snapshot.getValue();

                    if (data!= null) {
                        eventos.put(snapshot.getKey(),
data.get("Nom_evento"));
                    }
                }

                btn_Fecha.setOnClickListener(v -> {

                    final Calendar c = Calendar.getInstance();
                    anio=c.get(Calendar.YEAR);
                    mes=c.get(Calendar.MONTH);
                    dia=c.get(Calendar.DAY_OF_MONTH);
                    DatePickerDialog datePicker = new
DatePickerDialog(FormularioCurso.this, (view, year, month, dayOfMonth) ->
etFecha.setText(year+"/"+(month+1)+"/"+dayOfMonth),anio,mes,dia);

                    datePicker.show();

                });

                btn_HoraInicio.setOnClickListener(v -> {

                    Date date = new Date();
                    DateFormat hourFormat = new
SimpleDateFormat("HH:mm:ss", Locale.getDefault());
                    String[] hora_actual =
hourFormat.format(date).split(":");
                    horas = Integer.valueOf(hora_actual[0]);
                    minutos = Integer.valueOf(hora_actual[1]);

                    TimePickerDialog ponerhora= new
TimePickerDialog(FormularioCurso.this, (view, hourOfDay, minute) ->
etHoraInicio.setText(hourOfDay+":"+minute),horas,minutos,true);


                    ponerhora.show();
                });
                btn_HoraFin.setOnClickListener(v -> {

                    Date date = new Date();
                    DateFormat hourFormat = new
SimpleDateFormat("HH:mm:ss", Locale.getDefault());
```

```
                    String[] hora_actual =
hourFormat.format(date).split(":");
                    horaFin = Integer.valueOf(hora_actual[0]);
                    minFin = Integer.valueOf(hora_actual[1]);

                    TimePickerDialog ponerhoraFin= new
TimePickerDialog(FormularioCurso.this, (view, hourOfDay, minute) -> {

                        String horaFinal = hourOfDay+":"+minute;
                        CompararTiempo(etHoraInicio.getText().toString(),
horaFinal);

                    },horaFin,minFin,true);

                    ponerhoraFin.show();
                });

                btn_Ubicacion1.setOnClickListener(v -> {
                    if (isServiceOk()) {
                        numDispositivo = 1;
                        if (box_Dispositivo.isChecked()) {
                            String
name_disp=edtDispositivo.getText().toString();
                            dispositivoGPS(name_disp);
                        }else {
                            getLocationPermission();
                        }
                    }
                });
                btn_Ubicacion2.setOnClickListener(v -> {
                    if (isServiceOk()) {
                        numDispositivo = 2;
                        if (box_Dispositivo.isChecked()) {
                            String name_disp=
edtDispositivo.getText().toString();
                            dispositivoGPS(name_disp);
                        }else {
                            getLocationPermission();
                        }
                    }
                });


            }
            @Override
            public void onCancelled(@NonNull DatabaseError error) {
                Log.e(TAG, "Error!", error.toException());
            }
        });
    }
```

El metodo dispositivoGPS requiere del parametro nameDisp para obtener los datos del gps del dispositivo IOT cuyos datos se encuentran en la seccion Registros cuyos id's son el nombre o codigo unico del dispositivo.
Solo se implementa este metodo cuando se selecciona el checkBox box_Dispositivo.

```
    private void dispositivoGPS(String nameDisp){
```

```
        DatabaseReference db_dispositivo =
db_reference.child("Registros");

        db_dispositivo.addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot dataSnapshot)
{
                Boolean existe = false;
                HashMap<String, String> info_disp = null;

                for (DataSnapshot snapshot : dataSnapshot.getChildren())
{
                    if (snapshot!=null &&
snapshot.getKey().equals(edtDispositivo.getText().toString())){
                        System.out.println(nameDisp+"hola");
                        info_disp =(HashMap<String, String>)
snapshot.getValue();
                        existe = true;
                        break;
                    }
                }
                if (existe && info_disp!=null) {
                    System.out.println(existe);
                    DatabaseReference db_dispo =
db_dispositivo.child(nameDisp);
                    db_dispo.addValueEventListener(new
ValueEventListener() {
                        @Override
                        public void onDataChange(@NonNull DataSnapshot
dataSnapshot) {
                            HashMap<String, String> info_dps = null;
                            String key = null;
                            for (DataSnapshot snapshot :
dataSnapshot.getChildren()) {
                                info_dps =(HashMap<String, String>)
dataSnapshot.getValue();
                                key=snapshot.getKey();
                                break;
                            }
                            if (info_dps!=null && key!=null) {

                                DatabaseReference base =
db_dispo.child(key);
                                base.addValueEventListener(new
ValueEventListener() {
                                    @Override
                                    public void onDataChange(@NonNull
DataSnapshot dataSnapshot) {
                                        HashMap<String, String> info
=null;
                                        HashMap<String, String> punto
=null;
                                        Boolean nulidad = false;
                                        String keydato = null;
                                        for (DataSnapshot snapshot :
dataSnapshot.getChildren()){
                                            info = (HashMap<String,
```

```
String>) dataSnapshot.getValue();
                                            if (info!=null) {
                                                if
(!info.get("lat").equals("0")) {
                                                    punto =
(HashMap<String, String>) dataSnapshot.getValue();
                                                    nulidad = true;
                                                    keydato =
dataSnapshot.getKey();

System.out.println(keydato);
                                                    break;
                                                }
                                                System.out.println("no
existe");
                                            }
                                        }
                                        if (punto!=null && nulidad &&
keydato!=null){
                                            System.out.println(info);
                                            System.out.println("aki");
                                            if (numDispositivo == 1) {
                                                disp_Lat1 =
punto.get("lat");
                                                disp_Long1 =
punto.get("long");
                                                estadoBateria =
punto.get("estBateria");

System.out.println(numDispositivo);

System.out.println(numDispositivo);

System.out.println(disp_Lat1 + "-" + disp_Long1);

System.out.println(disp_Lat2 + "-" + disp_Long2);

Toast.makeText(FormularioCurso.this, "La bateria de su dispositivo es: "
+ estadoBateria, Toast.LENGTH_SHORT).show();
                                                DatabaseReference
eliminar = db_reference.child("Registros").child(nameDisp);

eliminar.child(keydato).removeValue();
                                            }
                                            if (numDispositivo == 2) {
                                                disp_Lat2 =
punto.get("lat");
                                                disp_Long2 =
punto.get("long");
                                                estadoBateria =
punto.get("estBateria");

System.out.println(numDispositivo);

System.out.println(disp_Lat1 + "-" + disp_Long1);

System.out.println(disp_Lat2 + "-" + disp_Long2);
```

```
Toast.makeText(FormularioCurso.this, "La bateria de su dispositivo es: "
+ estadoBateria, Toast.LENGTH_SHORT).show();
                                                    DatabaseReference
eliminar = db_reference.child("Registros").child(nameDisp);

eliminar.child(keydato).removeValue();
                                                    }
                                                    Guardar();
                                                }
                                            }

                                            @Override
                                            public void onCancelled(@NonNull
DatabaseError databaseError) {
                                                Log.e(TAG, "Error!",
databaseError.toException());
                                            }
                                        });

                                    }else{
                                        System.out.println("falla disp");
                                    }
                                }

                                @Override
                                public void onCancelled(@NonNull DatabaseError
databaseError) {
                                    Log.e(TAG, "Error!",
databaseError.toException());
                                }
                            });

                        }else{
                            Toast.makeText(FormularioCurso.this, "El codigo o
nombre del dispositivo ingresado no existe.", Toast.LENGTH_SHORT).show();
                        }

                    }

                    @Override
                    public void onCancelled(@NonNull DatabaseError databaseError)
{
                        Log.e(TAG, "Error!", databaseError.toException());
                    }
                });
            }
```

El método CompararTiempo toma dos parametros, la HoraInicio y la HoraFin del evento para compararlas entre si para validar que la hora final no sea menor a la hora incial del evento y se produzca un error en los registros delos estudiantes.

```
    private void CompararTiempo(String HoraInicio, String HoraFin){
        String[] TiempoInicial = HoraInicio.split(":");
        int horaI = Integer.valueOf(TiempoInicial[0]);
        int minuteI = Integer.valueOf(TiempoInicial[1]);
```

```
        String[] TiempoFinal = HoraFin.split(":");
        int horaF = Integer.valueOf(TiempoFinal[0]);
        int minuteF = Integer.valueOf(TiempoFinal[1]);

        if (horaF >= horaI){
            if (horaF == horaI && minuteF<=minuteI) {
                Toast.makeText(getApplicationContext(), "La Hora de
finalizacion no puede ser menor a la Hora de Inicio.",
Toast.LENGTH_SHORT).show();
            }
            else{
                etHoraFin.setText(HoraFin);
            }

        }else{
            Toast.makeText(getApplicationContext(), "La Hora de
finalizacion no puede ser menor a la Hora de Inicio.",
Toast.LENGTH_SHORT).show();
        }

    }
```

Se implementa la funcion del boton guardar que verifica que los campos no esten vacios y que el nombre del evento no sea repetido para poder subir la informacion llamando a los respectivos metodos de subirDispositivo(), subirLista(), y subirFormularioCurso, finalmente se vuelve a la anterior activity de Tutor.

```
    private void Guardar(){
        btn_Guardar.setOnClickListener(v -> {
            if ( Conectividad()) {

                if (disp_Lat1 != null && disp_Long1 != null && disp_Lat2
!= null && disp_Long2 != null) {
                    if (etNombreCurso.getText() != null &&
etFecha.getText() != null && etHoraInicio.getText() != null) {
                        if
(!eventos.containsValue(etNombreCurso.getText().toString())) {
                            if (box_Retraso.isChecked() &&
etTimeRetraso.getText() != null) {

subirFormularioCurso(etNombreCurso.getText().toString(), tutorID,
etDescripcion.getText().toString(), etFecha.getText().toString(),

etHoraInicio.getText().toString(), etHoraFin.getText().toString(),
etTimeRetraso.getText().toString(), box_Retraso.isChecked(),
box_CheckOut.isChecked());

                                subirDispositivo();
                                subirLista();
                                finish();
                            } else if (box_Retraso.isChecked() &&
etTimeRetraso == null) {
                                Toast.makeText(getApplicationContext(), "
Ingrese tiempo de atraso, para continuar.", Toast.LENGTH_SHORT).show();
                            } else {

subirFormularioCurso(etNombreCurso.getText().toString(), tutorID,
```

```
etDescripcion.getText().toString(), etFecha.getText().toString(),

etHoraInicio.getText().toString(), etHoraFin.getText().toString(), "0",
box_Retraso.isChecked(), box_CheckOut.isChecked());

                                subirDispositivo();
                                subirLista();
                                finish();
                            }
                        }else{
                            Toast.makeText(getApplicationContext(),
"Nombre de evento ya existe, ingrese otro nombre de evento.",
Toast.LENGTH_SHORT).show();
                        }
                    } else {
                        Toast.makeText(getApplicationContext(), "
Porfavor ingrese todos los campos obligatorios (*).",
Toast.LENGTH_SHORT).show();
                    }


                } else {
                    Toast.makeText(FormularioCurso.this, "No se pudo
marcar la zona de asistencia. Revise su conexion a internet y marcar de
nuevo la zona.", Toast.LENGTH_SHORT).show();
                }
            }else{
                Toast.makeText(FormularioCurso.this, "No dispone de
conexion a internet.", Toast.LENGTH_SHORT).show();
            }
        });

    }
```

El metodo subirFormularioCurso recibe los @parametros: nom_evento, tutor, descripcion, Fecha, horaInicio, horaFin, minRetraso, Retraso, Checkout para subir el nuevo evento o curso a la base de datos en Firebase en la sesion Evento.

```
    private void subirFormularioCurso(String nom_evento, String tutor,
String descripcion, String Fecha, String horaInicio, String horaFin,
String minRetraso, Boolean Retraso, Boolean CheckOut){
        DatabaseReference subir_data = db_reference.child("Evento");

        Map<String, String> dataCurso = new HashMap<String, String>();
        dataCurso.put("Nom_evento", nom_evento);
        dataCurso.put("Descripcion", descripcion);
        dataCurso.put("Fecha", Fecha);
        dataCurso.put("horaInicio",horaInicio);
        dataCurso.put("horaFin",horaFin);
        dataCurso.put("Retraso",Retraso.toString());
        dataCurso.put("Marcar_Salida", CheckOut.toString());
        dataCurso.put("minRetraso",minRetraso);
        dataCurso.put("tutorID",tutor);

        subir_data.push().setValue(dataCurso);
    }
```

Se suben los datos a la lista de asistencias que se encuentra en la sesion Asistencias en la base de datos de firebase, se suben los campos del nombre del evento, la fecha, y una lista vacia que cotendra los asistentes al evento.

```
    private void subirLista(){
        DatabaseReference db_lista = db_reference.child("Asistencias");
        HashMap<String, String> dataLista = new HashMap<String,
String>();
        dataLista.put("evento",etNombreCurso.getText().toString());
        dataLista.put("fecha", etFecha.getText().toString());
        dataLista.put("lista", "-");
        db_lista.push().setValue(dataLista);
    }
```

El metodo subirDispositivo permite subir las coordenadas de la zona del evento anteriormente marcada a la base de datos con el respectivo nombre del evento en l sesion Dispositivos.

```
    private void subirDispositivo(){
        DatabaseReference subir_data = db_reference.child("Dispositivo");
        Map<String, String> dataDispositivo = new HashMap<String,
String>();
        dataDispositivo.put("Evento",
etNombreCurso.getText().toString());
        dataDispositivo.put("Latitud1", disp_Lat1);
        dataDispositivo.put("Longitud1", disp_Long1);
        dataDispositivo.put("Latitud2", disp_Lat2);
        dataDispositivo.put("Longitud2", disp_Long2);
        subir_data.push().setValue(dataDispositivo);
    }
```

En la actividad de grafica de pastel se utiliza un line chart para exponer un diagrama explicativo de las asistencias obtenidas
La clase GraficaPastel implementa:

Se recorre la sesion Asistencias de la base de datos para determinar que lista de asistencia pertenece el nombre del evento y la recorre para determinar cuantos estudiantes o participantes tienen atraso, presente, o no asistieron y son agregados a la lista cantEstudiantes y se llama a createChart() para crear el grafico de pastel con los datos.

```
private void leerAsistencia() {
    db_reference.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
            for (DataSnapshot snapshot : dataSnapshot.getChildren()) {

                HashMap<String, String> data = (HashMap<String, String>)
snapshot.getValue();

                if (data!= null && data.get("evento").equals(evento)) {
                    DatabaseReference db_lista =
db_reference.child(snapshot.getKey()).child("lista");

                    db_lista.addValueEventListener(new
ValueEventListener() {
                        @Override
                        public void onDataChange(@NonNull DataSnapshot
dataSnapshot) {
```

```
                            int presente = 0;
                            int atrasado = 0;
                            int noAsiste = 0;

                            for (DataSnapshot snapshot :
dataSnapshot.getChildren()) {
                                HashMap<String, String> data =
(HashMap<String, String>) snapshot.getValue();
                                if (data!=null){
                                    if
(data.get("estado").equals("Presente")){
                                        presente+=1;
                                    }
                                    if
(data.get("estado").equals("Atrasado")){
                                        atrasado+=1;
                                    }
                                    if (data.get("estado").equals("No
asiste")){
                                        System.out.println("fallo");
                                        noAsiste+=1;
                                    }
                                }
                            }
                            totalAsistentes = presente+atrasado+noAsiste;
                            cantEstudiantes = new int[]{presente,
atrasado, noAsiste};
                            createChart();

                        }

                        @Override
                        public void onCancelled(@NonNull DatabaseError
databaseError) {
                            Log.e(TAG, "Error!",
databaseError.toException());
                        }
                    });
                    break;
                }
            }
        }

        @Override
        public void onCancelled(@NonNull DatabaseError error) {
            System.out.println(error.getMessage());
        }
    });
}
```

El metodo getSameChart() permite personalizar el contenido de la grafica, para ello toma parametros de ingreso como @parameters chart, descripcion, textColor, background y animateY y se agrega legenda de los datos.

```
private Chart getSameChart (Chart chart, String descripcion, int
```

```
textColor, int background, int animateY){
    chart.getDescription().setText(descripcion);
    chart.getDescription().setTextSize(30);
    chart.getDescription().setTextColor(textColor);
    chart.setBackgroundColor(background);
    chart.animateY(animateY);
    legend(chart);

    return chart;
}
```

El método legend requiere del @PARAMETRO de tipo Chart para configurar las entradas en la legenda del grafico de pastel.

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

legend.setHorizontalAlignment(Legend.LegendHorizontalAlignment.CENTER);
    legend.setTextSize(20);
    legend.setTextColor(Color.WHITE);

    ArrayList <LegendEntry> entries = new ArrayList<>();
    for (int i=0; i<estado.length; i++){
        LegendEntry entry =  new LegendEntry();
        entry.formColor = colors [i];
        entry.label = estado[i];
        entries.add(entry);
    }

    legend.setCustom(entries);
}
```

Se retorna una ArrayList con las entradas que se ingresaran en el grafico de pastel o PieChart, en donde se agregan objetos tipo PieEntry().

```
private ArrayList<PieEntry> getPieEntries(){
    ArrayList<PieEntry> entries = new ArrayList<>();
    for (int i=0; i<estado.length; i++){
        float promedio = cantEstudiantes[i]*(100/totalAsistentes);
        entries.add(new PieEntry(promedio, cantEstudiantes[i]));

    }
    return  entries;
}
```

createChart() permite crear una grafica de pastel y su respectiva configuracion de su forma, como el radio del circuilo interior y se inserta los datos obtendios del metodo getPieData().

```
private void createChart(){
    pieChart = (PieChart) getSameChart(pieChart, "Estado Asistencias",
Color.WHITE, Color.TRANSPARENT, 3000);
    pieChart.setHoleRadius(40);
    pieChart.setHoleColor(Color.TRANSPARENT);
```

```
        pieChart.setTransparentCircleRadius(50);
        pieChart.setDrawHoleEnabled(true);
        pieChart.setData(getPieData());
        pieChart.invalidate();

    }
```