

Aplicaciones Móviles y Servicios Telemáticos

***Manual Técnico***

Guía Técnica de la aplicación móvil SecTrack

Integrantes

-Arce Contreras Javier

-Soto Yance Joselo

-Vargas Vera Luis



Tabla de contenido

[Introducción 3](#_Toc30717410)

[Recursos 4](#_Toc30717411)

[Hardware 4](#_Toc30717412)

[Software 5](#_Toc30717413)

[Manejo de Software 7](#_Toc30717414)

[Códigos en Android Studio 12](#_Toc30717415)

[Diagramas del Proyecto 19](#_Toc30717416)

[Referencias 22](#_Toc30717417)

# Introducción

El presente manual técnico de la aplicación SecTrack se basa en explicar sobre todos los componentes que engloba el proyecto de localización de guardias para denuncias de la materia itineraria de Aplicaciones móviles y servicios telemáticos.

Este proyecto nació de la necesidad de tratar de solucionar un problema que se vive en el campus Gustavo Galindo de la Escuela Superior Politécnica del Literal, el cual se basa en que no existe una manera de denunciar los delitos que se pueden llegar a suscitar dentro de las instalaciones de la institución de educación superior, además de que los encargados del servicio de seguridad no se encuentran a todas horas en sus estaciones asignadas, por lo cual se desarrolla esta aplicación móvil con interacción con un dispositivo capaz de censar la ubicación de su poseedor.

Este manual técnico describe la utilización de dispositivos de Internet de las Cosas que trabajan con la tecnología de Sigfox, además de la manera de conectar estos terminales con software, es decir, de poder reflejar los datos que se obtienen en Firebase que es la base de datos utilizada, además de poder visualizarlas de manera sencilla en un dispositivo móvil con sistema operativo Android y así los usuarios puedan hacer las denuncias respectivas y los guardias puedan atenderlas.

# Recursos

Para la realización de esta posible solución, se utilizó tanto recursos de hardware como de software, los cuales son:

|  |  |
| --- | --- |
| Hardware | Software |
| SUNTECH | IDE ANDROID STUDIO |
|  | FIREBASE |
|  | SIGFOX |

Tabla 1 Tabla de recursos

# Hardware

**Suntech**

A partir de la utilización del Suntech, el cual es un dispositivo IoT que tiene como objetivo censar diferentes magnitudes como temperatura, longitud, latitud y mide el nivel de su batería. Por lo cual, en este proyecto se usa para poder localizar a los guardias que se encuentran trabajando en las diferentes partes del Campus Gustavo Galindo de la ESPOL.



Ilustración 1 Suntech

# Software

**IDE Android Studio**

Para el desarrollo de la aplicación móvil, se utiliza el IDE de Android Studio con el lenguaje de programación Java, el cual a partir de sus librerías y opciones disponibles permite la conexión con la base de datos en este caso Firebase, de manera que se puede subir y obtener datos a tiempo real.



Ilustración 2 IDE Android Studio

**Firebase**

Firebase es una base de datos no relacional, la cual se centra en el trabajo en tiempo real con aplicaciones web y móviles, por lo cual se utiliza en este proyecto, debido a que además de mantener los datos con los que se trabaja, también cuanta con diferentes opciones para el registro y autenticación de los usuarios.



Ilustración 3 Firebase

**Sigfox**

La tecnología Sigfox la cual utiliza LPWAN tiene el objetivo principal de comunicar los dispositivos IoT con la opción de enviar datos de hasta 12 bytes y 140 mensajes diarios, además que esta tecnología facilita la comunicación del dispositivo con la base de datos a partir del Backend en el cual se puede administrar los mensajes que se reciben del dispositivo IoT y con la opción de los Callbacks conectarla directamente a la base de datos, de manera que se facilita la obtención de estos datos para la aplicación móvil.



Ilustración 4 Tecnología Sigfox

# Manejo de Software

**Sigfox**

A partir de la utilización del Backend de Sigfox se puede visualizar los datos que censa el dispositivo IoT:

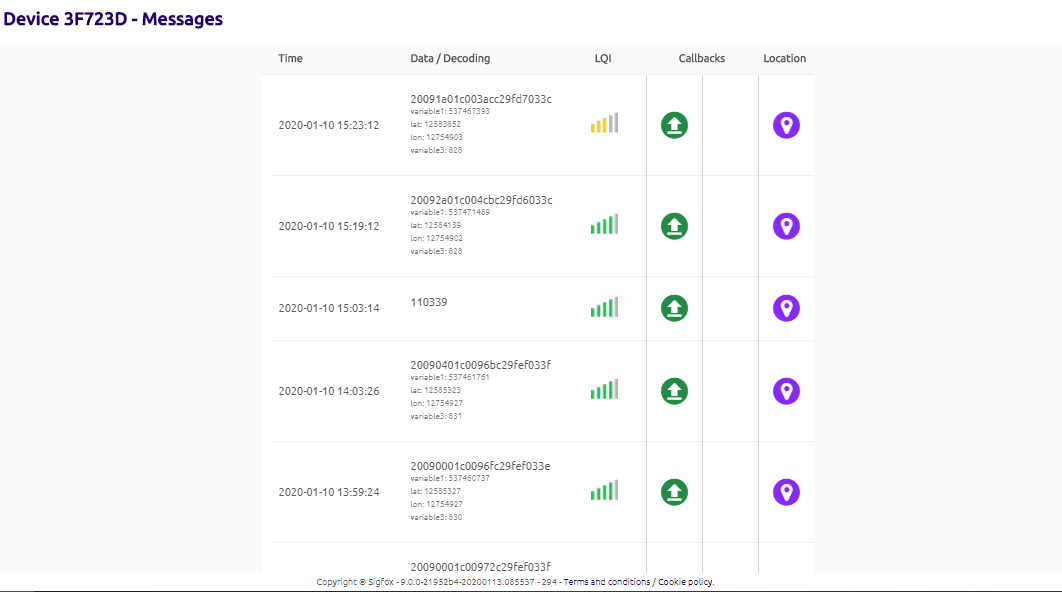


Ilustración 5 Visualización de los mensajes enviados por el dispositivo

Para poder enviar los datos que son censados por el dispositivo IoT a la Base de datos que en este caso es Firebase, es necesario utilizar la opción de Callbacks que permite Sigfox, en donde se configura que dipo de dato, por cual canal y en este caso que es URL, luego se segmentan los bytes en variables con el tipo de datos que se requiere, luego en Url pattern se ingresa el JSON de la Firebase del proyecto, teniendo en cuenta que en el Content type se especifique que es aplication/json y al final se ingresa el código en el cual se asignan las variables con su dato.

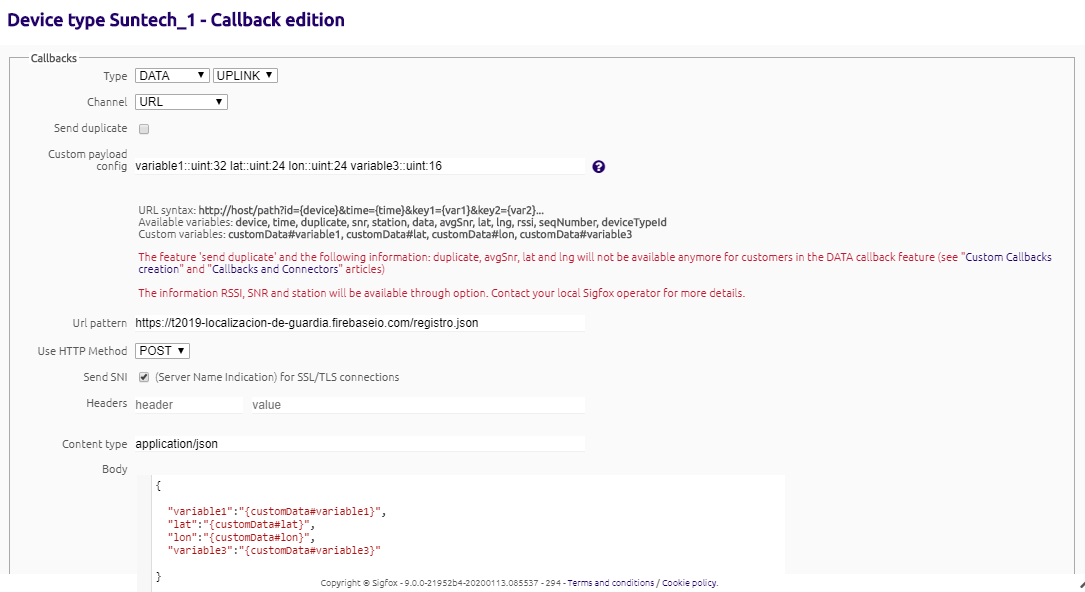


Ilustración 6 Callback del dispositivo IoT

Y de esta manera se presenta en la Base de datos:

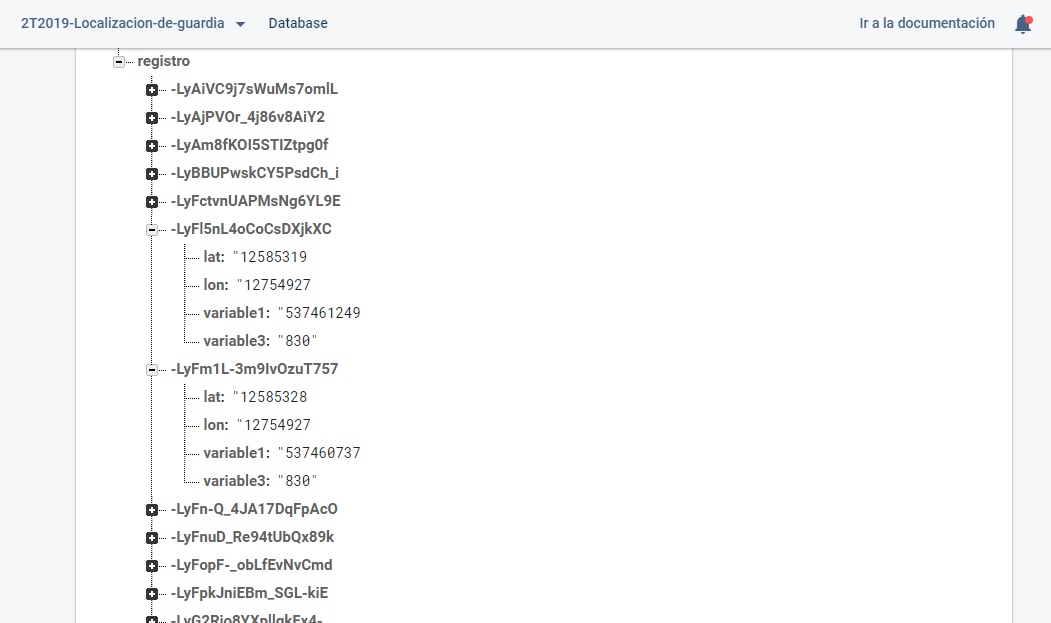


Ilustración 7 Registro de los datos censados por el dispositivo IoT en la Firebase

**Firebase**

En esta base no relacional, se encuentran los diferentes nodos en los cuales se encuentran toda la información que se utiliza en la aplicación móvil, como información de los administradores, guardias, usuario, los registros censados del dispositivo IoT, además de las denuncias realizadas por los usuarios, la ubicación en tiempo real de los usuarios y la localización de todas las estaciones de guardias de la universidad.

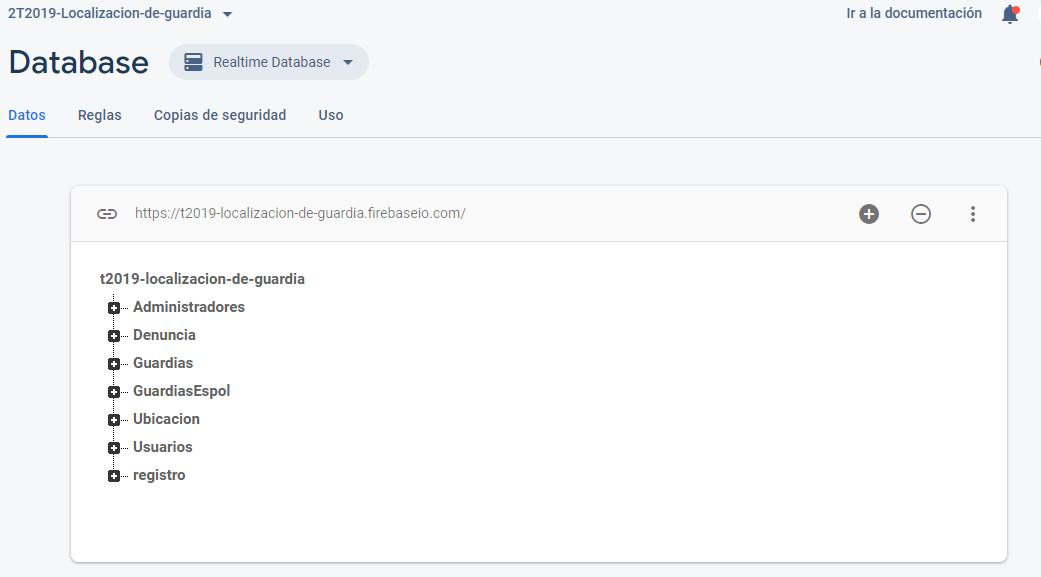


Ilustración 8 Base de datos en Firebase

Además del servicio de almacenar los datos de manera ordenada, la Firebase también tiene la opción de Autenticación con la cual los usuarios pueden iniciar sesión directamente desde su cuenta de Google y también se almacenan los correos de usuarios con cuentas registrados de otra compañía de correo como se muestra en la ilustración 9.

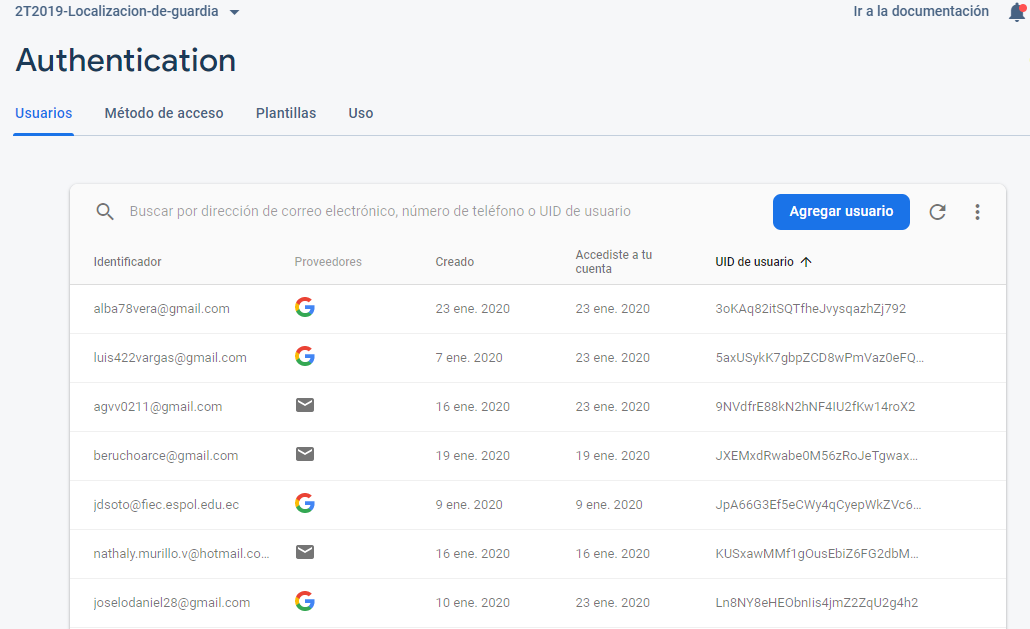


Ilustración 9 Autenticación de las cuentas

# Códigos en Android Studio

|  |  |
| --- | --- |
| **Nombre de archivo:** LoginActivity  **Nombre de función:** login(String email, String password)  **Descripción:** Permite iniciar sesión a un usuario ya registrado dentro de la base de datos de Firebase. El método a traves de signInWithEmailAndPassword verifica la existencia del usuario dentro de la base de datos. | |
|  | private void login(String email, String password){  progressDialog.show();  mAuth.signInWithEmailAndPassword(email,password).addOnCompleteListener(this, task -> {  if(task.isSuccessful()){  progressDialog.dismiss();  FirebaseUser user = mAuth.getCurrentUser();  Intent intent = new Intent(LoginActivity.this, DashboardActivity.class);  startActivity(intent);  intent.addFlags(Intent.*FLAG\_ACTIVITY\_CLEAR\_TOP* | Intent.*FLAG\_ACTIVITY\_NEW\_TASK* |Intent.*FLAG\_ACTIVITY\_CLEAR\_TASK*);  finish();  }else{  progressDialog.dismiss();  Toast.*makeText*(LoginActivity.this, "Por favor revise que sus credenciales sean correctas.",  Toast.*LENGTH\_SHORT*).show();  }  }).addOnFailureListener(e -> Toast.*makeText*(LoginActivity.this, ""+e.getMessage(),  Toast.*LENGTH\_SHORT*).show()); } |

|  |  |
| --- | --- |
| **Nombre de archivo:** RegistroActivity  **Nombre de función:** registrar(String email,String password, String nombre, String apellido, String telefono)  **Descripción:** Permite registrar al usuario mediante parámetro solicitados en la pantalla de Registro. El método guarda en Firebase los datos principales requeridos. | |
|  | private void registrar(String email,String password, String nombre, String apellido, String telefono){  progressDialog.show();  mAuth.createUserWithEmailAndPassword(email, password)  .addOnCompleteListener(this, task -> {  if (task.isSuccessful()) {  progressDialog.dismiss();   FirebaseUser user = mAuth.getCurrentUser();  String userEmail = user.getEmail();  String uid = user.getUid();  String tipoCuenta = getIntent().getStringExtra("tipo de cuenta");   HashMap<Object,String> hashMap = new HashMap<>();  hashMap.put("Email",userEmail);  hashMap.put("UID",uid);  hashMap.put("Nombre",nombre);  hashMap.put("Apellidos",apellido);  hashMap.put("Telefono",telefono);  hashMap.put("Imagen","");  hashMap.put("Tipo de cuenta", tipoCuenta);    //Instancia de Firebase  FirebaseDatabase database = FirebaseDatabase.*getInstance*();  DatabaseReference reference = database.getReference("Usuarios");  reference.child(uid).setValue(hashMap);  Toast.*makeText*(RegistroActivity.this, "Registro exitoso",  Toast.*LENGTH\_SHORT*).show();  startActivity(new Intent(RegistroActivity.this, DashboardActivity.class));  finish();  } else {  progressDialog.dismiss();  Toast.*makeText*(RegistroActivity.this, "Autenticacion falló",  Toast.*LENGTH\_SHORT*).show();  }  }).addOnFailureListener(e -> {  progressDialog.dismiss();  Toast.*makeText*(RegistroActivity.this,""+e.getMessage(),Toast.*LENGTH\_SHORT*).show();  }); } |

|  |  |
| --- | --- |
| **Nombre de archivo:** DashboardActivity  **Nombre de función:** subirLaLongFirebase()  **Descripción:** El objetivo del método es enviar a la base de datos de Firebase la ubicación propia del celular mediante el FusedLocationClient. Además, se solicita el permiso para la app acceda a la ubicación del propio dispositivo***.*** | |
|  | private void subirLatLngFirebase() {  if(ActivityCompat.*checkSelfPermission*(this, Manifest.permission.*ACCESS\_FINE\_LOCATION*) != PackageManager.*PERMISSION\_GRANTED* && ActivityCompat.*checkSelfPermission*(this,Manifest.permission.*ACCESS\_COARSE\_LOCATION*) != PackageManager.*PERMISSION\_GRANTED*){  ActivityCompat.*requestPermissions*(DashboardActivity.this, new String[]{Manifest.permission.*ACCESS\_FINE\_LOCATION*},MY\_PERMISSIONS\_REQUEST\_READ\_CONTACTS);  return;  }  fusedLocationClient.getLastLocation()  .addOnSuccessListener(this, location -> {  // Got last known location. In some rare situations this can be null.  if (location != null) {  // Logic to handle location object  Log.*e*("Latitud: ",+location.getLatitude()+"Longitud: "+location.getLongitude());  *lat*=location.getLatitude();  *lon*=location.getLongitude();  HashMap<String,Object> latLng = new HashMap<>();  latLng.put("latitud",location.getLatitude());  latLng.put("longitud",location.getLongitude());  databaseReference.child("Ubicacion").push().setValue(latLng);  Location guardiaCentral = new Location(" ");  guardiaCentral.setLatitude(-2.148348477036179);  guardiaCentral.setLongitude(-79.96371562917663);  Log.*e*("Distancia: ", String.*valueOf*(location.distanceTo(guardiaCentral)));  if(location.distanceTo(guardiaCentral) > 2000){  Toast.*makeText*(this,"No hay guardias cercanos",Toast.*LENGTH\_SHORT*).show();  }  }else{  Toast.*makeText*(this,"Por favor, asegúrese de tener el GPS activado",Toast.*LENGTH\_SHORT*).show();  }  }); } |

|  |  |
| --- | --- |
| **Nombre de archivo:** DashboardActivity  **Nombre de función:** checkConnection()  **Descripción:** Método para verificar el tipo de conexión en la que se encuentra el dispositivo móvil. Además, se muestra el mensaje que indica si no se posee acceso a internet. | |
|  | public void checkConnection(){  ConnectivityManager connectivityManager = (ConnectivityManager)getApplicationContext()  .getSystemService(Context.*CONNECTIVITY\_SERVICE*);  NetworkInfo activeNetwork = connectivityManager.getActiveNetworkInfo();  if(null != activeNetwork){  if(activeNetwork.getType() == ConnectivityManager.*TYPE\_WIFI*){  //Toast.makeText(this, "Wifi: Encendido",Toast.LENGTH\_SHORT).show();  }  if(activeNetwork.getType() == ConnectivityManager.*TYPE\_MOBILE*){  Toast.*makeText*(this, "Datos moviles: Encendido",  Toast.*LENGTH\_SHORT*).show();  }  }else{  Toast.*makeText*(this, "No hay conexion a Internet",  Toast.*LENGTH\_SHORT*).show();  } } |

|  |  |
| --- | --- |
| **Nombre de archivo:** DashboardActivity  **Nombre de función:** checkUserStatus()  **Descripción:** El objetivo del método es verificar el estado en el que se encuentra el usuario una vez ingresado o cerrado sesión mediante Firebase; es decir, si ha iniciado sesión con éxito por lo tanto el usuario puede usar la app, caso contrario se lo regresa a la actividad de inicio de sesión. | |
|  | private void checkUserStatus(){  FirebaseUser firebaseUser = firebaseAuth.getCurrentUser();  if(firebaseUser != null){  //Usuario que ha iniciado sesion se mantiene aqui  Toast.*makeText*(this,"Estoy en el Dashboard",Toast.*LENGTH\_SHORT*).show();  }else{  //Usuario se lo regresa a que inicie sesion  startActivity(new Intent(DashboardActivity.this, LoginActivity.class));  finish();  } } |
| **Nombre de archivo:** Reportar\_Denuncia  **Nombre de función:** Denunciar(View view)  **Descripción:** Este método sirve para que el usuario llene el formulario de la denuncia, la información del delito y del denunciante sean subidas a la base de datos. | |
|  | public void Denunciar(View view){  if(!checkConnection()){  Toast.*makeText*(this, "No hay conexion a Internet",  Toast.*LENGTH\_SHORT*).show();  }else{  String name= denunciante\_name.getText().toString();  String descripcion\_String= descrip\_txt.getText().toString();  String area=spinner1.getSelectedItem().toString();  latitud=DashboardActivity.*lat*;  longitud=DashboardActivity.*lon*;  Date date=new Date();  int month= date.getMonth()+1;  int day= date.getDay()+19;  int year=1900+date.getYear();  int hour=date.getHours()-5;  int min=date.getMinutes();  String date\_String= String.*valueOf*(day)+"/"+String.*valueOf*(month)+"/"+String.*valueOf*(year)+" Hora: "+ String.*valueOf*(hour)+":"+String.*valueOf*(min);   if(!TextUtils.*isEmpty*(descripcion\_String)){  String id=databaseDenuncia.push().getKey();  Denuncia denuncia=new Denuncia(id,HomeFragment.*tipo\_delito*,name,date\_String,latitud,longitud,area,descripcion\_String,"NO");   databaseDenuncia.child(id).setValue(denuncia);  Toast.*makeText*(this,"Denuncia realizada con exito",Toast.*LENGTH\_LONG*).show();  Intent intent= new Intent(this, DashboardActivity.class);  startActivity(intent);  finish();  }else{  Toast.*makeText*(this, "Debes describir el crimen",Toast.*LENGTH\_LONG*).show();  }  } } |

# Diagramas del Proyecto

**Diagrama del Proyecto**

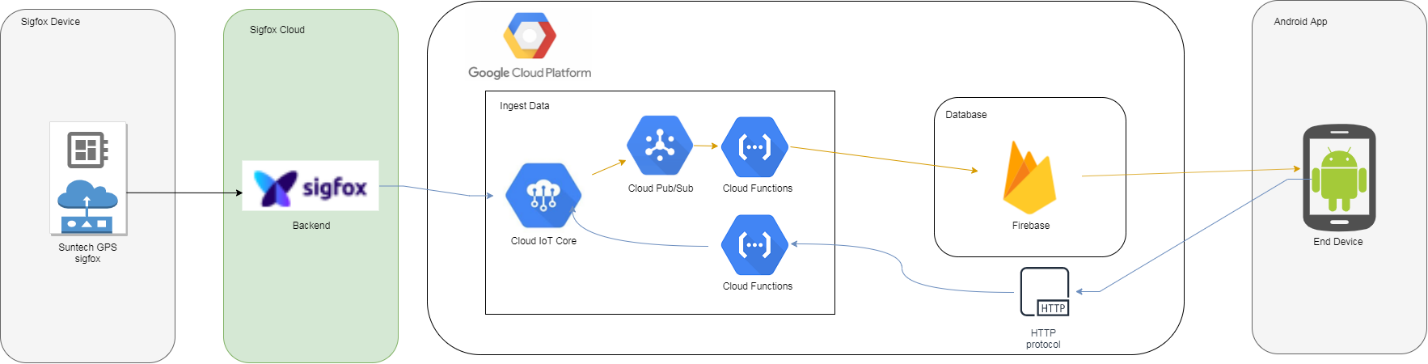


Ilustración 10 Diagrama del Proyecto

**Mapa de Despliegue**

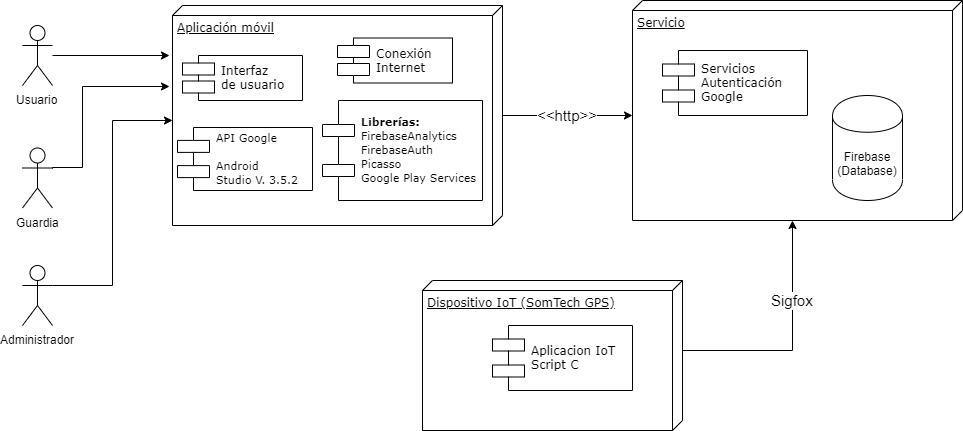


Ilustración 11 Mapa de despliegue

**Diagrama de Casos de Uso**

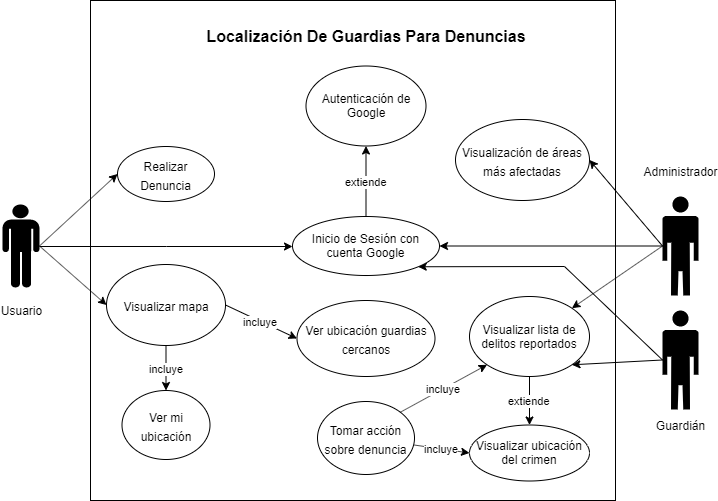


Ilustración 12 Diagrama de Casos de Uso

**Diagrama de Entidad – Relación**

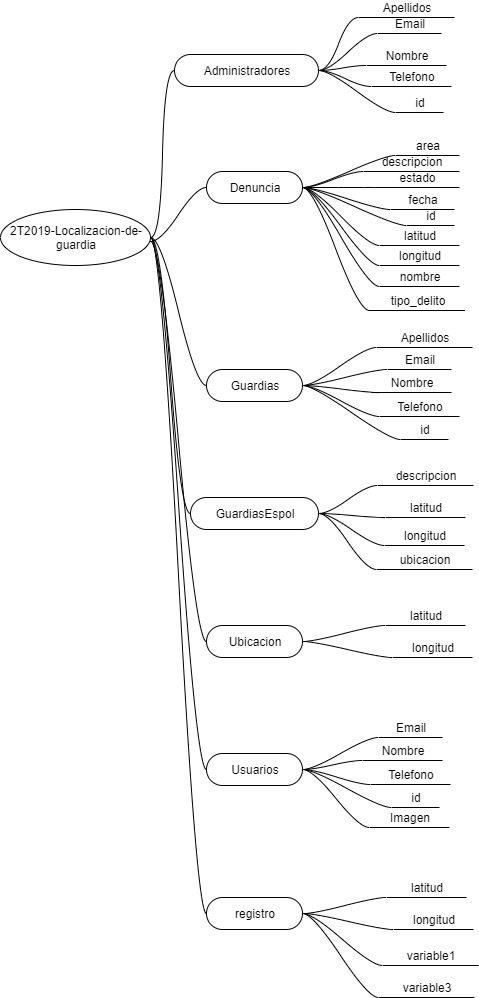


Ilustración 13 Diagrama de Entidad – Relación

# Referencias

|  |  |
| --- | --- |
| [1] | Android, «developers,» 2020. [En línea]. [Último acceso: Enero 2019]. |
| [2] | Suntech, «Ubidots,» 2018. [En línea]. Available: https://res.cloudinary.com/di2vaxvhl/image/upload/v1553304389/ST730\_-\_User\_manual\_V20\_-\_08012018.pdf. [Último acceso: Enero 2020]. |
| [3] | Sigfox, «sigfox support,» 2020. [En línea]. Available: https://support.sigfox.com/docs/api-documentation. [Último acceso: Enero 2020]. |
| [4] | Firebase, «Firebase,» 2020. [En línea]. Available: https://firebase.google.com/docs. [Último acceso: Enero 2020]. |