

Manual Técnico

Manual del sistema
integrado “Derbus”



Aplicaciones Móviles Y Servicios Telemáticos

Integrantes:

Herman Pineda, Jhostin Ocampo, Isabella Baquerizo, Kenneth Veintimilla

Finalmente, Derbus es una aplicación diseñada para los estudiantes de la ESPOL, que busca resolver la problemática de los irregulares horarios de parada, de modo que a través de un dispositivo diseñado comunicara por vía GSM y Sigfox el horario de llegada de cada bus, en cada parada designada previamente.

Hardware

La comunicación entre el PC y Arduino se produce a través del Puerto Serie. Posee un convertidor usb-serie, por lo que sólo se necesita conectar el dispositivo a la computadora utilizando un cable USB.

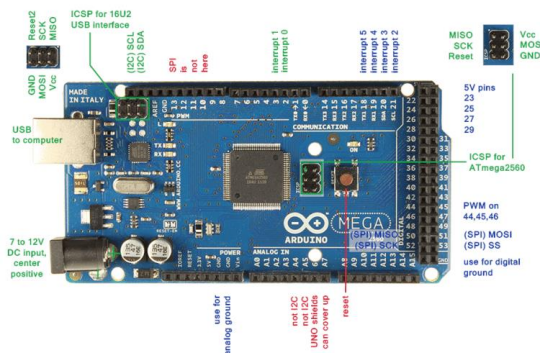


Ilustración 1 Arduino Mega

Thinxtra Sigfox

Sigfox es una tecnología complementaria de bajo costo y poco consumo de energía, considerada como parte de una red LPWAN. Sigfox es compatible con Bluetooth, GPS 2G / 3G / 4G y Wifi. Al combinar otras soluciones de conectividad con Sigfox.

La solución de conectividad única proporciona el dispositivo a la nube simplificar las comunicaciones, permitimos un inmejorable bajo consumo de energía pudiendo evitar la necesidad de reemplazar o recargar las baterías.



Ilustración 2 Modulo Thinxtra Sigfox

Modulo GSM

Un módulo GSM está integrado con una tarjeta SIM, de modo que podemos comunicarnos con él como si se tratase de un teléfono móvil. Este dispositivo nos permitirá enviar y recibir SMS y conectarnos a Internet, de modo que los datos recogidos por el Arduino serán enviados vía SMS.



Ilustración 3 Modulo GSM

Modulo GPS

El módulo GPS posee un un módulo de serie U-Blox NEO 6M equipado en el PCB, una EEPROM con configuración de fábrica, una pila de botón para mantener los datos de configuración en la memoria EEPROM, un indicador LED y una antena cerámica. También posee los pines o conectores Vcc, Rx, Tx y Gnd por el que se puede conectar a algún microcontrolador mediante una interfaz serial.



Ilustración 4 Modulo GPS6MV2

Software

IDE: Android Developer Tools Plugin IDE

Requisitos Sistema Operativo (Android Studio)

	Windows
OS version	Windows 10/8/7 (64-bit) 8Gb Ram (Recommended)

Manejador de Base de Datos: Firebase

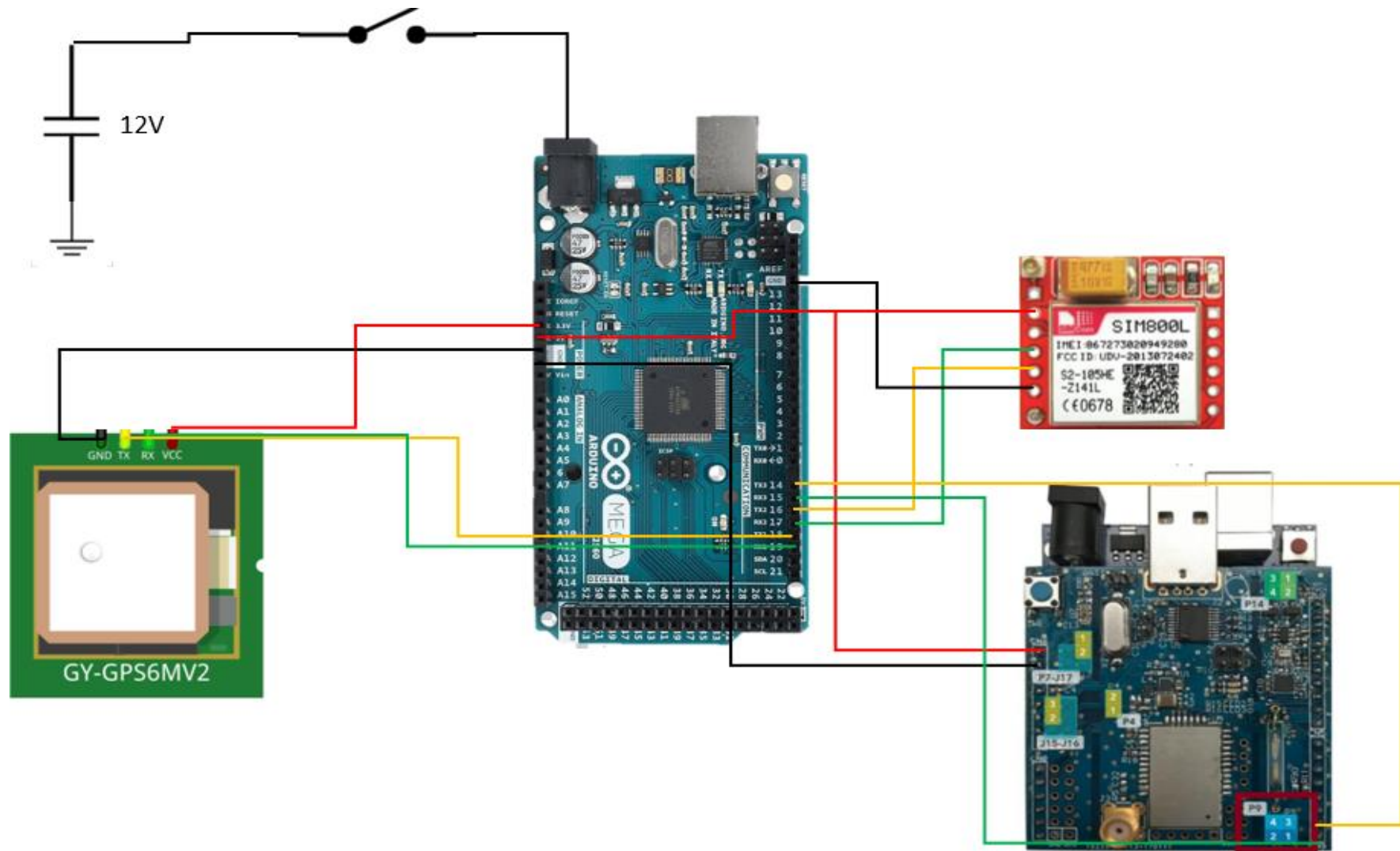
Lenguaje de Programación: XML, JavaScript.

Teléfono Celular: Sistema operativo Android 4.1 JellyBean en adelante.

Sigfox platform

En los proyectos que involucran el IoT, la conectividad al internet es la parte principal y actualmente Sigfox es el medio por el que se envía cualquier tipo de dato e información. Esta plataforma proporciona conectividad inalámbrica y fue creada para que funcione e interactúe con dispositivos de bajo consumo energético - tales como sensores que funcionan con pilas convencionales - con tasas de transferencias de datos de hasta 12 bytes y un máximo número de mensajes o peticiones de 140 diarias. [1]

Esquemático de conexión Hardware



Software

Módulo de Comunicaciones Arduino- Sigfox(THINXTRA).

```
#include <SoftwareSerial.h>

#include <TinyGPS.h>

// Pines para los LEDs

#define LEDVERDE 2

#define LEDAMARILLO 3

#define LEDROJO 4

#define ANALOGPILA 0

#define BOTON 2

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 disponibilidad del puerto Serial del GPS.
    while(serialgps.available()){

        //Lee el estado del boton
        int estado = digitalRead(BOTON);

        //Leyendo la informaci3n que viene del GPS.
        int c=serialgps.read();

        //Va agrupando toda la informaci3n 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 anal3gica
                analogValor = analogRead(ANALOGPIILA);

                // 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);
            }
        }
    }
}

```

```

    }

    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);
}

}

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');
}

```


Screenshoots de la Aplicación- Usuario

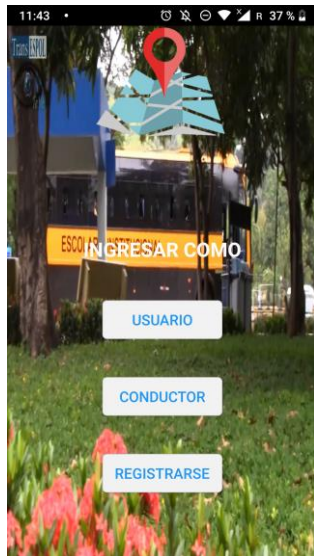


Ilustración 5 Pantalla de INICIO



Ilustración 6 Pantalla de Login del Usuario



Ilustración 7 Menú del Usuario



Ilustración 8 Visualización Sigfox

Screenshoots de la Aplicación -Usuario



Ilustración 9
Visualización GSM



Ilustración 10 Opción
Seleccionar Paradas

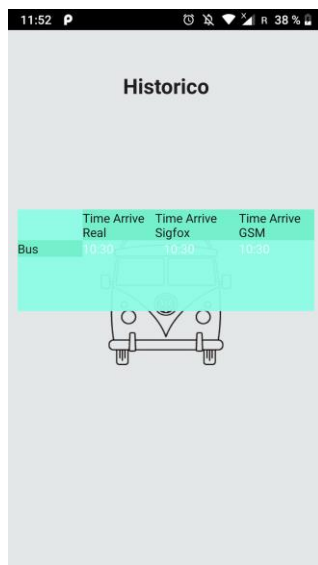


Ilustración 11 Opción
Histórico

Screenshoots de la Aplicación- Conductor

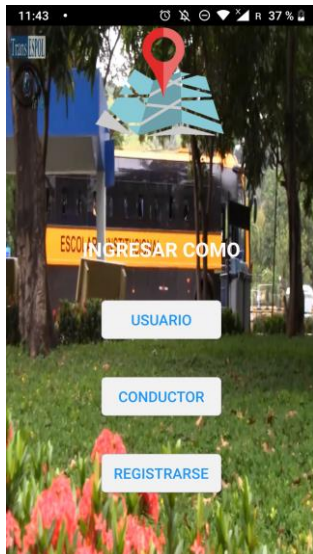


Ilustración 12 Pantalla de INICIO

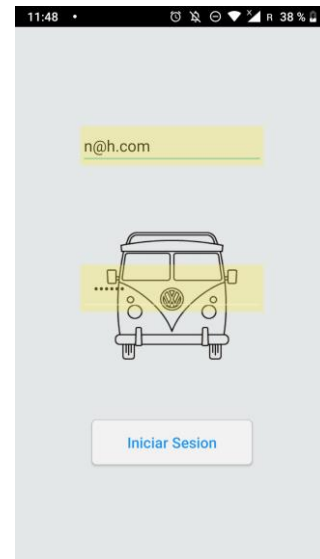


Ilustración 13 Pantalla de Login del Conductor



Ilustración 14 Menú de opciones del Conductor

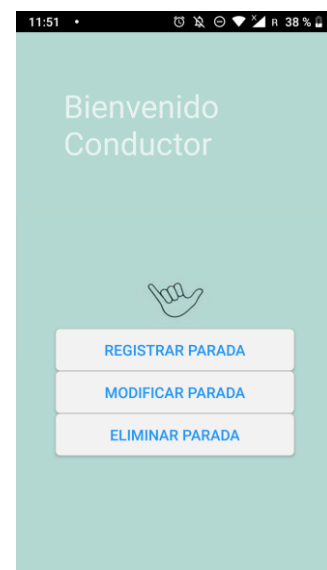


Ilustración 15 Opción de Paradas

Screenshoots de la Aplicación- Conductor



Ilustración 16 Opción Paradas- Agregar Parada

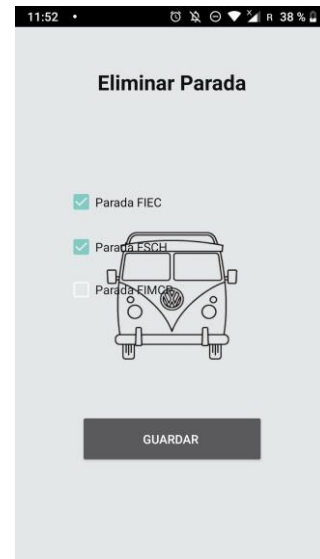


Ilustración 17 Opción Paradas- Eliminar Parada



Ilustración 17 Visualización Sigfox

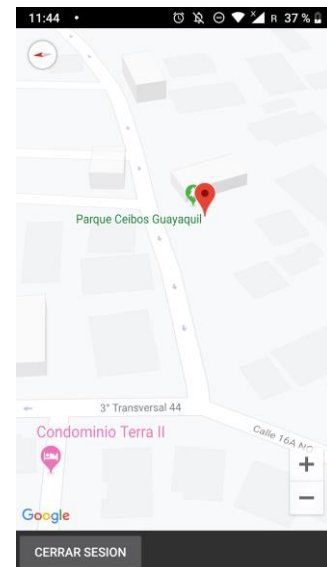


Ilustración 19 Visualización GSM

Screenshoots de la Aplicación

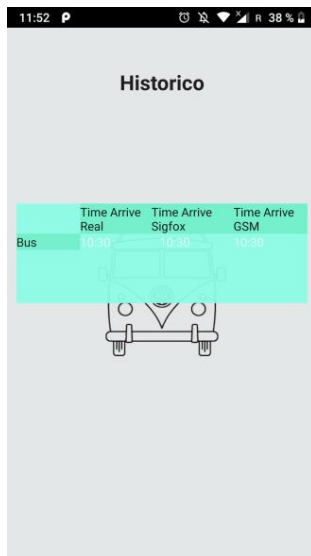


Ilustración 20 Pantalla de Histórico.

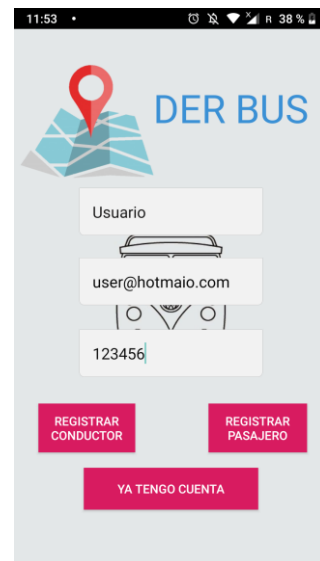


Ilustración 21 Pantalla de Registro

Codigos en Android Studio

Nombre de archivo: Inicio.class

Nombre de función: Videothing()

Descripción: Esta función permite presentar un video en Loop en la pantalla de Inicio



```
private void Videothing() {  
    videoBG = (VideoView) findViewById(R.id.videoView);  
    Uri uri = Uri.parse("android.resource://" // First start  
with this,  
                        + getPackageName() // then retrieve your package  
name,  
                        + "/" // add a slash,  
                        + R.raw.fire);  
    videoBG.setVideoURI(uri);  
    // Start the VideoView  
    videoBG.start();  
    videoBG.setOnPreparedListener(new  
MediaPlayer.OnPreparedListener() {  
        @Override  
        public void onPrepared(MediaPlayer mediaPlayer) {  
            mMediaPlayer = mediaPlayer;  
            // We want our video to play over and over so we  
set looping to true.  
            mMediaPlayer.setLooping(true);  
            // We then seek to the current position if it  
has been set and play the video.  
            if (mCurrentVideoPosition != 0) {  
                mMediaPlayer.seekTo(mCurrentVideoPosition);  
                mMediaPlayer.start();  
            }  
        }  
    });  
}
```


Nombre de archivo: MessageReceiver.class [2]

Descripción: Genera el punto de entrada cuando se recibieron nuevos sms.



```
public class MessageReceiver extends BroadcastReceiver {

    private static MessageListener mListener;

    @Override
    public void onReceive(Context context, Intent intent) {
        Bundle data = intent.getExtras();
        Object[] pdus = (Object[]) data.get("pdus");
        for(int i=0; i<pdus.length; i++){
            SmsMessage smsMessage =
            SmsMessage.createFromPdu((byte[]) pdus[i]);
            String message = "Sender : " +
            smsMessage.getDisplayOriginatingAddress()
                + "Email From: " + smsMessage.getEmailFrom()
                + "Email Body: " + smsMessage.getEmailBody()
                + "Display message body: " +
            smsMessage.getDisplayMessageBody()
                + "Time in millisecond: " +
            smsMessage.getTimestampMillis()
                + "Message: " + smsMessage.getMessageBody();
            mListener.messageReceived(message);
        }
    }

    public static void bindListener(MessageListener listener){
        mListener = listener;
    }
}
```

Nombre de archivo: MainActivity.class

Nombre de función: messageReceived(String message)

Descripción: Implementa el listener donde desea recibir su cuerpo sms y lo registrar. A medida que obtenemos datos sms de vuelta a mi clase.




```
@Override
    public void messageReceived(String message) {
        Toast.makeText(this, "New Message Received: " + message,
            Toast.LENGTH_SHORT).show();
    }
}
```

Nombre de archivo: MapaUsuario.class

Nombre de función: onMapReady(GoogleMap googleMap)

Descripción: Esta función permite generar un menú de Google Map en nuestra APP, previamente se debe haber obtenido una Api key.

	<pre> public void onMapReady(GoogleMap googleMap) { mMap = googleMap; mMap.setMapType(GoogleMap.MAP_TYPE_SATELLITE); UiSettings uiSettings= mMap.getUiSettings(); uiSettings.setZoomControlsEnabled(true); // Add a marker in Sydney and move the camera LatLng sydney = new LatLng(-2.1961601, -79.8862076); mMap.addMarker(new MarkerOptions().position(sydney).title("Marker")); mMap.moveCamera(CameraUpdateFactory.newLatLng(sydney)); } } </pre>
---	---

Nombre de archivo: Inicio_sesion.class	
Nombre de función: loginUser()	
Descripción: Esta función permite a un usuario ya registrado, iniciar sesión y validar si sus credenciales son correctas.	
	<pre> private void loginUser() { mAuth.signInWithEmailAndPassword(email, contraseña).addOnCompleteListener(new OnCompleteListener<AuthResult>() { @Override public void onComplete(@NonNull Task<AuthResult> task) { if (task.isSuccessful()) { startActivity(new Intent(Inicio_sesion.this, MapaUsuario.class)); finish(); } else{ Toast.makeText(Inicio_sesion.this, "No se puede iniciar sesion, compruebe los datos", Toast.LENGTH_SHORT).show(); } } }); } } </pre>

Nombre de archivo: MainActivity.class

Nombre de función: registerUser()

Descripción: Esta función permite registrar un Usuario así como validar si el usuario fue correctamente registrado.



```
private void registerUser() {

 mAuth.createUserWithEmailAndPassword(mail,password).addOnCompleteListener(new OnCompleteListener<AuthResult>() {

    @Override

    public void onComplete(@NonNull Task<AuthResult> task) {

        if (task.isSuccessful()){

            Map<String,Object> map=new HashMap<>();

            map.put("name",nombre);

            map.put("email",mail);

            map.put("password",password);

            String id= mAuth.getCurrentUser().getUid();
mDatabase.child("Conductor").child(id).setValue(map).addOnCompleteListener(new OnCompleteListener<Void>() {

                @Override

                public void onComplete(@NonNull Task<Void> task2) {

                    if (task2.isSuccessful()){

                        startActivity(new Intent(MainActivity.this,Inicio_sesion.class));

                        finish()

                    }

                    else{

Toast.makeText(MainActivity.this,"No se pudo crear los datos correctamente", Toast.LENGTH_SHORT).show(); }}

                });}

                else{

                    Toast.makeText(MainActivity.this,"No se pudo registrar este Conductor", Toast.LENGTH_SHORT).show();

                }

            }

        }

    }

});}
```

Diagrama casos de uso

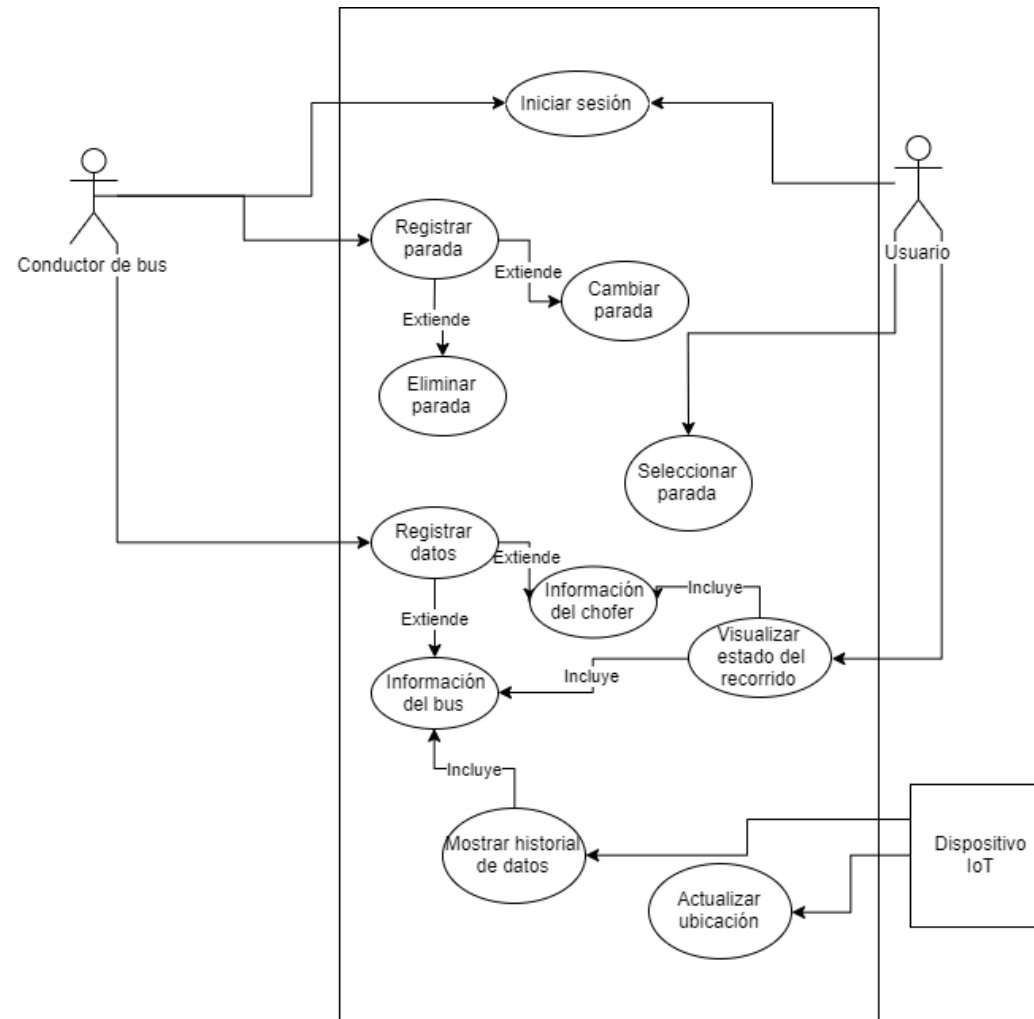


Diagrama Entidad-Relación

JSON Editor Online

New Open Save Settings Help

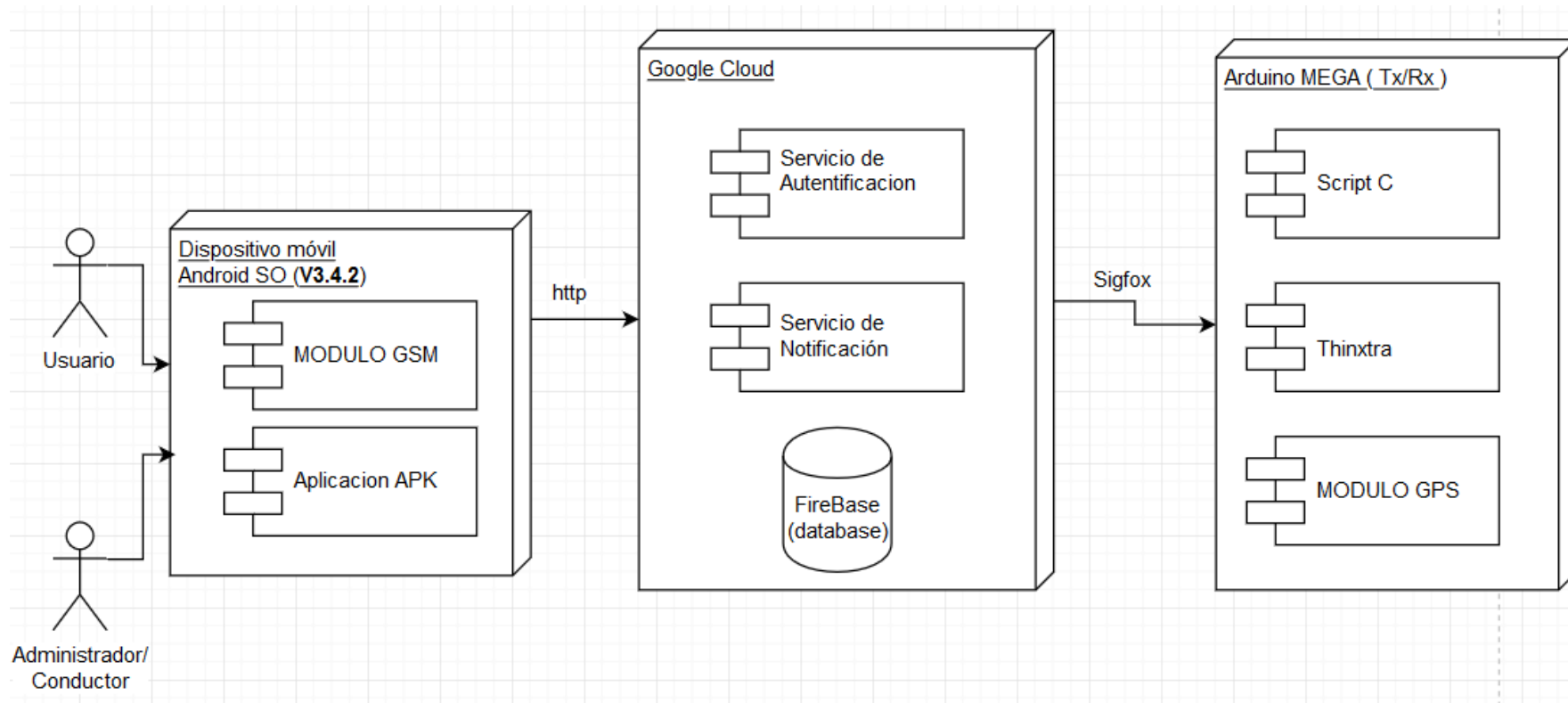
```
1 {
2   "datos":{
3     "GPS":{
4       "matricula":""
5     },
6   "bus":{
7     "matricula":""
8   },
9   "Pasajero":{
10    "nombre": "",
11    "ID": "",
12    "nombre_usuario": "",
13    "contraseña": ""
14  },
15  "celular":{
16    "user_name": "",
17    "coordenadas": ""
18  },
19  "Conductor":{
20    "ID": "",
21    "name": "",
22    "user_name": "",
23    "contraseña": "",
24    "email": ""
25  },
26  "historico":{
27    "ID": "",
28    "coordenada": "",
29    "fecha": "",
30    "hora": ""
31  }
32 }
33 }
34 }
35 }
36 }
37 }
38 }
39 }
40 }
41 }
42 }
```

Ln: 31 Col: 1

object ▶ datos ▶ historico ▶

- object {1}
- datos {6}
- GPS {1}
- matricula :value
- bus {1}
- matricula :value
- Pasajero {4}
- nombre :value
- ID :value
- nombre_usuario :value
- contraseña :value
- celular {2}
- user_name :value
- coordenadas :value
- Conductor {5}
- ID :value
- name :value
- user_name :value
- contraseña :value
- email :value
- historico {4}
- ID :value
- coordenada :value
- fecha :value
- hora :value

Diagrama de Despliegue



Limitaciones

El uso de la plataforma Sigfox para el envío de datos e información mediante Internet es limitado ya que nos permite un máximo de 140 mensajes diarios lo que se verá afectado a la cantidad de aplicaciones que se pueden usar mediante esta plataforma en nuestros aplicativos móviles.

Uso de lenguaje de programación orientado a objetos JAVA para la realización del programa y de ciertas librerías que permitan realizar las presentaciones de gráficos y multimedia.

Bibliografía

- [1] Sigfox, «aprendiendo arduino,» [en línea]. available:
<https://aprendiendoarduino.wordpress.com/tag/backend-sigfox/>. [Último acceso: 24 1 2020].

- [2] M. Yadav, «medium,» 9 7 2017. [En línea]. Available:
<https://medium.com/@muku.hbti/read-incoming-message-in-android-35db3c08709>.
[Último acceso: 24 1 2020].