

Guía | Firebase & App Movil

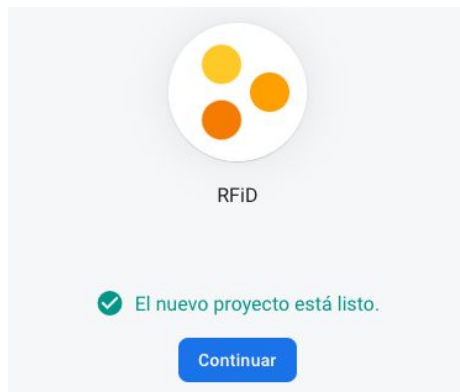
En este documento se detallan los pasos necesarios para implementar la base de datos para el control remoto del sistema RFID (desarrollado por Bernardo Urriza, Antonio Rivera y Gerardo Arizmendi) que utiliza una Raspberry Pi y módulo M6E Nano.

Sección 1

Creación de cuenta y proyecto dentro de firebase.

Desde una PC con acceso a internet:

1. Abrir un navegador web e ingresar a la liga de firebase: <https://firebase.google.com> iniciar sesión con su cuenta ITESM. Después deberás de ingresar a la consola para crear un nuevo proyecto.
2. Completar algunos pasos para crear un proyecto: ponerle nombre al proyecto, aceptar los términos y desactivar *Google Analytics*. Al completar estos pasos, apretar el botón de “Crear Proyecto”.



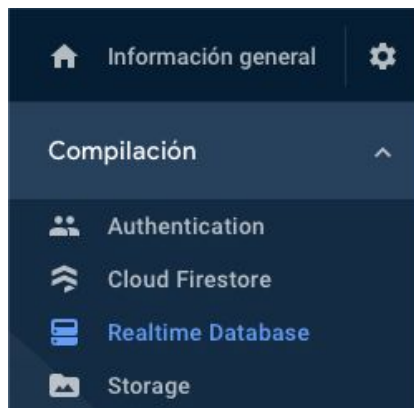
3. Al presionar “Continuar” la consola del nuevo proyecto aparecerá. La base de datos está lista para ser usada, pero debe de ser configurada para que pueda ser accedida desde un dispositivo remoto, para lograr esto se debe de generar un SDK (Software Development Kit) y agregarlo al dispositivo móvil. El SDK tendrá las credenciales necesarias para poder hacer uso de todos los servicios de la base de datos.

Sección 2

Preparar la base de datos para ser usada.

Dentro de la consola del proyecto creado

1. Ingresar a la base de datos “Realtime Database”, esto se encuentra en el panel izquierdo de la consola al presionar la opción de compilaciones.



2. Crear base de datos al presionar “Crear base de datos”.

3. Configurar la base de datos, agregando la ubicación de Estados Unidos (us-central1) y seleccionar el modo prueba. Después seleccionar “Configurar”.



4. (Opcional) La seguridad de la base de datos se puede modificar ingresando a “Reglas” dentro de la base. Cambiar “now <16106040....” en ambos casos por “request.auth.uid == null”, esto provocará que la base de datos se mantenga en modo prueba.

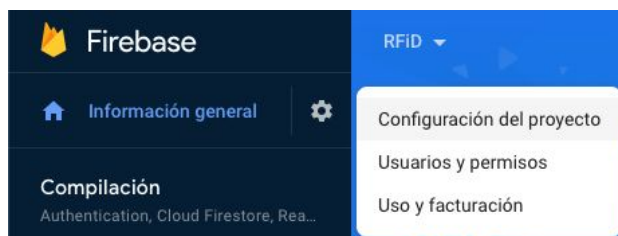
Sección 3

Configurar el proyecto para python, los parámetros:

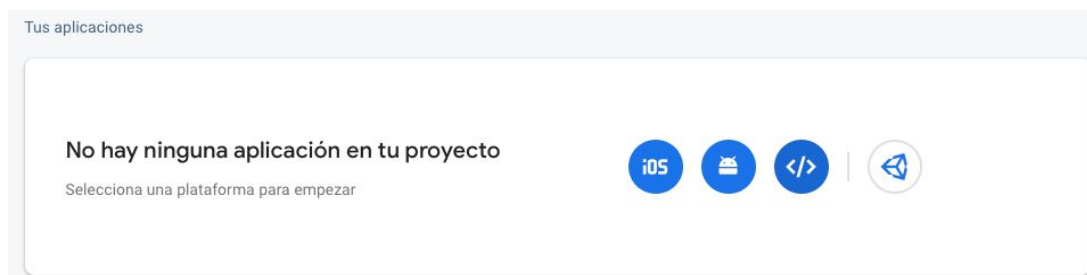
```
config = {  
    "apiKey": "AIzaSyDIPBHyX22JwNw_AOGMrTwDaXDPkAyGKKw",  
    "authDomain": "t3c-inc.firebaseio.com",  
    "databaseURL": "https://t3c-inc.firebaseio.com",  
    "storageBucket": "t3c-inc.appspot.com",  
    "serviceAccount": "/home/pi/Documents/M6E/serviceAccount.json"}  
}
```

Dentro de la consola del proyecto creado

1. En la parte izquierda de la consola del proyecto encontrarás la barra de opciones del nuevo proyecto, en la parte superior encontrarás un icono que desplegará la opción para ir a “Configuración del proyecto”.



2. Dentro de las configuraciones del proyectos, en la pestaña “General”, en la parte inferior de la página aparece una sección de “Tus aplicaciones”, aquí se debe configurar una aplicación web.



3. Registrar la aplicación con el nombre de su preferencia y no seleccionar el “Hosting”.

4. Añadir los parámetros del SDK de:

- apiKey
- authDomain
- databaseURL
- storageBucket

×

Añadir Firebase a tu aplicación web

✓

Registrar la aplicación

2

Añadir SDK de Firebase

Antes de utilizar cualquier servicio de Firebase, copia y pega estas secuencias de comandos en la parte inferior de la etiqueta <body>:

```
<!-- The core Firebase JS SDK is always required and must be listed first -->
<script src="https://www.gstatic.com/firebasejs/8.2.1/firebase-app.js"></script>

<!-- TODO: Add SDKs for Firebase products that you want to use
https://firebase.google.com/docs/web/setup#available-libraries -->

<script>
  // Your web app's Firebase configuration
  var firebaseConfig = {
    apiKey: "AIzaSyCI3Vw4ILju1iQg_gN3Fuy0LA6wZcotbUo",
    authDomain: "rfid-f7cc9.firebaseio.com",
    databaseURL: "https://rfid-f7cc9-default-rtdb.firebaseio.com",
    projectId: "rfid-f7cc9",
    storageBucket: "rfid-f7cc9.appspot.com",
    messagingSenderId: "414019992701",
    appId: "1:414019992701:web:91636ba9659a7759e9c514"
  };
  // Initialize Firebase
  firebase.initializeApp(firebaseConfig);
</script>
```



5. Navegar a la pestaña de “Cuentas de servicio”, aquí se generará la clave privada con las credenciales necesarias para que el dispositivo remoto pueda acceder a la base de datos.

Settings

General

Mensajería en la nube


Integraciones

Cuentas de servicio


Privacidad de datos

Usuarios y permisos


6. Se generará un SDK de administrador para la base de datos, este deberá de ser generado para Python.

 SDK de administrador de Firebase

Credenciales heredadas

 Secretos de la base de datos

Otras cuentas de servicio

 2 cuentas de servicio de Google Cloud Platform [↗](#)

SDK del administrador de Firebase

Tu cuenta de servicio de Firebase se puede utilizar para autenticar varias funciones de Firebase (como Database, Storage y Auth) de forma automatizada a través del SDK de Admin unificado.

[Más información](#) [↗](#)

Cuenta de servicio de Firebase

firebase-adminsdk-ywjrj@rfd-f7cc9.iam.gserviceaccount.com

Fragmento de configuración del SDK de administración

☐ Node.js


☐ Java

☒ Python

☐ Go

```
import firebase_admin
from firebase_admin import credentials

cred = credentials.Certificate("path/to/serviceAccountKey.json")
firebase_admin.initialize_app(cred)
```



Generar nueva clave privada

7. Al seleccionar la opción de python apretar “Generar nueva clave privada”, esta se descargara a tu computadora. Podrás cambiarle el nombre del archivo a “ServiceAccount.json” y este archivo deberá estar en la misma localidad que el código M6E.py.

8. Asegurarse que la dirección del “ServiceAccount.json” sea la misma que la que aparece en “ServiceAccount” en el código python.

Sección 4

Crear la aplicación móvil. (Requisito tener instalado Visual Studio Code y Flutter)

Flutter: <https://flutter.dev/docs/get-started/install>

Visual Studio Code: <https://code.visualstudio.com/download>

Android Studio: <https://developer.android.com/studio/install>

Correr 'flutter doctor' en la terminal y asegurar que todos los requisitos se cumplan.

Dentro de Visual Studio Code

1. Abrir una terminal nueva, navegar a una localidad donde se desea crear el proyecto y teclear "flutter create *nombreProyecto*". Al oprimir enter comenzará la creación del proyecto dentro de un folder.

```
Creating project RFID...
RFID/ios/Runner.xcworkspace/contents.xcworkspacedata (created)
RFID/ios/Runner.xcworkspace xcsharedata/IDEWorkspaceChecks.plist (created)
RFID/ios/Runner.xcworkspace xcsharedata/WorkspaceSettings.xcsettings (created)
RFID/ios/Runner/Info.plist (created)
RFID/ios/Runner/Assets.xcassets/LaunchImage.imageset/LaunchImage@2x.png (created)
RFID/ios/Runner/Assets.xcassets/LaunchImage.imageset/LaunchImage@3x.png (created)
RFID/ios/Runner/Assets.xcassets/LaunchImage.imageset/README.md (created)
RFID/ios/Runner/Assets.xcassets/LaunchImage.imageset/Contents.json (created)
RFID/ios/Runner/Assets.xcassets/LaunchImage.imageset/LaunchImage.png (created)
RFID/ios/Runner/Assets.xcassets/AppIcon.appiconset/Icon-App-76x76@2x.png (created)
RFID/ios/Runner/Assets.xcassets/AppIcon.appiconset/Icon-App-29x29@1x.png (created)
RFID/ios/Runner/Assets.xcassets/AppIcon.appiconset/Icon-App-40x40@1x.png (created)
RFID/ios/Runner/Assets.xcassets/AppIcon.appiconset/Icon-App-20x20@1x.png (created)
```

2. Entrar al documento 'pubspec.yaml' y agregar las siguientes dependencias. (De preferencia buscar la versión más reciente en la página web pub.dev, debajo de la pestaña de "Installing".) Si flutter se instaló de la forma recomendada, las dependencias se agregaran por su cuenta, en caso de no ser así se deberá de escribir en la terminal "pub get".

```
dependencies:
  flutter:
    sdk: flutter

# The following adds the Cupertino Icons font to your application.
# Use with the CupertinoIcons class for iOS style icons.
cupertino_icons: ^0.1.3
http: ^0.12.2
scoped_model: ^1.1.0
google_fonts: ^1.0.0
```

3. Navegar al documento 'main.dart', borrar el código que está y pegar el siguiente código:

```
import 'package:flutter/material.dart';
```

```

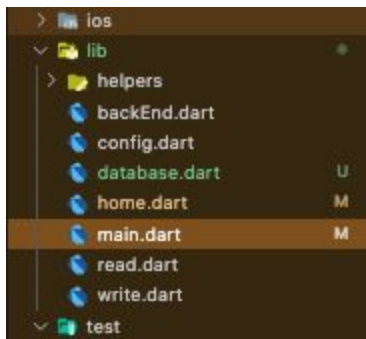
import 'package:scoped_model/scoped_model.dart';
import 'package:RFIDInterface/home.dart';
import 'package:RFIDInterface/backEnd.dart';
import 'package:RFIDInterface/config.dart';
import 'package:RFIDInterface/read.dart';
import 'package:RFIDInterface/write.dart';
import 'package:RFIDInterface/database.dart';

void main() {
  runApp(MyApp());
}

//Static page that has no changes on its structure
class MyApp extends StatelessWidget {
  final MainModel _model = MainModel();
  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    //Allows to pass down data from a parent widget to its descendants
    return ScopedModel<MainModel>(
      model: _model,
      //Material body of the app
      child: MaterialApp(
        title: 'RFID Tag reader',
        theme: ThemeData(
          primarySwatch: Colors.blueGrey,
          visualDensity: VisualDensity.adaptivePlatformDensity,
        ),
        //Home Page
        home: MyHomePage(title: 'RFID Tag reader'),
        //Navigation to other pages
        routes: {
          'config': (BuildContext context) => ConfigRFID(),
          'read': (BuildContext context) => ReadPage(),
          'write': (BuildContext context) => WritePage(),
          'configRFID': (BuildContext context) => ConfigRFID(),
          'dataBase': (BuildContext context) => DataBase(),
        },
      ));
  }
}

```

4. Crear cinco nuevos documentos dentro del folder lib y un folder más con los siguientes nombres.



5. Agregar el siguiente código al documento de 'home.dart':

```
import 'dart:ui';

import 'package:flutter/material.dart';

//Page that have changes on its structure and will be seen in a visual way
class MyHomePage extends StatefulWidget {
  MyHomePage({Key key, this.title}) : super(key: key);
  final String title;

  @override
  _MyHomePageState createState() => _MyHomePageState();
}

class _MyHomePageState extends State<MyHomePage> {
  @override
  //Build context contains the MaterialPage data.
  Widget build(BuildContext context) {
    //Scaffold widget contains the structure of a full page
    return Scaffold(
      //Left Side menu bar
      drawer: Drawer(
        //Asures the content is available in any mobile phone
        child: SafeArea(
          //Container that allows many widgets to be placed in a column
          child: Column(
            children: <Widget>[
              ListTile(
                //Text widget
                title: Text("RFiD Remote V1"),
```



```

        subtitle: Text("Powered by QUANTHEM"),
    ),
    Divider(),
    //Detects a gesture, used to know when the users clicks a button
    GestureDetector(
        child: ListTile(
            title: Text("Read"),
            //Icon widget
            leading: Icon(Icons.receipt),
        ),
        onTap: () {
            //Navigation, will push a page on top of the current,
            //the route is the same as the one at our main
            Navigator.pushNamed(context, 'read');
        },
    ),
    GestureDetector(
        child: ListTile(
            title: Text("Write"),
            leading: Icon(Icons.edit),
        ),
        onTap: () {
            Navigator.pushNamed(context, 'write');
        },
    ),
    GestureDetector(
        child: ListTile(
            title: Text("Data Base"),
            leading: Icon(Icons.folder),
        ),
        onTap: () {
            Navigator.pushNamed(context, 'dataBase');
        },
    ),
    Divider(),
    GestureDetector(
        child: ListTile(
            title: Text("Config"),
            leading: Icon(Icons.pan_tool),
        ),
        onTap: () {
            Navigator.pushNamed(context, 'config');
        },
    ),

```

```

    },
  ),
],
))),
//Top bar of the application
appBar: AppBar(
  title: Text(widget.title),
),
//The body of the scaffold widget,
// Center is a layout widget. It takes a single child and positions it
// in the middle of the parent.
body: Center(
  //Widgets stack in top of each other
  child: Stack(
    children: <Widget>[
      //Widget Container that can have one child
      Container(
        //Makes the container use up all available space
        constraints: BoxConstraints.expand(),
        //Box design
        decoration: BoxDecoration(
          image: DecorationImage(
            image: new AssetImage('assets/background.JPG'),
            fit: BoxFit.cover),
        ),
        child: BackdropFilter(
          filter: ImageFilter.blur(sigmaX: 5, sigmaY: 5),
          child: Container(color: Colors.black.withOpacity(0))),
      //Diagram
      Center(
        child: Padding(
          child: Container(
            child: Image.asset('assets/diagrama.png'),
          ),
          padding: EdgeInsets.all(20)),
      )
    ],
  ),
),
// This trailing comma makes auto-formatting nicer for build methods.
);
}

```

```
}
```

6. Agregar el siguiente código al documento 'read.dart':

```
import 'package:flutter/material.dart';
import 'package:scoped_model/scoped_model.dart';
import 'package:google_fonts/google_fonts.dart';
import 'dart:async';

import 'package:RFIDInterface/backEnd.dart';
import 'package:RFIDInterface/helpers/ensure_visible.dart';

class ReadPage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    // TODO: implement build
    return Scaffold(
      appBar: AppBar(
        title: Text("Read from Device"),
      ),
      body: ReadConfig(),
    );
  }
}

//Page that allows changes on its widgets
class ReadConfig extends StatefulWidget {
  @override
  State<StatefulWidget> createState() {
    // TODO: implement createState
    return _ConfigurationState();
  }
}

class _ConfigurationState extends State<ReadConfig> {
  //This is used to validate the typed information
  final GlobalKey<FormState> _formKeyData = GlobalKey<FormState>();
  //To focus on a text field
  final _readFocusNode = FocusNode();
  //Flag when there is a device registered
  bool isDevice = false;
```

```

//Widget used to visually confirm the registered device
Widget _deviceId(String deviceID) {
  String device = "No device configured";
  if (deviceID != null) {
    device = deviceID;
  }
  return Center(
    child: Container(
      decoration: BoxDecoration(
        borderRadius: BorderRadius.circular(20),
        color: deviceID != null ? Colors.green[200] : Colors.red[200]),
      child: Padding(
        padding: EdgeInsets.all(20),
        child: Text(device,
          //Google fonts is used to give specific fonts for the text.
          style: GoogleFonts.roboto(
            fontSize: 17, fontWeight: FontWeight.w600))));
  }

//Button to read the device or to configure a device
Widget _setReadDevice(MainModel model) {
  //Button widget
  return FlatButton(
    //Async function determines a asynchronous function
    onPressed: () async {
      //Is there a device Id configured
      if (model.deviceIdGet != null) {
        //sends the command to reader to start reading
        bool temp = await model.setReadDevice(model.deviceIdGet);
        setState(() {
          isDevice = temp;
        });
        //After 5 seconds the app reads the database to see what has been read
        new Timer(
          const Duration(seconds: 5),
          () => model
            .getDataFromDevice()
            //reads the database to see if it has been registered
            .then((value) => model.fetchEPC()));
      } else {
        //In case there is no device Id configured then the app will navigate to
        the config page
      }
    },
  );
}

```

```

        Navigator.pushNamed(context, 'config');
    }
},
child: Container(
  child: Padding(
    padding: EdgeInsets.all(10),
    child: Text(model.deviceIdGet != null
      ? "Read RFID Tag"
      : "Configure Device")),
  decoration: BoxDecoration(
    borderRadius: BorderRadius.all(Radius.circular(20)),
    color: Colors.blueGrey[300]),
));
}

//Widget that shows the read EPC or a blank text field
Widget _epcID(String epc) {
  Widget content = EnsureVisibleWhenFocused(
    focusNode: _readFocusNode,
    //Form field for text
    child: TextFormField(
      focusNode: _readFocusNode,
      //No keyboard is needed
      keyboardType: null,
      initialValue: epc == null ? '' : epc,
      decoration: InputDecoration(
        labelText: 'EPC',
        fillColor: Colors.blueGrey[300],
        border: new OutlineInputBorder(
          borderRadius: BorderRadius.circular(10.0))),
      // initialValue: perfil == null ? '' : perfil.title,
    ));

  return content;
}

@override
Widget build(BuildContext context) {
  //Device dimensions for a proper visualisation
  final double deviceWidth = MediaQuery.of(context).size.width;
  final double targetWidth = deviceWidth > 550.0 ? 500.0 : deviceWidth * 0.95;
  final double targetPadding = deviceWidth - targetWidth;

```

```

// TODO: implement build
return ScopedModelDescendant(
  builder: (BuildContext context, Widget child, MainModel model) {
    return Center(
      child: Form(
        key: _formKeyData,
        child: Column(
          // padding: EdgeInsets.symmetric(horizontal: targetPadding / 2),
          children: <Widget>[
            SizedBox(height: 20),
            Center(
              child: Text("Read a RFID tag and then save it in the database",
                style: GoogleFonts.roboto(fontSize: 20))),
            SizedBox(height: 20),
            //Device configured widget
            _deviceId(model.deviceIdGet),
            //Flat button to either configure device or read device
            _setReadDevice(model),
            Divider(
              thickness: 5,
            ),
            //Loading indicator
            model.isLoading
              ? Padding(
                  padding: EdgeInsets.only(left: 90, right: 90),
                  child: CircularProgressIndicator())
              : Expanded(
                  child: ListView.builder(
                    itemCount: model.epcID.length > 0
                      ? model.epcID[model.epcID.length - 1] == "null"
                        ? 0
                        : model.epcID.length
                      : model.epcID.length,
                    itemBuilder: (BuildContext context, int index) {
                      return Container(
                        decoration: BoxDecoration(
                          color: model.epcName[index] == "null"
                            ? Colors.red[200]
                            : Colors.blue[200],
                          borderRadius: BorderRadius.circular(20)),
                        child: Column(
                          children: <Widget>[

```

```

        SizedBox(height: 10),
        _epcID(model.epcID[index]),
        SizedBox(height: 10),
        (model.epcID[index] != null &&
            model.epcName[index] == "null")
            ? Column(
                children: <Widget>[
                    SizedBox(height: 5),
                    Text("Register EPC in Database"),
                    SaveEPC(index)
                ],
            )
            : Center(
                child: model.epcName[index] != "null"
                    ? Text(model.epcName[index],
                        style: GoogleFonts.roboto(
                            fontSize: 25))
                    : Text("Read a EPC")),
                // Container(height: 10, color: Colors.white),
            ],
        ));
    })),
    SizedBox(height: 10)
  ],
),
));
});
}
}

//EPC -> OLD EPC
class SaveEPC extends StatefulWidget {
  final int index;

  SaveEPC(this.index);
  @override
  State<StatefulWidget> createState() {
    return _SaveEPCState();
  }
}

class _SaveEPCState extends State<SaveEPC> {

```



```

        _dataTextController.text, widget.index);
    if (mounted) {
      //Change value for a local variable used on the page
      setState(() {
        isDevice = temp;
      });
    }
    // _formKeyData.currentState.validate();
    // _formKeyData.currentState.save();
    print("Result: " + isDevice.toString());
  },
  child: Container(
    child: Padding(
      padding: EdgeInsets.all(10), child: Text("Save")),
    decoration: BoxDecoration(
      borderRadius:
        BorderRadius.all(Radius.circular(20)),
      color: Colors.green[200]),
  ))
  ],
  ));
},
);
}
}

```

7. Agregar el siguiente código al documento 'write.dart':

```

import 'package:flutter/material.dart';
import 'package:scoped_model/scoped_model.dart';
import 'package:google_fonts/google_fonts.dart';
import 'dart:async';

import 'package:RFIDInterface/backEnd.dart';
import 'package:RFIDInterface/helpers/ensure_visible.dart';

class WritePage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    // TODO: implement build
    return Scaffold(
      appBar: AppBar(
        title: Text("Write new EPC"),

```

```

    ),
    body: Configurations(),
  );
}
}

class Configurations extends StatefulWidget {
  @override
  State<StatefulWidget> createState() {
    // TODO: implement createState
    return _ConfigurationState();
  }
}

class _ConfigurationState extends State<Configurations> {
  final GlobalKey<FormState> _formKeyData = GlobalKey<FormState>();

  final _dataFocusNode = FocusNode();
  final _dataTextController = TextEditingController();

  final _readFocusNode = FocusNode();

  bool isDevice = false;

  Widget _deviceId(String deviceId) {
    String device = "No device configured";
    if (deviceId != null) {
      device = deviceId;
    }
    return Center(
      child: Container(
        decoration: BoxDecoration(
          borderRadius: BorderRadius.circular(20),
          color: deviceId != null ? Colors.green[200] : Colors.red[200]),
        child: Padding(
          padding: EdgeInsets.all(20),
          child: Text(device,
            style: GoogleFonts.roboto(
              fontSize: 17, fontWeight: FontWeight.w600))));
    )
  }

  Widget _setEPC(MainModel model) {

```

```

return FlatButton(
  onPressed: () async {
    if (model.deviceIdGet != null) {
      // bool temp = await model.setWriteDevice(model.deviceIdGet);
      setState(() {
        // isDevice = temp;
      });
      new Timer(
        const Duration(seconds: 10),
        () => model
          .getDataFromDevice()
          .then((value) => model.fetchEPC()));
    } else {
      Navigator.pushNamed(context, 'config');
    }
  },
  child: Container(
    child: Padding(
      padding: EdgeInsets.all(10),
      child: Text(model.deviceIdGet != null
        ? "Write RFID Tag"
        : "Configure Device")),
    decoration: BoxDecoration(
      borderRadius: BorderRadius.all(Radius.circular(20)),
      color: Colors.blueGrey[300]),
  ));
}

Widget _epcData() {
  return EnsureVisibleWhenFocused(
    focusNode: _dataFocusNode,
    child: TextFormField(
      focusNode: _dataFocusNode,
      keyboardType: null,
      decoration: InputDecoration(
        labelText: 'Save TAG ',
        border: new OutlineInputBorder(
          borderRadius: BorderRadius.circular(25.0))),
    controller: _dataTextController,
    validator: (String value) {
      if (value.isEmpty || value.length < 3) {
        return 'The name must be atleast +3 characters';
      }
    }
  );
}

```

```

        } else {
            return null;
        }
    },
));
}

@override
Widget build(BuildContext context) {
    final double deviceWidth = MediaQuery.of(context).size.width;
    final double targetWidth = deviceWidth > 550.0 ? 500.0 : deviceWidth * 0.95;
    final double targetPadding = deviceWidth - targetWidth;
    // TODO: implement build
    return ScopedModelDescendant(
        builder: (BuildContext context, Widget child, MainModel model) {
            return Center(
                child: Form(
                    key: _formKeyData,
                    child: Column(
                        // padding: EdgeInsets.symmetric(horizontal: targetPadding / 2),
                        children: <Widget>[
                            SizedBox(height: 20),
                            Center(
                                child: Text("Write EPC in a Tag",
                                    style: GoogleFonts.roboto(fontSize: 20))),
                            SizedBox(height: 20),
                            _deviceId(model.deviceIdGet),
                            model.isLoading
                                ? Padding(
                                    padding: EdgeInsets.only(left: 90, right: 90),
                                    child: CircularProgressIndicator())
                                : Expanded(
                                    child: ListView.builder(
                                        itemCount: model.epcID.length > 0
                                            ? model.epcID[model.epcID.length - 1] == "null"
                                                ? 0
                                                : model.epcID.length
                                            : model.epcID.length,
                                        itemBuilder: (BuildContext context, int index) {
                                            return WriteEPC(model, index);
                                        }
                                    )),
                            Divider(

```

```

        thickness: 5,
    ),
    _setEPC(model),
],
),
));
});
}
}

class WriteEPC extends StatefulWidget {
    final MainModel model;
    final int index;

    WriteEPC(this.model, this.index);
    @override
    State<StatefulWidget> createState() {
        return _WriteEPCState(model);
    }
}

class _WriteEPCState extends State<WriteEPC> {
    final MainModel model;

    _WriteEPCState(this.model);

    final _epcFocusNode = FocusNode();
    final _epcTextController = TextEditingController();

    final GlobalKey<FormState> _formKeyData = GlobalKey<FormState>();

    bool isDevice = false;

    Widget epcID(String epc, int index) {
        //Gives initial value to the text form field
        if (epc == null && _epcTextController.text.trim() == '') {
            _epcTextController.text = '';
        } else if (epc != null && _epcTextController.text.trim() == '') {
            _epcTextController.text = epc;
        } else if (epc != null && _epcTextController.text.trim() != '') {
            _epcTextController.text = _epcTextController.text;
        } else if (epc == null && _epcTextController.text.trim() != '') {

```

```

        _epcTextController.text = _epcTextController.text;
    } else {
        _epcTextController.text = '';
    }
    Widget content = EnsureVisibleWhenFocused(
        focusNode: _epcFocusNode,
        child: TextFormField(
            focusNode: _epcFocusNode,
            keyboardType: null,
            // initialValue: epc == null ? '' : epc,
            controller: _epcTextController,
            decoration: InputDecoration(
                labelText: 'EPC' + index.toString(),
                fillColor: Colors.blueGrey[300],
                border: new OutlineInputBorder(
                    borderRadius: BorderRadius.circular(10.0)),
            // initialValue: perfil == null ? '' : perfil.title,
            validator: (String value) {
                if (value.isEmpty || value.length < 24 || value.length > 24) {
                    return 'The EPC must be have 24 Hexadecimal values';
                } else {
                    return null;
                }
            }
        )),

    return content;
}

@override
Widget build(BuildContext context) {
    bool changed = false;
    return Center(
        child: Form(
            key: _formKeyData,
            child: Column(
                children: <Widget>[
                    SizedBox(height: 5),
                    Text(model.epcID[widget.index]),
                    SizedBox(height: 5),
                    epcID(model.epcID[widget.index], widget.index + 1),
                    FlatButton(
                        onPressed: () async {

```

```

        if (!_formKeyData.currentState.validate()) {
          print("returning");
          return;
        }
        bool temp = await model.changeEPC(
          _epcTextController.text, widget.index);
        if (temp) {
          print("changed Temp");
          new Timer(const Duration(seconds: 5), () async {
            print("Time Is UP");
            bool temp1 = await model.epcIsChanged(
              _epcTextController.text, widget.index);
          });
        }
        // _formKeyData.currentState.validate();
        // _formKeyData.currentState.save();
      },
      child: Icon(Icons.update)),
      SizedBox(height: 10),
    ],
  ));
}
}

```

8. Agregar el siguiente código al documento 'database.dart':

```

import 'package:RFIDInterface/read.dart';
import 'package:flutter/material.dart';
import 'package:scoped_model/scoped_model.dart';
import 'package:google_fonts/google_fonts.dart';
import 'dart:async';

import 'package:RFIDInterface/backEnd.dart';
import 'package:RFIDInterface/helpers/ensure_visible.dart';

class DataBasePage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    // TODO: implement build
    return Scaffold(
      appBar: AppBar(
        title: Text("Database"),
      ),
    ),
  },
}

```

```

        body: DataBase(),
    );
}
}

//Non Static Page
class DataBase extends StatefulWidget {
  @override
  State<StatefulWidget> createState() {
    // TODO: implement createState
    return _ConfigurationState();
  }
}

class _ConfigurationState extends State<DataBase> {
  //This is used to validate the typed information
  final GlobalKey<FormState> _formKeyData = GlobalKey<FormState>();

  //List of each item read from database
  List<bool> _edit = [];

  @override
  Widget build(BuildContext context) {
    //Device dimensions for a proper visualisation
    final double deviceWidth = MediaQuery.of(context).size.width;
    final double targetWidth = deviceWidth > 550.0 ? 500.0 : deviceWidth * 0.95;
    final double targetPadding = deviceWidth - targetWidth;
    // TODO: implement build
    return ScopedModelDescendant(
      builder: (BuildContext context, Widget child, MainModel model) {
        return Center(
          child: Form(
            key: _formKeyData,
            child: Column(
              children: <Widget>[
                SizedBox(height: 20),
                Center(
                  child: Column(children: [
                    Text("Read from Database saved EPCs ",
                      style: GoogleFonts.roboto(fontSize: 20)),
                    Text(
                      "To confirme read the database again, refreshing with the button.",

```



```

        style: GoogleFonts.roboto(fontSize: 12))
    ])),
    SizedBox(height: 20),
    Divider(
      thickness: 5,
    ),
    FlatButton(
      child: Icon(Icons.refresh),
      onPressed: () async {
        //From scoped model, return the quantity of items
        model.fetchEPCsFromDatabase().then((size) {
          _edit = [];
          for (var i = 0; i < size; i++) {
            //start by viewing the save EPCs, no edit
            _edit.add(false);
          }
        });
      },
    ),
    model.isLoading
      ? Padding(
          padding: EdgeInsets.only(left: 90, right: 90),
          child: CircularProgressIndicator()
        )
      : Expanded(
          child: ListView.builder(
            itemCount: model.epcID.length > 0
              ? model.epcID[model.epcID.length - 1] == "null"
                ? 0
                : model.epcID.length
              : model.epcID.length,
            itemBuilder: (BuildContext context, int index) {
              return Container(
                decoration: BoxDecoration(
                  color: model.epcName[index] == "null"
                    ? Colors.red[200]
                    : Colors.blue[200],
                  borderRadius: BorderRadius.circular(20)),
                child: Column(
                  children: <Widget>[
                    SizedBox(height: 20),
                    (model.epcID[index] != null &&
                      model.epcName[index] == "null")

```

```

        ? Column(
          children: <Widget>[
            SizedBox(height: 5),
            Text("Register EPC in Database"),
          ],
        )
      : Center(
        child: model.epcName[index] != "null"
          //When pressed the item should toggle to
edit

          ? GestureDetector(
            onTap: () {
              setState(() {
                //Toggle item's index flag
                _edit[index] =
                  !_edit[index];
              });
            },
            //Edit or view
            child: !_edit[index]
              ? Text(
                model.epcName[index],
                style: GoogleFonts
                  .roboto(
                    fontSize: 25))
              : SaveEPC(index)
            : Text("Read a EPC")),
          // Container(height: 10, color: Colors.white),
        ],
      );
    })),
    SizedBox(height: 10)
  ],
),
));
});
}
}

```

9. Agregar el siguiente código al archivo 'config.dart':

```
import 'package:flutter/material.dart';
```

```

import 'package:scoped_model/scoped_model.dart';

import 'package:RFIDInterface/backEnd.dart';
import 'package:RFIDInterface/helpers/ensure_visible.dart';

class ConfigRFID extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    // TODO: implement build
    return Scaffold(
      appBar: AppBar(
        title: Text("RFID Configurations"),
      ),
      body: Configurations(),
    );
  }
}

class Configurations extends StatefulWidget {
  @override
  State<StatefulWidget> createState() {
    // TODO: implement createState
    return _ConfigurationState();
  }
}

class _ConfigurationState extends State<Configurations> {
  final GlobalKey<FormState> _formKey = GlobalKey<FormState>();
  final GlobalKey<FormState> _formKeyDevice = GlobalKey<FormState>();

  final _deviceFocusNode = FocusNode();
  final _deviceTextController = TextEditingController();

  final _readFocusNode = FocusNode();
  final _readTextController = TextEditingController();

  final _writeFocusNode = FocusNode();
  final _writeTextController = TextEditingController();

  bool isDevice = true;

  String _deviceIdValue;

```

```

int _readPowerValue;
int _writePowerValue;

Widget _deviceId(String deviceID) {
  if (deviceID == null && _deviceTextController.text.trim() == '') {
    _deviceTextController.text = '';
  } else if (deviceID != null && _deviceTextController.text.trim() == '') {
    _deviceTextController.text = deviceID;
  } else if (deviceID != null && _deviceTextController.text.trim() != '') {
    _deviceTextController.text = _deviceTextController.text;
  } else if (deviceID == null && _deviceTextController.text.trim() != '') {
    _deviceTextController.text = _deviceTextController.text;
  } else {
    _deviceTextController.text = '';
  }
  return EnsureVisibleWhenFocused(
    focusNode: _deviceFocusNode,
    child: TextFormField(
      focusNode: _deviceFocusNode,
      decoration: InputDecoration(
        labelText: 'Device ID',
        border: new OutlineInputBorder(
          borderRadius: BorderRadius.circular(25.0))),
      controller: _deviceTextController,
      // initialValue: perfil == null ? '' : perfil.title,
      validator: (String value) {
        if (value.isEmpty || value.length < 3) {
          return 'The device ID must be at least +3 characters';
        } else if (!isDevice) {
          return 'Device Id Invalid';
        } else {
          return null;
        }
      },
      onSave: (String value) {
        _deviceIdValue = value;
      },
    ));
}

Widget _setDeviceId(MainModel model) {
  return FlatButton(

```

```

onPressed: () async {
  isDevice = true;
  if (!_formKeyDevice.currentState.validate()) {
    print("returning");
    return;
  }
  print("GOOD");
  bool temp = await model.setDeviceId(_deviceTextController.text);
  setState(() {
    isDevice = temp;
  });
  _formKeyDevice.currentState.validate();
  _formKeyDevice.currentState.save();

  print("Result: " + isDevice.toString());
},
child: Container(
  child: Padding(
    padding: EdgeInsets.all(10), child: Text("Set Device ID")),
  decoration: BoxDecoration(
    borderRadius: BorderRadius.circular(20),
    color: Colors.blueGrey),
));
}

Widget _readPower(int readPower) {
  if (readPower == null && _readTextController.text.trim() == '') {
    _readTextController.text = '';
  } else if (readPower != null && _readTextController.text.trim() == '') {
    _readTextController.text = readPower.toString();
  } else if (readPower != null && _readTextController.text.trim() != '') {
    _readTextController.text = readPower.toString();
  } else if (readPower == null && _readTextController.text.trim() != '') {
    _readTextController.text = readPower.toString();
  } else {
    _readTextController.text = '';
  }
  return EnsureVisibleWhenFocused(
    focusNode: _readFocusNode,
    child: TextFormField(
      focusNode: _readFocusNode,
      keyboardType: TextInputType.number,

```

```

        decoration: InputDecoration(
          labelText: 'Set the Read Power (dBm)',
          border: new OutlineInputBorder(
            borderRadius: BorderRadius.circular(25.0))),
        controller: _readTextController,
        validator: (String value) {
          if (value.isEmpty ||
            value.length > 2 ||
            !RegExp(r'^(?:[1-9]\d*|0)?(?:\.\d+)?$').hasMatch(value)) {
            return 'The power must be set by two digit';
          }
        },
        onSave: (String value) {
          _readPowerValue = int.parse(value);
        },
      ));
}

Widget _setReadPower(MainModel model) {
  return FlatButton(
    onPressed: () {
      if (!_formKey.currentState.validate() || _deviceIdValue == null) {
        return;
      }
      _formKey.currentState.save();
      model.setReadPower(
        _deviceTextController.text, int.parse(_readTextController.text));
    },
    child: Container(
      child: Padding(
        padding: EdgeInsets.all(10),
        child: Text("Set readPower"),
      ),
      decoration: BoxDecoration(
        borderRadius: BorderRadius.all(Radius.circular(20)),
        color: Colors.blueGrey));
    ),
  );

Widget _writePower(int writePower) {
  if (writePower == null && _writeTextController.text.trim() == '') {
    _writeTextController.text = '';
  } else if (writePower != null && _writeTextController.text.trim() == '') {

```

```

        _writeTextController.text = writePower.toString();
    } else if (writePower != null && _writeTextController.text.trim() != '') {
        _writeTextController.text = writePower.toString();
    } else if (writePower == null && _writeTextController.text.trim() != '') {
        _writeTextController.text = writePower.toString();
    } else {
        _writeTextController.text = '';
    }
    return EnsureVisibleWhenFocused(
        focusNode: _writeFocusNode,
        child: TextFormField(
            // initialValue: writePower == null ? '' : writePower.toString(),
            keyboardType: TextInputType.number,
            focusNode: _writeFocusNode,

            decoration: InputDecoration(
                labelText: 'Set the Write Power (dBm)',
                border: new OutlineInputBorder(
                    borderRadius: BorderRadius.circular(25.0))),
            // initialValue: perfil == null ? '' : perfil.title,
            controller: _writeTextController,
            validator: (String value) {
                // if (value.trim().length <= 0) {
                if (value.isEmpty || value.length > 2) {
                    return 'The power must be set by two digit';
                }
            },
            onSave: (String value) {
                setState(() {
                    _writePowerValue = int.parse(value);
                });
            },
        ));
}

Widget _setWritePower(MainModel model) {
    return FlatButton(
        onPressed: () {
            if (!_formKey.currentState.validate() || _deviceIdValue == null) {
                return;
            }
            _formKey.currentState.save();
        },
    );
}

```

```

        model.setWritePower(
            _deviceTextController.text, int.parse(_writeTextController.text));
    },
    child: Container(
      child: Padding(
        padding: EdgeInsets.all(10),
        child: Text("Set writePower"),
      ),
      decoration: BoxDecoration(
        borderRadius: BorderRadius.all(Radius.circular(20)),
        color: Colors.blueGrey));
  }

  @override
  Widget build(BuildContext context) {
    final double deviceWidth = MediaQuery.of(context).size.width;
    final double targetWidth = deviceWidth > 550.0 ? 500.0 : deviceWidth * 0.95;
    final double targetPadding = deviceWidth - targetWidth;
    // TODO: implement build
    return ScopedModelDescendant(
      builder: (BuildContext context, Widget child, MainModel model) {
        return Center(
          child: ListView(
            padding: EdgeInsets.symmetric(horizontal: targetPadding / 2),
            children: <Widget>[
              SizedBox(height: 20),
              Form(
                key: _formKeyDevice,
                child: Column(
                  children: <Widget>[
                    _deviceId(model.deviceIdGet),
                    _setDeviceId(model),
                    Divider(),
                  ],
                )),
            model.isLoading
              ? Padding(
                  padding: EdgeInsets.only(left: 90, right: 90),
                  child: CircularProgressIndicator())
              : Form(
                  key: _formKey,
                  child: Column(

```



```

        children: <Widget>[
          SizedBox(height: 10),
          _readPower(model.readPower),
          _setReadPower(model),
          SizedBox(height: 10),
          _writePower(model.writePower),
          _setWritePower(model),
          SizedBox(height: 10),
        ],
      )),
    ],
  ),
);
});
}
}

```

10. La funcionalidad de la aplicación, agregar el siguiente código al archivo 'backEnd.dart':

```

import 'package:scoped_model/scoped_model.dart';
import 'dart:convert' as convert;
import 'dart:async';
import 'package:http/http.dart' as http;

class MainModel extends Model {
  //Indicator if process is running
  bool _isLoading = false;
  bool get isLoading {
    return _isLoading;
  }

  //Device ID
  String _deviceId;
  String get deviceIdGet {
    return _deviceId;
  }

  //EPC temp
  String _epc;
  String get epc {
    return _epc;
  }
}

```

```
//ReadPower
int _readPower;
int get readPower {
    return _readPower;
}

//WritePower
int _writePower;
int get writePower {
    return _writePower;
}

//Array of EPCs read from M6E or Database
List<String> _epcID = [];
List<String> get epcID {
    return _epcID;
}

//EPC to be written
String _writeEPC;
String get writeEPC {
    return _writeEPC;
}

//Array of EPCs name read from Database
List<String> _epcName = [];
List<String> get epcName {
    return _epcName;
}

//Clean EPC's array
void clearDevices() {
    _epcID = [];
    notifyListeners();
}

//Set the device ID on the database
Future<bool> setDeviceId(String deviceId) async {
    _isLoading = true;
    _deviceId = deviceId;
    notifyListeners();
}
```

```

//URL to database
var url = 'https://t3c-inc.firebaseio.com/devices/' + deviceId + '/.json?';
var response = await http.get(url);
print("Response: " + response.statusCode.toString());
//Response code 200 is okay
//Search for Http rest response codes for more info
if (response.statusCode == 200) {
  print("Response body: " + response.body);
  if (response.body != "null") {
    //The data stored at the database contains a JSON format
    var jsonResponse = convert.jsonDecode(response.body);
    _readPower = jsonResponse['read power'];
    _writePower = jsonResponse['write power'];
    _isLoading = false;
    notifyListeners();
    return true;
  } else {
    print("deleting DeviceId");
    _deviceId = null;
    _isLoading = false;
    notifyListeners();
    return false;
  }
} else {
  _isLoading = false;
  notifyListeners();
  return false;
}
}

Future<bool> setReadPower(String deviceId, int _power) async {
  print("Inside ReadPower New");
  _readPower = _power;
  print(_power);
  var url = 'https://t3c-inc.firebaseio.com/devices/' + deviceId + '/.json';
  print(url);
  var response = await http.put(url,
    body: convert.json.encode({
      'read': false,
      'write': false,
      'read EPC': 'null',
      'write EPC': 'null',

```

```

        'read power': _readPower,
        'write power': _writePower
    }));
print(response.statusCode);

    return true;
}

Future<bool> setWritePower(String deviceId, int _power) async {
    print("Inside WritePower New");
    _writePower = _power;
    print(_power);
    var url = 'https://t3c-inc.firebaseio.com/devices/' + deviceId + '.json';
    var response = await http.put(url,
        body: convert.json.encode({
            'read': false,
            'write': false,
            'read EPC': 'null',
            'write EPC': 'null',
            'read power': _readPower,
            'write power': _writePower
        }));
    print(response.statusCode);
    return true;
}

Future<bool> setReadDevice(String deviceId) async {
    print("Inside Set Read Device");
    _isLoading = true;
    _epcID = [];
    notifyListeners();
    var url = 'https://t3c-inc.firebaseio.com/devices/' + deviceId + '.json';
    print(url);
    var response = await http.put(url,
        body: convert.json.encode({
            'read': true,
            'write': false,
            'read EPC': 'null',
            'write EPC': 'null',
            'read power': _readPower,
            'write power': _writePower
        }));
}

```

```

    if (response.statusCode == 200) {
        _isLoading = false;
        notifyListeners();
        return true;
    } else {
        _isLoading = false;
        notifyListeners();
        return false;
    }
}

Future<bool> getDataFromDevice() async {
    _epcName = [];
    _epcID = [];
    _isLoading = true;
    notifyListeners();

    var url = 'https://t3c-inc.firebaseio.com/devices/' + _deviceId + '/.json?';
    var response = await http.get(url);
    if (response.statusCode == 200) {
        print("Response body: " + response.body);
        if (response.body != "null") {
            var jsonResponse = convert.jsonDecode(response.body);
            _readPower = jsonResponse['read power'];
            _writePower = jsonResponse['write power'];
            String _tempEPC = jsonResponse['read EPC'];
            _writeEPC = jsonResponse['write EPC'];
            _epcID = _tempEPC.split(',');
            print("EPCs Length: " + _epcID.length.toString());
            _isLoading = false;
            notifyListeners();
            http.put(url,
                body: convert.json.encode({
                    'read': false,
                    'write': false,
                    'read EPC': 'null',
                    'write EPC': 'null',
                    'read power': _readPower,
                    'write power': _writePower
                }));
            if (_epcID.length < 1) {
                _epcID[0] == "null" ? _epcID = [] : _epcID;
            }
        }
    }
}

```

```

        _isLoading = false;
        notifyListeners();
        return true;
    } else {
        _isLoading = false;
        notifyListeners();
        return false;
    }
} else {
    _isLoading = false;
    notifyListeners();
    return false;
}
}

Future<bool> saveDataEPC(String name, int index) async {
    print("Inside Set Read Device");
    _isLoading = true;
    notifyListeners();
    var url = 'https://t3c-inc.firebaseio.com/EPC/' + epcID[index] + '.json';
    print(url);
    var response =
        await http.put(url, body: convert.json.encode({'name': name}));
    if (response.statusCode == 200) {
        _epcName[index] = name;
        _isLoading = false;
        notifyListeners();
        return true;
    } else {
        _isLoading = false;
        notifyListeners();
        return false;
    }
}

Future<bool> fetchEPC() async {
    print("Inside Fetch EPC");
    _isLoading = true;
    notifyListeners();
    if (epcID != null) {
        print("_read EPC Length: " + _epcID.length.toString());
        for (var i = 0; i < _epcID.length; i++) {

```

```

        print("EPC[" + i.toString() + "]: " + epcID[i]);
        var url = 'https://t3c-inc.firebaseio.com/EPC/' + epcID[i] + '/.json?';
        var response = await http.get(url);
        if (response.statusCode == 200) {
            print("Response body: " + response.body);
            if (response.body != "null") {
                print("EPC SAVED");
                var jsonResponse = convert.jsonDecode(response.body);
                _epcName.add(jsonResponse['name']);
                notifyListeners();
            } else {
                print("EPC NOT SAVED");
                _epcName.add("null");
                notifyListeners();
            }
        }
    }
    print("EPCs Name Length: " + _epcName.length.toString());
    _isLoading = false;
    notifyListeners();
    return true;
} else {
    _epcID = [];
    _isLoading = false;
    notifyListeners();
    return false;
}
}

```

```

Future<bool> changeEPC(String newEPC, int index) async {
    print("Inside Set Read Device");
    _isLoading = true;
    notifyListeners();
    var url = 'https://t3c-inc.firebaseio.com/devices/' + _deviceID + '.json';
    print(url);
    var response = await http.put(url,
        body: convert.json.encode({
            'read': false,
            'write': true,
            'read EPC': epcID[index],
            'write EPC': newEPC,
            'read power': _readPower,

```

```

        'write power': _writePower,
        'EPC': false
    }));
    if (response.statusCode == 200) {
        _isLoading = false;
        notifyListeners();
        return true;
    } else {
        _isLoading = false;
        notifyListeners();
        return false;
    }
}

Future<bool> epcIsChanged(String _epcToChange, int index) async {
    print("Inside EPC Is Changed");
    _isLoading = true;
    notifyListeners();

    var url = 'https://t3c-inc.firebaseio.com/devices/' + _deviceId + '/.json?';
    var response = await http.get(url);
    if (response.statusCode == 200) {
        if (response.body != "null") {
            var jsonResponse = convert.jsonDecode(response.body);
            if (jsonResponse['EPC']) {
                _epcID[index] = _epcToChange;
                _isLoading = false;
                notifyListeners();
                return true;
            } else {
                _isLoading = false;
                notifyListeners();
                return false;
            }
        } else {
            print("Error Reading EPC Flag");
            _isLoading = false;
            notifyListeners();
        }
    }
    _isLoading = false;
    notifyListeners();
}

```



```

        return false;
    }

//Fetch all saved EPC in the database
//Set URL -> response -> save to _epcName & _epcID
Future<int> fetchEPCsFromDatabase() async {
    //prepare the data arrays.
    _epcID = [];
    _epcName = [];
    print("Inside Fetch EPC DATABASE");
    _isLoading = true;
    notifyListeners();
    var url = 'https://t3c-inc.firebaseio.com/EPC/.json?';
    var response = await http.get(url);
    if (response.statusCode == 200) {
        // print("Response body: " + response.body);
        if (response.body != "null") {
            print("EPC SAVED");
            Map jsonResponse = convert.jsonDecode(response.body);
            //Iterate through the each Map item
            //Key value is EPC and value contains: 'name: abcd'
            //split value to get the name
            jsonResponse.forEach((key, value) {
                _epcID.add(key);
                _epcName.add(value
                    .toString()
                    .split(':')[1]
                    .substring(0, value.toString().split(':')[1].length - 1));
            });
            print(_epcName[0]);
            print(_epcID[0]);
            print(jsonResponse);
            _isLoading = false;
            notifyListeners();
            return _epcName.length;
        } else {
            print("EPC NOT SAVED");
            _epcName.add("null");
            notifyListeners();
            return 0;
        }
    }
}

```

```

    _isLoading = false;
    notifyListeners();
    return 0;
  }
}

```

11. Dentro del folder helpers crear un documento llamado 'ensure_visible.dart' y agregar el siguiente código:

```

import 'dart:async';

import 'package:flutter/rendering.dart';
import 'package:flutter/widgets.dart';
import 'package:meta/meta.dart';

class EnsureVisibleWhenFocused extends StatefulWidget {
  const EnsureVisibleWhenFocused({
    Key key,
    @required this.child,
    @required this.focusNode,
    this.curve: Curves.ease,
    this.duration: const Duration(milliseconds: 100),
  }) : super(key: key);

  /// The node we will monitor to determine if the child is focused
  final FocusNode focusNode;

  /// The child widget that we are wrapping
  final Widget child;

  /// The curve we will use to scroll ourselves into view.
  ///
  /// Defaults to Curves.ease.
  final Curve curve;

  /// The duration we will use to scroll ourselves into view
  ///
  /// Defaults to 100 milliseconds.
  final Duration duration;

  EnsureVisibleWhenFocusedState createState() => new EnsureVisibleWhenFocusedState();
}

```

```

class EnsureVisibleWhenFocusedState extends State<EnsureVisibleWhenFocused> {
  @override
  void initState() {
    super.initState();
    widget.focusNode.addListener(_ensureVisible);
  }

  @override
  void dispose() {
    super.dispose();
    widget.focusNode.removeListener(_ensureVisible);
  }

  Future<Null> _ensureVisible() async {

    await Future.delayed(const Duration(milliseconds: 300));

    if (!widget.focusNode.hasFocus)
      return;

    final RenderObject object = context.findRenderObject();
    final RenderAbstractViewport viewport = RenderAbstractViewport.of(object);
    assert(viewport != null);

    ScrollableState scrollableState = Scrollable.of(context);
    assert(scrollableState != null);

    ScrollPosition position = scrollableState.position;
    double alignment;
    if (position.pixels > viewport.getOffsetToReveal(object, 0.0).offset) {
      // Move down to the top of the viewport
      alignment = 0.0;
    } else if (position.pixels < viewport.getOffsetToReveal(object, 1.0).offset) {
      // Move up to the bottom of the viewport
      alignment = 1.0;
    } else {
      // No scrolling is necessary to reveal the child
      return;
    }
    position.ensureVisible(
      object,

```

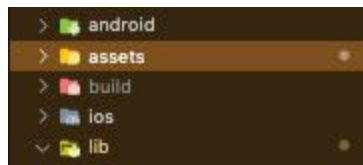
```

        alignment: alignment,
        duration: widget.duration,
        curve: widget.curve,
    );
}

Widget build(BuildContext context) => widget.child;
}

```

12. Agregar el folder de 'assets' en el folder principal.



13. Agregar el nombre de las imágenes del folder de `assets` en el documento 'pubspeck.yaml'.

```

# To add assets to your application, add an assets section, like this:
assets:
  - assets/diagrama.png
  - assets/background.JPG

```

Sección 5

Correr la aplicación en un emulador o en un teléfono físico.

Desde el IDE de Visual Studio Code:

1. Tener un dispositivo disponible, ya sea un emulador corriendo en la computadora o un teléfono físico conectado a la computadora.

1.1 Abrir un emulador Android con android studio:

<https://developer.android.com/studio/run/managing-avds>

1.2 Abrir un emulador iOS (solo Mac):

https://medium.com/@abrisad_it/how-to-launch-ios-simulator-and-android-emulator-on-mac-cd198295532e

1.3 En caso de tener el dispositivo físico, conectarlo a la computadora por USB.

2. El nombre del dispositivo deberá aparecer en la parte inferior izquierda del IDE, correr 'flutter run' en la terminal para asegurar que todo este en orden. En caso de no ser así corregir con la ayuda de los mensajes.

3. Correr 'flutter run'.