12. Anexos

Antes de mostrar lo anexos, comentar que aquí es dónde podremos encontrar la mayor parte del desarrollo de nuestra aplicación. Pues aparece gran parte del código que la conforma. No ha sido fácil organizarlo. Aunque preferíamos redactarlo de nuevo para que así apareciera de forma clara en el memória en vez de realizar capturas de pantalla. De esta forma también disponíamos de un apartado al cual hacer referencia a nuestro propio código sin necesidad de mostrarlo a lo largo de la redacción del trabajo.

Puesto que puede que no todos los lectores de este trabajo estén interesados en leer y comprender nuestro código.

El anexo comprende los siguientes puntos:

12.1. Respuestas JSON

12.1.1. getFlightInfo *response*

```
{
    "flightInfo": {
        "flight_date": "2020-06-07",
        "flight_status": "scheduled",
        "departure": {
            "airport": "Zurich",
            "timezone": "Europe/Zurich",
            "iata": "ZRH",
            "icao": "LSZH",
            "terminal": "1",
            "gate": "A74",
            "delay": null,
            "scheduled": "2020-06-07T11:50:00+00:00",
            "estimated": "2020-06-07T11:50:00+00:00",
            "actual": null,
            "estimated_runway": null,
            "actual_runway": null
        },
        "arrival": {
            "airport": "Schiphol",
            "timezone": "Europe/Amsterdam",
            "iata": "AMS",
            "icao": "EHAM",
            "terminal": "3",
            "gate": null,
            "baggage": null,
            "delay": null,
            "scheduled": "2020-06-07T13:35:00+00:00",
            "estimated": "2020-06-07T13:35:00+00:00",
            "actual": null,
            "estimated_runway": "2020-06-07T13:35:00+00:00",
            "actual_runway": null
        },
        "airline": {
            "name": "KLM",
            "iata": "KL",
```

```json
            "icao": "KLM"
        },
        "flight": {
            "number": "1958",
            "iata": "KL1958",
            "icao": "KLM1958",
            "codeshared": null
        },
        "aircraft": {
            "registration": "PH-EZS",
            "iata": "E190",
            "icao": "E190",
            "icao24": "484CC3"
        },
        "live": {
            "updated": "2020-06-07T09:43:59+00:00",
            "latitude": 52.25,
            "longitude": 4.63,
            "altitude": 0,
            "direction": 351.56,
            "speed_horizontal": 4.644,
            "speed_vertical": 0,
            "is_ground": false
        }
    },
    "depCity": {
        "airportName": "Zürich Airport",
        "latitude": 47.458218,
        "longitude": 8.555475
    },
    "arrCity": {
        "airportName": "Amsterdam Airport Schiphol",
        "latitude": 52.31054,
        "longitude": 4.7682743
    }
}
```

12.1.2. getArrivalTimeTable *response*

```
{
    "arrivalInfo": [
        {
            "flight_date": "2020-06-06",
            "flight_status": "active",
            "departure": {
                "airport": "Ataturk Airport",
                "timezone": "Europe/Istanbul",
                "iata": "IST",
                "icao": "LTBA",
                "terminal": null,
                "gate": null,
                "delay": null,
                "scheduled": "2020-06-06T04:10:00+00:00",
                "estimated": "2020-06-06T04:10:00+00:00",
                "actual": null,
                "estimated_runway": "2020-06-06T03:56:00+00:00",
                "actual_runway": "2020-06-06T03:56:00+00:00"
            },
            "arrival": {
                "airport": "El Prat De Llobregat",
                "timezone": "Europe/Madrid",
                "iata": "BCN",
                "icao": "LEBL",
                "terminal": "1",
                "gate": null,
                "baggage": null,
                "delay": null,
                "scheduled": "2020-06-06T06:00:00+00:00",
                "estimated": "2020-06-06T06:00:00+00:00",
                "actual": null,
                "estimated_runway": "2020-06-06T05:53:00+00:00",
                "actual_runway": null
            },
            "airline": {
                "name": "Turkish Airlines",
                "iata": "TK",
                "icao": "THY"
            },
            "flight": {
                "number": "6351",
                "iata": "TK6351",
                "icao": "THY6351",
                "codeshared": null
            },
            "aircraft": null,
            "live": null
        }
    ]
}
```

12.1.3. getAirlineInfo *response*

```
{
    "airlineData": {
          "fleet_average_age": "6.7",
          "callsign": "VUELING",
          "hub_code": "BCN",
          "iata_code": "VY",
          "icao_code": "VLG",
          "country_iso2": "ES",
          "date_founded": "2004",
          "iata_prefix_accounting": "30",
          "airline_name": "Vueling",
          "country_name": "Spain",
          "fleet_size": "105",
          "status": "active",
          "type": "scheduled"
      }
  }
```

12.1.4. getAirportInfo *response*

```
{
    "airportInfo": {
        "id": 583,
        "iata": "BCN",
        "icao": "LEBL",
        "name": "Barcelona-El Prat Airport",
        "location": "Barcelona, Catalonia, Spain",
        "street_number": "",
        "street": "",
        "city": "El Prat de Llobregat",
        "county": "Barcelona",
        "state": "Catalonia",
        "country_iso": "ES",
        "country": "Spain",
        "postal_code": "8820",
        "phone": "+34 902 40 47 04",
        "latitude": 41.297443,
        "longitude": 2.0832942,
        "uct": 120,
"website": "http://www.aena-aeropuertos.es/csee/Satellite/Aeropuerto-Barcelona"
    },
    "airportDepartures": [
        {
            "flight_date": "2020-06-07",
            "flight_status": "active",
            "departure": {
                "airport": "El Prat De Llobregat",
                "timezone": "Europe/Madrid",
                "iata": "BCN",
                "icao": "LEBL",
                "terminal": "2",
                "gate": null,
                "delay": null,
                "scheduled": "2020-06-07T09:50:00+00:00",
                "estimated": "2020-06-07T09:50:00+00:00",
                "actual": null,
                "estimated_runway": "2020-06-07T10:20:00+00:00",
                "actual_runway": "2020-06-07T10:20:00+00:00"
        },
        "arrival": {
            "airport": "Düsseldorf International Airport",
            "timezone": "Europe/Berlin",
            "iata": "DUS",
            "icao": "EDDL",
            "terminal": "2",
            "gate": "B92",
            "baggage": null,
            "delay": 1,
            "scheduled": "2020-06-07T12:15:00+00:00",
            "estimated": "2020-06-07T12:15:00+00:00",
            "actual": null,
            "estimated_runway": "2020-06-07T12:04:00+00:00",
            "actual_runway": null
        },
        "airline": {
            "name": "Lufthansa",
            "iata": "LH",
            "icao": "DLH"
        },
        "flight": {
            "number": "5193",
            "iata": "LH5193",
            "icao": "DLH5193",
            "codeshared": {
                "airline_name": "eurowings",
                "airline_iata": "ew",
                "airline_icao": "ewg",
                "flight_number": "9445",
                "flight_iata": "ew9445",
                "flight_icao": "ewg9445"
            }
        },
        "aircraft": null,
        "live": null
    },
],
"airportArrivals": [
    {
```

```
"flight_date": "2020-06-06",
"flight_status": "active",
"departure": {
    "airport": "Ataturk Airport",
    "timezone": "Europe/Istanbul",
    "iata": "IST",
    "icao": "LTBA",
    "terminal": null,
    "gate": null,
    "delay": null,
    "scheduled": "2020-06-06T04:10:00+00:00",
    "estimated": "2020-06-06T04:10:00+00:00",
    "actual": null,
    "estimated_runway": "2020-06-06T03:56:00+00:00",
    "actual_runway": "2020-06-06T03:56:00+00:00"
},
"arrival": {
    "airport": "El Prat De Llobregat",
    "timezone": "Europe/Madrid",
    "iata": "BCN",
    "icao": "LEBL",
    "terminal": "1",
    "gate": null,
    "baggage": null,
    "delay": null,
    "scheduled": "2020-06-06T06:00:00+00:00",
    "estimated": "2020-06-06T06:00:00+00:00",
    "actual": null,
    "estimated_runway": "2020-06-06T05:53:00+00:00",
    "actual_runway": null
},
"airline": {
    "name": "Turkish Airlines",
    "iata": "TK",
    "icao": "THY"
},
"flight": {
    "number": "6351",
    "iata": "TK6351",
    "icao": "THY6351",
                "codeshared": null
            },
            "aircraft": null,
            "live": null
        }
    ]
}
```

12.1.5. getCityData *response*

```
{
    "cityName": "New York City",
    "cityPopulation": 8175133,
    "cityGeoname_id": 5128581,
    "countryName": "United States",
    "countryIso3": "USA",
    "countryPopulation": 310232863,
    "countryCurrency": "USD",
    "countryGeoname_id": 6252001,
    "scoreData": {
        "scoreInfo": [
            {
                "color": "#f3c32c",
                "name": "Housing",
                "score_out_of_10": 1
            },
            {
                "color": "#f3d630",
                "name": "Cost of Living",
                "score_out_of_10": 2.342
            }
        ],
        "summary": " The New York metropolitan area is one of the most
populated cities in the world. New York City itself is an international
center for numerous industries including finance, theater, television,
film, arts, and technology."
    },
    "images": [
"https://images.pexels.com/photos/466685/pexels-photo-466685.jpeg?auto=
CUT",

"https://images.pexels.com/photos/290386/pexels-photo-290386.jpeg?auto=
CUT",

"https://images.pexels.com/photos/313782/pexels-photo-313782.jpeg?auto=
CUT",

"https://images.pexels.com/photos/1486222/pexels-photo-1486222.jpeg?aut
o=CUT",

"https://images.pexels.com/photos/378570/pexels-photo-378570.jpeg?auto=
CUT",

"https://images.pexels.com/photos/1525612/pexels-photo-1525612.jpeg?aut
o=CUT"
    ]
}
```

NOTE: “CUT” should be replaced by:
compress&cs=tinysrgb&fit=crop&h=1200&w=800

12.1.6. getKiwiFlightTickets *response*

```
[
    {
        "airline": "VY",
        "flightId": "VY1420",
        "from": "BCN",
        "to": "BIO",
        "from_city": "Barcelona",
        "to_city": "Bilbao",
        "country_from": "ES",
        "country_to": "ES",
        "depTime": "2020-07-28T07:00:00.000Z",
        "arrTime": "2020-07-28T08:15:00.000Z",
        "date": "07/07/2020",
        "duration": "1h 15m",
        "distance": 467.32,
        "price": 35,
        "availability": 4,
        "bookingToken": "AyiACQOiOy6ht-ClNNhM_7hf1ONCcvDCkk5zIMfa40",
        "deep_link":
"https://www.kiwi.com/deep?from=BCN&to=BIO&departure"
    },
    {
        "airline": "VY",
        "flightId": "VY1420",
        "from": "BCN",
        "to": "BIO",
        "from_city": "Barcelona",
        "to_city": "Bilbao",
        "country_from": "ES",
        "country_to": "ES",
        "depTime": "2020-07-24T07:00:00.000Z",
        "arrTime": "2020-07-24T08:15:00.000Z",
        "date": "07/07/2020",
        "duration": "1h 15m",
        "distance": 467.32,
        "price": 35,
        "availability": 1,
        "bookingToken": "A1yXd6kqKBGCnEetIRrT2gHJpHwOc2MJ",
        "deep_link":
"https://www.kiwi.com/deep?from=BCN&to=BIO&departure"
    }
]
```

12.2. Ejemplos de Código

12.2.1. Ejemplo de Código *Flutter template*

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

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

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
        visualDensity: VisualDensity.adaptivePlatformDensity,
      ),
      home: MyHomePage(title: 'Flutter Demo Home Page'),
    );
  }
}

class MyHomePage extends StatefulWidget {
  MyHomePage({Key key, this.title}) : super(key: key);

  final String title;

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

class _MyHomePageState extends State<MyHomePage> {
  int _counter = 0;

  void _incrementCounter() {
    setState(() {
      _counter++;
    });
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text(widget.title),
      ),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: <Widget>[
            Text(
              'You have pushed the button this many times:',
            ),
            Text(
              '$_counter',
              style: Theme.of(context).textTheme.headline4,
            ),],),
      ),
      floatingActionButton: FloatingActionButton(
        onPressed: _incrementCounter,
        tooltip: 'Increment',
        child: Icon(Icons.add),
      ),);
  }
}
```

Aunque pueda parecer extenso, no lo es, puesto que con este trozo de código ya disponemos de una aplicación que gestiona el estado de un contador, así como también incluye el diseño de la página mostrada en la figura 17.

El código que conforma la figura 17, existen 2 clases. Una para crear la aplicación MyApp y MyHomePage dónde se define la estructura, la lógica y los elementos visuales que aparecen, la lógica sería la función _incrementCounter(), el estado estaría almacenado en la variable _counter y el accionador de la función sería el onPressed del widget FloatingActionButton.

12.2.2. Ejemplo de CF onRequest

```
exports.getAirlineInfo  =  functions.https.onRequest(async  (req,
resp)=>{

    let airlineSearch:string = req.query.airline;

    const optionAirline = {
        method: 'GET',
      url:
      `http://api.aviationstack.com/v1/airlines?access_key=${acces
      s_key}`,
        headers: {
        'content-type': 'application/json'
      }
    };
    try{
  await request(optionAirline, function(error:any, notShadow:any,
body:any){
          const airlinesDoc = Converted.toAirlines(body);
          console.log(airlinesDoc);
          const airlinesList = airlinesDoc.data;
          let i = 0;
          let trobat = 0;
          while(i < airlinesList.length && trobat == 0){
            let anal = airlinesList[i].airline_name;
            console.log(anal + ' ' + i);
           if(anal.toLowerCase() == airlineSearch.toLowerCase()){
              console.log(airlinesList[i].toString());
              trobat = 1;
            }else{
              i++;
            }}
          if(trobat === 1){
            return resp.send({airlineData: airlinesList[i]});
          }else{
            return resp.send({airlineData: 'NO match found'});
          }
      })
    }catch(e){
      console.log(e.toString());
    }
  })
```

En el código anterior existen funciones como por ejemplo `.toAirlines`, que nos permiten procesar un objeto JSON y convertirlo a un objeto manipulable para construir nuestra respuesta.

Ahora podemos ver que la CF devuelve de la lista de aerolíneas que procesa, el elemento [i]. El objeto JSON respuesta de esta función ya ha sido mostrado en el apartado de APIs bajo el nombre de getAirlineInfo().

12.2.3. Ejemplo de CF onCall

```
exports.sendDeviceToken = functions.https.onCall(async (data, context)
=> {
    const uid = context.auth?.uid;
    return db.doc(`users/${uid}`).update({
        tokenId: data.tokenId
    }).then(() => {
        console.log('Done');
        return true;
    }).catch(console.error);
})
```

Para que este código realmente sea ejecutado se debe establecer acceso de administrador.

En esta CF se puede apreciar cómo somos capaces de extraer el uid (user identifier) del parámetro data, que va intrínseco en la request. Esta función nos permite conocer a qué dispositivo debemos enviar las notificaciones, puesto que el tokenId es único para cada usuario junto con su dispositivo.

12.2.4. Ejemplo de CF Automatic Trigger

```
exports.decideToDeleteTopicSubscription =
functions.firestore.document('subscriptions/{subscription}').onUpdate((
change, context) => {

    const subscription = context.params.subscription;
    const newValue = change.after.data();
    const oldValue = change.before.data();
    const oldToken = oldValue?.fcmTokens;
    const newToken = newValue?.fcmTokens;
    const oldUid = oldValue?.subscribersUid;
    const newUid = newValue?.subscribersUid;

     if (oldToken.length == 1 && oldUid.length == 1 && newToken.length
== 0 && newUid.length == 0) {
        return db.doc(`subscriptions/${subscription}`).delete()
.then(() => {
            return db.doc(`topics/${subscription}`).delete()
.then(() => {
                console.log('Topic & subscription were deleted');
            })
        })
    } else {
          console.log(oldToken.length, oldUid.length, newToken.length,
newUid.length)
        return;
    }})
```

Existen distintos tipos de *triggers*. Principalment *.onCreate()*, *.onUpdate()*, *.onWrite()* y *.onDelete()*.

En esta CF al tratarse de onUpdate() podemos analizar los cambios que se producen antes y después de la modificación pudiendo comprobar cuál ha sido el cambio producido en dicho documento y actuar en consecuencia.

Hay que ir con sumo cuidado con este tipo de funciones pues al no disponer de un accionador directo, podríamos generar un bucle dentro de nuestras CF de forma que pese a disponer de un número elevado de CF de forma gratuïta el hecho de crear un bucle podría llegar a generar una facturación desorbitada. Algo muy desagradable y a tener muy presente en el desarrollo de este proyecto.

12.2.5. Ejemplo de Código *ChangeNotifier*

A continuación se muestra un ejemplo del primer ChangeNotifier que implementamos en el proyecto. Su funcionalidad es gestionar la navegación principal de la App.

```
class NavegacionModel with ChangeNotifier {
  int _paginaActual = 0;
  PageController _pageController = new PageController();

  int get paginaActual => this._paginaActual;
  set paginaActual(int valor) {
    this._paginaActual = valor;
    _pageController.animateToPage(valor,
        duration: Duration(milliseconds: 450),
        curve: Curves.fastLinearToSlowEaseIn);
    notifyListeners();
  }
  PageController get pageController => this._pageController;
}
```

Como podemos ver la classe NavegacionModel extiende ChangeNotifier lo que implica que incluye las características y hereda sus propiedades.

12.2.6. Ejemplo de Código *ChangeNotifierProvider*

```
return MultiProvider(

      providers: [
        ChangeNotifierProvider(create: (_) => HomeService()),
        ChangeNotifierProvider(create: (_) => SearchProvider()),
        ChangeNotifierProvider(create: (_) => ProfileServices()),
        ChangeNotifierProvider(create: (_) => TopicServices()),
        Provider<ImagePickerService>(
          create: (_) => ImagePickerService(),
        ),
      ]);
```

12.2.7. Ejemplo de Código *Consumer*

```
class _CurvedNavegacion extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    final navegacionModel = Provider.of<NavegacionModel>(context);

    return CurvedNavigationBar(
      color: Colors.blue.withOpacity(0.8),
      backgroundColor: Colors.grey.withOpacity(0.4),
      items: <Widget>[
        Icon(Icons.flight, size: 25, color: Colors.white),
        Icon(Icons.search, size: 25, color: Colors.white),
        Icon(Icons.notifications, size: 25, color: Colors.white),
        Icon(Icons.person_outline, size: 25, color: Colors.white)
      ],
      index: navegacionModel.paginaActual,
      onTap: (i) => navegacionModel.paginaActual = i,
    );}}
```

Si atendemos al ejemplo mostrado previamente del *ChangeNotifier*, se puede observar cómo hacemos la llamada al modelo de la clase NavegacionModel y saber en qué página se encuentra el usuario y cúal está solicitando para poder mostrarla.

12.2.8. Ejemplo de Código setState

```
CarouselSlider(
                items: imageSliders,
                options: CarouselOptions(
                    autoPlay: true,
                    enlargeCenterPage: true,
                    aspectRatio: 24 / 9,
                    onPageChanged: (index, reason) {
                      setState(() {
                        _current = index;
                      });
                    }),
              );
```

12.2.9. Ejemplo de Código Provider

```dart
import 'package:flights/src/models/dbuser_model.dart';
import 'package:flights/src/services/auth_service.dart';
import 'package:flutter/material.dart';

class ProfileServices with ChangeNotifier {
  int _bodySelected = 0;
  DbUser _dbUser;
  User _userUid;

  String _userName;
  String _nickName;
  String _frase;
  String _userWeb;
  String _phoneNumber;

  int _menuSelected = 0;

  int get bodySelected => this._bodySelected;

  set bodySelected(int valor) {
    this._bodySelected = valor;
    notifyListeners();
  }
  DbUser get dbUser => this._dbUser;

  set dbUser(DbUser valor) {
    this._dbUser = valor;
    notifyListeners();
  }
  int get menuSelected => this._menuSelected;

  set menuSelected(int valor) {
    this._menuSelected = valor;
    notifyListeners();
  }

  User get userUid => this._userUid;
  set userUid(User valor) {
    this._userUid = valor;
    notifyListeners();
  }

  String get userName => this._userName;

  set userName(String valor) {
    this._userName = valor;

    notifyListeners();
  }

  String get nickName => this._nickName;

  set nickName(String valor) {
    this._nickName = valor;

    notifyListeners();
  }

  String get frase => this._frase;

  set frase(String valor) {
    this._frase = valor;
    notifyListeners();
  }

  String get userWeb => this._userWeb;

  set userWeb(String valor) {
    this._userWeb = valor;
    notifyListeners();
  }
  String get phoneNumber => this._phoneNumber;
  set phoneNumber(String valor) {
    this._phoneNumber = valor;
    notifyListeners();
  }
}
```

12.2.10. Ejemplo de Código *SubscriptionsRegister*

```
return db.doc(`users/${uid}`).update({
  subscriptionsRegister:admin.firestore.FieldValue.arrayUnion({
                      orig: req.query.orig.toUpperCase(),
                      dest: req.query.dest.toUpperCase(),
                      price: parseFloat(req.query.price),
                      dayInici: parseInt(req.query.dayInici),
                      monthInici: parseInt(req.query.monthInici),
                      yearInici: parseInt(req.query.yearInici),
                      dayFinal: parseInt(req.query.dayFinal),
                      monthFinal: parseInt(req.query.monthFinal),
                      yearFinal: parseInt(req.query.yearFinal)
             })})
```

12.2.11. Ejemplo de Código *onMessage*

```
onMessage: (info) async {
      dynamic argumento = 'no-data';
      if (Platform.isAndroid) {
        argumento = info['notification']['body'] ?? 'no-data';
      } else {
        argumento = info['aps']['alert']['body'] ?? 'no-data-iOS';
      }
      final didRequestSeeNotification = await PlatformAlertDialog(
        title: 'New Notification', cancelActionText: 'Cancel',
        content: argumento, defaultActionText: 'See',
      ).show(context);
      if (didRequestSeeNotification == true) {
        _mensajesStreamController.sink.add(argumento);
      }
    }
```

12.2.12. Ejemplo de Código *onResume*

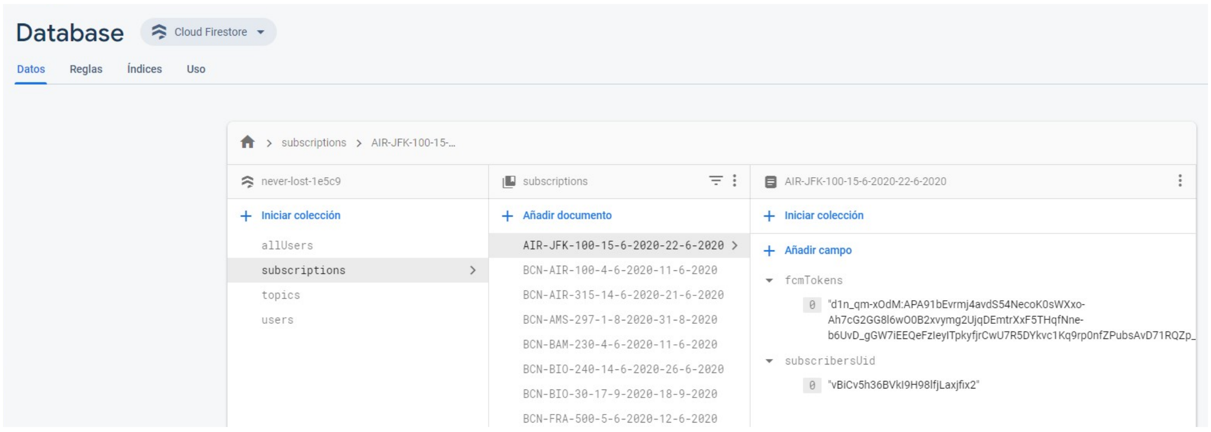
```
onResume: (info) async {
      String argumento = 'no-data';
      if (Platform.isAndroid) {
        argumento = info['notification']['body'] ?? 'no-data';
      } else {
        argumento = info['aps']['alert']['body'] ?? 'no-data-iOS';
      }
   _mensajesStreamController.sink.add(argumento);
}
```

12.2.13. Ejemplo de Código *onLaunch*

```
onLaunch: (info) async {
      _mensajesStreamController.sink.add(info['data']);
    }
```

12.3. Firestore

12.3.1. Subscriptions



Database · Cloud Firestore

Datos · Reglas · Índices · Uso

subscriptions > AIR-JFK-100-15-...

never-lost-1e5c9

+ Iniciar colección

```
allUsers
subscriptions
topics
users
```

subscriptions

+ Añadir documento

```
AIR-JFK-100-15-6-2020-22-6-2020
BCN-AIR-100-4-6-2020-11-6-2020
BCN-AIR-315-14-6-2020-21-6-2020
BCN-AMS-297-1-8-2020-31-8-2020
BCN-BAM-230-4-6-2020-11-6-2020
BCN-BIO-240-14-6-2020-26-6-2020
BCN-BIO-30-17-9-2020-18-9-2020
BCN-FRA-500-5-6-2020-12-6-2020
```

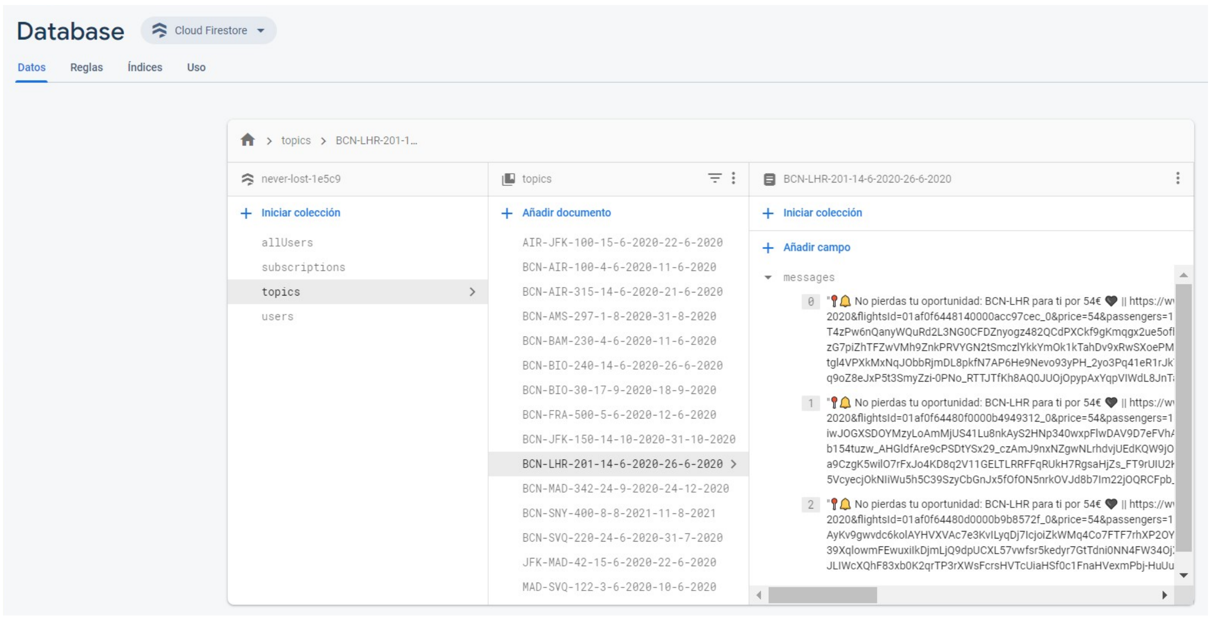
AIR-JFK-100-15-6-2020-22-6-2020

+ Iniciar colección

+ Añadir campo

```
fcmTokens
  0 "d1n_qm-xOdM:APA91bEvrmj4avdS54NecoK0sWXxo-
    Ah7cG2GG8l6wO0B2xvymg2UjqDEmtrXxF5THqfNne-
    b6UvD_gGW7iEEQeFzIeyITpkyfjrCwU7R5DYkvc1Kq9rp0nfZPubsAvD71RQZp_
subscribersUid
  0 "vBiCv5h36BVkI9H98lfjLaxjfix2"
```

12.3.2. Topics



Database · Cloud Firestore

Datos · Reglas · Índices · Uso

topics > BCN-LHR-201-1...

never-lost-1e5c9

+ Iniciar colección

```
allUsers
subscriptions
topics
users
```

topics

+ Añadir documento

```
AIR-JFK-100-15-6-2020-22-6-2020
BCN-AIR-100-4-6-2020-11-6-2020
BCN-AIR-315-14-6-2020-21-6-2020
BCN-AMS-297-1-8-2020-31-8-2020
BCN-BAM-230-4-6-2020-11-6-2020
BCN-BIO-240-14-6-2020-26-6-2020
BCN-BIO-30-17-9-2020-18-9-2020
BCN-FRA-500-5-6-2020-12-6-2020
BCN-JFK-150-14-10-2020-31-10-2020
BCN-LHR-201-14-6-2020-26-6-2020
BCN-MAD-342-24-9-2020-24-12-2020
BCN-SNY-400-8-8-2021-11-8-2021
BCN-SVQ-220-24-6-2020-31-7-2020
JFK-MAD-42-15-6-2020-22-6-2020
MAD-SVQ-122-3-6-2020-10-6-2020
```

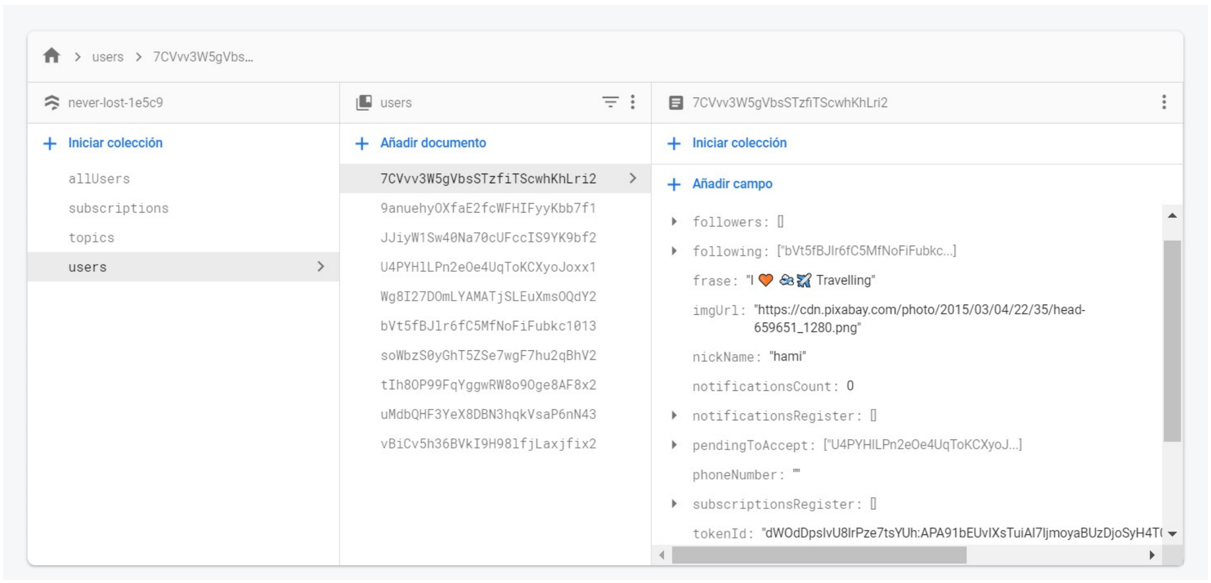
BCN-LHR-201-14-6-2020-26-6-2020

+ Iniciar colección

+ Añadir campo

```
messages
  0 "📍🔔 No pierdas tu oportunidad: BCN-LHR para ti por 54€ 🖤 || https://w
    2020&flightsId=01af0f6448140000acc97cec_0&price=54&passengers=1
    T4zPw6nQanyWQuRd2L3NG0CFDZnyogz482QCdPXCkf9gKmqgx2ue5ofl
    zG7piZhTFZwVMh9ZnkPRVYGN2tSmczlYkkYmOk1kTahDv9xRwSXoePM
    tgl4VPXkMxNqJObbRjmDL8pkfN7AP6He9Nevo93yPH_2yo3Pq41eR1rJk
    q9oZ8eJxP5t3SmyZzi-0PNo_RTTJTfKh8AQ0JUOjOpypAxYqpVIWdL8JnT
  1 "📍🔔 No pierdas tu oportunidad: BCN-LHR para ti por 54€ 🖤 || https://w
    2020&flightsId=01af0f64480f0000b4949312_0&price=54&passengers=1
    iwJOGXSDOYMzyLoAmMjUS41Lu8nkAyS2HNp340wxpFlwDAV9D7eFVh
    b154tuzw_AHGldfAre9cPSDtYSx29_czAmJ9nxNZgwNLrhdvjUEdKQW9jO
    a9CzgK5wilO7rFxJo4KD8q2V11GELTLRRFFqRUkH7RgsaHjZs_FT9rUIU2
    5VcyecjOkNIiWu5h5C39SzyCbGnJx5fOfON5nrkOVJd8b7Im22jOQRCFpb_
  2 "📍🔔 No pierdas tu oportunidad: BCN-LHR para ti por 54€ 🖤 || https://w
    2020&flightsId=01af0f64480d0000b9b8572f_0&price=54&passengers=1
    AyKv9gwvdc6kolAYHVXVAc7e3KvILyqDj7IcjoiZkWMq4Co7FTF7rhXP2OY
    39XqlowmFEwuxiIkDjmLjQ9dpUCXL57vwfsr5kedyr7GtTdni0NN4FW34Oj
    JLIWcXQhF83xb0K2qrTP3rXWsFcrsHVTcUiaHSf0c1FnaHVexmPbj-HuUu
```

12.3.3. Users



users > 7CVvv3W5gVbs...

never-lost-1e5c9

+ Iniciar colección

```
allUsers
subscriptions
topics
users
```

users

+ Añadir documento

```
7CVvv3W5gVbsSTzfiTScwhKhLri2
9anuehyOXfaE2fcWFHIFyyKbb7f1
JJiyW1Sw40Na70cUFccIS9YK9bf2
U4PYHlLPn2eOe4UqToKCXyoJoxx1
Wg8I27DOmLYAMATjSLEuXms0QdY2
bVt5fBJlr6fC5MfNoFiFubkc1013
soWbzS0yGhT5ZSe7wgF7hu2qBhV2
tIh8OP99FqYggwRW8o90ge8AF8x2
uMdbQHF3YeX8DBN3hqkVsaP6nN43
vBiCv5h36BVkI9H98lfjLaxjfix2
```

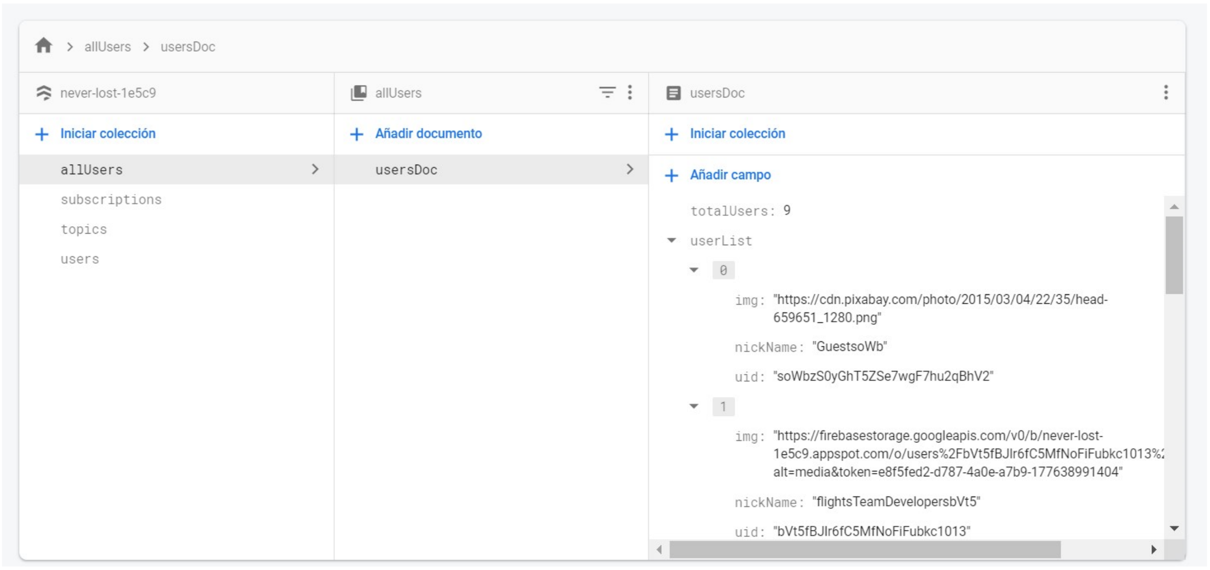
7CVvv3W5gVbsSTzfiTScwhKhLri2

+ Iniciar colección

+ Añadir campo

```
followers: []
following: ["bVt5fBJlr6fC5MfNoFiFubkc...]
frase: "I 🧡 🚭🗺 Travelling"
imgUrl: "https://cdn.pixabay.com/photo/2015/03/04/22/35/head-
         659651_1280.png"
nickName: "hami"
notificationsCount: 0
notificationsRegister: []
pendingToAccept: ["U4PYHlLPn2eOe4UqToKCXyoJ...]
phoneNumber: ""
subscriptionsRegister: []
tokenId: "dWOdDpslvU8lrPze7tsYUh:APA91bEUvlXsTuiAl7ljmoyaBUzDjoSyH4T
```

12.3.4. allUsers



```
totalUsers: 9
userList
    0
        img: "https://cdn.pixabay.com/photo/2015/03/04/22/35/head-659651_1280.png"
        nickName: "GuestsoWb"
        uid: "soWbzS0yGhT5ZSe7wgF7hu2qBhV2"
    1
        img: "https://firebasestorage.googleapis.com/v0/b/never-lost-1e5c9.appspot.com/o/users%2FbVt5fBJlr6fC5MfNoFiFubkc1013%2
             alt=media&token=e8f5fed2-d787-4a0e-a7b9-177638991404"
        nickName: "flightsTeamDevelopersbVt5"
        uid: "bVt5fBJlr6fC5MfNoFiFubkc1013"
```

12.3.5. Ejemplo Subscriptions Document

```
{
    "fcmTokens": [
        "cWF3sWkgnUo:APA91bH0-8",
        "QKw3NcyJnUo:BsdwaWeQb-",
        "lshvijfih-CETMcm@3rtxe"
    ],
    "subscribersUid": [
        "JJiyW1Sw40Nl70cUFccIS9YK9bf2",
        "vBiCv5h36BVkI0H98lfjLaxjfix2",
        "U4PYHlLPn2eOe4UqToKCXpoJoxx1"
    ]}
```

12.3.6. Ejemplo Topics Document

```
{
    "messages": [
        "📍🔔 No pierdas tu oportunidad: BCN-LHR para ti por 54€ 🖤 ||
https://www.kiwi.com/deep?from=BCN&to=LHR",
        "📍🔔 No pierdas tu oportunidad: BCN-LHR para ti por 49€ 🖤 ||
https://www.kiwi.com/deep?from=BCN&to=LHR" ],    "msg_count": 2 }
```

12.3.7. Ejemplo Users Document

```
{
    "backgroundColor": "Colors.blue",
    "followers": [
        "Wg8I27DOmLYAMATLSLEuXmsOQdY2",
        "U4PjHlLPn2eOe4UqToKCXyoJoxx1"
    ],
    "following": [
        "I27DOmLYAMU4PjHlL90dasmJDDII"
    ],
    "frase": "I ❤️ exploring 🌍",
    "imgUrl": "https://cdn.pixabay.com/photohead-659651_1280.png",
    "nickName": "Real Testers 👨🏻‍💻👨🏾‍💻",
    "notificationsCount": 2,
    "notificationsRegister": [
        {
            "deep_link": " https://www.kiwi.com/deep?from=BCN&to=LHR",
            "msg": "📍🔔 No pierdas: BCN-LHR para ti por 54€ 🖤 "
        },
        {
            "deep_link": " https://www.kiwi.com/deep?from=BCN&to=LHR",
            "msg": "📍🔔 No pierdas: BCN-LHR para ti por 49€ 🖤 "
        }
    ],
    "pendingToAccept": [
        "bVt5fBJlr6ZC5MfNoFiFubkc1013"
    ],
    "phoneNumber": "+34 643219044",
    "subscriptionsRegister": [
        {
            "dayFinal": 26,
            "dayInici": 14,
            "dest": "LHR",
            "monthFinal": 6,
            "monthInici": 6,
            "orig": "BCN",
            "price": 201
        },
        {
            "dayFinal": 12,
            "dayInici": 1,
            "dest": "MAD",
            "monthFinal": 10,
            "monthInici": 8,
            "orig": "BCN",
            "price": 35
        }
    ],
    "tokenId": "cwU4pNsXQsU:APA91bH6NXLyLWt",
    "uid": "JJi8I27DOmLYAMATjSEuXmsQdY2",
    "userEmail": "developer_testing@mail.com",
    "userName": "UFly Team",
    "userWeb": "www.uflyNow.com"
}
```

12.3.8. Ejemplo allUsers Document

```
{
    "totalUsers": 3,
    "userList": [
        {
            "img": "https://cdn.pixabay.com/photohead-659651_1280.png",
            "nickName": "GuestsoWb",
            "uid": "soWbzS0yGhT5ZSe7wgF7hu4qBhV2"
        },
        {
            "img": "https://cdn.pixabay.com/photohead-659651_1280.png",
            "nickName": "flightsTeamDevelopers",
            "uid": "bVt5fBJlr6fC5MfNoFiFubkc10163"
        },
        {
            "img": "https://cdn.pixabay.com/photohead-659651_1280.png",
            "nickName": "Testing Nick Names",
            "uid": "tIh8OP99FqYggwRz8o9Oge8AF8x2"
        }
    ]
}
```

12.4. Cloud Functions

12.4.1. Modelo User

```
export class User {
    img: string;
    nickName: string;
    uid: string;

    constructor(img: string, nickName: string, uid: string) {
        this.img = img;
        this.nickName = nickName;
        this.uid = uid;

    }}
```

12.4.2. Index.ts

12.4.2.1. Imports

```
import * as functions from 'firebase-functions';
import * as admin from 'firebase-admin';

const  now = require("performance-now")
const request = require("request");
const {Storage} = require('@google-cloud/storage');
const storage = new Storage();

//myImports
import {AviationStack2} from './AviationStack2';
import {Airport} from './Airport';
import {ArrivalTimeTable} from './ArrivalTimeTable';
import {Airlines} from './Airlines';
import {Score, ScoreData} from './ScoreData';
import {TripAdvisorPollResp} from './TripAdvisorPollResp';
import {User} from './UserList';
import {KiwiResponse} from './kiwiResponse';
import {KiwiTicket} from './kiwiTickets';
//end myImports
export class Convert {
    public static toAviationStack2(json: string): AviationStack2 {
        return JSON.parse(json);
    }}
export class Converted {
    public static toAirlines(json: string): Airlines {
        return JSON.parse(json);
    }}
export class Convertion {
    public static toArrivalTimeTable(json: string): ArrivalTimeTable {
        return JSON.parse(json);
    }}
export class Converting {
    public static toAirport(json: string): Airport {
        return JSON.parse(json);
    }}
export class Conversion {
 public static toScore(json: string): Score[] {
     return JSON.parse(json);
 }
 public static scoreToJson(value: Score[]): string {
       return JSON.stringify(value);
   }}
export class ConvertTransf {
    public static toTripAdvisorPollResp(json: string):
TripAdvisorPollResp {
        return JSON.parse(json);
    }}
export class Convertir {
    public static toKiwiResponse(json: string): KiwiResponse {
        return JSON.parse(json);
    }}

admin.initializeApp();
const db = admin.firestore();
const fcm = admin.messaging();
const access_key = 'Our private API Key';
```

12.4.2.2. onUserWelcome

```
// AUTOTRIGGER: Cuando un usuario se registra por primera vez
exports.onUserWelcome = functions.auth.user().onCreate(async (user) =>
{
  let userName4Display:string;
  let userName:string;
  let userUID = user.uid;
  if(user.displayName !== null){
    if(user.displayName !== undefined){
    userName = user.displayName;
    userName4Display = userName;
    }else{
      userName4Display = 'Guest';
    }}else{
      userName4Display = 'Guest';
    }
    userUID.split('');
    userName = `${userName4Display.replace(/
/g,"")}${userUID[0]}${userUID[1]}${userUID[2]}${userUID[3]}`;
  return db.doc(`users/${user.uid}`).create({
    uid: userUID,
    userEmail: user.email,
    userName: user.displayName == null? 'Guest': user.displayName,
    nickName: `${userName4Display.replace(/
/g,"")}${userUID[0]}${userUID[1]}${userUID[2]}${userUID[3]}`,
    imgUrl: user.photoURL == null?
'https://cdn.pixabay.com/photo/2015/03/04/22/35/head-659651_1280.png':
user.photoURL,
    frase: "I 🧡 🛫✈️ Travelling",
    backgroundColor: "Colors.blue",
    tokenId: "12345678909876543mytokenidpendentdeagadar",
    following: [],
    followers: [],
    pendingToAccept:[],
    userWeb: "",
    phoneNumber: "",
    notificationsCount: 0,
    notificationsRegister: [],
    subscriptionsRegister: []
  }).then(()=>{
    if(user.email !== null){
    return db.doc('allUsers/usersDoc').set({
      totalUsers : admin.firestore.FieldValue.increment(+1),
      userList : admin.firestore.FieldValue.arrayUnion({
        uid: user.uid,
          img: user.photoURL == null?
'https://cdn.pixabay.com/photo/2015/03/04/22/35/head-659651_1280.png':
user.photoURL,
        nickName: userName
      })
    }, {merge:true});
  }else{
    console.log('Anonimous users like u dont fit in our
UserSystemList');
    return;
  }
  }).catch(err=>{
      console.log(err)
    }) })
```

12.4.2.3. onUserGoodbye

```
// AUTOTRIGGER: Cuando un usuario es eliminado
exports.onUserGoodbye = functions.auth.user().onDelete((user,
context)=>{
  const uid = user.uid;
  let followers:[];
  let following:[];
  let subscriptions:[];
  let tokenId:string;
  let imgUrl:string;
  let subscripcio:string;

  return db.doc(`users/${uid}`).get().then((userDoc)=>{
    const data = userDoc.data();
    followers = data?.followers;
    following = data?.following;
    subscriptions = data?.subscriptionsRegister;
    tokenId = data?.tokenId;
    imgUrl = data?.imgUrl;
  }).then(()=>{
    return db.doc(`users/${user.uid}`).delete().then(()=>{
      return db.doc('allUsers/usersDoc').get().then((list)=>{
        let img;
        let nickName;
        let found = false;
        const userList = list.data()?.userList;
        for (let index = 0; index < userList.length; index++) {
            if(userList[index].uid === uid){
               img = userList[index].img;
               nickName = userList[index].nickName;
              found = true;
            }}
        if(found == true){
          console.log(`DELETING USER: ${uid}`);
          return db.doc('allUsers/usersDoc').update({
            totalUsers: admin.firestore.FieldValue.increment(-1),
            userList: admin.firestore.FieldValue.arrayRemove({
              "img":img,
              "nickName":nickName,
              "uid":uid,
            }) })
        }else{
          return;
        }
      }).then(()=>{
        followers.forEach((follower)=>{
          return db.doc(`users/${follower}`).update({
            following: admin.firestore.FieldValue.arrayRemove(uid)
          })
        })
        following.forEach((followed)=>{
          return db.doc(`users/${followed}`).update({
            followers: admin.firestore.FieldValue.arrayRemove(uid)
          })
          })
          }).then(()=>{
            console.log(subscriptions);
            if(subscriptions !== undefined){
            subscriptions.forEach((subscripcioElement:any)=>{
             subscripcio =
`${subscripcioElement.orig}-${subscripcioElement.dest}-${subscripcioEle
ment.price}-${subscripcioElement.dayInici}-${subscripcioElement.monthIn
ici}-${subscripcioElement.yearInici}-${subscripcioElement.dayFinal}-${s
ubscripcioElement.monthFinal}-${subscripcioElement.yearFinal}`;
             console.log(subscripcio);
              return db.doc(`subscriptions/${subscripcio}`).update({
                fcmTokens:
admin.firestore.FieldValue.arrayRemove(tokenId),
                subscribersUid:
admin.firestore.FieldValue.arrayRemove(uid)
              })
            })
          }else{
              return;
            }
          }).then(()=>{
            if(imgUrl !==
'https://cdn.pixabay.com/photo/2015/03/04/22/35/head-659651_1280.png'){
            const path = `users/${uid}/avatar.png`;

            const bucket = storage.bucket('Project-root');
            const file = bucket.file(path)
            return file.delete().then(()=>{
              console.log('Everything was fine 🤭🙈');
              return;
            })
          }else{
            return;
          }})
        })
    }).catch(console.error);
})
```

12.4.2.4. onAvatarChange

```
// AUTOTRIGGER: Cuando un usuario cambia su Imágen de Perfil
exports.onAvatarChanged = functions.storage.object().onFinalize((event)
=>{
  const object = event;
  const filePath = object.name;
  let userUid = filePath?.split('/')[1];
  console.log(`userUID 1st parse:   ${userUid}`);
  userUid = userUid?.split('/')[0];
  console.log(`userUID 2nd parse:   ${userUid}`);

  return db.doc(`users/${userUid}`).get().then((doc)=>{
    const docData = doc.data();
    const imgURL = docData?.imgUrl;
    console.log(`this is the new imgURL --> ${imgURL}`);
    return db.doc('allUsers/usersDoc').get().then((userDoc)=>{
      const usersData = userDoc.data();
      const userList = usersData?.userList;
      let userNickName:string='';
      let oldImg;
      for (let index = 0; index < userList.length; index++) {
        if(userList[index].uid === userUid){
           userNickName = userList[index].nickName;
           oldImg = userList[index].img;
        }}
      return db.doc('allUsers/usersDoc').update({
        userList : admin.firestore.FieldValue.arrayRemove({
          "img":oldImg,
          "nickName": userNickName,
          "uid":userUid,
        })}).then(()=>{
        console.log('USER IS GOING TO BE SAVED WITH NEW IMGURL ((:')
        return db.doc('allUsers/usersDoc').update({
          userList: admin.firestore.FieldValue.arrayUnion({
            "img":imgURL,
            "nickName": userNickName,
            "uid":userUid,
          })})})})}).catch(console.error) })
```

12.4.2.5. searchUser

```
//ONCALL: Cuando se busca un usuario

exports.searchUser = functions.https.onCall(async (data, context) => {

  const userToFind = data.userToFind;
  const uid = context.auth?.uid;

  return db.doc('allUsers/usersDoc').get().then((allUsersDocu)=>{

    const allUsers = allUsersDocu.data()?.userList;
    let usersMatching: User[] = new Array();

    for (let index = 0; index < allUsers.length; index++) {

if(allUsers[index].nickName.toString().toUpperCase().includes(userToFin
d.toUpperCase())){
        if(allUsers[index].uid == uid){
          console.log('Users profile, no need to display')
        }else{
        usersMatching.push({img:allUsers[index].img,
nickName:allUsers[index].nickName ,uid: allUsers[index].uid});
        console.log(`${allUsers[index].nickName} inserted to the
ARRAY`);
      }}
      console.log(`Evaluating ${allUsers[index].nickName}`)
    }
    return usersMatching;

  })
  .catch(console.error);
})
```

12.4.2.6. getUserProfileData

```
//ONCALL: Cuando se ha encontrado un usuario y se quiere obtener su
información
exports.getUserProfileData = functions.https.onCall(async (data,
context) => {
  const reqUser = data.uid;
  const getUid = data.getUid;

  return db.doc(`users/${getUid}`).get().then((userInfo)=>{
    const userData = userInfo.data();
    for (let index = 0; index < userData?.followers.length; index++) {
      if(userData?.followers[index] == reqUser){
        return {
          nickName: userData?.nickName,
          imgUrl: userData?.imgUrl,
          backgroundColor: userData?.backgroundColor,
          frase: userData?.frase,
          followers: userData?.followers,
          following: userData?.following,
          friendOrNot: "true"
        };
      }}
    for (let index = 0; index < userData?.pendingToAccept.length;
index++) {
      if(userData?.pendingToAccept[index] == reqUser){
        return {
          nickName: userData?.nickName,
          imgUrl: userData?.imgUrl,
          backgroundColor: userData?.backgroundColor,
          frase: userData?.frase,
          followers: userData?.followers,
          following: userData?.following,
          friendOrNot: "pendent"
        };
      }}
      return {
        nickName: userData?.nickName,
        imgUrl: userData?.imgUrl,
        backgroundColor: userData?.backgroundColor,
        frase: userData?.frase,
        followers: userData?.followers,
        following: userData?.following,
        friendOrNot: "false"  };})})
```

12.4.2.7. getUserInfo

```
//ONCALL: Cuando se quiere cargar la información del usuario

exports.getUserInfo = functions.https.onCall(async (data,
context:any)=>{

  const reqUser = data.uid.toString();
  const userId = context.auth.uid.toString();
  //NEED TO STORE THE TOKEN!
  //security lvl.1 checking
  if(reqUser === userId){
    try{
      return db.doc(`users/${userId}`).get()
      .then((doc)=>{
        console.log(doc.data());
        return doc.data();
      });

    }catch(e){
      console.log(e);
      return 'Error in the CLOUD FUNC :()'
    }
    //userTryingToFake a req UID
}else{
  return 'GLMF';
  }
})
```

12.4.2.8. updateUserInfo

```
//ONCALL: Cuando un usuario desea actualizar su información de la BD
exports.updateUserInfo = functions.https.onCall(async (data,
context:any)=> {
  const userId = context.auth.uid.toString();
  const reqUser = data.uid.toString(); //ON REQUEST: req.query.uid;
  const nickName = data.nickName;       //ON REQUEST:
req.query.nickName;
  const userName = data.userName;//req.query.userName;
  const frase = data.frase;//req.query.frase;
  const phoneNumber = data.phoneNumber;//req.query.phoneNumber;
  const userWeb = data.userWeb;//req.query.userWeb;
  let notFound = true; //si el trobo aleshores --> FALSE
  console.log(userId, reqUser, nickName, userWeb, userName,
phoneNumber, frase);

  if(userId === reqUser){
    return new Promise((resolve, reject)=>{
      return db.doc('allUsers/usersDoc').get()
     .then((usersDoc) => {
       const usersList =  usersDoc.data()?.userList;
       for (let i = 0; i < usersList.length; i++) {
           if(usersList[i].nickName === nickName && usersList[i].uid
!== reqUser){
             console.log(`${usersList[i].nickName} == ${nickName} (?)`)
             notFound = false;
           }
            console.log(i);
       }
       if(notFound == true){
         console.log('not Found == TRUE');
         return notFound;
       }else{
         return notFound;
       }
     }).then(async(found)=>{
      console.log(found);
      if(found === true){
        console.log('going toChangeList');
        const p1 = await changeFromList(reqUser, nickName);
        console.log('going toChangeUser');
        const p2 = await changeFromUser(reqUser, nickName);
        return Promise.all([p1,p2]).then(async()=>{
          console.log(userWeb, userName, phoneNumber, frase);
          await db.doc(`users/${reqUser}`).update({
            userWeb: userWeb.toString(),
            userName: userName.toString(),
            phoneNumber: phoneNumber.toString(),
            frase: frase.toString()
          })
     .then(()=>{
            console.log('TOT A ANAT PERFECTE');
            resolve(true);
            });
        })
      }else{
        console.log('EUREKA arribo aqui (:');
        resolve(false);
      }});
    }).then((val)=>{
      console.log(val);
      return val;
    })
}else{
  return 'GLMF';
}})

  async function changeFromList(uid:string, newNickName:string){
    let img:any;
    let oldNickName:string;
    return db.doc('allUsers/usersDoc').get().then((list)=>{

      const mylist = list.data()?.userList;
      for (let index = 0; index < mylist.length; index++) {
        if(mylist[index].uid == uid){
          img = mylist[index].img;
        oldNickName = mylist[index].nickName;        }}

      return db.doc('allUsers/usersDoc').update({
        userList : admin.firestore.FieldValue.arrayRemove({
          "img":img,
          "nickName": oldNickName,
          "uid":uid,
        })
      }).then(()=>{
        return db.doc('allUsers/usersDoc').update({
          userList: admin.firestore.FieldValue.arrayUnion({
            "img":img,
            "nickName": newNickName == null? oldNickName : newNickName,
            "uid":uid,
          })
        })})
    });
  }
  async function changeFromUser(uid:string, newNickName:string){
    if(newNickName !== null){
    return db.doc(`users/${uid}`).update({
      nickName: newNickName
    });
  }else{
      return;
    }}
```

12.4.2.9. sendDeviceToken

```
//ONCALL: Establecer el Token Id del dispositivo en la BD

exports.sendDeviceToken = functions.https.onCall(async (data, context)
=> {
  const uid = context.auth?.uid;
  return db.doc(`users/${uid}`).update({
    tokenId: data.tokenId
  }).then(()=>{
    console.log('Done');
    return true;
  }).catch(console.error);
})
```

12.4.2.10. askForFriendship

```
//ONCALL: Cuando un usuario solicita una petición de amistad a otro

exports.askForFriendship = functions.https.onCall(async (data, context)
=> {
  const uid = data.uid;
  const suposedFriend =  data.futureFriendUid;
//req.query.futureFriendUid;
  return db.doc(`users/${suposedFriend}`).update({
    pendingToAccept: admin.firestore.FieldValue.arrayUnion(uid)
  }).then(()=>{
    console.log('Petition submitted');
    return true;
  }).catch(console.error);
})
```

12.4.2.11. cancelAskForFriendship

```
//ONCALL: Cuando un usuario desea anular una amistad

exports.cancelAskForFriendship = functions.https.onCall(async (data,
context) => {

  const uid =  data.uid;       //req.query.uid;
  const cancelFriend =  data.cancelFriend;    //req.query.cancelFriend;

  return db.doc(`users/${cancelFriend}`).update({
    pendingToAccept: admin.firestore.FieldValue.arrayRemove(uid)
  }).then(()=>{
    console.log(`${cancelFriend} will no be able to accept your petiton
for a friendship`);
    return true;
  }).catch(console.error);
})
```

12.4.2.12. acceptFriend

```
//ONCALL: Cuando se acepta una solicitud de seguimiento
exports.acceptFriend = functions.https.onCall(async (data, context) =>
{
  const uid = data.uid;                           //req.query.uid;
  const friendAccepted = data.newFriend;

  return db.doc(`users/${uid}`).update({
    pendingToAccept:
admin.firestore.FieldValue.arrayRemove(friendAccepted),
    followers: admin.firestore.FieldValue.arrayUnion(friendAccepted)
  }).then(()=>{
    return db.doc(`users/${friendAccepted}`).update({
      following: admin.firestore.FieldValue.arrayUnion(uid)
    }).then(()=>{
      console.log(`Now you allowed ${friendAccepted} to be your friend,
you are: ${uid}`);
      return true;
    })
  }).catch(console.error);
})
```

12.4.2.13. declineFriend

```
//ONCALL: Cuando se quiere denegar una petición de seguimiento

export const declineFriend = functions.https.onCall(async (data,
context) => {
  const uid = data.uid;            //req.query.uid;
  const noFriend =  data.noFriend;        // req.query.noFriend;

  return db.doc(`users/${uid}`).update({
    pendingToAccept: admin.firestore.FieldValue.arrayRemove(noFriend)
  }).then(()=>{
    console.log(`${noFriend} is like shit! I dont want it as
friend...`);
    return true;
  }).catch(console.error);})
```

12.4.2.14. deleteFriend

```
//ONCALL: Cuando un usuario quiere dejar de seguir a otro (Unfollow)

export const deleteFriend = functions.https.onCall(async (data,
context) => {

  const uid =  data.uid;                    //req.query.uid;
  const noMoreFriend = data.noMoreFriend; //req.query.noMoreFriend;

  return db.doc(`users/${uid}`).update({
    following: admin.firestore.FieldValue.arrayRemove(noMoreFriend),
  }).then(()=>{
    console.log(`Ya no te sigo ${noMoreFriend}`);
    return db.doc(`users/${noMoreFriend}`).update({
      followers: admin.firestore.FieldValue.arrayRemove(uid)
    }).then(()=>{
      console.log('-1 AMIGO :((');
      return true;
    })
  }).catch(console.error);
})
```

12.4.2.15. getUsersPendingToAccept

```
//ONCALL: Conseguir la lista de usuarios pendientes de acceptar

export const getUsersPendingToAccept = functions.https.onCall(async
(data, context) => {

  const uid =  data.uid;      //req.query.uid;
  let pdtList:[];
  let finalArray:any[] = new Array;

  return db.doc(`users/${uid}`).get().then((doc)=>{
    const docData = doc.data();
     pdtList = docData?.pendingToAccept;
  }).then(()=>{
    return db.doc('allUsers/usersDoc').get().then((usersDoc)=>{
      const usersData = usersDoc.data();
      for (let index = 0; index < pdtList.length; index++) {
        usersData?.userList.forEach((user:any)=>{
          if(pdtList[index] == user.uid){
            finalArray.push({img: user.img, nickName: user.nickName,
uid: user.uid, frase: ''});
          // }else{
          //   console.log(user);
          }
        })
      }
    }).then(()=>{
      console.log(finalArray);
      return finalArray;
    })
  }).catch(console.error);})
```

12.4.2.16. getFlightInfo

```
//ONREQUEST: Cuando se busca un vuelo en función de su ID
exports.getFlightInfo = functions.https.onRequest(async (req, resp)=>{
  let respostaFuncio:any;
  const flightId = req.query.flightId;
  // let vols:number = 0;
  let depCity:any;
  let arrCity:any;
  const optionAviationStack = {
    method: 'GET',
    url:
`http://api.aviationstack.com/v1/flights?access_key=${access_key}&fligh
t_iata=${flightId}`,
  };

  try{
     await request(optionAviationStack, async
function(error:any,algo:any, body:any){
       console.log(body);
      const flightAS:AviationStack2 = Convert.toAviationStack2(body);
      console.log(flightAS);
        respostaFuncio = flightAS.data[0];
        depCity = respostaFuncio.departure.iata;
        arrCity = respostaFuncio.arrival.iata;
        depCity = await getCityName(depCity);
        arrCity = await getCityName(arrCity);

      await Promise.all([depCity, arrCity]).then(()=>{
        return resp.send({flightInfo:respostaFuncio , depCity: depCity,
arrCity: arrCity});
      });
});

}catch(e){
  console.log(e.toString());
}
})

  async function getCityName(iataAirport:string){
   let cityName = iataAirport;
   const optionCity = {
     method: 'GET',
     url: 'https://airport-info.p.rapidapi.com/airport',
     qs: {iata: iataAirport},
     headers: {
       'x-rapidapi-host': 'airport-info.p.rapidapi.com',
       'x-rapidapi-key': 'Rapid_API_KEY'
     }
     };

   return new Promise(function(resolve, reject){
     request.get(optionCity, cityName,
function(error:any,something:any, body:any){
       if(error){
         console.log(error.toString());
         reject(error);
       }else{
         console.log(body);
         const airportInfo:Airport = Converting.toAirport(body);
         const cityInfo = {
           airportName: airportInfo.name,
           latitude: airportInfo.latitude,
           longitude: airportInfo.longitude,
         };

         resolve(cityInfo);
       }
     })
   }) }
```

12.4.2.17. getArrivalTimeTabel

```
//ONREQUEST: Cuando se solicitan las llegadas (ARRIVALS)

exports.getArrivalTimeTable = functions.https.onRequest(async (req,
resp)=>{

  let respostaFuncio:any;
  const airport = req.query.airport;
  const limit = 50;
  const options = {
    method: 'GET',
    url:
`http://api.aviationstack.com/v1/flights?access_key=${access_key}&limit
=${limit}&arr_iata=${airport}&flight_status=active`,    }
  try{
    await request(options, function(error:any,response:any,body:any){
      const airportAS = Convertion.toArrivalTimeTable(body);
      respostaFuncio = airportAS.data;
      return resp.send({arrivalInfo:respostaFuncio});
    })
  }catch(e){
    console.log(e.toString());
  }
})
```

12.4.2.18. getDepartureTimeTable

```
//ONREQUEST: Cuando se solicitan las salidas (DEPARTURES)

exports.getDepartureTimeTable = functions.https.onRequest(async (req,
resp)=>{

  let respostaFuncio:any;
  const airport = req.query.airport;
  const limit = 50;
  const options = {
    method: 'GET',
    url:
`http://api.aviationstack.com/v1/flights?access_key=${access_key}&limit
=${limit}&dep_iata=${airport}&flight_status=active`,
  }
  try{
    await request(options, function(error:any,response:any,body:any){
      const airportAS = Convertion.toArrivalTimeTable(body);
      respostaFuncio = airportAS.data;
      return resp.send({departureInfo:respostaFuncio});
    })
  }catch(e){
    console.log(e.toString());
  }
})
```

12.4.2.19. getAirlineInfo

```
//ONREQUEST: Cuando se solicitan información de una aerolínea

exports.getAirlineInfo = functions.https.onRequest(async (req, resp)=>{

  let airlineSearch:string = req.query.airline;

  const optionAirline = {
      method: 'GET',
      url:
`http://api.aviationstack.com/v1/airlines?access_key=${access_key}`,
      headers: {
      'content-type': 'application/json' }  };
  try{
    await request(optionAirline, function(error:any, notShadow:any,
body:any){
        const airlinesDoc = Converted.toAirlines(body);
        console.log(airlinesDoc);
        const airlinesList = airlinesDoc.data;
        let i = 0;
        let trobat = 0;
        while(i < airlinesList.length && trobat == 0){
          let anal = airlinesList[i].airline_name;
          console.log(anal + ' ' + i);
          if(anal.toLowerCase() == airlineSearch.toLowerCase()){
            console.log(airlinesList[i].toString());
            trobat = 1;
          }else{
            i++;
          }
        }
        if(trobat === 1){
          return resp.send({airlineData: airlinesList[i]});
        }else{
          return resp.send({airlineData: 'NO match found'});
        }})
  }catch(e){
    console.log(e.toString());
  }
})
```

12.4.2.20. getAirportInfo

```
//ONREQUEST: Cuando se solicita información del aeropuerto (info
airport + ARRIVALS + DEPARTURES)
exports.getAirportInfo = functions.https.onRequest(async (req, resp)=>{

  const airport = req.query.airport;
  const optionCity = {
    method: 'GET',
    url: 'https://airport-info.p.rapidapi.com/airport',
    qs: {iata: airport},
    headers: {
      'x-rapidapi-host': 'airport-info.p.rapidapi.com',
      'x-rapidapi-key': 'Rapid_API_KEY'
    }
  };

  await request(optionCity, async function(error:any,something:any,
body:any){
    const airportInfo:Airport = Converting.toAirport(body);
    const myInfo = airportInfo;
    const arrivals = await getAirportArrivals(airport);
    const departures = await getAirportDepartures(airport);

    await Promise.all([arrivals, departures]).then(()=>{
      resp.send({airportInfo: myInfo, airportDepartures: departures,
airportArrivals: arrivals});

    })
  })
})

  async function getAirportDepartures(airport:string){

    const limit = 20;
    const optionAirport = {
        method: 'GET',
        url:
`http://api.aviationstack.com/v1/flights?access_key=${access_key}&limit
=${limit}&dep_iata=${airport}&flight_status=active`,       }

    return new Promise(function(resolve, reject){
      request.get(optionAirport, airport,
function(error:any,something:any, body:any){
        if(error){
          console.log(error.toString());
          reject(error);
        }else{
          const airportInfo:ArrivalTimeTable =
Convertion.toArrivalTimeTable(body);
```

```
          let departures = airportInfo.data;
          console.log('DEPARTURES');
          console.log(departures);
          if(departures === undefined){
            console.log('I got departures === undefined');
            return;
          }else{
          departures.sort(function (a,b){
            let h1 = new Date(a.departure.estimated).getTime();
            let h2 = new Date(b.departure.estimated).getTime();
            if(h1 > h2){
              return 1;}
            if(h1 < h2){
            return -1;}
            return 0;
          });

          resolve(departures);
        }}
      })})}


  async function getAirportArrivals(airport:string){

    const limit = 20;
    const optionAirport = {
        method: 'GET',
        url:
`http://api.aviationstack.com/v1/flights?access_key=${access_key}&limit
=${limit}&arr_iata=${airport}&flight_status=active`,
      }

    return new Promise(function(resolve, reject){
      request.get(optionAirport, airport,
function(error:any,something:any, body:any){
        if(error){
          console.log(error.toString());
          reject(error);
        }else{
          const airportInfo:ArrivalTimeTable =
Convertion.toArrivalTimeTable(body);
          const arrivals = airportInfo.data;
          console.log('ARRIVALS');
          console.log(arrivals);

          if(arrivals === undefined){
            console.log('I got arrivals === undefined');
            return;
          }else{
          arrivals.sort(function (a,b){
            let h1 = new Date(a.arrival.estimated).getTime();
            let h2 = new Date(b.arrival.estimated).getTime();
            if(h1 > h2){
              return 1;}
            if(h1 < h2){
            return -1;}
            return 0;
          });

          resolve(arrivals);
        }}
      })
    })

  }
```

12.4.2.21. getFirebaseAirport

```
exports.getFirebaseAirport = functions.https.onRequest(async
(req,resp)=>{

  const airport = req.query.airport;
  const airportDoc = req.query.firebase;
    try{
      const airDoc =  await
db.collection('AirData').doc(`${airportDoc}`).get();
        if(airDoc.exists){
          const json = airDoc.data();
          if(json!==undefined){
            Object.keys(json).forEach(async(ele)=>{
              if(ele.toLowerCase() === airport.toLowerCase() &&
json!==undefined){
                resp.send({airportData:json[ele]})
              } })
          }
        }else{
          console.log('Doc does not exists');
        }
    }catch(e){
      console.log(e.toString());
      resp.send(e.toString());
    }

});
```

12.4.2.22. getCityData

```
//ONREQUEST: Cuando se solicita información de una ciudad

exports.getCityData = functions.https.onRequest(async (req,resp)=>{
  let t0 = now();
  const cityName = req.query.city;
  const optionCity = {
      method: 'GET',
      url:
`https://api.teleport.org/api/cities/?search=${cityName}&embed=city:search-results/city:item/city:country&embed=city_search-results[0]/city:item`,
      headers: {
        "templated": true
      }};

    try{
      await request(optionCity,  async
function(error:any,something:any, body:any){
       const jsonDoc = JSON.parse(body);
       let t1 = now()
       console.log('Time to get Teleport Data: '+ (t0-t1).toFixed(3));
       const vari = jsonDoc._embedded["city:search-results"];
```

```
        const vari2 = vari[0]._embedded["city:item"];
        const scoreUrl = vari2._links["city:urban_area"].href;

         const scoreData = await getCityScores(scoreUrl);
         const images = await getCityImages(cityName);

       await Promise.all([scoreData, images]).then(()=>{
         let t2 = now()
         console.log('Time to get send resp: ' + (t2-t0).toFixed(3));
         resp.send({
           cityName: vari2.name,
           cityPopulation: vari2.population,
           cityGeoname_id: vari2.geoname_id,
           countryName: vari2._embedded["city:country"].name,
           countryIso3: vari2._embedded["city:country"].iso_alpha3,
           countryPopulation:
vari2._embedded["city:country"].population,
           countryCurrency:
vari2._embedded["city:country"].currency_code,
           countryGeoname_id:
vari2._embedded["city:country"].geoname_id,
           scoreData: scoreData,
           images: images
           })
         })
       })
   }catch(e){
     console.log(e.toString());
   }
})

   async function getCityScores(url:string):Promise<ScoreData>{
     let t3 = now();
     const cityUrl = {
       method: 'GET',
       url: url + 'scores',     }

     return new Promise(function(resolve, reject){
       request.get(cityUrl, function(error:any,something:any, body:any){
         if(error){
           console.log(error.toString());
           reject(error);
         }else{
           const data = JSON.parse(body);
           let summ = data.summary.replace(/<\/?[^>]+(>|$)/g,'');
           summ = summ.replace(/[\n\r]/g,'');
           summ = summ.replace('   ','');
           const jsonCat = Conversion.scoreToJson(data.categories);
           const cat = Conversion.toScore(jsonCat);
           const resolving = {
             scoreInfo : cat,
             summary   : summ
           }
           let t4 = now()
           console.log('Time to resolve City Scores: '+
(t4-t3).toFixed(3));
           resolve(resolving);
         }
       })
     })
   }

   async function getCityImages(cityName:string){
     let t5 = now()
     const imgNum = 7;
     const imgUrl = {
         method: 'GET',
         url:
`https://api.pexels.com/v1/search?query=${cityName}&per_page=${imgNum}&
page=1`,
         headers: {
           'Authorization': 'Authoritzation_String',
         }
       }

     return new Promise(function(resolve, reject){
       request.get(imgUrl, function(error:any,something:any, body:any){
         if(error){
           console.log(error.toString());
           reject(error);
         }else{
           const data = JSON.parse(body);
           let picsArr:string[] = [];
           for (let i = 0; i < data.photos.length; i++) {
             if(i == 6 || data.photos[i] == null || data.photos[i] ==
 undefined){
               break;
             }
             picsArr.push(data.photos[i].src.portrait);
           }
           let t6 = now()
           console.log('Time to resolve Pexel Images: '+
 (t6-t5).toFixed(3));
           resolve(picsArr);
         }
       })
     })
   }
```

12.4.2.23. setNotification

```
//ONREQUEST: Se activa cuando un usuario desea guardar un tópico,
analiza si existe o si debe crear uno nuevo, pues no existe en nuestra
BD

exports.setNotification = functions.https.onRequest(async (req,
resp)=>{

  const uid = req.query.uid;

  const newTopic =
`${req.query.orig.toUpperCase()}-${req.query.dest.toUpperCase()}-${pars
eFloat(req.query.price)}-${req.query.dayInici}-${req.query.monthInici}-
${req.query.yearInici}-${req.query.dayFinal}-${req.query.monthFinal}-${
req.query.yearFinal}`;

  // if(uid == reqUid){

  const userData = await db.doc(`users/${uid}`).get();
  Promise.all([userData]).then(()=>{
    const tokenId = userData.data()?.tokenId;

    return db.doc(`topics/${newTopic}`).get().then((doc)=>{
      if (!doc.exists) {
       console.log('No topics, so lets CREATE IT!');
       return db.doc(`topics/${newTopic}`).create({
         messages: [],
         msg_count: 0
       }).then(()=>{
         return db.doc(`subscriptions/${newTopic}`).create({
           subscribersUid: [uid],
           fcmTokens: [tokenId]
         }).then(()=>{
           return db.doc(`users/${uid}`).update({
             subscriptionsRegister:
admin.firestore.FieldValue.arrayUnion({
               orig: req.query.orig.toUpperCase(),
               dest: req.query.dest.toUpperCase(),
               price: parseFloat(req.query.price),
               dayInici: pfarseInt(req.query.dayInici),
               monthInici: parseInt(req.query.monthInici),
               yearInici: parseInt(req.query.yearInici),
               dayFinal: parseInt(req.query.dayFinal),
               monthFinal: parseInt(req.query.monthFinal),
               yearFinal: parseInt(req.query.yearFinal)
             })
           }).then(()=>{
              resp.send('TOPIC CREATED, USER SUBSCRIBE IT & user has
recorded (:');
           })
         })
       })

      } else {
       console.log('We found the document, so lets SUBSCRIBE to
TOPIC');
       return db.doc(`subscriptions/${newTopic}`).update({
         subscribersUid: admin.firestore.FieldValue.arrayUnion(uid),
         fcmTokens: admin.firestore.FieldValue.arrayUnion(tokenId)
       }).then(()=>{
         return db.doc(`users/${uid}`).update({
           subscriptionsRegister:
admin.firestore.FieldValue.arrayUnion({
             orig: req.query.orig.toUpperCase(),
             dest: req.query.dest.toUpperCase(),
             price: parseFloat(req.query.price),
             dayInici: req.query.dayInici,
             monthInici: req.query.monthInici,
             yearInici: req.query.yearInici,
             dayFinal: req.query.dayFinal,
             monthFinal: req.query.monthFinal,
             yearFinal: req.query.yearFinal
           })
         }).then(()=>{
           resp.send('USER WAS SUBSCRIBED, TOPIC WAS ALREADY CREATED
(:')
         })
       })

     }
    });

  }).catch(console.error);
})
```

12.4.2.24. deleteNotification

```
//ONREQUEST: Se encarga de eliminar un tópico, la suscripción y el
registro (user) correspondiente

exports.deleteNotification = functions.https.onRequest(async (req,
resp) => {
  const uid = req.query.uid;
  const topic =
`${req.query.org.toUpperCase()}-${req.query.dest.toUpperCase()}-${parse
Float(req.query.price)}-${req.query.dayInici}-${req.query.monthInici}-$
```

```
{req.query.yearInici}-${req.query.dayFinal}-${req.query.monthFinal}-${r
eq.query.yearFinal}`
  let userToken:string;
  let elementToDelete = {
    dayFinal: parseInt(req.query.dayFinal),
    dayInici: parseInt(req.query.dayInici),
    dest: req.query.dest.toUpperCase(),
    monthFinal: parseInt(req.query.monthFinal),
    monthInici: parseInt(req.query.monthInici),
    orig: req.query.org.toUpperCase(),
    price: parseFloat(req.query.price),
    yearFinal: parseInt(req.query.yearFinal),
    yearInici: parseInt(req.query.yearInici),
  }
  console.log(elementToDelete);
  return db.doc(`users/${uid}`).get().then((user)=>{
    const userData = user.data();
    userToken = userData?.tokenId;
    console.log(userToken);
  }).then(()=>{
    return db.doc(`subscriptions/${topic}`).update({
      fcmTokens: admin.firestore.FieldValue.arrayRemove(userToken),
      subscribersUid: admin.firestore.FieldValue.arrayRemove(uid)
    }).then(()=>{
      return db.doc(`subscriptions/${topic}`).get().then((topicData)=>{
        const topicInfo = topicData.data();
        if(topicInfo?.fcmTokens.length === 0 &&
topicInfo.subscribersUid.length === 0){
          return db.doc(`topics/${topic}`).delete().then(()=>{
            return db.doc(`subscriptions/${topic}`).delete().then(()=>{
              return db.doc(`users/${uid}`).get().then((userData)=>{
                const userInfo = userData.data();
                const myTopics = userInfo?.subscriptionsRegister;
                console.log('MYTOPICS:', myTopics)
                for (let index = 0; index < myTopics.length; index++) {
                  console.log(myTopics[index].dayFinal,
parseInt(req.query.dayFinal), myTopics[index].dayInici,
parseInt(req.query.dayInici) , myTopics[index].dest,
req.query.dest.toUpperCase(),
myTopics[index].monthFinal,parseInt(req.query.monthFinal),
myTopics[index].monthInici,parseInt(req.query.monthInici),
myTopics[index].orig,req.query.org.toUpperCase(),parseFloat(myTopics[in
dex].price),parseFloat(req.query.price),
myTopics[index].yearFinal,parseInt(req.query.yearFinal),
myTopics[index].yearInici,parseInt(req.query.yearInici))
                  if(myTopics[index].dayFinal ==
parseInt(req.query.dayFinal) && myTopics[index].dayInici==
parseInt(req.query.dayInici) && myTopics[index].dest ==
req.query.dest.toUpperCase() &&
myTopics[index].monthFinal==parseInt(req.query.monthFinal) &&
myTopics[index].monthInici==parseInt(req.query.monthInici) &&
myTopics[index].orig==req.query.org.toUpperCase() &&
parseFloat(myTopics[index].price) == parseFloat(req.query.price) &&
myTopics[index].yearFinal==parseInt(req.query.yearFinal) &&
myTopics[index].yearInici==parseInt(req.query.yearInici)){
                    console.log('IM IN THE RIGHT ITERATION (:)');
                    return db.doc(`users/${uid}`).update({
                      subscriptionsRegister:
admin.firestore.FieldValue.arrayRemove(myTopics[index])
                    }).then(()=>{
                      resp.send('User subscription was deleted ;)');
                    })
                  }
                }
                return;
              })})})
        }else{
           console.log('Still having subscriptions');
           return db.doc(`users/${uid}`).get().then((userData)=>{
             const userInfo = userData.data();
             const myTopics = userInfo?.subscriptionsRegister;
             console.log('MYTOPICS:', myTopics)
             for (let index = 0; index < myTopics.length; index++) {
               console.log(myTopics[index].dayFinal,
parseInt(req.query.dayFinal), myTopics[index].dayInici,
parseInt(req.query.dayInici) , myTopics[index].dest,
req.query.dest.toUpperCase(),
myTopics[index].monthFinal,parseInt(req.query.monthFinal),
myTopics[index].monthInici,parseInt(req.query.monthInici),
myTopics[index].orig,req.query.org.toUpperCase(),parseFloat(myTopics[in
dex].price),parseFloat(req.query.price),
myTopics[index].yearFinal,parseInt(req.query.yearFinal),
myTopics[index].yearInici,parseInt(req.query.yearInici))
               if(myTopics[index].dayFinal ==
parseInt(req.query.dayFinal) && myTopics[index].dayInici==
parseInt(req.query.dayInici) && myTopics[index].dest ==
req.query.dest.toUpperCase() &&
myTopics[index].monthFinal==parseInt(req.query.monthFinal) &&
myTopics[index].monthInici==parseInt(req.query.monthInici) &&
myTopics[index].orig==req.query.org.toUpperCase() &&
parseFloat(myTopics[index].price) == parseFloat(req.query.price) &&
myTopics[index].yearFinal==parseInt(req.query.yearFinal) &&
myTopics[index].yearInici==parseInt(req.query.yearInici)){
                 console.log('IM IN THE RIGHT ITERATION (:)');
                 return db.doc(`users/${uid}`).update({
                   subscriptionsRegister:
admin.firestore.FieldValue.arrayRemove(myTopics[index])
                 }).then(()=>{
                   resp.send('User subscription was deleted ;)');
                 })
               }else{console.log('sorry')}
             }
             return;
           })}})
      })
  }).catch(console.error)
})
```

12.4.2.25. deliverTopicMessage

```
//AUTOTRIGGER: Se encarga de analizar si se ha añadido un nuevo MSG en
el tópico, de ser así procede al envío de las notificaciones

exports.deliverTopicMessage =
functions.firestore.document('topics/{topicName}').onUpdate((change,
context)=>{
  const topic = context.params.topicName;
  const newValue = change.after.data();
  const oldValue = change.before.data();
```

```
    const message = newValue?.messages;
    const oldCounter = oldValue?.msg_count;
    const newCounter = newValue?.msg_count;
    let uidList:[];
    if(newCounter > oldCounter){
    const messageRecived = message.pop();
    const arrMsg = messageRecived.split('||');
    const messageToSend = arrMsg[0];
    const link = arrMsg[1];
    console.log(arrMsg);

    const payload : admin.messaging.MessagingPayload ={
      notification: {
        title: 'Your Trip',
        body: messageToSend,
        icon:
'https://cdn.clipart.email/9d52b5a36143d2fd830cce1f67bf82bf_universal-h
oliday-travel-route-world-travel-vacation-travel-png-_650-651.jpeg',
        clickAction: 'FLUTTER_NOTIFICATION_CLICK'
      }
    }

    return db.doc(`subscriptions/${topic}`).get().then((data)=>{
      const docData = data.data();
      uidList  = docData?.subscribersUid;
      const tokenList = docData?.fcmTokens;
      for (let index = 0; index < tokenList.length; index++) {
        console.log(tokenList[index]);
      }
      console.log(payload);
      return fcm.sendToDevice(tokenList,payload);
    }).then(()=>{
      console.log('EY DONT WORRY 2MUCH')
      uidList.forEach((uid)=>{
        return db.doc(`users/${uid}`).update({
          notificationsCount: admin.firestore.FieldValue.increment(+1),
          notificationsRegister: admin.firestore.FieldValue.arrayUnion({
            msg: messageToSend,
            deep_link: link
          })
        })
      })
    })
  }else{
      console.log('A message was deleted so no msg is send through
  FCMessaging (:');
    return;
  }
  })
```

12.4.2.26. decideToDeleteTopicSubscribtion

```
//AUTOTRIGER: Evalua si es necesario o no eliminar un topico en función
de sus subscripciones (creada para el soporte técnico de los
desarrolladores)

exports.decideToDeleteTopicSubscription =
functions.firestore.document('subscriptions/{subscription}').onUpdate((
change, context)=>{

  const subscription = context.params.subscription;
  const newValue = change.after.data();
  const oldValue = change.before.data();
  const oldToken = oldValue?.fcmTokens;
  const newToken = newValue?.fcmTokens;
  const oldUid = oldValue?.subscribersUid;
  const newUid = newValue?.subscribersUid;

  if( .length == 1 && oldUid.length == 1 && newToken.length == 0 &&
newUid.length == 0 ){
    return db.doc(`subscriptions/${subscription}`).delete().then(()=>{
      return db.doc(`topics/${subscription}`).delete().then(()=>{
        console.log('Topic & subscription were delteted');
      })
    })
  }else{
    console.log(oldToken.length, oldUid.length, newToken.length,
newUid.length)
    return;
  }})
```

12.4.2.27. getKiwiFlightTickets

```
//ONREQUEST: Permite encontrar un vuelo en función de los parámetros de
búsqueda

exports.getKiwiFlightTickets = functions.https.onRequest(async (req,
resp) => {

  const origen = req.query.orig.toUpperCase();
  const destino = req.query.dest.toUpperCase();
  const date =`${req.query.day}/${req.query.month}/${req.query.year}`;
//'15/08/2020';
  console.log(date);
  const currency = 'EUR';
  const flight_type = 'oneway'; // 'round' --> with nights_in_dst of
return date is given
  const direct_flights = 1; // 1 = YES, 0 = NO
  const adults = 1;
  const children = 0;
  const infants = 0;

  const kiwiOption = {
      method: 'GET',
      url:
`https://api.skypicker.com/flights?fly_from=${origen}&fly_to=${destino}
&date_from=${date}&partner=picky&curr=${currency}&direct_flights=${dire
ct_flights}&flight_type=${flight_type}&adults=${adults}&children=${chil
dren}&infants=${infants}`,
      headers: {
      'content-type': 'application/json'
    }};

  let kTicket:KiwiTicket;
  let kTickets:KiwiTicket[] = new Array();
  let lowPrice:number[];

    try{
      await request(kiwiOption, async function(error:any,
something:any, body:any){
          const kiwiResponse = Convertir.toKiwiResponse(body);
          const myData = kiwiResponse.data;

          if(myData !== undefined){
          for (let index = 0; index < myData.length; index++) {
            if(myData[index].availability?.seats !== null){
            let flightId
=`${myData[index].airlines[0]}${myData[index].route[0].flight_no}`
            kTicket = new KiwiTicket(myData[index].airlines[0],
flightId, myData[index].cityCodeFrom, myData[index].cityCodeTo,
myData[index].cityFrom, myData[index].cityTo,
myData[index].countryFrom.code, myData[index].countryTo.code, new
Date(myData[index].dTime * 1000), new Date(myData[index].aTime * 1000),
date, myData[index].fly_duration, myData[index].distance,
myData[index].price, myData[index].availability.seats,
myData[index].booking_token, myData[index].deep_link);
            kTickets.push({airline: kTicket.airline, flightId:
kTicket.flightId, from: kTicket.from, to: kTicket.to, from_city:
kTicket.from_city, to_city: kTicket.to_city, country_from:
kTicket.country_from, country_to: kTicket.country_to, depTime:
kTicket.depTime, arrTime: kTicket.arrTime, date: kTicket.date,
duration: kTicket.duration, distance: kTicket.distance, price:
kTicket.price, availability: kTicket.availability, bookingToken:
kTicket.bookingToken, deep_link: kTicket.deep_link});
            }}

            lowPrice = getLowestPrice(kTickets);
            console.log(lowPrice);
            await publishMessage(origen, destino, lowPrice[0],
req.query.day, req.query.month, req.query.year,
kTickets[lowPrice[1]].deep_link).then(()=>{
              console.log(kTickets)
              resp.send(kTickets);
            })
        }else{
          console.log('my data is equal to UNDEFINED');
          return;
        }
```

```
})
}catch(e){
  console.log(e.toString());
}})

    function getLowestPrice(ticketInfo:KiwiTicket[]){

      let array:number[] = [];
      let position:number = 0;
      let response:number[]= [];
      console.log(ticketInfo);
      for (let index = 0; index < ticketInfo.length; index++) {
          array.push(ticketInfo[index].price);
        }

      let min = array[0];

      for(let i = 0; i < array.length; i++){
        if(array[i] < min){
          min = array[i];
          position = i;
          console.log(position);
        }
      }
      response.push(min,position);
      console.log(response);
      return response;
    }

    async function publishMessage(orig:string, dest:string,
price:number, day:number, month:number, year:number, url:string){
//day:number, month:number, year:number
      //date te format == ( DD/MM/YYYY )
      console.log(url)
      console.log('url');

      await db.collection('topics').get().then((docs)=>{
        if(!docs.empty){
          docs.forEach((doc)=>{
            let topic = doc.id;
            console.log(topic);
            let arrTopic = topic.split('-');
             let or = arrTopic[0];
             let des = arrTopic[1];
             let preu = parseFloat(arrTopic[2]);
             const start = new Date(parseInt(arrTopic[5]),
parseInt(arrTopic[4]), parseInt(arrTopic[3]));
             const startEpoch = start.getTime()/1000.0;
             const end = new Date(parseInt(arrTopic[8]),
parseInt(arrTopic[7]), parseInt(arrTopic[6]));
             const endEpoch = end.getTime()/1000.0;
             const anal = new Date(year, month, day);
             const analEpoch = anal.getTime()/1000.0;
             // console.log(or, des, preu, url);
             if(or === orig && des === dest && preu >= price &&
analEpoch >= startEpoch && analEpoch <= endEpoch){
               console.log(topic,or, orig,des, dest, preu)
               return db.doc(`topics/${topic}`).update({
                 messages: admin.firestore.FieldValue.arrayUnion(`📍🔔
No pierdas tu oportunidad: ${orig}-${dest} para ti por ${price}€ 🖤 ||
${url}`),
                 msg_count: admin.firestore.FieldValue.increment(+1)
               }).then(()=>{
                 console.log('messageDelivered!');
                 return;
               })
             }
             return;
          })
        }else{
          return;
        }})
      }
```

12.4.3. searchUser JSON *response*

```
      {
            "img": "https://cdn.pixabay.com/photohead-659651_1280.png",
            "nickName": "GuestsoWb",
            "uid": "soWbzS0yGhT5ZSe7wgF7hu4qBhV2"
```

```
    }
```

12.5. Modelos Dart

12.5.1. Modelo dbUser

```
class DbUser {
    String uid;
    String userEmail;
    String userName;
    String nickName;
    String imgUrl;
    String frase;
    String backgroundColor;
    String tokenId;
    List<String> following;
    List<String> followers;
    List<String> pendingToAccept;
    String userWeb;
    String phoneNumber;
    int notificationsCount;
    List<NotificationsRegister> notificationsRegister;
    List<SubscriptionsRegister> subscriptionsRegister;

    DbUser({
        this.uid,
        this.userEmail,
        this.userName,
        this.nickName,
        this.imgUrl,
        this.frase,
        this.backgroundColor,
        this.tokenId,
        this.following,
        this.followers,
        this.pendingToAccept,
        this.userWeb,
        this.phoneNumber,
        this.notificationsCount,
        this.notificationsRegister,
        this.subscriptionsRegister,
    });

    DbUser copyWith({
        String uid,
        String userEmail,
        String userName,
        String nickName,
        String imgUrl,
        String frase,
        String backgroundColor,
        String tokenId,
        List<String> following,
        List<String> followers,
        List<String> pendingToAccept,
        String userWeb,
        String phoneNumber,
        int notificationsCount,
        List<NotificationsRegister> notificationsRegister,
        List<SubscriptionsRegister> subscriptionsRegister,
    }) =>
        DbUser(
            uid: uid ?? this.uid,
            userEmail: userEmail ?? this.userEmail,
            userName: userName ?? this.userName,
            nickName: nickName ?? this.nickName,
            imgUrl: imgUrl ?? this.imgUrl,
            frase: frase ?? this.frase,
            backgroundColor: backgroundColor ?? this.backgroundColor,
            tokenId: tokenId ?? this.tokenId,
            following: following ?? this.following,
            followers: followers ?? this.followers,
            pendingToAccept: pendingToAccept ?? this.pendingToAccept,
            userWeb: userWeb ?? this.userWeb,
            phoneNumber: phoneNumber ?? this.phoneNumber,
            notificationsCount: notificationsCount ??
this.notificationsCount,
            notificationsRegister: notificationsRegister ??
this.notificationsRegister,
            subscriptionsRegister: subscriptionsRegister ??
this.subscriptionsRegister,
        );

    factory DbUser.fromJson(Map<String, dynamic> json) => DbUser(
        uid: json["uid"] == null ? null : json["uid"],
        userEmail: json["userEmail"] == null ? null :
json["userEmail"],
        userName: json["userName"] == null ? null : json["userName"],
        nickName: json["nickName"] == null ? null : json["nickName"],
        imgUrl: json["imgUrl"] == null ? null : json["imgUrl"],
        frase: json["frase"] == null ? null : json["frase"],
        backgroundColor: json["backgroundColor"] == null ? null :
json["backgroundColor"],
        tokenId: json["tokenId"] == null ? null : json["tokenId"],
        following: json["following"] == null ? null :
List<String>.from(json["following"].map((x) => x)),
        followers: json["followers"] == null ? null :
List<String>.from(json["followers"].map((x) => x)),
        pendingToAccept: json["pendingToAccept"] == null ? null :
List<String>.from(json["pendingToAccept"].map((x) => x)),
        userWeb: json["userWeb"] == null ? null : json["userWeb"],
        phoneNumber: json["phoneNumber"] == null ? null :
json["phoneNumber"],
        notificationsCount: json["notificationsCount"] == null ? null :
json["notificationsCount"],
        notificationsRegister: json["notificationsRegister"] == null ?
null :
List<NotificationsRegister>.from(json["notificationsRegister"].map((x)
=> NotificationsRegister.fromJson(x))),
        subscriptionsRegister: json["subscriptionsRegister"] == null ?
null :
List<SubscriptionsRegister>.from(json["subscriptionsRegister"].map((x)
=> SubscriptionsRegister.fromJson(x))),
    );

    Map<String, dynamic> toJson() => {
        "uid": uid == null ? null : uid,
        "userEmail": userEmail == null ? null : userEmail,
        "userName": userName == null ? null : userName,
        "nickName": nickName == null ? null : nickName,
        "imgUrl": imgUrl == null ? null : imgUrl,
        "frase": frase == null ? null : frase,
        "backgroundColor": backgroundColor == null ? null :
backgroundColor,
        "tokenId": tokenId == null ? null : tokenId,
        "following": following == null ? null :
List<dynamic>.from(following.map((x) => x)),
        "followers": followers == null ? null :
List<dynamic>.from(followers.map((x) => x)),
        "pendingToAccept": pendingToAccept == null ? null :
List<dynamic>.from(pendingToAccept.map((x) => x)),
        "userWeb": userWeb == null ? null : userWeb,
        "phoneNumber": phoneNumber == null ? null : phoneNumber,
        "notificationsCount": notificationsCount == null ? null :
notificationsCount,
        "notificationsRegister": notificationsRegister == null ? null :
List<dynamic>.from(notificationsRegister.map((x) => x.toJson())),
        "subscriptionsRegister": subscriptionsRegister == null ? null :
List<dynamic>.from(subscriptionsRegister.map((x) => x.toJson())),
    };
}
```

```
class NotificationsRegister {
    String msg;
    String deep_link;

    NotificationsRegister({
        this.msg,
        this.deep_link,
    });

    NotificationsRegister copyWith({
        String msg,
        String deep_link,
    }) =>
        NotificationsRegister(
            msg: msg ?? this.msg,
            deep_link: deep_link ?? this.deep_link,
        );

    factory NotificationsRegister.fromJson(Map<String, dynamic> json)
=> NotificationsRegister(
        msg: json["msg"] == null ? null : json["msg"],
        deep_link: json["deep_link"] == null ? null :
json["deep_link"],
    );

    Map<String, dynamic> toJson() => {
        "msg": msg == null ? null : msg,
        "deep_link": deep_link == null ? null : deep_link,
    };
}

class SubscriptionsRegister {
    String orig;
    String dest;
    int price;
    int dayInici;
    int monthInici;
    int yearInici;
    int dayFinal;
    int monthFinal;
    int yearFinal;

    SubscriptionsRegister({
        this.orig,
        this.dest,
        this.price,
        this.dayInici,
        this.monthInici,
        this.yearInici,
        this.dayFinal,
        this.monthFinal,
        this.yearFinal,
    });

    SubscriptionsRegister copyWith({
        String orig,
        String dest,
        int price,
        int dayInici,
        int monthInici,
        int yearInici,
        int dayFinal,
        int monthFinal,
        int yearFinal,
    }) =>
        SubscriptionsRegister(
            orig: orig ?? this.orig,
            dest: dest ?? this.dest,
            price: price ?? this.price,
            dayInici: dayInici ?? this.dayInici,
            monthInici: monthInici ?? this.monthInici,
            yearInici: yearInici ?? this.yearInici,
            dayFinal: dayFinal ?? this.dayFinal,
            monthFinal: monthFinal ?? this.monthFinal,
            yearFinal: yearFinal ?? this.yearFinal,
        );

    factory SubscriptionsRegister.fromJson(Map<String, dynamic> json)
=> SubscriptionsRegister(
        orig: json["orig"] == null ? null : json["orig"],
        dest: json["dest"] == null ? null : json["dest"],
        price: json["price"] == null ? null : json["price"],
        dayInici: json["dayInici"] == null ? null : json["dayInici"],
        monthInici: json["monthInici"] == null ? null :
json["monthInici"],
        yearInici: json["yearInici"] == null ? null :
json["yearInici"],
        dayFinal: json["dayFinal"] == null ? null : json["dayFinal"],
        monthFinal: json["monthFinal"] == null ? null :
json["monthFinal"],
        yearFinal: json["yearFinal"] == null ? null :
json["yearFinal"],
    );

    Map<String, dynamic> toJson() => {
        "orig": orig == null ? null : orig,
        "dest": dest == null ? null : dest,
        "price": price == null ? null : price,
        "dayInici": dayInici == null ? null : dayInici,
        "monthInici": monthInici == null ? null : monthInici,
        "yearInici": yearInici == null ? null : yearInici,
        "dayFinal": dayFinal == null ? null : dayFinal,
        "monthFinal": monthFinal == null ? null : monthFinal,
        "yearFinal": yearFinal == null ? null : yearFinal,
    };
}}
```

12.5.2. Modelo GoSoon

```
import 'package:sqflite/sqflite.dart';

class GoSoon {
  int id;
  String name;
  String img;
  GoSoon({this.id, this.name, this.img});

  Map<String, dynamic> toMap() {
    return {"name": name, "img": img};
  }

  GoSoon.fromMap(Map<String, dynamic> map) {
    name = map['name'];
    img = map['img'];
    id = map["id"];
  }}
```

12.5.3. Pendent Bloc

```
import 'package:bloc/bloc.dart';
import 'package:equatable/equatable.dart';

class PendentEvent extends Equatable {
  @override
  //Definimos una clase evento (tipo) --> PendentEvent
  List<Object> get props => [];}
```

```
//A - Evento que solicita la lista Solicitudes Pendientes
class FetchPendent extends PendentEvent {
  final _uid;
  FetchPendent(this._uid);
  List<Object> get props => [_uid];}

//B - Evento que obliga a solicitar de nuevo la lista
class ReFetchPendent extends PendentEvent {
  final _uid;
  ReFetchPendent(this._uid);
  List<Object> get props => [_uid];}

class PendentState extends Equatable {
  @override
  // Definimos una clase estado (tipo) --> PendentEvent
  List<Object> get props => [];}



//A continuación se listan los distintos estados

//1 - La lista esta siendo cargada
class PendentListIsLoading extends PendentState {}

//2 - La lista ha sido cargada, lista para ser usada
class PendentListLoaded extends PendentState {
  final _pendentList;
  PendentListLoaded(this._pendentList);

  List<UsersData> get getPdtList => _pendentList;
  @override
  List<Object> get props => [_pendentList];}

//3 - La lista no ha podido ser cargada
class PendentListNotLoaded extends PendentState {}

//Aquí aparece la lógica
class PendentBloc extends Bloc<PendentEvent, PendentState> {
  final uid;
  PendentBloc({this.uid});
  @override
  // Definimos una clase BLOC que permite generar la relación entre
  // eventos (A-B) y los diferentes estados (1-3)
  PendentState get initialState => PendentListIsLoading();

  @override
  Stream<PendentState> mapEventToState(event) async* {
    // TODO: implement mapEventToState
    if (event is FetchPendent) {
      try {
        List<UsersData> pendentList =
            await DataBase(uid: event._uid).getUsersPendingToAccept();
        //NOTA: getUsersPendingToAccept es una CF que devuelve
        //      la lista de solicitudes de seguimiento
        yield PendentListLoaded(pendentList);
      } catch (_) {
        yield PendentListNotLoaded();
      }

    } else if (event is ReFetchPendent) {
      try {
        List<UsersData> pendentList =
            await DataBase(uid: event._uid).getUsersPendingToAccept();
        yield PendentListLoaded(pendentList);
      } catch (_) {
        yield PendentListNotLoaded();
      }
    }}}
```

12.6. Scripts

12.6.1. fireAirport.ts

```
import {Airport} from './Airport.js';

import {Airports} from './Airports.js'
const request = require('request')
const sortBy = require('lodash.sortby');
import * as _ from "lodash";
const cliProgress = require('cli-progress');
const fs = require('fs');
import {AirportInfo} from './AirportInfo.js';
import {Airporty} from './Airporty.js';


// Converts JSON strings to your object
export class Convert {
    public static toAirporty(json: string): Airporty {
        return JSON.parse(json);
    }}

let array = [];
main();

async function main(){
 const done = await getMyDataForMyList()
 Promise.all([done]).then(()=>{
   console.log(array);
 })
}

async function getMyDataForMyList(){
    let airCount = 0;
    const docuNum = 5;
    console.log('N3D Files processing: (AirportList)')
    await
db.collection('AirData').doc(`airports${docuNum}`).get().then((function
(doc) {
        if (doc.exists) {
            let json = doc.data();
            Object.keys(json).forEach(async (ele)=>{
              // let country =new Airport(json[ele]).country_name;
              let iata = new Airport(json[ele]).iata_code;
               let name =  await printIata(iata);
               console.log(name);
               await Promise.all([name]).then(()=>{
                 array = array.concat({[ele]:{
                   cityName: name,
                   iata: iata,
                   firebase: `airports${docuNum}`
                 }});
                 airCount++;
                 console.log(airCount);
                 if(airCount == 865){
                   console.log(array)
                    let data = JSON.stringify(array);
                    fs.writeFileSync(`airports${docuNum}`,data);
                 }
               })
            })
        } else {
            console.log("No such document!");
        }
    }));}

async function printIata(iata:string){
  const optionCity = {
    method: 'GET',
    url: 'https://airport-info.p.rapidapi.com/airport',
    qs: {iata: iata},
    headers: {
      'x-rapidapi-host': 'airport-info.p.rapidapi.com',
      'x-rapidapi-key': Rapid_API_KEY
    }
  };

   return new Promise(function(resolve, reject){
     let cityInfo:string;
```

```
      try{
        request.get(optionCity, function(error:any,something:any,
body:any){

        if(error){
          console.log(error.toString());
          reject(error);
        }else{
            if(body.startsWith('<',0) !== true){
          const airportInfo:Airporty = Convert.toAirporty(body);
         if(airportInfo.location !== undefined){
           const cityArr = airportInfo.location.split(",");
           cityInfo = cityArr[0];
          // console.log(cityInfo[0]);
        }else if(airportInfo.city!== ""){
           cityInfo = airportInfo.city;
        }else if(airportInfo.county!==""){
           cityInfo = airportInfo.county;
        }else{
           cityInfo = "N/A";
        }
        }else{
            cityInfo = "N/A";
        }

          resolve(cityInfo);
        }
      })

    }catch(e){
      console.log(e.toString())
    }
    })
}
```

12.6.2. Aviation.ts

```
var request = require('request');

import {AviationStack} from './aviationStack';
const access_key = A private ACCES_KEY;

var d = new Date();
var curr_date = d.getDate();
var curr_month = d.getMonth() + 1; //Months are zero based
var curr_year = d.getFullYear();
const date = (curr_year + '-' + curr_month + '-' + curr_date);
const flightId = 'VY1156'

 console.log(date);

class Convert {
    public static toAviationStack(json: string): AviationStack {
        return JSON.parse(json);
    }
}

var optionAviationStack = {
  method: 'GET',
  url:
`http://api.aviationstack.com/v1/flights?access_key=${access_key}
        &flight_date=${date}&flight_iata=${flightId}`,
  headers: {
  }
}

getFlightInfo();

function getFlightInfo(){
  request(optionAviationStack, function(err,response,body){
    if(err) throw Error(err);
    const data = body;
    console.log(body)
  })}
```