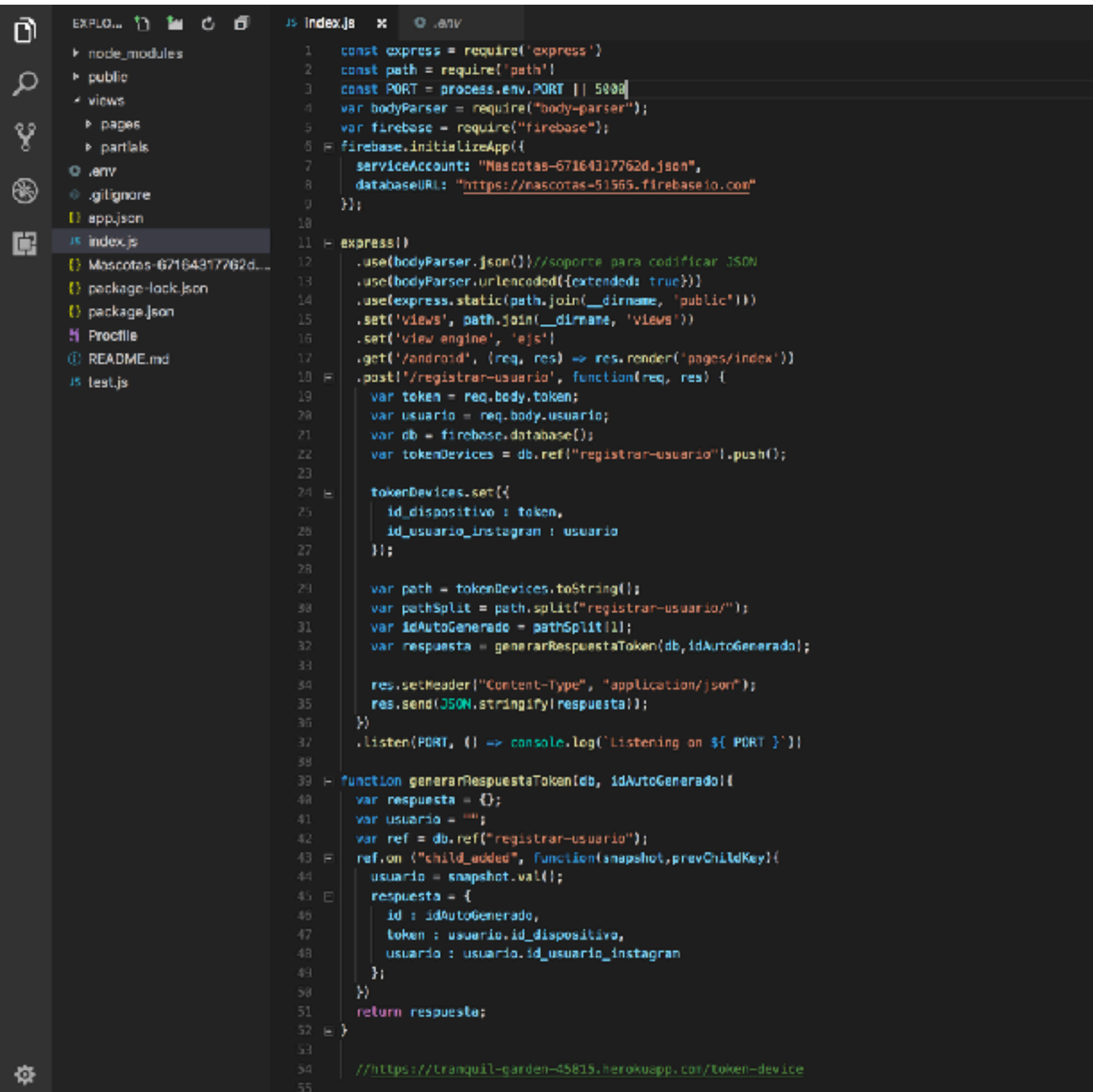


Fernando Rojas Hidalgo
Servicio realizado en Node.js
2018, Costa Rica



The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a project structure with files like `node_modules`, `public`, `views`, `pages`, `partials`, `.env`, `.gitignore`, `app.json`, `index.js`, `Mascotas-67164317762d...`, `package-lock.json`, `package.json`, `Procfile`, `README.md`, and `test.js`. The code editor shows the `index.js` file with the following code:

```
1  const express = require('express')
2  const path = require('path')
3  const PORT = process.env.PORT || 5000
4  var bodyParser = require("body-parser");
5  var firebase = require("firebase");
6  firebase.initializeApp({
7    serviceAccount: "Mascotas-67164317762d.json",
8    databaseURL: "https://mascotas-51565.firebaseio.com"
9  });
10
11  express()
12    .use(bodyParser.json())//soporte para codificar JSON
13    .use(bodyParser.urlencoded({extended: true}))
14    .use(express.static(path.join(__dirname, 'public')))
15    .set('views', path.join(__dirname, 'views'))
16    .set('view engine', 'ejs')
17    .get('/android', (req, res) => res.render('pages/index'))
18    .post('/registrar-usuario', function(req, res) {
19      var token = req.body.token;
20      var usuario = req.body.usuario;
21      var db = firebase.database();
22      var tokenDevices = db.ref("registrar-usuario").push();
23
24      tokenDevices.set({
25        id_dispositivo : token,
26        id_usuario_instagram : usuario
27      });
28
29      var path = tokenDevices.toString();
30      var pathSplit = path.split("registrar-usuario/");
31      var idAutoGenerado = pathSplit[1];
32      var respuesta = generarRespuestaToken(db, idAutoGenerado);
33
34      res.setHeader("Content-Type", "application/json");
35      res.send(JSON.stringify(respuesta));
36    })
37    .listen(PORT, () => console.log('Listening on ${ PORT }'))
38
39  function generarRespuestaToken(db, idAutoGenerado){
40    var respuesta = {};
41    var usuario = "";
42    var ref = db.ref("registrar-usuario");
43    ref.on("child_added", function(snapshot, prevChildKey){
44      usuario = snapshot.val();
45      respuesta = {
46        id : idAutoGenerado,
47        token : usuario.id_dispositivo,
48        usuario : usuario.id_usuario_instagram
49      };
50    })
51    return respuesta;
52  }
53
54  //https://tranquil-garden-45815.herokuapp.com/token-device
55
```

```
MainActivity.java x PreferidosActivity.java x EndpointsApi.java x UsuarioResponse.java x ConstantesRestApi.java x
EndpointsApi registrarTokenID()
1 package net.rohisa.mascotas.restApi;
2
3 import ...
4
5 /**
6  * Created by frojash on 1/13/18.
7  */
8
9 public interface EndpointsApi {
10     @GET(ConstantesRestApi.URL_GET_RECENT_MEDIA_USER)
11     Call<MascotaResponse> getRecentMedia();
12
13     @FormUrlEncoded
14     @POST(ConstantesRestApi.KEY_POST_ID_TOKEN)
15     Call<UsuarioResponse> registrarTokenID(@Field("token") String token, @Field("usuario") String usuario);
16 }
17
18 25
```

```
public EndpointsApi establecerConexionesRestApiIdToken(){
    Retrofit retrofit = new Retrofit.Builder()
        .baseUrl(ConstantesRestApi.ROOT_URL_BASE)
        .addConverterFactory(GsonConverterFactory.create())
        .build();

    return retrofit.create(EndpointsApi.class);
}
```

```
MainActivity.java x PreferidosActivity.java x EndpointsApi.java x RestApiAdapter.java x UsuarioResponse.java x ConstantesRestApi.java x
UsuarioResponse usuario
1 package net.rohisa.mascotas.restApi.model;
2
3 /**
4  * Created by frojash on 2/1/18.
5  */
6
7 public class UsuarioResponse {
8
9     private String id;
10     private String token;
11     private String usuario;
12
13     public UsuarioResponse(String id, String token, String usuario){
14         this.id = id;
15         this.token = token;
16         this.usuario = usuario;
17     }
18
19     public UsuarioResponse(){
20     }
21
22     public String getId(){ return id; }
23
24     public void setId(String id){ this.id = id; }
25
26     public String getToken(){ return token; }
27
28     public void setToken(String token){ this.token = token; }
29
30     public String getUsuario(){ return usuario; }
31
32     public void setUsuario(String usuario){ this.usuario = usuario; }
33 }
34
35 47
```

```

146 private void enviarTokenRegistro(String token, String usuario){
147     RestApiAdapter restApiAdapter = new RestApiAdapter();
148     EndpointsApi endpointsApi = restApiAdapter.establishConexionRestApiIdToken();
149     retrofit2.Call<UsuarioResponse> usuarioResponseCall = endpointsApi.registrarTokenID(token, usuario);
150     usuarioResponseCall.enqueue(new Callback<UsuarioResponse>() {
151         @Override
152         public void onResponse(retrofit2.Call<UsuarioResponse> call, Response<UsuarioResponse> response) {
153             UsuarioResponse usuarioResponse = response.body();
154             Log.d("ID_FIREBASE", usuarioResponse.getId());
155             Log.d("TOKEN_FIREBASE", usuarioResponse.getToken());
156             Log.d("Cuenta_FIREBASE", usuarioResponse.getUserName());
157         }
158     });
159     @Override
160     public void onFailure(retrofit2.Call<UsuarioResponse> call, Throwable t) {
161     }
162 }
163 }
164 }
165 }

```