

Para que todo esto funcione se deben de tener instaladas las herramientas de heroku, npm, el git para manejar el repositorio.

La secuencia de comandos a utilizar para crear el repositorio de Heroku fue el siguiente:

1. `heroku login`
2. `git clone https://github.com/heroku/node-js-getting-started.git`
3. `cd node-js-getting-started`
4. `heroku create`
5. `git push heroku master`
6. `heroku ps:scale web=1`
7. `heroku open`

Cuando se hace un heroku create desde el command prompt, no hay necesidad de crear la aplicación desde la página de heroku. Ahora bien, con este comando se crea un nombre dinámico y no el nombre que usted dese ponerle a la aplicación.

instalar las dependencias siguientes

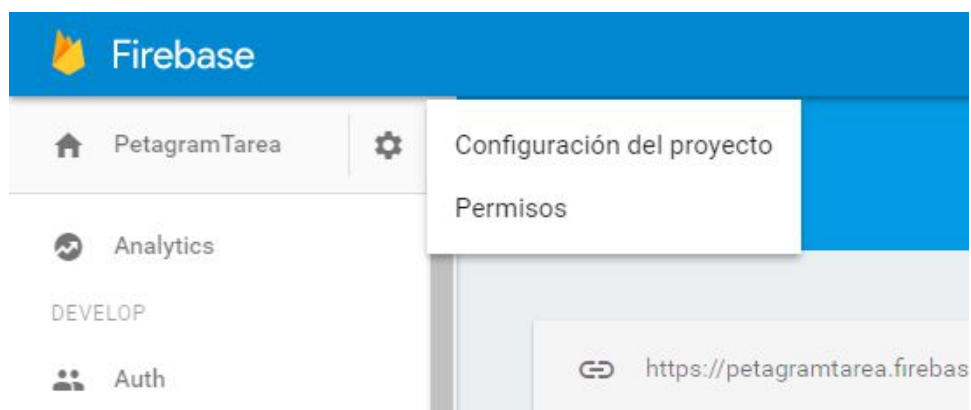
1. `npm install --save --save-exact cool-ascii-faces`
2. `npm install --save --save-exact body-parser\`
3. `npm install --save --save-exact firebase`
4. `npm install`

FIREBASE

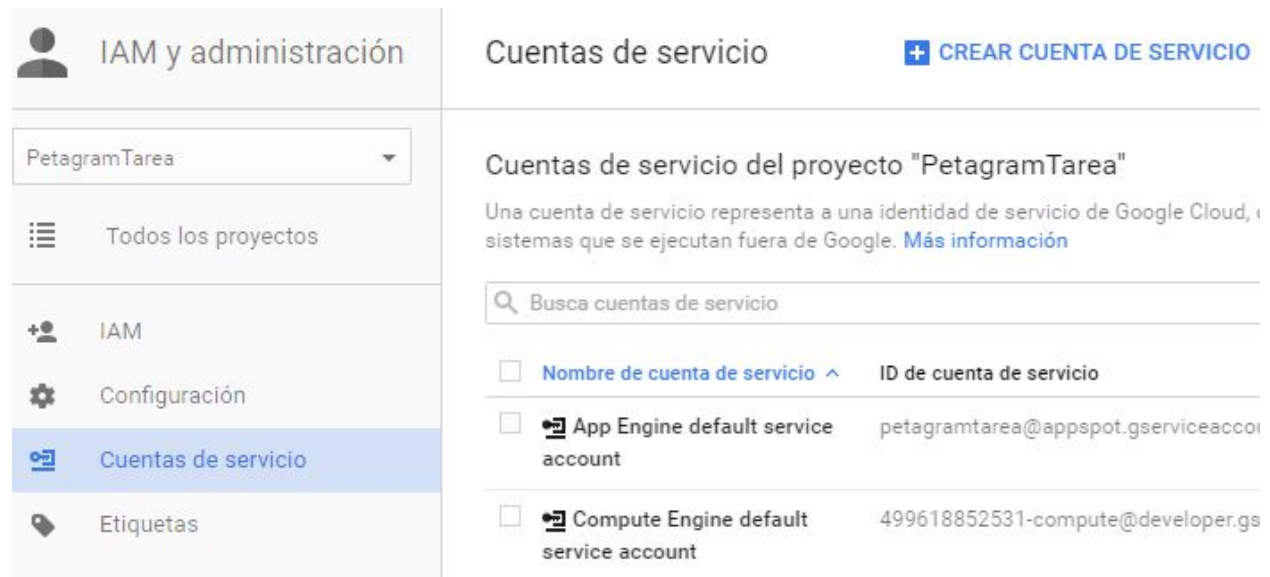
URL: <https://petagramtarea.firebaseio.com/>

Script: PetagramTarea-3e1624ace3b4.json

Es bueno aclarar que para poder obtener el archivo json se debe crear un servicio, para esto se debe ir a la parte izquierda del nombre de la base datos donde aparece una rueda dentada y elegir permisos, una vez ahí debe elegir crear servicio.



Luego elegir Cuentas de Servicio



The screenshot shows the Google Cloud IAM and Administration console. On the left, the 'IAM y administración' sidebar is visible with a dropdown menu set to 'PetagramTarea'. The main content area is titled 'Cuentas de servicio' and includes a '+ CREAR CUENTA DE SERVICIO' button. Below this, it says 'Cuentas de servicio del proyecto "PetagramTarea"' and provides a brief explanation of service accounts. A search bar is present, and a table lists the existing service accounts:

<input type="checkbox"/>	Nombre de cuenta de servicio ^	ID de cuenta de servicio
<input type="checkbox"/>	App Engine default service account	petagramtarea@appspot.gserviceaccount.com
<input type="checkbox"/>	Compute Engine default service account	499618852531-compute@developer.gcp

Luego la opción Crear Cuenta de Servicio, dar el nombre, crear la llave privada, habilitar delegación de todo el dominio de Google Apps. Esto nos dará un archivo .json que debemos copiar en tu servidor de heroku.

La configuración de mi archivo index.js en heroku es:

```
var cool = require('cool-ascii-faces');
var express = require('express');
var app = express();

app.set('port', (process.env.PORT || 5000));

var bodyParser = require("body-parser"); //agregado por mi
app.use(bodyParser.json()); //soporte para codificar json
app.use(bodyParser.urlencoded({extended: true})); //soporte para decodificar las url

var firebase = require("firebase"); //para conectarse a la base de datos firebase
firebase.initializeApp({
  serviceAccount: "PetagramTarea-3e1624ace3b4.json", //archivo descargado cuando se creo la
  cuenta de servicio
  databaseURL: "https://petagramtarea.firebaseio.com" //database url creada en firebase.
});

app.use(express.static(__dirname + '/public'));

// views is directory for all template files
app.set('views', __dirname + '/views');
app.set('view engine', 'ejs');

app.get('/android', function(request, response) {
  response.render('pages/index');
});

// POST
// https://still-woodland-31434.herokuapp.com/token-device
```

```

// token -> parametro
// token vendra en request
// matricula -> response.send(request.body.matricula);
var tokenDevicesURI = "registrar-usuario";
var tblName = "usuario_instagram"; //para que la tabla no tome el nombre del URI registrar
usuario
app.post('/' + tokenDevicesURI , function(request, response) {
    var token = request.body.token;
    var id_dispositivo = request.body.id_dispositivo; //este parametro se debe pasar en el
request usando postman
    var id_usuario_instagram = request.body.id_usuario_instagram; //este parametro se debe
pasar en el request usando postman
    var db = firebase.database();
    var tokenDevices = db.ref(tblName+'/'+id_dispositivo+'/'); //indica que inicia con
token-device el nombre como un path

    // al final esta opcion tenia .push().. con el push siempre me crea un id que lo inserta
como parte del registro
    // con esta modalidad no hago push y envio el set al final y asi tengo el id_dispositivo
como campo unico que no se repite
    //var tokenDevices = db.ref(tblName);
    // al colocarlo el id_dipositivo como campo llave, el sistema actualiza los datos y nunca
hay duplicados.
    tokenDevices.set(
        {
            //token: token,
            id_dispositivo: id_dispositivo,
            id_usuario_instagram: id_usuario_instagram
        }
    );

    var path = tokenDevices.toString(); //https://petagram-7b392.firebaseio.com/token-
device/xxxxxxxxx
    var pathSplit = path.split(tblName + "/");
    var idAutoGenerado = pathSplit[1];
    var respuesta = generarRespuestaAToken(db, idAutoGenerado);
    response.setHeader("Content-Type", "application/json");
    // cuando obtengo los datos tengo que devolver algo
    //response.send(request.body.token);
    response.send(JSON.stringify(respuesta));
});

//para generar respuesta
function generarRespuestaAToken(db, idAutoGenerado)
{
    var respuesta = {};
    var usuario = "";
    var ref = db.ref(tblName);
    ref.on("child_added", function(snapshot, prevChildKey){ //devuelve la uultima llave
generada
        usuario = snapshot.val();
        respuesta = {
            id : idAutoGenerado,
            id_dispositivo : usuario.id_dispositivo,
            id_usuario_instagram : usuario.id_usuario_instagram
        };
    })
    return respuesta;
}

app.get('/cool', function(request, response) {
    response.send(cool());
});

app.listen(app.get('port'), function() {
    console.log('Node app is running on port', app.get('port'));
});

}

```

Estructuctura de datos

```
var tokenDevicesURI = "registrar-usuario";
var tblName = "usuario_instagram"; //para que la tabla no tome el nombre del URI registrar usuario
app.post('/' + tokenDevicesURI , function(request, response) {
  var token = request.body.token;
  var id_dispositivo = request.body.id_dispositivo; //este parametro se debe pasar en el request usando postman
  var id_usuario_instagram = request.body.id_usuario_instagram; //este parametro se debe pasar en el request usando postman
  var db = firebase.database();
  var tokenDevices = db.ref(tblName+'/'+id_dispositivo+'/'); //indica que inicia con token-device el nombre como un path

  // al final esta opcion tenia .push().. con el push siempre me crea un id que lo inserta como parte del registro
  // con esta modalidad no hago push y envio el set al final y asi tengo el id_dispositivo como campo unique que no se repite
  //var tokenDevices = db.ref(tblName);
  // al colocarlo el id_dispositivo como campo llave, el sistema actualiza los datos y nunca hay duplicados.
  tokenDevices.set(
    {
      //token: token,
      id_dispositivo: id_dispositivo,
      id_usuario_instagram: id_usuario_instagram
    }
  );
});
```

postman

<https://powerful-dawn-51191.herokuapp.com/registrar-usuario>

id usuario instagram: elmasbananero

id dispositivo : cristian

En la parte del proyecto de Android, cuando eligen la opción de Recibir notificaciones, por el momento, está programado fijo que sea mi usuario elmasbananero. Va a cambiar en cada dispositivo que se utilice.
