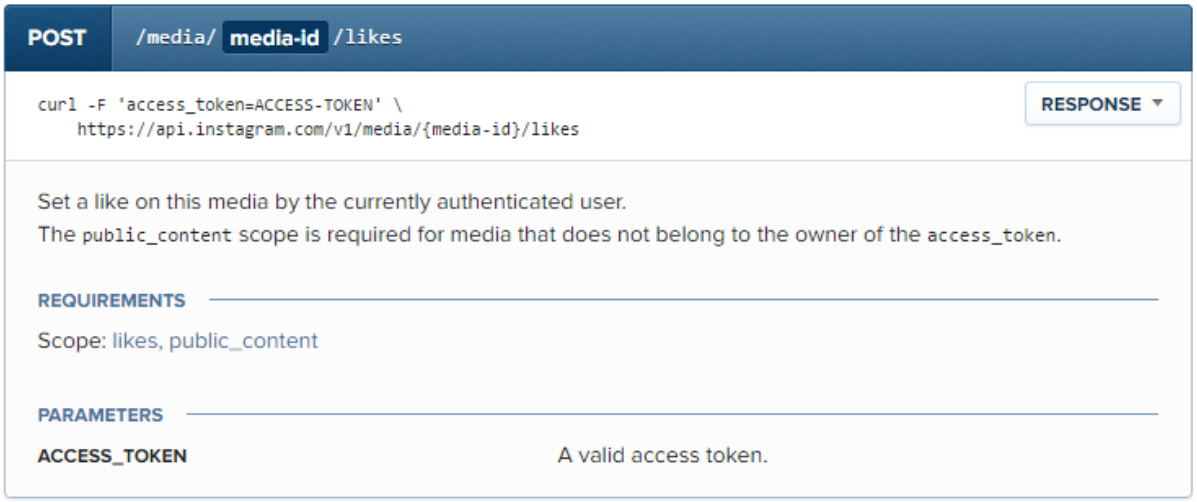
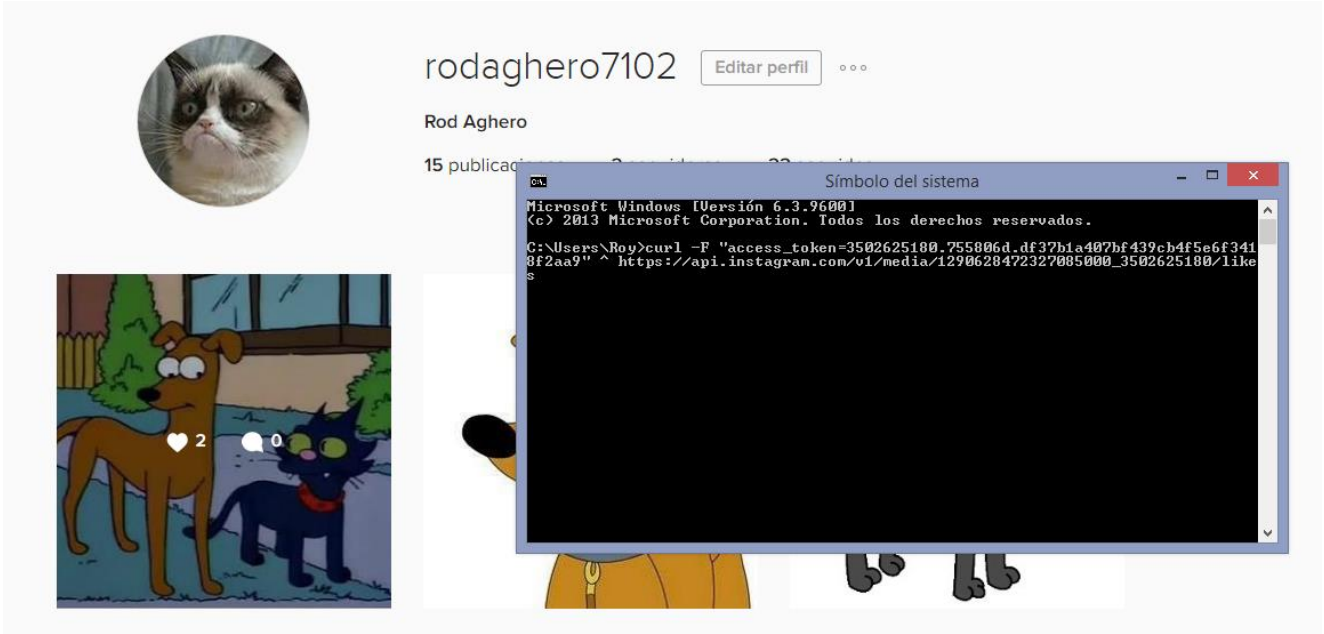
Assignment: creando, recibiendo y push notifications.

Con el proyecto de la actividad anterior (Configurando mi propio Endpoint en mi servidor):

1. Implementa el Endpoint del API de Instagram para dar like a una foto del timeline que diseñaste en actividades pasadas.

    ➔ Pendiente: no encontré algún ejemplo que pudiera usar para guiarme e implementar el Endpoint /media/media-id/likes, solamente pude utilizarlo con el **símbolo de sistema** de Windows, tal como se observa en las siguientes imágenes.



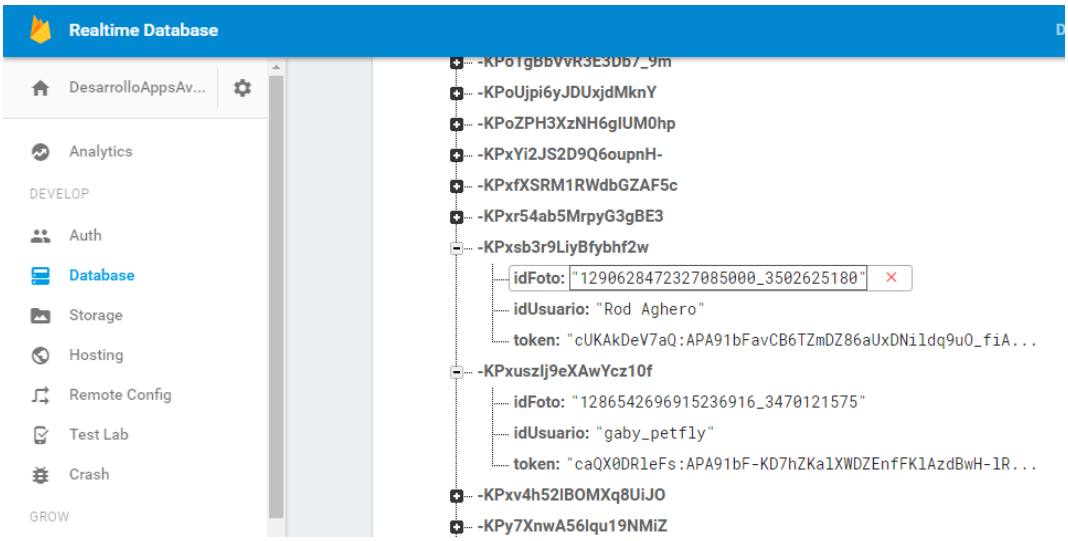


➔ Les pido, por favor, retroalimentación para complementar el assignment.

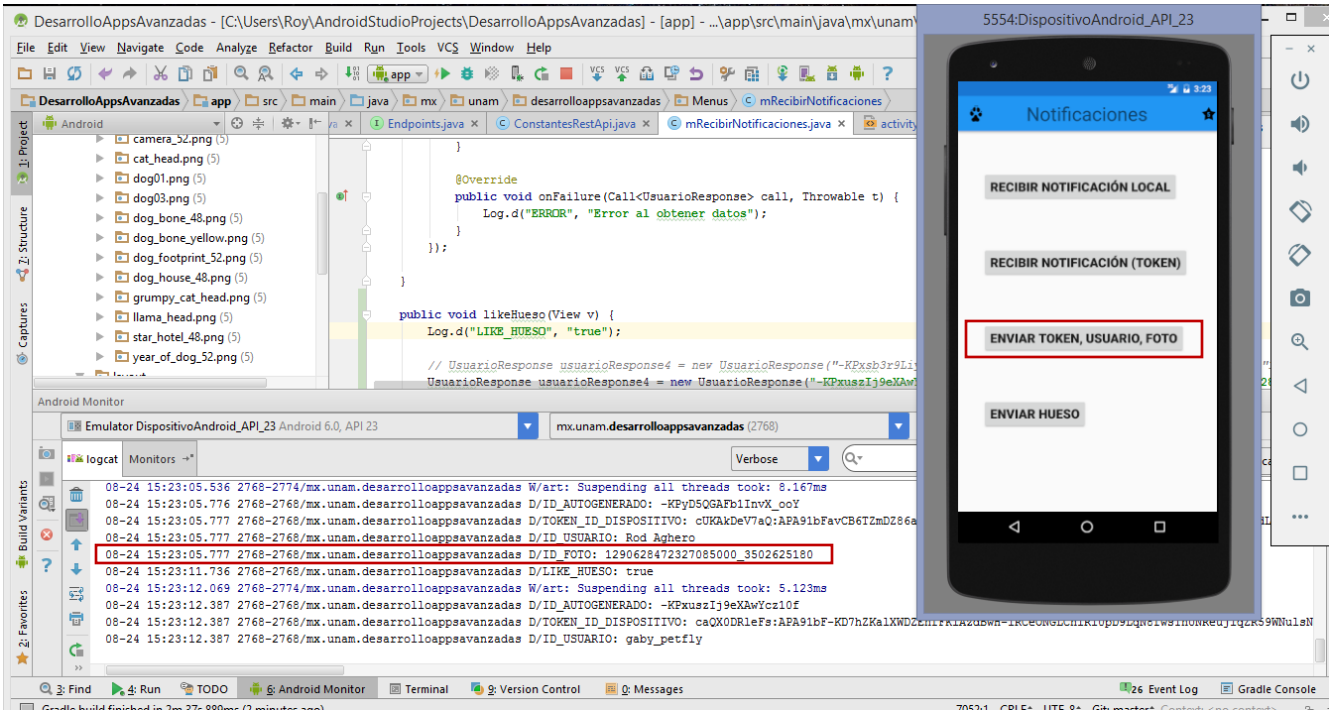
2. Además de enviar el like al API de Instagram, crea un Endpoint más en tu servidor que procese la siguiente recepción:

- id_foto_instagram
- id_usuario_instagram
- id_dispositivo

➔ para esta sección se reutilizó el Endpoint previamente creado, solamente añadiendo el idFoto.



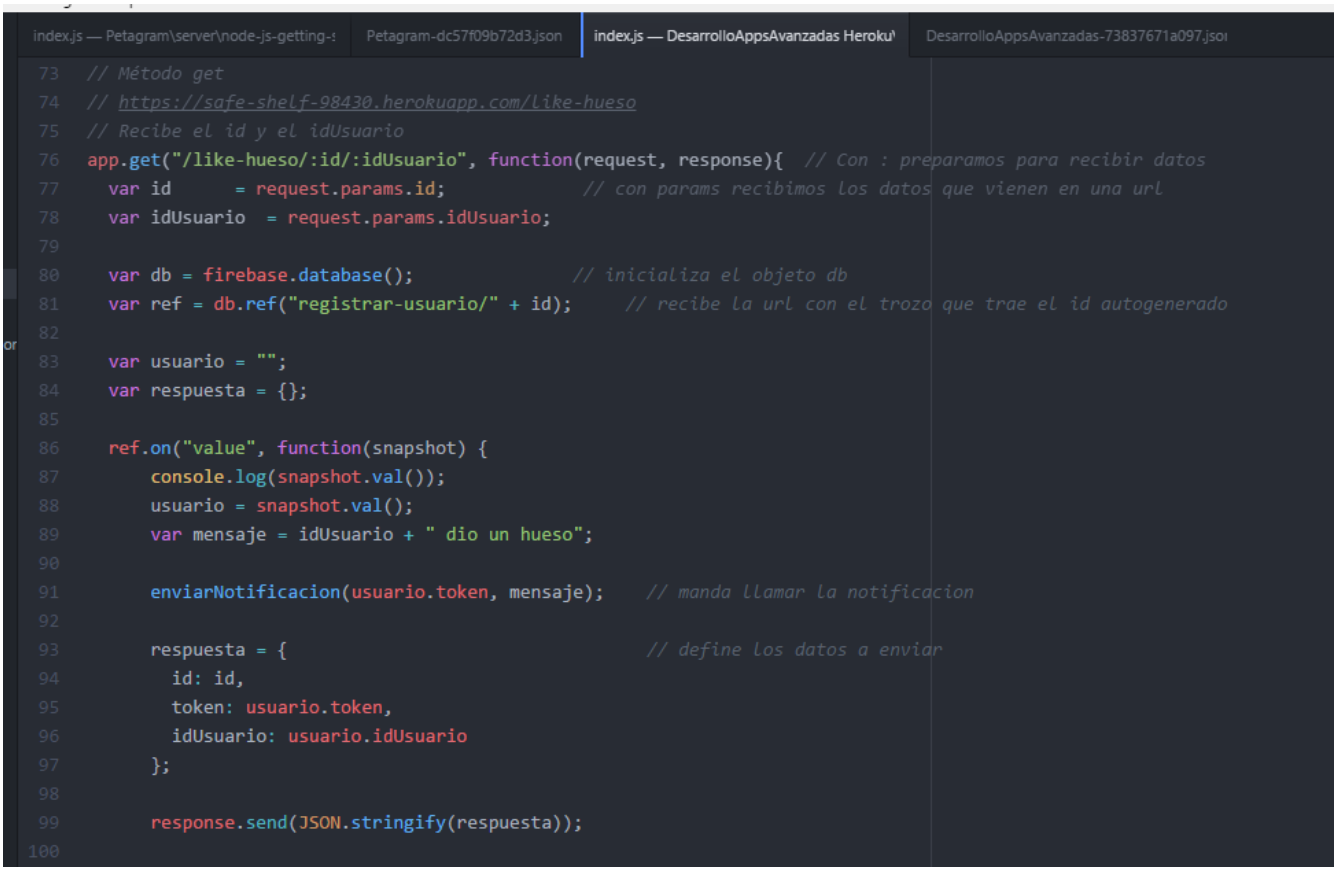
```
// Método post
// https://safe-shelf-98430.herokuapp.com/
// recibe el token y el idUsuario
var tokenDevicesURI = "registrar-usuario";
app.post("/" + tokenDevicesURI, function(request, response) {
  var token = request.body.token;                    // id_dispositivo
  var idUsuario = request.body.idUsuario;            // id_usuario
  var idFoto = request.body.idFoto;                  // id_foto


  var db = firebase.database();
  var tokenDevices = db.ref(tokenDevicesURI).push();    // PUSH para no sobre-escribir
  tokenDevices.set({
    token: token,
    idUsuario: idUsuario,
    idFoto: idFoto
  });

var path = tokenDevices.toString();

var pathSplit = path.split(tokenDevicesURI + "/");

var idAutoGenerado = pathSplit[1];

var respuesta = generarRespuestaAToken(db, idAutoGenerado);
response.setHeader("Content-Type", "application/json");
response.send(JSON.stringify(respuesta));
});
```



➔ En las imágenes anteriores se observa tanto en la consola de Android Studio como en Firebase el manejo del dato idFoto.

Es decir, este Endpoint también llevará su registro propio de likes de fotos de tu aplicación. Al recibir algún like, no sólo deberá insertarlo en tu BD, también deberás configurar una notificación y enviarla al dispositivo que tiene configurada la cuenta de Instagram a la cual pertenece la foto raiteada.

➔ Parte del método para obtener el id y el idUsuario para la notificación.

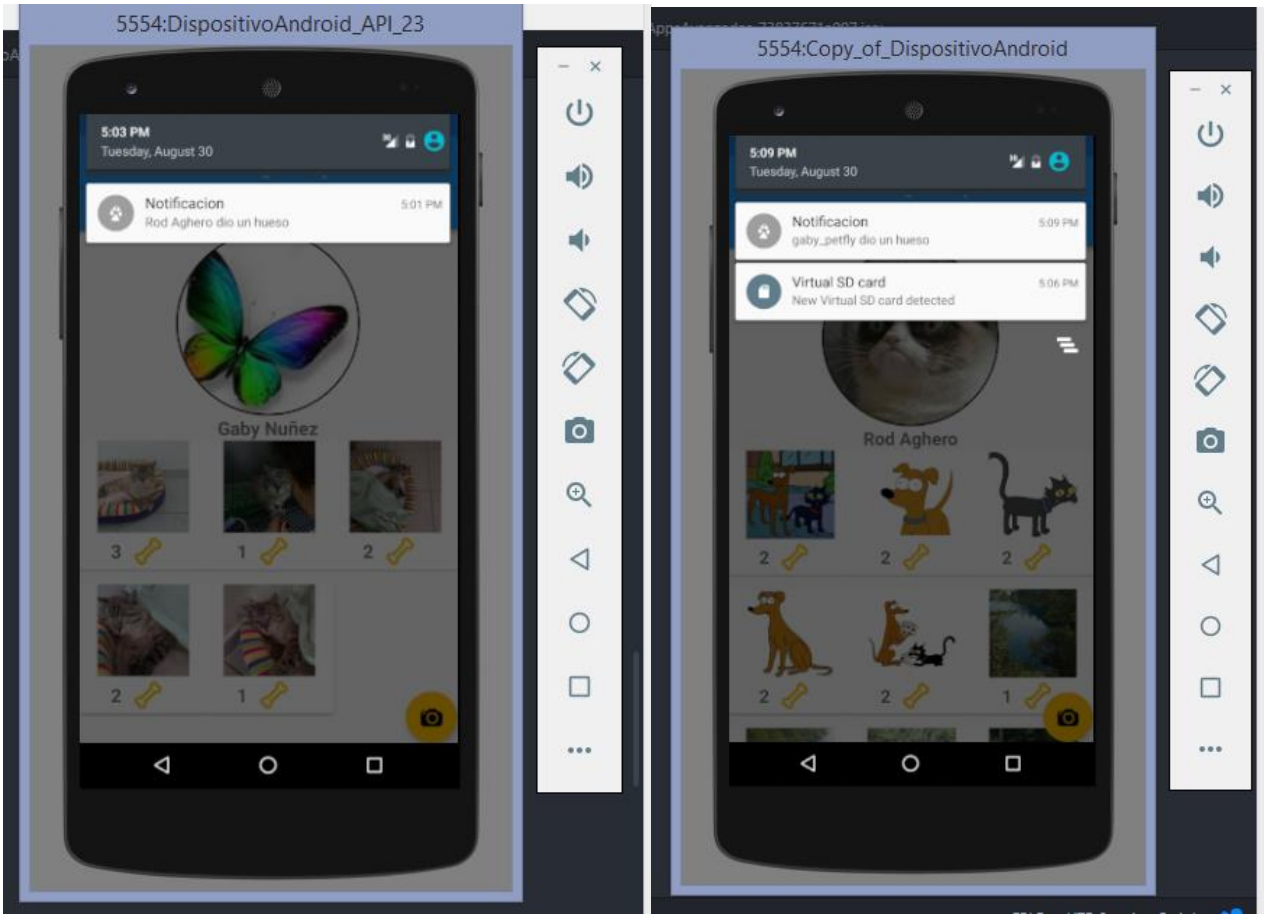


➔ Función para enviar la notificación.

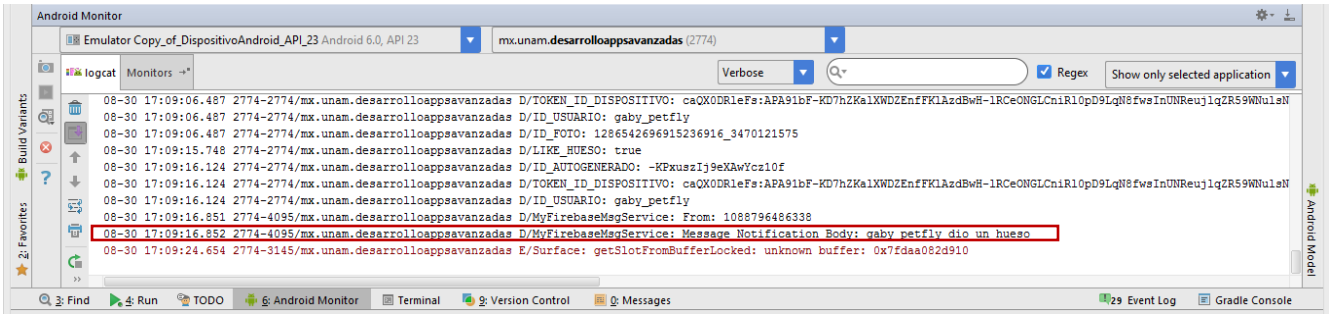
index.js — Petagram\server\node-js-getting-s | Petagram-dc57f09b72d3.json | index.js — DesarrolloAppsAvanzadas Heroku\ | DesarrolloAppsAvanzadas-73

```
});

function enviarNotificacion(tokenDestinatario, mensaje) {
  var serverKey = 'AIzaSyA3LqaTFfn0ZYgpYdZb0SCP7V8XCBkVhbQ';
  var fcm = new FCM(serverKey);

  var message = {
      to: tokenDestinatario, // required
      collapse_key: '',
      data: {},
      notification: {
          title: 'Notificacion desde Servidor',
          body: mensaje,
          icon: "dog_bone_48",
          sound: "default",
          color: "#00BCD4"
      }
  };

  fcm.send(message, function(err, response){
      if (err) {
          console.log("Something has gone wrong!");
      } else {
          console.log("Successfully sent with response: ", response);
      }
  });
}
```

➔ Para simular el intercambio de notificaciones entre dispositivos se utilizaron 2 emuladores, cada uno con su token (idDispositivo) y una cuenta asignada.



➔ Ejemplo de la interacción de la notificación del servido en la consola.



3. Una vez recibida la notificación, al pulsarla, abrirá la aplicación en la pantalla del perfil del usuario.

➔ Nota, para este assigment primero fue creado el método (como en las lecciones) adjunto en un botón de la clase **mRecibirNotificaciones.java** y posteriormente se copió el mismo método en el adaptador, para que al presionar el hueso amarillo se registrara el like (hueso).

➔ Método en **mRecibirNotificaciones.java**

```
}

public void likeHueso(View v) {
    Log.d("LIKE_HUESO", "true");

    // La cuenta que configure aquí será la que de el hueso (like)
     //UsuarioResponse usuarioResponse4 = new UsuarioResponse("-KPxsb3r9LiyBfybhf2w", "-KPxsb3r9LiyBfybhf2w", "Rod Aghero", "12906284
    UsuarioResponse usuarioResponse4 = new UsuarioResponse("-KPxuszIj9eXAwYcz10f", "-KPxuszIj9eXAwYcz10f", "gaby_petfly", "1286542696

    RestApiAdapter restApiAdapter = new RestApiAdapter();
    Endpoints endpoints1 = restApiAdapter.establecerConexionRestApiNHF();

    Call<UsuarioResponse> usuarioResponseCall = endpoints1.likeHueso(usuarioResponse4.getId(), usuarioResponse4.getIdUsuario());

    usuarioResponseCall.enqueue(new Callback<UsuarioResponse>() {
        @Override
        public void onResponse(Call<UsuarioResponse> call, Response<UsuarioResponse> response) {
            UsuarioResponse usuarioResponse5 = response.body();
            Log.d("ID_AUTOGENERADO", usuarioResponse5.getId());
            Log.d("TOKEN_ID_DISPOSITIVO", usuarioResponse5.getToken());
            Log.d("ID_USUARIO", usuarioResponse5.getIdUsuario());
            //Log.d("ID_FOTO", usuarioResponse5.getIdFoto());
        }

        @Override
        public void onFailure(Call<UsuarioResponse> call, Throwable t) {

        }
    });
}
```

➔ Método en **PerfilMascotaAdaptador.java**, en el método onBindViewHolder con el ImageView imgHuesoAmarilloPerfilMascotaCV.

```
perfilViewHolder.imgHuesoAmarilloPerfilMascotaCV.setOnClickListener(new View.OnClickListener(){
    @Override
    public void onClick(View v){
        Toast.makeText(activity, "Diste un hueso", Toast.LENGTH_SHORT).show();
        // Método para dar like/hueso
        Log.d("LIKE_HUESO", "true");

        // La cuenta que configure aquí será la que de el hueso (like)
        //UsuarioResponse usuarioResponse4 = new UsuarioResponse("-KPxsb3r9LiyBfybhf2w", "-KPxsb3r9LiyBfybhf2w", "Rod Aghero", "12
        UsuarioResponse usuarioResponse4 = new UsuarioResponse("-KPxuszIj9eXAwYcz10f", "-KPxuszIj9eXAwYcz10f", "gaby_petfly", "128

        RestApiAdapter restApiAdapter = new RestApiAdapter();
        Endpoints endpoints1 = restApiAdapter.establecerConexionRestApiNHF();

        Call<UsuarioResponse> usuarioResponseCall = endpoints1.likeHueso(usuarioResponse4.getId(), usuarioResponse4.getIdUsuario()

        usuarioResponseCall.enqueue(new Callback<UsuarioResponse>() {
            @Override
            public void onResponse(Call<UsuarioResponse> call, Response<UsuarioResponse> response) {
                UsuarioResponse usuarioResponse5 = response.body();
                Log.d("ID_AUTOGENERADO", usuarioResponse5.getId());
                Log.d("TOKEN_ID_DISPOSITIVO", usuarioResponse5.getToken());
                Log.d("ID_USUARIO", usuarioResponse5.getIdUsuario());
                //Log.d("ID_FOTO", usuarioResponse5.getIdFoto());
            }

            @Override
            public void onFailure(Call<UsuarioResponse> call, Throwable t) {

            }
        });
```