ANEXOS

ANEXO 1 – Programa AndroidGPSProyecto

package com.sawada.gpsproyecto;

import java.io.IOException; /*Cuando se programa, hay una sentencia de programación llamada trycatch que evita que la aplicación se caiga cuando hay un error de entrada y salida*/

import java.io.UnsupportedEncodingException; /*Cuando hay errores que no soportan el empaquetado*/

import java.util.ArrayList; /*Es para poder usar arreglos. En este caso se usan para guardar la latitud y longitud*/

import java.util.List; /*Es para poder usar listas. El arreglo se tiene que tranformar en lista para que pueda ser enviado y recibido*/

import org.apache.http.HttpResponse; /*Conjunto de librerías para poder utilizar el servicio con la web.*/

import org.apache.http.NameValuePair;

import org.apache.http.client.ClientProtocolException;

import org.apache.http.client.HttpClient;

import org.apache.http.client.entity.UrlEncodedFormEntity;

import org.apache.http.client.methods.HttpGet;

import org.apache.http.client.methods.HttpPost;

import org.apache.http.impl.client.DefaultHttpClient;

import org.apache.http.message.BasicNameValuePair;

import com.sawada.gpsproyecto.R;

import android.app.Activity; /*La clase que maneja lo que se muestra en la pantalla. Es una librería que llama a la interface activity*/

import android.os.Bundle; /*Es un framework que tiene los recursos para poder programar en android. Contiene las sentencias más usadas */

import android.util.Log; /*Sirve para guarder cualquier error que ocurra. Lo guarda en una bitácora*/

```
        import android.view.View;

        import android.widget.Button;

        import android.widget.EditText;

        import android.widget.Toast;


        public class AndroidGPSProyecto extends Activity { /*declaración de la clase*/

        // Email, password edittext

           EditText txtUsuario, txtPassword;


           // login button

        Button btnShowLocation;


        // Clase GPSTracker  /*Llamo a la clase GPSTracker donde están todas las
funciones para utilizar el GPS*/

        GPSTracker gps; /*Clase – Variable*/


           @Override

           public void onCreate(Bundle savedInstanceState) { /*Oncreate es la primera
función de esta clase*/

              super.onCreate(savedInstanceState);

              setContentView(R.layout.main); /*Llamo al diseño gráfico de la interfaz*/

              txtUsuario = (EditText) findViewById(R.id.txtUsuario); /*Indico que la
variable estará almacenada con este nombre*/

              txtPassword  = (EditText) findViewById(R.id.txtPassword);

              btnShowLocation = (Button) findViewById(R.id.btnShowLocation);


              // muestra  evento del click  para el boton locación /*Se espera click para
iniciar sentencias siguientes*/

              btnShowLocation.setOnClickListener(new View.OnClickListener() {
```

```
                        @Override
                        public void onClick(View arg0) {
                           // Obtengo el usuario y el password de los campos
                  String usuario = txtUsuario.getText().toString();
                  String password = txtPassword.getText().toString();
                        // Creando el cliente HTTP
               HttpClient httpClient = new DefaultHttpClient();
               HttpGet httpGet = new HttpGet("http://where.puercopop.com/status");
               httpGet.setHeader("Authorization",usuario+" "+password);
               HttpResponse response2;
                                 try {
                                          response2 = httpClient.execute(httpGet); /*Si esta
respuesta es 204, confirma que el usuario y el password existen*/

      if(response2.getStatusLine().getStatusCode()==204){

                                                   // creo el objeto clase
                                       gps = new GPSTracker(AndroidGPSProyecto.this);

                                                   // verifico si el GPS esta encendido

                                       if(gps.canGetLocation()){

                                          double latitude = gps.getLatitude();
                                          double longitude = gps.getLongitude();

                                          // Creando el HTTP Post
                                          HttpPost         httpPost          =          new
HttpPost("http://where.puercopop.com/update_position");
```

```
                    // Building post parameters, key and value pair


                    httpPost.setHeader("Authorization",usuario+"
"+password);

                    List<NameValuePair>   nameValuePair   =   new
ArrayList<NameValuePair>(2);

                    nameValuePair.add(new
BasicNameValuePair("lat", Double.toString(latitude) ));

                    nameValuePair.add(new
BasicNameValuePair("long", Double.toString(longitude)));

                    // Url Encoding a los parametros POST

                    try {

                      httpPost.setEntity(new
UrlEncodedFormEntity(nameValuePair));

                    }

                    catch (UnsupportedEncodingException e) {

                      // si hay error

                      e.printStackTrace();

                    }

                    // Haciendo el  HTTP Request

                    try {

                      HttpResponse        response          =
httpClient.execute(httpPost);


                      // writing response to log

                      Log.d("Http Response:", response.toString());




                    } catch (ClientProtocolException e) {
```

```
                                        // writing exception to log
                                        e.printStackTrace();
                                        Toast.makeText(getApplicationContext(),
"ERROR EN RESPUESTA", Toast.LENGTH_SHORT).show();

                                    } catch (IOException e) {
                                        // writing exception to log
                                        e.printStackTrace();
                                    }

                                }else{
                                    // no obtiene la logacion
                                    // GPS no esta prendido
                                    // Verificar el GPS
                                    gps.showSettingsAlert();
                                }


                    Intent i = new Intent(getApplicationContext(), Conectado.class);
                     i.putExtra("usuario", usuario);
                     i.putExtra("password", password);
                     startActivity(i);
                     finish();

                    } else {
                            if(response2.getStatusLine().getStatusCode()==401){
```

```
        Toast.makeText(getApplicationContext(), "Password o Usuario Erroneo",
Toast.LENGTH_SHORT).show();

                                            }else{
                              Toast.makeText(getApplicationContext(), "ERROR EN
CONECTAR", Toast.LENGTH_SHORT).show();

                              }}

                              } catch (ClientProtocolException e1) {
                                      // TODO Auto-generated catch block
                                      e1.printStackTrace();
                              } catch (IOException e1) {
                                      // TODO Auto-generated catch block
                                      e1.printStackTrace();
                              }

                      }
              });
          }

        }
```

ANEXO 2 – Programa Conectado

```
package com.sawada.gpsproyecto;


import java.io.IOException;

import java.io.UnsupportedEncodingException;

import java.util.ArrayList;

import java.util.List;

import java.util.Timer;

import java.util.TimerTask;


import org.apache.http.HttpResponse;

import org.apache.http.NameValuePair;

import org.apache.http.client.ClientProtocolException;

import org.apache.http.client.HttpClient;

import org.apache.http.client.entity.UrlEncodedFormEntity;

import org.apache.http.client.methods.HttpPost;

import org.apache.http.impl.client.DefaultHttpClient;

import org.apache.http.message.BasicNameValuePair;


import android.os.Bundle;

import android.os.CountDownTimer;

import android.os.Handler;

import android.app.Activity;

import android.content.Intent;

import android.util.Log;

import android.widget.ProgressBar;

import android.widget.TextView;

import android.widget.Toast;
```

```
public class Conectado extends Activity {
 private Handler mHandler = new Handler();
 String usuario;
 String password;
 GPSTracker gps;
  Timer timer;
  ProgressBar pb;
  TextView mTextField;
    boolean enProgreso;
  Handler handler;

@Override
protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_conectado);
             Toast.makeText(getApplicationContext(),                   "CONECTADO",
Toast.LENGTH_SHORT).show();
             Bundle extras = getIntent().getExtras();
             usuario= extras.getString("usuario");
             password= extras.getString("password");
             pb = (ProgressBar) findViewById(R.id.progressBar1);
             mTextField= (TextView) findViewById(R.id.tview1);

         pb.setMax(1000);
         pb.setProgress(0);
         handler = new Handler();
         enProgreso = true;
```

```
        TimerTask tarea = new TimerTask(){
            @Override
            public void run() {

                handler.post(new Runnable(){
                    public void run() {

                        progreso();
                    };
                });

                if(!enProgreso){
                    timer.cancel();//esto finaliza el hilo
                }
            }
        };

        timer = new Timer();
        timer.schedule(tarea, 100,100);//se crea un hilo
    }
  public void progreso(){
        int n = pb.getProgress() + 1;
        pb.setProgress(n);
        if (n==1000){
            enProgreso = false;
            gps = new GPSTracker(Conectado.this);
                if(gps.canGetLocation()){
                        double latitude = gps.getLatitude();
```

```
double longitude = gps.getLongitude();

// Creando el cliente HTTP

HttpClient httpClient = new DefaultHttpClient();

// Creando el HTTP Post

HttpPost httpPost = new
HttpPost("http://where.puercopop.com/update_position");

// construye post parameters, key y value pair

httpPost.setHeader("Authorization",usuario+" "+password);

List<NameValuePair> nameValuePair = new
ArrayList<NameValuePair>(2);


nameValuePair.add(new BasicNameValuePair("lat",
Double.toString(latitude) ));

nameValuePair.add(new BasicNameValuePair("long",
Double.toString(longitude)));

// Url Encoding a los parametros POST

try {

    httpPost.setEntity(new
UrlEncodedFormEntity(nameValuePair));

}

catch (UnsupportedEncodingException e) {

    // si hay error

    e.printStackTrace();

}

// Haciendo el  HTTP Request

try {

    HttpResponse response = httpClient.execute(httpPost);


    // escribe respuesta al log

    Log.d("Http Response:", response.toString());
```

```
                    } catch (ClientProtocolException e) {
                      // escribe excepcion al log
                      e.printStackTrace();
                      Toast.makeText(getApplicationContext(),       "ERROR       EN
RESPUESTA", Toast.LENGTH_SHORT).show();

                    } catch (IOException e) {
                      // escribe excepcion al log
                      e.printStackTrace();
                    }
                }else{
                    // no obtiene la logacion
                    // GPS no esta prendido
                    // Verificar el GPS
                    gps.showSettingsAlert();
                  }
              Toast.makeText(this, "Ubicandonos", 1000).show();
              pb.setProgress(0);
          }
        }



      }
```

ANEXO 3 – Programa GPSTracker

```
package com.sawada.gpsproyecto;


import android.app.AlertDialog;

import android.app.Service;

import android.content.Context;

import android.content.DialogInterface;

import android.content.Intent;

import android.location.Location;

import android.location.LocationListener;

import android.location.LocationManager;

import android.os.Bundle;

import android.os.IBinder;

import android.provider.Settings;

import android.util.Log;


public class GPSTracker extends Service implements LocationListener {


private final Context mContext;


// flag para GPS status

boolean isGPSEnabled = false;


// flag para network status

boolean isNetworkEnabled = false;


// bandera de status GPS
```

```
	boolean canGetLocation = false;


	Location location; // location

	double latitude; // latitude

	double longitude; // longitude


	// distancia minima para updates en metros

	private static final long MIN_DISTANCE_CHANGE_FOR_UPDATES = 10; // 10
meters


	// tiempo minimo para updates en ms

	private static final long MIN_TIME_BW_UPDATES = 1000 * 60 * 1; // 1 minute


	// declaro a Location Manager

	protected LocationManager locationManager;


	public GPSTracker(Context context) {

		this.mContext = context;

		getLocation();

	}


	public Location getLocation() {

		try {

			locationManager = (LocationManager) mContext

					.getSystemService(LOCATION_SERVICE);


			// obtengo GPS status

			isGPSEnabled = locationManager
```

```
.isProviderEnabled(LocationManager.GPS_PROVIDER);

                // obtengo stado de red
                isNetworkEnabled = locationManager

.isProviderEnabled(LocationManager.NETWORK_PROVIDER);

                if (!isGPSEnabled && !isNetworkEnabled) {
                        // si no hay proveedor de red
                } else {
                        this.canGetLocation = true;
                        if (isNetworkEnabled) {
                                locationManager.requestLocationUpdates(

LocationManager.NETWORK_PROVIDER,
                                                MIN_TIME_BW_UPDATES,

MIN_DISTANCE_CHANGE_FOR_UPDATES, this);
                                Log.d("Network", "Network");
                                if (locationManager != null) {
                                        location = locationManager

.getLastKnownLocation(LocationManager.NETWORK_PROVIDER);
                                        if (location != null) {
                                                latitude = location.getLatitude();
                                                longitude = location.getLongitude();
                                        }
                                }
                        }
```

```
                    // if GPS Enabled get lat/long using GPS Services
                    if (isGPSEnabled) {
                        if (location == null) {
                            locationManager.requestLocationUpdates(

    LocationManager.GPS_PROVIDER,

    MIN_TIME_BW_UPDATES,

    MIN_DISTANCE_CHANGE_FOR_UPDATES, this);
                            Log.d("GPS Enabled", "GPS Enabled");
                            if (locationManager != null) {
                                location = locationManager

    .getLastKnownLocation(LocationManager.GPS_PROVIDER);
                                if (location != null) {
                                    latitude                    =
location.getLatitude();

                                    longitude                   =
location.getLongitude();

                                }
                            }
                        }
                    }
                }


            } catch (Exception e) {
                e.printStackTrace();
            }
```

```
        return location;
}

/**
 * deja de usar GPS

public void stopUsingGPS(){
        if(locationManager != null){
                locationManager.removeUpdates(GPSTracker.this);
        }
}

/**
 * Function to get latitude
 * */
public double getLatitude(){
        if(location != null){
                latitude = location.getLatitude();
        }

        // return latitude
        return latitude;
}

/**
 * obtengo longitud
 * */
public double getLongitude(){
			if(location != null){
				longitude = location.getLongitude();
			}

			// devuelvo longitud
			return longitude;
		}

		/**
		 * chequeo GPS/wifi activados
		 * */
		public boolean canGetLocation() {
			return this.canGetLocation;
		}

		public void showSettingsAlert(){
			AlertDialog.Builder alertDialog = new AlertDialog.Builder(mContext);

		  // titulo
		  alertDialog.setTitle("GPS is settings");

		  //mensaje
		  alertDialog.setMessage("GPS is not enabled. Do you want to go to settings
	menu?");

		  alertDialog.setPositiveButton("Settings",                          new
	DialogInterface.OnClickListener() {
			public void onClick(DialogInterface dialog,int which) {
```

```
                        Intent                intent               =                new
Intent(Settings.ACTION_LOCATION_SOURCE_SETTINGS);

                        mContext.startActivity(intent);

                    }

                });


                alertDialog.setNegativeButton("Cancel",                                new
DialogInterface.OnClickListener() {

                    public void onClick(DialogInterface dialog, int which) {

                    dialog.cancel();

                    }

                });



                alertDialog.show();

            }


            @Override

            public void onLocationChanged(Location location) {

            }


            @Override

            public void onProviderDisabled(String provider) {

            }


            @Override

            public void onProviderEnabled(String provider) {

            }
```

```
@Override
public void onStatusChanged(String provider, int status, Bundle extras) {
}


@Override
public IBinder onBind(Intent arg0) {
        return null;
}


}
```

ANEXO 4: Programación Python

```
# -*- coding: utf-8 -*-

import re
from datetime import datetime
from functools import wraps
from pytz import timezone
from flask import (Flask, render_template, request, make_response, redirect,
                   url_for, session)
from flask.ext.sqlalchemy import SQLAlchemy

app = Flask(__name__)
app.config['debug'] = True
app.secret_key = "ProyectoGPSAndroid"

##############################################################################
#
##########################   SQL Alchemy STUFF
##############################
##############################################################################
#

app.config['SQLALCHEMY_DATABASE_URI'] = ('postgresql://'
                                         'PuercoPop:@localhost/where')

db = SQLAlchemy(app)

def lima_now():
    return datetime.now(timezone('America/Lima'))

class Location(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    lat = db.Column(db.Float, nullable=False)
    lng = db.Column(db.Float, nullable=False)
    date = db.Column(db.DateTime,
                     default=lima_now)
    user_id = db.Column(db.Integer, db.ForeignKey('user.id'))

    def __init__(self, lat, lng, user_id):
        self.lat = lat
        self.lng = lng
        self.user_id = user_id

    def __repr__(self):
        return '<Location %s %s>' % (self.lat, self.lng,)


class User(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    email = db.Column(db.String(120), unique=True)
    password = db.Column(db.String(120),)
    localations = db.relationship('Location', backref=db.backref('user'),
                                  lazy='dynamic',)

    def __init__(self, email, password):
        self.email = email
        self.password = password

    def __repr__(self):
        return '<User %r>' % self.email


##############################################################################
#
```

```
###                        decorators                                ###
########################################################################
#
def check_authorization(f):
    @wraps(f)
    def decorated_function(*args, **kwargs):
        response = make_response("",)

        auth_header = request.headers.get('Authorization')
        if auth_header is None:
            response.status_code = 401
            response.data = "Setea el authorization header"
            return response

        auth_regexp = re.compile("(.*) (.*)")
        match = auth_regexp.search(auth_header)
        if match is None:
            response.status_code = 400
            response.data = ("El Authorization no está bien "
                             "formado. El formato correcto es "
                             "\"Authorization: USER PASS\"")
            return response

        login, password = auth_header.split(" ")

        user = User.query.filter_by(email=login).first()

        if user is None:
            response.status_code = 401
            response.data = "El usuario %s no se encuentra "\
                            "registrado." % (login,)
            return response
        else:
            if user.password == password:
                response.status_code = 204
            else:
                response.status_code = 401
                response.data = "Wrong Password"

                return response
        return f(*args, **kwargs)
    return decorated_function


########################################################################
#
###                        Views
###
########################################################################
#

@app.route("/",  methods=['GET', 'POST'])
@app.route("/login", methods=['GET', 'POST'])
def login():
    if request.method == "POST":
        email = request.form.get('email', None)
        password = request.form.get('password', None)
        if (email is not None) and (password is not None):
            user = User.query.filter(User.email == email).first()

            if user is None or not (user.password == password):
                response = make_response()
                response.status_code = 401
                response.data = "Password equivocado"
                return response

            session["user_id"] = user.id
```

```
                return redirect(url_for("display_position"))
            else:
                response = make_response()
                response.status_code = 401
                response.data = "Faltan datos"
                return response
        else:
            return render_template('login.html')


@app.route("/logout", methods=['GET'])
def logout():
    del session['user_id']
    return redirect(url_for('login'))


@app.route("/status", methods=['GET'])
@check_authorization
def status():
    response = make_response("",)
    response.status_code = 204
    return response


@app.route("/register", methods=['GET', 'POST'])
def register():
    if request.method == "POST":
        email = request.form.get('email', None)
        password = request.form.get('password', None)

        if (email is not None) and (password is not None):
            user = User(email, password)
            db.session.add(user)
            db.session.commit()
            return redirect(url_for('login'))
        else:
            return render_template('registration_error.html')
    else:
        return render_template('register.html')


@app.route("/update_position", methods=['POST'])
@check_authorization
def update_position():
    """
    Example request
    curl -H "AUTHORIZATION: pepe pepe" --data
    "long=-77.0595471&lat=-12.0949494" -X POST 127.0.0.1:6968/update_position
    """
    response = make_response("",)

    latitude = request.form.get('lat', None)
    longitude = request.form.get('long', None)

    if latitude is None or longitude is None:
        response.status_code = 400
        response.data = "Falta el parametro lat o long"
        return response

    auth_header = request.headers.get('Authorization', None)
    login, password = auth_header.split(" ")
    user = User.query.filter_by(email=login).first()

    location = Location(latitude, longitude, user.id)
    db.session.add(location)
    db.session.commit()
    response.status_code = 204

    return response


@app.route("/retrieve_position",
           defaults={'location_id': None}, methods=['GET'])
@app.route("/retrieve_position/<location_id>",  methods=['GET'])
def display_position(location_id):
    """We must check that the location displayed belongs to the logged user to
    prevent another user seeing the users location
    """
    user_id = session.get('user_id', None)
    if user_id is None:
        return redirect(url_for("login"))

    user = User.query.filter_by(id=user_id).first()
    if location_id is None:
        current_location = Location.query.filter_by(user_id=user_id).order_by(
            Location.date.desc()).first()
    else:
        current_location =
Location.query.filter_by(user_id=user_id).filter_by(
            id=location_id).first()

    if current_location:
        latitude = current_location.lat
        longitude = current_location.lng
    else:
        latitude = None
        longitude = None

    locations = Location.query.filter_by(user_id=user_id).order_by(
        Location.date.desc())
    return render_template('where_are_you.html',
                           email=user.email,
                           user_id=user.id,
                           latitude=latitude,
                           longitude=longitude,
                           locations=locations, )

@app.route("/delete_past_locations", methods=['POST'])
def delete_past_locations():
    user_id = session.get('user_id', None)
    if user_id is None:
        return redirect(url_for("login"))

    locations = Location.query.filter_by(user_id=user_id).delete()
    db.session.commit()
    return redirect(url_for("display_position"))


if __name__ == "__main__":
    app.run(debug=True, host='0.0.0.0', port=6968)
```

ANEXO 5: Configuración del Servidor Web

```
server {
    listen 80;
    server_name where.puercopop.com;
        access_log /var/log/nginx/example.log;
    location / {
        proxy_pass        http://localhost:6968;
                proxy_set_header Host $host;
        proxy_set_header  X-Real-IP  $remote_addr;
                proxy_set_header X-Forwarded-For
$proxy_add_x_forwarded_for;
    }
}
```