Apéndice A
Código del servidor en node.js

A.1 Archivos del proyecto

```
{
  "name": "gps-api",
  "version": "1.0.0",
  "description": "simple api to return fake gps coordinates",
  "main": "index.js",
  "scripts": {
    "test": "./node_modules/.bin/mocha test/*/*.spec.js"
  },
  "author": "Roberto Espejo",
  "license": "",
  "devDependencies": {
    "chai": "^3.5.0",
    "mocha": "^3.2.0",
    "request": "^2.79.0"
  },
  "dependencies": {
    "body-parser": "^1.15.2",
    "express": "^4.14.0",
    "underscore": "^1.8.3"
  }
}
```

Código A.1 package.json.

```
var express = require('express');
var app = express();
var _ = require('underscore');

var gpsCalculator = require('./scripts/GpsCalculator');

var _acceptedRequests = [];

var _knownPlaces = [
  {
    lat: 37.4116727,
    long: -6.0022922,
    name: 'Labs'
  },
  {
    lat: 37.4086703,
    long: -5.9971619,
    name: 'Parking Isla Magica'
  },
  {
    lat: 37.4043522,
    long: -5.9888324,
```

```
    name: 'Parliament'
  }
]

app.get('/help', function(req, res) {

  _logRequest('HELP', req);

  var keys = Object.keys(req.query);

  if (!_validateArguments(keys)) {
    console.log(new Date() + ' [HELP] Returning 400: { msg: GPS coordinates need to be provided.
        Please try again.}');
    res.status(400).send({
      msg: 'GPS coordinates need to be provided. Please try again.'
    });
  } else if (!gpsCalculator.validateCoordinates(parseFloat(req.query.lat), parseFloat(req.query.long))) {
    console.log(new Date() + ' [HELP] Returning 422: { msg: Sorry, but we were not able to get your
        position. Please try again.}');
    res.status(422).send({
      msg: 'Sorry, but we were not able to get your position. Please try again.'
    });
  } else {
    console.log(new Date() + ' [HELP] Returning 200');
    var closestPlace = gpsCalculator.getNearestPoint(_knownPlaces, parseFloat(req.query.lat), parseFloat(req.query.
        long));

    res.status(200).send( closestPlace );
  }

});

app.get('/rescue', function(req, res) {

  _logRequest('RESCUE', req);

  var keys = Object.keys(req.query);

  if (!_validateArguments(keys)) {
    console.log(new Date() + ' [RESCUE] Returning 400: { msg: GPS coordinates need to be provided.
        Please try again.}');
    res.status(400).send({
      msg: 'GPS coordinates need to be provided. Please try again.'
    });
  } else if (!gpsCalculator.validateCoordinates(parseFloat(req.query.lat), parseFloat(req.query.long))) {
    console.log(new Date() + ' [RESCUE] Returning 400: { msg: Sorry, but we were not able to get your
        position. Please try again.}');
    res.status(422).send({
      msg: 'Sorry, but we were not able to get your position. Please try again.'
    });

  } else if (_alreadySentHelp(req, _acceptedRequests)) {
    console.log(new Date() + ' [HELP] Returning 200 to an already existing rescue request}');

    var originalDate = _.findWhere(_acceptedRequests, {ip: req.ip}).date;
    var timeDiff = new Date() - originalDate;

    timeDiff /= 1000;

    var seconds = Math.round(timeDiff % 60);
    timeDiff = Math.floor(timeDiff / 60);
    var minutes = Math.round(timeDiff % 60);

    res.status(200).send({
      msg: 'Help is on the way. You asked for help on ' + originalDate + ', ' + minutes + ' minutes and ' +
          seconds + ' seconds ago.'
    });

  } else {
    console.log(new Date() + ' [HELP] Returning 200: { msg: Worry not, help is on the way.}');
```

```
      _acceptedRequests.push( {
        ip: req.ip,
        date: new Date()
      });

      res. status (200).send({
        msg: 'Worry not, help is on the way.'
      });
    }

});

app.get('/requests', function(req, res) {

  var keys = Object.keys(req.query);

  _logRequest('REQUESTS', req);

  if (keys.length === 1 && keys.indexOf('flush') !== 1) {
    _acceptedRequests = [];
    console.log(new Date + ' [REQUESTS] Flushing already received requests!');
  }

  res. status (200).send(_acceptedRequests);

});

app. listen (3000);

_validateArguments = function (Arguments) {
  return Arguments.length !== 0 && (Arguments.indexOf('lat') !== -1 || Arguments.indexOf('long') !== -1);
};

_logRequest = function (endpoint, req) {
  console.log(new Date() + ' ' + '[' + endpoint + '] Received request on /' + endpoint + ' from ' + req.ip + ',
      params: ' + req.query.lat + ' ' + req.query.long);
};

_alreadySentHelp = function (req, acceptedRequests) {
  if (_.findWhere(acceptedRequests, {ip: req.ip})) {
    return true;
  } else {
    return false;
  }
}
```

Código A.2 server.js.

A.2 Tests

```
var request = require('request');
var chai = require('chai');
var expect = chai.expect;

describe('GPS API', function () {

  describe('Help', function () {

    var url = 'http://localhost:3000/help';

    describe('Error', function () {

      it('should return an error if no parameters provided', function (done) {
        request(url, function(error, response, body) {
              expect(response.statusCode).to.equal(400);
              expect(body).to.equal(JSON.stringify({
              msg: 'GPS coordinates need to be provided. Please try again.'
                 }));
              done();
            });
      });

      it('should return an error if the parameters are not GPS coordinates', function(done) {

        request(url + '?abc=123&123=abc', function(error, response, body) {
              expect(response.statusCode).to.equal(400);
              expect(body).to.equal(JSON.stringify({
              msg: 'GPS coordinates need to be provided. Please try again.'
                 }));
              done();
            });

      });

      it('should return an error if the GPS coordinates are wrong', function(done) {

        request(url + '?lat=123&long=abc', function(error, response, body) {
              expect(response.statusCode).to.equal(422);
              expect(body).to.equal(JSON.stringify({
              msg: 'Sorry, but we were not able to get your position. Please try again.'
                 }));
              done();
            });

      });
    });

    describe('Success', function () {

      it('should return status 200 if the GPS coordinates are ok', function(done) {

        request(url + '?lat=45&long=73', function(error, response, body) {
              expect(response.statusCode).to.equal(200);
              done();
            });

      });

      it('should return the closest help point if the GPS coordinates are ok', function(done) {

        request(url + '?lat=37.392894&long=-5.994779', function(error, response, body) {
              expect(response.statusCode).to.equal(200);
              expect(body).to.equal(JSON.stringify({
             lat: 37.4043522,
            long: -5.9888324,
            name: 'Parliament',
             distance: 1378.135254316005
```

```
            }));
                done();
              });

      });

    });

});

describe('Rescue', function () {

  var url = 'http://localhost:3000/rescue';

  describe('Error', function () {

    it('should return an error if no parameters provided', function (done) {
      request(url, function(error, response, body) {
            expect(response.statusCode).to.equal(400);
            expect(body).to.equal(JSON.stringify({
            msg: 'GPS coordinates need to be provided. Please try again.'
              }));
            done();
          });
    });

    it('should return an error if the parameters are not GPS coordinates', function(done) {
      request(url + '?abc=123&123=abc', function(error, response, body) {
            expect(response.statusCode).to.equal(400);
            expect(body).to.equal(JSON.stringify({
            msg: 'GPS coordinates need to be provided. Please try again.'
              }));
            done();
          });
    });

    it('should return an error if the GPS coordinates are wrong', function(done) {
      request(url + '?lat=123&long=abc', function(error, response, body) {
            expect(response.statusCode).to.equal(422);
            expect(body).to.equal(JSON.stringify({
            msg: 'Sorry, but we were not able to get your position. Please try again.'
              }));
            done();
          });

    });

  });

  describe('Success', function () {

    it('should return status 200 if the GPS coordinates are ok', function(done) {
      request(url + '?lat=45&long=73', function(error, response, body) {
            expect(response.statusCode).to.equal(200);
            expect(body).to.equal(JSON.stringify({
            msg: 'Worry not, help is on the way.'
              }));
            done();
          });
    });

    it('should return status 200 if a second request arrives', function(done) {
      request(url + '?lat=45&long=73', function(error, response, body) {
            expect(response.statusCode).to.equal(200);
            expect(body).not.to.equal(JSON.stringify({
            msg: 'Worry not, help is on the way.'
              }));
            done();
          });
    });
```

```
    });

  });

  describe('Requests', function () {
    var url = 'http://localhost:3000/requests';

    it('should return the received requests', function(done) {
      request(url, function(error, response, body) {
        expect(response.statusCode).to.be.equal(200);
        done();
      })
    });

    it('should flush the request if asked to', function(done) {
      request(url + '?flush', function(error, response, body) {
        expect(response.statusCode).to.be.equal(200);
        done();
      })
    });
  });

});
```

Código A.3 test/server.spec.js.

```
var gpsCalculator = require('../../scripts/GpsCalculator');

var request = require('request');
var chai = require('chai');
var expect = chai.expect;

describe('gpsCalculator', function () {

  describe('Validate coordinates', function () {

    describe('error', function () {
      it('should return false if gps coordinates are not present', function () {
        expect(gpsCalculator.validateCoordinates()).to.be.false;
      });

      it('should return false if gps coordinates are not numbers', function () {
        expect(gpsCalculator.validateCoordinates(20,'cbahhadhadhada')).to.be.false;
      });

      it('should return false if gps coordinates are not valid (positive numbers)', function () {
        expect(gpsCalculator.validateCoordinates(95, 190)).to.be.false;
      });

      it('should return false if gps coordinates are not valid (negative numbers)', function () {
        expect(gpsCalculator.validateCoordinates(-95, -190)).to.be.false;
      });

    });

    describe('success', function () {
      it('should return true if gps coordinates are valid (positive numbers)', function () {
        expect(gpsCalculator.validateCoordinates(85, 170)).to.be.true;
      });

      it('should return true if gps coordinates are valid (negative numbers)', function () {
        expect(gpsCalculator.validateCoordinates(-85, -170)).to.be.true;
      });
    });

  });

  describe('Get nearest point', function () {

    var knownPlaces = [
        {
          lat: 37.4116727,
          long: -6.0022922
        },
        {
          lat: 37.4086703,
          long: -5.9971619
        }
    ];

    describe('error', function () {
      it('should return an error if no arguments are provided', function () {
        expect(gpsCalculator.getNearestPoint()).to.be.equal(-1);
      });

      it('should return an error if no knownPlaces are provided', function () {
        expect(gpsCalculator.getNearestPoint(null,84,127)).to.be.equal(-1);
      })

      it('should return an error if no coordinates are provided', function () {
        expect(gpsCalculator.getNearestPoint(knownPlaces)).to.be.equal(-1);
      })
    });

    describe('success', function () {
      it('should calculate the nearest point properly', function() {
        expect(gpsCalculator.getNearestPoint(knownPlaces,37.4043522,-5.9888324)).to.deep.equal({
```

```
            lat: 37.4086703,
            long: -5.9971619,
            distance: 878.5390121762819
          });
        })
      });
    });

});
```

Código A.4 scripts/GpsCalculator.spec.js.

A.3 Scripts

```
var _ = require('underscore');

exports.getNearestPoint = function(knownPlaces, lat, long){
  if(!knownPlaces || !lat || !long) {
    return -1;
  }

  var locationsWithDistance =_.map(knownPlaces, function(location) {
    location.distance = _getDistanceBetweenTwoPoints(location.lat, location.long, lat, long);
    return location;
  });

  var result = _.where(locationsWithDistance, {distance: _.min(_.map(locationsWithDistance, function(location) {
      return location.distance; }))});

  return result[0];
};

exports.validateCoordinates = function(lat, long) {

    if (!lat || !long) {
      return false;
    } else if (isNaN(lat) || isNaN(long)) {
      return false;
    } else {
      return ((Math.abs(lat) < 90) && (Math.abs(long) < 180));
    }
};

_getDistanceBetweenTwoPoints = function (lat1,long1,lat2,long2) {
  var R = 6371e3;
  var phi1 = _toRadians(lat1);
  var phi2 = _toRadians(lat2);
  var inc_phi = _toRadians(lat2-lat1);
  var inc_lambda = _toRadians(long2-long1);

  var a = Math.sin(inc_phi/2) * Math.sin(inc_phi/2) +
          Math.cos(phi1) * Math.cos(phi2) *
          Math.sin(inc_lambda/2) * Math.sin(inc_lambda/2);
  var c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1-a));

  return R * c;
}

_toRadians = function (degrees) {
  var pi = Math.PI;
  return degrees * pi / 180;
}
```

Código A.5 scripts/GpsCalculator.js.

Apéndice B
Código de la aplicación Android

B.1 *manifests*

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.pfc.app.iHelp">

    <uses-feature android:name="android.hardware.location.gps" />

    <uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
    <uses-permission android:name="android.permission.CHANGE_WIFI_STATE" />
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission android:name="android.permission.INTERNET" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name="com.pfc.app.iHelp.activities.MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name="com.pfc.app.iHelp.activities.MainMenuActivity" />
        <activity android:name="com.pfc.app.iHelp.activities.RescueMessageActivity"></activity>
    </application>

</manifest>
```

Código B.1 manifest.xml.

B.2 *java*

B.2.1 *activities*

```
package com.pfc.app.iHelp. activities ;

import android.content. Intent ;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.View;

import com.pfc.app.iHelp.R;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState ) {
        super.onCreate( savedInstanceState );
        setContentView(R.layout. activity_main );
    }

    public void startScanning (View view) {
        Intent intent = new Intent( this , MainMenuActivity.class);
         startActivity ( intent );
    }

}
```

Código B.2 MainActivity.java.

```
package com.pfc.app.iHelp. activities ;

import android.Manifest;
import android. annotation .TargetApi;
import android.app. Activity ;
import android.app.AlertDialog;
import android.content.Context;

import android.content. DialogInterface ;
import android.content. Intent ;
import android.content.pm.PackageManager;
import android. location .Location;
import android. location .LocationManager;
import android.net. wifi .WifiManager;
import android.os.Build;
import android.os.Bundle;
import android. provider . Settings ;
import android.support.v4.content.ContextCompat;
import android. util .Log;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

import com.pfc.app.iHelp. classes .GPSHandler;
import com.pfc.app.iHelp.R;
import com.pfc.app.iHelp. classes . RestClientTask ;
import com.pfc.app.iHelp. classes .ToastHandler;
import com.pfc.app.iHelp. classes .WifiHandler;

import org.json.JSONException;
import org.json.JSONObject;

import java.net.MalformedURLException;
import java.net.URL;
import java. util . concurrent .ExecutionException;
import java. util . concurrent .ExecutorService;
import java. util . concurrent .Executors;
```

```
public class MainMenuActivity extends Activity {

    private static final String TAG = "MainMenuActivity";

    private Context context;

    private ToastHandler toastHandler;

    private WifiHandler wifiHandler;
    private GPSHandler gpsHandler;

    private static final String[] LOCATION_PERMS = {
            Manifest.permission.ACCESS_FINE_LOCATION
    };

    private Location location;

    @TargetApi(Build.VERSION_CODES.M)
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main_menu);

        context = getApplicationContext();

        toastHandler = new ToastHandler(context);

        wifiHandler = new WifiHandler((WifiManager) getSystemService(Context.WIFI_SERVICE), toastHandler);
        gpsHandler = new GPSHandler((LocationManager) getSystemService(Context.LOCATION_SERVICE), this);

        wifiHandler.scanNetworksAndConnectToBest();

        if (!canAccessLocation()) {
            Log.d(TAG, "GPS Permission: requesting ...");
            requestPermissions(LOCATION_PERMS, 1);
        }

        checkIfGpsIsEnabled();

        ExecutorService fixedThreadPoolExecutor = Executors.newFixedThreadPool(1);
        fixedThreadPoolExecutor.submit(gpsHandler);

        location = gpsHandler.getLocation();

        if(location != null) {
            Log.d(TAG, location.toString());
        } else {
            toastHandler.showUserMessage("Cannot get GPS coordinates");
            disableButtonsAndShowMessage();
        }
    }

    public void setLocation(Location location) {
        this.location = location;
    }

    private boolean canAccessLocation() {
        Log.d(TAG, "GPS Permission: Permission is " + String.valueOf(ContextCompat.checkSelfPermission(this,
                android.Manifest.permission.ACCESS_FINE_LOCATION) == PackageManager.PERMISSION_GRANTED));
        return ContextCompat.checkSelfPermission(this, android.Manifest.permission.ACCESS_FINE_LOCATION) ==
                PackageManager.PERMISSION_GRANTED;
    }

    public void help(View view) throws ExecutionException, InterruptedException, JSONException {
        RestClientTask restClientTask = new RestClientTask();

        restClientTask.execute(composeURL("help", location));
        JSONObject result = restClientTask.get();
        if(result != null) {
            Log.d(TAG, result.toString());
```

```
            Intent intent = new Intent(this, HelpMessageActivity.class);
            intent.putExtra("locationName", result.get("name").toString());
            intent.putExtra("distance", (Double) result.get("distance"));
            intent.putExtra("lat", (Double) result.get("lat"));
            intent.putExtra("lon", (Double) result.get("long"));
            intent.putExtra("deviceLat", location.getLatitude());
            intent.putExtra("deviceLon", location.getLongitude());
            startActivity(intent);
        } else {
            handleNetworkError("Connection lost. Restarting app...");
        }
    }

    public void rescue(View view) throws ExecutionException, InterruptedException, JSONException {
        RestClientTask restClientTask = new RestClientTask();

        restClientTask.execute(composeURL("rescue", location));
        JSONObject result = restClientTask.get();
        if (result != null) {
            Log.d(TAG, result.toString());
            Intent intent = new Intent(this, RescueMessageActivity.class);
            intent.putExtra("message", result.get("msg").toString());
            startActivity(intent);
        } else {
            handleNetworkError("Connection lost. Restarting app...");
        }
    }


    private URL composeURL(String endpoint, Location location) {
        try {
            return new URL("http://192.168.42.1:3000/" + endpoint + "?lat=" + location.getLatitude() + "&long=" +
                    location.getLongitude());
        } catch (MalformedURLException e) {
            e.printStackTrace();
            return null;
        }
    }

    public void handleNetworkError(String message) {
        toastHandler.showUserMessage(message);
        Intent intent = new Intent(this, MainActivity.class);
        startActivity(intent);
    }

    private void checkIfGpsIsEnabled() {
        LocationManager locationManager = (LocationManager) getSystemService(Context.LOCATION_SERVICE);

        if (!locationManager.isProviderEnabled(LocationManager.GPS_PROVIDER)) {
            Log.d(TAG, "GPS is not enabled");
            showAlert();
        } else {
            Log.d(TAG, "GPS is enabled");
        }
    }

    public void showAlert() {
        Log.d(TAG, "ShowAlert");
        final AlertDialog.Builder dialog = new AlertDialog.Builder(this);
        dialog.setCancelable(true);
        dialog.setTitle("Enable Location")
                .setMessage("Your Locations Settings is set to 'Off'.\nPlease Enable Location to " +
                        "use this app")
                .setPositiveButton("Location Settings", new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface paramDialogInterface, int paramInt) {
                        Intent myIntent = new Intent(Settings.ACTION_LOCATION_SOURCE_SETTINGS);
                        startActivity(myIntent);
                    }
                })
                .setNegativeButton("Cancel", new DialogInterface.OnClickListener() {
```

```
                    @Override
                    public void onClick(DialogInterface paramDialogInterface, int paramInt) {
                    }
                });
        AlertDialog alertDialog = dialog.create();
        Log.d(TAG, "ShowAlert after create");

        alertDialog.show();
    }

    private void disableButtonsAndShowMessage() {
        Button btn1 = (Button) findViewById(R.id.button2);
        btn1.setEnabled(false);
        btn1.setAlpha(.5f);

        Button btn2 = (Button) findViewById(R.id.button3);
        btn2.setEnabled(false);
        btn2.setAlpha(.5f);

        TextView text = (TextView) findViewById(R.id.textView);
        text.setText("Sorry, but were unable to get your GPS coordinates.");
    }

}
```

Código B.3 MainMenuActivity.java.

```
package com.pfc.app.iHelp.activities;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.TextView;

import com.pfc.app.iHelp.R;

import java.text.DecimalFormat;

public class HelpMessageActivity extends AppCompatActivity {

    private TextView textTopView;
    private TextView textBottomView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_help_message);

        textTopView = (TextView) findViewById(R.id.textView3);
        textBottomView = (TextView) findViewById(R.id.textView4);

        Bundle bundle = getIntent().getExtras();

        String name = bundle.getString("locationName");
        double lat = bundle.getDouble("lat");
        double lon = bundle.getDouble("lon");
        double distance = bundle.getDouble("distance");

        double deviceLat = bundle.getDouble("deviceLat");
        double deviceLon = bundle.getDouble("deviceLon");

        changeTopText(name, lat, lon);
        changeBottomText(distance, getDirection(deviceLat,deviceLon, lat,lon));
    }

    private String getDirection(Double deviceLat, Double deviceLon, Double lat, Double lon) {
        if (lat > deviceLat) {
            if (lon >= deviceLon + 0.1) {
                return "North-East";
            } else if (deviceLon > lon - 0.1 && deviceLon < lon + 0.1) {
```

```
                return "North";
            } else {
                return "North-West";
            }
        } else {
            if (lon >= deviceLon + 0.1) {
                return "South-East";
            } else if (deviceLon > lon - 0.1 && deviceLon < lon + 0.1) {
                return "South";
            } else {
                return "South-West";
            }
        }
    }

    private void changeTopText(String name, double lat, double lon) {
        textTopView.setText("The closest help point is " + name + ", situated in coordinates " + Double.toString(
            lat) + ", " + Double.toString(lon) + ".");
    }

    private void changeBottomText(double distance, String direction) {
        DecimalFormat twoDForm = new DecimalFormat("#.##");
        Double roundedDistance = Double.valueOf(twoDForm.format(distance));

        textBottomView.setText("This is " + roundedDistance + "m away, walking in direction " + direction + ".");
    }
}
```

Código B.4 HelpMessageActivity.java.

```
package com.pfc.app.iHelp.activities;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.TextView;

import com.pfc.app.iHelp.R;

public class RescueMessageActivity extends AppCompatActivity {

    private TextView textView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_rescue_message);

        textView = (TextView) findViewById(R.id.textView2);

        String text = getIntent().getStringExtra("message");

        changeText(text);
    }

    public void changeText(String text) {
        textView.setText(text);
    }
}
```

Código B.5 RescueMessageActivity.java.

B.2.2 *classes*

```
package com.pfc.app.iHelp.classes;

import android.location.Location;
```

```
import android.location.LocationListener;
import android.location.LocationManager;
import android.os.Bundle;
import android.util.Log;

import com.pfc.app.iHelp.activities.MainMenuActivity;

import java.util.concurrent.Callable;

public class GPSHandler implements Callable<Integer> {

    private static final String TAG = "GPSHandler";

    private MainMenuActivity mainMenu;
    private LocationManager locationManager;

    public GPSHandler(LocationManager locationManager, MainMenuActivity mainMenu) {
        this.locationManager = locationManager;
        this.mainMenu = mainMenu;
    }

    public Integer call() {
        try {

            // Define a listener that responds to location updates
            LocationListener locationListener = new LocationListener() {
                public void onLocationChanged(Location location) {
                    Log.d(TAG, location.toString());
                    mainMenu.setLocation(location);
                }

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

                public void onProviderEnabled(String provider) {
                    Log.d(TAG, "GPS enabled");
                }

                public void onProviderDisabled(String provider) {
                    Log.d(TAG, "GPS disabled");
                }
            };

            // Register the listener with the Location Manager to receive location updates
            locationManager.requestLocationUpdates(
                    LocationManager.GPS_PROVIDER, 5000, 10, locationListener);
        } catch (SecurityException e) {
            Log.d("GPS Location: ", e.toString());
        }
        return new Integer(1);
    }

    public Location getLocation() {
        Log.d(TAG, "Getting location ... ");
        try {
            Log.d(TAG, "Returning location: " + locationManager.getLastKnownLocation(LocationManager.
                GPS_PROVIDER));
            return locationManager.getLastKnownLocation(LocationManager.GPS_PROVIDER);
        } catch(SecurityException e) {
            Log.e(TAG, "Exception getting GPS location: " + e.toString());
            return null;
        }
    }
}
```

Código B.6 GPSHandler.java.

```
package com.pfc.app.iHelp.classes;
```

```
import android.os.AsyncTask;
import android.util.Log;

import org.json.JSONException;
import org.json.JSONObject;

import java.io.BufferedInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.net.HttpURLConnection;
import java.net.ProtocolException;
import java.net.URL;
import java.util.Scanner;

public class RestClientTask extends AsyncTask<URL, Integer, JSONObject> {

    private static final String TAG = "Rest client";

    @Override
    protected JSONObject doInBackground(URL... urls) {
        HttpURLConnection urlConnection = null;
        try {
            URL url = urls[0];

            Log.d(TAG, "Calling url: " + url);
            urlConnection =
                    (HttpURLConnection) url.openConnection();

            Log.d(TAG, "Instantiated connection");

            configureConnection(urlConnection);

            int statusCode = urlConnection.getResponseCode();

            Log.d(TAG, "Response code from the server: " + statusCode);

            if (statusCode != HttpURLConnection.HTTP_OK) {
                Log.d(TAG, "Connection failed: statusCode: " + statusCode);
                return new JSONObject();
            }

            InputStream in = new BufferedInputStream(
                    urlConnection.getInputStream());

            String responseText = getResponseText(in);
            Log.d(TAG, "responseText: " + responseText);

            return new JSONObject(responseText);

        } catch (IOException | JSONException e) {
            e.printStackTrace();
        } finally {
            if (urlConnection != null) {
                urlConnection.disconnect();
            }
        }

        return null;
    }

    private HttpURLConnection configureConnection(HttpURLConnection urlConnection) {
        try {
            urlConnection.setConnectTimeout(20000);
            urlConnection.setReadTimeout(20000);
            urlConnection.setRequestMethod("GET");
            Log.d(TAG, "Configured connection");
        } catch (ProtocolException e) {
            Log.d(TAG, "Something went wrong configuring the connection");
            e.printStackTrace();
        }
```

```
            return urlConnection;
        }

        private static String getResponseText(InputStream inStream) {
            return new Scanner(inStream).useDelimiter("\\A").next();
        }
    }
```

Código B.7 RestClientTask.java.

```
package com.pfc.app.iHelp.classes;

import android.content.Context;
import android.widget.Toast;

public class ToastHandler {

    private static int duration = Toast.LENGTH_SHORT;
    private Context context;

    public ToastHandler(Context context) {
        this.context = context;
    }

    public void showUserMessage(String message) {
        Toast toast = Toast.makeText(context, message, duration);
        toast.show();
    }

}
```

Código B.8 ToastHandler.java.

```
package com.pfc.app.iHelp.classes;

import android.net.wifi.ScanResult;
import android.net.wifi.WifiConfiguration;
import android.net.wifi.WifiInfo;
import android.net.wifi.WifiManager;
import android.util.Log;

import java.util.List;

public class WifiHandler {

    private static final String TAG = "WifiHandler";

    private WifiManager wifiManager;
    private ToastHandler toastHandler;

    public WifiHandler(WifiManager wifiManager, ToastHandler toastHandler) {
        this.wifiManager = wifiManager;
        this.toastHandler = toastHandler;
    }

    public boolean isWifiEnabled() {
        return wifiManager.isWifiEnabled();
    }

    public boolean isAlreadyConnectedToANetwork() {
        WifiInfo wifiInfo = wifiManager.getConnectionInfo();
        if(wifiInfo.getNetworkId() < 0) {
            Log.d(TAG, "Not connected to any network.");
            return false;
        } else {
            return true;
        }
```

```
}

public boolean isConnectedNetworkValid() {
    WifiInfo wifiInfo = wifiManager.getConnectionInfo();
    Log.d(TAG, "WIFI INFO: " + wifiInfo.toString());
    if (wifiInfo.getSSID().contains("RP")) {
        Log.d(TAG, "Already connected to " + wifiInfo.getSSID());
        return true;
    } else {
        return false;
    }
}

public void scanNetworksAndConnectToBest() {

    Log.d(TAG, "Scanning networks. Checking if Wifi is enabled ...");

    if (!isWifiEnabled()) {
        Log.d(TAG, "Wifi not enabled. Enabling ...");

        toastHandler.showUserMessage("Wifi not enabled. Enabling ...");

        wifiManager.setWifiEnabled(true);
        Log.d(TAG, "Wifi enabled.");
    } else {
        Log.d(TAG, "Wifi was already enabled. Checking if it's already connected to a network ...");
        if (isAlreadyConnectedToANetwork()) {
            Log.d(TAG, "Already connected to a network.");
            if (isConnectedNetworkValid()) {
                Log.d(TAG, "Connected network is valid.");
                return;
            } else {
                Log.d(TAG, "Connected network is not valid - disconnecting ...");
                WifiInfo wifiInfo = wifiManager.getConnectionInfo();
                wifiManager.disableNetwork(wifiInfo.getNetworkId());
                wifiManager.disconnect();
            }
        }
    }

    wifiManager.startScan();
    Log.d(TAG, "Scanning ...");

    List<ScanResult> connections = wifiManager.getScanResults();
    Log.d(TAG, "List of connections: " + connections.toString());

    ScanResult bestWifi = getBestWifi(connections);

    int networkId = findExistingNetworkConfig(bestWifi.SSID);

    if (networkId < 0) {
        Log.d(TAG, "Unknown network. Adding it now...");
        addNewAccessPoint(bestWifi);
    } else {
        boolean b = wifiManager.enableNetwork(networkId, true);
        Log.d("WifiPreference", "enableNetwork returned " + b);
    }

    toastHandler.showUserMessage("Connected to Network " + bestWifi.SSID);
}

private ScanResult getBestWifi(List<ScanResult> connections) {
    Log.d(TAG, "Getting the best WiFi connection ...");

    ScanResult best = null;
    for (ScanResult scanResult : connections) {
        if (best == null || (wifiManager.compareSignalLevel(best.level, scanResult.level) < 0 && scanResult.
            SSID.contains("RP"))) {
            best = scanResult;
        }
    }
```

```
        Log.d(TAG, "Returning best WiFi SSID: " + best.SSID);

        return best;
    }

    private int findExistingNetworkConfig(String ssid) {

        Log.d(TAG, "Checking if the configuration for best WiFi already exists ...");

        if (ssid != null && !ssid.isEmpty()) {
            for (WifiConfiguration wifiConfig : wifiManager.getConfiguredNetworks()) {
                if (ssid.equals(wifiConfig.SSID)) {
                    Log.d(TAG, "It does.");
                    return wifiConfig.networkId;
                }
            }
        }
        // Didn't find a matching network ssid
        Log.d(TAG, "It doesn't.");
        return -1;
    }

    public void addNewAccessPoint(ScanResult scanResult) {

        WifiConfiguration wc = new WifiConfiguration();
        wc.SSID = '\"' + scanResult.SSID + '\"';
        wc.priority = 10000;
        wc.status = WifiConfiguration.Status.ENABLED;
        wc.allowedKeyManagement.set(WifiConfiguration.KeyMgmt.NONE);
        int res = wifiManager.addNetwork(wc);
        Log.d(TAG, "WifiPreference: add network returned " + res);
        boolean b = wifiManager.enableNetwork(res, true);
        wifiManager.saveConfiguration();

        Log.d(TAG, "WifiPreference: enableNetwork returned " + b);
    }
}
}
```

Código B.9 WifiHandler.java.

B.3 *res*

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.pfc.app.iHelp.activities.HelpMessageActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:text="Large Text"
        android:id="@+id/textView3"
        android:layout_marginTop="68dp"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true" />

    <TextView
        android:layout_width="wrap_content"
```

```
        android:layout_height ="wrap_content"
        android:textAppearance ="? android:attr /textAppearanceLarge"
        android:text ="Large Text"
        android:id ="@+id/textView4"
        android:layout_alignParentBottom ="true"
        android:layout_alignLeft ="@+id/textView3"
        android:layout_alignStart ="@+id/textView3"
        android:layout_marginBottom="81dp" />
</RelativeLayout>
```

Código B.10 activity_help_message.xml.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:background="@color/lightBlue"
    android:layout_width ="match_parent"
    android:layout_height ="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context =". activities .MainActivity">

    <Button
        android:onClick="startScanning"
        android:padding="2dp"
        android:scaleX="2"
        android:scaleY="2"
        android:background="@color/darkBlue"
        android:textColor ="@color/white"
        android:layout_width ="wrap_content"
        android:layout_height ="wrap_content"
        android:text ="Start scanning"
        android:id ="@+id/button"
        android:layout_centerVertical ="true"
        android:layout_centerHorizontal ="true" />
</RelativeLayout>
```

Código B.11 activity_main.xml.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation =" vertical " android:layout_width ="match_parent"
    android:background="@color/lightBlue"
    android:layout_height ="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:textAlignment ="center"
    android:touchscreenBlocksFocus ="true"
    android:weightSum="1">

    <TextView
        android:layout_width ="239dp"
        android:paddingTop="64dp"
        android:paddingBottom="64dp"
        android:layout_height ="wrap_content"
        android:text ="How can we help you?"
        android:textSize ="24sp"
        android:id ="@+id/textView"
        android:layout_gravity =" center_horizontal "
        android:layout_weight ="0.13" />

    <Button
        android:onClick="help"
        android:padding="2dp"
```

```
            android:scaleX="2"
            android:scaleY="2"
            android:background="@color/darkBlue"
            android:textColor="@color/white"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_centerHorizontal="true"
            android:layout_marginBottom="64dp"
            android:text="Check nearest help point"
            android:textSize="12sp"
            android:id="@+id/button2"
            android:layout_gravity=" center_horizontal "
            android:breakStrategy="simple" />

        <Button
            android:onClick="rescue"
            android:padding="2dp"
            android:scaleX="2"
            android:scaleY="2"
            android:background="@color/darkBlue"
            android:textColor="@color/white"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_centerVertical="true"
            android:layout_centerHorizontal="true"
            android:text="Ask for rescue"
            android:id="@+id/button3"
            android:layout_gravity=" center_horizontal " />

    </LinearLayout>
```

Código B.12 activity_main_menu.xml.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".activities.RescueMessageActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:text="Large Text"
        android:id="@+id/textView2"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="193dp" />
</RelativeLayout>
```

Código B.13 activity_rescue_message.xml.