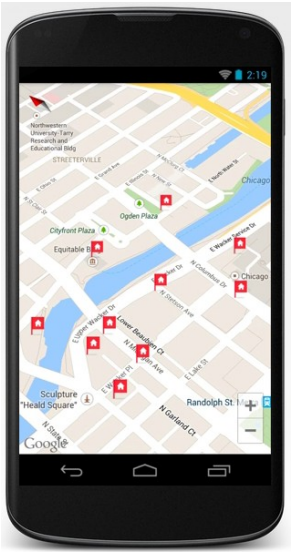
Assignment 2: Your Friends in the World

Introduction

In this task, you should build an app that shows your and your friends' geographical positions in a map. The app will send the phone's position to a server. It will also receive your friends' locations from the server. In an extended version, the app will also give the opportunity to send text and picture messages to your friends.

Goal



This is an individual task. The student will, after the task, understand:

- How to use the sensors in the device from an application.
- How to use services from the Google Maps API from an application.
- On a fundamental level, how communication with a server works.

The student will, after the task, be able to:

- Using GPS in an application.
- Communicating with a server via TCP / IP.
- Generate and interpret JSON-formatted text.
- Using the services of Google Maps API.
- Create an application with a functional UI.
- Use threads in an application.

Requirements for a Godkänt

In order to achieve a grade Godkänt in this task:

- Have a map that works, preferably Google Maps because it is well documented. The user must be able to zoom in and out, as well as to move around the map.
- The map shows the connected devices (friends) by means of an icon of any look. One should be able to identify other users.
- Supply the device's location to a server. Receive others' locations from the server.
- The default language of the app will be English. The user should be able to switch to Swedish.
- The user will be able to register at one group on the server. We therefore need a way to show the current groups so that the user can choose the group to sign up to. The user can also start a new group by registering a new group. The server returns an ID that you should use for later communications.
- Limitations:
    - Up to 20 groups can coexist in the server.
    - Up to 20 users can be registered to a group.
    - If you do not update your position in the last two minutes you will be deregistered from the group. This prevents the groups from growing while developing the app.
    - A group with no members is removed from the server.
- The user will be able to unsubscribe from the group.
- The app will send the device's position to the server twice per minute.
- The application should listen to the messages with the group members' locations. Before a member have sent any position to the server, its longitude and latitude values will be "NaN".
- All communications with the server will be done via TCP/IP.

Requirements for a Väl Godkänt

In order to achieve a grade Väl Godkänt:

- Provide a chat so that the user sends and receives text and image messages. Those messages are stored in the server to that other users can see them. An image message consists of a picture taken from the application, with some extra info added such as text and the geographic position.
- Allow the user to participate in several groups, so that the user can choose the members of which groups will be displayed on the map.
- The app will allow a user to be registered to multiple groups. The user should be able to choose which groups are displayed on the map.
- The user should be able to send text messages to / receive text messages from other members in the group. Received messages will be displayed. Messages do no need to be filtered to several groups.
- The user should be able to send images to / receive images from other members in the group. Received image messages (picture + text) will be displayed. Messages do no need to be filtered to several groups. Optionally, it would be nice that the downloaded image thumbnails can be displayed on the exact locations of the map where they were taken. If the user clicks on the thumbnail, the content of the image message (ie, picture + text) is shown.
- The application handles the device's rotation.

Description

Google Maps

You will need some specific setup for your application in order to use Google Maps (Google API installed, Google Maps key, permissions, etc.). You can read about this on:

https://developers.google.com/maps/documentation/android/

Information about the server

Ip: 195.178.227.53

Port to connect to the server: 7117

Communicating with the server

All communications take place on separate threads, never in the UI - thread. You can assume that the user does not rotate the device for a grade G. The best solution will let a service handle all the network traffic.

• A thread will listen to incoming messages while the app is running. Use the method readUTF () to receive a message (DataInputStream). Every request to the server will get a response. Between the request and the response you can get extra information from the server regarding "locations", "text chat", and "image chat". For the grade G you have to deal with "locations" but you can ignore "text chat" and "image chat". For a grade VG "locations", "text chat" and "image chat" should be handled.

• Let a thread handle the rest of the network communication (connecting at the beginning of the app, send messages while the app is running, disconnect when the app is terminated). Use the method writeUTF() to send a message (DataOutputStream).

• If your app supports picture messaging you can code the upload / download of images by using a AsyncTask. Use writeObject() (in ObjectOutputStream) to upload an image and readObject () (in Object InputStream) to download an image. The image is transferred as a byte array.

When the app starts, the connection to the server takes place:

```
Socket socket = new Socket( inetAddress, port );
InputStream    is    =    socket.getInputStream();
DataInputStream   dis  =  new  DataInputStream(is)
```

```
OutputStream    os    =    socket.getOutputStream();
DataOutputStream  dos = new DataOutputStream(os);
```

The socket should be closed right before terminating the app:

socket.close ();

Sending messages with writeUTF and receiving message with readUTF:

```
dos.writeUTF(  jsonMessage  );

dos.flush();

------------------------------

String  message  = dis.readUTF();
```

Image Message

The image must not be larger than 64KB. This means that you have to resize it. The format of the image should be jpg.

The user sends an image message. Then the server responds with an upload message that contains information for the upload (ImageID and port). An image is uploaded to the server as a byte array.

After uploading, all users in the group of the sender will get a message from the server with the ImageID and the port number where the image can be retrieved from. The image is retrieved as a byte array on the specified port.

If the user sends a wrong ImageID to the server, the server will respond with a byte array with length=0. After receiving a message about an image that can be downloaded, the user has approximately 5 minutes to download it. The user will have to make a decision during this time.

An image is uploaded to the server as a byte array:

1. Send and image message.

2. Afterwards, you'll receive an upload message containing an ImageID plus an uploading port.

3. Connect to the server on the specified port and send the ImageID (String) and then the image.

```
byte[]  uploadArray=...;  //  byte array containing the image data

Socket  = new Socket(  InetAdress,  port  );

ObjectOutputStream  output=  new ObjectOutputStream(socket.getOutputStream());

output.flush();

output.writeUTF(imageID);
output.flush();
output.writeObject(uploadArray);
output.flush();

:

socket.close();
```

An image is downloaded from the server as a byte array:

1. The server sends a message indicating that an image message is available. This message contains the ImageID and the port to connect to.

2. Connect to the server on the specified port, sending the specified ImageID. Then download byte array.

```
byte[]  downloadArray;  //  the image from the server will be stored here

Socket  socket  = new Socket(  InetAdress,  port  );

ObjectInputStream  input=  new ObjectInputStream(socket.getInputStream());
```

```
ObjectOutputStream  output= new ObjectOutputStream(socket.getOutputStream());
output.flush();
output.writeUTF(imageid);
output.flush();
downloadArray  = (byte[])input.readObject();
:
socket.close();
```

Working things out

Remember to develop the app part by part. Test separately each part before adding it to the app. Examples of some of these parts:

Google Maps

• Get Google Maps to work.

• Method of receiving a number of users + position and showing on the map

• Somehow the show marked the user's identity

Manage JSON

• Be able to create strings formatted as JSON, maybe using different methods (JSONObject):

```
Public String members( String group ) {
                StringWriter stringWriter = new StringWriter();
                JsonWriter writer = new JsonWriter( stringWriter );
                writer.beginObject()
                                .name(”type”).value(”members”)
                                .name(”group”).value(group)
                                .endObject();
                return stringWriter.toString();
}
```

• Retrieve information from the JSON - formatted message.

Android

* Provide a good design of the app (UX, UI)

* Create functional UI - fragments. ...

Able to convert between image data and byte array

* Convert image to byte arrays to send them to the server.

* Convert byte arrays back to images.

Communicating with server

* Send and receive JSON - formatted messages.

* Upload a byte - array to the server via ObjectOutputStream - writeObject

* Download a byte - array from the server via ObjectInpputStream - readObject

Communication with the server via JSON - formatted text:

Hanteras på G och VG nivå	Meddelande till server	Svar från server
Registration	{ ”type”:”register”, ”group”:”*NAME*”, ”member”:”*NAME*” }	{ “type”:”register”, “group”:”NAME”, ”id”:”*ID*” }
Deregistration	{ ”type”:”unregister”, ”id”:”*ID*” }	Same message in return
Members in a group	{ “type”:”members”, “group”,”*NAME*” }	{ “type”:”members”, “group”:”*NAME*”, “members”:[{“member”:”*NAME*”},…] }
Current groups	{ ”type”:”groups” }	{ “type”:”groups”, ”groups”:[{”group”:”*NAME*”}, …] }
Set position	{ “type”:”location”, ”id”:”*ID*”, ”longitude”:”*LONGITUDE*”, “latitude”:”*LATITUDE*” }	Samma meddelande i retur
	Meddelande från server	
Positions in a group	{ ”type”:”locations”, ”group”:”*NAME*”, ”location”:[{”member”:”*NAME*”, ”longitude”:”*LONGITUDE*”, ”latitude”:”*LATITUDE*” }, …] }	

Wrong requests	{ ”type”:”exception”, ”message”:”*MESSAGE*” }	
Hanteras på VG-nivå	**Meddelande till server**	**Svar från server**
Enter text message	{ ”type”:”textchat”, ”id”:”*ID*”, ”text”:”*TEXT*” }	Same message in return
Enter image message	{ ”type”:”imagechat”, ”id”:”*ID*”, “text”:”*TEXT*”, ”longitude”:”*LONGITUDE*”, “latitude”:”*LATITUDE*” }	{ “type”:”upload”, “imageid”:”*IMAGEID*”, “port”:”*PORT*” }
	Meddelande från server	
Text message for the app	{ ”type”:”textchat”, ”group”:”*NAME*”, “member”:”*NAME*”, ”text”:”*TEXT*” }	
Image message for the app	{ ”type”:”imagechat”, ”group”:”*NAME*”, “member”:”*NAME*”, “text”:”*TEXT*”, ”longitude”:”*LONGITUDE*”,, “latitude”:”*LATITUDE*”, “imageid”:”*IMAGEID*”, “port”:”*PORT*” }	

When you're done

Deliver the whole project folder compressed as a ZIP file. The file must be named as follows:

DA345AHT17_P2_Efternamn_Förnamn.zip

Accounting Session 1: Rating U / G / VG

Upload your solution to It's learning by 12/10/2017 at 11:00. The evaluation will take place on 12/10/2017 at 13.15.

Accounting Sessions 2 and 3: Rating U / G / VG

To be announced.