DA345A, Utveckling av mobila applikationer
TS, Malmö Högskola

Världen och vännerna

Version 1.0	2017-10-01

Inledning

Du ska bygga en app som visar din och dina vänners geografiska position på en karta. Appen skickar mobilens position till en server och erhåller dina vänners positioner från servern.

I en utvidgad variant så kommer appen även ge möjlighet att skicka text- och bildmeddelanden till dina vänners mobiler.

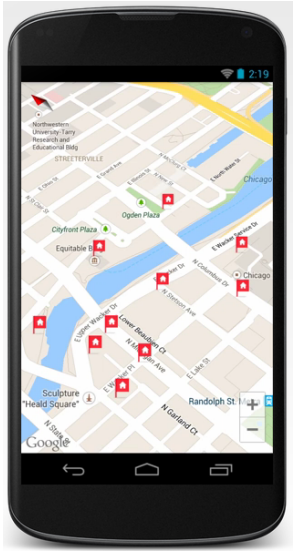
Mål

Studenten skall, efter uppgiften, förstå:

- Hur man använder sensorer i mobilen i en applikation
- Hur man använder tjänster från Google Maps API i en applikation
- På en grundläggande nivå hur kommunikation med en server fungerar

Studenten skall, efter uppgiften, kunna:

- Använda GPS-sensorn i en applikation
- Kommunicera med en server via TCP/IP
- Generera och tolka JSON-formatterad text
- Använda tjänster i Google Maps API
- Skapa en applikation med funktionellt användargränssnitt
- Använda trådar i en applikation



Betygsvärdering

Kriterier för Godkänt

För att få Godkänt på denna individuella uppgift krävs att applikationen:

- Har en karta som fungerar tillfredställande, förslagsvis Google Maps eftersom den är väl dokumenterad. Användaren skall kunna zooma in och ut samt flytta kartan med ett finger.
- På kartan visar uppkopplade enheter (vänner) med hjälp av en ikon av valfritt utseende. Användarnas identitet bör framgå.
- Levererar mobilens position till en server, och tar emot vänners position från servern.
- Default-språk i appen ska vara engelska. Om användaren har svenska som inställning på mobilen ska språket i appen vara svenska.

Kriterier för Väl godkänt

För att få Väl godkänt på denna individuella uppgift krävs att applikationen dessutom:

- Har en chat-funktion i vilken användaren, till servern, ska kunna skicka textmeddelande och bildmeddelande, och kunna se andras textmeddelande och bildmeddelande. Bildmeddelandet består av en bild tagen från applikationen kompletterad med text och geografisk position.
- Låter användaren delta i flera grupper och som låter användaren välja vilken av grupperna som ska synas på kartan.

DA345A, Utveckling av mobila applikationer
TS, Malmö Högskola

Beskrivning av uppgiften

I den här uppgiften skall du bygga en app som visar din och dina vänners geografiska position på en karta.

Funktioner på G-nivå:

- Appen ska kunna registrera sig till en grupp på servern. Det krävs alltså något sätt att visa aktuella grupper och välja grupp att registrera sig till. Man kan även starta en ny grupp genom att registrera sig till en ny grupp. Servern svarar med ett ID vilket du använder senare vid kommunikation.
  Begränsingar:
  * Max 20 grupper kan existera samtidigt i servern
  * Max 20 användare kan vara registrerade till en grupp
  * Om du inte uppdaterat din position senast 2 minuterna så avregistreras du från gruppen. Detta för att hålla antalet grupper nere / antalet medlemmar nere då ni utvecklar era appar.
  * En grupp utan medlemmar tas bort från servern.
- Appen ska kunna avregistrera sig från en grupp.
- Appen ska skicka mobilens position till servern ett par gånger per minut.
- Appen ska lyssna efter meddelande om gruppmedlemmarnas positioner. Innan en medlem har skickat någon position till servern är longitud=”NaN” och latitud=”NaN”.
- Kommunikationen med servern sker via TCP/IP.

Funktioner på VG-nivå:

- Appen ska hantera att en användare kan vara registrerad till flera grupper. Användaren ska kunna välja vilken grupp som syns på kartan.
- Användaren ska kunna skicka textmeddelande till / ta emot textmeddelande från övriga medlemmar i en grupp. Mottagna textmeddelande ska kunna visas. Meddelanden behöver inte filtreras på olika grupper.
- Användaren ska kunna skicka bildmeddelande till / ta emot bildmeddelande från övriga medlemmar i en grupp. Mottagna bildmeddelande (bild + text) ska kunna visas. Meddelanden behöver ej filtreras på olika grupper.
  Det är väldigt trevligt, men ej nödvändigt, att nerladdade bilder visas, där bilden togs, på en karta. Och kanske varje bild är klickbar varvid innehållet i bildmeddelandet (dvs. bild+text) visas.
- Applikationen ska klara rotation av mobilen

Google Maps

För att använda Google Maps krävs en del pillande (Google API installerad, Google Maps nyckel, permissions mm). Du kan läsa om detta bl.a. på:
https://developers.google.com/maps/documentation/android/

Information om server

Ip: 195.178.227.53
Port att ansluta till servern på: 7117

Kommunikation med servern:

Kommunikationen med servern kräver en del. I appen ska all kommunikation ske via separata trådar, ej via UI-tråden. Du kan förutsätta att användaren inte roterar sin telefon på G-nivå. Men det är

naturligtvis en god lösning att låta en service ta hand om nätverkstrafiken.

- Låt en tråd lyssna efter inkommande meddelande under tiden appen körs. Använd metoden *readUTF()* för att ta emot ett meddelande (*DataInputStream*).
  Då du gjort en förfrågan till servern kommer det att komma ett svar. Mellan förfrågan och svar kan det komma annan information från servern – ”locations”, ”textchat” och ”imagechat” – dvs information som servern skickar ut utan tidigare förfrågan från användaren.
  G-nivå: du måste hantera ”locations” men kan strunta i ”textchat” och ”imagechat”
  VG-nivå: ”locations”, ”textchat” och ”imagechat” ska hanteras
- Låt en tråd sköta resten av nätverkskommunikationen (koppla upp i början av appen, skicka meddelande under tiden appen körs, koppla ner då appen avslutas). Använd metoden *writeUTF()* för att skicka ett meddelande (*DataOutputStream*).
- Om din app stödjer bildmeddelande så kan du låta uppladdningen/nerladdningen av bilder ske med hjälp av ett AsyncTask-objekt. Använd *writeObject()* (i *ObjectOutputStream*) för att ladda upp en bild och *readObject()* (i *ObjectInputStream*) för att ladda ner en bild. Bilden överförs som ett bytearray-objekt.

När appen startar ska uppkoppling mot servern ske:

```
Socket socket = new Socket( inetAddress, port );
InputStream is = socket.getInputStream();
DataInputStream dis = new DataInputStream(is);
OutputStream os = socket.getOutputStream();
DataOutputStream dos = new DataOutputStream(os);
```

När appen avslutas ska nerkoppling mot servern ske:

```
socket.close();
```

Ett meddelande skickas med writeUTF och tas emot med readUTF:

```
dos.writeUTF( jsonMessage );
dos.flush();
------------------------------
String message = dis.readUTF();
```

Bildmeddelande

Bilden får ej vara större än 64KB. Det innebär att du måste skala om bilden till lämplig storlek.
Formatet på bilden ska vara jpg.
Användaren skickar ett imagechat-meddelande. Servern svarar med ett upload-meddelande som innehåller uppladdningsinformation. En bild laddas upp till servern som ett bytearray-objekt. Efter uppladdningen får samtliga användare i gruppen ett meddelande från servern i vilket det framgår vilken port bilden hämtas från på servern. Bilden hämtas som ett bytearray-objekt på angiven port.
Om något går fel vid uppladdningen så erhåller gruppens medlemmar ej något imageachat-meddelande.
Om användaren anger felaktigt imageid vid nerladdningen så överförs en byte-array med längden 0.
Användaren har ca 5 minuter på sig att ladda ner bilden. Det betyder att appen kan låta användaren avgöra om bilden ska laddas ner eller ej.

En bild laddas upp på servern som ett bytearray-objekt:

1. Skicka imagechat-meddelande
2. Sedan tar du emot ett upload-meddelande som innehåller imageID+uppladdningsport
3. Koppla upp mot servern på angiven port och skicka imageID (som String) och sedan bild

```
byte[] uploadArray=...; // tilldelas en byte-array som innehåller bilddata
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();
```

En bild laddas ner från servern som ett bytearray-objekt

1. Servern skickar meddelande om att ett bildmeddelande finns att hämta. Meddelandet innehåller imageID och port att ansluta på.

2. Koppla upp mot servern på angiven port, skicka angiven imageID och ladda ner bytearray-objekt.

```
byte[] downloadArray;  // tilldelas bilddata från servern, sista instruktionen nedan
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();
```

Att arbeta med uppgiften

Tänk på att urveckla appen bit för bit och testa olika delar innan de infogas i appen. Exempel på olika delar:

Google Maps

- Få Google Maps att fungera.
- Metod som tar emot ett antal användare + position och visar på kartan
- På något sätt visa markerade användares identitet

Hantera JSON

- Kunna skapa strängar formatterade som JSON, kanske med olika metoder:
  // members-meddelande
  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();
  }
  Men det går lika bra att arbeta direkt med JSONObject.
- Kunna plocka ut information från ett JSON-formatterat meddelande.

Android

* Skapa en vettig design av appen.
* Skapa funktionella UI-fragment. …

Kunna konvertera mellan bilddata och bytearray

* Konvertera bild till byte-array för att skicka till servern.
* Konvertera byte-array till visbar bilddata.

Kommunicera med servern

- Kunna skicka och ta emot JSON-formatterade meddelanden.
  * Kunna ladda upp en byte-array till servern via ObjectOutputStream – writeObject
  * Kunna ladda ner en byte-array från servern via ObjectInpputStream – readObject

Inlämning

Lämna din lösning på It's learning under namnet DA345AHT17_P2_Efternamn_Förnamn.zip. Detta ska ske senast den dag lösningen presenteras.

När du är klar

Vid inlämningen ska du placera main-katalogen i en zip-fil. Filen skall namnges på följande sätt:
DA345AHT17_P2_Efternamn_Förnamn.zip

Redovisningstillfälle 1: Betyg U/G/VG

Lämna in din lösning på It's learning senast den 15/10-2017, kl. 23.55. Redovisning äger rum den 12/10, kl 13.15-

Redovisningstillfälle 2: Betyg U/G/VG

Lämna in din lösning på It's learning senast den 5/11-2017, kl. 23.55. Redovisning äger rum den 2/11, kl 13.15-.

Ytterligare redovisningstillfälle meddelas: Betyg U/G

Kommunikation med servern sker via JSON-formatterad text (preliminär):

Hanteras på G och VG nivå	Meddelande till server	Svar från server
Registrering	{ ”type”:”register”, ”group”:”*NAME*”, ”member”:”*NAME*” }	{ “type”:”register”, ”group”:”*NAME*” ”id”:”*ID*” }
Avregistrering	{ ”type”:”unregister”, ”id”:”*ID*” }	Samma meddelande i retur
Medlemmar i grupp	{ “type”:”members”, “group”,”*NAME*” }	{ “type”:”members”, “group”:”*NAME*”, “members”:[{“member”:”*NAME*”},…] }
Aktuella grupper	{ ”type”:”groups” }	{ “type”:”groups”, ”groups”:[{”group”:”*NAME*”}, …] }
Ange position	{ “type”:”location”, ”id”:”*ID*”, ”longitude”:”*LONGITUDE*”, “latitude”:”*LATITUDE*” }	Samma meddelande i retur
	Meddelande från server	
Positioner i grupp	{ ”type”:”locations”, ”group”:”*NAME*”, ”location”:[{”member”:”*NAME*”, ”longitude”:”*LONGITUDE*”, ”latitude”:”*LATITUDE*” }, …] }	
Vid felaktiga förfrågningar	{ ”type”:”exception”, ”message”:”*MESSAGE*” }	
Hanteras på VG-nivå	**Meddelande till server**	**Svar från server**
Ange textmeddelande	{ ”type”:”textchat”, ”id”:”*ID*”, ”text”:”*TEXT*” }	Samma meddelande i retur
Ange bildmeddelande	{ ”type”:”imagechat”, ”id”:”*ID*”, “text”:”*TEXT*”, ”longitude”:”*LONGITUDE*”, “latitude”:”*LATITUDE*” }	{ “type”:”upload”, “imageid”:”*IMAGEID*”, “port”:”*PORT*” }
	Meddelande från server	
Textmeddelande till applikation	{ ”type”:”textchat”, ”group”:”*NAME*”, “member”:”*NAME*”, ”text”:”*TEXT*” }	
Bildmeddelande till applikation	{ ”type”:”imagechat”, ”group”:”*NAME*”, “member”:”*NAME*”, “text”:”*TEXT*”, ”longitude”:”*LONGITUDE*”,, “latitude”:”*LATITUDE*”, “imageid”:”*IMAGEID*”, “port”:”*PORT*” }	