Datateknik C, Examensarbete, 15 högskolepoäng

Lokalisering och visualisering av område
En smartphone-applikation för en ökad trygghetskänsla

Anders Alfredsson och Gustav Larsson

Dataingenjörsprogrammet, 180 högskolepoäng

Örebro vårterminen 2016

Examinator: Lars Karlsson

LOCALIZATION AND VISUALIZATION OF AN AREA,
A SMARTPHONE APPLICATION FOR AN INCREASED SENSE OF SECURITY

Örebro universitet
Institutionen för
naturvetenskap och teknik

Örebro University
School of Science and Technology
SE-701 82 Örebro, Sweden 701 82 Örebro

Sammanfattning

Rapporten handlar om olika metoder för att lokalisera smartphones och skapandet av en Androidapplikation. Applikationen skulle visualisera Campusområdet vid Örebro universitet för att öka medvetenheten och säkerhetskänslan för personer som är där kvällar och nätter. Implementationen av systemet beskrivs tillsammans med de problem som uppstod, samt dess lösningar.

Abstract

The report is about different methods of localizing smartphones and the creation of an Android application. The application should visualize the Campus for Örebro university to raise awareness and the sense of security for people who are there at night. The implementation of the system is described along with the problems during development, and how they were solved.

Förord

Vi vill tacka våran uppdragsgivare och handledare, professor Franziska Klügl för att vi fick möjligheten att utföra detta projekt och hjälpen vi fick under projektets gång.

Innehållsförteckning

1 Inledning

1.1 Bakgrund

Efter ett flertal överfall i närheten av, och på Campus under hösten 2014 så fanns behovet av att öka trygghetskänslan och öka medvetenheten om antal personer som rör sig vid Campus nattetid.

Tidigare fanns det endast en trygghetstelefon som man kunde ringa till om man kände sig otrygg. Denna trygghetstelefon hade dock begränsade öppettider och begränsad kapacitet så då fanns behovet av ett komplement till denna. En forskare vid AASS fick då idén att skapa en alternativ lösning med hjälp av en smartphoneapplikation och ljusbarriärsensorer och skapade ett forskningsprojekt runt det. Projektet är en del av ett större projekt som kallas "Take Back the Night".

Under höstterminen 2015 så gjordes ett arbete av två studenter vid kriminologiprogrammet som undersökte vad som bidrog till att folk kände sig otrygga när de vistades runt campusområdet. De undersökte hur trygga både män och kvinnor kände sig i vissa situationer dagtid, nattetid med varierande belysning. De undersökte också hur trygg de kände sig beroende på hur många andra personer som befann sig i närheten. Där det då påvisades att de som tillfrågades skulle känna sig tryggare vid användandet av en applikation som gjorde det möjligt att kunna se var det fanns andra människor och hur många de var. [1]

Målet med projektet var att skapa en applikation till smartphones som med hjälp av GPS, sensorer och en server som tar emot information från sensorerna skapa en helhetslösning som gör att personer kan känna sig tryggare när de befinner sig på och runt om Campusområdet. Informationen som servern tar emot från sensorerna och användarnas smartphones GPS-enhet används för att estimera antalet personer och deras position på campus. Detta gör att de som använder smartphoneapplikationen kan få en ökad medvetenhet om personer i närheten och utefter det ta ett beslut om vilken väg de vill ta när de går på Campusområdet.

1.2 Projekt

I detta arbetsprojekt skulle ett system bestående av en Android applikation och en server som användare kan logga in på utvecklas. Applikationen skulle med hjälp av sensorer och inloggade användares smartphones visa var på Campusområdet personer befann sig. Med sensorerna delades campusområdet in i sektioner, sedan kunde antalet personer som befann sig i en sektion estimeras. Informationen ifrån sensorerna skickas till servern där den tolkas och lagras för att sedan kunna skicka ut den till anslutna användare. I applikationen ska informationen visualiseras på en karta med hjälp av OpenStreetMap som bara inloggade användare ska ha tillgång till. Informationen från GPS och sensorer skulle även loggas.

1.3 Användningsfall

1.3.1 Mall för generella användningsfall

Nedanstående mall kan användas för att skapa användningsfall för systemet. Under utvecklingen av systemet utgick designen från ett enda användningsfall. Mallen användes för att identifiera det viktigaste användningsfallet. Som har fokus på att användaren ska känna sig tryggare genom att ha möjligheten att välja den väg på Campus som användaren känner sig tryggast med.

Vilken tid på dygnet går personen på Campus?

- Natt/Kväll
- Dag

Känner personen sig trygg?

- Ja
- Nej

Använder personen applikationen för att se var andra är?

- Ja
- Nej

Välja väg utifrån det estimerade antalet personer i sektionerna.

- Sektioner där sensorer inte gett utslag
- Sektioner där sensorer gett flera utslag

Välja väg utifrån var på Campus det befinner sig andra inloggade användare.

- Gå där det inte finns inloggade användare
- Gå där det finns inloggade användare

1.3.2 Mest relevanta användningsfallet

Detta användningsfallet sågs som det viktigaste och har varit grundläggande när systemet utvecklats.

> En person är på väg hem på kvällen/natten och ska gå över Campusområdet. Personen känner sig otrygg eftersom att det är mörkt ute. Personen öppnar appen i sin Androidtelefon och registrerar sig som användare och loggar in. När personen har loggat in skickas GPS-koordinater till servern som i sin tur skickas ut till alla andra inloggade användare. Personen tar emot sensorinformation och GPS-information om andra användare. Personen kan sedan välja en väg över Campus beroende på vart andra inloggade användare befinner sig och hur många personer som befinner sig i sektionerna.

1.4 Syfte

Syftet med projektet var att applikationens användare ska få en bättre trygghetskänsla när de vistas på och runt Campusområdet. Om användaren blir mer medveten om andra personer som vistas på Campusområdet kan användaren välja en väg som användaren känner sig trygg med. Användaren ska kunna “se” vad som händer runt ett hörn. Projektet erbjöd också att en fördjupning inom utvecklingen av mobilapplikationer, GPS-lokalisering och kommunikation med extern hårdvara.

1.5 Krav

De kraven som ställdes upp vid projektet uppstart var följande:

- Fungerande Androidapplikation som stödjer registrering och inloggning och kan visualisera en karta där man ser andra användare via GPS-koordinater, samt sensordata visualiserad i sektioner.
- En server som tar emot data från klienter och sensorer och behandlar data innan den skickar ut informationen till klienterna.
- En databas för lagring av användarkonton samt loggning av applikationsanvändning.
- Att eventuellt lagra sensor- och GPS-data på databasen för att senare ha möjlighet att analysera rörelsemönster.

1.6 Arbetsfördelning

Under projektets gång så har Anders arbetat stora delar med Androidapplikationen och Gustav arbetade mycket med servern. När problem uppstod så har vi parprogrammerat för att lösa dessa. Vi har också diskuterat olika lösningsförslag på både server och applikation innan vi har implementerat dem för att få en bra lösning som båda var nöjd med.

2 Teoretisk bakgrund: Lokalisering av mobiltelefoner

2.1 Översikt av olika metoder

Det finns ett flertal sätt som man kan använda sig av för att upptäcka smartphones inom ett område. I dagsläget används dessa metoder vid kartläggning av personers rörelsemönster. De kan också användas för att förhindra användning av mobiltelefoner där det inte är tillåtet. För projektet var det viktigt att undersöka dessa metoderna för att hitta den bästa lösningen till vårat projekt.

2.1.1 GPS

Global positioning system är en metod som använder sig av satelliter som ligger i omloppsbana runt jorden. Satelliterna används sedan för triangulering för att få fram positionen för en GPS-mottagare som befinner sig på marken eller i luften. För att kunna använda trianguleringen behöver GPS-mottagaren ha kontakt med minst fyra satelliter samtidigt. Den första 22 april 2016 fanns det 31 GPS-satelliter i omloppsbana runt jorden, för att ha täckning överallt i världen behövs ca 24 GPS-satelliter jämnt utspridda runt jorden[2]. GPS är utvecklat av den amerikanska militären och till att börja med så var det bara de som hade tillgång till full precision. År 2000 låstes den fulla precisionen upp även för allmänheten[2]. För att inte vara beroende av USA för att använda precis navigering så utvecklar nu European Space Agency ett nytt system som heter GALILEO. Detta system använder sig också av satellitnät för att beräkna positioner genom triangulering. Detta ska ge en bättre precision för Europa jämfört med GPS, som just nu har en felmarginal på högst 7.8 meter[2]. Det ger en bättre precision eftersom att satelliterna är placerade på ett sätt som ger bättre täckning i norra Europa[3]. GPS är en bra metod då de flesta moderna smartphones har en GPS-enhet installerad, men en begränsning är strömförbrukningen eftersom att smartphones GPS-enhet kontinuerligt letar efter GPS-signaler och beräknar positionen utifrån dessa. Vilket C.-H. Chen har kommit fram till i [4,s.806] “GPS has high accuracy. It requires dedicated electronics that have high power consumption, making it less suitable for mobile devices to operate for a prolonged period of time”.

2.1.2 Wifi-Lokalisering

I stället för att använda GPS för att hitta positionen för en smartphone kan Wifi användas, detta alternativ gör att man kan minska batteriförbrukningen med upp till 35% jämfört med användning av GPS med en förlust i precision. Detta då Wifi är mindre resurskrävande än GPS. Denna metod använder sig av APs (access points) för att avgöra var en användare med en smartphone befinner sig. Genom att en server analyserar signaler som skickas antingen från telefonen till APs eller från APs till telefonen kan triangulering användas för att få fram en ungefärlig position för telefonen. Servern måste veta positionen för alla APs som används vid trianguleringen. Positionen för en smartphone räknas ut med hjälp av en algoritm som kan vara baserad på bland annat hur lång tid det tar för en signal att nå mottagaren. Minst 3 APs behövs för att triangulering ska vara möjligt. Denna lösning fungerar endast där det finns APs tillgängliga.[4]

2.1.3 Wifi-monitorer

När en smartphone inte är uppkopplad till något nätverk, men har Wifi aktiverat så letar den aktivt efter tillgängliga nätverk genom att med jämna mellanrum skicka ut Wifi-paket. Genom att installera modifierad mjukvara på en router, kan routern användas som en Wifi-monitor. Genom att simulera ett nätverk så kan routern fånga upp Wifi-paket från smartphones utan att låta telefonerna ansluta till nätverket. Genom att använda telefonens MAC-adress (Media Access Control), som är unik för varje enhet och skickas med i varje Wifi-paket så kan den användas som en digital identifikation och man kan då veta när det finns människor inom området av sin router och den kommer bara räknas en gång. Denna metod har använts för att till exempel se hur många som åker på en väg under en dag[5, 6]. Detta är en billig metod då man endast behöver ha tillgång till en router om endast uppfångande av paketen är intressant och det fungerar även fast telefonen redan är ansluten till ett annat nätverk. Minst tre routers behövs om någon lokalisering skall göras, då Wifi-lokalisering måste användas.

2.1.4 Mikrovågssensorer

De olika tjänsterna som finns tillgängliga för mobiler som GSM och SMS använder sig av en bestämd frekvens. Med hjälp av en mikrovågssensor kan dessa signaler upptäckas i närheten av sensorn. Detta gör att man kan avgöra om signalerna som blir uppfångade är relevanta och inte kommer från någon annan källa. Denna metod har tidigare använts för att upptäcka förbjuden användning av mobiltelefoner i ett fängelse. Dock så ger denna metod inte någon precis lokalisering av mobiltelefonen som upptäckts. Det går heller inte att identifiera telefonen som signalerna kommer ifrån.[7]

2.2 Jämförelse av metoder

Vissa krav ställdes på metoderna för att hitta den bästa metoden för systemet. Dessa var precision, energiförbrukning, tillgänglighet & täckning, data. Där precisionen skulle ha en minimal felmarginal, men ändå kunde lokalisera en mobiltelefon. Energiförbrukningen skulle vara så låg som möjligt för att spara på batteri på mobiltelefonerna. Täckningen och tillgängligheten motsvarar radien, eller ytan som kan täckas med så få enheter som möjligt för att utföra lokalisering och om det behövdes någon extra hårdvara/mjukvara för att använda metoden. Med data menas om mobiltelefonen behöver göra lokaliseringen och skicka till servern, eller om den går att utföra på serversidan för att minska energiförbrukningen ytterligare för telefonen.

Precision

Metod	**Precision utomhus (Felmarginal)**
GPS	Inom 7.8m
Wifi-Lokalisering	Genomsnitt 18m
Wifi-Monitor	Inte tillgänglig
Microvågssensor	Inte tillgänglig

Störst krav ställdes på precisionen därför att systemet bygger på att användaren ska få en så bra uppfattning som möjligt av hur omgivningen på Campus ser ut och var personer befinner sig. GPS har den bästa precisionen med en felmarginal på högst 7.8 meter. Felmarginalen beror mycket på hur länge GPS-enheten beräknat positionen och blir mer precis under en längre körtid. Wifi-lokalisering har en genomsnittlig felmarginal på 18 meter[4]. Wifi-monitorer och microvågssensorer har inte någon precision eftersom att de bara kan upptäcka att en telefon är inom deras räckvidd.

Energiförbrukning

Metod	**Energi-förbrukning (Smartphone)**
GPS	Hög
Wifi-Lokalisering	Medel/Låg
Wifi-Monitor	Ingen extra
Microvågssensor	Ingen extra

Systemet ställde låga krav på energiförbrukning eftersom att applikationen inte kommer att användas under längre perioder. Detta eftersom att det inte tar mer än några minuter att gå över Campus. GPS har en hög energiförbrukning eftersom att GPS-enheten måste kommunicera med GPS-satelliter och göra trianguleringsberäkningar kontinuerligt. Som tidigare nämnts har Wifi-Lokalisering upp till 35% lägre energiförbrukning än GPS[4]. Med Wifi-monitorer och microvågssensorer innebär ingen extra energiförbrukning. Eftersom att dessa metoder fångar upp signaler ifrån tjänster som körs på telefonen t.ex. Skicka/ta emot SMS, söka efter nätverk o.s.v.

Tillgänglighet & Täckning

Metod	**Täckning (Radie)**	**Tillgänglighet (Extra hårdvara/Mjukvara)**
GPS	Global	Ingen (De flesta smartphones har GPS inbyggt)
Wifi-Lokalisering	Medel	Minst tre routrar
Wifi-Monitor	Medel/Hög	Minst en router och modifierad mjukvara
Microvågssensor	Mycket Liten	Minst en sensor

Kraven som ställdes på täckning och tillgänglighet var medelstora då systemet skulle täcka så stort område som möjligt med en minimal kostnad. GPS kräver ingen extra hårdvara, då systemet bygger på en smartphoneapplikation och de flesta smartphones har en inbyggd GPS-enhet. För att kunna utföra Wifi-lokalisering med bra precision så behövs minst tre routrar för trianguleringen. Eftersom telefonen som ska lokaliseras måste vara inom räckvidden för alla tre routrarna så krävs det en del överlappning. Detta gör att täckningen blir mindre än för en Wifi-monitor. För att använda Wifi-monitor krävs endast en router med modifierad mjukvara

och den kan utnyttja hela täckningen. En router har en räckvidd på ungefär 50 - 100 meter. Med täckning menas en cirkel runt routern/sensorn som har räckvidden som radie. Eftersom att mikrovågssensorer har en räckvidd på mindre än 15 meter krävs det ett stort antal för att få täckning över större områden.

Data

Metod	**Data (Server)**	**Tillgänglighet (hur kommer man åt data)**
GPS	Skickas	Programvara krävs
Wifi-Lokalisering	Skickas	Programvara krävs
Wifi-Monitor	Beräknas	Tillgänglig
Microvågssensor	Beräknas	Tillgänglig

Systemet behöver positionen för både applikationens användare och de som inte använder applikationen. Därför var det viktigt att data om positionen var åtkomlig utan att installera någon programvara på alla smartphones som skulle upptäckas på Campus. GPS och Wifi-lokalisering kräver programvara på en smartphone för beräkna position och för att skicka vidare positionen till servern. Wifi-monitorer och microvågssensorer kräver ingen programvara på mobiltelefonen eftersom att metoderna fångar upp signaler som skickas från telefonen. Med hjälp av signalerna sker lokaliseringsberäkningen på servern.

Slutsats
GPS var den metod som passade bäst för lokaliseringen av applikationsanvändarna. Eftersom att GPS har bäst precision utomhus. Vi ansåg att det inte var värt att offra precision för att förbättra energiförbrukningen. Att ha en dålig precision på ett litet område så som Campus gör att ett system som fokuserar på precis lokaliering blir oanvändbart. Då en användare endast får en ungefärlig position för andra användare, alltså personen vet endast att det finns andra där men vet inte den exakta positionen. Varken Wifi-monitor eller microvågssensorer har någon precision. Dessa metoder kan endast att upptäcka att en smartphone är inom sensorn/monitorns räckvidd. Detta gör metoderna olämpliga för att lokalisera applikationsanvändarna.

GPS har den bästa täckningen då det enda kravet på hårdvara är att en GPS-mottagare finns. Eftersom de flesta smartphones har en inbyggd GPS-enhet så är det en metod som inte har några extra kostnader, medan de övriga metoderna skulle medföra detta. För data var metodvalet irrelevant vid lokaliseringen av inloggade användare. Då en applikation skulle utvecklas som lokaliserar och kommunicerar med en server.

För lokalisering av personer som inte använder applikationen behövs en metod som inte kräver att programvara installeras på dessa personers smartphones. För detta ändamål passar varken Wifi-lokalisering eller GPS. Det skulle krävas ett stort antal microvågssensorer för att få täckning i en sektion, detta gör metoden opraktisk. Av de fyra metoder som undersökts skulle Wifi-monitor passa bäst eftersom att de har en bra räckvidd och inte kräver någon programvara på mobiltelefonen. Det skulle räcka med en router per sektion för att täckning. Men eftersom att metoden kräver modifierad mjukvara på routern vilket det inte fanns tid för utveckling av passar metoden inte. Därför användes ljusbarriärsensorer istället. Ljusbarriärsensorerna kan inte visa positioner för personerna i sektionerna. Det är bara antalet personer som befinner sig i en sektion som kan estimeras.

3 Metoder och verktyg

3.1 Programmeringsspråk & API

För klienten så användes Android API 23 med bakåtkompatibilitet till och med API 15. Vi valde detta då det skulle bidra till att ungefär 97% av alla Androidtelefoner kunde använda applikationen. Vi valde att programmera applikationen i Android eftersom att vi båda använder Androidtelefoner. För visualiseringen på kartan så användes OpenStreetMap [8]. I början var informationen tänkt att komma från sensorer som skulle placeras på utvalda platser på Campusområdet. Då dessa blev försenade under hela projektet skapades istället ett program för att simulera sensorerna.

För kommunikationen mellan applikationen och servern så behövdes ett kodbibliotek där meddelandetyper definieras som skulle finnas tillgänglig på både klient- och serversidan. För all dokumentation i koden så användes Javadoc-kommentarer för att få en bättre översikt och tydligare kod.

3.2 Metoder

Vi har jobbat enligt Rapid Prototyping[9] som innebär att man snabbt bygger prototyper av systemet för att lättare få förståelse för hur systemet designas på bästa sätt. Vi har även använt inslag av SCRUM då vi använt en SCRUM-tabell för att få en bättre översikt på hur utvecklingen låg till. Parprogrammering har använts då speciellt avancerade implementationer gjorts eller buggar uppstått.

3.3 Verktyg

Utvecklingsmiljön för implementationen av servern var i IntelliJ IDEA och språket Java, Android studio för smartphoneapplikationen och språket Android. Android Studio är en utvecklingsmiljö som är utvecklat av Google och är baserat på IntelliJ IDEA, förutom att det har större fokus på utvecklingen av applikationer för Android-telefoner. Det har också en inbyggd simulator där man kan testa sin applikation utan att ha tillgång till en smartphone.

Vi valde att använda SQLite för att det är enkelt att sätta upp en databas. Det passade också bra då vi endast tänkt använda databasen från serverdatorn och inte från nätverket, vi behövde alltså inte en server för databasen. Vi trodde inte heller att vi skulle ha speciellt mycket trafik eller mycket data lagrat i databasen. Programmeringen skedde på egna datorer som körde Windows.

Då vi behövde ett verktyg för att kunna visualisera användare och sektioner på en karta så blev vi rekommenderade att använda OSM (OpenStreetMap) av vår handledare. Det är en gratis och gemenskapsdriven karttjänst där kartografer ifrån hela världen håller kartorna uppdaterade. OSM är helt öppen för användning, så länge som OSM anges som källa. Sedan finns OSMDroid som är ett hjälpmedel för utveckling av Androidapplikationer. Vi använde detta istället för t.ex. Google Maps just för att tjänsten är öppen om några förändringar behövdes göras på kartan så hade vi rättigheterna att göra de förändringar som krävdes.[8] För visualiseringen av kartan i applikationen så användes den inbyggda kartvyn i OsmDroid där man kan lägga på overlays. OsmDroid har en inbyggt GPS-tjänst som gör att man kan hämta enhetens kontinuerligt uppdaterade GPS-koordinater. OsmDroid erbjuder också en samling ikoner för att representera olika positioner på kartan, till exempel en “person”-ikon eller en pil som visar rörelseriktningen vid användandet av GPS.

Vi har använt GIT som versionshanterare och källkoden till projektet har laddats upp och lagrats på GIThub. Vi valde att använda GIT eftersom att vi har använt det tidigare och det blir smidigt då flera personer arbetar med samma filer samtidigt.

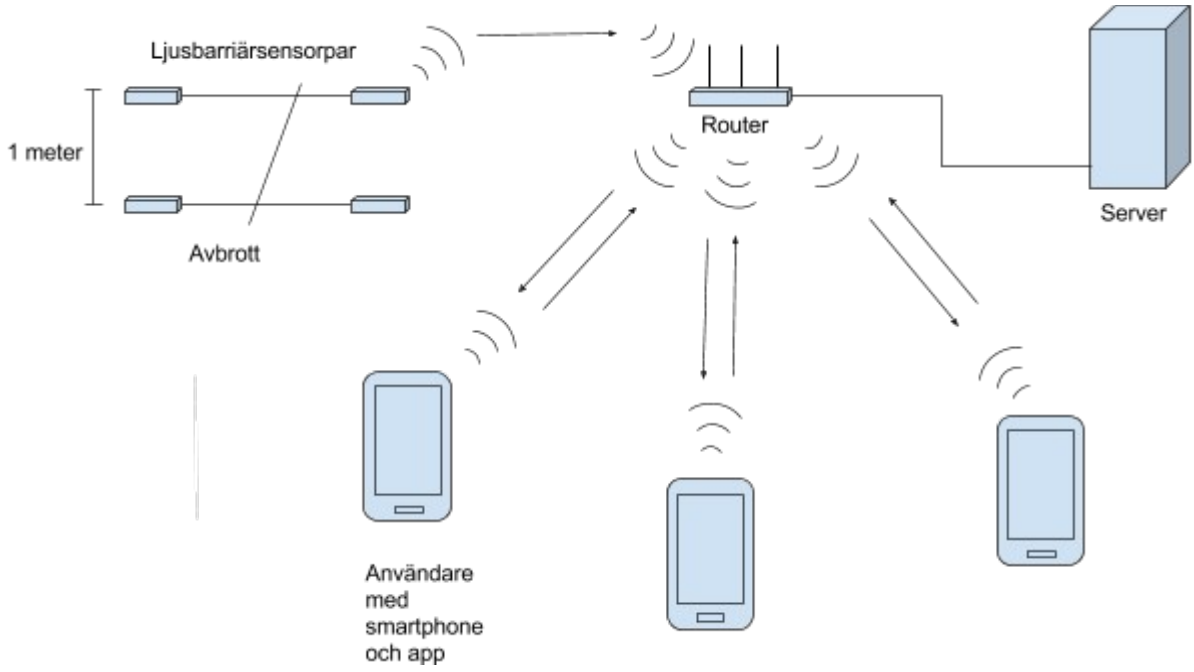
3.4 Övriga resurser

De resurser som planerades att användas var sex stycken ljusbarriärsensorer från ABUS för att kunna avgöra när personer passerade in och ut ur sektionerna. Sensorerna vart försenade och användes inte i projektet. Servern installerades på en dedikerad dator på Campusområdet. Sensorerna och servern var anslutna till Guestnet som är ett öppet nätverk på Campusområdet. Både servern och sensorerna har en statisk IP-adress på nätverket. De verktyg som att användes för testning var Androidtelefoner, en Google Nexus 5X och en Samsung Galaxy S6, samt en Samsung Galaxy S2.

4 Genomförande

4.1 Översikt

Hela projektet består av en server och en applikation. Servern fungerar efter en klient/server-modell, det vill säga att servern är en central komponent och alla klienter kopplar upp sig mot den. Applikationen får information skickad ifrån servern och visualiserar denna informationen. Efter att klienten skickat sin position tills servern skickar servern tillbaka information om hur sektionerna är placerade genom exakta GPS-koordinater, och hur många personer som estimeras befinna sig i sektionerna, samt positioner från alla andra inloggade användare. All utdaterad data städas upp på servern med jämna mellanrum för att minska/eliminera risken för krascher. När en uppkoppling sker mot servern så skapas en dedikerad tråd för att hantera kommunikationen med klienten. Tråden finns kvar så länge som klienten inte stänger sin uppkoppling. Figur 1 beskriver en översikt på hela systemet, där det visar att sensorer är placerade i par med 1 meters mellanrum mellan de båda sensorerna i paret och att ett avbrott för båda sensorerna behöver ske för att det ska ge ett resultat på servern.



Figur 1. Översikt över hela systemet

4.2 Designval

Delen handlar om viktiga designval som gjordes innan och under implementationen av systemet och hade stor påverkan på det slutliga resultatet för projektet.

4.2.1 Delning av data

Om en användare kunde ta del av all information utan att dela med sig av sin egen position kan det leda till att systemet inte kan ge lika precis information. Det kan också leda till ökad risk för att applikationen används med fel avsikter, till exempel att använda applikationen för att leta reda på personer för att överfalla dem. Metoden, som går ut på att en användare måste dela med sig av sin data för att ta del av andras data var en idé vi hade för att både öka säkerheten genom att en användare som har fel avsikter med applikationen också kommer att synas på kartan, samt att hela systemet bygger på att användare bidrar med information. Så för att en användare ska få ta del av informationen så måste denne också bidra med sin egen information i form av GPS-koordinater. Om en användare ser andra på kartan så är personen i alla fall beredd på att någon annan kan vara där. Så om man måste dela med sig av sin position för att ta del av data så tros risken för felaktig användning kommer minska.

4.2.2 Anonymt

Det enda som visas på kartan är en inloggad persons position om denne befinner sig på campus eller runt om campus. Ingen information förutom den e-mail som personen registrerar som sitt användarkonto sparas. e-mail är inte heller något som delas med andra inloggade personer, den används endast för att identifiera en person vid inloggning. Vi valde att använda denna metod eftersom att vi inte såg några fördelar med att visa ytterligare information om en person i applikationen. Om man visar mer information om en person i applikation kan denna information i värsta fall användas för att välja ut ett offer om man använder applikationen med fel avsikter.

4.2.3 Snabbt & Simpelt

Vi har valt att göra Androidapplikationen så simpel som möjligt för att den ska vara lätt och snabb att använda. Därför finns det bara en inloggningsskärm och en skärm med kartan. Det finns inte heller någon mer funktionalitet på kartskärmen mer än att man kan se sensorinformation och andra inloggade användares positioner.

4.2.4 DAO-Modellen & PreparedStatement

Kommunikationen med databasen när användare hanteras sköts enligt Data Access Object-modellen. Det innebär att ett interface skapas som visar hur ett objekt hanteras vid kommunikation med databasen. Detta gör att det är enkelt att byta ut den befintliga databasen mot något annat lagringssätt eller en annan databas. Det enda som behöver göras är att skapa en ny klass som kommunicerar med det nya lagringsmediet. PreparedStatement används vid SQL-frågor, detta förebygger SQL-injektion. SQL-frågan parsas och kompileras först utan någon data från användaren. Detta gör att data som användare skrivit in inte misstolkas som kod som ska köras. Data läggs sedan till för att SQL-frågan ska kunna exekveras mot databasen.[10]

4.2.5 Externt kodbibliotek

För kommunikationen mellan servern och applikationen så valde vi att använda serialiserade objekt och objektströmmar för att snabbt och enkelt kunna börja skicka meddelanden. Det bidrog till att både servern och applikationen behövde tillgång till de objekt som serialiserades. Vi behövde implementera ett externt kodbibliotek som innehöll alla meddelanden som skickades mellan klient och server. Båda projekten importerade biblioteket för att göra detta möjligt. Ett annat sätt att implementera meddelandekommunikationen skulle till exempel vara att använda JSON på både server och klient. som konverterar objekt till ett läsbart textformat och sedan skickar meddelandet som en sträng. Vi valde att använda ett externt kodbibliotek då vi inte hade tidigare erfarenheter av JSON och vi ansåg att tiden inte räckte till för att lära oss det.

4.2.6 Boundingboxes & låst MapView

För att minimera data som skickas till och från servern har vi låst kartan så att en användare bara ser campusområdet. Vi använder också två boundingboxes, en inre och en yttre. Den inre boundingboxen gränsar av området som är synlig på kartan, koordinater till de användare som är utanför den inre boundingboxen skickas därför inte ut till andra användare. Användare som är utanför den yttre boundingboxen får inte heller ut någon information om sensordata eller positionen för andra användare. Vi har låst vår MapView till att bara visa campusområdet eftersom att allt som är relevant för användaren visas där, detta gör också så att så lite data som möjligt behöver laddas ner till telefonen.

En annan lösning skulle vara att ha upplåsta zoom-nivåer så användaren kan välja själv en nivå som fungerar bra på användarens smartphone. Vi testade detta, men då lösningen inte blev bra och vi inte var nöjda med den. Vi ändrade senare så zoom-nivån automatiskt justeras efter telefonens upplösning, vilket var en lösning som fungerade bra.

4.2.7 Funktionalitet på telefoner med olika upplösning

Till att börja med hade vi tänkt att låsa zoomen på kartan till högsta nivån men efter test på äldre telefoner med lägre upplösning insåg vi att man bara kunde se en liten del av kartan om man har lägre upplösning. Vi löste detta genom att ställa in zoomnivån automatiskt beroende på telefonens upplösning. Texten som visar hur många personer som befinner sig i en sektion på campus är inte dynamisk så den vart så stor att den inte syntes på telefoner med låg upplösning och för liten på telefoner med hög upplösning. Detta löste vi genom att sätta textstorleken beroende på telefonens upplösning.

4.2.8 Färgval

Vi har valt att göra sektionerna på kartan gröna eftersom att färgen grön vanligtvis förknippas med att det är säkert som vid till exempel trafikljus. Vi har inte använt oss av röd eller liknande färger då röd och gul ofta förknippas med fara och det hade passat dåligt i vårat system då meningen är att användarna ska känna sig tryggare. [11]

Vi funderade också på att använda färgkodning i sektionerna så att de skulle byta färg beroende på hur många personer som befinner sig i sektionen. Men vi kom fram till att det var mycket tydligare att skriva antalet personer med siffror, man slipper då också beskriva hur många personer de olika färgerna representerar.

4.2.9 Executors & CachedThreadPool

För att hantera den samtidiga kommunikationen mellan alla klienter och servern så använde vi oss av en ExecutorService. Det är det säkraste alternativet för trådprogrammering eftersom man överlåter trådhanteringen till Java istället för att manuellt hantera och skapa trådar. Först använde vi en FixedThreadPool till serverns ExecutorService, d.v.s att det bara fanns ett begränsat antal trådar som kunde hantera kommunikation. Detta begränsade antalet användare som systemet kunde hantera och gick ej att utöka utan att göra ändringar i källkoden. Vi ändrade detta senare så servern använde en CachedThreadPool istället, vilket är en dynamisk trådpool. Den skapar nya trådar när behovet för fler trådar finns. Vi valde det andra alternativet då vi ville att servern skulle vara dynamisk och kunna hantera många användare, men inte tillräckligt många användare för att att en CachedThreadPool skulle vara ett problem istället då den inte har någon inbyggd begränsning på hur många trådar som kan skapas.

4.2.10 Definition av sensorområden

För att det ska vara lätt att utöka antalet sensorer och sektioner för systemet har vi valt att definiera upp dessa i tre olika XML-filer som sedan läses in av servern. De tre filerna är en för sektioner som definierar sektionernas exakta GPS-koordinater. En för sensorpar, som definierar vilka sensorer som hör ihop i par och vilken sektion som paret hör till. Den sista filen definierar sensorerna och vilken IP-adress som servern kopplar upp sig mot för att börja kommunicera med sensorn. Ingen kod i systemet måste läggas till eller ändras för att de nya sektionerna och sensorerna ska fungera och visas på kartan i Androidapplikationen. Först använde vi vanliga textfiler för att utföra detta, men efter implementationen av den metoden upptäckte vi hur svårt det var att lägga till nya sensorer eller sektioner utan att det blev fel. Därför gjordes detta om till den aktuella versionen med XML-filer. Vi bestämde oss för att sensorerna inte ska ha någon bestämd position. Detta gör att bara sektionen finns definierad med GPS-koordinater för alla hörn, genom detta så kan sensorpar enkelt läggas till bara man definierar vilken sektion de tillhör.

4.2.11 Sensorsimulator

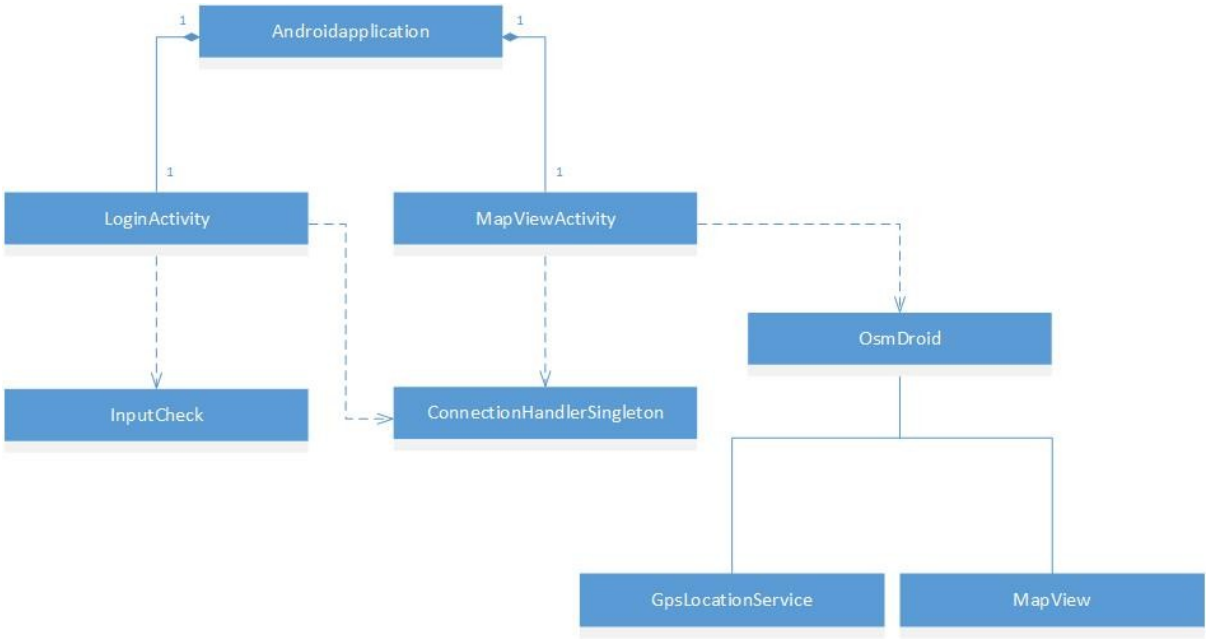
Då ljusbarriärsensorena som skulle användas blev försenade så var vi tvungna att implementera något temporärt för att kunna implementera och test sensorlogiken i systemet. Så därför implementerade vi en simulator som kopplades upp mot servern genom 5 localhost sockets, där varje socket simulerade två sensorer och hade två olika värden som den kunde skicka. Detta till skillnad från de riktiga sensorerna, där en sensor kopplar upp mot en socket. Så för att simulera sensorerna så hade socketen 50% chans att skicka ett meddelande till servern, varje 5-20 sekunder och sedan 95% chans att skicka det andra meddelandet. Vi gjorde så för att testa att servern kunde hantera både att båda sensorerna i ett par hanteras korrekt, men också att servern kunde hantera ifall den fick in skräpvärden, eller att bara en sensor i paret skickar ett värde.

Vi bestämde att om en sensor i ett par skickar ett värde och den andra inte skickar sitt värde inom fem sekunder så sågs det som ett skräpvärde och raderas. Genom att varje gång en sensor skickar en signal så söker servern igenom en lista med förväntade värden och tidpunkter, om den hittade ett förväntat värde som motsvarade värdet som sensor skickade så hade vi fått en lyckad signal och metoden för att lägga till/ta bort i sektioner anropades. Om det förväntade värdet inte fanns i listan hämtades lades värdet för den andra sensorn i paret in i listan och sparade tidpunkten som det gjordes. Varje gång som ett nytt värde kommer in söks listan med förväntade värden igenom först och varje förväntat värde som är äldre än 5 sekunder flaggas att det är utdaterat och raderas nästa gång uppstädning av sensordata sker.

4.3 Implementation

I detta kapitel beskrivs till en början i 4.3.1 implementationen av Androidapplikationen, implementationen av servern beskrivs senare i 4.3.2.

4.3.1 Applikation



Figur 2: Klass-diagram för Androidapplikationen.

Applikationen skrevs i Android med Activities. En Activity är en klass som tar hand om en aktivitet som användaren kan utföra i applikationen. I våran applikation har vi två Activities, en för inloggning och registrering och en för att visa kartan med positioner och sensorinformation se figur 2. MapViewActivity använder biblioteket OsmDroid som innehåller metoder för att hämta enhetens GPS-koordinater samt att visa en karta i smartphonen. På kartan kan enhetens position visas och overlays kan läggas till. I ett overlay kan ikoner läggas till. Ikonerna kan användas för att visa sektioner, antal personer som befinner sig i sektionerna samt inloggade personers positioner. Layouten till de olika vyerna lagras och kan redigeras i XML-filer.

4.3.1.3 Inloggning och registrering

Varje gång applikationen startar kontrolleras tillstånd att använda telefonens GPS och lagringsenhet. Om något av dessa två tillstånd inte är beviljade uppmanas användaren att ge tillstånd till applikationen. Tillstånden krävs för att användaren ska kunna logga in.

Användarens GPS måste vara aktiverad för att logga in. Vid inloggning körs kontroller för att se till att användaren har skrivit en e-mailadress som matchar en regex och ett lösenord som inte är för kort. Vid registrering kontrolleras även att användaren har angett samma lösenord två gånger. Om kontrollerna är lyckade skickas ett meddelande till servern med e-mailadress och lösenord. Om inloggningen eller registreringen lyckas startar applikationen MapActivity. E-mailadressen som används när användaren loggar in sparas och fylls i automatiskt nästa gång användaren startar applikationen. Genom att trycka på registreringsknappen kan användaren växla mellan inloggningsläge och registreringsläge.

4.3.1.2 Kartvyn

När 'MapActivity' skapas ställer den in kartan så att den är inzoomad på Campusområdet. Zoom-nivån och storleken på text ställs in så att den passar telefonen som applikationen körs på, detta görs beroende på telefonens skärmupplösning. Scrollning och zoomning låses för att all relevant information redan finns på skärmen. En GPS-tjänst startas för att kontinuerligt hämta enhetens GPS-koordinater. Tjänsten kopplas sedan till ett overlay som håller en ikon på kartan uppdaterad för att visa användarens position.

För att skapa och visa sektionerna skapas ett overlay som heter polygon som sedan ritas ovanpå kartan. För att skapa en polygon definieras polygonens alla hörn med GPS-koordinater, detta betyder att applikationen stödjer sektioner med godtyckligt antal hörn. När eventet för ny position triggas av GPS-tjänsten skickas ett meddelande till servern innehållande de nya GPS-koordinaterna. Ett nytt meddelande skickas endast om det har passerat minst 2 sekunder sedan senaste meddelande med koordinater skickades. Efter koordinaterna skickats till servern väntar applikationen på att servern ska skicka tillbaka sensordata och andra inloggade användares GPS-koordinater. GPS-koordinaterna till de andra användarna läggs sedan i ett overlay som visar användarnas position på kartan. För att visa antalet personer som befinner sig i en sektion skrivs en siffra i respektive sektion. Informationen om andra användares position och antalet personer i sektionerna uppdateras varje gång applikationen tar emot den informationen från servern. Genom att trycka på bakåtknappen loggas användaren ut från servern.

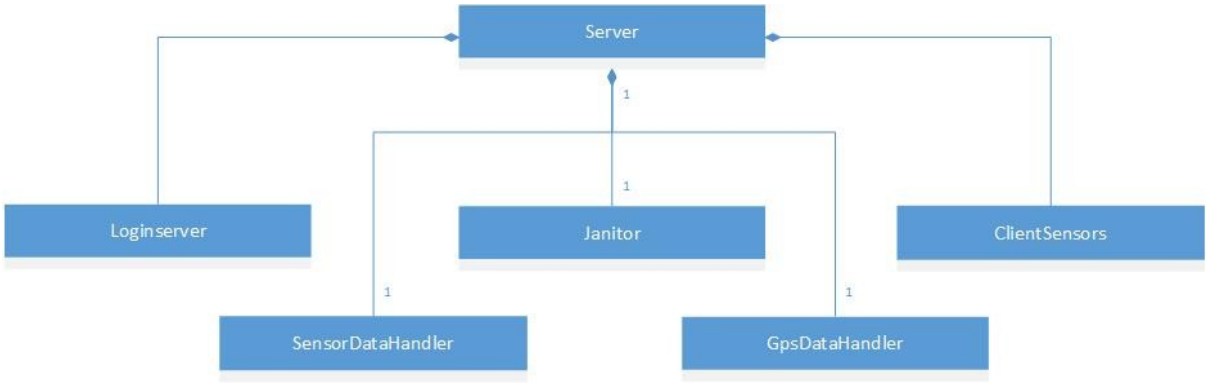
4.3.1.1 Nätverkskommunikation

Kommunikationen med servern sköts av klassen ‘ConnectionHandler’ som är en singleton. Klassen hanterar uppkoppling och nedkoppling mot servern. När en activity skickar ett meddelande startas en tråd som kallar på en ‘sendMessage’ i ‘ConnectionHandler’. När ett meddelande har skickats väntar tråden alltid på ett svar från servern. Svaret returneras sedan till den activity som meddelandet skickades från.

4.3.2 Server

Figur 3 visar vilka fem huvudklasser servern består av. ‘LoginServer’ är klassen som hanterar all kommunikation med smartphoneklienter. ‘Janitor’ hanterar all uppstädning av data för att minska den mängd information som skickas till klienter. Klassen som hanterar all kommunikation med sensorerna är ‘ClientSensors’ och de två klasserna ‘SensorDataHandler’ och ‘GpsDataHandler’ hanterar all logik för information från smartphone-klienter och sensorer.

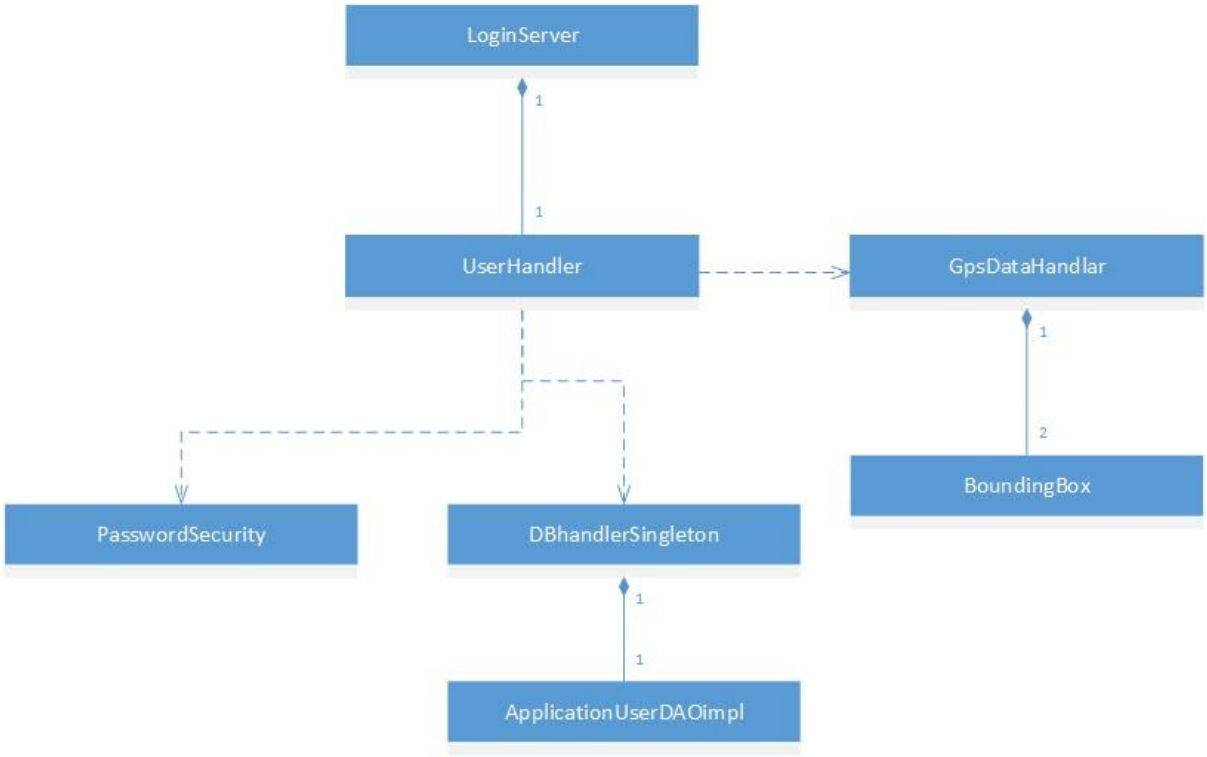
.



Figur 3.Klass-diagram för Server

4.3.2.1 Lösenordskryptering

När ett inloggningsförsök eller en registrering av ett nytt konto görs på servern så används den statiska klassen ‘PasswordSecurity’ som hanterar kryptering av lösenord, se figur 4. Ett användarnamn och lösenord skickas som parametrar till funktionen. Först så konverteras lösenordet till en byte-array, sedan så genereras ett så kallat salt som också är en array på 64 byte, som genereras genom den säkra slumptalsalgoritmen ‘SHA1PRNG’. De två arrayerna konkateneras sedan för att bilda en större array. För att kryptera lösenordet så används ‘SHA-256’-algoritmen (Secure Hashing Algorithm) vilket genererar en array på 32 byte. Denna slutgiltiga array konverteras sedan till hexadecimalt för att sparas på databasen som text. Samma sak görs för saltet, det konverteras också till hexadecimalt och sparas på databasen. När ett inloggningsförsök sker så hämtas lösenord och salt ifrån databasen också det lösenordet som användaren matat in, och saltet ifrån databasen konverteras till byte för att sedan använda samma funktioner som vid den första krypteringen. Lösenorden jämförs sedan, och om användarnamnet, lösenordet ifrån databasen och den nya krypteringen matchar så lyckas inloggningsförsöket och användaren och blir inloggad.[12]



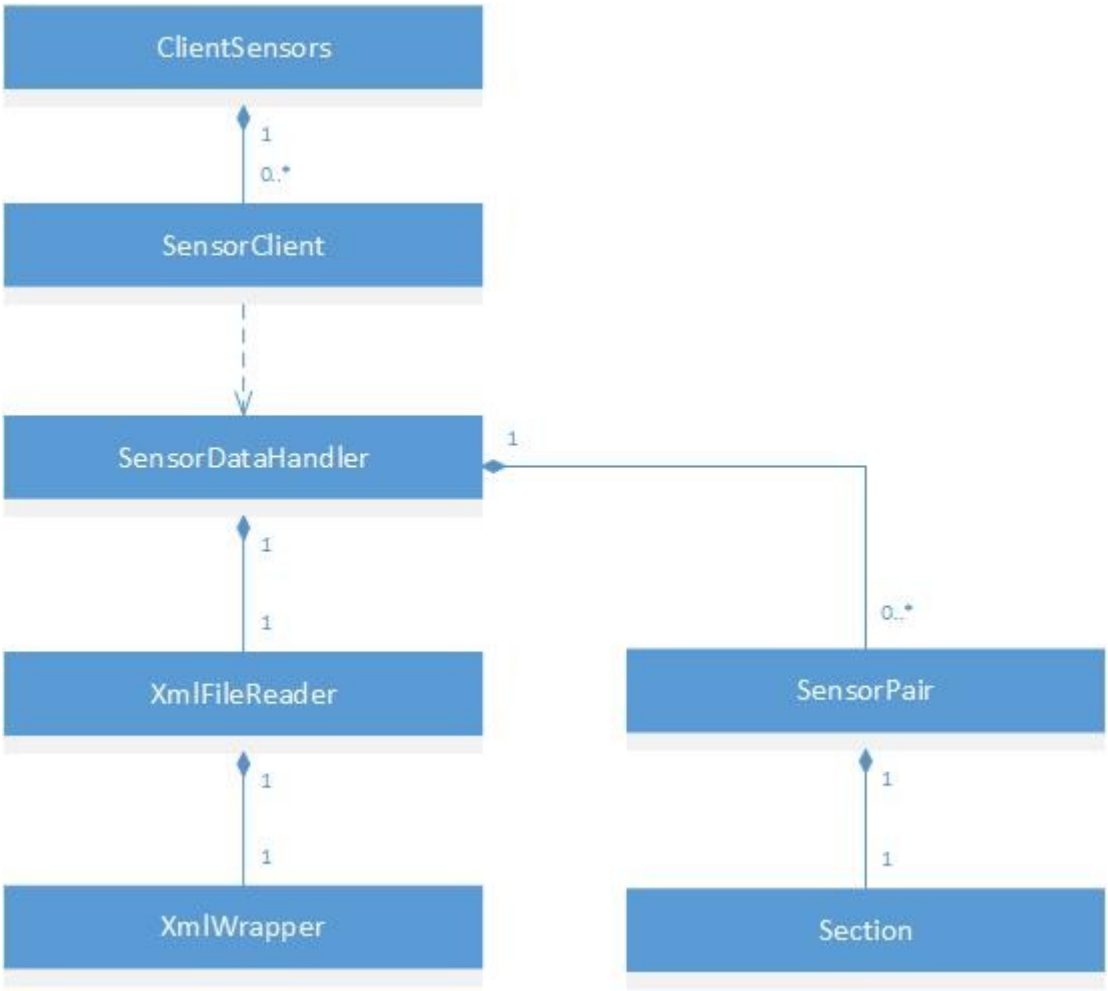
Figur 4.Klass-diagram för LoginServer

4.3.2.2 Sensor- & GPS-logik

För att göra systemet dynamiskt utökningsbart. Där det var enkelt att lägga till nya sensorer och sektioner utan att behöva ändra i koden. För att lösa problemet implementerades XML-filer som servern läser in vid uppstart. Dessa filer innehåller information om hur sektionerna och sensorerna är kopplade och vilka relationer de har till varandra. Relationerna kopplas via id-nummer, där filen för sensorerna definierar vilka sensorer som hör ihop i ett par, vilken sektion som är den som sensorparet hör till och vilken sektion paret gränsar till. Eftersom ett par bara kan höra till en sektion, medan en sektion kan ha flera sensorpar så behövs det bara definieras en gång och på ett ställe. Om sensorparet inte gränsar till en annan sektion så sätts ‘-1’ som id för gränsande sektion. I filen för sektioner så definieras vilka exakta GPS-koordinater som sektionens alla hörn har och vilket id sektionen har. I den tredje och sista filen definieras alla enskilda sensorer med vilket id de har och vilket IP-adress för uppkopplingen ifrån servern.

Denna lösning användes istället för att leta på och sätta exakta GPS-koordinater för varje enskild sensor. Det är inte relevant för systemet att veta exakt vart varje sensor finns, utan bara vilken sektion som den tillhör. Så tar systemet hand om logiken för att lägga till/ta bort personer ur sektionerna utan att känna till exakt var ett sensorpar gav ett utslag. Lösningen gör det enkelt att lägga till nya komponenter eftersom det finns tydligt i en XML-fil. En readme-fil skapades som beskriver exakt hur nya komponenter läggs till. Readme-filen återfinns i Bilaga A.

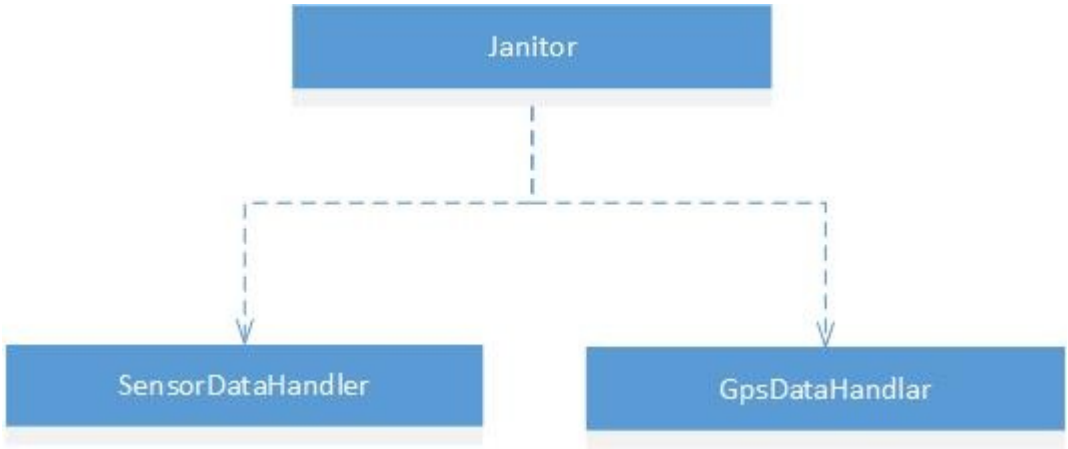
Uppkopplingen mot sensorerna sker vid uppstart av servern då servern läser in IP-adresserna från XML-filen. Servern skapar därefter så många ‘SensorClient’ som finns i XML-filen. Sensorklienterna som skapas har en referens till klassen där sensordata sparas, se figur 5.



Figur 5. Klass-diagram för ClientSensors

4.3.2.3 Uppstädning i datahanterare

Då det finns ett behov av servern att kunna ta hand om stora mängder data under en kort tid och att data kan ses som utdaterad fort i ett system som hanterar hur människor rör sig. Då en människa rör sig i ungefär 4.6 km/h, alltså ungefär 1.2 m/s [13] så kan en person röra sig en stor sträcka på kort tid, och eftersom campusområdet endast är ungefär 600x300 meter så för att gå den korta sträckan på 300 meter tar (300*1.2)/60 = 6 minuter. För att rensa bort gammal data så implementerades klassen 'Janitor' som startar en tråd som har tillgång till sensordata och en tråd som har tillgång till GPS-data ifrån klienterna, se figur 6. Båda dessa trådar är bakgrundstrådar, så om någon annan tråd arbetar med någon av dessa listor med värden så väntar uppstädningen på sin tur, samt att trådarna stängs ned om servern stängs ned utan att de fått köra färdigt. Den klassen går in i vardera dataklasser och kör funktionen för uppstädning, sedan så sover den tråden en specifik tid innan nästa försök att städa data. För GPS-data så försöker den städa upp i data en gång varje minut, medan den försöker städa upp sensordata varannan minut. Det finns ingen anledning att spara utdaterad data i minnet då all information loggas till databasen.



Figur 6. Klass-diagram för Janitor

För att en GPS-koordinat ska ses som utdaterad så ska det vara mer än två minuter sedan servern lagrade den. Detta kan vi anta då applikationen ska skicka sina GPS-koordinater varje gång var 3-20 sekund, så om inget nytt kommer in på 2 minuter så kan vi också anta att användaren inte längre använder applikationen och dess position som visas för andra kan raderas.

När en sensorbarriär bryts och skickar en signal till servern så läggs det värde för den andra sensorn i paret till i en lista över förväntade värden som bör skickas till servern. Sensorerna i ett par är placerade med ungefär 1 meters mellanrum. Om detta värde i listan inte kommer in inom 5 sekunder så flaggas värdet för borttagning och anses som utdaterat. Sedan när uppstädning anropas på sensordata så raderas värden som är flaggade för borttagning, samt

alla värden som är äldre än 5 sekunder och inte blivit flaggade än. Om någon listorna med förväntade värden, eller GPS-koordinater är tom så händer ingenting förutom att tråden somnar igen.

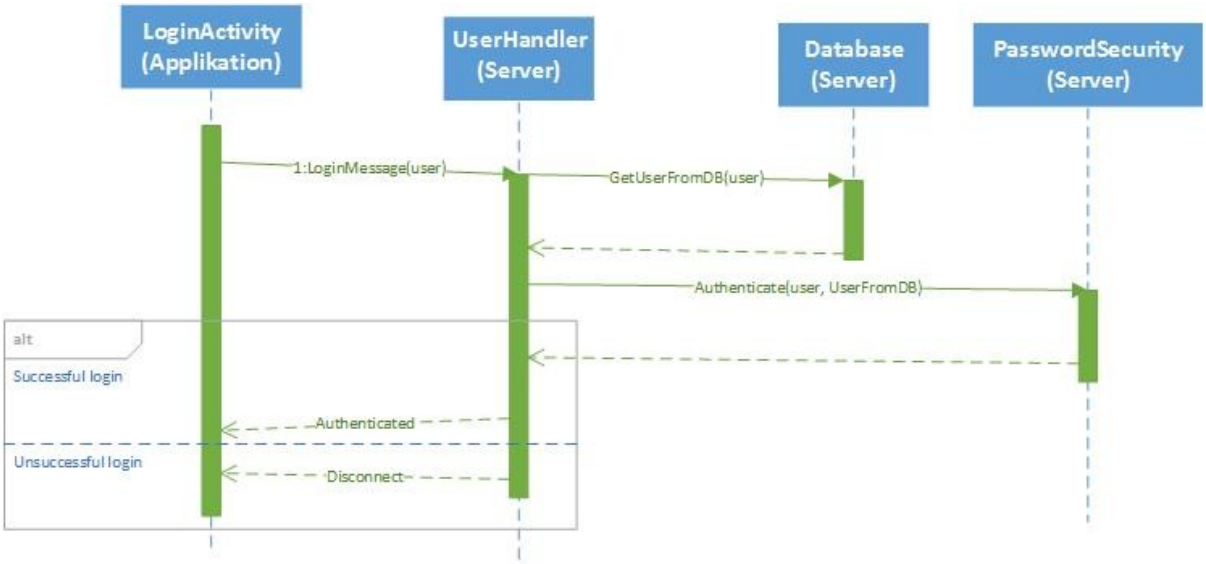
4.3.2.4 Nätverkskommunikation

Kommunikationen på serversidan sker genom att en server startar upp så startar också ny tråd med en lyssnare på port 8181 som lyssnar efter uppkopplingar. Denna lyssnare körs under hela programmets gång. När en uppkoppling sker så startar serverns trådpool automatiskt en ny tråd mot uppkopplingen för att kunna hantera kommunikationen med klienten, se figur 4. Tråden finns kvar så länge som klienten fortsätter att vara uppkopplad och försöker läsa meddelanden om inte servern har påbörjat sin nedstängningsprocess. När ett meddelande kommer in skickas det till en funktion som hanterar meddelandet. Sedan svarar klienten med ett annat meddelande. De meddelanden som används på både server- och applikationssidan skickas som serialiserade objekt och är följande:

Message - Superklass för meddelanden. Alla klasser utom ‘ServerMessage’ ärver ifrån ‘Message’. Klassen har endast en sträng som tar en e-mail från användaren då den används i alla meddelandetyperna. E-mailadressen används som en identifikation på servern vid mottagna meddelanden.

LoginMessage - Utöver e-mailadressen så skickas även en sträng för lösenordet med till servern för att kunna utföra ett inloggningsförsök.

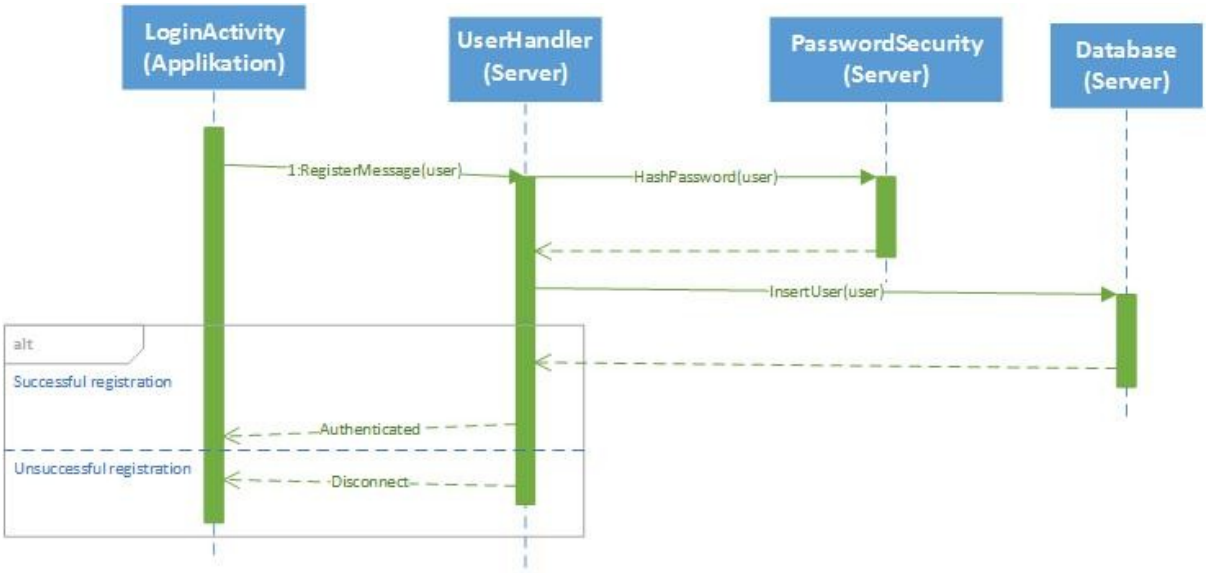
Tar servern emot ett ‘LoginMessage’ så försöker servern hämta en användare och dess lösenord ifrån databasen och krypterar lösenordet på samma sätt som vid registrering och jämför sedan dessa. Vid matchande lösenord svarar servern med ‘Authenticated’, annars ‘Disconnect’, se figur 7.



Figur 7. Sekvens-diagram vid ett inloggningsförsök

RegisterMessage - Precis som ett ‘LoginMessage’ så skickas en sträng för lösenord för att kunna registrera ett konto. Föregående meddelande och detta ser likadana ut, men utför olika saker på serversidan. Vi gjorde så då vi tyckte det var en tydligare lösning istället för att ha ett meddelande som utförde båda operationerna. Eftersom att ‘RegisterMessage’ och ‘LoginMessage’ är olika klasser så kan servern kolla vilken sorts instans objektet är instansierat som och därför vet servern vilket sorts meddelande den läst in och kan därefter utföra olika operationer.

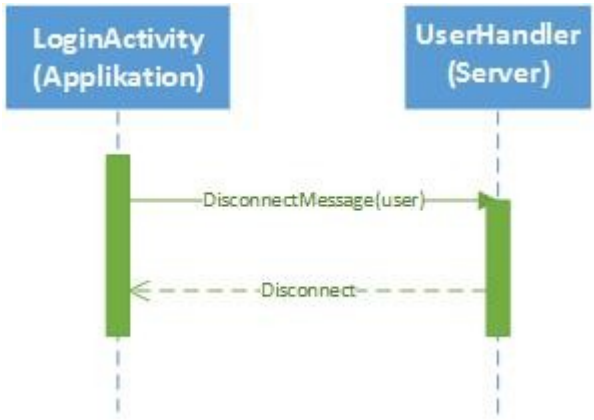
Om servern tar emot ett ‘RegisterMessage’ så anropas ‘PasswordSecurity’ för att generera salt, kryptera lösenord och sedan spara användarkontot på databasen. Vid lyckad registrering så svarar servern med ‘Authenticated’ och vid misslyckad registrering ‘Disconnect’, se figur 8.



Figur 8. Sekvens-diagram vi ett registreringsförsök

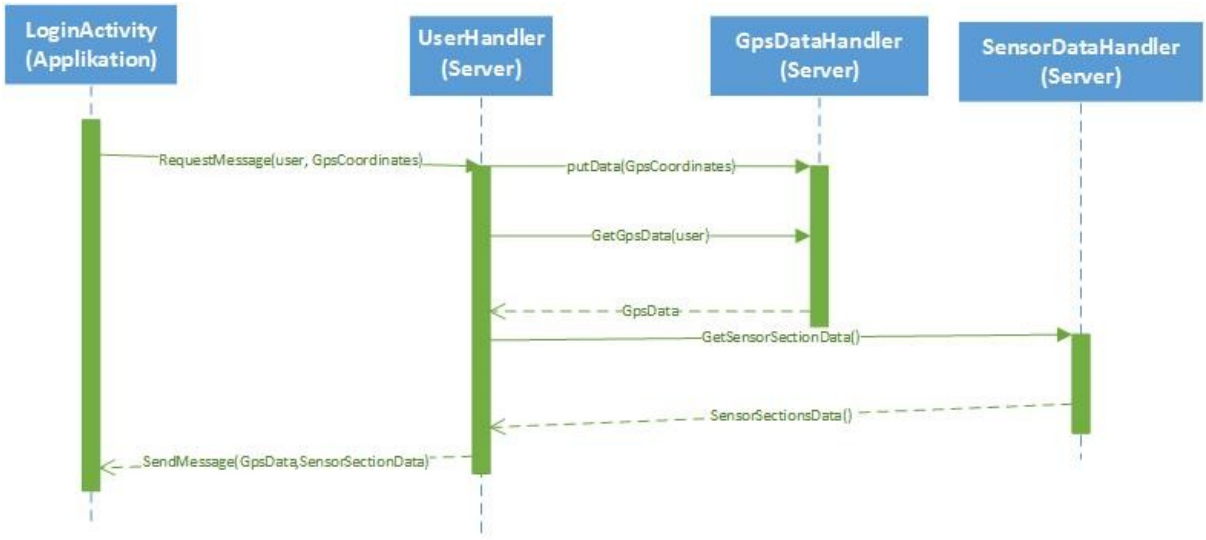
DisconnectMessage - Klassen används när en klient vill stänga ner sin uppkoppling till servern. Meddelandet skickar bara en e-mailadress, men återigen så var det en tydligare lösning istället för att skicka ett vanligt meddelande.

Tar servern emot ett ‘DisconnectMessage’ så svarar servern med ett meddelande med ‘Disconnect’ tillbaka för att bekräfta nedkopplingen, se figur 9.



Figur 9. Sekvens-diagram för Disconnect

RequestMessage - Meddelandet skickas till servern när klienten vill ha tillgång till data ifrån sensorer och andra klienter. Eftersom klienten måste bidra med sin egna data för att kunna ta emot data så skickas GPS-koordinater med i meddelandet, samt e-mail eftersom klassen ärver av ‘Message’.



Figur 10. Sekvens-diagram vid ett RequestMessage

ServerMessage - Detta meddelande är det enda som servern svarar med. Den har en enumerator för att beskriva vilken sorts meddelande som skickas. Sedan har den möjligheten

att skicka med ett objekt. Det vanligaste som skickas är en sträng för att meddela klienten om hur deras meddelande hanterades på servern, men vid ett ‘RequestMessage’ så skickas istället ett ‘SensorDataObject’, se figur 10.

SensorDataObject - Objektet skickas med i ’ServerMessage’ när en klient vill ha tillgång till data. Den har en lista med alla GPS-koordinater från alla inloggade klienter, utom den klienten som ville ha tillgång till data, samt en hashkarta med information ifrån de sektioner som sensorerna är kopplade till.

Uppkopplingen mot klienten finns kvar så länge som servern inte får in ett meddelande om att stänga ned från klienten. Om servern har påbörjat sin nedstängningsprocess så svarar servern alltid med nedstängning och meddelar klienten om detta oavsett vilket meddelande som kommer in. Servern stänger också ned uppkopplingen till klienten om något fel inträffar, till exempel att Androidapplikationen kraschar, ett okänt meddelande kommer in eller ett misslyckat inloggningsförsök/registreringsförsök gjordes. Servern stänger endast ned om ett större fel inträffar, eller om en administratör anger att den ska stängas ned.

4.4 Testning

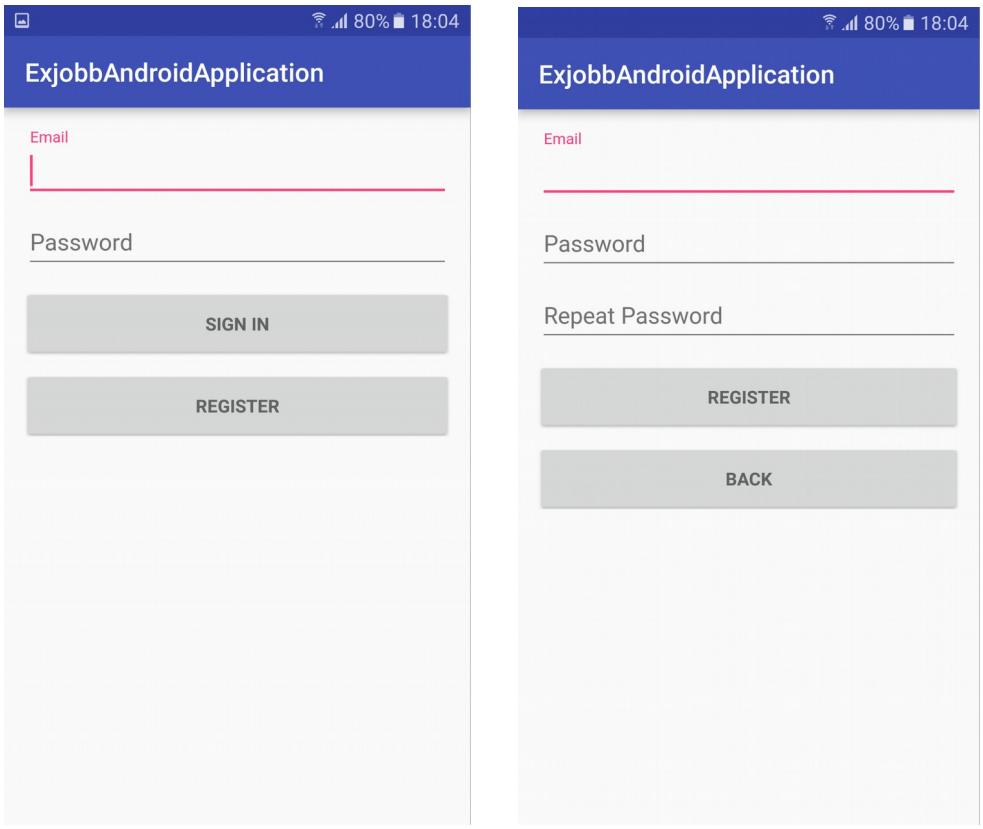
Under projektets gång har vi testat kontinuerligt. När vi implementerat något nytt har detta testats med huvudsakligen tre olika enheter. Mestadels har testningen skett inomhus, mest för att se att meddelanden skickas som de ska mellan server och klient, också för att se att det som ska visas i Androidapplikationen visas på rätt sätt. Men för att se att det GPS-positionen fungerar korrekt har vi också testat utomhus med jämna mellanrum.

Eftersom att vi inte har haft tillgång till sensorerna under projektets gång har vi implementerat en simulator som simulerar ett antal sensorpars beteende. Detta har gjort att vi har kunnat till viss del testa systemet utan sensorerna. Simulatorn är dock relativt simpel eftersom att den inte simulerar en användare som går över campus utan bara slumpar att en sensor gör en avläsning. Sedan görs en slumpgenerering igen som bestämmer om den andra sensorn i sensorparet också ska göra en avläsning, detta gör att en sektion antingen räknar upp eller ner antalet estimerade personer som befinner sig i sektionen.

5 Resultat

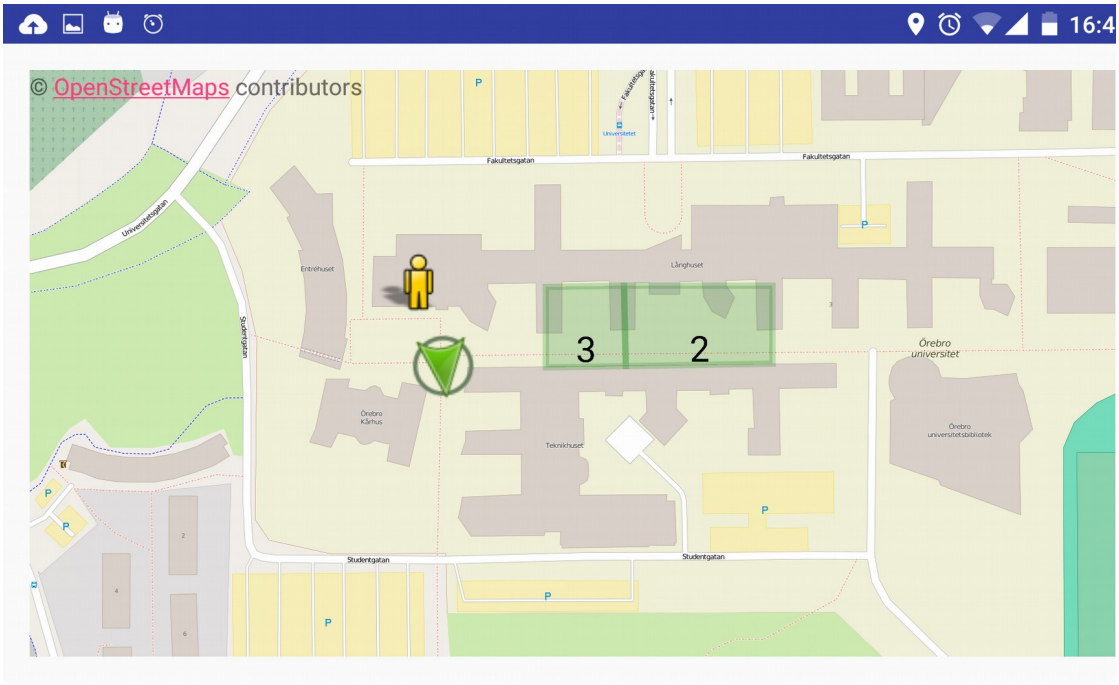
5.1 Hur systemet fungerar från användares perspektiv

För att en användare ska kunna använda applikationen måste ett konto registreras. Användarnamnet för ett konto måste vara en e-mailadress som inte redan finns registrerad i databasen. Man måste också ange ett lösenord som är minst sex tecken långt och bekräfta detta genom att ange samma lösenord två gånger. Figur 11 visar inloggningsskärmen samt registreringsskärmen. Med hjälp av 'Register'-knappen kan användaren växla från inloggning till registrering och med 'Back' knappen gå från registrering tillbaka till inloggning. Vid lyckad registrering så loggas användaren in på servern och kan börja ta del av informationen samt bidra med sin position. Denna informationsdelning pågår så länge som användaren är inloggad och är möjlig endast om platsdata är aktiverat på användarens smartphone. Figur 12 och 13 visar hur applikationen ser ut när användaren är inloggad, på bilderna visas användarens position när denne är i rörelse samt stillastående. På bilderna visas också en annan användares position och de nuvarande sektionerna på Campus samt det estimerade antalet personer som befinner sig i dessa sektioner. När användaren sedan trycker på "bakåtknappen" så blir denne utloggad ifrån servern igen.



Figur 11: Inloggningsskärm och Registreringsskärm

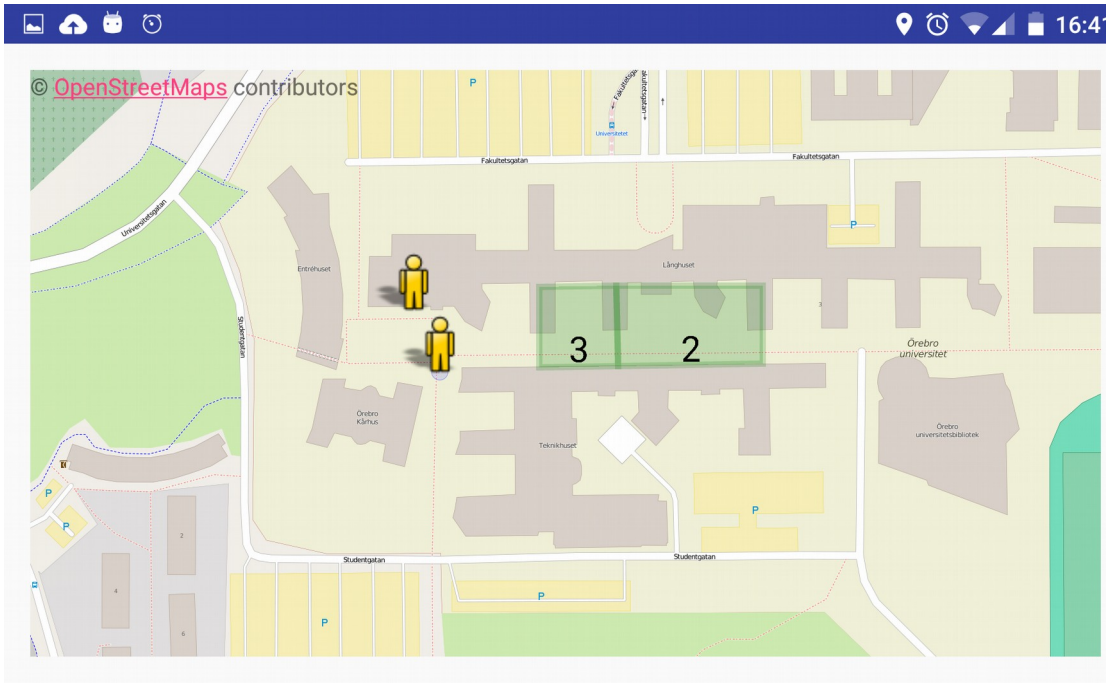
Om applikationen inte lyckas ansluta till servern inom fyra sekunder avbryter den inloggningsförsöket och meddelar användaren att inloggningen inte lyckades och borde försöka igen vid ett senare tillfälle. Detta kan bero på att servern inte är i drift, eller att problemet är på användarens sida till exempel att användaren inte är ansluten till något nätverk.



Figur 12: Huvudskärmen med den inloggade användaren i rörelse, samt en annan inloggad användare. Siffrorna 3 och 2 är estimerat antal personer som befinner sig i sektionerna.

När en användare startar applikationen för första gången så kommer den att fråga efter rättigheter att använda telefonens platsdata och interna lagringsenhet. Rättigheterna till platsdata behövs för att applikationen ska kunna använda GPS-enheten i telefonen för att kunna hitta sin position, medans rättigheterna till den interna lagringsenheten behövs för lagring av kartan innan den kan visas i applikationen. Om dessa rättigheter inte beviljas så kan inte applikationen fungera korrekt och en inloggning kommer inte kunna ske. Efter att användaren har loggat in och GPS-enheten registrerar en ny position skickar den positionen till servern. Applikationen väntar sedan på att servern skickar tillbaka information som innehåller andra inloggade användares positioner, samt data från sensorerna som estimerar antalet personer i sektionerna på Campus.

När en klient ansluter till servern så startar servern upp en egen tråd för kommunikation med klienten. Det första meddelandet som servern tar emot är ett inloggningsmeddelande eller ett registreringsmeddelande. Om det är ett registreringsmeddelande så kollar den först om dem e-mailadress som användaren angett inte redan används på databasen, så krypteras lösenordet och användaren sparas till databasen. Är det ett inloggningsmeddelande som servern tar emot så kollar servern istället upp användaren i databasen, krypterar lösenordet och jämför det med lösenordet som står i databasen.



Figur 13: Huvudskärmen med den inloggade användaren stillastående, samt en annan inloggad användare. Siffrorna 3 och 2 är estimerat antal personer som befinner sig i sektionerna.

När servern startar så försöker servern ansluta sig till sensorerna som finns utplacerade på Campusområdet, när servern har upprättat en uppkoppling till en sensor så börjar den ta emot data ifrån den. Genom informationen som servern läser ifrån konfigurationsfilerna så vet servern exakt vad den ska göra med informationen ifrån en speciell sensor.

Bilderna i kapitlet är skärmdumpar tagna på en Google Nexus 5X och en Samsung Galaxy S6. Ikonerna som kan ses på bilderna är standard i OsmDroid. När positionsberäkningen blir tillräckligt exakt, och användaren är i rörelse så byts “person”-ikonen ut mot en pil som visar i vilken riktning användaren rör sig, se figur 12. Detta sker endast lokalt på användarens smartphone. Andra användares rörelseriktning visas inte.

6 Diskussion

Vi har skapat ett system som ger användaren bättre översikt var på Campus det befinner sig personer. Användaren kan med informationen som visas i applikationen välja den väg över Campus som användaren känner sig tryggast med. Vi hade som mål att skapa en applikation som var simpel och lättanvänd men samtidigt gav användaren så mycket relevant information som möjligt. Vi tycker att vi lyckades med detta. Vi tror att systemet har stor potential om det skulle vidareutvecklas med bland annat en mer precis lösning för att upptäcka personer som inte använder applikationen.

En stor nackdel med systemet är att sensorerna har dålig precision när det kommer till flera människor som passerar samtidigt. Om flera personer går inom en sekunds mellanrum mellan varandra så kommer sensorn bara att upptäcka en av dessa. Samma gäller alltså om de går på rad. För att kunna lösa detta problem så kan en mätning göras och räkna mängden personer som går förbi och hur många de är varje gång. Sedan kan en sannolikhetsberäkning göras på servern eftersom den bara visar estimerade värden.

Sensorerna kan även ge utslag på sådant som vi inte vill upptäcka så som att ett löv blåser förbi eller en fågel flyger förbi och bryter strålen. Det löste vi genom att varje sensor ingår i ett par och båda sensorerna måste utlösas om det ska ge ett resultat på servern. Självklart kan det fortfarande hända att lövet/fågeln flyger förbi båda sensorerna i paret, men det är mindre troligt att det sker. Detta skapar givetvis nya problem då båda sensorerna måste utlösas inom fem sekunder om det ska räknas, så om någon utlöser den första sensorn och sedan väntar mellan sensorerna i fem sekunder så kan personen röra sig in i en sektion utan att bli upptäckt.

Ett annat problem är då det bara fanns tillgång till sex stycken sensorer och dessa skulle placeras i par så betydde det att de endast kunde placeras på tre platser. Det gjorde att dörrar tappade prioritet och gör att personer som går in/ut genom byggnaderna kommer inte ge något utslag på sensorerna. Detta är inte ett jättestort problem då applikationen främst kommer att användas på kvällar och nätter, då dörrarna är låsta för obehöriga. Om det ändå behövs ett sätt att upptäcka de som går in/ut genom dörrarna så går det alltid att installera fler sensorer vid dörrarna då det enkelt går att lägga till nya komponenter till systemet.

6.1 Etiska aspekter angående övervakning

Vi har haft som mål att göra ett system som använder övervakning på ett bra sätt, så att systemet hjälper personer utan att kränka någons integritet. Därför är mycket anonymt i vårat system, till exempel så kan man inte se vilka de andra personerna på kartan är. Även fast vi loggar användardata så som GPS-koordinater så sker detta helt anonymt. Där vi istället för att logga med användarens e-mailadress så får en användare ett anonymt id vid inloggning som används när GPS-data loggas till databasen. Vi ville inte att data skulle vara tillgänglig för alla då inte alla i hela världen behöver ha tillgång till information om hur många som befinner sig på campus. Därför implementerade vi boundingboxes för att bara spara relevant information och för att bara skicka data till relevanta användare. Om användaren inte befinner sig på, eller i närheten av Campus får användaren inte tillgång till informationen. För sensorerna så loggas bara data när servern får in en signal för båda sensorerna i ett par. Då kommer givetvis frågan:

När är det okej att övervaka? I vårat syfte så är det ingen som aktivt övervakar var som händer runt Campus, utan en användare väljer själv när man vill visa vart man befinner sig och stänger av applikationen så visas användarens sista position bara en liten stund innan det slutligen raderas. Övervakning är alltid något som måste hanteras varsamt då övervakning inte är något som ska ske för övervakningens skull, utan det måste ha ett extra syfte. Till exempel som vårat system, för att hjälpa personer som känner sig otrygga när de är ute på natten så de kan välja en väg som de känner sig bekväma med.

Det mest använda sättet att övervaka ett område är att använda övervakningskameror. Undersökningar har gjorts på kamerornas effekt på brottslighet och kommit fram till olika slutsatser. I vissa områden har det inte gjort någon skillnad men i andra områden som till exempel Stockholms tunnelbana har kamerorna gjort att brottsligheten minskat. En jämförelse mellan antalet brott som begicks innan och efter att kamerorna installerades visade att brottsligheten gick ner med 25% och att 15% av de brotten istället begicks i närområdet istället.[14]

Vi har förhoppningar om att detta system kan minska risken för överfall på Campus. Genom att ge användare en möjlighet att se i förväg var det befinner sig personer och kunna välja en väg som personen känner sig bekväm med.

Om vi jämför vårat system med till exempel övervakningskameror så får som sagt tidigare att användaren själv väljer att visa sig för systemet genom att starta applikationen, medan om övervakningskameror finns installerade så har personen inget val att visa sig för kamerorna, om inte personen vet om vart kamerorna sitter och väljer en väg runt dessa då folk ofta känner sig obekväma med videoövervakning. Detta kan alltså ge en direkt motsatt effekt om syftet med kamerorna är att göra så människor kan känna sig säkrare.

6.2 Uppfyllande av projektets krav

Alla kraven som ställdes upp inför projektets uppstart uppfylldes. Där vi har en applikation som användare kan registrera sig i och logga in. När en användare är inloggad kan denne ta del av information om andra inloggade användares positioner, samt sensordata med estimerat antal personer som befinner sig i varje sektion.

Det krav som nämndes tidigare som vi hade för servern var följande:

- En server som tar emot data från klienter och sensorer och behandlar data innan den skickar ut informationen till klienterna.
- En databas för lagring av användarkonton samt loggning av applikationsanvändning.
- Att eventuellt lagra sensor- och GPS-data på databasen för att senare ha möjlighet att analysera rörelsemönster.

Vi uppfyllde alla dessa krav, där vi hade en databas som hanterade användarkonton på ett säkert sätt. Då säkerhetsavdelningen ville ha tillgång till databasen för att analysera data så gjorde vi så databasen hanterar sensor- och GPS-data var för sig. Själva servern kan hantera uppkopplingar ifrån användare och kan koppla upp sig mot sensorerna och kan försöka koppla upp sig igen mot sensorerna om servern tappar uppkopplingen med en av dem. Den kan också rensa bort utdaterad data.

6.3 Projektets utvecklingspotential

Då systemet endast är en prototyp så är hanteringen av användarkonton ett stor brist då det endast krävs en e-mailadress och den inte behövs bekräftas vid registrering så finns ett område att utveckla på att till exempel implementera auto-genererade e-mail som skickas ut vid registrering för att ett konto ska vara giltigt. Studentkonto/ORU-konto är också en möjlighet då det är en lokal applikation och endast ska kunna användas runt Campusområdet så endast studenter eller människor som har någon koppling till universitetet kan använda applikationen. Då det just nu endast är en applikation tillgänglig för Androidtelefoner så skulle användarens Google-konto kunna användas istället, men det get istället en begränsning då ifall en liknande applikation skulle utvecklas till iOS så blir det genast krångligare med denna metod då inget garanterar att användaren då har och använder ett Google-konto.

Just nu kan systemet endast öka trygghetskänslan hos användaren genom att ge användaren en bättre översikt över antalet personer och deras ungefärliga position på campus.

Trygghetskänslan skulle eventuellt kunna ökas ytterligare om det fanns någon typ av larm i applikationen som man kan använda om man blir/tror att man ska bli överfallen på campus. Detta larm skulle kunna kopplas till andra användare som är på campus eller till väktare.

För att utveckla och förbättra användarupplevelsen för administratören för servern så kan en extern applikation utvecklas för att göra utökningen av systemet simplare. Vid avslutandet av projektet så gjordes detta genom att ändra i tre olika XML-filer för att lägga till nya sektioner och sensorer. Detta skulle kunna göras genom att den externa applikationen läser in filerna och genom att till exempel klicka på en karta (som motsvarar den i applikationen) kunna lägga till nya sektioner, istället för att leta upp GPS-koordinater och lägga till dessa manuellt i filerna. Den befintliga metoden tar lång tid att göra, det är enkelt att göra fel, och man behöver tillgång till applikationen för att se att det blev rätt då det inte finns något sätt att visualisera ändringarna som görs.

6.5 Reflektion kring eget lärande

6.5.1 Kunskap och förståelse

Vid projektets uppstart så hade ingen av oss tidigare arbetat med Androidutveckling eller lokalisering av mobiltelefoner. Under projektets gång så har vi fått en bättre insikt hur Androidapplikationer utvecklas, samt fått en bättre insikt i hur operativsystemet fungerar och hur det hanterar applikationer. Vi har också lärt oss hur GPS fungerar både teoretiskt inför projektet och praktiskt under utvecklingen av applikationen.

För att förstå oss bättre på hur lokalisering av mobiltelefoner används och fungerar så var vi tvungna att undersöka hur andra metoder fungerade, inte bara GPS. Vi undersökte mycket om hur Wifi-lokalisering fungerade, men kom fram till att GPS ändå var bättre lämpat för vårat system då Campusområdet är ett relativt litet område och Wifi-lokalisering har sämre precision än GPS.

För att kunna bygga ett system som både skulle fungera och som vi var nöjda med så behövde vi också undersöka och lära oss hur en bra databas byggs upp och hur man skyddar den ifrån attacker, samt hur användarkonton lagras säkert på databasen, där då kryptering av lösenord ingick.

6.5.2 Färdighet och förmåga

6.5.2.1 Arbetsgång

Då projektet fokuserade på övervakning och vi båda ansåg att det var viktigt att kunna skydda användare så lärde vi oss att utveckla en prototyp-produkt med stor fokus på anonymitet. Där varje steg i projektet utgick ifrån punkten att kunna hålla en användare anonym och hur det inte skulle vara kränkande genom att övervaka. Därför så fokuserade specifikationen och utvecklingsprocessen mycket på just detta.

6.5.2.2 Samarbete

Eftersom utvecklingen av server och applikation skedde parallellt så krävdes ett gott samarbete mellan båda parter där vi frågade varandra om t.ex. hur något fungerade på den andra delen av systemet. Så vi lärde oss att bli bättre på att samarbeta för att kunna uppnå ett gemensamt mål med projektet. Där vi alltid satt på samma ställe och utvecklade ifall frågor skulle uppstå, samt att kunna diskutera lösningar för att hitta en som båda var nöjda med. Detta gjorde vi genom att en ritade på whiteboard och förklarade sin tankegång, medan den andra kunde kommentera vad som kunde vara bra eller dåligt med den lösningen. Om vi snabbt kom fram att lösningen var dålig så bytte vi ofta plats och började om.

6.5.3 Värderingsförmåga och förhållningssätt

Vi planerade vår utvecklingsprocess bra vid projektets uppstart vilket ledde till att vi kunde följa den tidsplan som vi gjort. När problem uppstod eller en svårare del skulle implementeras så diskuterade vi, ritade, beräknade och planerade på whiteboardtavla för att hitta den bästa lösningen till problemet. Det var ett viktigt arbetssätt för oss då vi ofta kom fram till att en lösning inte var bra, eller inte skulle fungera i längden. Det gav oss smarta och tydliga lösningar på problem och det ledde också till en snabbare implementation av problemet.

För att hitta den bästa lösningen till problemen som skulle lösas under projektets gång har vi behövt utvärdera ett flertal olika lösningar till varje problem. Detta har gjort att vi blivit bättre på att kritiskt värdera olika lösningar.

Vi hade också handledarmöte varje vecka där vi diskuterade hur projektet låg till i jämförelse med tidsplanen. Vi visade också upp demoversioner av systemet för att kunna få kritik på något som behövde förändras/förbättras.

Eftersom att systemet till stor del handlar om övervakning har vi tänkt mycket på hur vårat system påverkar samhället. Vi har strävat efter att göra systemet så anynomt som möjligt för att inte kränka någons integritet. Men inte på ett sätt som gör att systemet inte kan ge någon hjälpsam information. Vi har också lagt fokus på att minska risken för att systemet används på ett felaktigt sätt, till exempel att leta på personer att överfalla. Detta eftersom att meningen med systemet är att hjälpa personer som inte känner sig trygga.

7 Referenser

[1] Cornelia Thalin & Lina Vidarsson Hane, *Kan en smartphone-applikation bidra till ökad bedömning av trygghet och minskad bedömning av rädsla att utsättas för brott?*. Örebro universitet. 2015

[2] U.S Government. *Official U.S. Government information about the Global Positioning System (GPS) and related topics* [Internet]. Washington D.C: U.S Government; 2016 [Uppdaterad 2016-01-14; citerad 2016-04-27]. Hämtad från: http://www.gps.gov/

[3] European Space Agency. *What is GALILEO?* [Internet]. Frascati: ESA Communication Department; 2015 [uppdaterad 2015-12-18; citerad 2016-04-27].
Hämtad från: http://www.esa.int/Our_Activities/Navigation/Galileo/What_is_Galileo

[4] Chi-Hua Chen; Chi-Ao Lee; Chi-Chun Lo. *Vehicle Localization and Velocity Estimation Based on Mobile Phone Sensing.* IEEE Access. 2016;4:803-17.

[5] Musa A.B.M & Eriksson J. *Tracking Unmodified Smartphones Using Wi-Fi Monitors.* SenSys '12 Proceedings of the 10th ACM Conference on Embedded Network Sensor Systems. SenSys '12. November 6-9, 2012; Toronto, ON, Canada. New York, NY, USA: ACM; 2012. s. 281-94.
Hämtad från: https://www.cs.uic.edu/~amusa/resources/musa-eriksson-sensys12.pdf

[6] Yohan Chon; Suyeon Kim; Seungwoo Lee; Dongwon Kim; Yungeun Kim; Hojung Cha, *Sensing WiFi Packets in the Air: Practicality and Implications in Urban Mobility Monitoring.* UbiComp '14 Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing. UbiComp '14. September 13- 17, 2014; Seattle, WA, USA. New York, NY, USA: ACM; 2014. s. 189-200.
Hämtad från: http://dl.acm.org.db.ub.oru.se/citation.cfm?id=2636066

[7] Premysl Hudec; Milan Polivka; Pavel Pechac. *Microwave System for the Detection and Localization of Mobile Phones in Large Buildings.* IEEE Trans Microw Theory Tech. 2005;53(6): 2235-9.

[8] OpenStreetMap Foundation. OpenStreetMap[Internet]. [citerad 2016-04-11]
Hämtad från: http://www.openstreetmap.org

[9] V. Scott Gordon & James M. Bieman, *Rapid Prototyping: Lessons Learned.* IEEE Software, 1995;12(1):85-95.

[10] Jayesh Patel. *How PreparedStatement in Java prevents SQL Injection?*. 2015, September [citerad 2016-06-16]. javabypatel [Internetblogg].
Hämtad från: http://javabypatel.blogspot.se/2015/09/how-prepared-statement-in-java-prevents-sql-injection.html

[11] Andrew J. Elliot and Markus A. Maier. *Color Psychology: Effects of Perceiving Color on Psychological Functioning in Humans.* Annual Review of Psychology. 2014; 65: 95-120.

[12] Defuse Security. *Salted Password Hashing - Doing it Right* [Internet]. CrackStation: [uppdaterad 2016-04-13; citerad 2016-05-17].
Hämtad från: https://crackstation.net/hashing-security.htm

[13] Satish Chandraa, Anish Kumar Bharti. *Speed Distribution Curves for Pedestrians during Walking and Crossing.* Procedia - Social and Behavioral Sciences. 2013;104: 660 – 7.

[14] Mikael Priks, *The Effects of Surveillance Cameras on Crime: Evidence from the Stockholm Subway.* The Economic Journal. 2015; 125 (588): *F*289–F305.

(Bilaga A)

Documentation for creating and managing sensors and sections

- SensorIps.xml -
Has the base tag <IpContainer>
uses two tags <ID> and <IP_ADDRESS>

<ID> is the number that the sensor is sending over the network
so if it sends 5, the tag should be <ID>5</ID>

<IP_ADDRESS> is just the ip-address to the sensor
<IP_ADDRESS>127.0.0.1</IP_ADDRESS>

```
<IpContainers>
   <IpContainer>
      <ID></ID>
      <IP_ADDRESS></IP_ADDRESS>
   </IpContainer>
</IpContainers>
```

- SectionGpsCoordinates.xml -
The main file for defining a new section on the Campus

The Tags that is used are
<ID>
<Coordinates>
<Coordinate>
<Latitude>
<Longitude>

<ID> is the ID number of the section, used in sensorPairs
for defining what section a sensor belongs to.

<Coordinates> is defining that a list of Coordinate is defined

<Coordinate> is what is used for defining a coordinate
should always be inside a <Coordinates> tag

<Latitude> Should be inside a <Coordinate> with a number defining the latitude coordinate
for example <Latitude>59.254562</Latitude>

<Longitude> Should be inside a <Coordinate> with a number defining the longitude coordinate
for example <Longitude>15.248144</Longitude>

NOTE!
To add new sections, just define the corners of the section in the order
there can be more than 4 corners in a section. The first row of the section
MUST be the North-East corner, and then defined in a counter-clockwise direction.

```
<Sections>
   <Section>
      <ID></ID>
      <Coordinates>
         <Coordinate>
            <Latitude></Latitude>
            <Longitude></Longitude>
```

```
        </Coordinate>
        <Coordinate>
          <Latitude></Latitude>
          <Longitude></Longitude>
        </Coordinate>
        <Coordinate>
          <Latitude></Latitude>
          <Longitude></Longitude>
        </Coordinate>
        <Coordinate>
          <Latitude></Latitude>
          <Longitude></Longitude>
        </Coordinate>
      </Coordinates>
    </Section>
</Sections>
```

- SensorPairs.xml -
The file defines the pairs of sensors placed on Campus,

<innerSensor> - The ID of the sensor that is the closest one to the center of the section
<outerSensorID> - The ID of the sensor that is the furthest from the center of the section
<innerSectionID> - The ID of the section that is closest to the InnerSensorID
<outerSectionID> - The ID of the section that is closest to the OuterSensorID

NOTE!
If there is no NeighboringSection set NeighboringSectionID to -1.

```
<SensorPairs>
   <SensorPair>
      <innerSensor></innerSensor>
      <outerSensor></outerSensor>
      <innerSectionID></innerSectionID>
      <outerSectionID></outerSectionID>
   </SensorPair>
</SensorPairs>
```