Funktionalitetsdelning

- Med fokus på användbarhet

LUNDS UNIVERSITET
Campus Helsingborg

LTH Ingenjörshögskolan vid Campus Helsingborg
Institutionen för datavetenskap

Examensarbete:
William Phan
Simon Wessel

LTH Ingenjörshögskolan vid Campus Helsingborg
Lunds universitet
Box 882
251 08 Helsingborg

LTH School of Engineering
Lund University
Box 882
SE-251 08 Helsingborg
Sweden

Tryckt i Sverige
Lunds universitet
Lund 2015

Sammanfattning

I en värld med många typer av enheter, alltifrån persondatorer till mobiltelefoner, från nätverksansluten belysning till smarta kylskåp, har varje kategori av enheter sin typiska hårdvara och funktionalitet. Dessa funktionaliteter är vad som kännetecknar enheten, men sätter också begränsningar för vad den kan göra.

Med utgångspunkten att majoriteten av alla enheter har någon form av nätverksåtkomst, föreslår detta examensarbete att man bör överge synen på enheter som diskreta "saker". Istället bör man se dem som att de bidrar med funktionalitet som kan lånas ut till andra enheter i det omgivande ekosystemet. Denna företeelse benämns i rapporten som "funktionalitetsdelning".

Examensarbetet kartlägger typiska funktionaliteter hos vanligt förekommande konsumentelektronik samt några enhetskategorier som är på framgång. Därefter identifieras ett antal egenskaper som ett system för funktionalitetsdelning behöver.

Resultatet visar att en viktig egenskap är ett gemensamt språk. För att tekniken verkligen ska underlätta människors vardag är intelligenta förslag, enkel konfiguration och automatiserad aktivering viktiga tillskott till funktionalitetsdelningssystemet. Automatiserad aktivering kräver framförallt framsteg inom positionsbestämning.

En prototyp utvecklas enligt en server-klient-modell. Prototypen visar ett exempel på funktionalitetsdelning genom att låta användaren kontrollera en Google Street View på en dator med hjälp av gyroskopet i en mobiltelefon, för att på så sätt erhålla en annorlunda upplevelse. Prototypens plattformar utgörs av Android, Google Chrome och Node.js. För att sammanlänka dessa används JavaScript-biblioteket Socket.IO.

Användartester visar att ett mycket tydligt gränssnittsspråk är viktigt, eftersom funktionalitetsdelning är ett nytt koncept för många användare. Testerna visar också att notifikationer, som är prototypens huvudsakliga metod för att hjälpa användaren, även de måste vara tydliga. Deras placering på skärmen är en viktig aspekt för hur upptäckbara de är.

Funktionalitetsdelning enligt den föreslagna modellen är möjlig, men många pusselbitar behöver falla på plats. Innan ett praktiskt användbart system kan tas i bruk måste säkerhetsaspekten noga utredas.

Efter examensarbetets genomförande presenterade Google mjukvaruplattformen ”Brillo”, och bekräftade därmed att det finns ett fortsatt starkt intresse för IoT. Plattformen tillhandahåller bl a det gemensamma språk som examensarbetet identifierar som en nödvändighet.

Nyckelord: funktionalitetsdelning, Internet of Things, gyroskop, Street View, Socket.IO

Abstract

In a world of multiple devices, ranging from personal computers to mobile phones, from network attached lighting to smart refrigerators, every category of devices has its typical hardware and capabilities. These capabilities are what characterize the device, but it also sets limitations on what it can do.

With the assumption that the majority of all devices have some form of network connectivity, this report suggests that one should abandon the view of devices as discrete "things". Instead one should see them as contributing capabilities that can be shared with other devices in the surrounding eco-system. This phenomenon is termed "capability sharing" in this report.

The thesis maps typical capabilities in commonly available consumer electronics, as well as some emerging device categories. Thereafter, characteristics of a capability sharing system are identified.

The results show that an important characteristic is a common language. To ensure that the technology really improves every-day life, other important additions are intelligent suggestions, ease of configuration and automated activation. Automated activation requires further progress in the field of positioning.

A prototype is developed using a sever-client model. The prototype demonstrates capability sharing by letting the user control a Google Street View on a computer using the gyroscope of a mobile phone, thus providing a different experience. The target platforms of the prototype consist of Android, Google Chrome, and Node.js. To link these together, the JavaScript library Socket.IO is used.

User tests show that a very clear interface language is important, since capability sharing is a new concept for many users. The tests also show that notifications, which are the prototype's primary means of helping the user, must be clear. Their placement on the screen is an important factor of their discoverability.

Capability sharing according to the proposed model is possible, but many pieces of the puzzle are missing. Before a usable system is put into production, security aspects will have to be thoroughly investigated.

After the thesis had been completed, Google announced the software platform "Brillo", thereby confirming a continued strong interest in IoT. The platform provides the common language that this thesis identifies as a necessity.

Keywords: capability sharing, Internet of Things, gyroscope, Street View, Socket.IO

Förord

Detta examensarbete har genomförts i samarbete med Tactel AB i Malmö. Det har gett oss möjligheten att utforska morgondagens användning av konsumentelektronik, och har varit en utmanande avrundning på våra tre år på högskoleingenjörsprogrammet i datateknik vid LTH, Campus Helsingborg.

Tack till vår handledare på Tactel, Magnus Sjungare, som kommit med idéer vid de tillfällen vi kört fast. Tack till Olof Råborg och Andreas Markewärn, som bidragit med värdefulla synpunkter. Tack även till vår handledare Christian Nyberg och vår examinator Christin Lindholm.

Malmö, 2015-05-26

William Phan
Simon Wessel

Innehållsförteckning

1 Inledning

1.1 Bakgrund

I en värld med många typer av enheter, alltifrån persondatorer till mobiltelefoner, från nätverksansluten belysning till smarta kylskåp, har varje kategori av enheter sin typiska hårdvara och funktionalitet. För tjugo år sedan var det inte självklart att man hade en dator hemma, medan dagens hem ofta huserar flertalet datorer, surfplattor och mobiltelefoner. Hushållsapparater och bilar börjar även de få betydande beräkningskapacitet, och kan ofta kommunicera med omvärlden på ett eller annat sätt.
Ordet "mobiltelefon" har idag en annan betydelse än den ursprungliga - den är inte bara telefon, utan många människors huvudsakliga dator. Marknaden för mobilappar var med dagens mått obefintlig före år 2000, men har idag växt till att bli många utvecklares huvudsakliga inkomstkälla.[1] Surfplattor används i skolundervisning och TV-apparater kan göra mycket mer än att ta emot de sändningar TV-kanalerna tillhandahåller.

Det finns dock inneboende begränsningar i respektive enhetskategori. En mobiltelefon kan inte bli mycket större utan att mobiliteten försämras. Detta sätter en begränsning på skärmens fysiska dimensioner. Ett sätt att kompensera för denna begränsning skulle kunna vara att mobiltelefonen lånar skärmen från en större enhet, exempelvis en TV. Tar man denna tanke vidare leder den till en mängd möjligheter där konsumentelektronik kan kringgå sina begräsningar genom att de får möjligheten att dra nytta av varandras funktionaliteter.

Om man gör antagandet att fler människor använder ny teknik om den är lätt att konfigurera inser man att det finns en stor vinning i att göra installation och konfiguration så enkel som möjligt för en ny användare. Framtidsvisionen är ett system som sköter stora delar av konfigurationen själv utan att användaren behöver övervaka varje steg.

Denna rapport undersöker hur ett system för funktionalitetsdelning skulle kunna byggas. Det föreslagna systemet har utvecklats med enkel konfiguration i åtanke för att underlätta för nya användare att anamma tekniken.

1.1.1 Tidigare arbete

Ett område som liknar funktionalitetsdelning är sessionsöverföringar, då båda handlar om att överföra ett tillstånd från ett ställe till ett annat. Olika system har föreslagits, dels de där hela sessionen överförs (Phan, Guy, Gu & Bagrodia 2001), och dels de där varje enhet får en egen kopia av sessionen (Alapetite 2009). Sessionsöverföringar lägger grunden för funktionalitetsdelningar genom att tillhandahålla principer för hur data kan överföras från en enhet till en annan.

[1] Download Catalog, för Danger OS lanserades 2002, och är numera nedlagd. Apples App Store och Googles motsvarighet Play kom under år 2008.

Det finns flera mobilapplikationer som gör en mobiltelefons kamera tillgänglig som en IP-kamera via Wifi (IP Webcam, Wifi Camera samt Web of Cam är några exempel[2]). Detta är en intressant kategori av applikationer, men ur ett funktionalitetsdelningsperspektiv inte tillräckligt flexibel. För det första delar de endast en funktionalitet. För det andra så vill man i ett system för funktionalitetsdelning kunna fråga efter en funktionalitet från den enhet som saknar den. Detta är inget ovan nämnda appar tillgodoser.

1.1.2 Tactel

Detta examensarbete har genomförts i samarbete med Tactel AB i Malmö. Tactel är ett IT-bolag med cirka 80 anställda som under 2015 blivit en del av det multinationella elektronikbolaget Panasonic. De utvecklar programvara med fokus på människan under devisen "Humanizing Technology", med tanken att tekniken ska anpassa sig efter användaren och inte tvärtom. Ett av företagets huvudfokus är utveckling av mobilapplikationer och man har kunder som exempelvis Swedbank, Boxer och Telia.

1.2 Syfte och mål

Denna rapport undersöker hur funktionalitet kan delas mellan olika typer av enheter för att användaren ska ha tillgång till en större spektrum av funktionaliteter. Delningen och konfigurationen av densamma bör ske på ett för användaren smidigt sätt.

Målet för examensarbetet är att utveckla en prototyp för delning av en funktionalitet från en mobil enhet till en dator. Funktionaliteten som delas från mobiltelefonen är dess gyroskop. Denna funktionalitet används i en webbläsare för att navigera en gatuvy i karttjänsten Google Maps. Det övergripande målet innefattar i huvudsak två delmål. Dels en litteraturstudie där existerande arbeten och modeller granskas men även framtida användningsområden. Efter litteraturstudien implementeras en prototyp. Om prototypen visar potential ska denna rapport och den framtagna prototypen kunna användas som utgångspunkt för vidareutveckling.

1.3 Problemformuleringar

Här listas de problemformuleringar som besvaras av examensarbetet.

- Vilka egenskaper behöver ett system för funktionalitetsdelning?
- Hur kan funktionalitet delas från en enhet till en annan?
- Hur kan en prototyp som visar på funktionalitetsdelningens användbarhet utvecklas med hjälp av dagens teknik?
- Hur kan funktionalitetsdelningen initieras av en ovan användare?

[2] Apparna hittas i Google Play via följande länkar:
https://play.google.com/store/apps/details?id=com.pas.webcam
https://play.google.com/store/apps/details?id=teaonly.droideye
https://play.google.com/store/apps/details?id=com.webofcam [2015-04-13]

1.4 Avgränsningar

Systemet kommer inte fokusera på säkerhet, då det finns lösningar för att kryptera trafik över de protokoll prototypen använder. Förslag på framtida arbete med avseende på säkerhet återfinns i avsnitt 6.3.

Prototypen kommer att begränsa sig till tre plattformar. En enhet som lånar ut sin funktionalitet (Android), en enhet som lånar in funktionaliteten (Chrome) och en server (Node.js) som styr trafiken mellan de båda enheterna. Dessa val baseras dels på att utvecklingsverktygen är fritt tillgängliga, och dels på att de delvis byggs i samma programmeringsspråk.

1.5 Byte av inriktning

Examensarbetets ursprungliga riktning var en undersökning inom sessionsöverföringar. Detta förändrades under arbetets gång till funktionalitetsdelning, eftersom sessionsöverföringar redan hade en hel del forskning och funktionalitetsdelning var ett nytt begrepp. På grund av inriktningsbytet i kombination med tidsbegränsningen lades mycket lite arbete på prototypens säkerhet. Detta är dock ett område som har flera möjliga lösningar, och prototypen har utvecklats med tanken att man lätt ska kunna byta säkerhetsmodell.

Den första prototypen som utvecklades med inriktningen funktionalitetsdelning baserades på user story A.1 (appendix A). Under arbetets gång påträffades en begränsning i webbläsarens JavaScript-API, som gjorde överskuggning av en specifik funktion omöjlig p g a det säkerhetshål detta skulle medföra. En vinst med överskuggningen hade varit att en Chrome-extension hade fungerat på många redan existerande hemsidor.

1.6 Definitioner

Här presenteras de ord och begrepp som denna rapport använder ofta eller som har en betydelse som avviker från vanligt språkbruk.

Användbarhet I detta examensarbete utgår begreppet användbarhet från Lauesens definition (2002). Användbarhetsfaktorerna presenteras här på engelska, eftersom de är svåra att översätta på ett bra sätt:

1. Ease of learning. Hur lätt ett system är att lära sig.
2. Task efficiency. Hur effektivt systemet är för en van användare.
3. Ease of remembering. Hur lätt systemet är att komma ihåg för den som bara använder det ibland.
4. Subjective satisfaction. Hur nöjd användaren är med systemet.
5. Understandability. Hur lätt det är att förstå vad systemet gör.

Användartesterna har främst undersökt tre av dessa faktorer, nämligen 1, 4 och 5. 1 och 5 för att det är frågan om en ovanlig tillämpning där användaren dels måste lära sig ett nytt flöde, men även förstå syftet med systemet. 4 för att få värdefull feedback utöver de problem som upptäcktes under pågående test.

Enhet (device) Datorer, mobiltelefoner, surfplattor etc benämns som “enheter” i denna rapport.

Funktionalitet Ett karaktärsdrag eller hårdvara hos en enhet som man vill kunna låna ut eller in. Exempelvis fingeravtrycksläsare, kamera eller stor skärm.

Kapabilitet Se funktionalitet

Projektgrupp Examensarbetarna

Session En applikations tillstånd och variabler.

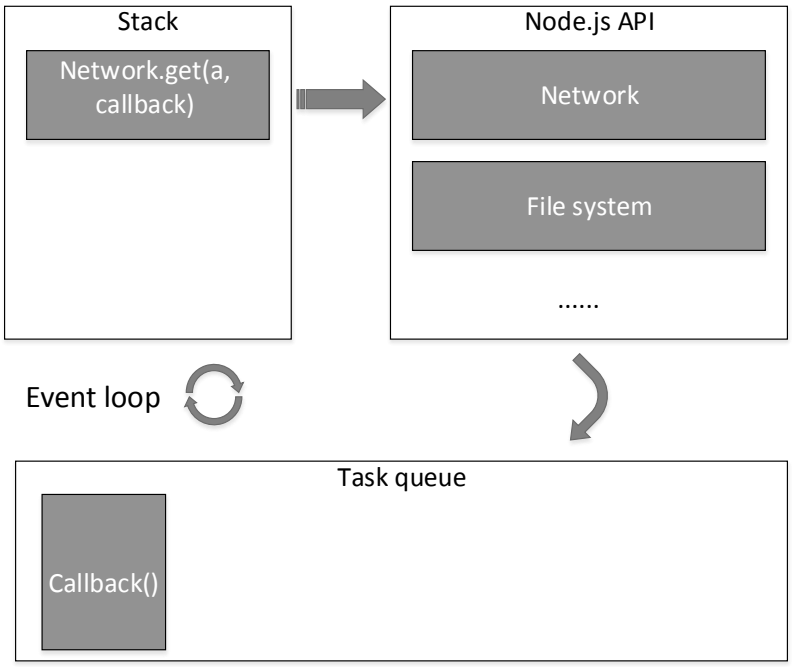
2 Teknisk bakgrund

I detta kapitel presenteras de tekniker som använts för att implementera prototypen som var ett av examensarbetets delmål.

2.1 Node.js

En server används för att hantera kommunikationen mellan två enheter.
Servern fungerar som "spindeln i nätet" och är utvecklad för plattformen Node.js. Node.js är baserat på Googles JavaScript-motor V8 (Google Inc. 2014). JavaScript är således det språk som används för att programmera Node.js. Node.js är från början tänkt att användas för utveckling av asynkrona, skalbara nätverksapplikationer (Joyent Inc. 2015), men har sedan sin tillblivelse använts även för andra ändamål.[3]

Något som är typiskt för JavaScript och därmed även Node.js är dess sätt att hantera asynkrona anrop (figur 2.1). I korta drag är JavaScript enkeltrådigt till sin natur, och överlåter parallellismen åt andra, mer maskinnära och minneseffektiva bibliotek.



Figur 2.1: En visualisering av Nodes parallellism-modell. En funktion anropas, och när den är klar anropas dess callback-funktion.

Nackdelen med mer traditionella modeller, där nya trådar eller processer skapas av den aktuella runtimen, är att varje tråd eller process medför en viss overhead vad gäller arbetsminne och kontextbyten. Joyent, som underhåller Node.js, marknadsför plattformen som "non-blocking", och menar att denna modell för realtidsprogrammering är lättare för oerfarna programmerare att förstå och använda. Som utvecklare behöver man inte bekymra sig om dödlägen och andra typiska problem inom realtidsprogrammering. För detta examensarbete lämpar sig Node.js bra, och har

[3] Texteredigeringsverktyget Atom är delvis skrivet i Node.js.

den extra fördelen att servern då skrivs i samma språk (JavaScript) som insticksprogrammet till Chrome.

För att förstärka Node.js används ett antal ramverk. De fyller olika syften, som exempelvis tydligare konfiguration, enklare databashantering och säkrare och mer flexibla inloggningsprocesser. Dessa ramverk beskrivs i avsnitt 2.1.1 till 2.1.3.

2.1.1 Express

Express är ett JavaScript-ramverk för Node.js som gör det enklare att utveckla web-appar och API:er (Express u.å.) genom att tillhandahålla funktioner, exempelvis för att snabbt skapa en server eller skicka svar på HTTP-anrop. Serverkomponenten utgörs av ett REST-API. Med Express.js blir det mindre kod att skriva, och den kod som skrivs blir mer koncis. Det finns andra liknande ramverk, exempelvis Koa och Hapi, men Express valdes dels för att det fungerar bra ihop med Passport, men också på grund av att tidigare erfarenhet fanns inom projektgruppen.

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

app.get('/', function (req, res) {
  res.send('Hello World!');
});

var server = app.listen(3000);
```

Figur 2.2: Ett exempel som visar hur lite kod som krävs för att skapa en serverapplikation i Node.js och Express. Detta exempel lyssnar efter GET-requests på http://localhost:3000/ och svarar med "Hello World".

2.1.2 Sequelize

Sequelize är en så kallad Object Relational Mapper (ORM), som sköter kontakten med serverns underliggande databas (Sequelize, u.å.). Med hjälp av Sequelize kan all interaktion med databasen ske programmatiskt i JavaScript, vilket minskar antalet språk som måste användas. Utvecklaren behöver inte definiera tabeller och relationer manuellt i databasen. Det görs istället av Sequelize, baserat på de modeller som definierats i programmet. Sequelize tillhandahåller sedan JavaScript-funktioner för vanligt förekommande SQL-frågor. Om behovet uppstår går det att använda ren SQL. En annan fördel med Sequelize är att den stöder flera databashanterare. För utveckling används SQLite3, en serverlös databashanterare som inte kräver någon konfigurering och som arbetar direkt mot en fil i utvecklingsmiljön. Om systemet någon gång behöver skalas upp är det enkelt att byta ut databashanterare till en mer omfattande lösning, som exempelvis MySQL eller PostgreSQL.

Inför detta val gjordes en jämförelse av Sequelize med ett liknande alternativ - node-orm2. Båda uppfyller önskemålet om utbytbar databashanterare, men valet föll på Sequelize, då det är ett äldre projekt och därmed förhoppningsvis mer moget och stabilt.

```
User.find(123).then(function(user) {
  // Gör något med objektet om det hittades
});

SELECT * FROM USERS WHERE ID=`123`;
```

Figur 2.3: De tre första raderna visar en inbyggd Sequelize-funktion för att hitta ett objekt med avseende på ID. Den sista raden visar motsvarande SQL-fråga.

2.1.3 Passport

Passport är ett ramverk som förenklar autentisering av olika slag (Passport u.å.). Det finns stöd för olika autentiseringslösningar, och för prototypen är främst användarnamn/lösenord samt OAuth2 intressanta som inloggningsalternativ. Användarman/lösenord används under utvecklingen, då det går snabbt att konfigurera. OAuth2 är intressant ur ett användbarhetsperspektiv då användaren kan använda ett befintligt konto hos exempelvis en epost-leverantör för att autentisera sig. Denna egenskap är viktig vid en eventuell vidareutveckling av prototypen. Passport integrerar dessutom väl med Express, vilket är en annan anledning att välja Passport framför andra lösningar.

2.2 Android-applikation

En mobilapplikation utvecklas för en mobil plattform. Android är ett open source-operativsystem som främst används på mobila enheter. Android har ett mycket stort antal användare - över en miljard aktiva (Google I/O 2014). Operativsystemet utvecklas primärt av Google, men det förekommer skräddarsydda versioner från flera mobiltelefonutvecklare. De huvudsakliga målplattformarna är mobiltelefoner och surfplattor men systemet finns även i andra former som till exempel TV och underhållningsmaskiner.

Android-applikationen som utvecklas är tänkt att sköta länken mellan mobiltelefonen och servern. Huvudfunktionen är således att dela funktionalitet och överföra nödvändiga data till servern.

Android-utveckling sker i normalfallet i språket Java. Förutom Javas standardbibliotek tillhandahåller Google Android-specifika API:er. I Android finner man olika API-nivåer som kort beskriver vilken version av Android som mobilapplikationen är tänkt för. En högre API-nivå innehåller fler och nya funktioner men har sämre stöd för äldre versioner och vice versa för äldre versioner. Prototypen utvecklas mot API-nivå 21 som infördes vid övergången till Android version 5.0, som släpptes hösten 2014 (Google Inc u.å.a).

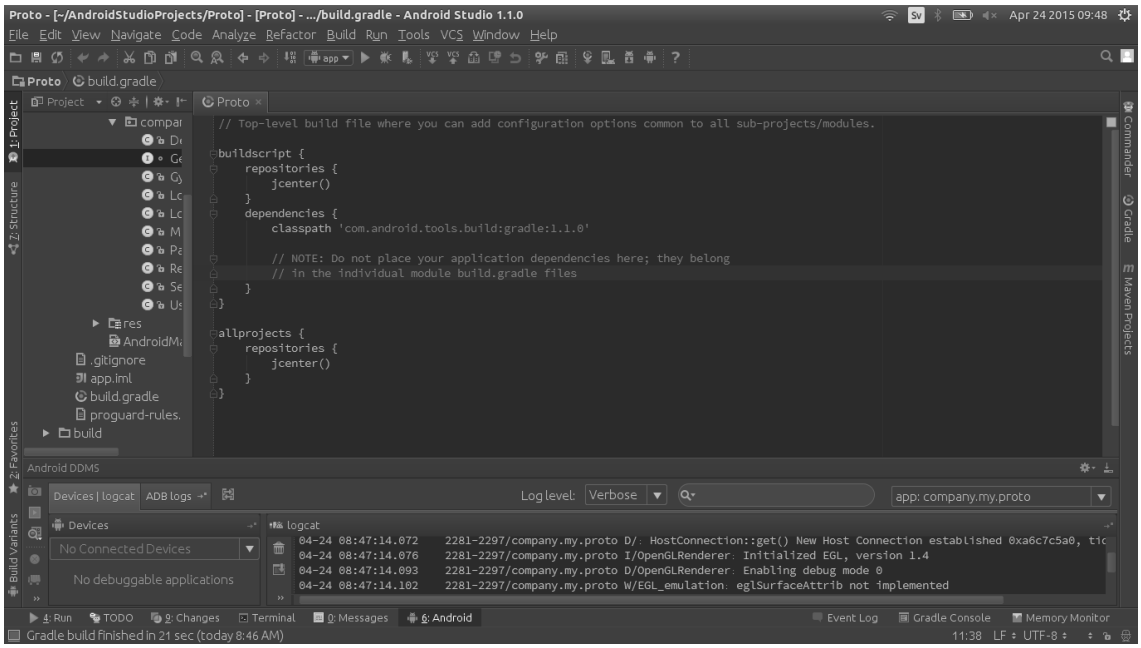
Beslutet för en mobil plattform grundas på att mobiltelefoner och datorer ser väldigt olika ut ur ett funktionalitetsperspektiv. En prototyp som använder två datorer hade istället kunnat utvecklas men skulle inte ge ett lika trovärdigt scenario, då

funktionaliteten som erbjuds är väldigt lika. Med en mobiltelefon å andra sidan, som har andra egenskaper jämfört med en dator, blir funktionalitetsdelning mer motiverat.

Angående valet av mobil plattform, så finns det andra alternativ som till exempel iOS, men valet blev Android på grund av dess stöd och öppenhet i form av publika API:er. I Android är Java det programmeringsspråk som används, vilket är ett språk projektgruppen känner sig bekväma med. Vid val av en annan plattform hade ett nytt programmeringsspråk behövt läras in. Eftersom det är frågan om prototyputveckling med agila metoder finns en vinst i att använda ett redan känt programmeringsspråk, och Android har därmed en fördel.

2.2.1 Android Studio

Mobilapplikationen utvecklas med hjälp av verktyget Android Studio som är nästa generations utvecklingsmiljö och utvecklas av Google. Den är baserad på IDEA IntelliJ som utvecklas av JetBrains och ersätter det tidigare utvecklingsverktyget ADT Bundle (Android Developer Tools Bundle) som använder Eclipse IDE. Android Studio anses nu vara så pass stabil att den rekommenderas framför ADT Bundle som så småningom lär fasas ut (Google Inc. u.å.b). Huvudegenskaperna för Android Studio är realtidsrendering, vilket låter utvecklaren se layouten direkt i IDE:n, och stöd för Gradle, som är ett verktyg för att bygga projekt automatiskt.



Figur 2.4: Utvecklingsmiljön Android Studio.

2.3 Chrome-extension

Ett insticksprogram - en extension - till webbläsaren Chrome utvecklas med syfte att vara länken mot servern från en dator. Insticksprogram kan ges åtkomst till stora delar av webbläsaren genom att utvecklaren specificerar vilka rättigheter insticksprogrammet använder. Användaren måste sedan godkänna dessa vid installationen. Rättigheterna

kan exempelvis ge instickspprogrammet tillgång till innehållet i webbläsarens öppna flikar, eller ge möjligheten att anropa externa tjänster via HTTP-anrop.

Prototypens extension kontrollerar ifall webbinnehållet har stöd för någon extra funktionalitet, och lyssnar efter tillgängliga funktionaliteter via serverkomponenten.

Instickspprogram utvecklas med standardiserade webbteknologier som HTML, CSS och JavaScript. JavaScript används för logiken och beteendet, medan HTML och CSS används för att bygga gränssnittet. Gränssnittet kan bestå av en egen flik eller nya knappar bredvid webbläsarens adressfält. Instickspprogram måste inte ha något gränssnitt - ett JavaScript som körs i bakgrunden räcker.

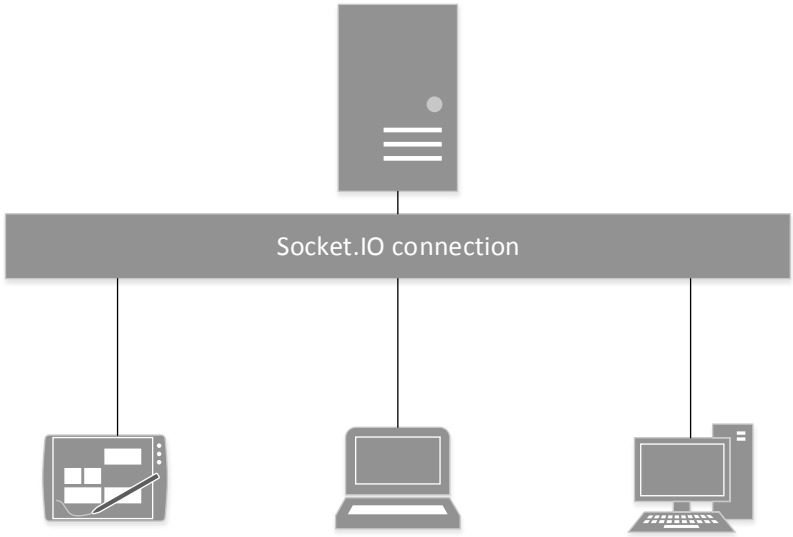
2.3.1 jQuery

För att underlätta manipulation av det innehåll som visas i extensionens gränssnitt används ramverket jQuery.

Flera olika sorters dokument kan beskrivas med hjälp av en Document Object Model (DOM), vilket i korta ordalag är trädstrukturerad kod som ger en bild av dokumentets olika element, till exempel titel, rubriker, bilder och brödtexter (Le Hégaret 2002). En webbläsare bygger utifrån en HTML-fil en sådan DOM, som sedan ritas upp i webbläsarens fönster. JavaScript kan sedan användas för att i efterhand ändra på DOM:en, och för att underlätta denna procedur finns ramverket jQuery. Förutom DOM-manipulation kan jQuery användas för bland annat händelsehantering och asynkrona anrop (The jQuery Foundation 2015). jQuery är ett mycket vanligt verktyg i webbutvecklarens arsenal eftersom det är byggt med prestanda och kompatibilitet i åtanke och därmed överbryggar skillnader de vanligaste webbläsarna på marknaden. Enligt statistikbyrån BuiltWith.com använder omkring 66 % av Internets mest populära webbsidor jQuery, och 17 % av alla webbsidor på Internet (BuiltWith u.å.).

2.4 Socket.IO

För att implementera själva kommunikationen mellan Android-applikationen och instickspprogrammet till Chrome används ramverket Socket.IO. Anledningen till detta är dels att det lägger vikt vid prestanda (Socket.IO u.å.), men även att det är portat till flera plattformar och språk, exempelvis Android och iOS. Ramverket fungerar så att klienter skapar en “socket” - en anslutning mot en server (figur 2.5). När de väl är anslutna kan de välja att lyssna efter specifika meddelandetyper, och reagera då dessa meddelanden anländer. En fördel med Socket.IO är att det är mycket flexibelt. Det går att skicka meddelanden direkt till en specifik klient, eller göra ett massutskick till alla klienter. Meddelanden kan även ordnas i namnrymder (som liknar URL:er) och rum.



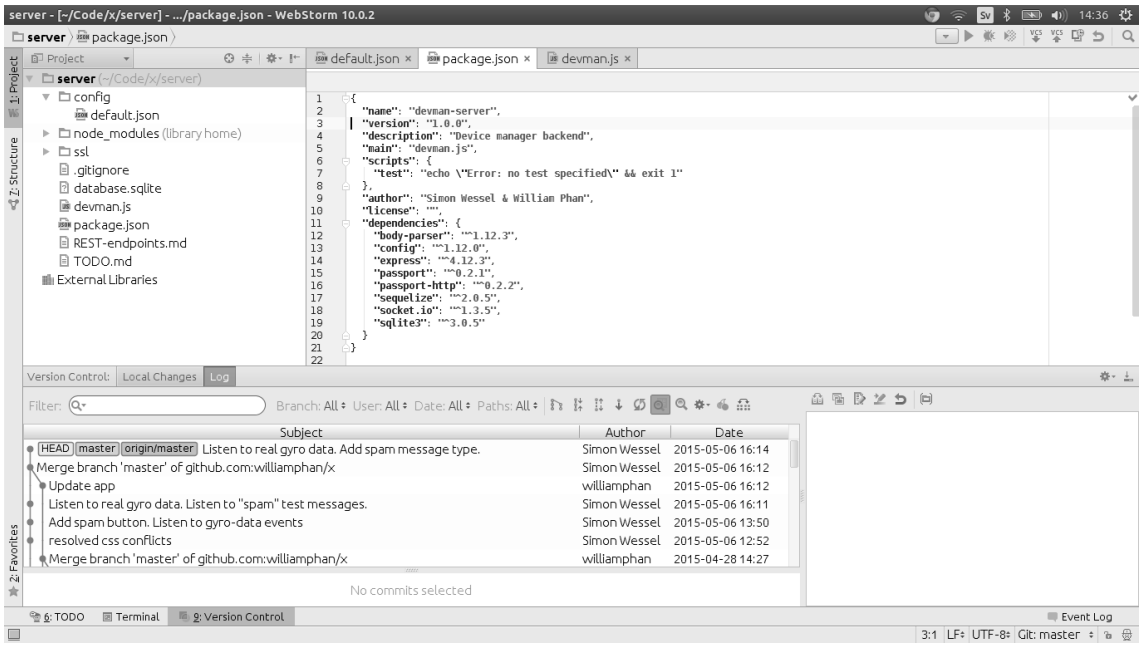
Figur 2.5: Figuren illustrerar hur Socket.IO kräver en server som mellanhand då enheter ska kommunicera med varandra.

För detta examensarbete behövs förmodligen inte all funktionalitet. Socket.IO är enkelt att konfigurera, och har stor flexibilitet. Ramverket passar därför ett projekt med okända behov. Innan valet gjordes undersöktes även WebRTC. Jämfört med Socket.IO kan WebRTC innebära en möjlig prestandavinst, särskilt i de fall stora datamängder ska överföras. Socket.IO har ändå valts, då det lämpar sig utmärkt för signalering, vilket är ett av examensarbetets huvudproblem.

2.5 Övrigt

2.5.1 Kodredigering

För kodredigering i server- och Chrome-komponenterna används utvecklingsmiljön WebStorm. WebStorm utvecklas av företaget JetBrains, som även är kända för IDEA IntelliJ. En fördel med detta verktyg är att det har inbyggt stöd för Node.js och vanliga webbteknologier (JetBrains s.r.o. 2015). Det har funktioner som exempelvis kodkomplettering och versionshanteringsintegration.



Figur 2.6: Utvecklingsmiljön WebStorm.

2.5.2 Versionshantering

Versionshantering sker med hjälp av verktyget Git under prototyputvecklingen. Git är ett distribuerat versionshaneringssystem, vilket innebär att varje utvecklare har en egen kopia av kodbasen (Git u.å.). Detta skiljer sig från centraliserade system där kodbasen finns på ett ställe och där utvecklarna hämtar ut de filer de vill arbeta på, varvid de låses, och sedan lämnar tillbaka dem när de är klara. Ett distribuerat system är att föredra för prototyputveckling, eftersom den sker på olika platser med varierande nätverksåtkomst. En annan anledning är att Git vinner allt större mark i industrin, och därför kan det vara en bra erfarenhet att ha fått under studietiden.

3 Metod

I detta kapitel beskrivs den metod som använts under examensarbetet. Metoden är uppdelad i två huvuddelar, först en förstudie som sätter ramarna för utvecklingsarbetet, därefter följer utvecklingen av prototypen. Figur 3.1 ger en översikt över examensarbetets metod.

Den första delen, 3.1, har att göra med inhämtande av den domänkunskap som krävs för att utveckla den prototyp som är målet med examensarbetet.

I avsnitt 3.1.1 görs en intressentanalys för att identifiera de personer som bör kontaktas under examensarbetets gång.

I avsnitt 3.1.2 definieras den kravhantering som används under examensarbetet. En viktig del i kravhanteringen utgörs av brainstormingmöten.

Avsnitt 3.1.3 beskriver den litteraturstudie som ska ge en bättre förståelse för existerande tekniker och lösningar, och hur de skulle kunna användas för att besvara problemformuleringarna.

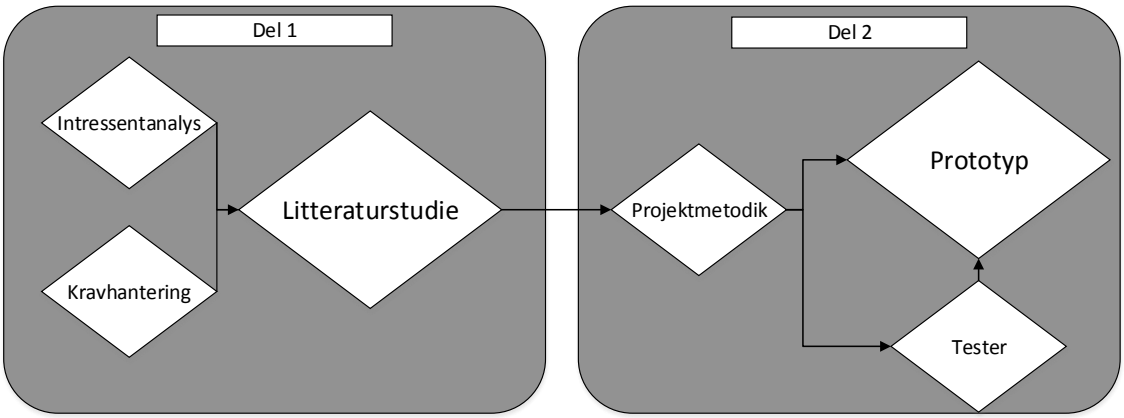
I avsnitt 3.1.4 utvärderas de metoder som beskrivs i avsnitten 3.1.1, 3.1.2 och 3.1.3.

Den andra delen, 3.2, beskriver hur utvecklingen av prototypen ska gå till. I avsnittets inledning ges en redogörelse för de utvecklingsmetodiker som övervägts.

I avsnitt 3.2.1 presenteras den utvecklingsmetodik som slutligen valts för utvecklingsarbetet.

Avsnitt 3.2.2 beskriver de användartester som ska genomföras.

I 3.2.3 ges en utvärdering av utvecklingsmetodiken som beskrivs i 3.2.



Figur 3.1: Arbetsflödet för del 1 och del 2.

3.1 Del 1

I det här avsnittet beskrivs den metod som användes för att inhämta den domänkunskap som behövdes för att genomföra utvecklingen av prototypen.

3.1.1 Intressentanalys

En översiktlig intressentanalys gjordes för att identifiera vem som borde tillfrågas vid utformandet av prototypen, men också för att hitta lämpliga testpersoner. Intressentanalysen hade formen av ett internt brainstormingmöte, där de följande frågorna försökte besvaras:

- Vem är beställaren av prototypen?
- Vem kan ha viktig kunskap om funktionalitetsdelning?
- Vem ska använda funktionalitetsdelning?

Intressentanalysen kom fram till att personal vid Tactel är en viktig kunskapskälla. För att få en bred representation av människor vid användartestningen borde testpersonerna ha olika teknisk bakgrund. Examensarbetets nyckelintressenter är:

- Enhetschef på Tactel
- Projektledare/handledare på Tactel
- Creative director på Tactel
- Användare med gedigen teknisk bakgrund
- Användare utan teknisk bakgrund

Enhetschefen var den första kontakten som upprättades med Tactel och i projektets början den enda kontakten. En bit in i projektet utsågs en handledare på företaget, en projektledare.

En projektledare på Tactel åtog sig rollen som handledare inom företaget. Denna person är nyckelintressent på grund av sin kunskap om projektledning, prototyputveckling och förmåga att bedöma vad som är rimligt att genomföra inom ramen för ett examensarbete. En annan viktig egenskap hos projektledaren är dennes tekniska kunnande och kontakter mot företagets specialister. Genom projektledaren upprättades kontakten med nästa nyckelintressent.

Nästa nyckelintressent är creative director på Tactel. Denna person hade arbetat med att ta fram funktionalitetsdelning som koncept och var en viktig källa för kravelicitering.

De sista intressenterna är användare med varierande teknisk kunskap. Den utvecklade prototypen användartestades för att finna eventuella fallgropar. Eftersom prototypen var tänkt att utforska ett scenario som i dagsläget inte har någon standardlösning, spelade användartester en viktig roll i att identifiera grundläggande designproblem som till exempel användarflöden. För närmare beskrivning av testpersonerna, se avsnitt 3.2.2 samt appendix B.

3.1.2 Kravhantering
Examensarbetet var mycket öppet. Tactel hade inga konkreta krav, varken på undersökningen eller prototypen. Tactel ville undersöka om det är värt att arbeta vidare med funktionalitetsdelning som koncept. På grund av den korta återkopplingen med Tactel och examensarbetets öppna målbild ansågs user stories tillgodose kravhanteringsbehoven. En mer formell kravspecifikation hade gett tydligare målbild, men hade också tagit mer tid i anspråk.

För att ha någon slags kravtext att utveckla prototypen mot genomfördes vid projektets början ett antal brainstormingmöten. Vid dessa närvarade projektgruppen och någon av Tactel-representanterna som nämns i avsnitt 3.1.1. Under mötena togs möjliga scenarion för funktionalitetsdelning upp, vilka sedan sammanställdes som user stories. Brainstormingmöten är enligt Lauesen (2002) en lämplig metod för att ta fram krav för nya system.

I samråd med handledare och creative director valdes följande user story (A.2, appendix A) ut som mål för prototypen:

As a Google Maps user, I want to be able to use my mobile device's gyroscope as input, so I can control the street view orientation by moving my device.

Valet baserades på att tekniken bakom denna user story skulle kunna användas av Tactel vid presentationer och demonstrationer.

3.1.3 Litteraturstudie
Informationsinsamlingen tog sin utgångspunkt i litteraturstudier men även undersökningar av liknande problem och deras lösningar. Metoden bör betraktas som kvalitativ, då det till stor del handlar om att läsa och tolka texter.

En del av datainsamlingen bestod i att identifiera en gångbar arkitektur och för vilka plattformar prototypen skulle utvecklas. Denna insamling utgick främst från jämförelse av olika existerande produkter och plattformar. Det kan finnas produkter som har en passande arkitektur, men ett annat användningsområde. En viktig del var också att undersöka vilka API:er och protokoll som fanns tillgängliga på de plattformar som var aktuella för prototypen.

3.1.4 Utvärdering av metod, del 1
Intressentanalysen som gjordes visade sig värdefull ur två perspektiv. Dels hjälpte den till att identifiera de personer som kunde tillföra värdefull information. Den gav också upphov till insikten att användartesterna skulle genomföras på testpersoner med varierande grad av teknisk bakgrund.

Den user story-baserade kravhanteringen fungerade, men kunde ha inneburit stora problem. De första mötena med Tactel mynnade ut i att begreppet "sessionsöverföring" etablerade sig som mål för undersökningen. Efter några möten framgick det att

“funktionalitetsdelning” förmodligen passade bättre. Denna begreppsförvirring hade förmodligen kunnat undvikas med en striktare och mer formell kravhantering. En positiv effekt av den lösare kravhanteringen har varit att den uppmuntrat en undersökande attityd vilket troligtvis gynnat ett examensarbete som detta.

Litteraturstudien skedde främst via internet vilket lämpade sig väl. Den tryckta litteraturen finns tillgänglig som e-resurser och dokumentation till utvecklingen finns att tillgå på webben.

3.2 Del 2

Till en början var prototypens omfattning väldigt öppen, och fastställdes inte förrän examensarbetets första del (avsnitt 3.1) var genomförd. Detta för att säkerställa att det som skulle utvecklas var passande och tillförde något nytt enligt projektets huvudmål. De user stories[4] som var lämpliga som utgångspunkt för examensarbetet användes och de som var mer omfattande kunde bortses från.

Då alla user stories inte kunde uppfyllas inom tidsramen, valdes endast en user story som projektets basmål. Genom att implementera de icke-triviala delarna och visa att prototypen fungerar kan man argumentera för att konceptet som helhet är värt att vidareutveckla. Den user story som valdes ut (A.2, appendix A), undersöktes mer noggrant och bröts ner i tasks (figur 3.2).

Utifrån den information som togs fram i examensarbetets första del (avsnitt 3.1) och den user story som valdes (A.2, appendix A), valdes utvecklingsmetoder som samverkar på ett bra sätt. Det här avsnittet presenterar tankar som lyfts och de val som gjorts under processens gång.

Då en projektmetodik skulle väljas framtogs följande krav på projektmodellen:

1. Modellen måste vara **lättrörlig** eftersom kunskapen om målet till en början var väldigt begränsad och troligen skulle ändras under projektets gång.
2. Modellen måste **minimera överflödigt arbete**. Med begränsade resurser måste överflödig byråkrati och dokumentation undvikas.
3. Modellen bör ha så få **organisatoriska krav** som möjligt. Att arbeta efter en modell som kräver till exempel fem roller, och fördela dessa på två personer kan upplevas som rörigt.

Med utgångspunkt i krav 1 ovan undersöktes projektmetodikerna RUP, Scrum och Kanban närmare, eftersom det redan fanns kännedom om dessa inom projektgruppen.

Rational Unified Process - RUP (IBM 2003) är uppbyggd som ett ramverk för iterativ mjukvaruutveckling. Meningen med RUP är att man anpassar metoden efter projektets behov. Denna modell användes inte på grund av att det skulle krävas betydande efterforskningar, dels för att lära sig modellen och dels för att anpassa den för projektet.

[4] Se 3.1.2 samt Appendix A.

Det ansågs att detta var tid som bättre kunde läggas på arbete som rapportskrivning och utveckling, och RUP var därmed inte förenligt med krav nummer två.

Ett önskemål på den valda projektmetodiken var ett visuellt arbetsflöde. Sedan tidigare fanns det erfarenhet av att jobba med visuella arbetsflöden, vilket sågs som en stor tillgång. Detta önskemål sammantaget med ovanstående krav resulterade i att man valde att titta närmare metoder inom Lean/Agile-fåran, mer specifikt Scrum och Kanban.

3.2.1 Utvecklingsmetodik - Kanban-ish

Två projektmodeller valdes som inspirationskällor, Kanban och Scrum (Kniberg & Skarin 2010). De delar många egenskaper, men har även en hel del skillnader. Båda metoderna är agila, eller lättrörliga. Som grund för de agila metoderna ligger Manifesto for Agile Software Development (2001). En princip agil utveckling tar fasta på är att man värderar individer och interaktioner högre än processer och verktyg. Alla projekt innebär ett visst mått av båda delar, men manifestet menar att för stort fokus på processen och dokumentation kan göra att personal och slutprodukt glöms bort.

Kanban är från början ett schemaläggningssystem för just-in-time-produktion. Systemet har anammats inom mjukvaruutveckling. Produktionen synliggörs på en tavla (en så kallad Kanban board). Tavlan har ett antal kolumner som beskriver olika stadier i produktionen (till exempel To do, Doing och Done). De olika arbetsuppgifterna förs in på kort i kolumnen längst till vänster, och allt eftersom arbetet fortskrider flyttas korten med respektive arbetsuppgift åt höger på tavlan. Kanban begränsar hur många arbetsuppgifter som får finnas i varje kolumn. Detta har effekten att Kanban blir ett pull-system. Om en kolumn är full måste man dra en arbetsuppgift vidare till nästa kolumn innan man kan börja på något nytt. I Kanban mäter man ledtid, d v s den tid det tar från att en arbetsuppgift påbörjas tills det att den avslutas. Systemet anpassas sedan genom exempelvis ändring av kolumnbegränsningar för att ledtiden ska bli så kort som möjligt (Kniberg & Skarin 2010).

Allt arbete som ska utföras i ett Scrum-projekt förs in i en backlog. Scrum är baserat på iterationer, eller sprintar. Före en ny sprint bestämmer sig teamet för vilka arbetsuppgifter i backlogen man ska implementera. Därefter börjar sprinten, och arbetsuppgifterna sätts upp på en tavla liknande den man använder i Kanban. Under sprinten, allt eftersom arbetsuppgifterna implementeras, vandrar de över tavlan från vänster till höger. Vid sprintens slutförande rensas tavlan, och teamet genomför ett så kallat retrospective. Här utvärderas sprinten i syfte att effektivisera processen inför nästa sprint. Scrum föreskriver två roller förutom utvecklingsteamet. Dessa är produktägare och scrum master. Produktägaren är med och tar fram backlogen och prioriterar de olika arbetsuppgifterna. Scrum mastern hjälper utvecklingsteamet att undanröja hinder, ofta med hjälp av så kallade scrum-möten eller stand-up-möten (Kniberg & Skarin 2010).

Båda modellerna uppfyller alltså kraven på ett visualiserat arbetsflöde tack vare sina tavlor. De fungerar dock på något olika sätt. Kanban tar in nya arbetsuppgifter efter kapacitet. Produktägaren (eller motsvarande) kan när som helst gå in och omprioritera

de arbetsuppgifter som ännu inte påbörjats om man valt att använda någon form av prioritering. I Scrum låser man istället fast ett antal arbetsuppgifter och förbinder sig att implementera dem inom den kommande sprinten. När arbetsuppgifterna väl är låsta kan de inte ändras.

Här följer några av de för detta projekt viktigaste skillnaderna mellan de två metoderna (tabell 1).

	Kanban	**Scrum**
Föreskrivna roller	Nej (valfritt).	Utvecklingsteam, produktägare och scrum master.
Iterationer	Nej.	Ja.
Begränsning av arbetsbelastning	Maxgränser på tavlans kolumner.	Begränsning av arbete per sprint.
Nya arbetsuppgifter tillåts	Om ledig kapacitet finns.	Ej under pågående sprint.
Tavlan återställs	Aldrig.	Efter varje sprint.
Teamsammansättning	Valfritt.	Korsfunktionellt.

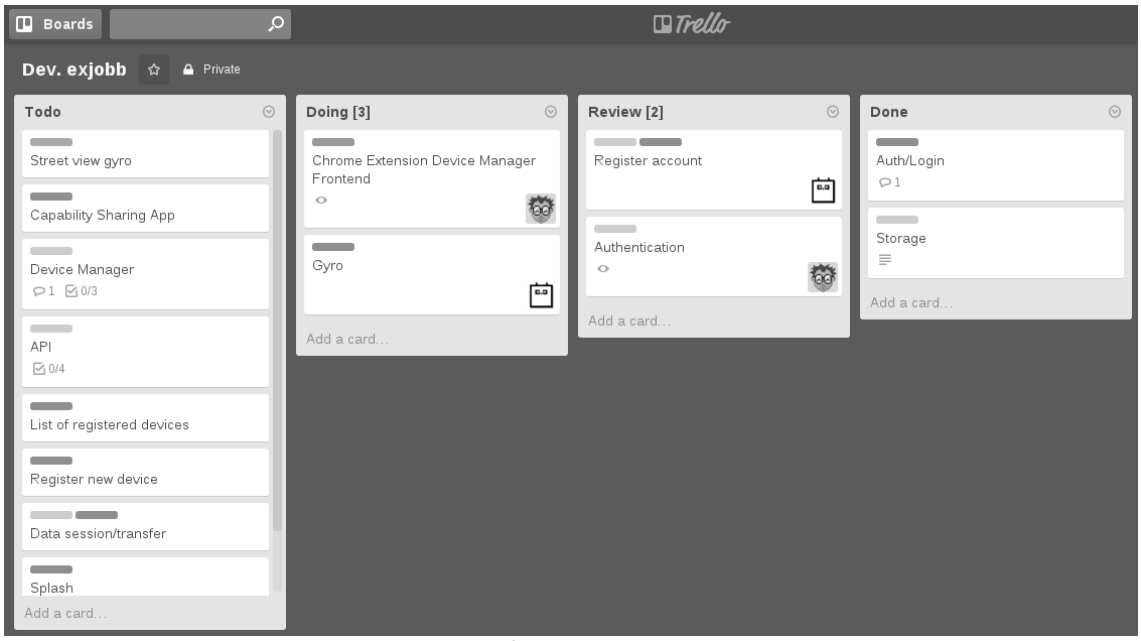
Tabell 1: Viktigaste skillnaderna mellan Kanban och Scrum för examensarbetet.

De två första kraven på en lättrörlig modell utan onödig dokumentation ansågs mötas av både Scrum och Kanban. Båda modellerna förespråkar en nära kontakt med kunden. Eftersom mycket av arbetet utfördes på plats hos kunden var det en självklarhet att dra nytta av denna korta återkoppling. Det tredje kravet tillgodoses bättre av Kanban än av Scrum, eftersom Kanban inte förskriver några roller. Ett varningens finger måste dock höjas inför en så fri metod. Det krävs en stor tillit inom utvecklingsteamet för att en sådan metod ska fungera optimalt. Utan tillit är föreskrivna roller ett utmärkt sätt att fördela ansvar i en projektgrupp.

Ytterligare en anledning att välja Kanban var att iterationer eller sprintar inte skulle tillföra så mycket till detta projekt. Under examensarbetets gång bestod första delen mest av litteraturstudie och rapportskrivning, för att sedan gå över i rapportskrivning varvat med implementation. Upplägget skulle helt enkelt kräva sprintar av varierande längd och man skulle då gå miste om den återkommande rytm sprintarna var tänkta att tillföra. Kanban tar istället in nya arbetsuppgifter allt eftersom kapacitet blir tillgänglig. Denna modell var mer förenlig med projektets upplägg.

Skillnaden i hur tavlorna fungerar var ännu en anledning att välja Kanban. Eftersom projektet inte hade något behov av iterationer hade man heller inget behov av delleveranser vid specifika tidpunkter. Därmed fanns heller ingen poäng med låsa (eller "committa" enligt Scrumterminologi) vissa arbetsuppgifter. Tvärtom, sågs en styrka

med att när som helst kunna omdirigera och byta vad man jobbade på, just eftersom målet var så otydligt.



Figur 3.2: Bild på arbetsflödet som användes.

Figur 3.2 visar det visuella arbetsflödet. Fyra olika kolumner beskriver tillståndet för olika tasks. Notera siffrorna inom hakparenteserna. De anger kolumnens "WIP limit". I figuren är kolumnen "Review" full, vilket innebär att en task måste flyttas vidare till "Done" innan en ny task kan komma in i "Review". Detta har effekten att ett pull-system bildas. Tasks dras framåt i systemet och lämnar outnyttjad kapacitet efter sig dit nya tasks kan dras.

Kolumnen "Review" infördes eftersom olika personer skulle ha huvudansvar för olika komponenter i prototypen. För att öka graden av kunskapsdelning och minska missförstånd gjordes kodgranskningar, eller kanske snarare presentationer där personen som implementerade koden gick igenom och förklarade den för den andre personen.

3.2.2 Tester

Under prototyputvecklingens gång utfördes användartester. Detta för att upptäcka potentiella problem ur en användares perspektiv. Många av dessa problem och användningsfall är inte alltid lätta att hitta som utvecklare. Testernas utformning följde det format Lauesen (2002) beskriver som slutna och meningsfulla uppgifter utan dold hjälp för testpersonen. Resultatet av testerna dokumenterades (appendix B) för att kunna användas vid en eventuell vidareutveckling.

För dessa tester var det främst användbarheten som sattes i fokus. Registrering, inloggning, och aktivering av funktionalitetsdelning skulle genomföras av användare med varierande teknisk bakgrund. Det är till stor del fråga om mjuka värden, eftersom det är användarens upplevelse och känsla som står i fokus. För detta ändamål ansågs enkla användartester ge stor vinst i förhållande till kostnaden.

Användartesterna (appendix B) genomfördes då en första fungerande prototyp byggts. Serverkomponenten och webbläsaren med extensionen kördes på en bärbar dator och Android-applikationen var installerad på en mobiltelefon. Användaren fick en kort introduktion till vad funktionalitetsdelning är, samt en påminnelse om att det var programmet som skulle testas och inte testpersonen. Därefter fick användaren uppgifter att utföra. Användaren uppmanades berätta högt vad vederbörande tänkte under tiden testet pågick. Efter testet välkomnades synpunkter på mjukvaran och upplevelsen som helhet.

För att inte styra användaren gavs endast nödvändig information under tiden testerna pågick, och det var således upp till användaren att välja väg själv. Testerna utgick från att programmen hade installerats korrekt och startats på förhand.

För att få goda resultat utfördes testerna utföras av fyra personer. Enligt Virzi (1992) täcker det ungefär 80 % av användbarhetsproblemen. Med fler än fyra testpersoner kan fler problemområden identifieras, men det tar också värdefull tid i anspråk, och ansågs därmed inte ge tillräckligt stor vinst.

Som testpersoner eftersträvades så stor spridning som möjligt med avseende på kunskaper och intressen. Önskade testpersoner karakteriseras på följande sätt:

- **Mjukvaruutvecklare.** Har gedigen teknisk bakgrund med erfarenhet av programvaruutveckling samt test.
- **Teknisk ointresserad.** En person som inte följer teknikutvecklingen och gärna stannar i det gamla. Förlitar sig heller inte för mycket på tekniken och lär sig endast något nytt vid behov.
- **Studerande.** Följer gärna teknikutvecklingen och behärskar en del grundläggande kunskaper inom utveckling. Har en teknisk bakgrund och är en användare som gärna testar det senaste.
- **Teknikintresserad, men inte insatt.** Följer teknikutvecklingen och använder gärna tekniker som är välbeprövade.

På detta sätt fångades flera perspektiv och synvinklar upp vilket ledde till att problem identifierades. En närmare beskrivning av testerna finns i appendix B.

3.2.3 Utvärdering av metod, del 2

Tidigt i processen valdes en agil utvecklingsmetodik, vilket i efterhand har visat sig vara ett passande val. Den första prototypen som skulle utvecklas kom att bero på en specifik funktion i webbläsaren, som av säkerhetsskäl hade lämnats stängd. Tack vare projektmetodikens lättrörlighet och att den inte lade så stor vikt vid dokumentation, kunde man snabbt gå tillbaka till ritbordet och ta fram en ny prototyp som demonstrerar funktionalitetsdelning. Med striktare krav på dokumentation hade dyrbar tid behövt läggas på dokumentation av problemet och den nya prototypen.

Testerna belyste på ett tydligt sätt brister i användbarheten, mer om detta i kapitel 5. Med tanke på vinsterna var förberedelsetiden väl investerad. Varken förberedelserna eller själva genomförandet tog särskilt mycket tid i anspråk, men lyckades ändå identifiera några problem. I ett par av testerna misslyckades testledaren med att ge tillräcklig information innan testet startade. Testerna gav ändå värdefull feedback, men vikten av att testledaren har ett tydligt formulerat scenario är en bra lärdom.

3.3 Källkritik

I detta avsnitt presenteras de källor som använts. Avsnittet är uppdelat i två delar; källor från litteratur och källor från Internet. Som utgångspunkt för källkritiken har Alexandersons *Källkritik på Internet* (2012) använts. För varje källa har följande frågor ställts:

- Vem är författaren?
- Vem är den tänkta mottagaren?
- Varför har källan författats?
- När författades den?
- Hur har den publicerats?

3.3.1 Källor från litteratur

Alapetite (2009) kan anses pålitlig eftersom den är publicerad i Personal and Ubiquitous Computing, en referentgranskad tidskrift med redaktörer från välrenommerade universitet och organisationer.

Alexanderson (2012) kan anses pålitlig eftersom det är en guide i källkritik publicerad av IIS. IIS – Stiftelsen för Internetinfrastruktur – är en organisation hängiven åt Internets utveckling, och att de ger ut en text som denna ligger i linje med deras syfte att utveckla Internet.

Bagrodia et al. (2003) kan anses pålitlig eftersom den presenterats vid MobiSys, en konferens som sponsras av intresseorganisationen SIGMOBILE. I organisationens styrelse sitter representanter dels från olika universitet, men också från storföretag som Microsoft och Google.

Gomez och Paradells (2010) kan anses pålitlig eftersom den publicerats i tidskriften IEEE Communications Magazine.

Kniberg och Skarin (2010) kan anses pålitlig eftersom de har mångårig erfarenhet av agila metoder. Boken har även använts som kurslitteratur inom högre utbildning.

Mack et al. (2005) kan anses pålitlig eftersom den citerats och använts flitigt som metodreferens i verk publicerade av såväl universitet som myndigheter.

Lauesen (2002) kan anses pålitlig eftersom den citerats flitigt och används som kurslitteratur. Boken är mycket omfattande och ger flera perspektiv på kravhantering.

Phan et al. (2001) kan anses pålitlig eftersom den citerats flitigt, både i andra artiklar och i patent. Arbetet har dessutom presenterats på en IEEE-workshop. IEEE kan därmed sägas stå som garanter för att arbetet håller en viss kvalitet.

Pippal et al. (2013) har inte citerats av andra skrifter och bör därför användas försiktigt. I denna rapport har den huvudsakligen syftet att visa att forskning på sessionsöverföringar pågår så sent som 2013, och det anses därför vara i sin ordning att referera till denna artikel.

Shacham et al. (2009) kan anses pålitlig eftersom den är publicerad på standardiseringsorganisationen IETF:s domän.

Virzi (1992) kan anses pålitlig eftersom den publicerats i den referentgranskade tidskriften The Journal of the Human Factors and Ergonomics. Virzis rekommendationer har sedan publiceringen kommit att bli vedertagen praxis.

3.3.2 Källor på Internet

Ashton (2009) kan anses pålitlig eftersom artikeln beskriver ett begrepp myntat av författaren, och vad vederbörande menar med begreppet. Artikeln är publicerad i RFID Journals webbtidning. RFID Journal är en organisation som rapporterar om vad som händer inom området RFID. Organisationens tidningar publicerar artiklar skrivna av akademiker och industrirepresentanter inom området.

Asus (u.å.a.) och (u.å.b.) är produktbeskrivningar från tillverkaren själv. Källorna anses i detta sammanhang trovärdiga eftersom syftet är att visa att produkterna finns, men man bör vara medveten om att sidan förmodligen försöker framställa produkterna i så god dager som möjligt.

Beck et al. (2001) kan anses pålitlig, eftersom deras agila manifest anammats i industrin, och även refererats till i forskning som rör projektmetodik. Manifestet tillkom efter att författarna efter flera år i mjukvaruindustrin formulerade i sina ögon bra principer för utvecklingsprojekt.

BuiltWith (u.å.) är ett företag som säljer statistik om webbteknologier. Källan kan anses pålitlig eftersom de har flera välkända kunder och därmed även har ett rykte att uppehålla.

Git (u.å.) kan anses pålitlig eftersom detta är den officiella dokumentationen för verktyget.

IBM (2003) kan anses pålitlig eftersom IBM dels äger företaget som utvecklade RUP, och dessutom säljer en produkt för processutveckling med stöd för RUP. IBM har därför ett ekonomiskt intresse i att informationen är korrekt.

JetBrains s.r.o. (2015) kan anses pålitlig eftersom det är en produktbeskrivande webbsida. Man bör dock vara medveten om att denna sida förmodligen inte lyfter fram produktens eventuella nackdelar.

Joyent Inc. (2015) kan anses pålitlig eftersom sidan beskriver en open source-produkt som används i stora system av välkända företag, som exempelvis Microsoft och PayPal.

Le Hégaret (2002) kan anses pålitlig eftersom den publicerats på World Wide Web Consortiums (W3C) hemsida. W3C är en organisation som arbetar med att standardisera webbteknologier.

Mell och Grance (2011) kan anses pålitlig eftersom publikationen är en officiell handling utgiven av USA:s statliga standardiseringsorganisation.

Videon *Neptune Suite - Seamless, Secure and Portable* (2015) används i denna rapport i syfte att lyfta framtida möjliga utvecklingar. Produkten videon visar är endast en prototyp och ska ses i det ljuset.

Pushbullet Inc. (u.å.) kan anses pålitlig eftersom det är en produktbeskrivning. Man bör vara medveten om att källan förmodligen vill framställa produkten i så god dager som möjligt. Pushbullet Inc. har antytt att man i framtiden ska lansera en premiumtjänst, och företaget kan därmed förmodligen anses vara måna om sitt rykte och kvaliteten på produkten.

WhatsApp Inc. (u.å.) kan anses pålitlig eftersom det är en produktbeskrivning. Man bör vara medveten om att källan förmodligen försöker framställa produkten i så god dager som möjligt.

Wood (2015) kan anses pålitlig eftersom syftet med referensen är att påvisa att det finns ett starkt intresse för Internet of Things. Artikeln är publicerad i den brittiska dagstidningen The Guardian.

Google
Denna rapport refererar på ett flertal ställen till olika Google-webbsidor. De fem första, (2014), (2015), (u.å.a.), (u.å.b.) samt (u.å.c) är avsedda som dokumentation för sina respektive produkter, och kan därför anses vara pålitliga. Den sjätte referensen - *Google I/O Keynote* - är en film från Googles årliga utvecklarkonferens. Filmen är väldigt säljande och sensationsbetonad. Statistiken bör ändå kunna användas i syfte att betona Androids framgång.

Ramverk
Express (u.å.), The jQuery Foundation (2015), Passport (u.å.), Sequelize (u.å.) samt Socket.IO (u.å.) får alla anses pålitliga, då de utgör den officiella dokumentationen för respektive ramverk. Generellt hade dokumentationen bra kvalitet med två undantag. jQuery hade exceptionellt ingående och tydlig dokumentation, medan Socket.IO:s dokumentation upplevdes som svårförstådd och förvirrande.

4 Analys

Insamlade data från 3.1 utgörs dels av litteraturstudier, men även observationer av produkter som löser liknande problem. Enligt Mack, Woodsong, MacQueen, Guest och Namey (2005) kännetecknas kvalitativa studier av att de utforskar ett fenomen, snarare än att de bekräftar en hypotes. Kvalitativa studier är ofta iterativa och problemformuleringarna anpassas efter dragna lärdomar. Mack et als beskrivning ligger i linje med detta examensarbete, och insamlad data är därför att betrakta som kvalitativ.

Den första utgångspunkten för examensarbetet var att besvara frågor om sessionsöverföringar, men tack vare analysens undersökande natur kom den så småningom istället att rikta in sig mot funktionalitetsdelning, som kan ses som ett specialfall av sessionsöverföring. På grund av detta handlar avsnitt 4.1 om just sessionsöverföringar mellan enheter.

Avsnitt 4.2 handlar om visionen “Internet of Things”, och hur den relaterar till denna rapport.

Avsnitt 4.3 handlar om funktionalitetsdelning. Det tar sin utgångspunkt i tanken på att olika enheter i framtiden samverkar och lånar av varandras funktionaliteter. Huvudmotivationen är ett bättre utnyttjande av för användaren tillgänglig funktionalitet, och därmed en bättre användarupplevelse. I detta avsnitt identifieras ett antal nyckelenheter och deras funktionaliteter.

I avsnitt 4.4 görs en ansats att identifiera funktioner nödvändiga för funktionalitetsdelningssystem. Funktionerna och egenskaperna beskrivs i mer generella ordalag.

Det sista avsnittet, 4.5, formaliserar föregående avsnitt och lägger fram ett designförslag för ett funktionalitetsdelningssystem. Designen utgör grunden för den prototyp som presenteras i denna rapports resultatdel.

4.1 Sessionsöverföring

Idag är molnteknik en vanlig lösning för att dela datorresurser mellan olika typer av enheter. Med datorresurser menas exempelvis lagring, beräkningskapacitet och nätverk (NIST 2011). Innan molntekniken blev det vanligaste sättet att som användare växla mellan olika enheter (som mobiltelefon, bärbar eller stationär dator), forskades det en hel del kring hur man kan överföra användarsessioner mellan olika enheter. De tekniker som tagits fram för sessionsöverföring kan spela en roll i utformandet av en modell för funktionalitetsdelning. Därför ges här en historisk överblick över tidigare tekniker och hur de påverkat dagens tekniker.

Alapetite (2009) visar hur 2D-streckkoder/QR-koder kan användas för att överföra sessions-id och därmed komma åt samma användarsession på flera enheter. Han menar att streckkoder är att föredra framför mer hårdvarubaserade lösningar så som NFC, eftersom de är billiga och finns tillgängliga på många plattformar.

Även om sessionsöverföringar inte görs på detta sätt längre, är QR-koder fortfarande gångbara som autentiseringstekniker. Som exempel, så använder chatt-applikationen WhatsApp QR-koder för autentisering i sitt web-gränssnitt (WhatsApp Inc. u.å.).

TWIST (Phan, Guy, Gu & Bagrodia 2001) är en ansats att definiera en arkitektur för sessionsöverföring. Här används en middleware server (MWS) mellan applikationsservern och klienten. MWS:en har till uppgift att ge klienten de resurser som efterfrågas, filtrerade på ett sådant sätt så att klienten kan presentera dem (exempelvis nedskalning av bilder för att passa en enhet med begränsad bandbredd). MWS:en agerar också som cache för sessioner som ska överföras. En intressant vinkling är att då en session tillfälligt cachas på MWS:en, är det inte all sessionsdata som sparas, utan endast de operationer som utförts på ursprunglig data.

Delar av denna funktionalitet används idag, när man anpassar den efterfrågade resursen efter vilken sorts enhet som gör förfrågan, men själva sessionsöverföringen är alltså inte lika aktuell som teknologi.

Att överföra en användarsession från en enhet till en annan är ett säkerhetskritiskt företag. I ett försök att åstadkomma säker sessionsöverföring har Pippal, Jaidhar och Tapaswi (2013) designat ett protokoll för ömsesidig autentisering med smart cards. Enheter registreras på förhand, och den bakomliggande applikationsservern vet därför vilka enheter som är tillåtna som mottagare för sessionsöverföringar.

Pushbullet används för att skicka data och notifikationer mellan enheter (Pushbullet Inc. u.å.). Som användare måste man på förhand registrera de enheter man vill använda, och är därför ett bra exempel på hur man använder en teknik lik den föreslagen av Pippal et al.

Funktionalitetsdelning och sessionsöverföring är forskningsområden som till viss del överlappar. I de fall man vill ha tillgång till samma resurs i mer än en enhet behöver

man nödvändigtvis inte dela funktionalitet från en enhet till en annan, utan det blir då fråga om någon sorts sessionsöverföring eller -delning för att enheterna ska kunna samverka. För att implementera denna delning kan man antingen hålla sessionen gemensam, så att varje enhet kommer åt resursen oberoende av andra enheter, eller så förekommer sessionen endast i en upplaga och överförs till den enhet där resursen är tänkt att användas. Det förra fallet ger möjligheten till simultan användning av den efterfrågade resursen. Det senare fallet lämpar sig möjligen bättre för exklusiv användning, d v s att endast en enhet kommer åt resursen åt gången. Naturligtvis finns ett spann mellan dessa ytterligheter. Man skulle till exempel kunna tänka sig att en användare av en videoströmmningstjänst kan ansluta max tre enheter till tjänsten och därefter betala extra för att kunna ansluta fler enheter. Man vill ändå dela sessionen mellan de olika enheterna för att komma ihåg sedda filmer och senaste uppspelningspositionen i olika filmer. Märk väl att varje användningsfall kräver sin egen definition av vad som utgör en session.

Ett fall där exklusiv sessionsöverföring skulle kunna förekomma är då en användare har en aktiv webbsession på exempelvis en dator, och vill byta till en annan enhet, till exempel en mobiltelefon. Istället för att påbörja en ny session, vill användaren kanske istället göra en sessionsöverföring och återuppta arbete som redan påbörjats på mobiltelefonen.

Stöd för denna sorts sessionsöverföring finns delvis idag och kan hittas i vissa applikationer och plattformar. Det stora hindret för denna teknologi är att lösningarna är väldigt applikationsspecifika. Det finns flera webbläsare som klarar av att dela exempelvis lösenord och öppna flikar mellan olika enheter, men lösningarna är som sagt specifika för respektive applikation. Det finns ingen gemensam standard för denna sessionsdelning, och varje applikation kräver således sin egen sessionshantering i form av användarkonton och inloggningar. För att minska antalet användarkonton en användare måste hålla i minnet utvecklas autentiseringslösningar som exempelvis OpenID Connect. Problemet kvarstår dock, eftersom tjänsten som ska använda OpenID Connect måste utvecklas med stöd för detta.

Att man inte ser sessionsöverföringar, där sessionen bara ges åt en klient åt gången, i någon större utsträckning kan bero på att moln-arkitekturen blivit mer populär. Sessionsöverföringar är svåra att implementera, och vinsterna med moln-applikationer är (förutom åtkomst från flera enheter) exempelvis att applikationen och applikationsdata går att komma åt var som helst så länge det finns en nätverksuppkoppling. Som användare behöver man inte längre bekymra sig för att synkronisera applikationsdata mellan enheter, eftersom data finns i molnet.

4.1.1 Avsaknad av standardsessioner

Ordet “session” är ett väldigt flytande begrepp i tekniska sammanhang. Beroende på vilken kontext man befinner sig i betyder session olika saker. I grunden handlar det om en konversation mellan två parter, och konversationens tillstånd. Tyvärr gör olika kontexter bruk av skilda protokoll, och har olika variabler för att beskriva sessionens tillstånd.

Som exempel på detta kan man först tänka sig en användarsession i ett operativsystem. En användare loggar in på datorn och operativsystemet hanterar sessionen. Användaren öppnar en webbläsare, som kommer ihåg de senaste öppna flikarna. Detta är en annan sorts session. Slutligen loggar användaren in på en webbsida med användarnamn och lösenord, vilket resulterar i tillämpning av sessioner i en tredje bemärkelse. Då dessa befinner sig på olika nivåer och fyller olika syften blir det extremt svårt att generalisera problemet och definiera sessioner på ett sätt som gör att de kan överföras på samma sätt mellan olika enheter. Frågan är om det ens är önskvärt att ha en gemensam standard, eftersom det skulle tvinga att design av nya system följer på förhand givna mönster.

Session Initiaition Protocol (SIP) är ett protokoll för initiering av multimediaströmningssessioner och har använts exempelvis inom VoIP. Shacham, Schulzrinne, Thakolsri och Kellerer (2009) lägger fram en rekommendation för hur SIP-sessioner kan överföras från en enhet till en annan.

WebRTC som nämns i avsnitt 5.2 är ett exempel på teknik som använder SIP för upprättning av förbindelser.

4.1.2 Moln

En vanlig lösning idag är så kallade molnbaserade tjänster. Där användaren tidigare utfört arbete lokalt på sin egen dator, görs detta då istället i "molnet", vilket ofta innebär att tjänsten tillhandahålls av en leverantör, och är tillgänglig på leverantörens infrastruktur via nätverket. Inte sällan är gränssnittet till molnbaserade applikationer webbaserade, och tillgängliga via en webbläsare. Istället för att överföra en session direkt från en enhet till en annan, kan flera klienter ansluta till molntjänsten och komma åt den önskade sessionen. Detta kräver nätverksåtkomst och någon form av autentisering. Fördelen med detta är att användaren inte aktivt behöver hantera sessionsöverföringen, vilket gör det till en smidig lösning. En annan fördel är att applikationsdata inte behöver synkroniseras eller överföras mellan olika enheter, eftersom data också finns i molnet. Nackdelarna däremot är att man just måste blanda in en tredje part då applikationen finns i molnet, och för detta krävs en nätverksanslutning. Om användaren inte kommer åt det nätverk där applikationen finns, går det inte att använda applikationen, och man skulle då kunna argumentera för att användaren inte längre har kontroll över sina data, vilket kan vara ett problem, särskilt för företag och deras eventuella hemligheter.

Ett företag som har kommit en ganska bra bit på väg när det gäller molnbaserade tjänster är Google (Google Inc. u.å.c). Hos dem finner man bland annat Google Docs, som låter användaren spara och redigera dokument i realtid mot en server. Användaren kan komma åt sina dokument från såväl datorn som mobiltelefonen. Google Docs har stöd för samarbete mellan flera användare, och kan i och med senaste webbtekniken även köras i offline-läge. Google Docs får stå som exempel för hur mogen cloudteknologin ändå kan anses vara.

4.1.3 Proxyarkitektur

Ett sätt att hantera sessionsöverföringar är att koppla sin redan existerande applikationsserver till en proxyserver, som i sin tur sköter sessionsöverföringarna. Detta är precis vad Bagrodia, Battacharyya, Cheng, Gerdind, Glazer, Guy, Ji, Lin, Phan, Skow, Varshney och Zorpas (2003) försöker åstadkomma med sin iMASH-arkitektur. Då klienten ansluter till proxyservern skapas en session på proxyn. Om klienten vill överföra sessionen till en annan enhet, tar proxyservern hand om det och applikationsservern behöver inte bry sig. Proxyservern har även som uppgift att presentera olika vyer, anpassade till vilken sorts klient som ansluter. Om klienten är en mobiltelefon visas mobilvyn, om klienten är en dator, visas vyn som är anpassad för större skärmar, o s v.

Sammanfattningsvis kan man säga att sessionsöverföring som metod för att sömlöst kunna byta enhet förmodligen är besegrad av cloudteknologin. Så länge synkroniseringen sker på ett korrekt sätt, finns ingen förlust i att flera enheter kommer åt samma session, och då priser på hårdvara tenderar att gå ner med tiden, blir kostnaden för cloudinfrastruktur allt mer överkomlig samtidigt som prestandakraven på klienterna avtar.

4.2 Internet of Things

Ett begrepp som undersöktes under examensarbetets gång är “Internet of Things”, förkortat IoT. Termen myntades av Kevin Ashton, och har kommit att beteckna en framtidsvision där saker med inbyggd elektronik kan kommunicera med varandra för att underlätta för användaren. Utvecklingen går framåt, och det kommer fler och fler produkter som går att ansluta till Internet och styra med hjälp av tillhörande mobilapplikationer (The Guardian 2015).

Användningsområdena är många. Ett exempel kan vara att använda IoT-teknik för att övervaka ett jordbruk. Komponenterna kan då exempelvis mäta jord- och väderförhållanden, varpå ett automatiskt bevattningssystem anpassar sig efter mätvärdena.

Denna utveckling var dock inte riktigt vad Ashton menade när han första gången använde begreppet. Det han efterlyste var snarare ett sätt för sakerna att kommunicera med varandra, någon slags gemensam standard. För detta krävs adressering, och Ashton är intresserad av RFID-tekniker. I och med IPv6:s införande öppnar det för ännu en möjlighet till unik identifiering av saker, då den har en extremt stor adressrymd.

En annan enligt Ashton önskvärd egenskap är att sakerna måste kunna upptäcka varandra och bilda uppfattningar om vad som finns i deras omgivning, och på det området finns fortfarande mycket arbete att göra (Ashton 2009).

4.3 Funktionalitetsdelning

Om sessionsöverföring innebär att flytta en session från en enhet till en annan, innebär funktionalitetsdelning att flytta en enhetsspecifik funktionalitet till en annan enhet. Områdena är relaterade men har vissa skillnader. Vid sessionsöverföring består resursen man önskar dela av applikationsdata och/eller applikationstillstånd. Samma applikation kan finnas för flera plattformar, och sessionsöverföringen bör därför inte ses som enhetsspecifik. Funktionalitetsdelning handlar om att dela enhetens tillstånd, vilket kan innebära vissa sensordata eller dylikt. Dagens mobiltelefoner är ofta utrustade med flertalet sensorer, två kameror och en skärm av mindre storlek. En dator kännetecknas av större beräkningskapacitet, större skärm, mer lagringsutrymme och avsaknad av i mobiltelefoner vanliga sensorer. Funktionalitetsdelning handlar i sin kärna om att låta dessa olika enheter dra nytta av varandras styrkor för att mitigera deras svagheter.

Till skillnad från Internet of Things handlar det inte om att enheter gör saker utan användarens vetskap för att förhöja användarens upplevelse, utan snarare ett av användaren aktivt val att låna in en specifik funktionalitet eller egenskap från en enhet till en annan under en begränsad tidsperiod.

Under förstudien påträffades områden där funktionalitetsdelning skulle kunna användas, men där andra lösningar utvecklats. Ett exempel på detta är mobiltelefoners funktionalitet att ringa samtal och skicka och ta emot textmeddelanden. Denna funktionalitet finns inte i exempelvis en dator då den saknar SIM-kort. Voice over IP-tjänster (VoIP) som exempelvis Skype och WhatsApp ger användaren möjligheten att dynamiskt växla mellan olika enheter för ringa samtal och skicka meddelanden. Tjänster som förut krävde anslutning till en telefonoperatörs nät får i och med VoIP konkurrens av tekniker som endast kräver nätverksanslutning. Man skulle istället kunna tänka sig att samtals- och textmeddelande-funktionaliteten kan lånas ut från en mobiltelefon till en dator. Detta kommer förmodligen inte hända, då tekniker som VoIP och oberoende meddelandetjänster vunnit mark och dessutom kan användas från fler enheter då i stort sett alla enheter som kan tänkas behöva funktionaliteten redan är utrustade med Wifi eller Ethernet.

Man kan tänka sig scenarier där funktionalitetsdelning kan vara önskvärt. Ett exempel skulle kunna vara en applikation som körs på en surfplatta. Där skulle applikationens detaljvy kunna visas på surfplattans skärm, medan en översiktsvy visas på en stor TV på väggen som lånats in via funktionalitetsdelning. Båda enheter, surfplattan och TV:n, har funktionaliteten att visa en bild på sin skärm, men vilken skärm som lämpar sig bäst kan vara situationsberoende.

För att komma fram till en lösning på problemet kan det kanske vara bättre att tänka sig en samling enheter som ett ekosystem, eller en enda stor och organisk enhet. Funktionaliteter kan komma och försvinna, beroende på vilka enheter som är anslutna till ekosystemet. Innan en lösning föreslås ska några olika enhetstyper undersökas närmare.

4.3.1 Mobiltelefon

De första mobiltelefonerna var just det - mobila telefoner, inget annat. Framförallt sedan lanseringen av Apples första iPhone, och sedermera även Android-telefoner har begreppet mobiltelefon kommit att beteckna en enhet med flertalet olika sensorer och funktionaliteter. Förutom förmågan att ringa och ta emot samtal och data via mobila telefon- och datanät har dagens mobiltelefoner ofta följande funktionaliteter:

Högupplöst men liten skärm. Även om mobiltelefonerna är små i relation till persondatorer har de ofta liknande skärmupplösning. Att skärmen är så liten men högupplöst gör att pixlarna ofta är så små att det mänskliga ögat inte kan urskilja enskilda pixlar.

Wifi. Kompletterar uppkopplingen till mobildatanätet med sin högre överföringshastighet. Mobilabonnemang har dessutom ofta en gräns på datatrafiken via mobilnätet, vilket är en ännu anledning till att föredra Wifi-anslutning före mobildata. Wifi kan användas för positionsbestämning, men kräver då att det finns ett antal trådlösa nät inom räckhåll, och att deras position är känd.

GPS-mottagare. Används för positionsbestämning. Wifi och GPS-mottagaren kompletterar varandra. GPS-mottagaren fungerar bäst utomhus.

Kamera. Mobiltelefoner har ofta två kameror, en mer högupplöst på baksidan för foton och filmer, och en med lägre upplösning på framsidan för självporträtt och videosamtal.

Accelerometer, kompass och gyroskop. Dessa används för att bestämma enhetens orientering samt känna rörelser och yttre kraftpåverkan.

Ljussensor och närhetssensor. Används för att känna av ljusförhållanden och göra avståndsbedömningar. Avståndsbedömningar är viktiga exempelvis för att kunna stänga av touch-skärmen då användaren lägger telefonen mot örat.

Touch. Att styra och navigera mobiltelefoner med touch-input har blivit de facto-standard och endast ett fåtal knapptelefoner finns numera i butikerna.

Fingeravtrycksläsare. Många stora tillverkare erbjuder numera mobiltelefonmodeller med fingeravtryckläsare. Ett stort potentiellt användningsområde för dessa är betalningar med fingeravtryck som autentiseringsmetod.

Sammanfattningsvis kan man dra slutsatsen att dagens mobiltelefoner är mycket mångfacetterade enheter, som ofta kan göra allt som persondatorer tidigare användes för. Generellt kan sägas att surfplattor ofta har samma funktionaliteter som mobiltelefoner, men med några skillnader. De finns både med och utan SIM-kort, och är i det senare fallet beroende av Wifi-anslutning för att kommunicera med omvärlden. De är också större till formatet, vilket gör att många användare föredrar surfplattor framför mobiltelefoner för att konsumera bildmedia såsom film.

4.3.2 Persondatorer

Med persondatorer menas bärbara och stationära datorer. Dessa används vanligen som arbetsstationer eller som underhållningsmaskiner för exempelvis datorspel. De kännetecknas av bland annat:

Hög prestanda. De har generellt hög prestanda i form av processorkraft och arbetsminne. Dock är prestandaspannet stort i denna kategori av enheter. Bärbara datorer med fokus på mobilitet, så kallade Ultrabooks, är långt ifrån lika kraftfulla som en stationär dator som inte har några begränsningar med avseende på fysiska dimensioner.

Stor bildskärm. Mindre bärbara datorer har en skärm jämförbar med surfplattor, och för stationära datorer är 24” en relativt vanlig skärmstorlek.

Fysiskt tangentbord och mus. Det förekommer att bärbara datorer har touchskärm, men generellt används tangentbord och mus, alternativt musplatta, för input.

Modularitet. Det går i allmänhet ofta att uppgradera en dators minne och hårddisk, med undantag för vissa mobilitetscentrerade bärbara datorer. Stationära datorer kan byggas ihop och byggas ut med extrem flexibilitet av en van användare. Denna modularitet förekommer inte i mobila enheter där alla komponenter är fastlödda på samma kretsskort. I någon mån har det blivit vanligare att bärbara datorer också är sammanlödda för att kunna byggas mer kompakta.

4.3.3 Kroppsnära teknik

Detta är en relativt ny kategori jämfört med de andra. Den största underkategorin till kroppsnära teknik är smarta klockor, som börjar bli allt vanligare. Det finns andra exempel, såsom Google Glass, som var en glasögonliknande plattform med inbyggd kamera och display.[5] Några karaktärsdrag hos den kroppsnära tekniken är:

Extremt små. Dessa enheter är extremt små till storleken, vilket är en förutsättning för dess existens. Detta medför att de har helt andra begränsningar jämfört med de större kategorierna, men också andra möjligheter.

Beroende av föräldraenhet. Ofta är dessa små enheter beroende av en förälder. De kommunicerar typiskt sinsemellan via Bluetooth, då i synnerhet Bluetooths Low Energy-läge. Det finns kroppsnära enheter med Wifi-funktionalitet, men dessa är få och utvecklingen är fortfarande i ett så pass tidigt stadium att det är osäkert vilken riktning den kommer ta.

Biometri. En utmärkande funktionalitet i den här kategorin är att dessa enheter ofta innehåller biometriska mätinstrument, i synnerhet pulsmätare, men även instrument för att mäta muskelrörelser. Den förstnämnda är vanligt förekommande i smarta klockor,

[5] Google Glass var ett pilotprojekt, som numera är nedlagt. Man är ganska förtegen om vad man tänker göra med de dragna lärdomarna från projektet.

medan den andra förmodligen fortfarande ska ses som experimentell. Denna typ av hårdvara kan få stora konsekvenser för tekniktillämpningar inom områden som idrott, vård och övervakning.

Denna kategori är fortfarande mycket ung, och tar just nu sina första staplande steg på konsumentmarknaden. Ur ett funktionalitetsdelningsperspektiv är denna kategori mycket intressant, då enheterna är begränsade och kan dra stor nytta av att låna in funktionalitet utifrån. Dessa enheter har som sagt dessutom unika funktionaliteter att erbjuda i utbyte. Med bättre hårdvara och längre batteritid finns möjligheten att dessa enheter i framtiden ses som "huvud-enhet", eftersom deras närhet kan göra dem mer lättillgängliga än mobiltelefoner.

4.3.4 Hemautomation

Något bör nämnas om hemautomation. Det finns redan produkter som Nests smarta termostater och Philips nätverksanslutna HUE-lampor. Det finns flera nätverksprotokoll som är tänkta att användas för hemautomation, så kallade Wireless Home Automation Networks (WHANs). ZigBee är ett mer generellt protokoll, och förutom detta finns en uppsjö specialprotokoll som utvecklats av sina respektive organisationer för deras specifika produktserier (Gomez & Paradells 2010).

Sensorer. Olika sorters sensorer spelar en central roll inom hemautomation. Rörelsesensorer, rökdetektorer och termostater är några exempel.

Aktuatorer. Sensorerna aktiverar aktuatorer som på mekanisk väg påverkar någonting. Aktuatorer kan exempelvis fälla ner persienner eller öppna garageporten.

Centralenhet. Många tillverkare av hemautomationssystem har någon form av centralenhet i sitt produktutbud. Dessa kopplar samman de olika sensorerna och aktuatorerna och gör dem tillgängliga för användaren, ofta i en mobilapplikation.

4.3.5 Hybrider

Det finns även en mycket flytande produktkategori som än så länge inte fått något stort genombrott. Denna kategori betecknas här "hybrider" och innefattar enhetstyper som inte självklart hör hemma i någon annan kategori. Här hittar vi till exempel Asus Transformer (Asus u.å.a.), som kan beskrivas som en blandning av bärbar dator och surfplatta. Huvuddelen av komponenterna sitter i skärmdelen, vilken går att koppla loss från underdelen där tangentbordet och extra batteri sitter. Vid bortkoppling fungerar enheten som en surfplatta.
Den andra varianten bygger på en centralenhet till vilken man kopplar diverse kringutrustning. Neptune Suite (Neptune Suite 2015) är ett projekt som försöker åstadkomma just detta. Centralenheten bärs kring handleden, och till denna kan olika stora skärmar eller tangentbord kopplas. En annan variant är PadFone från Asus (u.å.b.). Centralenheten i detta fall är en mobiltelefon som kan skjutas in i en öppning i en tillhörande surfplatta, som i sig inte är mycket mer än en skärm och extra batteri.

Centralenheten växlar därmed från att använda sin egen skärm till att använda surfplattans.

4.3.6 Övrigt

Det finns ännu fler produktkategorier, men dessa är svåra att placera i ett fack. Det kan ändå vara av nytta att nämna dem. TV-apparater blir allt smartare. Från att ha varit helt hårdvarubaserade, har de nu kommit att köra mer och mer avancerade operativsystem och har därmed en betydande mjukvarukomponent. Detta i kombination med möjlighet till nätverksanslutning gör att eventuella problem och buggar kan åtgärdas av tillverkaren efter att användaren installerat apparaten i hemmet. Det medför också möjligheten till kommunikation med andra enheter. Även vitvaror har idag betydande beräkningskraft, och en framtid där kylskåp själva beställer hem varor som håller på att ta slut är kanske inte så avlägsen. Spisar av nyare modell varnar om maten kokar över, och tvätt- och diskmaskiner har många olika program anpassade för olika situationer.

Bilar har under många år haft en port- och protokollstandard för diagnosticering och interaktion med bilens datorer och styrsystem. Detta gränssnitt är dock främst tänkt att användas av verkstadspersonal. Vad som håller på att hända är att bilar dessutom börjar få system som föraren eller passagerare direkt kan nyttja, så som underhållningssystem och navigationssystem, eller hjälpteknologi för att parkera och backa.

Ovanstående redogörelse för olika enhetskategorier ska inte ses som komplett, men en fullständig kartläggning är inte heller målet med denna rapport. Detta avsnitt har syftat till att ge en inblick i de olika funktionaliteter som erbjuds idag och som kan komma att erbjudas inom en snar framtid. Förhoppningsvis får läsaren en uppfattning om den mångfald ett system för funktionalitetsdelning måste ta hänsyn till.

4.4 Ett ekosystem av funktionaliteter

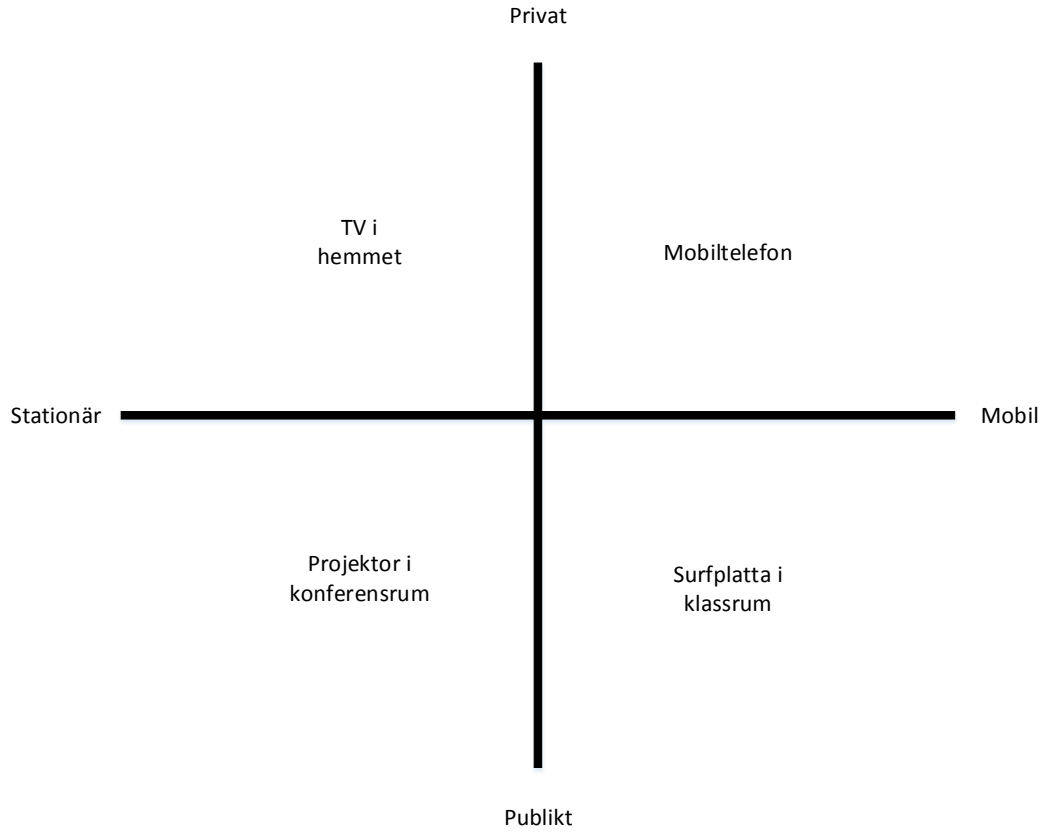
För att realisera ett system där funktionaliteter kan delas mellan olika enheter behöver många bitar falla på plats. Kanske finns det en vinst i att tänka sig systemet som ett ekosystem eller en organisk och flytande enhet, snarare än separata delar som pratar samma språk. Tanken på ett ekosystem poängterar den största vinsten med funktionalitetsdelning; som användare kan man dra nytta av enheternas samlade funktionaliteter. Kanske finns nya sätt att använda dessa tillsammans, som ingen ännu tänkt på. Nedan följer en redogörelse för identifierade delproblem som behöver en lösning för att underlätta funktionalitetsdelning.

4.4.1 Ett gemensamt språk

För att en enhet ska kunna dela med sig av en funktionalitet till en annan enhet behövs ett språk som båda förstår. Detta kan realiseras med hjälp av programvara på respektive enhet. Hur kommunikationen bör ske beror på vilken sorts funktionalitet som ska delas. Att dela en bildskärm kräver stor bandbredd under lång tid, medan andra funktionaliteter, som till exempel en temperaturavläsning, innehåller liten mängd data och har, beroende på sammanhanget, förmodligen inte lika strikta tidskrav.

Det gemensamma språket kan delas upp i två delar. Den första delen avser signalering mellan enheterna. Det används för att signalera när en ny enhet ansluter eller för att fråga vilka funktionaliteter en given användare har tillgång till. Vid implementation är det upp till utvecklaren att definiera olika signaler och deras format. Den andra delen av språket utgörs av själva dataöverföringen. Denna kan vara beroende av vilken sorts kommunikation det är frågan om. Nedan följer en lista över signaler eller meddelanden ett funktionalitetsdelningssystem bör stödja. Observera att fler signaler mycket väl kan behövas, beroende på systemets kontext.

Enhetshantering. För att skicka data mellan enheter i systemet måste de kunna identifieras och pekas ut. Därför krävs någon form av hantering av dessa enheter, inklusive registrering och avregistrering. Det är i detta sammanhang viktigt att skilja mellan privata och delade enheter. Privata enheter har bara en användare och kan direkt associeras med vederbörande. Delade enheter, å andra sidan, kan användas av flera användare och kräver då någon metod för att bestämma vem som för tillfället har kontrollen. Det finns även en viktig skillnad mellan mobila och stationära enheter - de stationära kan även associeras med en plats. Med utgångspunkt i dessa två dimensioner - tillträde och lokalitet - kan man konstruera figur 4.1.



Figur 4.1: Tillträdes- och lokalitetsplanet. Varje kvadrant innehåller en exempelenhet.

Användarhantering. Enhetshanteringen ovan nämner privata enheter. Dessa kan endast användas av en eller flera på förhand bestämda användare. Således måste specifika användare av systemet kunna identifieras och pekas ut, och det måste därmed finnas metoder för registrering och inloggning av användare.

Positionshantering. Om systemet ska ta hänsyn till olika platser måste dessa kunna definieras på något sätt. Olika positioner bör även kunna ha relationer till varandra. Ett exempel på detta skulle kunna vara en byggnad med flera rum. Byggnaden är en position, och varje rum är en underposition. Begreppet position behandlas vidare i avsnitt 4.3.3.

Konfigurationshantering. För att spara tid för användaren finns det en stor potentiell vinning i att låta användaren spara en konfiguration. Med konfiguration menas i detta sammanhang en uppsättning funktionalitetsdelningar. En "kontorskonfiguration" skulle kunna se ut på följande sätt:

- Mobiltelefonens notifikationer delas till datorns notifikationssystem
- En översiktsvy över exempelvis kalender eller agenda delas från datorn till en extra skärm

- Mobiltelefonens samtalsfunktionalitet delas till datorn för smidigare samtalshantering
- Belysningen anpassas

Konfigurationer behandlas vidare i avsnitt 4.3.3.

Förfrågningar. I ett funktionalitetsdelningssystem måste det gå att göra förfrågningar om vilka enheter som finns tillgängliga. Detta får konsekvensen att varje enhet måste kunna rapportera sina egna funktionaliteter till den som frågar. Beroende på hur systemet implementeras kan detta ske på olika sätt. I ett decentraliserat system rapporterar de tillfrågade enheterna sina funktionaliteter direkt till den som frågar. Ett centraliserat system har istället en komponent, här kallad enhetshanterare, som håller reda på vilken enhet som har vilka funktionaliteter. I en centraliserad kontext finns möjligheten att endast låta enheter rapportera sitt modellnummer och överlåta åt enhetshanteraren att ta reda på vilka funktionaliteter enheten stöder. Detta skulle kunna ske i en för enhetshanteraren intern databas eller mot en extern tjänst. Sedan är det upp till enhetshanteraren att rapportera vilka enheter som finns tillgängliga till den som frågar.

Fördelen med decentraliserad modell skulle kunna vara att systemet blir av med en potentiell flaskhals. Enhetshanteraren är en så kallad “single point of failure” - om den slutar fungera blir hela systemet obrukbart. Fördelen med ett centraliserat system är att enhetshanteraren bör kunna svara snabbare på förfrågningar eftersom den på förhand vet vad som finns tillgängligt. Ett decentraliserat system måste vänta på att enheter rapporterar in, men har svårt att avgöra när alla har svarat. Det kan också vara så att olika enheter med olika prestanda svarar olika snabbt, och att detta påverkar användarupplevelsen.

Dataöverföring. Under tiden då en funktionalitet delas sker dataöverföring från en enhet till en annan. Om en mobiltelefon lånar ut sin accelerometer till en dator går kommunikationen från mobiltelefonen till datorn. Ett vanligt användningsområde för accelerometrar är som input till mobilspel. Om denna funktionalitet ska delas för att styra en applikation på en annan plattform är det viktigt att hålla responstiden så kort som möjligt för att undvika att funktionalitetsdelningen upplevs som seg och långsam.
Om en mobiltelefon lånar en TV:s större skärm går kommunikationen istället från lånaren - mobiltelefonen - till TV:n. Om syftet med delningen enbart är att visa en filmsekvens är kraven på responstid inte lika höga - användaren kan förmodligen vänta några extra millisekunder utan större frustration. Däremot är det frågan om mycket större datamängder.
För att tillmötesgå dessa vitt skilda krav skulle en lösning kunna göra bruk av olika protokoll för olika ändamål. Ett lösningsförslag är att varje funktionalitet explicit specificerar hur dataöverföring går till.

4.4.2 Intelligenta förslag

För att underlätta för nya användare som kanske inte har så klar bild av vad som kan åstadkommas med ett funktionalitetsdelningssystem kan det vara bra att ha något sätt att ge användaren "intelligenta förslag", beroende på vilka funktionaliteter som finns tillgängliga. Om användaren exempelvis har en TV och en surfplatta tillgängliga, och på surfplattan använder en app som har flera vyer, kan appen föreslå för användaren att vederbörande kan välja att visa en vy på TV:n och den andra på surfplattan.

Datorer kan användas för att generera en lista över alla möjliga kombinationer av funktionalitetsdelningar och konfigurationer. En svår nöt att knäcka är hur man programmatiskt kan avgöra vilka av dessa som är av värde för användare, utan att på något sätt blanda in människor. Ett sätt att lösa problemet skulle kunna vara att genomföra ett större systemtest där användare får betygsätta automatiskt genererade delningar och konfigurationer. Risken med detta är att det försämrar användarupplevelsen, så det bör göras varsamt. Ett sätt att begränsa dålig användarupplevelse är att endast de som vill delta i testerna får göra det. Ett annat sätt skulle kunna vara att hålla betygsättningen extremt enkel, förslagsvis att man antingen röstar upp (bra) eller röstar ner (dåligt) den testade delningen eller konfigurationen.[6] På så vis kommer de funktionalitetsdelningar som får flest bra röster "bubbla upp till ytan".

4.4.3 Konfiguration

Om en användare har tillgång till flera enheter och deras funktionaliteter resulterar detta i en uppsjö kombinationer av funktionalitetsdelningar. För att underlätta för användaren kan en uppsättning funktionalitetsdelningar sparas som en konfiguration. Då användaren aktiverar konfigurationen aktiveras alla de funktionalitetsdelningar som ingår i den angivna konfigurationen.

Konfigurationer kännetecknas av det sammanhang användaren befinner sig i, det användaren för tillfället gör eller tar sig för. Ett exempel är en kontorskonfiguration, som nämnts ovan. Ett annat exempel är en användare som befinner sig i ett vardagsrum, där bland annat en TV och högtalare finns. Användaren kan då koppla samman dessa med sin mobiltelefon för att dela mobiltelefonens funktionalitet. Ett scenario där detta kan användas är när det ringer från mobilen under en film, TV:n och högtalarna känner av detta och pausar filmen. Utöver det så lånar TV:n och högtalarna mobiltelefonens funktionalitet att ringa och ta emot samtal, vilket gör att TV:n visar ett grafiskt gränssnitt och att högtalarna agerar som mobiltelefonens högtalare. Användaren behöver således aldrig ta upp sin mobiltelefon. Detta kan vara en typisk konfiguration där användaren har tillgång till TV och högtalare.

Förutom att bestämma vilka enheter och funktionaliteter som delas i en konfiguration, kan även själva innehållet beskrivas. Som tidigare nämnts, kan det väljas utifrån tid eller plats. En konfiguration som består av två skärmar och en mobil, behöver inte alltid

[6] Detta liknar det system som används på webbsidor som exempelvis Reddit och StackExchange för att rösta upp och ner inlägg.

skicka samma data till skärmarna. I en kontorsmiljö visas kanske börsnoteringar på skärmarna under förmiddagen, medan något annat visas under eftermiddagen.

4.4.4 Aktivering

För att minimera manuellt arbete för användaren kan det vara nödvändigt att enhetshanteraren sköter uppkopplingen på egen hand. En användare kan vid den första initieringen ange om den nya enheten ska tillhöra en konfiguration och om funktionalitetsdelning hädanefter ska initieras automatiskt. "Automatiskt" i detta sammanhang betyder att konfigurationen eller profilen aktiveras som svar på en viss händelse. De händelser som identifierats i denna rapport är tid och position.

Tid. Aktivering av en konfiguration kan ske automatiskt vid en specifik tidpunkt. I en centraliserad modell skulle denna aktiveringshändelse kunna hanteras av enhetshanteraren, eftersom den endast är beroende av en korrekt tid.

Position. Denna aktiveringshändelse kräver att systemet har kännedom om var användaren befinner sig, och är därför beroende av data från en enhet nära användaren. Systemet kan med denna kunskap automatiskt växla till en passande konfiguration baserat på användarens position. Ett exempel kan vara att användaren stiger in på sitt kontor. Användarens position manar enhetshanteraren att skifta till en mer passande konfiguration som använder de funktionaliteter som erbjuds på den aktuella platsen. Funktionaliteterna här skulle kunna vara extra skärmar, en stationär dators högre prestanda, den fasta telefonen m m. Den konfiguration som väljs vid automatisk positionsbaserad funktionalitetsdelning kan bestämmas utifrån några olika perspektiv. Tre olika fall har identifieras.

1. Byta konfiguration helt och hållet, exempelvis då användaren kommer till eller lämnar arbetsplatsen och då vill byta aktivitet.
2. Återuppta den konfiguration som senast användes på den aktuella positionen. Liknar första alternativet, men försöker återuppta en tidigare konfiguration.
3. Flytta föregående konfiguration till den nya positionen. Detta skulle kunna inträffa exempelvis då en användare tittar på TV, och bestämmer sig för att gå ut i köket och laga mat. Då användarens position ändras, visas TV-programmet istället på TV:n i köket.

Första och andra fallet är snarlika, med skillnaden att fall två försöker återuppta en tidigare konfiguration. Fall ett använder istället en på förhand definierad konfiguration. För att differentiera de båda fallen ges här ett exempel. Användaren Lina börjar alltid sin arbetsdag med att läsa mail, och vill därför ha en mailkonfiguration då hon anländer till arbetsplatsen. Efter lunch ägnar hon sig åt utvecklingsarbete och vill då ha en konfiguration anpassad för detta. Lina följer alltid detta mönster - först mail, sedan utveckling - och använder då lämpligen den första metoden för att välja konfiguration då hon anländer på morgonen. Lars, å andra sidan, mailar och utvecklar om vartannat och har ingen specifik tid på dagen avsatt för respektive uppgift. För hans ändamål passar konfigurationsval enligt metod två bättre.

Ett problem som uppstår i och med införandet av positionsbaserade konfigurationsval är att ta reda på vilken metod som ska användas. Det går att generalisera problemet, men specialfall förekommer alltid. I de flesta fall vill Lina i exemplet ovan börja dagen med mailkorrespondens, men inför en viktig deadline kanske hon hellre vill hoppa direkt på utvecklingen. Ett sätt att lösa detta skulle kunna vara att systemet gissar, och användaren helt enkelt byter konfiguration om systemet gissar fel. Ett annat sätt skulle kunna vara att användaren bekräftar den föreslagna konfigurationen innan den faktiskt aktiveras.

4.5 Prototyp för funktionalitetsdelning

Innan en prototyp kan utvecklas, behövs det en beskrivning på hur detta system kan se ut. I följande avsnitt läggs därför ett förslag på en arkitektur fram. Förslaget är baserat på user story A.2 (se appendix A), och ska därför inte ses som redo för en produktionsmiljö, utan snarare som en prototyp. Det är dock ett viktigt första steg mot en framtid där flera system arbetar tillsammans. Avsnittet avslutas med en beskrivning av de tester som planerats för prototypen.

4.5.1 Server

En server vars huvudsyfte är att vidarebefordra och administrera enheters kommunikation och funktionalitet. Servern agerar som en tredje part när två enheter vill prata med varandra, i detta fall är det omodifierad data från en mobiltelefons gyroskop som skickas mellan enheterna och servern. Enhetshanteraren exponerar två REST-ändpunkter som används av klientkomponenterna.[7] Dessa ändpunkter är följande:

Registrering. Används för att registrera en ny användare. Då användaren registreras genereras en unik identifierare för den enhet från vilken registreringen görs. Användaren associeras med enheten och kommer därefter ha tillgång från enhetens funktionalitet även från andra enheter.

Inloggning. Används för att logga in en redan registrerad användare. Om enheten som användaren loggar in från inte är associerad med vederbörande, genomförs en association.

Registrering och inloggning är nödvändigt eftersom systemet måste kunna avgöra vem som har tillgång till vad. Prototypens användarhantering är förhållandevis enkel, eftersom det är ett problem som redan har många lösningar, och dessutom inte huvudsyftet med denna rapport.

Enhetshanteraren har dessutom en annan kommunikationskanal med högre prestanda. Denna förbindelse upprättas vid inloggning och är tänkt att användas dels för funktionalitetsförfrågningar, och dels själva dataöverföringen under pågående funktionalitetsdelning.

[7] En arkitektur för webb-API:er som använder HTTP-protokollets verb (GET, POST etc). Se http://en.wikipedia.org/wiki/Representational_state_transfer för mer information.

Funktionalitetsförfrågning. För att initiera en funktionalitetsdelning måste man först veta vad som finns tillgängligt. Frågan ställs via högprestandakanalen, eftersom det här är lätt att avgöra vilka enheter som är uppkopplade. Om användaren har någon enhet med den efterfrågade funktionaliteten kommer enhetens unika identifierare returneras. Om frågan får flera träffar returneras flera identifierare, och användaren får välja en av dem.

Funktionalitetsdelning. Denna funktion är huvudsyftet med systemet. Prototypen delar sina gyroskopdata, vilken utgörs av tre reella tal per sampling. Tekniken som valts - Socket.IO - klarar av att skicka binärdata som exempelvis video med bra prestanda, vilket är en av anledningarna till att tekniken valts.

4.5.2 Klienter

Enheter som lånar eller delar sin funktionalitet klassas som klienter. Dessa använder servern för att kommunicera med varandra och för autentisering. I prototypen används en Android-applikation och en Chrome-extension som klienter. I en fullfjädrad miljö hade i princip alla möjliga sorters enheter kunna vara klienter, så länge de implementerar de grundläggande funktionerna. Det som behöver stödjas på de flesta plattformar är inloggning och registrering. Generellt bör ett registreringsalternativ finnas med där en inloggning krävs. Utöver detta behövs meddelande- och dataöverföring på samtliga plattformar, då en enhet antingen behöver skicka eller ta emot data. En klient behöver nödvändigtvis inte stödja både in- och utlåning, utan kan mycket väl endast stödja en av dessa.

Android. En klientapplikation i Android, skriven i Java ska användas för att skicka data. Denna data skickas till en server som vidarebefordrar den till slutmålet, en Chrome-extension. Applikationen ska kräva att användaren gör en inloggning mot en server för autentisering. Om användaren inte har ett konto kan ett sådant skapas direkt i applikationen. I prototypen är applikationens huvudsyfte att skicka gyroskopdata som ska användas för att styra en Street View i Google Maps.

Chrome-extension. För de data som skickas från Android-applikationen behövs en klient som ska ta emot den. I prototypen har en Chrome-extension valts ut som datamottagare då det är en dators webbläsare som är slutmålet. Likt Android-applikationen behöver användaren autentisera sig även i detta fall, så inloggning och registrering är därför en grundläggande del av Chrome-extensionen. Denna komponent ska lyssna efter gyroskopdata, och när sådan påträffas ska en Google Maps Street View uppdateras baserat på gyroskopdata.

Enhetshanterare. En enhet som till exempel en mobiltelefon kan agera som en enhetshanterare för användaren. Där kan användaren lista tillgängliga enheter, justera inställningar och administrera andra enheter. I listan går det även att skanna efter enheter och funktionalitet som finns i närheten. Prototypen implementerar delar av enhetshanteringen.

4.5.3 Användartester

I detta avsnitt presenteras de tester som är tänkta att genomföras då prototypen når ett första körbart stadium. Testresultaten används sedan för att förbättra prototypen. För mer detaljer kring testernas utformning, se appendix B.

Registrering i mobilapplikation. Detta test kommer att utföras i en mobilapplikation. Poängen är att användaren skall kunna registrera ett nytt användarkonto som skall användas för inloggning.

Applikationen kommer att vara installerad och startad i förväg. Skärmen som visas är inloggningsskärmen och inte registreringsskärmen. Målet är att användaren själv ska nå registreringsskärmen och skapa ett nytt konto. Det som kontrolleras här är om användaren uppfattat en genomförd registrering eller om något gått fel. Vid inmatningsfel får användaren felmeddelanden som beskriver felet och hur det åtgärdas. Instruktioner från testledaren kommer inte att ges så länge användaren själv kommer vidare.

Registrering i Chrome-extension. Testet kommer att vara likt föregående test men skall utföras i en Chrome-extension. Poängen är att användaren skall kunna registrera ett nytt användarkonto som skall användas för inloggning.

Chrome-extensionen kommer att vara förinstallerad och aktiverad. Det som visas är en liten ikon bredvid adressfältet. Användaren kommer att få instruktioner att registrera sig via ikonen. Målet är att användaren själv ska nå registreringsskärmen och skapa ett nytt konto. Det som kontrolleras här är om användaren uppfattat en genomförd registrering eller om något gått fel. Vid inmatningsfel får användaren felmeddelanden som beskriver felet och hur det åtgärdas. Instruktioner från testledaren kommer inte att ges så länge användaren själv kommer vidare.

Inloggning. Detta test kommer att bestå av två komponenter eftersom en inloggning krävs på minst två olika enheter samtidigt.

Första steget är inloggning via mobiltelefonens applikation. Användarens upplevelse står i centrum och vederbörande uppmanas reflektera över om något känns krångligt eller kan missuppfattas. Applikationen kommer att vara startad i förväg eftersom det är inloggningen som är i fokus. Målet är att användaren ska känna sig bekväm med inloggning och inte ha några funderingar eller känna att något är konstigt.

Andra steget är inloggning i webbläsaren via extension-gränssnittet. Användarens upplevelse står i centrum och vederbörande uppmanas reflektera över om något känns krångligt eller kan missuppfattas. Webbläsaren kommer vara startad i förväg, och extension-fönstret kommer vara öppnad eftersom det är inloggningen som är i fokus. Målet är att användaren ska känna sig bekväm med inloggning och inte ha några funderingar eller känna att något är konstigt.

I båda fallen kontrolleras det om användaren uppfattat en framgångsrik inloggning eller om någonting gått snett. Om det senare fallet sker, undersöks det om användaren kan göra en ny inloggning med endast hjälp av felmeddelanden som visas.

Gyrostyrning av Street View. I detta test skall användaren få pröva på att använda mobiltelefonen som en kontroll i Google Street View. Tanken är att de meddelanden som visas ska uppmana användaren att själv ta de steg som krävs för att kontrollera Street View med hjälp av mobiltelefonen.

Användaren kommer att behöva vara registrerad och inloggad i både en mobiltelefon och i Chrome-extension. Den information och instruktion användaren får är att besöka en hemsida som har en anpassad version av Street View. Användaren kommer få ett meddelande från Chrome-extensionen att en annan enhet är tillgänglig för användning på sidan, och uppmanas att initiera en funktionalitetsdelning.

Om användaren accepterar funktionalitetsdelningen på mobiltelefonen startas funktionalitetsdelningen, och användaren kan då styra webbsidans Street View genom att vrida mobiltelefonen.

5 Resultat

Vid examensarbetets början bestämdes det att prototypen skulle baseras på en user story. Under del 1 genomfördes ett flertal brainstormingmöten för att generera user stories. Inför del 2 valdes en user story ut, som grund för prototypen, då alla user stories inte skulle kunna uppfyllas inom tidsramen. Genom att implementera de icke-triviala delarna och visa att det fungerar kan man argumentera för att konceptet som helhet är värt att vidareutveckla (avsnitt 3.2).

Konceptet grundas på user story A.2 (appendix A).

> *"As a Google Maps user, I want to be able to use my mobile device's gyroscope as input, so I can control the street view orientation by moving my device."*

Denna user story valdes under ett möte med handledare och creative director på Tactel efter att ett hinder stötts på för user story A.1 (appendix A). Hindret utgjordes av en begränsning i webbläsarens JavaScript-API, som inte tillåter överskuggning av en specifik funktion p g a det säkerhetshål det skulle medföra.

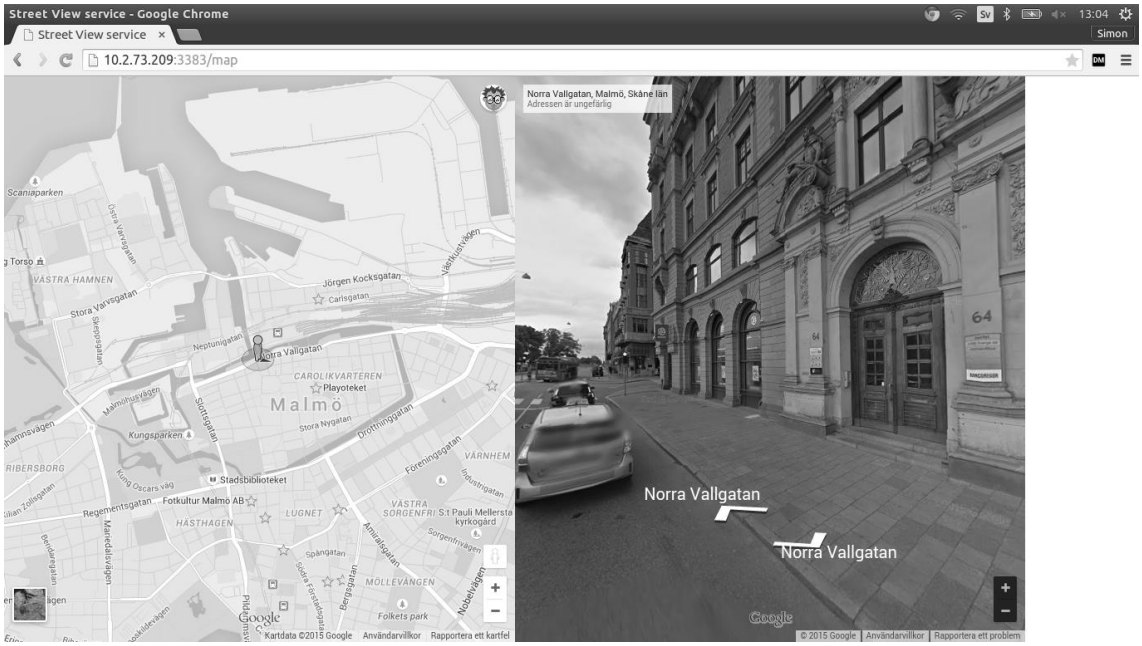
I grund och botten uppfyller user story A.2 samma förutsättningar som A.1 (avsnitt 3.1.2), vilket motiverade valet att basera prototypen på A.2 istället. Andra user stories som genererats återfinns i appendix A.

Kortfattat, ska användaren i realtid kunna använda sin mobiltelefon som en kontroll i Google Map Street View, med hjälp av mobiltelefonens sensorer. Scenariot ställer höga krav på responsivitet. Om användarens rörelser inte visas på kartan tillräckligt snabbt kommer systemet upplevas som segt och långsamt. Då responsivitet är viktigt ur ett användbarhetsperspektiv är det även ett lämpligt scenario för en prototyp.

I detta kapitel presenteras prototypen som utvecklats som ett proof-of-concept för funktionalitetsdelning. Kapitlet presenterar de tre komponenter som tagits fram. För varje komponent ges en beskrivning tillsammans med skärmdumpar. De val som gjorts motiveras om så behövs, med avseende på tekniker och implementationsdetaljer. Mer information angående tekniker kan hittas i teknisk bakgrund (kap. 2).

5.1 Gyrostyrning av Street View

I Google Maps finner man en funktion som tillåter användaren att observera miljöer direkt från gatan, precis som om man själv var där (figur 5.1). Dessa bilder är tagna med en sfärisk kamera och överlappar varandra. Detta ger en illusion av att gatuvyn är tredimensionell och gör det möjligt för användaren att ändra riktning. I dagsläget kan denna vy kontrolleras med hjälp av mus och tangentbord. Som ett proof-of-concept har en mobiltelefon valts ut för att ersätta musen och tangentbordet med en förhoppnings mer intuitiv användarupplevelse. I mobiltelefonen finner man flera sensorer, som nämnts i kapitel 4.3.1, och en naturlig sensor för att styra orienteringen i gatuvyn är gyroskopet. Detta får effekten att användaren kan vända sig och titta upp och ner genom att vrida och vända på mobiltelefonen.

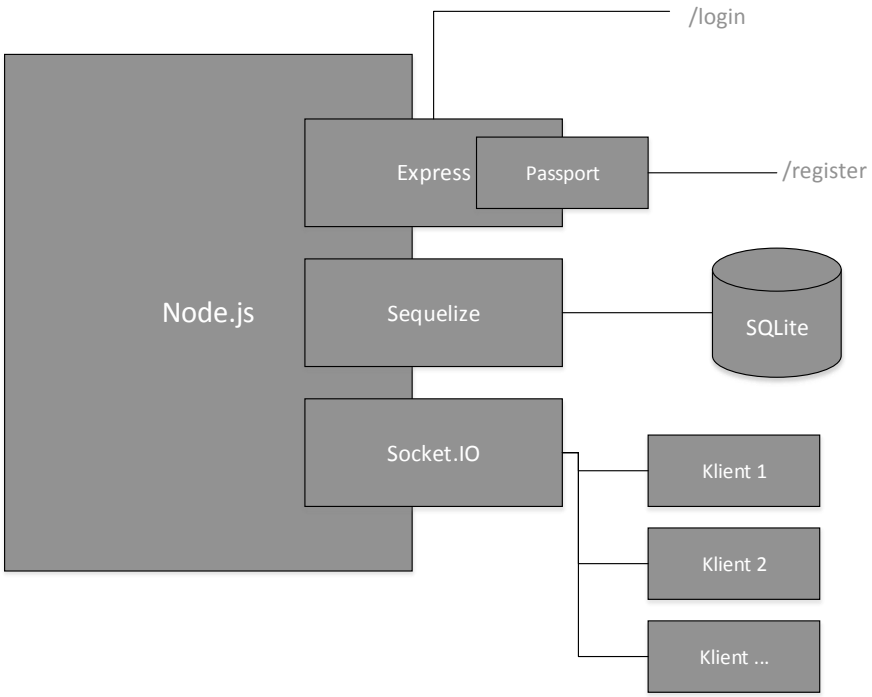


Figur 5.1: Figuren visar den Google Maps Street View som användes för prototypen. Till vänster visas en karta med den aktuella positionen. Till höger visas gatuvyn för den aktuella positionen. I gatuvyn kan användaren byta riktning för att se sig om kring.

För att bygga sidan har Googles Maps-API använts. Anledningen till detta val var att det helt enkelt är den mest mogna tjänsten i sitt slag. Ett önskemål hade varit att använda Googles Maps-tjänst direkt, men denna exponerar inte de variabler och funktioner som är nödvändiga. Istället har alltså Googles Maps-API använts för att lägga in kartan och gatuvyn på en egen webbsida.

5.2 Server

Den server som fungerar som spindel i nätet implementerades i Node.js. Den använder ramverken Express, Sequelize, Passport samt Socket.IO (figur 5.2). Express används för att enkelt skapa REST-ändpunkter, och Passport används för att lägga ett autentiseringslager på dem. En klient anropar ändpunkterna via HTTP-requests. Sequelize används som ett gränssnitt mot databasen, och Socket.IO används för direktkommunikationen mellan de enheter som pratar med varandra via servern.



Figur 5.2: Figuren beskriver hur serverns olika ramverk sitter ihop, och deras syften.

Att Node.js valdes som serverplattform hade mestadels att göra med att det är en populär serverplattform med många ramverk och paket redo att användas. En annan fördel är att Node.js har bra rykte när det gäller prestanda och minnesanvändning, vilket kan komma till nytta när det är mycket trafik som passerar. En tredje fördel med att använda Node.js var antalet programmeringsspråk som behövdes för prototypen då skulle hållas nere eftersom Chrome-extensionen också använder JavaScript.

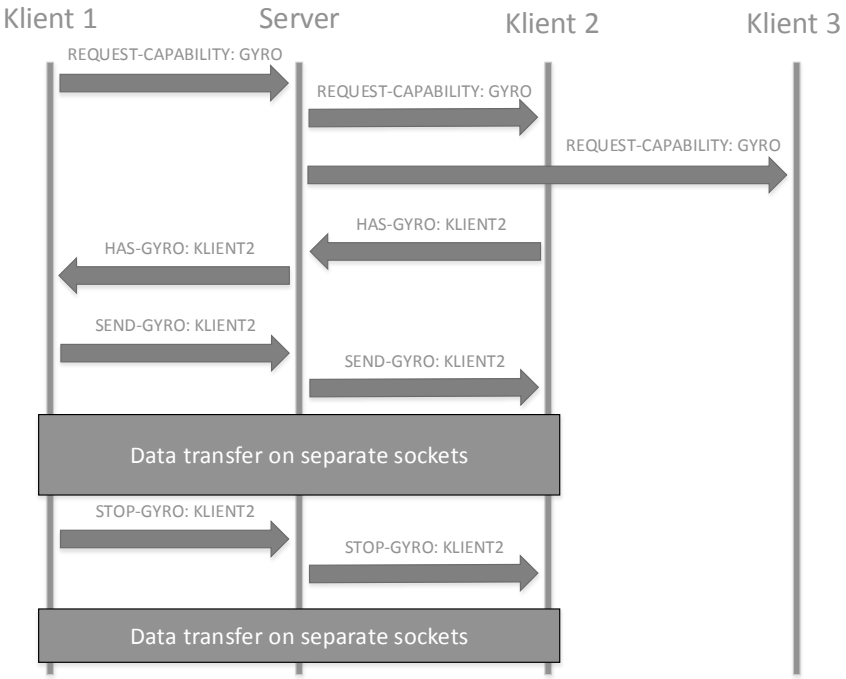
REST-API:n är ett mycket vanligt sätt att bygga webb-API:er, och därför valdes Express. Det finns andra liknande ramverk, exempelvis Koa och Hapi, men en fördel med Express är att det fungerar mycket bra ihop med Passport. De ändpunkter som implementerats är "/register" och "/login". De används genom att klientapplikationen skickar en POST-förfrågan till den aktuella ändpunkten via HTTP-protokollet. Användaren skickar i förfrågan med användarnamnet och lösenordet. Om förfrågan går till "/register" registreras den nya användaren, och om den går till "/login" görs en kontroll att användarnamnet och lösenordet är korrekta. Om förfrågan lyckas, returneras HTTP-status 200: OK. Ändpunkten "/login" är skyddad med hjälp av ramverket Passport. Detta får effekten att POST-förfrågan endast går igenom om autentiseringen

lyckas, som beskrivet ovan. Observera att en enkel form av autentisering används för prototypen, och skulle vid en vidareutveckling behöva förstärkas, antingen genom att autentiseringen sker över HTTPS/TLS, eller att en annan autentiseringsmetod används. Genom att ramverket Passport använts, är det förhållandevis enkelt att byta ut autentiseringsmetoden mot en säkrare variant.

Som gränssnitt mot databasen används ramverket Sequelize. Den stora vinsten med detta är att man inte behöver skriva databas-specifika frågor för att läsa och skriva till databasen (även om det är möjligt), utan man kan istället göra det genom att anropa funktioner i ramverket. Följden detta får är att man kan byta ut den underliggande databasen mot något mer passande om behoven förändras. Behoven under prototyputvecklingen var en lösning som krävde minimalt med konfiguration, varför valet föll på SQLite3. I ett annat läge skulle denna enkelt kunna bytas ut mot exempelvis MySQL eller PostgreSQL.

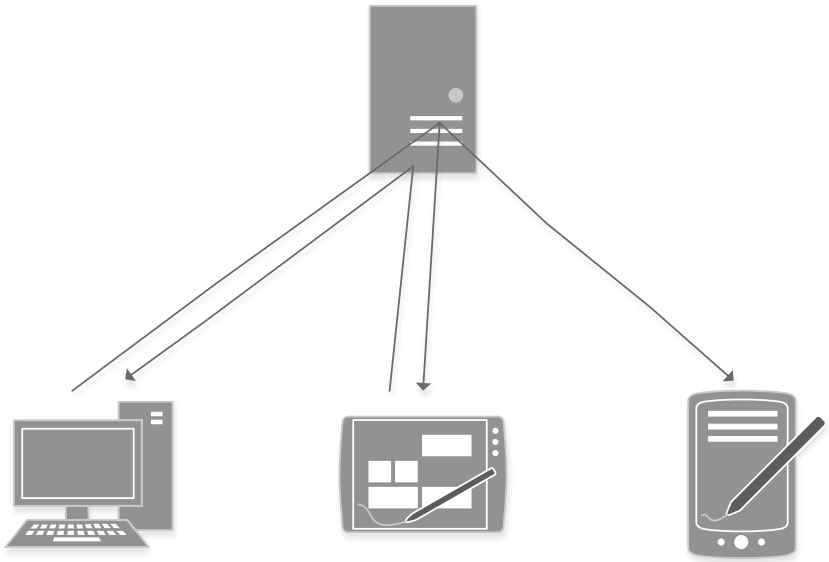
Socket.IO är ramverket som används för att skicka data mellan klienterna (figur 5.4). Socket.IO är en implementation av protokollet Websockets. Socket.IO har tillgång till för denna prototyp användbara egenskaper som exempelvis broadcasting, där meddelanden kan skickas till alla som lyssnar. Så fort en användare loggar in på en enhet skapas en socket mot servern. Denna socket är endast till för kontrollmeddelanden för att exempelvis fråga efter funktionaliteter eller för att begära att dataöverföringsförbindelser upprättas. Denna socket är knuten till det användarkonto som användes för att logga in på enheten. Alla enheter som är inloggade med samma användarnamn delar denna meddelandekanal. Figur 5.3 beskriver hur det kan gå till från att en enhet begär en funktionalitet tills det att funktionalitetsdelningen upprättas.

Först skickar klient 1 ut en förfrågan om funktionaliteten gyro, vilket här betecknar gyroskop. Servern broadcastar denna förfrågan till användarens två andra ansluta enheter. Klient 2 är en enhet med gyro-funktionalitet och svarar därför på begäran. Om flera enheter hade svarat hade klient 1 fått välja ut en av dem. När en är vald skickas en begäran att börja skicka gyro-data. En ny socket-förbindelse upprättas, vilken endast är avsedd för denna funktionalitetsdelning. Delningen fortlöper tills en av klienterna begär att den avslutas, varpå de sockets som användes för dataöverföringen stängs.



Figur 5.3: Figuren beskriver hur upprättning av en funktionalitetsdelning kan se ut.

Socket.IO har visat sig klara av den strida ström av meddelanden delning av gyrodata innebär. Under analysen undersöktes även biblioteket WebRTC. Det kan finnas en potentiell prestandavinst i att använda det istället, eftersom WebRTC skickar data direkt till mottagaren, till skillnad från Socket.IO, där all trafik går via server. En annan fördel är att WebRTC endast kräver en server för att upprätta förbindelsen. Efter detta sker kommunikationen direkt mellan parterna. Dock har det framkommit kritik mot WebRTC - det är möjligt att avslöja parternas riktiga IP-adresser, även om de använder en VPN-tunnel. Av denna anledning valdes slutligen Socket.IO.



Figur 5.4: Kommunikation mellan enheter som använder Socket.IO.

5.3 Klienter

5.3.1 Android-applikation

I avsnitt 4.5 nämns det att en komponent skall vara en mobiltelefon. Som val av plattform för detta har operativsystemet Android valts p g a av dess öppenhet. För att komma igång behövs endast Android Studio. En av klienterna i prototypen är således en Android-applikation, skriven i Java. Applikationens huvudsyfte är att skicka gyroskopdata till servern för att kunna styra en Street View.

För att kommunicera med servern används, precis som i servern, ramverket Socket.IO. Ursprungligen är Socket.IO skrivet i JavaScript vilket inte är kompatibelt med Android. Den version som används i applikationen är en portad version från JavaScript till Java och fyller samma funktion som den man finner i JavaScript.[8] Utöver det används de standardbibliotek som finns i Androids SDK.

Android-applikationen kompileras mot Android-API level 21 som är den näst senaste när rapporten författades. Skälet till att det valts en senare version är att det nyare API:et erbjuder fler funktioner. Dessutom hade utvecklingsenheterna stöd för denna version. Då det var tal om en prototyp, ställdes inga krav på bakåtkompabilitet.

Inloggningsskärm

Första skärmen som visas vid start av applikationen är en inloggningskärm (figur 5.5). Här kan en användare antingen logga in med ett befintligt konto, eller skapa ett nytt genom att trycka “Register Here”. Likt andra inloggningsalternativ, hanteras fel genom att visa felmeddelanden för användaren (figur 5.5), till exempel när användaren saknar förbindelse med servern. Dessa felmeddelanden varierar beroende på vad för sorts fel som fångats upp av applikationen.

De felmeddelanden som kan visas är:

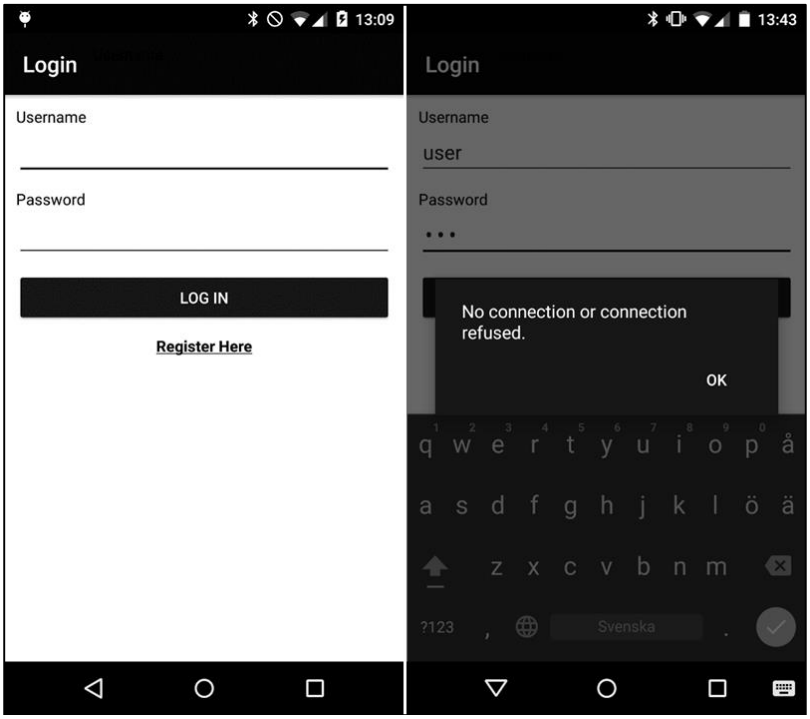
- **"Please enter username/password."**
  Tom inmatning av användarnamn och/eller lösenord.
- **"The username or password you entered is incorrect."**
  Fel inmatning av användarnamn och/eller lösenord.
- **"Connection timed out, check your internet connection."**
  Nätverksfel. Ingen server funnen, servern är nere eller har annan IP-adress.
- **"No connection or connection refused."**
  Nätverksfel. Åtkomst nekad eller ingen internetåtkomst tillgänglig på klienten.
- **"An error has occurred."**
  Generella fel som inte fångats upp av de ovanstående.

[8] https://github.com/nkzawa/socket.IO-client.java

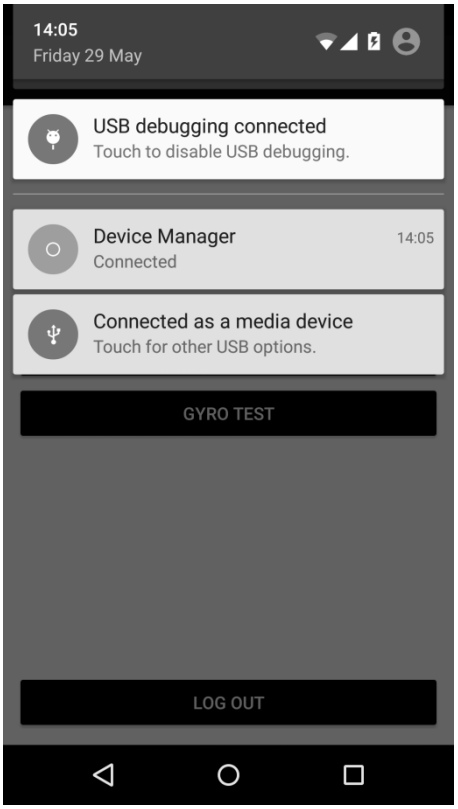
För att ge användaren feedback visas en förloppsindikator under tiden applikationen försöker ansluta sig till servern. Då alla nätverk har olika kapacitet, finns det en timeout som utlöses ifall det tar för lång tid. Användaren får då ett felmeddelande och anslutningsförsöket avbryts.

Om inga fel påträffas skickas användaren vidare till huvudskärmen (figur 5.10) och en bakgrundsprocess sätts igång som upprättar en konstant förbindelse till servern. En notifkation visar förbindelsens status så länge applikationen är igång (figur 5.6). När användaren drar ner notifkationen visas tillgängliga alternativ (figur 5.7). De status som kan visas är:

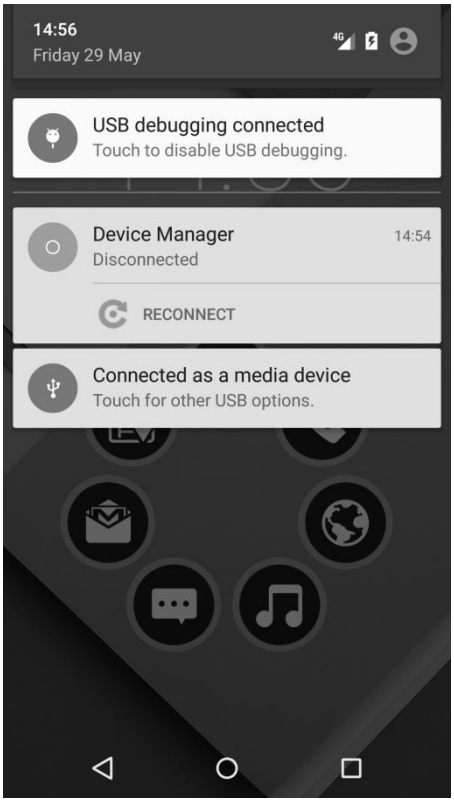
- **"Connected"**
  Applikationen har en aktiv uppkoppling mot servern.
- **“Disconnected”**
  Applikationen är bortkopplad, antingen av användaren eller nätverksfel.
- **“Connecting”**
  Applikationen försöker upprätthålla en anslutning.
- **“Reconnecting”**
  Applikationen försöker upprätta en anslutning efter bortkoppling.



Figur 5.5: Till vänster visas inloggningsskärmen vid uppstart. Till höger visas felmeddelandet som dyker upp vid nätverksfel.



Figur 5.6: Notifikationen som visas efter lyckad inloggning.



Figur 5.7: Notifikationens textmeddelande ändras beroende på nätverksanslutningens status.

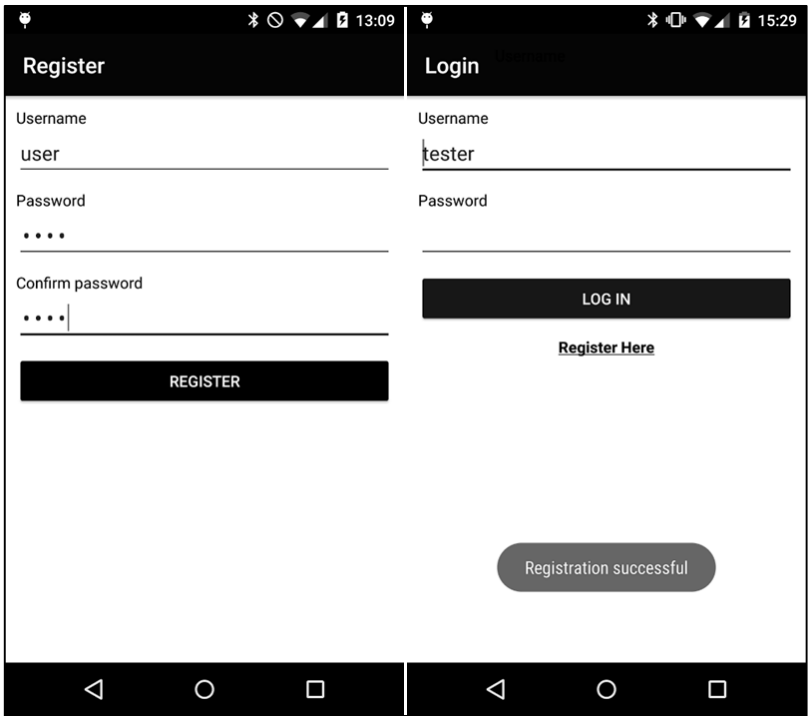
Registreringsskärm
Om användaren bestämmer sig för att registrera ett nytt konto kan det göras genom att trycka “Register Here” på inloggningsskärmen. Användaren skickas då till registreringsskärmen (figur 5.8). På denna skärmen visas tre textfält benämnda “Username”, “Password” och “Confirm password” och en knapp benämnd “Register”. Användaren förväntas fylla i samtliga textfält och trycka på knappen för att skapa ett konto. Användarnamn samt lösenord måste innehålla minst 3 tecken var för att anses vara godkända. Matas det in färre än så, får användaren ett felmeddelande. Förutom detta måste de två sistnämnda fälten matcha varandra. Om användaren misslyckas, visas ett felmeddelande om att lösenorden inte matchar. Figur 5.9 visar ett exempel för hur dessa felmeddelanden kan se ut.

De felmeddelande som kan visas är:

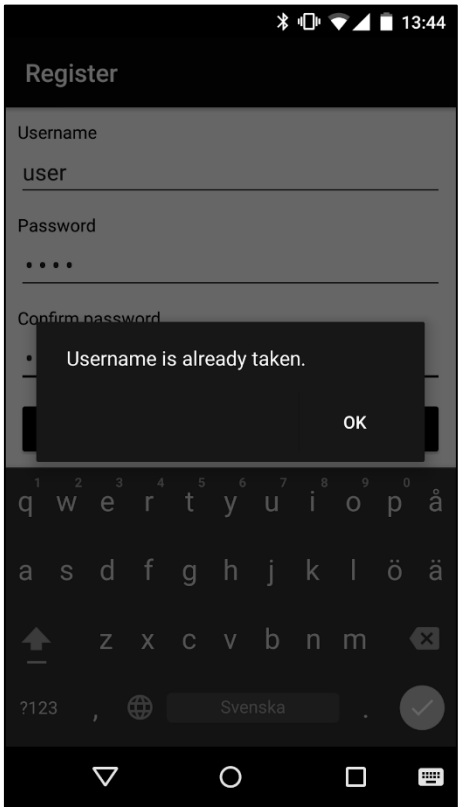
- **"Please enter a valid username. (min. 3 characters)"**
  Angivna användarnamnet är färre än 3 tecken.
- **"Password should be at least 3 characters."**
  Angivna lösenordet är färre än 3 tecken.
- **"Passwords do not match."**
  Lösenordsfältens innehåll matchar inte varandra.
- **"Username is already taken."**
  Angivna användarnamnet finns redan registrerat på servern.
- **"Connection timed out, check your internet connection."**
  Nätverksfel. Ingen server funnen, servern är nere eller har annan IP-adress.
- **"No connection or connection refused."**
  Nätverksfel. Åtkomst nekad eller ingen internetåtkomst tillgänglig på klienten.
- **"An error has occurred."**
  Generella fel som inte fångats upp av de ovanstående.

För att ge användaren feedback visas en förloppsindikator under tiden applikationen försöker ansluta sig till servern. Då alla nätverk har olika kapacitet, finns det en timeout som utlöses ifall det tar för lång tid. Användaren får då ett felmeddelande och anslutningsförsöket avbryts.

Om inga fel påträffas skickas användaren vidare till en inloggningsskärm och notifieras att registreringen lyckats.



Figur 5.8: Till vänster visas registreringsskärmen. Till höger visas notifikationen vid lyckad registrering.



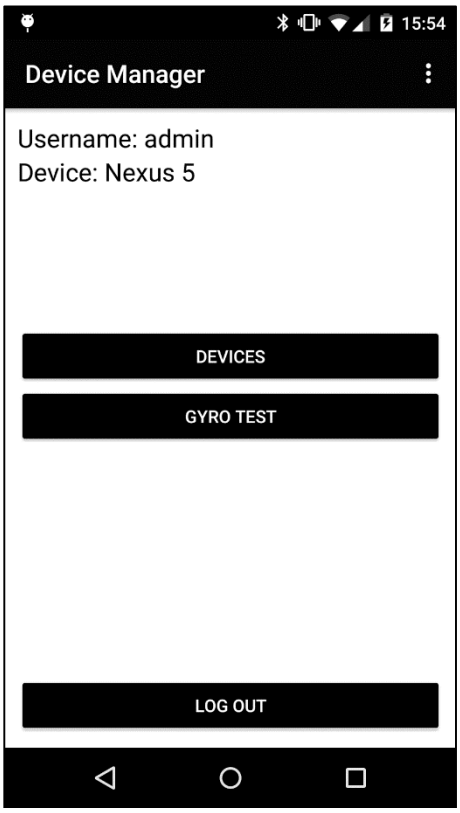
Figur 5.9: Felmeddelande som visas när ett taget användarnamn försöker väljas.

Huvudskärmen

När användaren gjort en lyckad inloggning, visas den här skärmen (figur 5.10). Här finns det ett antal alternativ som användaren kan välja mellan, bland annat två textfält benämnda "Username" och "Device". Första fältet visar användarens användarnamn och det andra fältet visar mobiltelefonens modellnamn. Förutom dessa fält visas även tre knappar benämnda "Devices", "Gyro test" och "Log out". Knappen "Devices" skickar användaren till enhetsskärmen, "Gyro test" till gyroskopskärmen och "Log out" loggar ut användaren och vidarebefordrar denne till inloggningsskärmen.

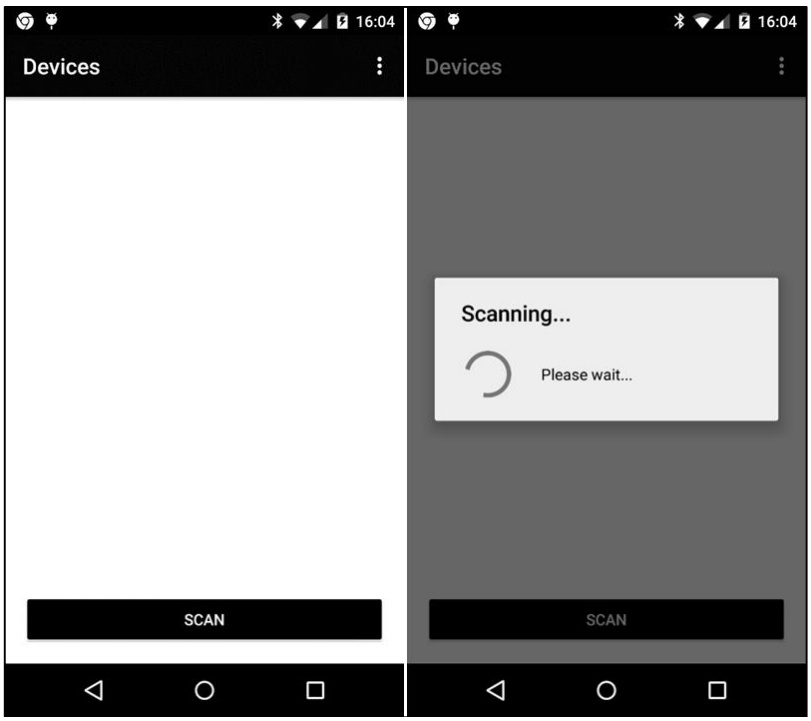
I övre högra hörnet finns en knapp som visar en inställningsmeny. I dagsläget finns endast ett alternativ i menyn - "Settings". Vid knapptryck skickas användaren till inställningsskärmen.

Värt att notera angående huvudskärmen är att den endast är åtkomlig efter lyckad inloggning. Detta gör att andra skärmar som har huvudskärmen som förälder ärver samma egenskap.



Figur 5.10: Huvudskärmen som visas vid lyckad inloggning.

Enhetsskärmen
Då mobiltelefonen kan användas för att administrera andra enheter, finns en skärm för detta ändamål (avsnitt 4.5.2). Vid första ögonkastet kan en enda knapp ses, benämnd "Scan". Vid knapptryck visas en förloppsindikator "Scanning…" och mobilapplikationen ansluter då till servern för att hitta tillgängliga enheter som samma användare loggat in på (figur 5.11). I den tomma ytan listas enheter som hittats asynkront.



Figur 5.11: Enhetsskärmen. Till höger ses den förloppsindikator som visas då användaren tryckt på "Scan".

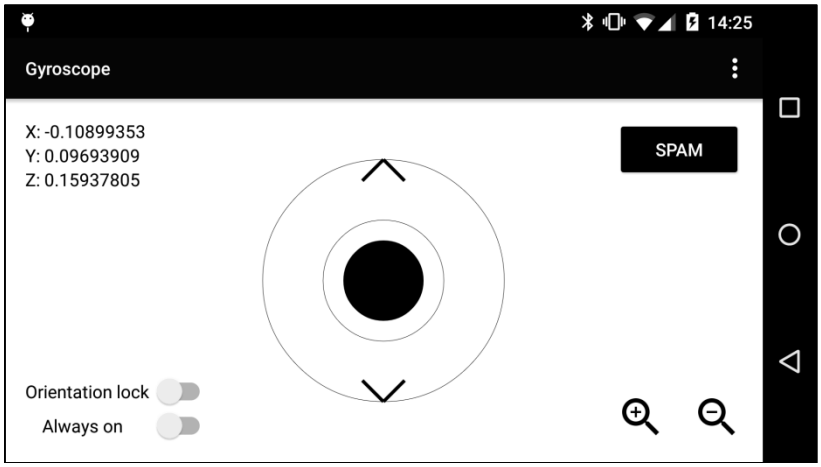
Gyroskopskärmen
Prototypen använder sig av gyroskopdata för att styra Street View. Den här skärmen agerar som en kontroll för att kommunicera med servern och Chrome-extensionen.

Vid uppstart, skapas en socket-anslutning mot servern. Denna anslutning använder samma socket-anslutning som upprättades vid inloggning för att skicka olika kontrollmeddelanden. Skärmens huvudsyfte är att skicka gyroskopdata, vilket sker genom att användaren håller ner joysticken inom cirklarna (figur 5.12, 5.13, 5.14). Joysticken tillåter även användaren att skicka flyttkommandon genom att peka i den yttre cirkeln. Detta kan ske samtidigt som gyroskopet är aktiverat vilket gör att användaren kan se sig omkring och flytta sig i street view utan att lyfta fingret från mobiltelefonens skärm.

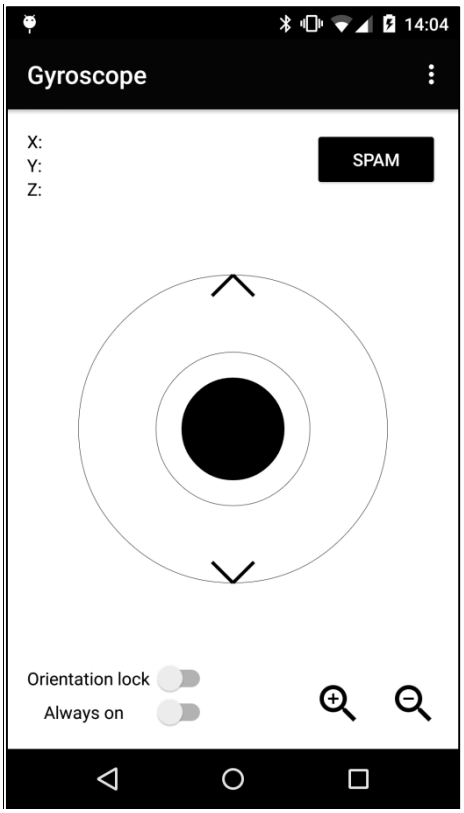
Förutom joysticken finns även två knappar och två switchar. Knapparna låter användaren att zooma in och ut i street view och switcharna låter användaren växla mellan lägen för orientering och ”alltid på”. När orienteringsswitchen är aktiverad låses skärmens orientering, t ex i landskapsläge (figur 5.12). När ”alltid på”-switchen är aktiverad skickas gyroskopdata oavsett om användaren håller nere fingret på joysticken eller ej.

För den här skärmen finns även utvecklarläge som visar extra komponenter. Dessa komponenter är tre textfält som visar gyroskopdata i form av x-, y- och z-led samt en ”spam”-knapp som skickar ett testmeddelande för felsökning (figur 5.12, 5.13). För ett läge med avaktiverad utvecklarläge se figur 5.14.

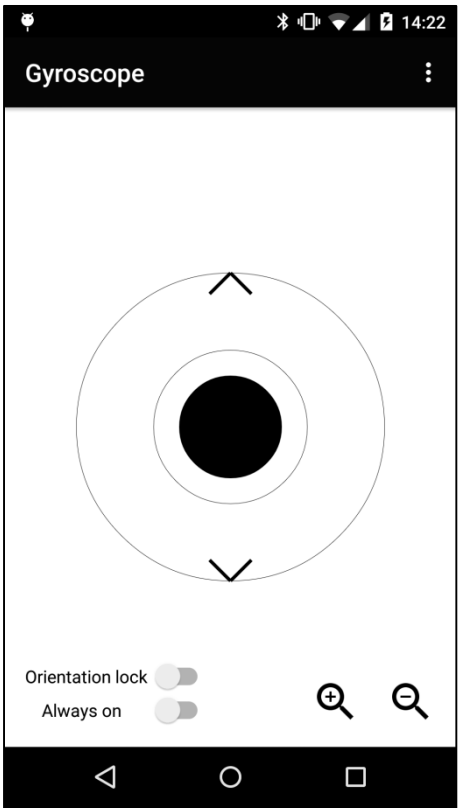
Då kommunikation kan ske från serverns håll, lyssnar applikationen efter inkommande meddelanden. Vid förfrågning av gyroskop från Chrome-extension visas en notifkation som låter användaren starta gyroskopskärmen direkt genom att trycka på notifkationen (figur 5.15).



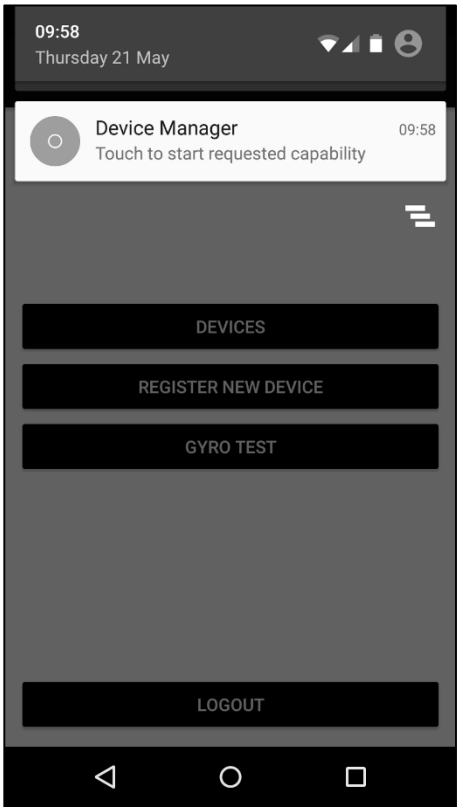
Figur 5.12: Gyroskopskärmen i landskapsläge. Jämfört med B.6.



Figur 5.13: Gyroskopskärmen i porträttläge.



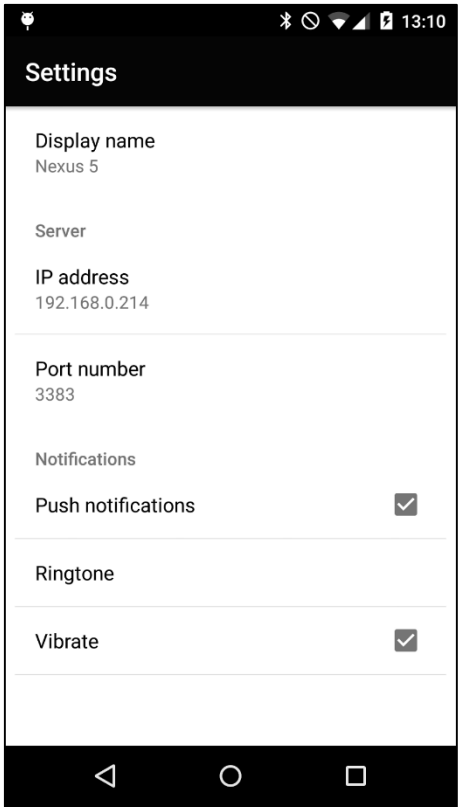
Figur 5.14: Gyroskopskärmen, utvecklarläge avaktiverat.



Figur 5.15: Notifikation som visas när användaren vill starta en styrning i Street View.

Inställningsskärmen
För att inte behöva använda utvecklingsverktyg för att ändra inställningar som är hårdkodade, kan den här skärmen användas. Skärmen har tre fält, benämnda “Display name”, “IP address” och “Port number” (figur 5.16). Första fältet anger vad enheten ska heta, standardinställningen är mobiltelefons modellnamn och är det namn som ska visas för andra enheter. De andra två fälten anger uppgifter som används för att ansluta till en specifik server, nämligen IP-adress och portnummer. Förutom dessa fält, finns även tre alternativ angående hur notifikationer ska hanteras. Dessa bestämmer om notifikationer ska vara aktiverade, vilken ljudsignal som ska användas och om vibration ska vara aktiverad.

De ifyllda inställningarna sparas automatiskt när användaren återvänder till huvudskärmen.



Figur 5.16: Inställningsskärmen.

5.3.2 Chrome-extension

Den klient som ska ta emot gyroskopdata utgörs av en extension till webbläsaren Chrome. Valet föll på Chrome för att den i många undersökningar visat sig vara den mest använda webbläsaren. Chrome har ett väl utbyggt stöd för tillägg - extensioner - som utvecklas med webbteknikerna HTML/CSS/JavaScript. Förutom JavaScripts inbyggda funktioner har man även tillgång till Chrome-API:er. Beroende på vilka API:er som ska användas måste man tala om vilka rättigheter användaren måste godkänna vid installation av extensionen. De rättigheter som krävs av prototypen är:

- **"storage":** Används för att spara data lokalt på den dator som kör Chrome.
- **"background":** Ger extensionen tillåtelse att köra i bakgrunden, i kontrast till den kod som körs direkt på webbsidorna.
- **"tabs":** Ger extensionen tillgång till innehållet i webbläsarens öppna flikar. Används för att kontrollera om det finns någon flik med innehåll som stöder gyro-data.
- **"http://*/*":** Gör att extensionen kan anropa externa tjänster. Används för att kommunicera med servern. I en skarp situation hade denna ersatts av serverns domännamn, men för att underlätta utvecklingen har en mer tillåtande variant valts.

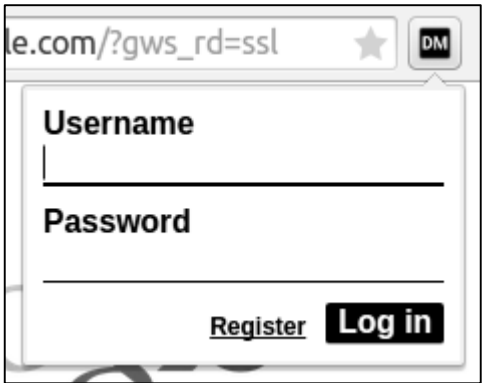
För att implementera den kommunikationskanal över vilken gyrodata och signalmeddelanden går har Socket.IO använts. För mer information, se 5.2.

Browser Action

Efter genomförd installation läggs en knapp, även kallad "browser action", till bredvid webbläsarens adressfält (figur 5.17). Genom att trycka på knappen visas huvudskärmen om användaren är inloggad. Om användaren inte är inloggad visas istället inloggningsskärmen.

Figur 5.17: Figuren visar extensionens knapp inringad.

Inloggningsskärm
Inloggningsskärmen (figur 5.18) gör det möjligt för användaren att logga in i systemet. Användaren fyller i sitt användarnamn och lösenord i respektive fält, klickar därefter på "Log in", och tas därmed vidare till huvudskärmen. Om användaren inte har något konto, kan ett sådant registreras via inloggningsskärmen som nås genom att klicka på "Register". Om något går fel, så som okontaktbar server eller felaktigt användarnamn eller lösenord visas felmeddelandet "Authentication error". Detta meddelande är något sparsmakat, då användaren inte kan avgöra om kontakten med servern är dålig, eller om användarnamnet och lösenordet är felaktigt. Då detta är en prototyp har ändå bedömningen gjorts att det är tillräckligt för detta ändamål.



Figur 5.18: Figuren visar extensionens inloggningsskärm.

Registreringsskärm
Registreringsskärmen har tre fält; "Username", "Password" och "Confirm password". Vid registrering fylls dessa i av användaren, som därefter klickar på knappen "Register". För att avbryta registreringen kan användaren istället klicka på länken "Cancel". Innan registreringsförfrågan skickas till servern görs en kontroll av de ifyllda fälten. Om fälten inte uppfyller kraven får användaren felmeddelanden som beskriver felet (figur 5.19).

De regler som gäller för registreringsfälten är följande:

- Fälten för användarnamn och lösenord måste vara minst 3.
- Fälten "Password" och "Confirm password" måste innehålla samma lösenord.

Om dessa regler följs, skickas registreringsförfrågan. Om användarnamnet är upptaget visas ett felmeddelande. Om användarnamnet är ledigt genomförs registreringen och användaren skickas tillbaka till inloggningsskärmen, nu med användarnamnet förifyllt för att indikera lyckad registrering (figur 5.20).

Figur 5.19: Figuren visar extensionens inloggningsskärm. Här har ett upptaget användarnamn angetts och ett felmeddelande visas.

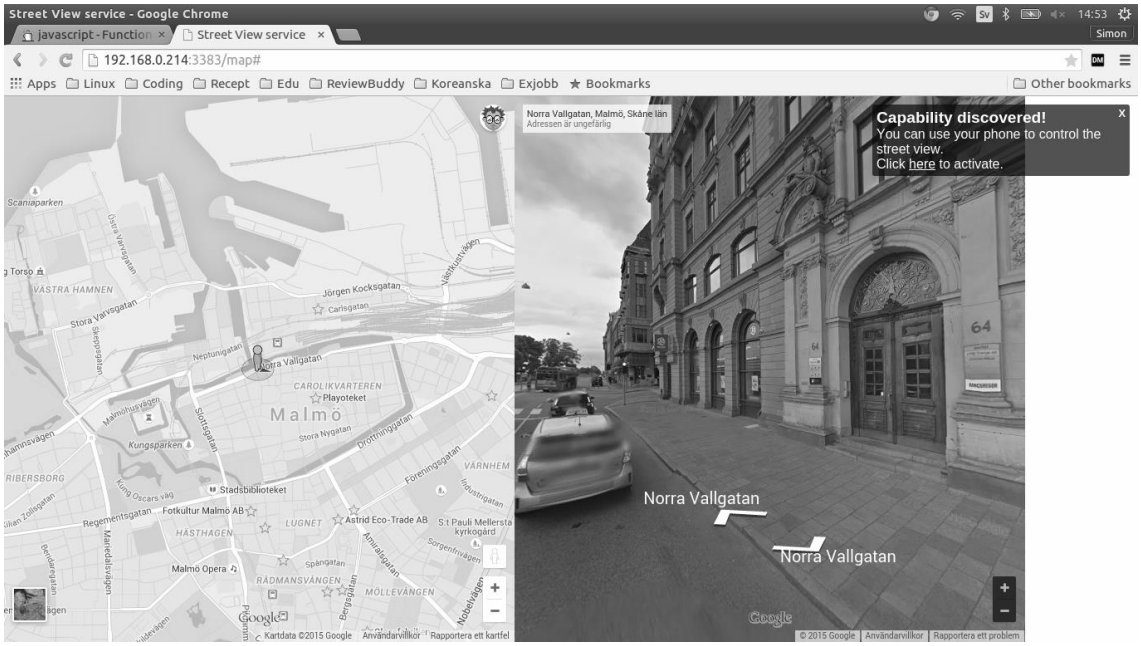
Figur 5.20: Efter lyckad registrering hamnar användaren på inloggningsskärmen igen. Notera meddelandet ”Registration successful”, som tillkom efter användartesterna.

Huvudskärm
Då användaren loggar in och kommer till huvudskärmen upprättas i bakgrunden en förbindelse mot servern med hjälp av Socket.IO. Denna används som meddelandekanal mellan den inloggade användarens registrerade och inloggade enheter. På inloggningsskärmen visas den inloggade användarens namn och den aktuella enhetens namn. Här finns också en utloggningsknapp och en utfällbar inställningsmeny (figur 5.21). Om man öppnar menyn finns en länk märkt "Spam server" som använts under utvecklingen för att logga meddelanden på servern i syfte att underlätta felsökning. Notera att här även finns en länk märkt "Devices". Den är tänkt att användas för att administrera en användares olika enheter, men har inte implementerats i extensionen.

Figur 5.21: Figuren visar extensionens huvudskärm.

Notifikation

Då extensionen upptäcker att en sida med gyroskopstöd besöks, visas en notifikation som talar om för användaren att vederbörande kan aktivera en funktionalitetsdelning (figur 5.22). Då användaren klickar på notifikationens aktiveringslänk skickas en förfrågan att starta funktionalitetsdelningen till användarens mobiltelefon.



Figur 5.22: Figuren visar den notifikation som användaren ser då vederbörande besöker en webbsida som stöder gyroskopdata. Jämfört med figur B.5 som var den ursprungliga notifikationen.

Funktionalitetsdelning

Då funktionalitetsdelningen är aktiverad kan användaren styra gatuvyn genom att röra mobiltelefonen (figur 5.23). Mobiltelefonens gyroskop uppfattar rörelserna, vilka skickas till extensionen som uppdaterar vyn.

Figur 5.23: Figuren visar en aktiv funktionalitetsdelning. Användaren rör på mobiltelefonen varpå gatuvyn ändras beroende på mobiltelefonens orientering.

5.4 Testresultat

I detta avsnitt presenteras resultaten av användartesterna. För mer ingående beskrivning, se appendix B.

B.1 Registrering i mobilapplikation
Flera användare var osäkra på om registreringen hade slutförts ordentligt. För att förbättra tydligheten lades en så kallad "toast"-notifikation till då användaren återvänder till inloggningsskärmen vid lyckad registrering.

B.2 Registrering i Chrome-extension
Flera användare var osäkra på om registreringen hade slutförts ordentligt. För att förbättra tydligheten lades en bekräftelse till på inloggningsskärmen då användaren återvänder dit efter lyckad registrering.

B.3.1 Inloggning i Mobilapplikation
Inga problem hittades.

B.3.1 Inloggning i Mobilapplikation
Inga problem hittades.

B.4 Gyrostyrning av Street View
Notifikationen som visas på webbsidan tog en stund att upptäcka för flera användare, och flyttades därför från nedre högra hörnet till en mer framträdande plats på sidan.
Ordvalet på gyroskopskärmens knapp upplevdes som förvirrande och ändrades därför från "Unpause" till "Start". Överlag önskades tydligare gränssnitt på gyroskopskärmen.

Önskvärda tillägg
Tydligare information angående regler för användarnamn och lösenord efterlystes av en testperson.
Möjlighet att ändra gyroskopets känslighet.
Möjlighet att förflytta sig till närliggande gatuvyer, istället för att endast kunna se sig omkring i aktuell vy. Detta ansågs vara en så pass viktig funktionalitet att den lades till.

6 Slutsats

I detta avsnitt presenteras de slutsatser som dragits efter examensarbetets genomförande. Först besvaras problemformuleringarna från avsnitt 1.3. Därefter ges en diskussion kring examensarbetets genomförande och resultat. Slutligen presenteras några framtida insatser som skulle kunna göras på området, dels med avseende på den utvecklade prototypen, och dels med avseende på funktionalitetsdelning generellt.

6.1 Problemformuleringar

I detta avsnitt besvaras examensarbetets problemformuleringar. Problemformuleringarna anges kursiverat.

Vilka egenskaper behöver ett system för funktionalitetsdelning?

Avsnitt 4.4 identifierar ett antal önskvärda egenskaper i ett funktionalitetsdelningssystem. Den viktigaste komponenten är det gemensamma språket. För att få ett system som dessutom hjälper användaren är tre mycket viktiga egenskaper intelligenta förslag, möjlighet att spara konfigurationer samt automatiserad aktivering. För automatiserad aktivering är dessutom förbättrade möjligheter för positionsbestämning av stor vikt.

Hur kan funktionalitet delas från en enhet till en annan?

Enheter som inte redan har någon sorts API eller stöd för funktionalitetsdelning behöver extra mjukvara som implementerar delningen. Prototypen som beskrivs i avsnitt 4.5 samt kapitel 5 visar hur mjukvara kan byggas för att implementera funktionalitetsdelning.

Hur kan en prototyp som visar på funktionalitetsdelningens användbarhet utvecklas med hjälp av dagens teknik?

Kapitel 5 visar upp en prototyp för funktionalitetsdelning från en mobiltelefon till en dator. Tekniken som använts har gett goda resultat, även på hårdvara som har några år på nacken. Prototypen har dessutom byggts för att vara modulär och på så sätt kunna anpassas bättre vid en eventuell vidareutveckling eller uppskalning.

Hur kan funktionalitetsdelningen initieras av en ovan användare?

Utvecklingen utgick från att en ovan användare behöver hjälp på traven i form av notifikationer. Användartester visade att sådana notifikationer behöver vara mycket tydliga, och deras placering på skärmen spelar stor roll för hur de uppfattas. Likaså är det viktigt med ett tydligt språk i användargränssnitt, särskilt då konceptet funktionalitetsdelning är nytt för många användare.

6.2 Diskussion

Denna rapport presenterar en arkitektur och prototyp för ett system för funktionalitetsdelning. Arbetet började med fokus på sessionsöverföringar, men kom sedan att istället rikta in sig på funktionalitetsdelningar. Detta fokusskifte gjorde att arbetet till viss del började om från början, men tack vare att projektet drevs agilt kunde man anpassa sig efter detta. Man kan i efterhand se för- och nackdelar med både agil och mer vattenfallsinspirerad utveckling. Det agila förhållningssättet kommer ofta igång och producerar kod snabbare, men man måste vara beredd på tvära kast. Om projektet hade bedrivits mer enligt vattenfallsmodellen hade fokusskiftet kunnat undvikas, men projektet hade då förmodligen utvecklat en helt annan produkt, inriktad mot sessionsöverföring. I slutändan var det agila arbetssättet ett bra val, eftersom det gett Tactel en prototyp som ligger närmare deras önskemål.

Prototypen som utvecklades har endast en delmängd av de funktioner som föreslås i avsnitt 4.3. Konfigurationer och automatiserade aktiveringar är exempel på funktioner som utelämnats. Detta med motivationen att dessa är beroende av en fungerande funktionalitetsdelning i sin grundläggande form. Det är funktionalitetsdelning i denna senare bemärkelse prototypen implementerar.

Den första prototypen som utvecklades hade user story A.1 (appendix A) som utgångspunkt. Arbetet fick avbrytas eftersom det visade sig att webbläsare medvetet inte exponerar den funktionalitet som krävs för att komma vidare i arbetet. Tanken var att uppladdningsfunktionen i webbläsarens JavaScript-API, som gör det möjligt att ladda upp en lokal fil, skulle kunna överskuggas med en egen definierad funktion. Vid en kort undersökning under prototyputvecklingen visade sig detta vara omöjligt p g a det säkerhetshål detta skulle medföra.

6.3 Framtid

6.3.1 Prototypen

I början av examensarbetet var det tänkt att säkerheten skulle spela en större roll och föreslogs som en huvudpunkt. Detta kom under tiden att ändras, och har därför inte undersökts i detalj. Anledningen till att säkerhetsaspekten fick ta mindre plats var att ge mer tid till att undersöka om funktionalitetsdelning som företeelse. Lösningar finns (för tillfället) för säker kommunikation, och dessa skulle utan problem kunna inkorporeras i prototypen tack vare att autentiseringsramverket Passport är så anpassningsbart.

För att förbättra autentiseringen skulle OAuth2 kunna användas. Det är en öppen standard, där användaren kan logga in via andra tjänster som tillhandahåller OAuth. Då användaren loggar in vidarebefordras vederbörande till OAuth-providern, exempelvis Google eller Twitter, loggar in, och skickas därefter tillbaka till det ställe där inloggningen startade. Fördelen med ett sådant system är att det är utvecklat av specialister med stor kunskap inom området. En bekvämlighetsaspekt är att användaren slipper registrera och komma ihåg ett nytt användarkonto.

Som en förbättring kan trafiken mellan de olika enheterna krypteras. Det gäller både autentisering och data som skickas vid funktionalitetsdelningen. I prototypen skickas inga känsliga uppgifter, men en fullfjädrad produkt kan mycket väl skicka information som är av känslig natur. Det bör dock noteras att krypteringen kan ha en negativ inverkan på prestandan, vilket kan ge en sämre upplevelse om responsivitet är viktigt. Eventuella tester kan därför vara nödvändiga för att mäta hur krypteringen påverkar responstiden. För de fall som inte kräver mycket korta responstider, till exempel skärmdelning kan det fungera bättre, då en kort fördröjning inte har så stor påverkan.

Vid valet av metod för dataöverföring föll valet på Socket.IO. Det har visat sig fungera bra, men har nackdelen att all trafik måste gå via en server och därmed bilda en möjlig flaskhals. Ett annat alternativ - WebRTC - utvärderades, men föll bort på grund av möjliga säkerhetsbrister. En stor fördel med WebRTC jämfört med Socket.IO är att WebRTC endast använder servern för att initiera förbindelsen, därefter skickas trafiken direkt mellan de två parterna, den kortaste möjliga vägen. Det finns en stor potential i detta, och tekniken bör därför undersökas vidare.

Det är möjligt att förbättra användningen av Socket.IO genom att använda det stöd för rum och namespaces ramverket erbjuder. En möjlighet är att varje användare har sitt eget rum, och att användarens enheter kommunicerar med varandra i detta rum.

Då prototypen kan användas exempelvis vid presentationer eller på teknikmässor skulle det vara en trevlig bonus att lägga till stöd för initiering via NFC. I ett sådant sammanhang skulle det kanske även passa bra att utveckla en mobil webbklient, så att en nyfiken användare kan starta upp en gyrofunktionalitetsdelning genom att hålla sin mobiltelefon över en NFC-skrivare. NFC-skrivaren skulle då ge mobiltelefonen en URL där den mobila webbklienten finns, och funktionalitetsdelningen startas.

Projektgruppen anser att dessa problem bör angripas i följande ordning:

1. Mer djupgående undersökning av möjligheterna med WebRTC.
2. Kryptering av datatrafiken, samt test av dess prestandapåverkan.
3. Införande av OAuth2 eller liknande autentiseringslösning.
4. Bättre utnyttjande av Socket.IO:s funktioner.
5. NFC-stöd.

6.3.2 Funktionalitetsdelning

Konceptet “funktionalitetsdelning” skulle kunna bli en vanlig företeelse i framtiden. En anledning till att det inte existerar idag är gapet mellan olika enheter. De har ofta inget gemensamt sätt att kommunicera med varandra. Synen på enheter som diskreta entiteter har varit förhärskande under lång tid, men denna rapport menar att tekniken för att få dem att samverka redan finns. Man börjar se enstaka fall, som exempelvis Google Chromecast och Android TV som på olika sätt gör det möjligt att spela upp media från mobiltelefonen på TV:n.

Ett problem som bör undersökas närmare är hur enheter upptäcks. Det vore önskvärt att kunna fråga efter enheter baserat på olika parametrar. Position, funktionalitet samt unik identifierare vore ett första steg mot en användbar lösning.

Ytterligare ett område som bör undersökas är hur konfigurationer kan representeras och sparas av systemet. Möjligtvis kan ytterligare begrepp införas för att skilja mellan funktionalitetsdelningen - förbindelsen - och själva innehållet som delas.

Projektgruppen rekommenderar att dessa prioriteras enligt:

1. Enhetsupptäckande.
2. Vidare utveckling av konfigurationer och relaterade begrepp.

28 maj 2015 presenterade Google (2015) vid sin utvecklarkonferens Google I/O produkten ”Brillo”. Produkten är en mjukvaruplattform för IoT-enheter, med bl a ett protokoll för kommunikation. Googles satsning på Brillo bekräftar att det finns ett starkt intresse för IoT. Med en väletablerad IoT-plattform kommer många hinder för funktionalitetsdelning försvinna, exempelvis tillhandahålls det gemensamma språk som diskuteras i 4.4.1.

7 Terminologi

Accelerometer	En typ av sensor som mäter acceleration.
API	Application Programming Interface.
Callback-funktion	En funktion som inparameter till en annan funktion, vars syfte är att anropas när den andra funktionen är klar.
DOM	Document Object Model, en trädstruktur som används för att beskriva dokument, till exempel webbsidor.
Gyroskop	Ett instrument som används för att mäta eller bibehålla ett vinkelläge. Vanligt förekommande i moderna mobiltelefoner där förändringar i olika vinklar kan mätas.
Hemautomation	Även home automation och domotics.
IDE	Integrated Development Environment, integrerad utvecklingsmiljö.
IoT	Internet of Things. Se 4.2.
NFC	Near Field Communication. En kontatktlös överföringsteknik som fungerar över korta avstånd.
Konfiguration	En kombination av funktionaliteter och enheter. Se 4.4.3 för mer information.
OAuth	Öppen standard för autentisering.
Protokoll	Ett avtal om hur kommunikationen mellan flera parter ska ske.
User story	Kortfattad beskrivning av arbetsuppgift från en slutanvändares perspektiv.
SDK	Software development kit, samling av utvecklingsverktyg riktad mot specifik plattform.
Street View	En gatuvy i Googles karttjänst. I en Street View kan användaren se sig omkring som om vederbörande stod på gatan.
Surfplatta	Tablet, mobil enhet liknande mobiltelefon, men större till formatet.
Task	Arbetsuppgift.

VoIP	Voice over IP, IP-telefoni.
WIP limit	Gräns för hur många tasks kan ligga på en specifik position i arbetsflödet vid en given tidpunkt.

8 Källförteckning

Alapetite, A. (2009). Dynamic 2D-barcodes for multi-device Web session migration including mobile phones. *Personal and Ubiquitous Computing*, 14(1), ss. 45-52.

Alexanderson, K. (2012). *Källkritik på Internet*. Stockholm: .SE:s Internetguider.

Ashton, K. (2009). *That 'Internet of Things' Thing*. http://www.rfidjournal.com/articles/view?4986 [2015-05-21]

Asus. (u.å.a.). *ASUS Transformer Pad (TF103C)*. http://www.asus.com/Tablets/ASUS_Transformer_Pad_TF103C/ [2015-05-05]

Asus. (u.å.b). *PadFone S (PF500KL)*. http://www.asus.com/Phones/PadFone_S_PF500KL/ [2015-05-05]

Bagrodia, R., Bhattacharyya, S., Cheng, F., Gerding, S., Glazer, G., Guy, R., Ji, Z., Lin, J., Phan, T., Skow, E., Varshney, G. & Zorpas, G. (2003). iMASH: Interactive mobile application session handoff. *I MobiSys '03, Proceedings of the 1st international conference on Mobile systems, applications and services*. New York, NY, 5 maj 2003 ss. 259-272. DOI: 10.1145/1066116.1066124.

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Margin, R. C., Mellor, S., Schwaber, K., Sutherland, J. & Thomas, D. (2001). *Manifesto for Agile Software Development*. http://agilemanifesto.org/ [2015-04-15]

BuiltWith. (u.å.). *jQuery Usage Statistics*. http://trends.builtwith.com/javascript/jQuery [2015-05-07]

Express. (u.å.). *Fast, unopinionated, minimalist web framework for Node.js*. http://expressjs.com/ [2015-04-20]

Git. (u.å.). *About - Git.* http://git-scm.com/about/distributed [2015-05-13]

Gomez, C. & Paradells, J. (2010). Wireless home automation networks: a survey of architectures and technologies. *IEEE Communications Magazine*, 48(6), ss. 92-101.

Google Inc. (2014). *What is V8?* https://developers.google.com/v8/ [2015-04-24]

Google Inc. (2015). *Project Brillo*. https://developers.google.com/brillo/ [2015-06-10]

Google Inc. (u.å.a) *SDK Platforms Release Notes.* http://developer.android.com/tools/revisions/platforms.html [2015-05-25]

Google Inc. (u.å.b). *Android Developer Tools.* https://developer.android.com/tools/help/adt.html [2015-05-05]

Google Inc. (u.å.c). *Google Docs*. https://www.google.com/docs/about/ [2015-04-22]

Google I/O Keynote (2014) [video]. San Francisco: Moscone Center. https://www.youtube.com/watch?v=wtLJPvx7-ys [2015-05-05]

IBM. (2003). *Rup in brief.* http://www.ibm.com/developerworks/rational/library/1826.html#N10099 [2015-05-19]

JetBrains s.r.o. (2015). *The Complete IDE for JavaScript Development.* https://www.jetbrains.com/webstorm/features/ [2015-05-13]

Joyent Inc. (2015). *About Node.js®.* https://nodejs.org/about/#index_md_about_node_js_reg [2015-04-24]

The jQuery Foundation. (2015). *What is jQuery?* http://jquery.com/ [2015-05-07]

Kniberg, H & Skarin, M. (2010). *Kanban and Scrum - making the most of both*, online edition. C4Media [2015-04-14]

Lauesen, S. (2002). *Software Requirements: Styles and Techniques*. Harlow: Pearson Education Limited.

Le Hégaret, P. (2002). *The W3C Document Object Model (DOM)*. http://www.w3.org/2002/07/26-dom-article.html [2015-05-13]

Mack, N., Woodsong, C., MacQueen, K. M., Guest, G. & Namey, E. (2005). *Qualitative Research Methods: A Data Collector's Field Guide*. Research Triangle Park, NC: Family Health International.

Mell, P. & Grance, T. (2011). *The NIST Definition of Cloud Computing* (SP800-145). Gaithersburg, MD: National Institute of Standards and Technology.

Neptune Suite - Seamless, Secure and Portable. (2015) [video]. Montreal. https://www.youtube.com/watch?v=tHZix18zVAE [2015-05-05]

Passport. (u.å.). *Overview*. http://passportjs.org/guide/ [2015-04-20]

Phan, T., Guy, R., Gu, J. & Bagrodia, R. (2001). A new TWIST on mobile computing: Two-Way Interactive Session Transfer. *I WIAPP 2001, Proceedings of The Second IEEE Workshop on Internet Applications*. San Jose, CA 23-24 juli 2001, ss. 2-11. DOI: 10.1109/WIAPP.2001.941864

Pippal, R.S., Jaidhar, C.D., Tapaswi, S. (2013). A novel smart card mutual authentication scheme for session transfer among registered devices. I *IACC 2013, Proceedings of the 3rd International Advanced Computing Conference*. Ghaziabad, Uttar Pradesh, India 22-23 februari 2013, ss. 1540-1547. DOI: 10.1109/IAdCC.2013.6514456

Pushbullet Inc. (u.å.). *About Us*. https://www.pushbullet.com/about [2015-04-01]

Sequelize. (u.å.). *Sequelize*. http://docs.sequelizejs.com/en/latest/ [2015-04-20]

Shacham, R., Schulzrinne, H., Thakolsri, S., & Kellerer, W. (2009). *Session initiation protocol (SIP) session mobility*. IETF RFC5631.

Socket.IO. (u.å.). *Socket.IO*. http://socket.IO/ [2015-05-14]

Virzi, R. A. (1992). Refining the Test Phase of Usability Evaluation: How Many Test Subjects is Enough?. *The Journal of the Human Factors and Ergonomics*, 34(4), ss. 457-468.

WhatsApp Inc. (u.å.). *How it works*. https://www.whatsapp.com/ [2015-04-01]

Wood, A. (2015). *The internet of things is revolutionising our lives, but standards are a must*. http://www.theguardian.com/media-network/2015/mar/31/the-internet-of-things-is-revolutionising-our-lives-but-standards-are-a-must [2015-05-21]

Appendix A - User stories

A.1

As a computer user uploading an image via a web form, I want to be able to immediately capture an image with my mobile device's camera and use it as input, so I don't have to manually transfer it to the computer.

Som datoranvändare stöter man förr eller senare på hemsidor som kräver registreringar. I många fall har man möjligheten att lägga upp en profilbild för att ge ens konto en mer personlig känsla. En vanlig förekommande följd är då att man laddar upp en existerande bild, eller tar fram en kamera för att fota och sedan överföra till datorn för att slutligen ladda upp den. Det senare alternativet kan vara svårhanterligt och tidskrävande. Detta koncept kan appliceras på andra fall, exempelvis då man laddar upp bilder för annonser.

Utifrån detta scenario ska en lösning tas fram för att ge användaren fler alternativ då vederbörande klickar på knappen för att välja fil i webbläsaren. Bilden ska då kunna väljas direkt i filsystemet eller hämtas från en kopplad mobil enhet. Om användaren väljer det senare alternativet vaknar den mobila enhetens kamera eller galleriapplikation. Därefter väljer användaren en bild och den överförs till datorn där den automatiskt väljs i webbformuläret.
Den här user story togs fram i ett intern brainstormingmöte.

A.2

As a Google Maps user, I want to be able to use my mobile device's gyroscope as input, so I can control the street view orientation by moving my device.

I Google Map Street View finns möjligheten för användaren att orientera sig i en tredimensionell miljö. Ofta används det för att kunna realisera en bild och för att man sedan ska kunna navigera i den verkliga miljön. I dagsläget sker denna orientering med mus och tangentbord, vilket kan kännas lite besvärlig. För användare av bärbara datorer är den här biten ännu knepigare, med en musplatta.

Att använda gyroskop för att kontrollera orientering i en tredimensionell miljö antas vara en intuitiv interaktion för användaren. Google Maps Street View är ett sammanhang där sådana tredimensionella miljöer förekommer i vardagen.

Ett annat sammanhang skulle kunna vara presentationer av tredimensionell grafik, exempelvis för att demonstrera en produkt. En lösning ska tas fram där användaren kan kontrollera en Street View med hjälp av mobiltelefonens sensorer.
Den här user story togs fram i en fokusgrupp med representanter från Tactel.

A.3 Slopade user stories

Här listas några av de user stories som togs fram tidigt under examensarbetet, men som sedan förkastades till förmån för A.1 och A.2.

A.3.1

As a user of a system (on my computer) requiring a GPS position, I want to be able to use my mobile device's GPS receiver to give the computer my position, so that I don't have to enter my position manually.

A.3.2

As a user of a system (on my computer) requiring authentication, I want to be able to authenticate myself on my computer using my mobile device's fingerprint reader, so that I don't have to manually enter username and password.

A.3.3

As a user of a mobile device, I want to be able to share my view with multiple devices, so that I can show a presentation to others

A.3.4

As person giving a presentation, I want to be able to use my mobile device's touch screen as a controller, so I can change slide remotely.

Appendix B – Test

För att undersöka möjliga fallgropar med avseende på användbarhet har användartester genomförts på fyra testpersoner. I detta appendix presenteras först de olika testpersonerna kort. Därefter ges en beskrivning av varje testfall, samt resultatet av testerna.

Testpersoner
För mer ingående beskrivning av respektive testperson, se avsnitt 3.2.2.

Person 1: Mjukvaruutvecklare

Person 2: Tekniskt ointresserad

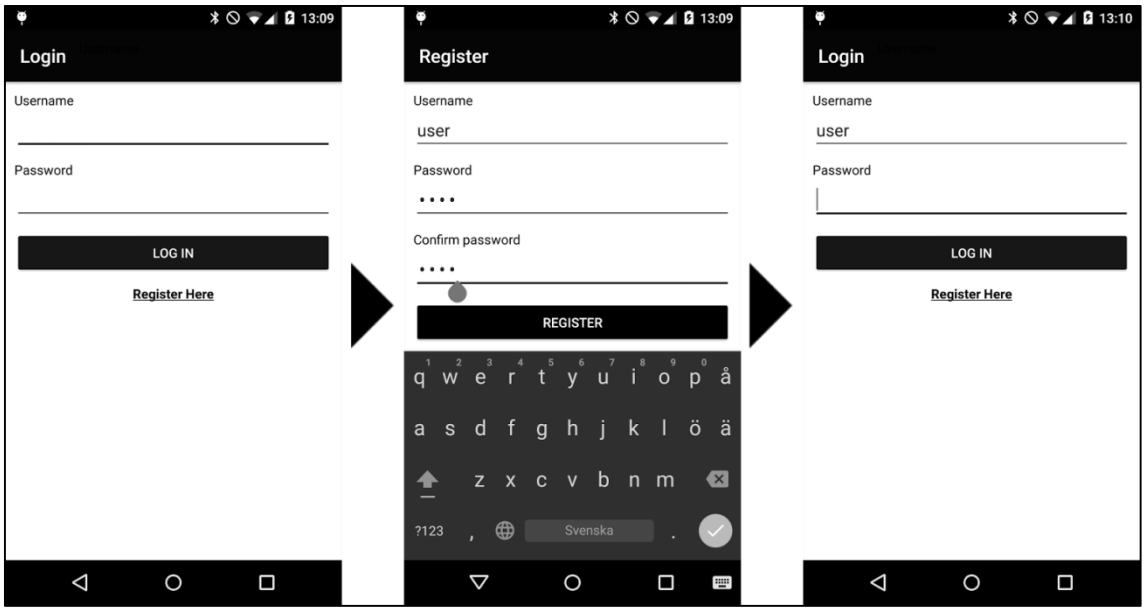
Person 3: Studerande

Person 4: Teknikintresserad, men inte insatt

B.1 Registrering i mobilapplikation (figur B.1)

Scenario: Applikationen är startad och visar en inloggningskärm med två fält benämnda “Username” och “Password”, en registreringslänk märkt “Register Here” och en knapp märkt “Log in”. Användaren vill registrera sig och trycker på registreringslänken. Användaren vidarebefordras till registreringsskärmen som visar tre nya fält benämnda “Username”, “Password” och "Confirm password" och en knapp märkt “Register”. Ett användarnamn och ett lösenord bestäms av användaren som sedan skriver in dem i respektive fält. Därefter trycker användaren på knappen “Register”. Användaren bemöts av förloppsindikator för att sedan åter hamna på inloggningsskärmen med det registrerade användarnamnet ifyllt, som en indikation på lyckad registrering.

Start: Applikationen har startats och är i fokus.
Slut: Applikationen visar inloggningsskärmen med det registrerade användarnamnet ifyllt.



Figur B.1: Figuren beskriver flödet vid registrering i Android-applikationen.

Person 1: Användaren slutförde testet utan problem. Enda funderingen var om det fanns begränsningar i val av användarnamn och lösenord.

Person 2: Användaren hade inga problem att slutföra uppgiften, men hade önskat att bekräftelse av att registreringen lyckades var tydligare.

Person 3: Användaren slutförde testet utan problem.

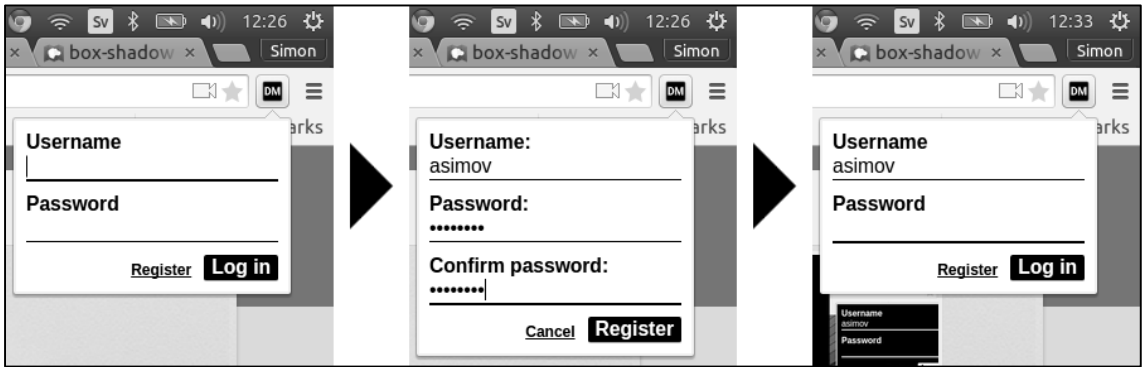
Person 4: Användaren slutförde testet utan problem men var osäker om registreringen genomfördes korrekt p g a brist av feedback.

Testet utfördes utan svårigheter av samtliga testare. Överlag önskades tydligare feedback vid lyckad registrering och information angående begränsningar i användarnamn och lösenord.

B.2 Registrering i Chrome-extension (figur B.2)

Scenario: Webbläsaren är startad och användaren får veta att knappen bredvid webbläsaren används för att komma åt inloggning och registrering. Då användaren klickar på knappen visas ett inloggningsfönster med två fält benämnda "Username" och "Password", en registreringslänk märkt "Register" och en knapp märkt "Log in". Användaren vill registrera sig och trycker på registreringslänken. Användaren vidarebefordras till registreringsskärmen som visar tre nya fält benämnda "Username", "Password" och "Confirm password" samt en knapp märkt "Register". Ett användarnamn och ett lösenord bestäms av användaren som sedan skriver in dem i respektive fält. Därefter trycker användaren på knappen "Register". Användaren hamnar sedan åter på inloggningsskärmen med det registrerade användarnamnet ifyllt, som en indikation på lyckad registrering.

Start: Webbläsaren är startad och har extensionen installerad och aktiverad. Användaren instrueras att klicka på extension-knappen för att komma åt inloggning och registrering.
Slut: Extensionen visar inloggningsskärmen med det registrerade användarnamnet ifyllt.



Figur B.2: Figuren beskriver flödet vid registrering i webbläsaren.

Person 1: Användaren slutförde testet utan problem. Användarnamnet som matades in var redan upptagen och användaren uppmanades att välja ett nytt (se avsnitt 5.3.2 för exempel).

Person 2: Testet fortlöpte utan problem, trots att det var användarens första kontakt med insticksprogram i webbläsare. Användaren hade inga problem att uppfatta när målet var nått.

Person 3: En bugg upptäcktes, vilket förvirrande användaren något. Bortsett från det slutförde användaren testet utan problem.

Person 4: Användaren slutförde testet utan problem men var osäker om registreringen genomfördes korrekt p g a brist av feedback. Användaren undrade även varför lösenordet var förifylld efter registreringen.

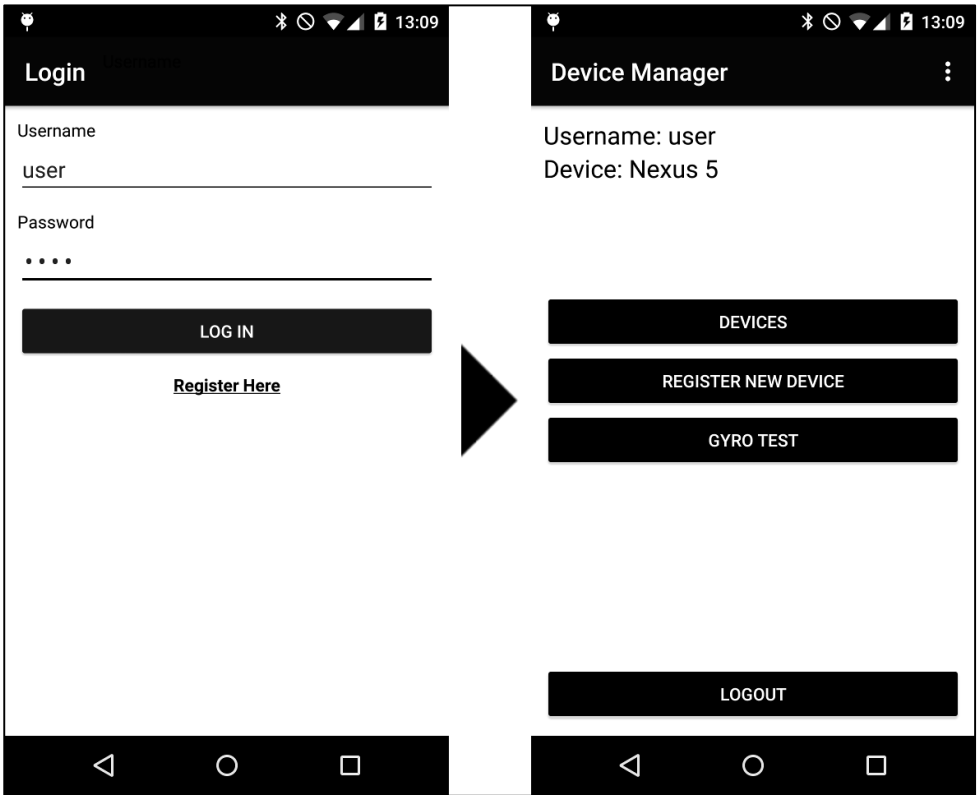
Testet utfördes utan svårigheter av samtliga testare. Överlag önskades tydligare feedback vid lyckad registrering och information angående begränsningar i användarnamn och lösenord.

B.3 Inloggning

B.3.1 Mobilapplikation (figur B.3)

Scenario: Applikationen är startad och visar en inloggningskärm med två fält benämnda “Username” och “Password”, en registreringslänk märkt “Register Here” och en knapp märkt “Log in”. Användaren matar in sitt användarnamn och lösenord i respektive fält och trycker därefter på “Log in”. Användaren bemöts av förloppsindikator för att sedan hamna på huvudskärmen som en indikation på lyckad inloggning.

Start: Applikationen har startats och är i fokus.
Slut: Applikationen visar huvudskärmen.



Figur B.3: Figuren beskriver flödet vid inloggning i Android-applikationen.

Person 1: Användaren slutförde testet utan problem.

Person 2: Användaren slutförde testet utan problem.

Person 3: Användaren slutförde testet utan problem.

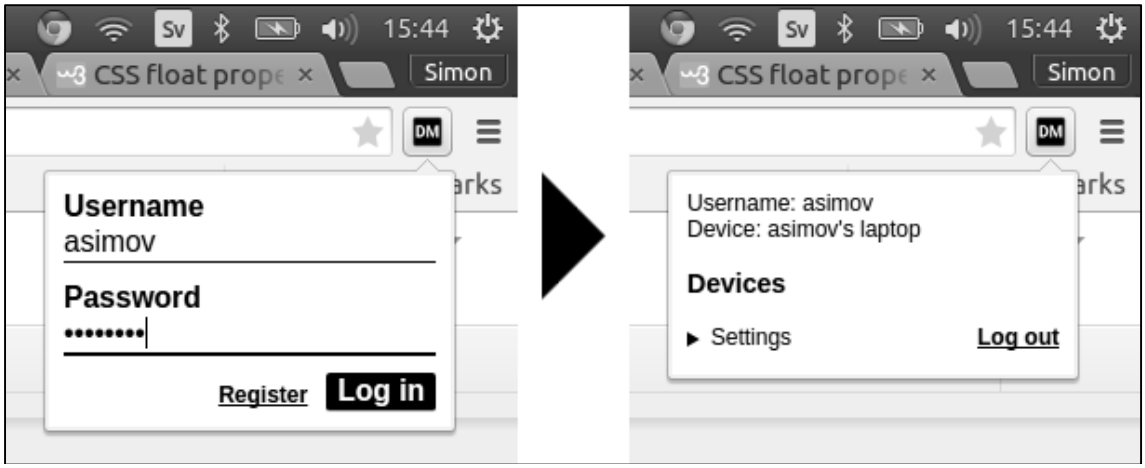
Person 4: Användaren slutförde testet utan problem.

Testet utfördes utan svårigheter av samtliga testare. Användarna hade inga problem att avgöra när de var inloggade.

B.3.2 Chrome-extension (figur B.4)

Scenario: Webbläsaren är startad och användaren får veta att knappen bredvid webbläsaren används för att komma åt inloggning och registrering. Då användaren klickar på knappen visas ett inloggningsfönster med två fält benämnda “Username” och “Password”, en registreringslänk märkt “Register” och en knapp märkt “Log in”. Användaren matar in användarnamn och lösenord i respektive fält och trycker därefter på “Log in”. Användaren hamnar därefter på huvudskärmen som en indikation på lyckad inloggning.

Start: Webbläsaren är startad och har extensionen installerad och aktiverad. Användaren instrueras att klicka på extension-knappen för att komma åt inloggning och registrering.
Slut: Extensionen visar huvudskärmen.



Figur B.4: Figuren beskriver flöden vid inloggning i webbläsaren

Person 1: Användaren slutförde testet utan problem.

Person 2: Användaren slutförde testet utan problem.

Person 3: Användaren slutförde testet utan problem.

Person 4: Användaren slutförde testet utan problem.

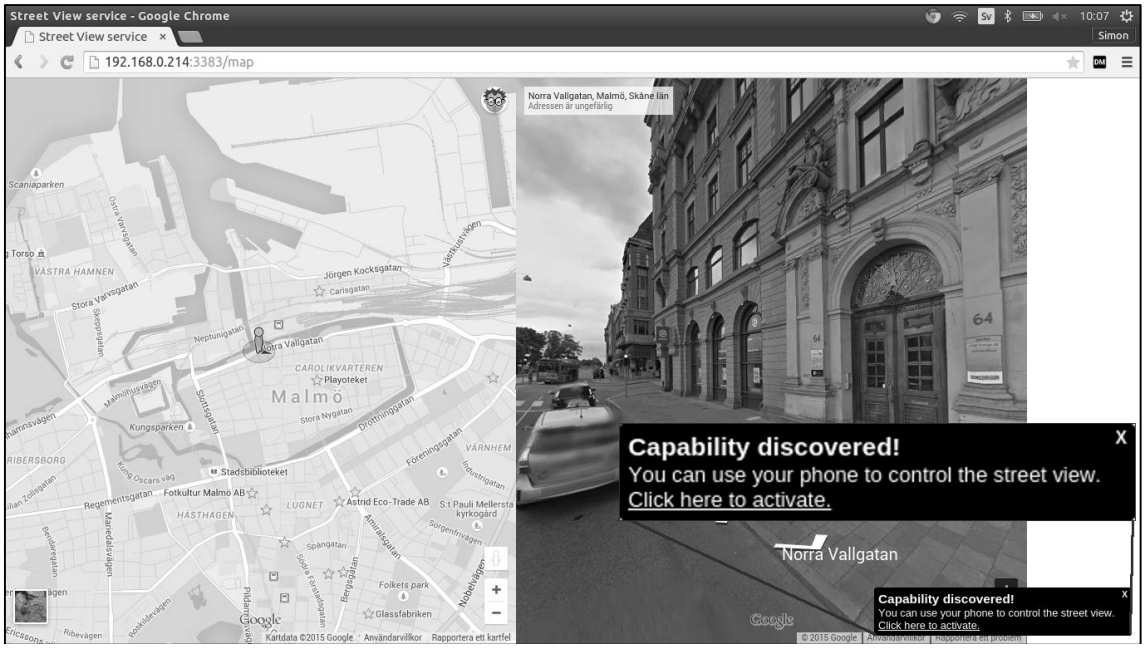
Testet utfördes utan svårigheter av samtliga testare. Användarna hade inga problem att avgöra när de var inloggade.

B.4 Gyrostyrning av Street View

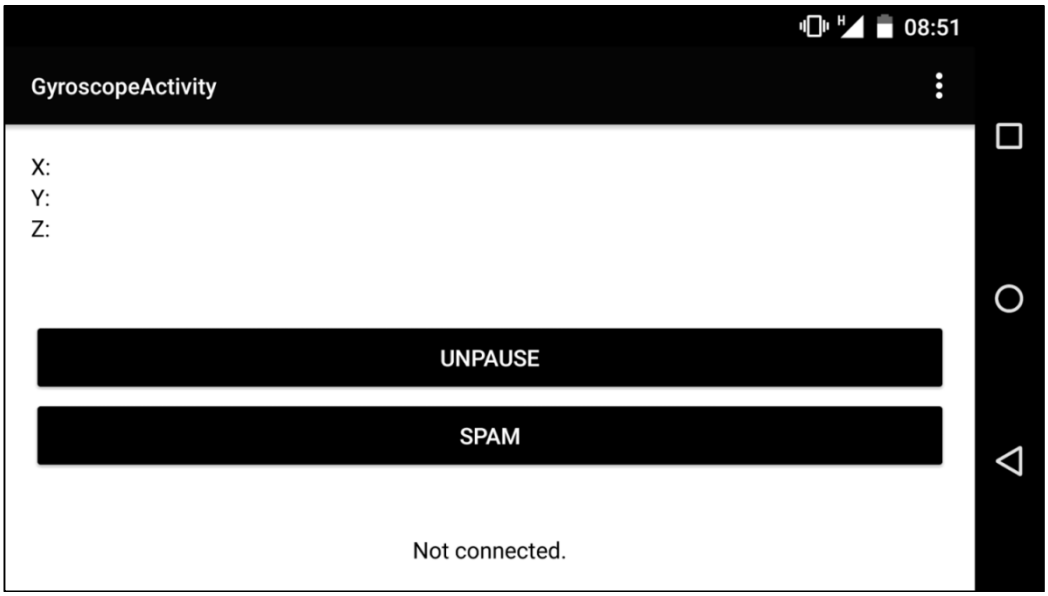
Scenario: Användaren uppmanas att besöka en specifik hemsida med en anpassad version av Google Maps. När hemsidan har laddat klart får användaren en notifikation i form av en liten pop-up från Chrome-extensionen (figur B.5). Den upplyser användaren att hemsidan stödjer gyrofunktionalitet som finns på en av användarens registrerade enheter och frågar om användaren vill påbörja en funktionalitetsdelning. Användaren initierar funktionalitetsdelningen genom en länk i pop-up:en.

På den mobila enheten dyker då en notifikation upp med en förfrågan om funktionalitetsdelning från Chrome-extensionen. Om användaren accepterar visas en skärm benämnd "Gyroscope", och funktionalitetsdelningen är därmed påbörjad (figur B.6).

Start: Användaren är inloggad i Chrome-extensionen på en dator samt i Androidapplikationen på en mobiltelefon.
Slut: Användaren kan manövrera Google Street View på datorn genom mobiltelefonens gyroskop.



Figur B.5: Figuren visar webbsidan och notifikationen - här förstorad - som talar om för användaren att mobiltelefonen kan användas för att styra gatuvyn till höger.



Figur B.6: Figuren visar Android-applikationens gyroskärm.

Person 1: Användaren slutförde testet utan problem med hade önskat en tydligare beskrivning. En inställning för känsligheten för gyroskopet önskades också för att inte behöva vrida telefonen till en onaturlig vinkel.

Person 2: Användaren hade inga problem att förstå att pop-up:ens länk. När appens Gyro-skärm visades var det oklart vad knapparna betydde. Användaren förstod vad som menades med "Unpause" (startläget) först efter att ha tryckt på knappen så att texten ändrades till "Pause".

Person 3: Användaren hittade efter en stund notifikationen och klickade då på länken. Användaren upplevde ordvalet på knapparna som förvirrande.

Person 4: Användaren slutförde testet utan problem men hade önskat ett tydligare gränssnitt på applikationens gyro-skärm och undrade om det fanns knappar för att gå.

Gränssnittet på gyroskärmen upplevdes generellt som förvirrande. Det framkom också önskemål på att kunna justera gyroskopets känslighet. Prototypen kan även bli tydligare om notifikationen får en mer framträdande plats.