CHALMERS

Resourcee

- en rumsbokningsapplikation för Android

Kandidatarbete inom Informationsteknik

HAMMAR, JOEL
HARRYSON, MARCUS
JONSSON, ANDERS
MANUCHERI, ARON
SUTER, LUKAS
SÖDERBERG, MÅRTEN

Chalmers tekniska högskola
Göteborgs universitet
Institutionen för Data- och Informationsteknik
Göteborg, Sverige, Juni 2013

Resourcee
- a room booking application for Android

Examiner: ARNE LINDE

Chalmers University of Technology
University of Gothenburg
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Department of Computer Science and Engineering
Göteborg, Sweden June 2013

Abstract

This thesis concerns the development of Resourcee, an Android application developed for tablet computers on behalf of the software company Understandit. Booking a conference room is more complicated than necessary in most room booking systems. Resourcee introduces the ability to book a room on-site, using a tablet mounted outside of each conference room, simplifying the room booking process.

Composition of a requirements specification, by the instructions given by Understandit as well as from a survey, preceeded this project's software development phase. Using this requirements specification, general development was handled in an iterative manner.

The prototype produced through this development phase compared well to the requirements given by Understandit. Future development of Resourcee has been held in regard throughout the entire development phase, meaning that further development should be relatively easy.

This project was carried out at Chalmers University of Technology in Gothenburg, Sweden.

Sammanfattning

Denna rapport behandlar utvecklingen av Androidapplikationen Resourcee som utvecklats på uppdrag av företaget Understandit. Att boka mötesrum är i de flesta bokningssystem mer komplicerat än nödvändigt. Genom Resourcee tillförs en utvidgning till det befintliga bokningssystemet i Google Apps. Applikationen har designats för surfplattor som monteras utanför bokningsbara rum. Genom applikationen kan bokning ske på plats.

Utvecklingen inleddes med att kravspecifikationen för Resourcee fastslogs utifrån uppdragsgivarens instruktioner samt en inledande enkätundersökning. Resourcee implementerades iterativt med utgångspunkt från kravspecifikationen.

En prototyp har färdigställts och överensstämmer väl med de funktionella krav uppdragsgivaren specificerat. Användartester visade att användargränssnittet var intuitivt och lättanvänt. Det finns dessutom en god grund för vidareutveckling av Resourcee, vilket eftersträvats genom utvecklingen.

Projektet har utförts i Göteborg, på Chalmers tekniska högskola.

Ordlista

Activity

En Activity är en komponent i Android vars uppgift är att sköta interaktion mellan applikation och användare.

Android

Android är ett Linuxbaserat operativsystem skapat främst för mobila enheter.

Användningsfall

Användningsfall är ett utvecklingsverktyg som beskriver flöden hos funktioner i program.

Application Programming Interface (API)

Application Programming Interface är ett gränssnitt där ett programs funktionalitet görs tillgänglig.

Broadcast Receiver

Broadcast Receiver är en Androidkomponent som lyssnar på meddelanden och förändringar skickade från applikationer och operativsystemet Android.

Content Provider

Content Provider är ett interface i Android som hanterar data mellan olika applikationer och processer.

Eclipse

Eclipse är en mjukvaruutvecklingsmiljö och används främst för programmering i Java.

Fragment

Fragment är en komponent som har stora likheter med Activity. Skillnaden består i att varje Fragment existerar inuti en Activity och att flera instanser kan vara aktiva samtidigt.

Git

Git är ett versionshanteringssystem för källkodshantering som möjliggör samarbete vid mjukvaruutveckling.

Google Calendar

Google Calendar är namnet på en kalendertjänst på webben, samt en kalenderhanteringsapplikation för Android. De två produkterna är anknutna till varandra.

Google Play Services

Google Play Services är en tjänst för Androidenheter. Tjänsten består av ett API som, bland annat, erbjuder möjlighet för enheter att autentisera sig mot och kommunicera med Googles API.

Grupprogram

Ett grupprogram är ett program som underlättar arbete i grupp. Grupprogram innehåller ofta kalenderfunktionalitet samt dokument- och kontakthantering.

HyperText Transfer Protocol (HTTP)

Hypertext Transfer Protocol är ett protokoll för överföring av data över internet.

Intent

Intent är ett Androidverktyg som sköter kommunikation både inom och mellan applikationer.

JavaScript Object Notation (JSON)

JavaScript Object Notation är ett standardiserat format för datautbyte.

Linux

Linux är ett operativsystem skrivet med öppen källkod.

Molnbaserad datortjänst

Molnbaserad datortjänst är ett begrepp som syftar på distribuerad datahantering. Även kallad molntjänst.

Model-View-Controller (MVC)

Model-View-Controller är ett välkänt mönster för systemdesign och arkitektur.

OAuth

OAuth är en metod för att autentisera att en tjänst har rätt att komma åt en särskild resurs.

Software Development Kit (SDK)

Software Development Kit är en samling mjukvara som möjliggör utveckling för en viss mjukvarumiljö.

Förord

Rapporten är ett kandidatarbete vid Institutionen för Data- och informationsteknik, Chalmers tekniska högskola, och utfördes våren 2013. Projektgruppen önskar tacka Understandit för deras projektidé och vägledning. Vi önskar även tacka Christer Carlsson för den tid han la ner på att handleda projektet.

Innehållsförteckning

1. Inledning

"Technological convergence is the tendency for different technological systems to evolve toward performing similar tasks." (Olawuyi och Mgbole 2012).

Att teknologisk konvergens är ett fenomen som kännetecknar vår tid råder det inga tvivel om. Mobiltelefonen och musikspelaren konvergerade till Iphone, dagens spelkonsoler kan visa film och internet finns redan i moderna kylskåp (Apple 2007; Xbox 2013; Samsung 2012). Programvaror som stödjer administration av arbete i grupp utgör ett teknikområde där teknisk konvergens är tydlig på två sätt.

För det första har spridd mjukvara samlats i *grupprogram*, vilket är program som förenklar samarbete i grupp (Hastings 2009). Exempelvis har e-postprogram, kalenderprogram och ordbehandlingsprogram samlats i grupprogram som Google Apps och Microsoft Exchange. För det andra har grupprogram även börjat erbjudas genom distribuerade internettjänster, så kallade *molntjänster*. Till exempel erbjuds Google Apps enbart som molntjänst, där grupprogrammets funktionalitet tillgängliggörs genom webbsidor. Även grupprogrammet Microsoft Exchange erbjuds som en molntjänst under namnet Exchange Online (Microsoft 2013 #1).

Företag övergår idag i stor omfattning från att hantera sina IT-tjänster till att använda molnbaserade grupprogram (Kelly 2012). Denna migration grundar sig sannolikt i den höga lönsamhet det innebär att överge det egna underhållet relaterat till tjänsterna (Forrester 2009; Forrester 2012). Så sent som år 2006 grundades Google Apps, en tjänst som numera har fem miljoner företagskunder (Lardinois 2012).

Marknadsundersökningsföretaget Forrester påvisar i två undersökningar den ekonomiska påverkan för företag att byta från eget underhåll av grupprogram till motsvarande externa tjänster. I båda fallen är det tydligt att ansenliga besparingar har gjorts vid byte till de undersökta grupprogrammen. Undersökningarna fokuserade på att beräkna lönsamheten på den investering det innebar att genomgå en migration till ett annat datorsystem, för de två exempelföretagen (Forrester 2009; Forrester 2012). För exempelföretaget som bytte till Microsoft Exchange betalade sig migrationen på 5,2 månader och för exempelföretaget som bytte till Google Apps betalade sig migrationen på 1,4 månader (Forrester 2009; Forrester 2012).

Såväl Google Apps som Microsoft Exchange inkluderar ett kalendersystem där användarna kan schemalägga händelser och boka rum. Användarna kan sedan dela med

sig av dessa händelser till varandra via ett webbgränssnitt (Google Calendar 2013; Microsoft 2013 #2).

Det finns utvidgningar till kalendersystemen som tillför möjlighet att kontrollera olika rums status utan att systemens webbgränssnitt används. Genom produkter som monteras vid varje rum och kopplas upp mot ett grupprogram ges användare möjligheten att boka rum på plats. Dessa lösningar säljs i ett paket med både hårdvaruplattform och mjukvara (Evoko 2013).

1.1. Bakgrund

Det Göteborgsbaserade företaget Understandit tog hösten år 2012 fram den projektbeskrivning som ligger till grund för detta kandidatarbete (se bilaga A). Visionen var att skapa en produkt som fysiskt representerade ett rum i ett grupprogram. Produkten skulle bestå av en mjukvarulösning till en utbytbar hårdvaruplattform. Detta skulle differentiera produkten från liknande lösningar som redan existerar och dessutom förenkla distribuering.

Understandit erbjöd sig att stå till förfogande under projektets utveckling. Företaget har arbetat med IT-lösningar sedan år 2007 och har stor kompetens inom området. Framför allt har företaget stor erfarenhet av arbete gentemot kunder samt utformning av användargränssnitt, två områden som båda är av betydelse för detta projekt (Understandit 2013).

1.2. Syfte

Syftet med detta projekt är att ta fram en prototyp av en rumsbokningsapplikation riktad mot företag och organisationer. Prototypen kallas Resourcee och ska integreras med etablerade grupprogram. Surfplattor som kör applikationen ska komplettera dessa grupprogram genom att monteras utanför rum och möjliggöra bokning. Den ska även interagera med användare såväl som kommunicera med grupprogrammet. Understandit hade som avsikt att projektet skulle resultera i en fungerande kommersiell prototyp som de kan vidareutveckla.

Resourcee ska kunna installeras på ett brett urval surfplattor. Applikationen ska vara lättanvänd, med ett tydligt användargränssnitt. Då Resourcee är tänkt att representera ett unikt rum i ett grupprogram ska det tydligt indikeras om rummet är ledigt. Endast bokningar av rummet med start i nutid ska erbjudas från applikationen.

1.3. Avgränsningar

Uppdragsgivaren uttryckte tidigt att projektet ska resultera i en applikation anpassad för operativsystemet Android. Anledningen till att Android valts av Understandit var för att ge stora valmöjligheter vad gäller hårdvara.

Tidigt bestämdes att projektets resulterande prototyp ska använda sig av Google Apps för datalagring och kommunikation. Detta innebär att enheter anslutna till systemet behöver en pålitlig internetuppkoppling.

2. Teknisk bakgrund

Detta kapitel beskriver den teknik som använts för att genomföra projektet. Den största delen av kapitlet berör operativsystemet Android och hårdvaran Android används på. Kapitlet tar upp designmönster som därmed haft betydelse för projektet. Vidare ger kapitlet en grundläggande beskrivning av grupprogrammet Google Apps, samt hur interaktion med detta sker. Därpå följer en kort beskrivning av utvecklingsmiljöer och versionshanteringssystem samt slutligen en redogörelse för användargränssnitt.

2.1. Surfplatta

Efter att Apple år 2010 lanserade Ipad slog surfplattan igenom (Nationalencyklopedin 2013) och blev samma år utsedd till årets julklapp av Handelns Utredningsinstitut (HUI Research 2013). Trots att surfplattan fick sitt kommersiella genombrott såpass nyligen är dess ursprungliga koncept minst ett tiotal år äldre (Markoff 1999).

Surfplattan har stora likheter med smarttelefoner, men har i allmänhet en större skärm. Denna skärm är ofta av typen lysdiod, vilket innebär att den tydligt presenterar innehåll. Innehållet blir därmed synligt från de flesta vinklar oavsett omgivningens belysning. För att surfplattan ska leva upp till sitt namn krävs det att den har möjlighet till trådlös internetanslutning (Nationalencyklopedin 2013).

Apple har sedan lanseringen av Ipad dominerat försäljningen av surfplattor, en dominans som hållit i sig åtminstone fram till år 2012. Dock har många nya aktörer tagit sig in på marknaden. Apples marknadsandel har sjunkit från att omfatta 81 % av innehavda surfplattor i USA år 2011, till 52 % året efter. Samtidigt som Apples dominans avtagit har den andel surfplattor som kör Android ökat från 15 % till 48 % (Pew Research Center 2012).

Situationen i Sverige år 2013 påminner om den i USA år 2012: Surfplattor som kör antingen Android eller Apples operativsystem Ios dominerar marknaden med marknadsandelar på 43 % respektive 48 % (Stray 2013). Sammanfattningsvis finns det idag en stor och fortfarande växande andel av Androidsurfplattor på marknaden (Gartner 2013).

2.2. Operativsystemet Android

I början av 2000-talet lanserades operativsystemet Android av det nystartade företaget Android Inc. Android var först tänkt att bli ett operativsystem för digitalkameror, vilket

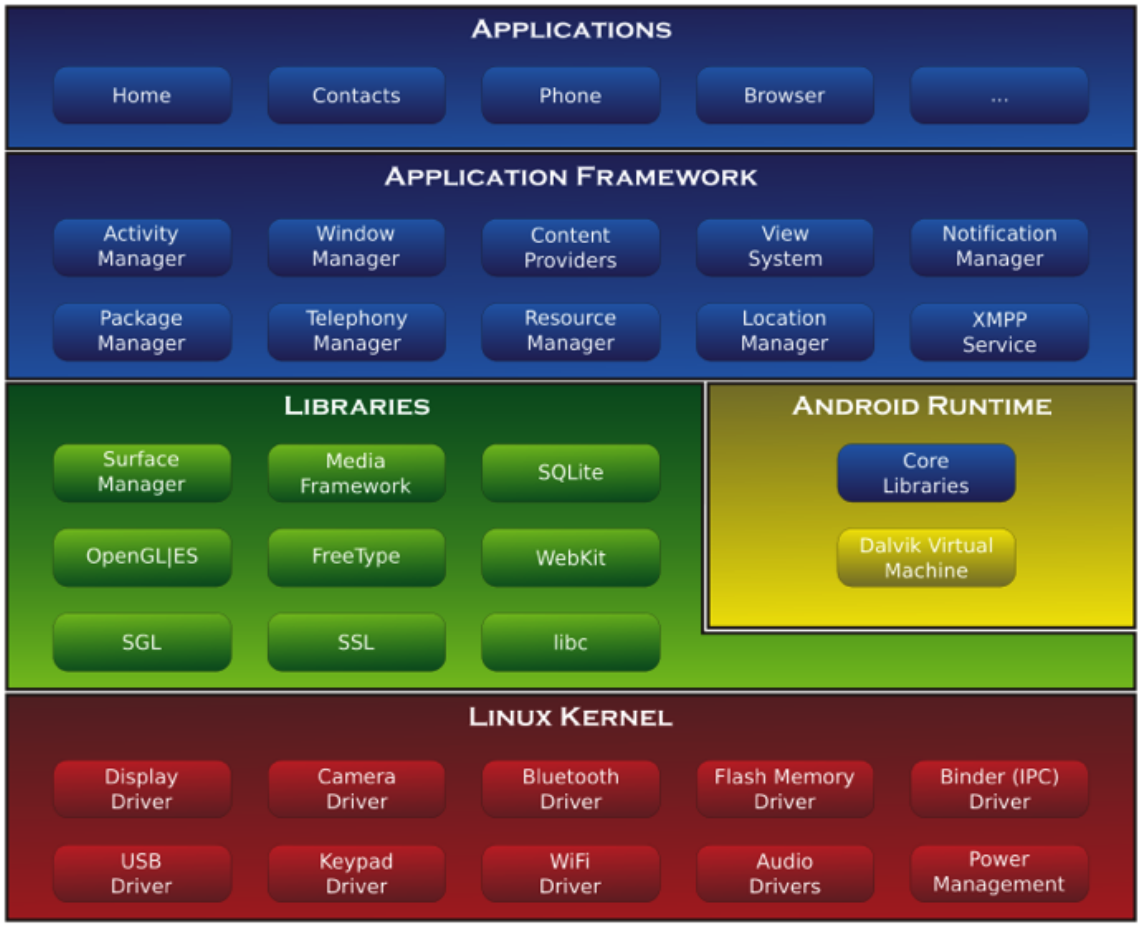
inte blev fallet då projektet saknade tillräcklig investering. Istället gick utvecklingen mot att ta fram ett operativsystem för det som numera kallas smarttelefoner (Markoff 2007).

År 2005 köptes Android Inc. av Google, som därmed erhöll rättigheterna till operativsystemet Android (Elgin 2005). Två år senare förklarades att Androids källkod skulle publiceras med en tillåtande licens, vilket innebar att fler smarttelefonaktörer kunde etablera sig med hjälp av operativsystemet. Detta var ett strategiskt beslut som innebar att Google kunde etablera sig på smarttelefonmarknaden (International Data Corporation 2013) trots att många aktörer redan fanns (Markoff 2007). Det sista kvartalet år 2012 var Android det vanligaste operativsystemet för mobila enheter. Vid den tidpunkten hade drygt sju av tio mobila enheter Android som operativsystem.

2.2.1. Androids grundläggande struktur

Android är uppbyggt med rötter i det öppna operativsystemet Linux och kan delas in i fyra mjukvarulager. Varje lager uppfyller en särskild uppgift, vilket illustreras i figur 1 (Jordan och Greyling 2011, s. 21-23).

Androids första och mest maskinnära lager, *Linux Kernel*, innehåller drivrutiner och annan hårdvaruabstraherande funktionalitet. Detta lager etablerar en standard för interaktion med enhetens hårdvara, vilket innebär att övrig mjukvara på enheten kan skrivas anpassat för detta mjukvarulager snarare än för specifik hårdvara. För att kunna stödja det breda utbud av Androidenheter som kan ha olika processorarkitektur (Android 2013) utgör det här lagret en viktig grundpelare.



Figur 1: En illustration över de fyra separata lager som utgör Androids struktur: Linux Kernel, Libraries (inklusive Android Runtime), Application Framework och Applications. Bildkälla: Smieh 2012.

Androids andra lager, *Libraries*, omfattar den funktionalitet som finns inbyggd i Android. Detta lager inkluderar logik för presentation av grafik, datahantering, kryptering med mera. Vidare ingår även kärnan i Androids applikationshantering, *Android Runtime*. Applikationer i Android programmeras mot denna kärna. Kärnan erbjuder ett grundbibliotek och en virtuell maskin med kapacitet att exekvera källkod.

Det tredje lagret i Android, *Application Framework*, erbjuder ett gränssnitt för installerade applikationer att kommunicera med varandra eller med inbyggda bibliotek. Lagret erbjuder bland annat möjlighet att lyssna efter statusuppdateringar i enheten, geografiskt lokalisera enheten och känna igen accelerationsförändringar.

Slutligen utgörs det fjärde lagret, *Applications*, av de applikationer som installeras på enheten. Applikationer kan installeras av användaren eller vara förinstallerade på

enheten. Oavsett hur applikationer installerats behandlas de lika av Androids virtuella maskin.

2.2.2. Dalvik

Virtuella maskiner används av många programmeringsspråk för att anpassa källkod till olika hårdvara. Källkod kompileras då för den virtuella maskinen istället för hårdvaran den körs på (Craig 2006). Detta medför att det blir enklare att stödja olika plattformar med samma källkodsbas. Istället för att skriva om applikationer till en annan plattform anpassas den virtuella maskinen. Detta leder till att applikationer skapade för den virtuella maskinen kan köras på den nya plattformen. Androids virtuella maskin, *Dalvik*, följer denna princip (Google I/O 2010).

Dalviks huvudsakliga uppgift består av att exekvera applikationer, samt sköta minneshantering och resursfördelning (Hyeong-Seok 2012). Dalvik tillhandahåller även en *skräpsamlare*, vilket är en mjukvaruprocess som automatiskt upptäcker allokerade minnesutrymmen som inte längre refereras till. Dalvik kan då frigöra dessa minnesutrymmen som därmed kan återanvändas (Ehringer 2010). Detta medför att utvecklaren inte behöver ta hänsyn till minneshantering.

Applikationer för Android skrivs huvudsakligen i Java med bibliotek ämnade för Androidutveckling. Den kompilerade källkoden körs i Android genom Dalvik. Vid kompilering genereras *bytekod*, vilket är plattformsoberoende källkod för virtuella maskiner. I ett första skede är detta Javabytekod, det vill säga den bytekod som körs i de flesta virtuella maskiner skapade för Java. Sedan konverteras denna bytekod så den kan köras av Dalvik (Ehringer 2010). Att Android har en egen virtuell maskin, Dalvik, för att exekvera Javakällkod grundar sig på att mobila enheter historiskt haft begränsad prestanda jämfört med persondatorer (Bornstein 2008).

2.2.3. Programmeringsspråk

De flesta Androidapplikationer skrivs uteslutande i Java (Android Developers 2013 #4). Det är dock fullt möjligt att skriva en applikation helt i *native code*, det vill säga källkod som kompileras för en specifik processorarkitektur (Android Developers 2013 #13). Detta medför att utvecklare kan använda en existerande källkodsbas, skriven i native code, och köra den på Android. Google avråder dock från att skriva applikationer i native code, eftersom dess komplexitet oftast överskuggar de eventuella prestandavinsterna (Android Developers 2013 #13).

Androidapplikationer kan till stor del skrivas i programmeringsspråk ämnade för webbutveckling. Detta görs genom teknik inbyggd i Android, tillgänglig via ett

Application Programming Interface (API), ett gränssnitt som specificerar den faktiska funktionaliteten (Android Developers 2013 #14).

2.2.4. Androidversioner

Med varje ny version av Android utökas funktionaliteten som applikationer kan ta nytta av, samt ger utvecklare mer och kraftfullare verktyg för att utnyttja operativsystemet i sina applikationer (Android Developers 2013 #9). Denna funktionalitet tillgängliggör Android genom sitt API. Den första stabila utgåvan av Android släpptes år 2008 (Zechner 2011, s. 2) och sedan dess har det släppts ett flertal nya versioner. Den senaste större uppdateringen var version 4.2 som lanserades i slutet av år 2012 (The Verge 2012). Android är under aktiv utveckling och operativsystemet har fått påtagliga förändringar sedan de tidigaste versionerna. Vid jämförelse av exempelvis version 1.5 och 4.2, illustrerade i figur 2 respektive figur 3, märks tydliga förändringar i användargränssnittet.

Figur 2: En bild av Android Emulators förvalda hemskärm i Android 1.5. Bildkälla: Unamed102 u.å. CC (BY SA 3.0)

Figur 3: En bild av Androids hemskärm i den senaste versionen av Android, 4.2. Bildkälla: Android Developers 2012 CC (BY 2.5)

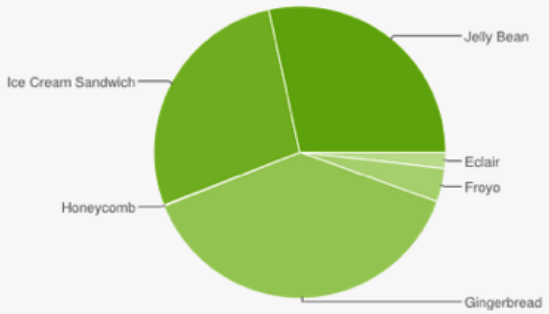
Varje version av Android har en bestämd *API-nivå* som anger vilken version av Androids API den stödjer (Android Developers 2013 #9). Denna API-nivå är till för att utvecklare på ett enkelt sätt ska kunna ange vilken version applikationen stödjer och anpassa applikationen därefter.

En Androidapplikation utvecklas alltid mot en lägsta API-nivå, vald av utvecklaren. Beroende på API-nivå tillhandahålls olika mycket funktionalitet, där högre API-nivå ger mer inbyggd funktionalitet (Android Developers 2013 #8). Däremot når applikationen

ut till fler användare ju tidigare version som väljs. Detta eftersom enheter med lägre API-nivå än den som stöds inte kommer att kunna installera applikationen. Detta bortfall kan bestå av en betydande mängd enheter eftersom de Androidenheter som idag finns på marknaden kör vitt spridda versioner av Android. Denna spridning visas av figur 4 för perioden april till maj år 2013.

Utvecklare uppmanas att stödja tidiga Androidversioner för att minska det potentiella bortfallet av användare. Det rekommenderas att applikationer dynamiskt anpassar sig efter Androidversionen på enheten (Android Developers 2013 #17). På så sätt ges tillgång till extra funktionalitet för senare versioner, utan att tidigare versioner utesluts.

Version	Codename	API	Distribution
1.6	Donut	4	0.1%
2.1	Eclair	7	1.7%
2.2	Froyo	8	3.7%
2.3 - 2.3.2	Gingerbread	9	0.1%
2.3.3 - 2.3.7		10	38.4%
3.2	Honeycomb	13	0.1%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	27.5%
4.1.x	Jelly Bean	16	26.1%
4.2.x		17	2.3%



Figur 4: Bild som visar fördelningen av Androidversion i aktiva enheter för perioden april-maj 2013 Bildkälla: Android Developers 2013 #1 CC (BY 2.5).

Nya versioner av operativsystemet medför ofta omfattande förändringar, vilket kan innebära att gamla principer överges. Utvecklare behöver ta hänsyn till detta, dels under den inledande utvecklingen men också vid uppdateringar av applikationen om denna ska följa nya principer (Android Developers 2013 #16). I samband med förändringar försöker dock Android, i möjligast mån, bevara bakåtkompatibilitet. Detta görs antingen genom bibliotek ämnade för att stödja äldre versioner eller genom att äldre applikationer körs i ett kompatibilitetsläge (Android Developers 2013 #9). Med bakåtkompatibilitet menas att en ny version av Android ska kunna köra applikationer som från början utvecklats för äldre versioner av operativsystemet. Detta är nödvändigt för att undvika för stor fragmentering, då utvecklingen av Android sker i hög takt (McFadden 2009).

Den första versionen av Android utvecklad för surfplattor, version 3.0, släpptes första halvan av år 2011 (Ducrohet 2011). Under den senare halvan av år 2011 släpptes version 4.0, med utökat stöd för surfplattor (Balolong 2011).

2.2.5. Androidenheter

I början av Androids utveckling var stödet för enheter begränsat enbart till smarttelefoner (Komatineni och MacLean 2012, s. 4). Med version 1.6 av Android kom stöd för enheter med olika upplösning och skärmstorlek, (Android Developers 2013 #18) och numera stöds även enheter som exempelvis surfplattor och TV-apparater (Android Developers 2013 #11; Android Developers 2013 #12).

Idag finns en mängd olika typer av Androidenheter. Enheterna har varierande skärmstorlek, form, upplösning, inmatningssätt med mera (Felker 2012). Android är tekniskt helt oberoende av vilken typ av enhet det körs på. Istället tar operativsystemet hänsyn till den hårdvara enheten består av. Exempelvis tas hänsyn till bland annat enhetens skärmupplösning och skärmstorlek. Operativsystemet skiljer inte på exempelvis telefoner eller surfplattor, utan märker enbart differenser i hårdvara (Android Developers 2013 #10).

Det finns dock regler för vad en Androidenhet måste stödja för att operativsystemet ska fungera korrekt. Exempel på detta är minimikrav på processorhastighet och minne. En Androidenhet måste även tillhandahålla någon sorts nätverksanslutning, antingen genom trådbunden eller trådlös teknik (Android Compatibility Program 2013).

2.3. Androidapplikationer

Applikationer i Android skiljer sig i både uppbyggnad mot traditionella Javaprogram. I Android är exempelvis källkod och tillhörande resurser paketerade i ett eget filformat kallat *application package file* (APK), istället för Javas motsvarande filformat JAR. Denna APK-fil innehåller allt som krävs för att installera en applikation (Android Developers 2013 #4).

Android har stöd för att installera applikationer utöver de som medföljer i operativsystemet. Dessa kan utöka enhetens funktioner och tillföra mervärde för användaren (Android Developers 2013 #4). Det finns flera sätt att installera applikationer, bland annat genom Googles applikationsmarknadsplats. Det går även att installera en applikation genom manuell nedladdning och exekvering av dess APK-fil (Whitwam 2012).

2.3.1. Activity

Activity är en av Androids mest grundläggande byggstenar. Det är den applikationskomponent som hanterar arbetsytan på skärmen som användaren kan interagera med, som visar information och använder inmatning. Varje Activity ges ett

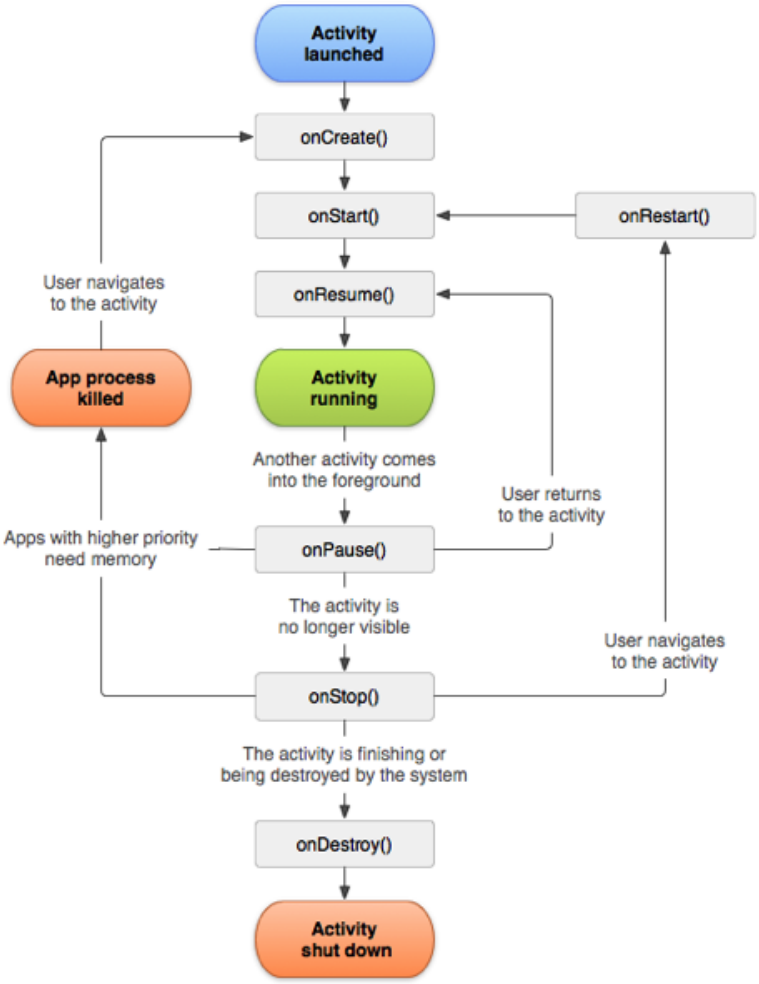
eget fönster som ofta tar upp hela skärmen och används för att rita upp användargränssnittet (Android Developers 2013 #3).

Varje applikation med ett användargränssnitt implementerar åtminstone en Activity, men vanligtvis flera. Dessa implementationer skapar tillsammans det som kallas en Androidapplikation. Samtidigt kan varje Activity köras oberoende av alla andra, vilket är en grundprincip i Android. Tanken är att detta ska underlätta för applikationer att använda sig av varandra (Android Developers 2013 #4; Android Developers 2013 #20).

Varje Activity har en egen livscykel. Denna livscykel är indelad i sju olika skeden, som vardera representeras av en specifik metod i Activityklassen. Dessa skeden listas nedan och deras kopplingar illustreras i figur 5.

- Skapande hanteras av metoden onCreate och medför att Activityinstansen påbörjar sin livscykel. Grundläggande kopplingar, framförallt till vyrepresentationen, etableras här.
- Start hanteras av metoden onStart och sker när instansen syns på skärmen.
- Omstart hanteras av metoden onRestart och sker när användaren navigerar tillbaka till applikationen efter att ha stängt ner den.
- Återupptagning hanteras av metoden onResume och sker när instansen är i förgrunden.
- Pausande hanteras av metoden onPause och sker när instansen hamnar i bakgrunden.
- Avslut hanteras av metoden onStop och sker när instansen försvinner från skärmen.
- Nedbrytande hanteras av metoden onDestroy och medför att instansen avslutar sin livscykel.

Genom att dessa metoder överskuggas kan en applikations beteende vid dessa skeden styras (Android Developers 2013 #3).



Figur 5: Illustration som visar livscykeln för Activity.
Bildkälla: Android Developers 2013 #3

2.3.2. Fragment

Fragment är ett komplement till Activity, som skapades för att förbättra både Androids modularitet och stöd för flera olika skärmstorlekar. Dess funktionalitet liknar Activity. Båda används för att presentera och hantera information för användaren. Skillnader är att ett Fragment inte representerar en hel arbetsyta, utan används inom en Activity. Ett Fragment är direkt knutet till den Activity den används inom, och kan dynamiskt ta upp en del av dess yta. Hur stor yta som tas upp beror på olika förutsättningar definierade i applikationen (Android Developers 2013 #19). Exempelvis kan en Activity som innehåller två Fragment antingen visa dessa bredvid varandra eller ett i taget beroende på enhetens skärmstorlek. Användargränssnittet varierar alltså beroende på skärmstorlek trots samma källkod.

2.3.3. Intent

Intent är en Androidkomponent som innehåller information för en operation som önskas utföras. Komponenten används för samarbete och informationsutbyte, och används i stor omfattning av Android för att starta instanser av Activity. Intent fungerar som en komponent som håller samman två Activityinstanser, vilket tillåter applikationer med mer än en Activity, samt interagering mellan olika applikationer (Android Developers 2013 #20).

2.3.4. Broadcast Receiver

Applikationer i Android behöver få tillgång till information från operativsystemet för att fungera korrekt. Denna information får applikationen med hjälp av komponenten *Broadcast Receiver* (Vogella 2013). Komponentens enda funktion består i att lyssna efter meddelanden, från operativsystemet eller applikationer, och därefter agera utifrån dessa meddelanden (Android Developers 2013 #4).

2.3.5. Content Provider

När en applikation behöver hämta data från andra applikationer eller från operativsystemet kan *Content Provider* användas. Content Provider hanterar tillgång till strukturerad data och möjliggör att data kan delas mellan olika applikationer i Android (Android Developers 2013 #5). En applikation kan därmed lagra sin data i filsystemet, och en Content Provider möjliggör åtkomst och modifiering av datan för en annan applikation. Exempelvis har Androids system en Content Provider som hanterar all kalenderdata för en användare (Android Developers 2013 #22).

2.3.6. Service

Service är en komponent som körs i bakgrunden utan något användargränssnitt (Android Developers 2013 #7). Det huvudsakliga användningsområdet för Service är att utföra uppgifter som sker under lång tid. Om en nedladdning körs som en Service, till exempel, fortsätter den i bakgrunden även medan användaren byter mellan olika applikationer.

En Service startas av en extern komponent, exempelvis en Activity eller en Broadcast Receiver, vilken den även kan kommunicera med (Android Developer 2013 #7). Noterbart är att Service körs i Androids huvudtråd. Därför rekommenderar Android att en bakgrundstråd används för krävande uppgifter för att inte låsa operativsystemet (Android Developers 2013 #21).

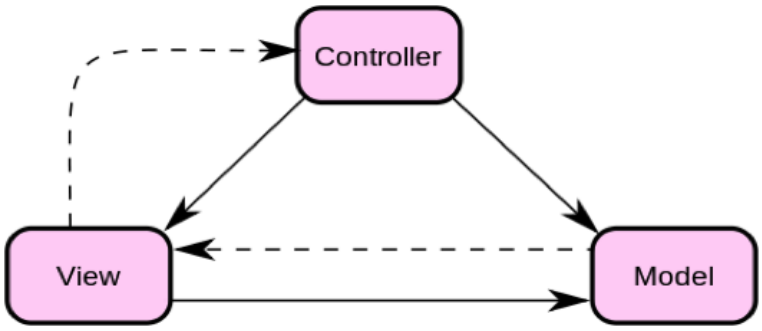
2.4. Designmönster

För att hålla hög kvalitet på ett mjukvaruprojekts källkod, samt förenkla framtida utbyggnad och underhåll, kan välbeprövade designmönster användas. Genom att använda designmönster inkapslas olika delar av källkoden. Samtidigt förblir källkodsbasen hanterbar även när projektets storlek ökar (Martin 2000).

2.4.1. Model-View-Controller

Ett välkänt designmönster är *Model-View-Controller* (MVC). Det lämpar sig väl vid skapandet av applikationer med ett användargränssnitt eftersom MVC delar upp källkoden som hanterar presentation, datahantering samt inmatning. MVC är mer omfattande än många andra designmönster, och påverkar hela applikationens arkitektur (Microsoft 2013 #3).

MVC-mönstret delar upp applikationen i tre olika delar: Modell, Vy och Kontroller. I modellen återfinns data samt logik för manipulering av data. Vyn sköter grafisk presentation av applikationen. Vyn meddelas av modellen när någon del har förändrats och uppdateras efter applikationens nuvarande tillstånd. Kontrollern interagerar med vyn och modellen, till exempel med kommandon om att data ska uppdateras (Microsoft 2013 #3). Figur 6 illustrerar hur delarna är kopplade till varandra, där streckade pilar representerar svagare länkar än hela pilar.



Figur 6: Illustration av strukturen hos MVC. Bildkälla: Wikimedia Commons 2010

Genom att dela upp applikationen i dessa delar erhålls en tydlig strukturering av källkoden, där varje del har ett specifikt ansvarsområde. Samtidigt blir det enklare att återanvända delar av källkoden. Exempelvis kan modellen i en applikation återanvändas vid skapandet av ett nytt användargränssnitt.

2.4.2. Singleton

Där det är viktigt att endast ha en instans av en klass kan designmönstret Singleton användas (OODesign u.å #1). En klass som implementerar Singleton sköter sin egen instansiering och har endast privata konstruktorer. Genom detta kan denna klass kontrollera om en instans redan existerar, i vilket fall denna returneras istället för att en ny skapas. I och med att ingen publik konstruktor finns, garanteras det att det enbart finns en instans av klassen (OODesign u.å #1).

2.4.3. Observer

Observermönstret är ett designmönster som hanterar övervakning av objekt (OODesign u.å #2). Ett eller flera objekt agerar som Observer, och lyssnar till förändringar i det objekt som är Observable. Dessa Observerobjekt underrättas om en förändring sker i det Observableobjekt de lyssnar på. Observerobjekten kan då uppdateras utifrån förändringarna. Användning av observermönstret medför separation av källkod samt skapar lösa kopplingar mellan de objekt som observerar och de som observeras.

2.5. Google Apps

Google Apps är ett molnbaserat grupprogram som hanterar bland annat mejl och kalendrar (Morel, et al. 2011). Kalenderfunktionen i Google Apps erbjuder rumsbokning via ett webbgränssnitt kallat *Google Calendar* (Google Support 2013 #1). Dessutom existerar en applikation för Android vid samma namn (Google Play u.å.). Denna applikation erbjuder hantering av kalendrar och händelser till andra applikationer på samma enhet med hjälp av en ContentProvider.

Rumsbokningen möjliggörs genom att varje rum tilldelas en särskild kalender. Kalenderfunktionen hanterar de kollisioner som uppstår om flera användare parallellt försöker boka samma rum (Google Support 2013 #2). Tjänsten hanterar även tidszoner och skickar ut uppdateringar till enheter som prenumererar på kalendern (Google Apps for Business u.å).

2.6. Googles API

För många av de tjänster Google erbjuder tillhandahålls ett API som ger tillgång deras funktioner. För att använda detta API behöver projektet först registreras genom Googles API-konsol (Google Developers 2013 #4).

Googles API är webbaserat; all dataöverföring sker med hjälp av dataöverföringsprotokollet *Hypertext Transfer Protocol* (HTTP). Datan som skickas

formateras som objekt i enlighet med dataöverföringsstandarden *Javascript Object Notation* (JSON). (Google Developers 2013 #2).

2.6.1. Autentisering

För att läsa och manipulera data via Googles API måste en autentiseringsprocess genomgås. Denna process använder den nyare versionen av autentiseringsprotokollet *OAuth* (Google Developers 2013 #1).

OAuth är en standard som beskriver en autentiseringsprocess där ett system behöver tillåtelse från en användare för att komma åt data som ett annat system tillhandahåller. Processens syfte är att användare ska slippa lämna ifrån sig sina inloggningsuppgifter. För användaren ska processen vara så enkel så möjligt, helst en enda knapptryckning där samtycke medges (Leiba 2012 s. 74-77).

Vid kommunikation med Googles API används version två av OAuth, kallad *OAuth2*. Till Android erbjuder Google ett lättanvänt stöd för hantering av de autentiseringsnycklar som krävs av OAuth2, känt som *Google Play Services* (Android Developers 2013 #15). Genom användning av denna stödtjänst kan all hantering av inloggningsuppgifter överlämnas till tjänsten.

När en Androidapplikation behöver åtkomst till data genom Googles API kan applikationen be operativsystemet generera en nyckel. Google Play Services kräver i sin tur en verifikation från användaren för att tillåta detta. Denna verifikation sker i form av en dialogruta som tar över skärmen, berättar för användaren att applikationen försöker kommunicera med användarens konto och begär en bekräftelse. När detta är gjort produceras en tidsbegränsad nyckel som applikationen kan använda för att kommunicera med Googles API (Android Developers 2013 #6). Dessutom krävs det att projektet byggs med en oföränderlig krypteringsnyckel. Denna krypteringsnyckel genereras genom en webbtjänst som Google tillhandahåller (Google Developers 2013 #4).

2.6.2. Javabiblioteket

Som komplement till Googles webbaserade API erbjuds ett Javabaserat bibliotek. Biblioteket kan hantera en uppkoppling till Googles API och all dataöverföring till denna. Det abstraherar dessutom användandet av detta API, eftersom program som implementerar biblioteket inte behöver hantera JSON-objekt. Google har även gett ut ett Javabibliotek specifikt för utveckling mot Android (Google Code 2013).

Anmärkningsvärt med det Javabibliotek som finns för Googles API är att det är automatiskt genererat. Detta är gjort för att en stor mängd Googletjänster ska kunna

erbjuda liknande gränssnitt. Automatiseringen kan dock försvåra gränssnittets användning eftersom dokumentation blir bristfällig (Google Developers 2013 #3).

2.7. Arbetsverktyg

Det finns olika tjänster för att underlätta arbete med utvecklingen av en Androidapplikation. Bland annat finns *Balsamiq Mockups*, som underlättar grafisk design och stöds av dokumenthanteringssystemet *Google Drive*. För källkodsutveckling finns utvecklingsmiljöer som *Eclipse* såväl som Googles utvecklingspaket för Androidutveckling, *Android Developer Tools* (ADT). Vidare kan arbete med källkod förenklas genom användandet av ett versionshanteringssystem som *Git*.

2.7.1. Google Drive

Google Drive är en kostnadsfri molntjänst erbjuden av Google som tillhandahåller webbaserad dokument- och filhantering. Utöver ordbehandlare finns verktyg för kalkylblad, presentationer och en mängd tredjepartsapplikationer (Google Drive u.å.). bland annat Balsamiq Mockups.

2.7.2. Balsamiq Mockups

Balsamiq Mockups är ett program för att skapa digitala skisser av användargränssnitt där navigation mellan skapade skisser är möjligt. Programmet har stöd i Google Drive vilket tillåter användare att dela skisser med varandra. Till programmet finns modeller av gränssnittskomponenter för olika operativsystem. Dessa modeller kan användas för att underlätta skapandet av skisser (Balsamiq u.å.).

2.7.3. Eclipse

Eclipse är en mjukvaruutvecklingsmiljö med stöd för flera operativsystem, bland andra Windows, Linux och OS X. Främst används Eclipse för Javautveckling, men det stödjer även andra programmeringsspråk (Eclipse u.å.).

Eclipse går att utvidga genom av insticksmoduler vilka möjliggör funktionalitet utöver den grundläggande källkodshanteringen (Eclipse Marketplace 2013 #3). Bland annat finns insticksmoduler för att rita UML-diagram och genomföra tester av källkod (Eclipse Marketplace 2013 #1; Eclipse Marketplace 2013 #2).

2.7.4. Git och Github

Versionshanteringssystemet Git används vid mjukvaruutveckling för att förenkla samarbete kring källkod. Med Git ges möjlighet att dela arbete i grenar för att kunna utveckla olika delar parallellt (Github 2013 #2). Utöver att finnas som ett

terminalprogram finns det även grafiska användargränssnitt för Git, som programmet *Github*. Vidare erbjuds även ett webbgränssnitt med samma namn för användare, i vilket uppgifter kan beskrivas och medarbetare kan åta sig att lösa dem. Dessa uppgifter kan också ordnas systematiskt med hjälp av milstolpar.

2.7.5. Android Developer Tools

Vid Androidutveckling krävs åtminstone tillgång till en textredigerare samt Androids *Software Development Kit* (SDK), vilket är en samling bibliotek och verktyg för utveckling. Google har dessutom släppt en uppsättning mer omfattande verktyg som underlättar utvecklingsarbetet, ADT (Android Developers 2013 #23).

ADT är en insticksmodul till Eclipse, som anpassar utvecklingsmiljön för Androidutveckling (Android Developers 2013 #2). Till exempel finns det ett verktyg som möjliggör för utvecklaren att designa användargränssnitt genom grafiskt uppritning. ADT integrerar också uppdaterad dokumentation för Androidspecifika klasser direkt i utvecklingsmiljön (Android Developers 2013 #23).

2.8. Användargränssnitt

För att åstadkomma kommunikation mellan system och användare kan ett grafiskt användargränssnitt utformas (Tidwell 2011, s. 1). Ett väldesignat grafiskt användargränssnitt tillfredsställer användarnas behov och kan uppnås genom att andra designer efterliknas. Genom användning av mönster som användaren känner igen skapas trygghet. Vid design av ett grafiskt användargränssnitt används grafiska designmönster som ett verktyg för att dela upp vanliga situationer och lösningar i kategorier. Det finns en mängd vedertagna grafiska designmönster som kan användas för att öka användbarheten (Tidwell 2011, s. xvi-xviii).

Beroende på vilken typ av enhet som applikationen körs på lämpar sig olika grafiska designmönster (Tidwell 2011, s. xiii). Design av grafiska användargränssnitt för enheter med pekskärm är fundamentalt olikt den för enheter som använder sig av tangentbord och mus (Tidwell 2011, s. 442).

3. Metod

Detta kapitel beskriver den arbetsprocess genom vilken projektet genomfördes. Arbetet utgick från den projektbeskrivning som erhållits från Understandit. Processen beskrivs i kronologisk ordning, med början i den inledande kravanalysen och dess tillhörande enkätundersökning. Vidare beskrivs utvecklingen av användargränssnittet, applikationsutvecklingen i helhet och de användartester som utfördes på den färdigställda prototypen.

Under hela projektets gång hölls möten med Understandit. Avsikten med dessa möten var att Understandit skulle ges möjlighet att påverka projektets utfall. De tog en rådgivande roll, samt försåg projektet med hårdvara för att förenkla utvecklingen. Totalt försågs projektet med fyra surfplattor av tre olika modeller med vitt skilda specifikationer, det vill säga olika skärmstorlek, pixeltäthet och prestanda.

3.1. Kravspecificering

Kravspecificeringen inleddes med att Android-version valdes. Stor vikt lades på detta val i och med att det dikterade vilken funktionalitet som finns inbyggd i Android samt hur många användare applikationen kan nå ut till.

En enkätundersökning bestående av slutna frågor distribuerades till potentiella användare i Understandits systerbolag. Dessa bolag använde redan ett liknande rumsbokningssystem och de potentiella användarna hade därmed redan introducerats till grundkonceptet. Syftet med enkäten var att ge underlag till vilka funktioner som skulle ingå i Resourcee. Funktionernas inbördes prioritering diskuterades sedan tillsammans med Understandit. Dessutom bestämdes vilka av dessa funktioner som var helt nödvändiga i prototypen som skulle utvecklas.

3.2. Användargränssnitt

Parallellt med genomförandet av enkäten togs en klickbar prototyp fram med hjälp av programmet Balsamiq Mockups. Användargränssnittet utvecklades iterativt, och prototypen visades upp för Understandit vid flera tillfällen under projektets gång. Google Drives inbyggda stöd för Balsamiq Mockups användes för samarbete i utvecklingen av användargränssnittet.

3.3. Applikationsutveckling

Mjukvaran programmerades i en iterativ process och testades kontinuerligt. Denna arbetsmetod var tänkt att ge projektet en större flexibilitet och göra det lättare att förbättra systemet under projektets gång. Arbetsmetoden skulle även öka Understandits möjlighet att testa mjukvaran och därmed påverka slutresultatet.

Implementationen av Resourcee inleddes med design av systemets underliggande arkitektur. Denna design kom dock att revideras flera gånger under utvecklingsprocessen som ett led i det kontinuerliga förbättrandet av applikationen.

Utvecklingen använde sig av de verktyg som ingår i ADT, samt Androids SDK. Som utvecklingsmiljö användes därför Eclipse med ADT integrerat som insticksmodul. Källkoden hanterades med versionshanteringssystemet Git och en central version av källkoden lagrades genom webbtjänsten Github.

När källkod skrevs kommenterades den fortlöpande, detta för att den skulle vara förståelig för någon som önskade sätta sig in i Resourcee. Det beslutades tidigt att utförliga kommentarer skulle vara av stor vikt, i och med att Understandit gavs tillgång till den centrala versionen av källkoden. Genom god dokumentation skulle behovet av fortlöpande förklaringar av källkodsförändringar minskas. Det faktum att Understandit genom Github hade tillgång till projektets versionshistorik underlättade även kommunikationen. Github användes även för att förenkla den iterativa arbetsprocessen genom beskrivning och tilldelning av uppgifter. Dessa problem delades sedan in i milstolpar och dessa milstolpar gavs slutdatum.

3.4. Användartest

Efter färdigställandet av Resourcee utformades ett användartest för att utvärdera applikationens användargränssnitt. Testet utfördes på fyra personer från Understandit. Testpersonerna hade varierande erfarenhet av rumbokningsapplikationer. Detta var tänkt att ge ett resultat innehållande flera infallsvinklar. Testet genomfördes genom att deltagarna erhöll en surfplatta med Resourcee uppstartad och fick utföra sju uppgifter.

Under varje uppgift fick användaren reflektera högt om applikationen och sedan berätta hur utförandet upplevdes, samtidigt som testledaren antecknade. Detta för att få fram information om något i applikationens användargränssnitt inte var tydligt nog. Efter att användartestet utförts gavs testpersonerna utrymme att kommentera om upplevelsen av användargränssnittet och om något saknades. För detaljerad testplan se bilaga F.

4. Kravanalys och grafisk design

Följande kapitel beskriver det arbete som har gjorts för att sätta upp Resourcees kravspecifikation och användargränssnitt. Även några av projektets begränsningar beskrivs i detta kapitel: den Androidversion mjukvaran utvecklats för samt vilka funktioner som har bortprioriterats.

4.1. Enkätundersökning

Ett av verktygen som användes för kravspecifikationen till Resourcee var resultatet från den genomförda enkätundersökningen. Enkäten bestod av ett antal slutna frågor och besvarades av 22 av de 60 tillfrågade personerna. För enkätundersökningens frågor och resultat se bilaga D respektive E.

Resultatet från enkätundersökningen understödde att det fanns en efterfrågan på den funktionalitet som Understandit föreslog. Majoriteten av respondenterna, 95 %, efterfrågade att på plats, via applikationen, kunna se om det representerade rummet är ledigt. Vidare önskade 91 % möjligheten att kunna boka rummet, och 86 % uppgav att de skulle uppskatta en möjlighet att avsluta bokningar i förtid. Dessutom visade resultatet att önskemål om funktioner utöver Understandits förslag fanns.

Viss önskad funktionalitet från respondenterna har dock inte gått i enighet med Understandits ursprungliga instruktioner. För att hålla Resourcee lättanvänd beslutades att enbart direktbokningar skulle tillåtas, vid övriga tillfällen skulle det befintliga bokningssystemet användas. Enkätresultatet visar dock att 95 % av respondenterna ville kunna utföra även framtida bokningar. Dessutom önskade 45 % att ha möjlighet att boka andra rum direkt från Resourcee.

4.2. Funktionella och icke-funktionella krav

Funktionella och icke-funktionella krav ställs på funktioner som ska finnas hos mjukvaran, samt generella villkor som ska uppfyllas av applikationen som helhet. De funktionella och icke-funktionella kraven bröts ner till *användningsfall*, vilka är beskrivningar av flödet hos en funktion (Larman 2005, kap 6).

Innan implementation påbörjades erhölls förslag på funktioner till Resourcee från Understandit. Med hjälp av dessa specificerades funktionella och icke-funktionella krav. Dessutom utökades kraven, delvis med grund i enkätundersökningen och delvis för att få differentiering mot konkurrerande produkter.

Projektets uppdragsgivare lämnade en projektbeskrivning, bilaga A, som innehöll följande funktionella krav:

- Det ska tydligt indikeras om ett rum är ledigt på skärmen.
- Bokning av rum skall möjliggöras från surfplattan.
- Dagens bokningar ska visas i en kalendervy.
- Integration med Exchange och Google Apps ska finnas.
- En extern statusindikator ska finnas.

Med utgång från dessa krav, och funktionalitet som utformats i projektets planeringsstadium, skapades en komplett lista funktionella och icke-funktionella krav. För hela listan, se bilaga B och C.

4.3. Fastställande av funktioner

Utifrån de funktionella och icke-funktionella kraven togs 18 användningsfall fram, vilka kan ses i bilaga H. Användningsfallen rangordnades efter hur viktiga de var för det mest grundläggande utförandet av Resourcee. För att ta hänsyn till de potentiella slutanvändarna togs enkätundersökningens resultat i beaktande vid prioriteringen av användningsfall.

Utöver inbördes rangordning delades användningsfallen upp i tre prioritetskategorier: hög, mellan och låg. Hög prioritet fick de användningsfall som bedömdes vara nödvändiga för att applikationen skulle kunna utföra samma huvudsakliga uppgifter som konkurrerande lösningar. I mellankategorin placerades de fall som skulle tillföra ytterligare funktionalitet gentemot konkurrenter. Låg prioritet tilldelades de användningsfall som representerade möjlig framtida funktionalitet. Denna funktionalitet kunde utvecklas om samtliga högre prioriterade användningsfall redan var färdigställda. Deras utveckling ansågs osannolik från början, på grund av projektets tidsram.

4.4. Grafisk design

Tidigt under utvecklingen ställdes krav på designen av det grafiska användargränssnittet. Eftersom interaktionen mellan systemet och användare ska ske på en pekskärm ska ett responsivt och lättanvänt användargränssnitt eftersträvas. Designen ska vara såpass tydlig att även nya användare ska kunna förstå systemet. Eftersom Resourcee ska användas på surfplattor behövde designvalen göras därefter. Tidigt i utformningen framkom att fem grafiska designmönster var väsentliga för att åstadkomma ett lätthanterligt och tydligt användargränssnitt.

Det första designmönstret var *Clear Entry Points* som uppfylls genom att presentera ett fåtal tydliga alternativ på startskärmen. På så sätt skapas ett intuitivt och lättöverskådligt användargränssnitt. Det ger användaren trygghet att ha alternativ som överensstämmer med dennes förväntningar (Tidwell 2011, s. 87-89).

För att göra användargränssnittet än mer lättöverskådligt beslutades att eftersträva en enhetlig utformning av Resourcee, det vill säga ett *Visual Framewor*k. Detta medför att användaren upplever det enkelt att navigera runt i applikationen. Den generella strukturen leder till att användaren inte behöver lägga tid på att förstå varje vy på nytt (Tidwell 2011, s. 141-145).

För att användare skulle våga utforska Resourcee skulle även designmönstret *Escape Hatch* användas. Detta används för att hjälpa användaren tillbaka till en bekant sida genom en tydlig väg bort från varje vy (Tidwell 2011, s. 104-106).

Designmönstret *Button Group* ansågs också lämpligt. Detta för att användargränssnittet ska underlätta för användaren att hitta den knapp som söks. Mönstret uppnås genom att gruppera knappar med liknande aktiviteter intill varandra (Tidwell 2011, s. 246-248).

Ett annat designmönster som valdes för att tillämpas på knappar var *Prominent 'Done' Button*. Detta tydliggör för användaren att en aktivitet avslutas genom en specifik knapp. Detta uppfylls ofta genom att göra knappen stor och framträdande med tydlig text (Tidwell 2011, s. 257-261).

5. Resultat

Detta kapitel inleds med att i stora drag återge den prototyp som projektet resulterat i. Resourcees utvecklade användargränssnitt och systemdesign beskrivs först övergripande, vilket följs av en detaljrik genomgång av vissa implementationsval. Slutligen ges även resultatet av det användartest som genomfördes efter projektets färdigställande.

5.1. Prototyp

Projektet resulterade i en applikation anpassad för Android. Inga andra system implementerades, utan systemet använder sig av Android och Google Apps för att utföra den funktionalitet som Resourcee själv inte omfattar, konto- och databashantering. Applikationen distribueras elektroniskt genom överföring av en APK-fil till de enheter som ska ingå i systemet.

Resourcee är utvecklad för Androidenheter som kör Android 4.0.3 eller en senare version. Detta beslut grundade sig i att de tidigare versionerna av Android med stöd för surfplattor, 3.0 och 4.0.0, endast används av en marginell andel enheter. Att stödja version 4.0.3 innebär att Resourcee är kompatibel med majoriteten av surfplattor på marknaden.

Nio av applikationens arton ursprungliga användningsfall har implementerats, vilket kan ses i tabell 1. Bland de implementerade användningsfallen finns krav på tydlig statusindikation, synkronisering med Google Apps och bokningar. Även funktionalitet för förlängning och avslutning av pågående bokningar har implementerats. För detaljerade användningsfall se bilaga H.

Tabell 1: Framtagna användningsfall, deras prioritet och status. För detaljerade användningsfall se bilaga H.

Användningsfall	Tilldelad Prioritet	Status
Konfigurera applikationsinställningar	1. Hög	Implementerat
Kontrollera ett rums status	2. Hög	Implementerat
Boka detta rum (omedelbart)	3. Hög	Implementerat
Förläng en pågående bokning	4. Hög	Implementerat
Avsluta en pågående bokning	5. Hög	Implementerat
Se information om en bokning	6. Hög	Implementerat
Se framtida schema	7. Mellan	Implementerat
Hitta lediga rum i närheten till plattan	8. Mellan	Ej implementerat
Boka andra rum	9. Mellan	Ej implementerat
Kontrollera information om ett rum	10. Mellan	Implementerat
Felanmäla	11. Mellan	Ej implementerat
Ändra språk	12. Mellan	Implementerat
Logga in	13. Låg	Ej implementerat
Logga ut	14. Låg	Ej implementerat
Boka detta rum (framåt i tiden)	15. Låg	Ej implementerat
Avboka detta rum i framtiden - inloggad	16. Låg	Ej implementerat
Ändra en framtida bokning	17. Låg	Ej implementerat
Se hjälpavsnitt	18. Låg	Ej implementerat

Samtliga användningsfall som tilldelades hög prioritet har färdigställts, vilket tabell 1 visar. Nuvarande status på Resourcee medför att vem som helst kan konfigurera applikationen, detta eftersom ingen funktionalitet för användarautentisering har utvecklats.

Resourcees konfiguration grundar sig på att de som ämnar använda sig av applikationen använder Google Apps för kalender- och rumshantering. Det krävs också att det existerar ett tjänstkonto i operativsystemets kontohantering. Detta tjänstkonto är ett Googlekonto som prenumererar på de rumskalendrar som är aktuella för applikationen. Slutligen krävs att applikationen Google Calendar är installerad och att de aktuella rumskalendrarna synkroniseras genom denna applikation.

5.2. Systemdesign

Resourcee har designats med stor tillit till funktionalitet inbyggd i Android 4.0.3 såväl som i Google Play Services. Androids inbyggda funktioner används för hantering av tjänstkontot, samt för tidsuppdateringar vilka lyssnas efter genom en instans av Broadcast Receiver. Funktioner inbyggda i Google Play Services används för att hantera

den autentiseringsprocess som krävs för att komma åt kalendrar som tjänstkontot abonnerar på.

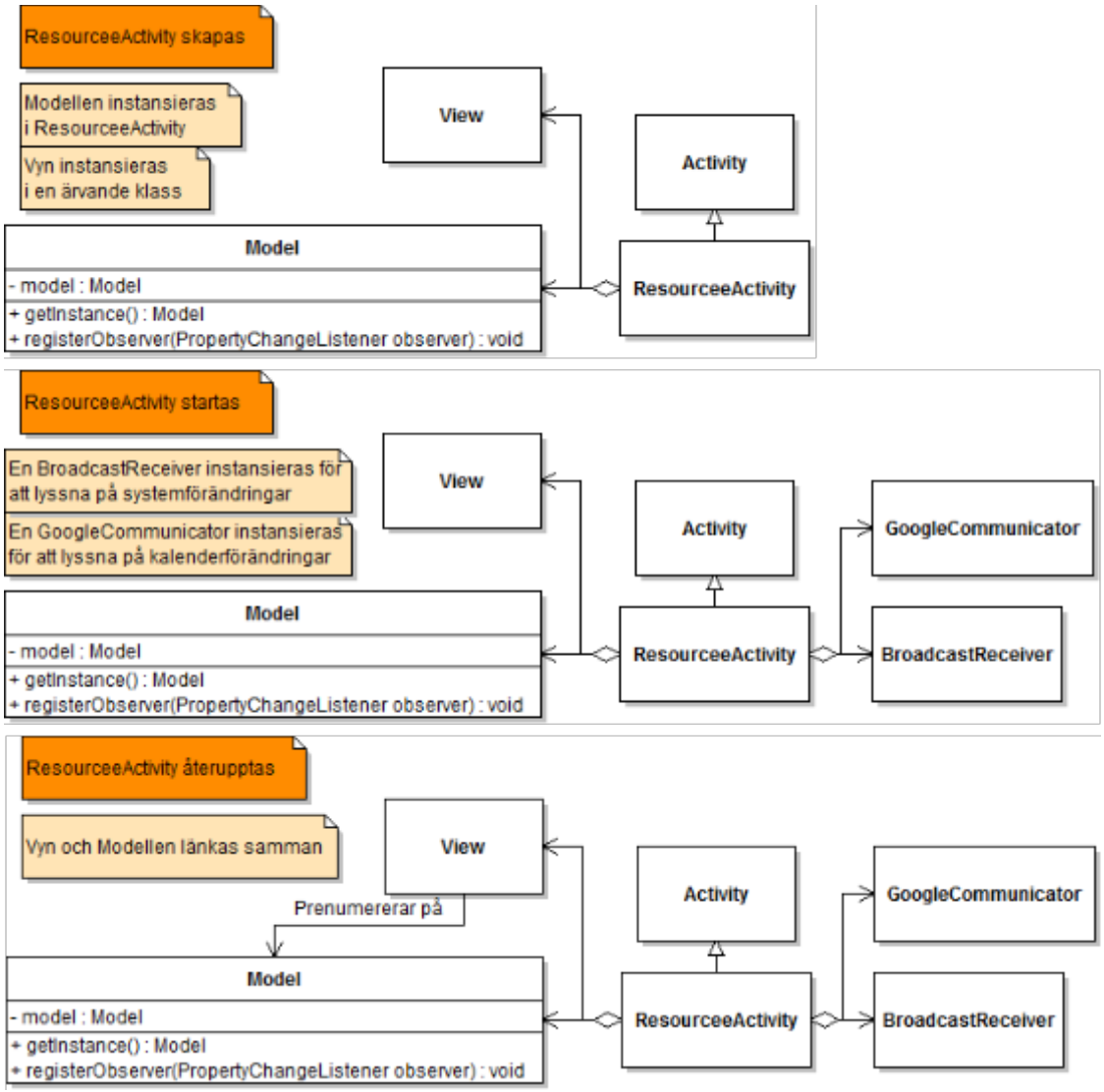
5.2.1. Övergripande implementation

Resourcee implementerar Model-View-Controller-mönstret (MVC), där de klasser som implementerar Activity agerar som kontroller. För att underlätta användandet av mönstret samlades en stor del av kontrollerfunktionaliteten i en abstrakt samlingsklass vid namn *ResourceeActivity*. ResourceeActivity innehåller en instans av ett vy-objekt såväl som ett modell-objekt. Denna abstrakta klass ärvs sedan av varje Activity i applikationen vilket upprätthåller MVC-mönstret och minimerar källkodsduplicering. För ett diagram över ResourceeActivity och dess ärvande klasser, se bilaga I.

ResourceeActivity innehåller även funktionalitet som underlättar ett byte från en av dess ärvande klasser till en annan, vilken kallas bytets målklass. Detta sker genom att metoden *changeActivity* tar målklassen som argument. Detta implementationsval grundar sig i att samtliga byten från en Activity till en annan ska ske på samma sätt. I metoden changeActivity skapas därför ett Intent i ResourceeActivity som beskriver det önskade utgångsläget för operativsystemet såväl som för nästa Activity.

Källkodsduplicering minimeras även genom att ResourceeActivity innehåller vissa procedurer som måste genomgås vid vissa tidpunkter i livscykeln hos en Activity som ingår i Resourcee, vilka listas nedan. Tre av dessa procedurer illustreras av figur 7.

- Vid skapande formar ResourceeActivity de mest grundläggande länkarna. Under denna procedur hämtas eventuella lagrade inställningar, samt en referens till modellen. Finns det ännu ingen instans av modellen skapas den här.
- Vid start upprättas en kontakt med operativsystemet som ser till att en uppdatering erhålls i denna ResourceeActivity varje gång systemets klocka tickar upp med en minut. Vidare undersöker denna ResourceeActivity om Google Play Services finns tillgängligt på enheten. Om så är fallet upprättas en kontakt med operativsystemets Content Provider för kalendrar, vilket vanligtvis sker genom applikationen Google Calendar.
- Vid återupptagning, samt även direkt efter en uppstart, registrerar ResourceeActivity sin vy med modellen.
- Vid paus avregistreras vyn ResourceeActivity hos modellen.
- Vid avslut bryter ResourceeActivity all kommunikation med Google Calendar.



Figur 7: Ett diagram över den uppstartsprocedur i tre steg som föranleder att en av Resourcees Activityinstanser körs och är förberett för användarinteraktion.

Var och en av dessa procedurer övergår efter färdigställande till den motsvarande procedur som finns inbyggd i Activity. Exempelvis övergår metoden onStart i ResourceeActivity till onStart i Activity. På så vis upprätthålls Androids egen programhantering, samtidigt som ResourceeActivity kan utöka den funktionalitet som finns i Activity.

För att Resourcee automatiskt ska startas samtidigt som enheten implementerades en klass kallad *BootupReceiver*. Denna klass är en förlängning av BroadcastReceiver och kan därmed lyssna på meddelanden från operativsystemet. I den metadata som följer med Resourcee vid installation deklareras att BootupReceiver vill lyssna efter meddelanden om uppstart. Därmed kommer sådana meddelanden mottagas av klassen

även om applikationen inte körs för tillfället. Meddelandetypen skickas av operativsystemet varje gång enheten startats. När detta meddelade tas emot startas applikationens Activity med en flagga som indikerar att det är en normal uppstart.

5.2.2. Externa beroenden

En av Resourcees mest grundläggande funktioner upprättas i uppstartsproceduren i ResourceeActivity: kontakt med Googles kalendertjänst. På grund av applikationens nära integrering med Googles tjänster krävs det att den senaste versionen av Google Play Services såväl som Google Calendar är installerade.

Google Play Services används för hantering av användaruppgifter och generering av autentiseringsnycklar. Google Calendar utnyttjas för snabba uppdateringar, i regel under en sekund, vid förändringar i kalendern. Dessa uppdateringar innehåller dock ingen information, utan påvisar bara att något har förändrats. Utöver detta lagras inte den unika identifieringssträng som krävs för att ändra en kalenderhändelse genom Googles API i Google Calendar. Dessa tillkortakommanden medför att kommunikation med applikationen Google Calendar endast används för att erhålla en notis när en förändring skett i aktuell kalender. När en notis erhålls söker Resourcee genom Googles API efter all relevant information.

För att upprätta kommunikation med Googles API använder sig Resourcee av det tillhandahållna Javabiblioteket. Biblioteket underlättar kommunikationen eftersom det abstraherar den process som dataöverföring till och från kalenderdatabasen innebär. Med hjälp av biblioteket kan en HTTP-förbindelse till Googles API startas genom ett enda metodanrop. Dessutom sker all hantering av JSON-objekt per automatik, serialisering såväl som deserialisering.

5.2.3. Nätverkskommunikation

Resourcee använder sig av Google Apps kalenderdatabas för fyra särskilda uppgifter, varav två förändrar databasen. Samtliga kräver att applikationen är konfigurerad med ett tjänstkonto. Tabell 2 beskriver dessa fyra uppgifter mer utförligt.

Tabell 2: De olika uppgifter som antingen förändrar eller inte förändrar databasen.

Uppgifter som ej förändrar databasen	**Uppgifter som förändrar databasen**
Hämtning av de rumskalendrar som tjänstkontot kommer åt.	Upprättande av en ny händelse i en given rumskalender.
Hämtning av de händelser som ingår i en given rumskalender.	Förändring av sluttiden av den pågående händelsen.

Att hämta rumskalendrar sker, jämfört med övriga uppgifter, förhållandevis sällan. En lista över samtliga kalendrar behövs endast när användaren konfigurerar Resourcee. Hämtning av händelser sker betydligt oftare, då detta utgör en del i den procedur som genomgås när en förändring upptäcks.

För att en händelse ska kunna skapas i en rumskalender måste denna ha ställts in under konfigurationsprocessen. Vidare måste rummet för tillfället vara ledigt, samt att det krävs en sluttid för den nya händelsen. Som starttid för händelsen används det nuvarande klockslaget. Att implementera stöd för att låta användaren välja starttid manuellt utelämnades medvetet i utvecklingsprocessen, då detta motverkar strävan att behålla Resourcee lättanvänd. Notera dock att applikationen tar full hänsyn till om dess representerade rum bokats genom Google Calendars webbgränssnitt.

För att stödja att en pågående bokning ska kunna avslutas, eller förlängas, har funktionalitet för att ändra sluttid implementerats. I båda dessa fall förändras endast händelsens sluttid. Genom att händelsen justeras istället för att tas bort existerar fortfarande bokningen i Googles kalenderdatabas. Eftersom händelsen fortfarande existerar kan dess uppgifter erhållas även i framtiden.

5.2.4. Flertrådning

Eftersom de fyra kommunikationsuppgifter som finns implementeras i varsin klass och exekveras i egna trådar blir trådsäkerhet en viktig fråga. Trådsäkerhetslogik implementeras därför i klassen *GAPIUpdateTask*, vilken ärvs av de fyra uppgiftsklasserna. Logiken tvingar fram ett köbeteende i det fall att två trådar skulle försöka exekvera flera kommunikationsuppgifter samtidigt. Inga uppgifter kan exekvera samtidigt med varandra. Samtliga uppgifter kan resultera i att Google Play Services kontaktas för generering av OAuth2-nycklar. Om två trådar skulle kontakta detta system samtidigt skulle Resourcee krascha.

När en kommunikationsuppgift utförs får inga byten av Activity ske utan byten måste vänta tills uppgiften är avslutad. Detta beror på att uppgiftens autentiseringsprocess kan behöva en användarverifiering, vilket är ett resultat av Google Play Services hantering av OAuth2-nycklar. När en sådan användarverifiering krävs kommer uppgiften att kalla på den Activity som skapade uppgiftsobjektet för att skapa nyckeln.

På grund av detta krav på en kvarvarande Activity finns trådsäkerhetslogik i ResourceeActivity som skjuter upp eventuella försök att byta Activity. Detta görs tills samtliga trådar som exekverar kommunikationsuppgifter har avslutats. När en sådan uppskjutning sker visas även en dialogruta för användaren. Denna dialogruta ber

användaren vänta tills samtliga bakgrundsprocesser avslutats, samtidigt som användarinteraktion förhindras.

Samtliga vyobjekt skapas i samma programtråd och alla förändringar i vyerna sköts av denna. Trots detta kommer majoriteten av all programlogik exekveras i andra trådar. Inte bara nätverkskommunikationen sker i separata trådar, parallell exekvering av källkod sker även i det fall där Resourcee tar emot information från en Content Provider eller Broadcast Receiver.

5.3. Användargränssnitt

Resourcees användargränssnitt designades till stor del utifrån de fastställda funktionernas rangordning. Funktioner med hög prioritet skulle vara enkla att använda och lättillgängliga för användaren. Målet var ett rent användargränssnitt som vägleder användaren genom de viktiga funktionerna.

5.3.1. Vyn ‘Ledigt’

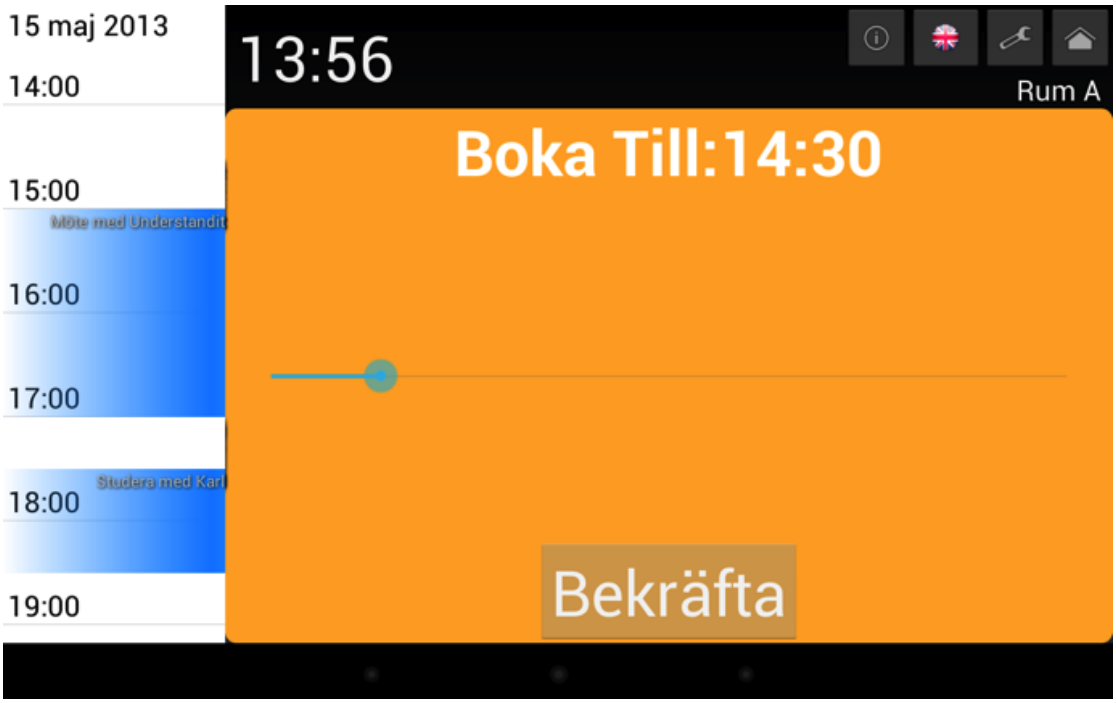
Är rummet som Resourcee representerar ledigt visas vyn ‘Ledigt’. Denna vy avbildas i figur 8. Att rummet är ledigt indikeras genom att bakgrunden är grön. Det finns en stor text som meddelar när nästkommande bokning börjar. Förekommer ingen bokning inom de nästa 24 timmarna berättar texten detta. För att göra en bokning används knappen ‘Boka’ i nederdelen av skärmen, vilken leder till vyn ‘Tidsinställning’

Figur 8: Skärmdump för ‘Rum A’ då rummet är ledigt.

5.3.2. Vyn ‘Tidsinställning’

När en ny bokning ska skapas eller en existerande bokning ska förlängas, görs detta i vyn ‘Tidsinställning’. Denna vy åskådliggörs i figur 9. För att tydliggöra att en bokning utförs är bakgrundsfärgen orange. Vidare har ‘Tidsinställning’ en slider där sluttiden på bokningen väljs i intervaller om femton minuter. Vald sluttid visas med en stor text i vyn. För att genomföra bokningen klickar användaren på knappen ‘Bekräfta’. Storleken och placeringen av denna knapp följer det grafiska designmönstret Prominent ‘Done’ Button. Då systemet verkställer bokningen är det inte möjligt för användaren att interagera med Resourcee. Detta tydliggörs genom att ‘Bekräfta’ byter text till “Bekräftar...”. När bokningen är utförd visas vyn ‘Upptaget’ och bokningen får en grafisk representation i komponenten ‘Dygnsschema’ som är placerad i vänsterkanten av vyn.

Skulle användaren i denna vy försöka dra slidern längre än vad som tillåts av dagens övriga bokningar, exempelvis till 16:00 i det scenario som presenteras i figur 9, så går slidern automatiskt tillbaka till en tillåten tid. Detta sker direkt när användaren släpper slidern. På detta vis delges användaren att en bokning kommer sluta tidigare än tänkt innan bokningen bekräftats.



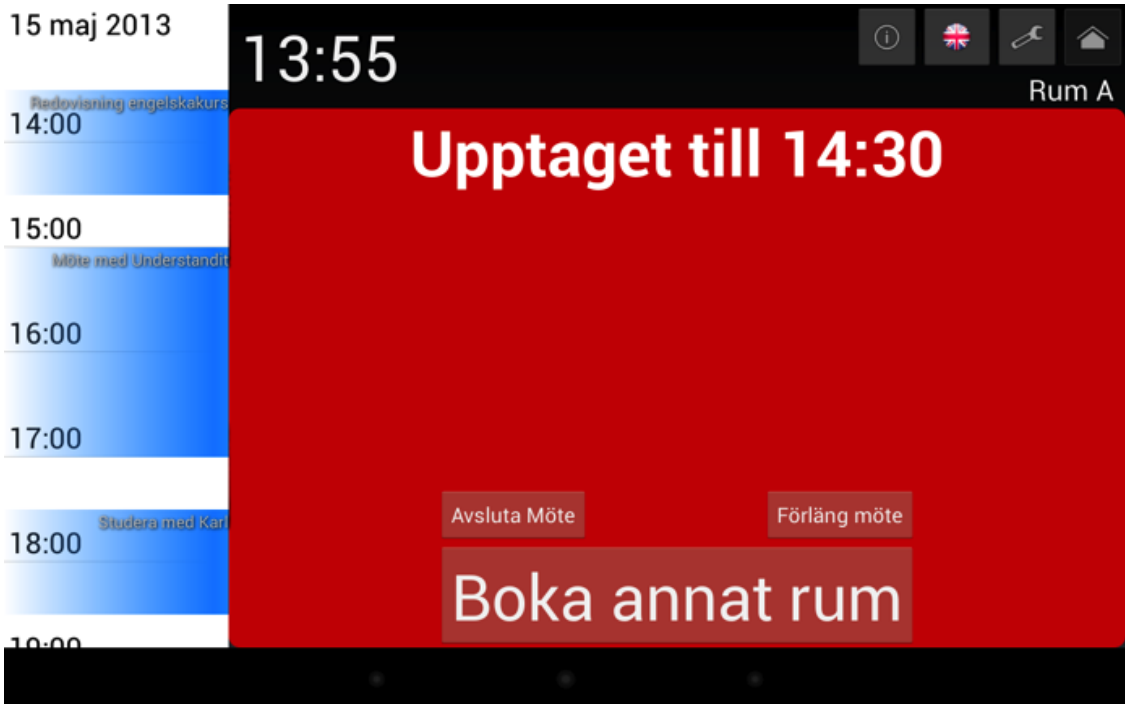
Figur 9: Skärmdump för ‘Rum A’ där en bokning utförs.

5.3.3. Vyn ‘Upptaget’

När det aktuella rummet är upptaget visas vyn ‘Upptaget’ som har röd bakgrundsfärg, vilket illustreras i figur 10. I ‘Upptaget’ visas hur länge rummet är bokat genom en stor text.

I nederdelen av vyn finns tre knappar som användaren kan interagera med: ‘Avsluta möte’, ‘Förläng möte’ samt ‘Boka annat rum’. ‘Avsluta möte’ avslutar den pågående händelsen genom att sätta dess sluttid till tidpunkten då knappen trycks. Om användaren klickar på ‘Förläng möte’ presenteras ‘Tidsinställning’ där användaren kan senarelägga sluttiden.

Knappen ‘Boka annat rum’ är stor och således framträdande, i likhet med knappen ‘Boka’ i ‘Ledigt’. Designmönstret Clear Entry Points har hänt använts för att implementera dessa knappar. När ‘Boka annat rum’ trycks händer ingenting då funktion att boka annat rum inte är fullt implementerad.



Figur 10: Skärmdump för ‘Rum A’ då rummet är upptaget.

5.3.4. Vyn ‘Inställningar’

Resourcee inkluderar även en simpel konfigurationsvy, ‘Inställningar’. Denna vy låter användaren välja det konto som ska användas för enheten från en lista som hämtas från operativsystemet. Dessutom kan användaren välja vilket rum enheten ska användas för, med hjälp av en lista som hämtas från Googles API. Vyn inkluderar även en inställning

för om språkbyte ska tillåtas. För att öppna vyn, vilket kan ske när som helst, krävs ingen autentisering.

5.3.5. Vystruktur

Alla vyer i Resourcee förutom vyn ‘Inställningar’ har en gemensam struktur. Genom denna gemensamma struktur implementeras det grafiska designmönstret Visual Framework. Denna struktur har två komponenter: dygnsschemat samt inmatningskomponenten. Dygnsschemat är placerat i vänsterkanten. Det består av en rullbar yta där bokningar kan placeras. Dygnsschemat representerar ett dygn och är uppritat med linjer samt klockslag vid varje heltimme. Som standard presenterar dygnsschemat de sex kommande timmarna. Har dygnsschemat rullats återgår det efter tio sekunders inaktivitet från användaren till sitt ursprungsläge.

Inmatningskomponenten består av menyn och statuskomponenten. Statuskomponenten har alltid en bakgrundsfärg och innehåller en centrerad text i dess överkant. Vidare innehåller den en, eller flera, knappar centrerade i nederkant.

Menyn är placerad i överkant av inmatningskomponenten. I menyn presenteras aktuell tid samt fyra knappar: ‘Information’, ‘Språk’, ‘Verktyg’ och ‘Hem’ som visualiseras med hjälp av ikoner. Knapparna är nära förankrade och är en typisk implementation av det grafiska designmönstret Button Group.

Knappen ‘Information’ presenterar en ruta mitt på skärmen som visar det aktuella rummets utrustning. Knappen ‘Språk’ möjliggör byte mellan två språk: svenska och engelska. Genom att klicka på ‘Språk’ byts det aktuella språket till det språkflaggan på knappen indikerar. Trycks ‘Verktyg’ leds användaren till ‘Inställningar’. Där kan byte av den kalender enheten kommunicerar med samt vilket rum den representerar göras. Val gällande rummets information samt aktivering av knappen ‘Språk’ görs också i ‘Inställningar’. Inaktiveras möjligheten till språkbyte i menyn försvinner knappen ‘Språk’ och resterande knappar skjuts ihop så att ‘Information’ tar dess plats.

Via ‘Hem’ kommer användaren tillbaka till ‘Ledigt’ eller ‘Upptaget’, beroende på rummets status, och där görs ‘Hem’ oklickbar. Att alltid kunna navigera sig tillbaka till en vy användaren känner igen är en typisk implementation av det grafiska designmönstret Escape Hatch.

5.4. Implementationsdetaljer

Det finns återkommande element i hur Resourcees vyer har implementerats. Vyerna innehåller dygnsschemat, menyn samt statuskomponenten. Implementationen av dygnsschemat och menyn ärver Fragment för att kunna återanvändas i varje Activity.

Ett genomgående begrepp i detta kapitel är *Layout*. Dessa används i Android för att strukturera upp och placera ut element på användargränssnittet. Annat som återkommer är olika 'View'-typer, däribland *TextView* och *ScrollView* vilka är komponenter för text respektive rullbara ytor.

5.4.1. Dygnsschema

Dygnsschemat innehåller bland annat en ScrollView som har en *FrameLayout*. När dygnsschemat skapas placeras 24 *RelativeLayout*-komponenter i dess FrameLayout. Dessa komponenter representerar dygnets 24 timmar och positioneras med olika marginal till toppen beroende på vilken timme komponenten representerar. Anledningen till att FrameLayout använts är för att denna layout utöver x- och y-koordinater även har z-koordinater. Dessa z-koordinater möjliggör placering av komponenter ovanpå varandra.

När en händelse ska presenteras i dygnsschemat skapas en händelsekomponent, vilket är en RelativeLayout-komponent. Denna har en bakgrund i form av en ritbar yta och en TextView som visar händelsens titel. En händelsekomponent placeras på dygnsschemats FrameLayout genom placering ovanpå de komponenter som motsvarar dygnets timmar. Storlek och position på händelsekomponenten bestäms utifrån händelsens längd och starttid. Varje gång dygnsschemat uppdateras tas alla händelsekomponenter bort för att sedan läggas till igen med de förändringar som skett.

5.4.2. Meny

Menyn består av en RelativeLayout med två TextView-komponenter för representation av klocka och rumsnamn, samt en *TableRow* innehållande fyra menyknapparna. TableRow underlättar enhetlig utformning eftersom storlek och placering för alla fyra knappar endast behöver specificeras en gång. Knapptryck hanteras i den Activity som menyn tillhör.

5.4.3. Vyer

Funktionaliteten för två huvudvyerna, 'Ledigt' och 'Upptaget', har implementeras med hjälp av tre klasser. Två av dessa klasser ärver från *MainActivity*, vilket är en abstrakt klass som i sin tur ärver från ResourceeActivity. Anledningen till att MainActivity används är att huvudvyerna representerar två olika status för rummet med olika

utseende och olika uppsättning knappar. Huvudvyerna har en RelativeLayout-komponent som består av statuskomponenten samt två Fragment: dygnsschemat och menyn. Statuskomponenten implementerar *LinearLayout*. Funktionaliteten bakom vyn 'Tidsinställning' har byggts upp på ett likartat sätt, men ärver från den abstrakta klassen *SetTimeActivity*, istället för MainActivity. Genom användning av abstrakta klasser undviks källkodsduplicering.

5.4.4. Uppdateringar

Gränssnittet *PropertyChangeListener* används för att hantera kalenderförändringar och klockförändringar i enlighet med Observermönstret. Detta gränssnitt implementeras av samtliga Fragment och vyer i Resourcee. I ResourceeActivity registreras dessa Fragment och vyer sedan som lyssnare i modellen. Vid förändring hos modellen skickas motsvarande *PropertyChangeEvent* till alla registrerade lyssnare som hanterar dessa uppdateringar. De olika typer av PropertyChangeEvent som modellen kan skicka och deras innehåll syns i tabell 3.

Tabell 3: Olika PropertyChangeEvent och deras innehåll.

PropertyChangeEvent	**Innehåll**
CALENDARS_UPDATE	Lista med Google Calendars.
ROOM_UPDATE	Ett booleskt värde representerande rummets pågående status (upptaget eller ej).
EVENT_UPDATE	En händelse representerande pågående eller nästa händelse.
DATE_TIME_UPDATE	Ett datumobjekt med nuvarande klockslag och datum.
CURRENT_CALENDAR_UPDATE	Aktuell Google Calendar.

5.5. Användartest

Användartesterna visade att Resourcees funktioner uppfattades som lättanvända. Genomgående ansåg testpersonerna att gränssnittet var tydligt och att knapparna var logiskt placerade. Dessutom upplevdes Resourcee som responsiv. Alla testpersonerna lyckades utföra uppgiften att boka ett rum på första försöket. Vissa testpersoner beskrev en osäkerhet vid fastställandet av sluttid, då slidern var svår att upptäcka. Det var också svårt att förstå i vilket intervall slidern rörde sig. Efter att testpersonerna hade utfört bokningsuppgiften en gång upplevdes dock även detta moment som enkelt.

Det fanns även efterfrågan på en funktion som kunde lägga till information om bokningar, både vid skapandet av bokningen men också till en tidigare skapad bokning.

Information kan bara läggas till om bokningen sker via webbgränssnittet till det existerande bokningssystemet. Vidare försökte flera testpersoner använda sig av dygnsschemat för att åstadkomma antingen en bokning eller försöka bestämma en boknings sluttid.

Det fanns ingen testperson som misslyckades med att på egen hand utföra någon av de uppgifter de blev tilldelade och alla kommenterade Resourcees huvuduppgift att boka rum som enkel. En testperson beskrev att de få alternativen i användargränssnittet gör det nästintill omöjligt att göra fel vid bokning, vilket är något som har eftersträvats vid den grafiska designen. För en komplett sammanställning av användartesternas resultat se bilaga G.

6. Diskussion

Detta kapitel diskuterar noterbara detaljer i projektet. Först behandlas utvecklingens förarbete och därefter dess implementation. Slutligen diskuteras möjligheter för framtida arbete och vidareutveckling.

6.1. Förarbete

Den genomförda enkätundersökningen, vilken bestod av slutna frågor, var tänkt att utforska vilka funktioner som eftersöks av potentiella användare. Enkäten kompletterade den uppfattning om projektets utformning som redan formats. I och med att enkäten riktade sig mot tidigare användare av rumsbokningssystem minskades risken för att respondenterna inte skulle förstå Resourcees användningsområde och koncept.

Enkätundersökningen fick 22 svar, samtliga från respondenter som redan var insatta i grundkonceptet med ett fysiskt rumsbokningssystem. Då systemets tänkta funktion kan vara svårförståeligt för oinsatta gjordes avvägningen att ett färre antal mer ämnesinsatta individer skulle ge ett bättre enkätresultat.

Att ta fram en kravspecifikation och en lista över användningsfall gav från början en tydlig struktur i arbetet. Detta förenklade det iterativa implementationsarbetet och gav möjlighet att sätta upp klara milstolpar. Genom inbördes prioritering av användningsfallen blev det dessutom enklare att veta vad som behövde göras. När ett användningsfall ansågs färdigutvecklat kunde arbete påbörjas på nästa. Implementeringen av användningsfallen följde i stora drag den bestämda prioriteringen. De undantag som gjordes berodde på att projektets tidsram inte tillät implementation av alla större användningsfall. Fokus lades istället på mindre användningsfall som bedömdes hinna bli klara i tid.

6.2. Implementation

Eftersom Resourcee utvecklats i en iterativ process blev dess implementation stegvis mer komplett. Mot projektets slutförande fanns flertalet lösningar på tidigare problem.

Strävan efter att ha ett gemensamt grafiskt utseende medförde att vissa element skulle existera i samtliga vyer. Genom att implementera vissa element som varsitt Fragment kunde källkodsduplicering undvikas.

Språkbytesfunktionen orsakade problem eftersom instanser av Fragment försvann när *setContentView* kallades för andra gången. Lades de till igen flimrade skärmen. Detta löstes genom att den Activityinstans som var aktiv vid språkändringen startades om.

6.2.1. Nätverkskommunikation

Tidigt upptäcktes att operativsystemets kalenderapplikation Google Calendar uppdaterades väldigt snabbt. Efter att en händelse skapats genom Google Apps webbgränssnitt tog det sällan mer än en sekund innan denna hade dykt upp i kalenderapplikationen. Utifrån detta togs beslutet att sköta all kommunikation endast genom denna applikation. Tanken var att nya händelser skulle läggas in i kalenderapplikationen, och helt överlåta synkronisering med webbtjänsten till denna.

Det tog dock lång tid för händelser skapade i kalenderapplikationen att dyka upp i webbtjänsten. Dessutom kan händelsers globala unika identifikationssträngar inte erhållas från kalenderapplikationen, då kalenderapplikationen helt enkelt inte lagrar dem.

Lösningen blev att skapa en hybridlösning där kalenderapplikationens uppgift endast består av att meddela Resourcee att en förändring skett. Vid denna tidpunkt undersöker Resourcee vad som skett genom att kontakta Googles kalender-API. Denna lösning innebar att uppdateringar erhålls inom ett par sekunder, samtidigt som processen att skapa nya bokningar i webbtjänsten sker så snabbt som enhetens internetuppkoppling tillåter.

Att all nätkommunikation sker såpass smidigt tillåter att Resourcee aldrig byter rumsstatus utan att det är helt säkerställt att den nya statusen gäller. Hade kommunikationen tagit längre tid hade detta varit svårgenomfört då användaren skulle blockeras från att interagera med applikationen.

6.2.2. Flertrådning

Hämtning av information från Googles kalender-API kan låsa en programtråd i flera sekunder. Därför utförs dessa hämtningar i separata trådar. Detta orsakade problem för insamlingen av information. *Race conditions* uppstod då trådar skrev över varandras information, och *deadlock* förhindrade vissa trådar från att någonsin avsluta sin exekvering. För att åtgärda detta implementerades grundläggande trådsäkerhetskonstruktioner i ResourceeActivity. Dessutom flyttades kommunikationslogik till klassen GAPIUpdateTask som implementerar Androidspecifika trådkonstruktioner.

6.3. Framtida arbete

Syftet med projektet var att utveckla och leverera en prototyp av Resourcee till Understandit för utvärdering av vidareutvecklingsmöjligheterna. Att Resourcee ska vidareutvecklas har därför hållits i åtanke genom hela projektet.

Källkoden i Resourcee är väldokumenterad och använder sig av kända designmönster, som MVC och Observer. Detta gör det enkelt att sätta sig in i dess struktur. Omfattande förstudier har även gjorts där potentiella kunders åsikter om användargränssnitt samt funktionalitet har samlats in. Med allt detta i åtanke anser vi oss ha utvecklat en applikation där möjligheterna för vidareutveckling är goda.

Programstabilitet har prioriterats framför detaljer i användargränssnittet. Av denna anledning rekommenderar vi en förbättring av användargränssnittet, för att göra det mer visuellt tilltalande, vid en vidareutveckling av Resourcee.

Alla funktioner med hög prioritet, och ett antal med mellanprioritet, har implementerats. Fokus har legat på att få basfunktionaliteten att fungera utan några programfel. På grund av projektets begränsade tidsram har övriga funktioner inte realiserats.

Resourcee är förberedd för implementation av bokning av andra rum än det rum applikationen är konfigurerad för. Den rudimentära version som finns av funktionen består av en prototyp till ett användargränssnitt. En fullständig funktion skulle göra Resourcee mer konkurrenskraftig. Samma kommunikationsuppgifter som för resten av Resourcee skulle kunna användas till funktionen, vilket medför att funktionen lär vara relativt enkel att implementera. Av dessa anledningar anser vi att denna funktion bör vara näst på tur att utveckla.

Ingen inloggningsfunktion har implementerats. Vyn ‘Inställningar’ kan kommas åt av vem som helst, vilket kan orsaka konfigurationsproblem. Utan autentisering hanteras dessutom bokningar anonymt, oavsett vem som skapat dem. Vidare kan bokningar skapas okontrollerat, utan möjlighet att spåra dessa tillbaka till specifika användare. Det kan dock vara intressant att kunna spåra den användare som använt rummet om skador uppstått eller om något lämnats kvar.

Ett inloggningsförfarande ger många fördelar, men kan också innebära nackdelar som rumsbokningsapplikationen uttryckligen inte skulle innehålla. Om det krävs inmatning av användaruppgifter vid hantering av en bokning blir det omöjligt att klara av det nuvarande icke-funktionella kravet att göra en bokning på maximalt fyra klick (se bilaga C). En lösning är att använda en identifieringsmetod som inte kräver inmatning av

användaruppgifter, exempelvis någon form av trådlös överföring. Då skulle ett passerkort, en aptusbricka eller en mobiltelefon kunna användas för identifikation. Detta skulle lösa säkerhetsaspekten för framtida bokningar samt bokningar av andra rum. Att kräva trådlös kommunikation för autentisering skulle dock innebära förlorat stöd för de enheter som inte stödjer denna teknik. Vi anser att en inloggningsfunktion bör implementeras, då fördelarna överväger nackdelarna.

Som enkätresultatet visade önskade användarna avancerad funktionalitet samtidigt som det grafiska användargränssnittet skulle vara enkelt. När en applikations funktionalitet utökas måste därför det grafiska användargränssnittet anpassas, eftersom användbarheten annars riskerar att försvåras. Alltså har utökad funktionalitet hela tiden vägts mot strävan att ha en lätthanterad applikation. Genom Resourcees utveckling har det senare prioriterats.

Funktionalitet för att boka rum framåt i tiden implementerades aldrig, trots att 95 % av användarenkätens respondenter önskade denna funktion. Vi ställer oss dock frågande till om det över huvud taget är fördelaktigt att implementera denna funktionalitet. Användare kan lika gärna logga in i Google Apps på webben för framtida bokningar, då applikationens syfte är att finnas till för spontana bokningar.

6.4. Användartest

När användargränssnittet utvärderades genom användartesterna erhölls givande resultat. Resourcee beskrevs av testpersonerna som responsivt, vilket reflekterar vårt fokus på snabba uppdateringar. Vidare beskrevs användargränssnittet som tydligt och lättanvänt. Samtliga deltagare lyckades snabbt genomföra alla tilldelade uppgifter, vilket antyder att användargränssnittet är lättförståeligt. Med detta har vårt mål med utformningen av användargränssnittet uppnåtts.

Trots positiv respons framkom möjlighet till förbättringar i användargränssnittet. Att somliga av testpersonerna tvekade när de skulle fastställa sluttiden för bokningar var en intressant iakttagelse från användartesten. Vi tror att den effektivaste metoden att sätta bokningens sluttid är med hjälp av en slider. Med en slider erbjuds användaren ett stor antal alternativ med en enda komponent som dessutom kräver lite interaktion. Ett alternativt tillvägagångssätt att välja sluttid skulle vara att presentera en knappgrupp med fördefinierade förslag. Detta ansåg vi skulle ge mindre flexibilitet än vår lösning.

Testresultatet antyder dock att sliderns funktion bör förtydligas. Det kan till exempel vara att ge slidern en röd färg de första sekunderna. Slidern bör även ha någon form av anvisning för skala för att förenkla användning. Överlag anser vi att användartesterna

gav bra förslag på hur Resourcee kan förbättras. Vidare indikerar den positiva responsen på en väl utformad prototyp.

6.5. Generalisering

Målet med projektet var att utveckla en välfungerande rumsbokningsapplikation mot etablerade grupprogram. Under projektets gång har det blivit tydligt att denna applikation skulle kunna användas till mer än bara rumsbokning. Stöd finns för hantering av andra resurser än rum i Google Apps. I Resourcee skulle endast textsträngar i användargränssnittet behöva ändras, vilka redan lagras separat från programkällkoden. Istället för att begränsa applikationen till rumsbokningar skulle Resourcee kunna användas som ett generellt system för bokningsbara resurser.

Bland de utvecklingsmässiga koncept som projektet bygger på är framförallt implementationen av MVC i en Androidmiljö något som framtida projekt kan ta del av. Att implementera MVC i en miljö med hårt sammanbundna komponenter är en svårighet projektet överkommit, och kan bidra med lösningar till.

7. Slutsats

Den färdigställda prototypen av Resourcee överensstämmer väl med den produkt som specificerades i Understandits projektförslag. Samtliga högprioriterade användningsfall har implementerats, liksom hälften av användningsfallen med mellanprioritet. Användargränssnittet är uppbyggt med hjälp av beprövade grafiska designmönster, vilket har lett till att applikationen är enkel att använda. Den källkod som ligger till grund för Resourcee är välstrukturerad och väldokumenterad samt implementerar etablerade designmönster.

Att arbeta iterativt tillät att åsikter kontinuerligt kunde hämtas från uppdragsgivaren. Utvecklingsprocessen hade kunnat förbättras genom att i större omfattning ha kontinuerliga möten avsedda för diskussion om uppdateringar av källkod. Gruppmedlemmarna hade då varit bättre uppdaterade gällande nyskriven källkod, samt haft möjlighet att tydligare och enklare dela upp implementationen. Senare i utvecklingen infördes problembeskrivningar och personliga åtaganden, vilket visade sig vara ett mycket effektivt arbetssätt.

Grunden har lagts för framtida utveckling av Resourcee. Applikationen är designad på ett så abstraherat vis som möjligt, för att förenkla vidareutveckling. Användartesterna har presenterat de styrkor Resourcee besitter, vilka överensstämmer väl med de krav uppdragsgivaren framställde. Således är uppdragsgivaren nöjd med den presenterade prototypen. Att generalisera applikationen för att stödja bokning av andra resurser än just rum förefaller inte heller vara en svår uppgift då det har tagits i beaktande under utvecklingsarbetet.

Källförteckning

Android, u.å. *Android Device Gallery* [webbsida] Tillgänglig via: http://www.android.com/devices/?country=all (Hämtad 2013-05-20)

Android Compatibility Program, 2013. *Android 4.2 Compatibility Definition.* [pdf] Tillgänglig via: https://static.googleusercontent.com/external_content/untrusted_dlcp/source.android.com/sv//compatibility/4.2/android-4.2-cdd.pdf (Hämtad 2013-05-08)

Android Developers, 2013 #1. *Dashboards.* [Webbsida] Tillgängligt via: http://developer.android.com/about/dashboards/index.html (Hämtad 2013-02-13)

Android Developers, 2013 #2. *ADT Plugin.* [Webbsida] Tillgängligt via: http://developer.android.com/tools/sdk/eclipse-adt.html (Hämtad 2013-05-03)

Android Developers, 2013 #3. *Activities.* [Webbsida] Tillgängligt via: http://developer.android.com/guide/components/activities.html (Hämtad 2013-04-19)

Android Developers, 2013 #4. *Application Fundamentals.* [Webbsida] Tillgängligt via: http://developer.android.com/guide/components/fundamentals.html (Hämtad 2013-04-17)

Android Developers, 2013 #5. *Content Providers.* [Webbsida] Tillgängligt via: http://developer.android.com/guide/topics/providers/content-providers.html (Hämtad 2013-04-22)

Android Developers, 2013 #6. *Authenticating to OAuth2 Services.* [Webbsida] Tillgängligt via: http://developer.android.com/training/id-auth/authenticate.html (Hämtad 2013-04-22)

Android Developers, 2013 #7. *Services.* [Webbsida] Tillgängligt via: http://developer.android.com/guide/components/services.html (Hämtad 2013-04-23)

Android Developers, 2013 #8. *Creating an Android Project.* [Webbsida] Tillgängligt via: http://developer.android.com/training/basics/firstapp/creating-project.html (Hämtad 2013-04-24)

Android Developers, 2013 #9. *What is API Level?*. [Webbsida] Tillgängligt via: http://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels (Hämtad 2013-04-30)

Android Developers, 2013 #10. *Supporting Multiple Screens.* [Webbsida] Tillgängligt via: http://developer.android.com/guide/practices/screens_support.html (Hämtad 2013-05-03)

Android Developers, 2013 #11. *Supporting Tablets and Handsets.* [Webbsida] Tillgänglig via: http://developer.android.com/guide/practices/tablets-and-handsets.html (Hämtad 2013-05-03)

Android Developers, 2013 #12. *Designing for TV*. [Webbsida] Tillgängligt via: http://developer.android.com/training/tv/index.html (Hämtad 2013-05-03)

Android Developers, 2013 #13. *Android NDK.* [Webbsida] Tillgänglig via: http://developer.android.com/tools/sdk/ndk/index.html (Hämtad 2013-05-08)

Android Developers, 2013 #14. *Web Apps Overview.* [Webbsida] Tillgänglig via: http://developer.android.com/guide/webapps/overview.html (Hämtad 2013-05-08)

Android Developers, 2013 #15. *Google Play Services.* [Webbsida] Tillgänglig via: http://developer.android.com/google/play-services/index.html (Hämtad 2013-05-08)

Android Developers, 2013 #16. *Ice Cream Sandwich* [Webbsida]. Tillgänglig via: http://developer.android.com/about/versions/android-4.0-highlights.html#DeveloperApis (Hämtad 2013-05-08)

Android Developers, 2013 #17. *Remaining Backward Compatible.* [Webbsida] Tillgänglig via: http://developer.android.com/training/search/backward-compat.html (Häm

Android Developers, 2013 #18. *Android 1.6 3 platform Highlights.* [Webbsida] Tillgänglig via: http://developer.android.com/about/versions/android-1.6-highlights.html (Hämtad 2013-05-08)

Android Developers, 2013 #19. *Fragments.* [Webbsida] Tillgänglig via: http://developer.android.com/guide/components/fragments.htm (Hämtad 2013-04-19)

Android Developers, 2013 #20. *Intent.* [Webbsida] Tillgänglig via: http://developer.android.com/reference/android/content/Intent.html (Hämtad 2013-05-09)

Android Developers, 2013 #21. *Service.* [Webbsida] Tillgänglig via: http://developer.android.com/reference/android/app/Service.html#WhatIsAService (Hämtad 2013-05-09)

Android Developers, 2013 #22. *Calendar Provider.* [Webbsida] Tillgänglig via: http://developer.android.com/guide/topics/providers/calendar-provider.html (Hämtad 2013-05-10)

Android Developers, 2013 #23. *Android Developer Tools.* [Webbsida] Tillgänglig via: http://developer.android.com/tools/help/adt.html (Hämtad 2013-05-07)

Apple, 2007. *Apple Reinvents the Phone with iPhone.* [pressmeddelande] 9 januari 2007. Tillgänglig via: http://www.apple.com/pr/library/2007/01/09Apple-Reinvents-the-Phone-with-iPhone.html (Hämtad 2013-05-19)

Balolong, F. R., 2011. Android 4.0 Ice Cream Sandwich SDK Now Available, *Social Barrel,* [Webbsida] Tillgänglig via: http://socialbarrel.com/android-4-0-ice-cream-sandwich-sdk-now-available/24093/ (Hämtad 2013-05-02)

Balsamiq, u.å. *Balsamiq Mockups* [webbsida] Tillgänlig via: http://www.balsamiq.com/products/mockups (Hämtad 2013-05-17)

Bornstein, D., 2008. *Dalvik VM Internals.* [pdf] Tillgänglig via: https://sites.google.com/site/io/dalvik-vm-internals/2008-05-29-Presentation-Of-Dalvik-VM-Internals.pdf (Hämtad 2013-05-08).

Craig, I., 2006. *Virtual Machines.* [e-bok] Springer London: USA. Tillgänglig via: Springer Link link.springer.com. (Hämtad 2013-05-04)

developer.android.com, 2012. *Nexus 4.* [Bild på webbsida] Tillgänglig via: http://commons.wikimedia.org/wiki/File:Nexus_4.png (Hämtad 2013-05-19)

Ducrohet, Xavier, 2011. Final Android 3.0 Platform and Updated SDK Tools. *Android Developers Blog,* [Blog] 22 februari. Tillgänglig via: http://android-

developers.blogspot.se/2011/02/final-android-30-platform-and-updated.html (Hämtad 2013-05-07)

Eclipse Marketplace, 2013 #1. *UMLet - UML Tool for Fast UML Diagrams 12.0.* [webbsida] Tillgänglig via: http://marketplace.eclipse.org/content/umlet-uml-tool-fast-uml-diagrams#.UYj25pWevaV (Hämtad 2013-05-03)

Eclipse Marketplace, 2013 #2. *FindBugs Eclipse Plugin.* [webbsida] Tillgänglig via: http://marketplace.eclipse.org/content/findbugs-eclipse-plugin#.UYj3AJWevaV (Hämtad 2013-05-03)

Eclipse Marketplace, 2013 #3. *New and Updated Solutions.* [webbsida] Tillgänglig via: http://marketplace.eclipse.org (Hämtad 2013-05-03)

Ehringer, D., 2010. *The Dalvik Virtual Machine Architechture.* [pdf] Tillgänglig via: http://davidehringer.com/software/android/The_Dalvik_Virtual_Machine.pdf (Hämtad 2013-05-08).

Elgin, B., 2005. *Google Buys Android for its Mobile Arsenal* [webbsida] Bloomberg Businessweek: 16 augusti 2005. Tillgänglig via: http://www.businessweek.com/stories/2005-08-16/google-buys-android-for-its-mobile-arsenal (Hämtad 2013-05-03)

Evoko, 2013. [webbsida] Tillgängligt via: http://www.evoko.se/ (Hämtad 2013-02-13)

Felker, D., 2012. *Android Tablet Application Development for Dummies.* [e-bok] New Jersey: John Wiley & Sons. Tillgänglig via: Chalmers Bibliotek <chans.lib.chalmers.se> (Hämtad 2013-04-30)

Forrester, 2009. *The Total Economic Impact Of Microsoft Exchange 2010* [pdf] Tillgänglig via: http://download.microsoft.com/download/7/5/0/75068b44-0a70-4bbf-9824-01ecf076f7ae/thetotaleconomicimpact_pdf_11042009.pdf (Hämtad 2013-02-13)

Forrester, 2012. *The Total Economic Impact Of Google Apps* [pdf] Tillgänglig via: http://www.linkgard.com/wp-content/uploads/2013/01/TEI-of-Google-Apps-2012_87903212.pdf (Hämtad 2013-02-13)

Gartner, 2013. *Gartner Says WorldWide PC, Tablet and Mobile Phone Combined Shipments to Reach 2.4 Billion Units in 2013* [pressmeddelande] 4 april 2013. Tillgängligt via: http://www.gartner.com/newsroom/id/2408515 (Hämtad 2013-05-07)

Github, 2013 #1. *About Github* [webbsida] Tillgänglig via: *https://github.com/about (*Hämtad *2013-04-17).*

Github, 2013 #2. *Features* [webbsida] Tillgänglig via: https://github.com/features/projects/codereview (Hämtad 2013-04-16).

Google Apps for Business, u.å. [webbsida] Tillgänglig via: http://www.google.com/enterprise/apps/business/products.html (Hämtad 2013-02-13)

Google Calendar, 2013. *Welcome to Google Calendar* [webbsida] Tillgänglig via: https://support.google.com/calendar/answer/2465776?hl=en (Hämtad 2013-05-04)

Google Code, 2013. *Google APIs Client Library for Java* [webbsida] Tillgänglig via: https://code.google.com/p/google-api-java-client/ (Hämtad 2013-03-10)

Google Developers, 2013 #1. *Using OAuth 2.0 to Access Google APIs* [webbsida] Tillgänglig via: https://developers.google.com/accounts/docs/OAuth2 (Hämtad 2013-03-10)

Google Developers, 2013 #2. *Google Calendar API Reference* [webbsida] Tillgänglig via: https://developers.google.com/google-apps/calendar/v3/reference/ (Hämtad 2013-04-22)

Google Developers, 2013 #3. *Calendar API v3 (revision 40)* [webbsida] Tillgänglig via: https://developers.google.com/resources/api-libraries/documentation/calendar/v3/java/latest/ (Hämtad 2013-04-24)

Google Developers, 2013 #4. *Google APIs Console Help* [webbsida] Tillgänglig via: https://developers.google.com/console/help/ (Hämtad 2013-04-24)

Google Drive, u.å. *About* [webbsida] Tillgänglig via: http://www.google.com/drive/about.html (Hämtad 2013-05-20)

Google I/O, 2010. *Google I/O 2010 - A JIT Compiler for Android's Dalvik VM.* [Video på webbsida] Tillgänglig via: https://www.youtube.com/watch?v=Ls0tM-c4Vfo (Hämtad 2013-04-22)

Google Play, u.å. *Google Calendar* [webbsida] Tillgänglig via: https://play.google.com/store/apps/details?id=com.google.android.calendar&hl=en (Hämtad 2013-05-19)

Google Support, 2013 #1. *Schedule Calendar Resources* https://support.google.com/calendar/answer/143753?hl=en (Hämtad 2013-05-14)

Google Support, 2013 #2. *Create a Calendar for Shared Resources (conference rooms, sports fields, etc.)* *https://support.google.com/calendar/answer/34584* (Hämtad 2013-05-14)

Hastings, R. 2009. Groupware. *Library Technology Reports.*[e-journal] 45(4). Endast sammanfattning. Tillgänglig via: Chalmers bibliotek <lib.chalmers.se> (2013-06-04)

HUI Research, 2013. *Årets Julklapp* [webbsida] Tillgänglig via: http://www.hui.se/arets-julklapp (Hämtad 2013-05-02)

Hyeong-Seok, O., 2012. Evaluation of Android Dalvik virtual machine. JTRES (Java Technologies for Real-time and Embedded Systems), *Proceedings of the 10th International Workshop on Java Technologies for Real-time and Embedded Systems.* Copenhagen, Denmark 24-26 October 2012. New York: ACM New York

International Data Corporation, 2013. *Gartner Says Worldwide Mobile Phone Sales Declined 1.7 Percent in 2012* [pressmeddelande] 13 februari 2013. Tillgänglig via: http://www.gartner.com/newsroom/id/2335616 (Hämtad 2013-05-03)

Jordan, L., Greyling, P., 2011. *Practical Android Projects*. [e-bok] Springer Science: New York, NY, USA. Tillgänglig via: Springer Link link.springer.com. (Hämtad 2013-05-03)

Kelly, M., 2012. *86 percent of companies use multiple cloud services, says study* [webbsida] Tillgänglig via: http://venturebeat.com/2012/05/10/cloud-services-data/ (Hämtad 2013-02-13)

Komatineni, S., MacLean, D., 2012. *Pro Android 4.* [e-bok] Apress: New York, NY, USA. Tillgänglig via: Springer Link link.springer.com. (Hämtad 2013-05-04)

Lardinois, F., 2012. *Gmail Now Has 425 Million Users, Google Apps Used By 5 Million Businesses And 66 Of The Top 100 Universities*. [webbsida] TechCrunch, 28 juni.

http://techcrunch.com (Hämtad 2013-02-13).

Larman, C., 2005. Applying UML and Patterns: an introduction to object-oriented analysis and design and iterative development. 3rd ed. Upper Saddle River, N.J. : Prentice Hall PTR, cop.

Leiba, B., 2012, OAuth Web Authorization Protocol. *IEEE Internet Computing* [e-journal] 16(1), s. 74-77. Tillgänglig via: Chalmers bibliotek <lib.chalmers.se> (Hämtad 2013-03-10)

Markoff, J., 1999. *Microsoft Brings In Top Talent To Pursue Old Goal: The Tablet*. [webbsida] The New York Times: 30 augusti 1999. Tillgänglig via: www.nytimes.com (Hämtad 2013-05-01)

Markoff, J., 2007. *I, Robot: The Man Behind the Google Phone* [webbsida] The New York Times: 4 november 2007. Tillgänglig via: www.nytimes.com (Hämtad 2013-05-03)

Martin, R. C., 2000. *Design Principles and Design Patterns*. [pdf] Tillgänglig via: http://www.objectmentor.com/resources/articles/Principles_and_Patterns.pdf (Hämtad 2013-05-16)

McFadden, Andy, 2009. Backward compatibility for Android applications. *Android Developers Blog,* [Blog] 28 april. Tillgänglig via: http://android-developers.blogspot.se/2009/04/backward-compatibility-for-android.html (Hämtad 2013-05-08)

Morel M., Alves M., Pascal C., Lemberger P., 2011. *Google Apps: Mastering Integration and Customization*. [e-bok] Packt Publishing Ltd: Olton Birmingham, GB. Tillgänglig via: Chalmers bibliotek <lib.chalmers.se> (Hämtad 2013-05-07)

Microsoft, 2013 #1. *Exchange Online: business-class email that's easy to manage*. [webbsida] Tillgänglig via: http://office.microsoft.com/en-us/exchange/microsoft-exchange-online-email-for-business-FX103739072.aspx (Hämtad 2013-02-13)

Microsoft, 2013 #2. *Share an Outlook calendar with other people*. [webbsida] Tillgänglig via: http://office.microsoft.com/en-us/outlook-help/share-an-outlook-calendar-with-other-people-HA010354420.aspx?CTT=1 (Hämtad 2013-05-04)

Microsoft, 2013 #3. *Model-View-Controller* [webbsida] Tillgänglig via: http://msdn.microsoft.com/en-us/library/ff649643.aspx (Hämtad 2013-05-17)

Nationalencyklopedin, 2013. *Surfplatta* [Webbsida] Malmö: NE Nationalencyklopedin AB (SWE) Tillgänglig via: http://www.ne.se/lang/surfplatta (Hämtad 2013-05-01)

OODesign, u.å #1. *Singleton Pattern* [webbsida] Tillgänglig via: http://www.oodesign.com/singleton-pattern.html (Hämtad 2013-05-15)

OODesign, u.å #2. *Observer Pattern* [webbsida] Tillgänglig via: http://www.oodesign.com/observer-pattern.html (Hämtad 2013-05-15)

Olawuyi, J.O. och Mgbole, F., 2012. Technological Convergence. *Science Journal of Physics, Volume 2012*, [e-journal] Tillgänglig via: www.sjpub.org (Hämtad 2013-02-13)

Pew Research Center, 2012. *Device Ownership* [pressmeddelande] 1 oktober 2012. Tillgänglig via: http://www.journalism.org/analysis_report/device_ownership (Hämtad 2013-05-02)

Samsung, 2012. *28 cu. ft. 4-Door Refrigerator and 8" LCD Digital Display with Apps* [webbsida] Tillgänglig via: http://www.samsung.com/us/appliances/refrigerators/RF4289HARS/XAA (Hämtad 2013-02-15)

Smieh, 2012. *Android System Architecture* [Bild på webbsida] Tillgänglig via: http://commons.wikimedia.org/wiki/File:Android-System-Architecture.svg (Hämtad 2013-04-18)

Stannered, 2010. *ModelViewController* [Bild på webbsida] Tillgänglig via: *http://commons.wikimedia.org/wiki/File:ModelViewControllerDiagram2.svg* (Hämtad 2013-05-17)

Stray, A., 2013. IOS och Android delar snart på plattmarknaden. *M3* [webbsida] Tillgänglig via: http://m3.idg.se/2.1022/1.504824/ios-och-apple-delar-snart-pa-plattmarknaden (Hämtad 2013-04-25)

The Verge, 2012. *Android 4.2 adds Gesture Typing, Wireless TV Display, Multiple User Support on Tablets, and more* [webbsida] Tillgänglig via:

http://www.theverge.com/2012/10/29/3569244/android-4-2-new-features-miracast-gesture-keyboard-multiple-users-photo-sphere (Hämtad 2013-04-16)

Tidwell, J. 2011. *Designing Interfaces*. 2nd ed. Sebastopol, CA: O'Reilly Media

Unamed102, u.å. *Android home* [Bild på webbsida] Tillgänglig via: http://commons.wikimedia.org/wiki/File:Android_home.png (Hämtad 2013-05-02)

Understandit, 2013. *En webbyrå i Göteborg*. [webbsida] Tillgänglig via: http://understandit.se/om-understandit (Hämtad 2013-04-30)

Vogella, 2013. *Android BroadCastReceiver Tutorial* [webbsida] Tillgänglig via: http://www.vogella.com/articles/AndroidBroadcastReceiver/article.html (Hämtad 2013-04-22)

Whitwam, R., 2012. How to install Android apps. *Geek.com* [webbsida] Tillgänglig via: http://www.geek.com/mobile/how-to-install-android-apps-1480675/ (Hämtad 2013-05-09).

Xbox, 2013. *Börja titta på en film hemma - se klart den på resande fot* [webbsida] Tillgänglig via: http://www.xbox.com/sv-SE/entertainment/video?xr=shellnav (Hämtad 2013-05-18)

Zechner, M., 2011. *Beginning Android Games.* [e-bok] Berkeley, CA: Apress. Tillgänglig via: Chalmers Bibliotek <chans.lib.chalmers.se> (Hämtad 2013-05-01)

Bilaga A – Projektbeskrivning från Understandit

En app för Android-tablets för resursbokning.

På en större arbetsplats finns det behov av ett system för att boka resurser (t.ex. mötesrum). De stora grupprogramvarorna (Exchange, Google Apps) har stöd för att lägga upp resurser som sedan kan bokas via kalender.

Projektet går ut på att implementera en app som kan köras på Android-tablets somt.ex. hänger utanför dörren till ett mötesrum och indikerar om rummet för tillfället är bokat eller inte, och erbjuda vissa funktioner såsom att boka rummet direkt (om det är ledigt).

Det finns existerande system för detta, T.ex. Evoko (http://www.evoko.se/), mende flesta använder specialtillverkad hårdvara och har begränsad funktionalitet.

Vi ser en möjlighet att utveckla en app som fungerar på billig "off the shelf"-hårdvara som kan säljas i paket med mjukvaran till ett konkurennskraftigt pris. Mjukvaran kan dessutom fungera som en produkt i sig och säljas separat.

Det finns också en tanke om att utveckla stöd för en extern indikator i stil med den som Evokos lösning har.

Förslag på funktioner:

- Tydligt se om resursen är ledig på skärm, eller med extern indikator. T.ex LED som lyser i olika färg beroende på status.
- Bokning av rum från tablet.
- Kunna se i en kalendervy
- Integrera med Exchange och Google Apps * Integration med extern statusindikator. (LED eller liknande)

Bilaga B – Funktionella krav

Applikation:

- Möjlighet att se om rummet i ögonblicket är upptaget eller ej
- Logga in/logga ut som användare/administratör
- Information kring rum (se utrustning, namn, och dylikt)
- Konfigurera rum
- Boka rummet (förlänga bokning, osv.) - Boka JUST NU, ej framtiden
- Avboka rummet - Ev tillåta, eller inte tillåta
- Se framtida schema
- Visa nuvarande tid, datum, osv.
- Synkronisering med externa system (Google Apps)
- Boka andra rum (Hitta ett icke-upptaget rum i närheten?)
- NFC-anslutning till annan enhet (som kör annan app) (Läsa RFID-nycklar?)
- Hjälpsektion
- Systeminställningar
- El- och internetanslutning
- Ev. skärmstorlekskrav? 7 tum och uppåt. Fungera på olika "bildförhållanden".
- Byta språk
- Felanmälan
- Information om specifik bokning

Systemet:

- Rumsbokning
- En vanlig användare skall inte kunna stänga av huvudapplikationen som körs på bokningsplattorna.
- Identifiera användare och admnistratörer.
- Valfri incheckning (avbokning/bestraffning)

Bilaga C – Icke-funktionella krav

Applikationen:

- Kalenderuppdateringstid under 10 sekunder
- Inte för många "nivåer". Ska kunna göra bokning osv. på maximalt 4 "klick".
- Responsivt användargränssnitt
- Hög uptime utan problem (månader i sträck)
- Kraschfrihet
- Uppdateringsmöjlighet
- Elavbrottssäkerhet
- Inputsäkerhet
- Internationalisering (flerspråksmöjlighet)
- Appen ska kunna fungera till olika plattor som uppfyller grundkraven (appen anpassningsbar)
- Huruvida ett rum är upptaget eller ej skall vara väldigt tydligt, exempelvis genom att bakgrunden

Systemet:

- Generalisera systemet på så sätt att resurserna som bokas kan bytas till vilken sorts resurs som helst (istället för endast rum). Exempelvis bilar, båtar, operabiljetter.

Bilaga D – Enkätundersökning – frågor

Vi är sex stycken civilingenjörsstudenter från Chalmers som tillsammans med företaget Understandit utför ett kandidatarbete på institutionen Informationsteknik. Målet är att utveckla en framtida applikation för rumsbokning. Det skulle därför uppskattas om ni tog er tid att svara på några frågor.

Information om applikationen:
Tanken är att applikationen ska installeras på surfplattor/tablets som sedan monteras på väggen invid alla mötesrum. Med applikationen ska anställda sedan kunna boka respektive mötesrum. Användare skall även kunna boka via sitt normala bokningssystem, t.ex. google kalender eller Microsoft Exchange vilket synkas till plattorna.

Hur ofta bokar du ett mötesrum?

- ◯ Mindre än 1 gång per vecka
- ◯ 1-2 ggr/vecka
- ◯ 3-4 ggr/vecka
- ◯ 5 eller fler gånger i veckan

När skulle du säga att majoriteten av dina bokningar främst görs...

- ◯ ... precis innan mötets start
- ◯ ... tidigare samma dag
- ◯ ... tidigare i veckan eller månaden

Hade du uppskattat att kunna boka ett mötesrum på plats?
Via surfplattan på väggen

- ◯ Ja
- ◯ Nej
- ◯ Vet inte

Hade du uppskattat att på avstånd kunna se ett mötesrums status (Ledigt eller Upptaget)?
Status indikerad av surfplattan

- ◯ Ja
- ◯ Nej
- ◯ Vet inte

Vilka av följande funktioner värdesätter du i ett bokningssystem på surfplattan?

- ☐ Att kunna göra en bokning enkelt
- ☐ Att kunna se rummets schema för dagen
- ☐ Att kunna boka andra rum
- ☐ Att kunna se information om rummet såsom antal stolar och annan utrustning (t.ex projektor)
- ☐ Att kunna hitta ett annat ledigt rum

Hur långt i förväg vill du kunna boka ett mötesrum på plats?
Via surfplattan på väggen

- ◯ Endast direktbokning
- ◯ Samma dag
- ◯ Samma vecka
- ◯ Längre fram

Hur stor del av dina möten avbokar du?

- ◯ 0-25 %
- ◯ 26-50 %
- ◯ 51-75 %
- ◯ 76-100 %

Hade du uppskattat möjligheten att kunna avboka en framtida bokning på plats?
Via surfplattan på väggen

- ◯ Ja
- ◯ Nej
- ◯ Vet inte

Hade du uppskattat möjligheten att kunna avsluta en pågående bokning på plats?
Via surfplattan på väggen

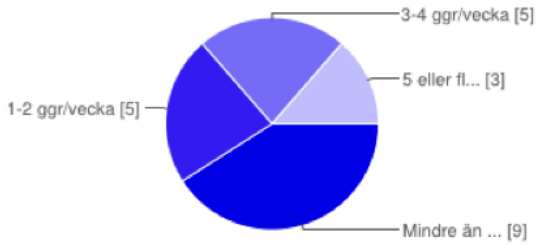
- ◯ Ja
- ◯ Nej
- ◯ Vet inte

Hade du uppskattat att kunna förlänga en pågående bokning på plats?
Via surfplattan på väggen

- ◯ Ja
- ◯ Nej
- ◯ Vet inte

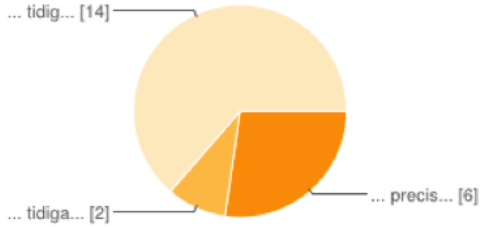
Bilaga E – Enkätundersökning – svar

Hur ofta bokar du ett mötesrum?



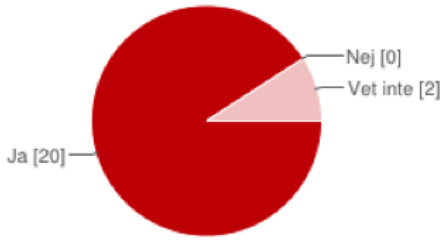
Mindre än 1 gång per vecka	9	41%
1-2 ggr/vecka	5	23%
3-4 ggr/vecka	5	23%
5 eller fler gånger i veckan	3	14%

När skulle du säga att majoriteten av dina bokningar främst görs...



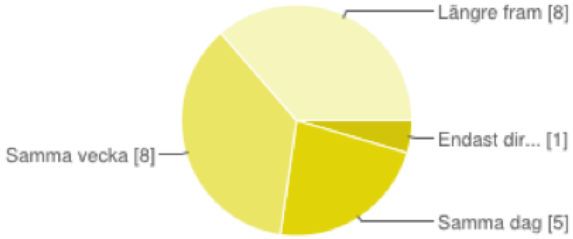
... precis innan mötets start	6	27%
... tidigare samma dag	2	9%
... tidigare i veckan eller månaden	14	64%

Hade du uppskattat att kunna boka ett mötesrum på plats?



Ja	**20**	91%
Nej	**0**	0%
Vet inte	**2**	9%

Hur långt i förväg vill du kunna boka ett mötesrum på plats?

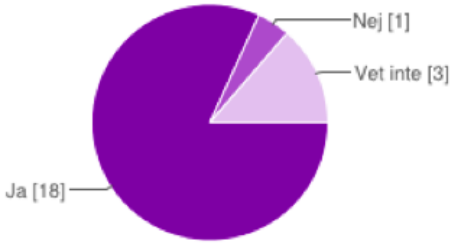


Endast direktbokning	**1**	5%
Samma dag	**5**	23%
Samma vecka	**8**	36%
Längre fram	**8**	36%

Hur stor del av dina möten avbokar du?

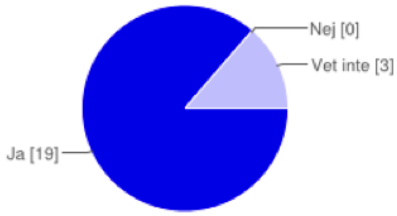
0-25 %	**22**	100%
26-50 %	**0**	0%
51-75 %	**0**	0%
76-100 %	**0**	0%

Hade du uppskattat möjligheten att kunna avboka en framtida bokning på plats?



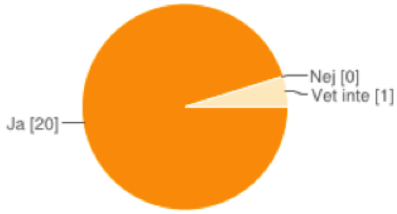
Ja	**18**	82%
Nej	**1**	5%
Vet inte	**3**	14%

Hade du uppskattat möjligheten att kunna avsluta en pågående bokning på plats?



Ja	**19**	86%
Nej	**0**	0%
Vet inte	**3**	14%

Hade du uppskattat att kunna förlänga en pågående bokning på plats?



Ja	**20**	95%
Nej	**0**	0%
Vet inte	**1**	5%

Bilaga F – Testplan

Testerna utfördes på en på en grupp med olika erfarenheter av liknande system och gjordes med ramverket DECIDE, vilket förklaras i detalj nedan.

Rogers, Sharp & Preece (p. 456 2011) har gjort ett ramverk kallat DECIDE för att planera tester av användargränssnitt. Ramverket är uppdelat i sex olika steg som består av: 'determining the goal', 'explore the questions', 'choosing the evaluation method', 'identify the practical issues', 'decide how to deal with the ethical issues', och 'evaluate, analyze, interpret, and the data' (Rogers, Sharp & Preece, 2011, p. 456).

Determine the goals
Målet med testerna är att se huruvida applikationens användargränssnitt är lätthanterligt och möjliggör bokning med få knapptryck.

Explore the questions
Hur lätt användaren uppfattar att det är att göra en bokning? Hur lätt användaren uppfattar att det är att avsluta en pågående bokning? Hur lätt användaren uppfattar att det är att avsluta en pågående bokning? Hur gör användaren för att få information om rummet?Hur gör användaren för att byta språk?Hur gör användaren för att undersöka om det finns en bokning senare under samma dag?Hur lyckas användaren ta sig tillbaka till första vyn från 'förlänga boknings-vyn'?Vilka misstag gör användaren?

Choose the evaluation method
Testerna av applikationen genomförs i en naturlig miljö för att skapa en verklighetskänsla för användaren. Det för att användaren skall känna sig bekväm och för att få en realistisk bild av användarbetendet under testets gång (Rogers, Sharp & Preece, 2011, p. 491).

I testet så kommer metoden think-aloud att användas. Det för att få användarens intryck under genomförandets gång. Think-aloud är en metod som bygger på att användaren berättar vad hen tänker vid varje steg i testet. Det för att säkerställa att användargränssnittet är tydligt och lätt att ta till sig samt att användaren berättar när hen upplever något problem.

Decide how to deal with the ethical issues
De som kommer att utföra testerna är helt anonyma och ingen personlig information kommer inte krävas.

Evaluate, analyze, interpret, and the data

Datan från testerna kommer främst ses som kavlitativ. Den kommer inte att statistiskt analyseras då testantalet är lågt och frågorna inte lämpar sig för det. Eftersom testerna utgörs av minst fem stycken så är kvantiteten tillräcklig (Rogers, Sharp & Preece, 2011, p. 477).

Referenser

Rogers, Y., Sharp, H., & Preece, J. (2011) Interaction design: beyond human-computer interaction. 3rd ed. Chichester: John Wiley & Sons.

Bilaga G – Tester

Test 1
Testperson: Erik
Tidigare erfarenhet: Hade tidigare erfarenhet av systemet Evoko och även insatt i projektet.

Hur lätt användaren uppfattar att det är att göra en bokning?
Lätt, bristen på alternativ är bra då det inte finns så många fel att göra. Man kan utan problem använda uteslutningsmetoden för att ta sig framåt.
Hur lätt användaren uppfattar att det är att förlänga en pågående bokning?
Självklart då det är så likt att göra en helt ny bokning.
Hur gör användaren för att få information om rummet?
Trycker på informationsknappen som är lätt att hitta i och med att användaren jobbat med alla andra ytor och på så vis kan utesluta stora delar av skärmen.
Hur gör användaren för att byta språk?
Trycker direkt på språkknappen i menyraden.
Var det något vid något av testerna som var förvirrande? (till exempel hitta bokningsknappen)
Slidern var lite otydlig, men det är bara första gången man använder applikationen, därefter är det inget problem.
Var det något som du förväntade dig av användargränssnittet? (men som saknades)
Någon slags animering på slidern när den dyker upp för att tydligare visa att den finns där och vad den ska användas till. Men som sagt bara ett problem första gången. Man skulle kanske låta sluttiden följa med ovanför sliderpluppen som man flyttar för att ställa in tiden.**Hur lätt användaren uppfattar att det är att göra en bokning?**
Lättare än Evokos lösning: 4 på en 5-gradig skala (måste finnas utrymme för förbättring).

Kommentar: Tänkte att man eventuellt skulle använda schemat för att göra bokningen, men knappen övertygade. Sliderns tydlighet lite tveksam. Lätt att lära sig och efter första gången är oklara saker självklara. Inställningarna var förvirrande, alla borde inte ha tillgång (Testövervakares kommentar: Login har ej hunnit implementeras). I övrigt bra, knapparna är där de förväntas vara.

Test 2
Testperson: Mimmi
Tidigare erfarenhet: Tidigare erfarenhet av Evokosystemet.

Hur lätt användaren uppfattar att det är att göra en bokning?
Förväntade sig alternativ för att välja sluttid (som på Evokos lösning).**Hur gör användaren för att få information om rummet?**
Trycker direkt på infoknappen i menyraden.

Hur gör användaren för att byta språk?
Trycker direkt på språkknappen i menyraden.
Var användargränssnittet tydligt överlag? (var det något som var oklart)
Det var simpelt, inte svårt. Lite oklart vad nuvarande klockslag är. Tänkte att man kanske kunde använt schemat för att ställa in tiden istället.
Var det något som du förväntade dig av användargränssnittet? (men som saknades)
Att skriva information om bokningen när man gör den. Även lägga till information till en pågående bokning, t. ex. att mötet blivit inställt, då det känns konstigt att bara ta bort det.**Var det lätt att hitta avsluta-knappen?**
Ja, logiskt att den låg till vänster och förlängaknappen till höger.**Var det lätt att förstå schemat?**
Ja men vill gärna trycka på schemat för att göra en senare bokning**Hur lätt uppfattar användaren att det är att göra en bokning?**
4 på en 5-gradig skala.

Kommentar: Tydligt gränssnitt, rätt klockrent, funktionellt. Knapparna logiskt placerade. Testpersonen gjorde rätt på alla uppgifter utan problem.

Test 3
Testperson: Alexandra
Tidigare erfarenhet: Ingen tidigare erfarenhet av liknande system.

Hur lätt uppfattar användaren att det är att göra en bokning?
Tydligare slider! Finns också ganska många visningar av klockslag så lite förvirrande vilken som visar vad. Tydligare visning av nuvarande klockslag. Slidern gör att man förväntar sig ett sekundintervall, visa spannet tydligare. Lite oklart att det är hela kvartar man ställer in.**Hur lätt användaren uppfattar att det är att avsluta en pågående bokning?**
Lätt men vill ha en form av bekräftelse, eller godkännande.
Hur lätt användaren uppfattar att det är att förlänga en pågående bokning?
Mer information om hur nuvarande bokning ser ut. Kändes som att man skrev över/pajade pågående bokning när man skulle förlänga.
Hur gör användaren för att få information om rummet?
Trycker direkt på infoknappen i menyraden.
Hur gör användaren för att byta språk?
Trycker direkt på språkknappen i menyraden, men lite oklart med vad flaggan visar. Visar den aktuellt språk eller språket den byter till?
Hur lyckas användaren ta sig tillbaka till första vyn från 'förlänga boknings-vyn'?
Med hjälp av hemknappen (Det gjordes dock från inställningsmenyn). Innan hemknappen hittades gjordes några försök med bakåt-systemknappen.
Vilka misstag gör användaren?
Använder systemknapparna för att ta sig hemmåt/tillbaka i applikationen.**Var det något som du förväntade dig av användargränssnittet? (men som saknades)**

Rummets namn kunde varit tydligare, förslagsvis lika stort som "ledigt till"-texten**Var det lätt att förstå schemat?**
Tiderna gör att man förstår att det går att scrolla. Förslagsvis önskar testpersonen kunna se vilken veckodag det är.
Hur lätt användaren uppfattar att det är att göra en bokning?
4 av 5

Kommentar: Överlag bra. Lite onödigt små menyknappar. Blev förvirrad av systemknapparna i nederkant.

Test 4

Testperson: Viktor
Tidigare erfarenhet: Tidigare erfarenhet av liknande system.

Hur lätt användaren uppfattar att det är att göra en bokning?
Kanske kunna använda schemat istället men rätt enkelt att göra en bokning. Sätt någon bättre standardtid istället för närmsta kvart. Kanske ha möjlighet att trycka på schemat och få en popup som bokar en timme frammåt vid bekräftelse.
Hur lätt användaren uppfattar att det är att avsluta en pågående bokning?
Lätt men saknar ett tack eller bekräftelse när man avslutat ett möte istället för att det bara bara försvinner/kapas och rummet blir ledigt.
Hur gör användaren för att få information om rummet?
Trycker direkt på infoknappen i menyraden.
Hur gör användaren för att byta språk?
Trycker direkt på språkknappen i menyraden.
Hur gör användaren för att undersöka om det finns en bokning senare under samma dag?
Användaren använder sig av schemat och läser statustiden.
Vilka misstag gör användaren?
Testpersonen försökte använda systemknapparna för att gå hem, men hittade därefter hemknappen.**Var det lätt att hitta avsluta-knappen?**
Ja.
Var det lätt att förstå schemat?
Hade velat att det gick att se saker om man trycker på en bokning i schemat, hellre röd färg, kanske info i bokningsrutan
Hur lätt användaren uppfattar att det är att göra en bokning? 5 av 5, väldigt snabbt, mycket snabbare än Evokos lösning.
Hur lätt användaren uppfattar att det är att avsluta en pågående bokning?
4-5av 5
Hur lätt användaren uppfattar att det är att förlänga en pågående bokning?
4-5 av 5

Kommentar: Gillade att man såg några timmar framåt i schemat. Byt plats på menyknapparna och klockan och lägg hemknappen till vänster som på en hemsidas menyrad. Ha snabbvalsalternativ bredvid slidern. Eller snabbval för en timmes möte genom att trycka på schemat.

Bilaga H – Användningsfall / Use-cases

Hög prioritet

Konfigurera Applikationsinställningar
Administratören vill konfigurera en applikations inställningar. **Applikationen** ber om information om rummet och konto. **Applikationen** ställer in rummet för applikationen, och påvisar för **administratören** att rummet är inställt.

Kontrollera ett rums status
Användaren vill se om ett rum är bokad eller ledigt. **Applikationen** visar via en färg eller text statusen på **rummet**.

Boka detta rum (omedelbart)
Användare önskar boka det rum som surfplattan hör till, omedelbart. **Användare** behöver endast förmedla denna avsikt till surfplattan, som bokar rummet (om ledigt). **Användare** får alternativet att ge bokningen en beskrivning.

Förläng en pågående bokning
Användaren vill förlänga en pågående bokning. **Användaren** specificerar en tid att förlänga bokning med. **Applikationen** förlänger bokningen och påvisar för **användaren** om förlängningen.

Avsluta en pågående bokning
Användaren vill avsluta en pågående bokning tidigare. **Användaren** väljer att avsluta den pågående bokningen. **Applikationen** bekräftar avslutningen.

Se information om en bokning
En **användare** vill se information rörande en bokning. Till exempel se vem som gjort bokningen, hur länge en bokning gäller med mera.

Medel prioritet

Se framtida schema
Användaren vill se ett schema för ett rums status en tid framöver.

Hitta lediga rum i närheten till surfplattan
Användaren står vid ett rum som är upptagen och vill se vilka lediga **rum** som finns i närheten. **Applikationen** presenterar vilka **rum** som är lediga och finns i närheten.

Boka andra rum

Användaren vill boka ett annat rum än den som hör till surfplattan.

Kontrollera information om ett rum
Användaren vill se information om ett rum (ex: antal platser, utrustning). **Applikationen** presenterar informationen till **användaren**.

Felanmäla
Användaren vill felanmäla något som är trasigt eller fel i rummet. **Användaren** informerar **Systemet** om vad som är fel och gör en felanmälan.

Ändra språk
Användaren vill byta språk på applikationen. **Applikationen** byter ut systemspråket.

Låg prioritet

Logga in
En **användare** eller en **administratör** skall kunna logga in för att koppla en bokning till sin användare eller administrera inställningar. **Applikationen** loggar in användaren genom en identifikationssekvens.

Logga ut
En **användare** eller **administratör** som är inloggad vill logga ut från kontot. **Systemet** loggar ut användaren och påvisar detta.

Boka detta rum i framtiden
Användare önskar boka det rum som surfplattan hör till, från en tidspunkt i framtiden. **Användare** specificerar tidspunkten, samt hur länge **användare** vill boka rummet. Efter bekräftelse från **användare** så bokas rummet.

Avboka detta rum i framtiden – inloggad
Användaren önskar att avboka ett rum på plats via surfplattan. **Användaren** specificerar vilken bokning som avses och anger att bokningen av **rummet** skall avbokas. **Systemet** tar bort bokningen.

Ändra en framtida bokning
Användaren vill ändra en bokning. **Användaren** väljer att ändra tiden för en bokning. **Systemet** ändrar tiden och bekräftar bokningens ändring.

Se hjälpavsnitt
Användaren vill se hjälpavsnitt om **applikationen**. **Applikationen** presenterar hjälpavsnitt.

Bilaga I – Diagram över ResourceeActivity samt ärvande klasser