Managed Operations Software

Schema och rapportering – en prototyp i form av en Androidapplikation

LUNDS UNIVERSITET
Lunds Tekniska Högskola

LTH Ingenjörshögskolan vid Campus Helsingborg
Datateknik

Examensarbete:
Iyad Abu-sneineh
Bahaderjon Ahunov

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

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

Tryckt i Sverige
Media-Tryck
Biblioteksdirektionen
Lunds universitet
Lund 2012

Sammanfattning

DISA är ett internationellt företag i gjuteribranschen. De bygger och utvecklar gjuterimaskiner till gjuteriföretag runt om i världen och hjälper även sina kunder att tillhandahålla servicen på maskinerna.

Idag har företagets kunder inget väl fungerande system för felrapportering, arbetsfördelning och dokumentation av service. Detta problem ledde till utvecklingen av ett servicesystem som underlättar och effektiviserar arbetet med underhåll av gjuterimaskinerna.

En mobil applikation för Android behövdes för att kunna ta emot uppgifter som servicechefen i ett företag vidarebefordrar till supportpersonal vilken de via applikationen kan rapportera användbar data för en specifik uppgift och felrapportera om det behövs.

Målet med examensarbetet var att utveckla en prototyp av denna applikation i Android.

Nyckelord: Android, applikation, DISA, servicesystem.

Abstract

DISA is an international company in the foundry industry. They build and develop foundry machinery to foundry companies around the world. They also help its customers to provide service on the machines.

Today, their customers have no effective system for reporting errors, work and documentation of service. This problem led to the development of a service system that facilitates and streamlines the process of maintenance of foundry machines.

The mobile application for the Android-devices should receive tasks from the service manager in a company who then forward it to the maintenance personnel. The maintenance personnel should through this application be able to report useful data for a specific task and send an error report if necessary.

The aim of our study was to develop a prototype of this application in Android.

Keywords: Android, application, DISA, service system

Förord

Detta examensarbete har utförts i samarbete med DISA Industries A/S under 2012 som avslutande del av vår utbildning som högskoleingenjörer vid Ingenjörshögskolan i Helsingborg på Högskoleingenjörsprogrammet i Datateknik.

Utvecklingen av denna applikation har varit väldigt lärorik. Det är kul att lära sig att programmera inom någonting nytt som Android. Det tog tid att anamma de teoretiska kunskaperna som sedan skulle användas praktiskt.

Vi vill tacka alla som varit till hjälp under detta examensarbete och inte minst vår handledare på Disa, Nils Assarsson. Vi vill även passa på att tacka vår examinator, Mats Lilja, som hjälpt till med tips och idéer under examensarbetet och den stora hjälp han gett oss under alla dessa år under utbildningen.

Helsingborg, november 2012.

Iyad Abu-sneineh
Bahaderjon Ahunov

Innehåll

1 Inledning

1.1 Översikt

Detta kapitel är en inledning till examensarbetet. Här beskriver vi bakgrunden till själva examensarbetet och varför det är viktigt för Disa. Vad vi ska leverera till Disa och avgränsningar behandlas även här.

1.1.1 Bakgrund

DISA är ett danskt företag inom gjuterimaskinbranschen som utvecklar egna gjuterimaskiner för järn- och aluminiumproduktion för den internationella gjuteriindustrin. De erbjuder leverans, installation samt service av maskinerna. Idag har servicen av maskinerna blivit den viktigaste inkomstkällan för DISA. Gjuterimaskinerna som DISA levererar har oftast en livslängd på minst 20 år. Därför blir det inte så mycket leveranser av maskiner, på så sätt att man måste köpa in nya maskiner med korta mellanrum.

Därför har försäljning av reservdelar och service av maskinerna blivit mer och mer viktigt för DISA. Något som blivit extra viktigt för företaget på senare tid är att just effektivisera servicen av maskinerna hos kunderna. Man skickar idag ut servicetekniker till kunderna för att lösa uppkomna problem och för att serva maskinerna. Just nu ger man service först efter att felet uppstått. Detta leder ofta till att man behöver avbryta maskinerna tills man åtgärdat felet vilket leder till att dyrbar tid går förlorad.

Ett stort fel som DISAs kunder gör idag är att de åtgärdar fel på maskinerna först efter att de upptäcker att något gått fel. Detta är en följd av att de inte servar maskinerna från början enligt de instruktioner som DISA givit dem.

För att göra det hela mer tidsbesparande och kostnadseffektivt vill nu DISA erbjuda sina kunder ett servicesystem som ska hålla reda på vad som behöver göras efter att ett visst antal gjutningar utförts av maskinerna. Dessa uppgifter skall sedan servicechefen på företaget skicka ut till personalen så att de vet vilka åtgärder som krävs. Idag blir nämligen saker och ting helt enkelt inte gjorda just för att man inte har något strukturerat system att gå efter. När man sedan inte servar maskinerna så leder det ofta till andra fel i maskinerna som DISA måste åtgärda.

1.1.2 Syfte och målsättning

För att lösa dessa problem vill DISA utveckla detta servicesystem som ska användas av deras kunder. Servicechefen på företaget ska då fördela olika arbetsuppgifter till de anställda på företaget genom en hemsida som sedan skickar ut de olika uppgifterna till en Android–applikation som de anställda använder. Applikationen ska alltså användas som ett schemasystem där användaren ser ett schema som ska följas.

Applikationen ska vara enkel att använda och ska endast omfatta de funktioner som behövs. Användaren ska i princip kunna se sitt schema och felrapportera när det behövs.

Målet är att utveckla en applikation som kommunicerar med den relaterade hemsidan genom en gemensam databas.

Dessa krav var formulerade efter uppdragsgivarens önskemålom applikationen:

1. Applikationen ska vara enkel och användarvänlig.
2. Applikationen ska vara på engelska, men det ska finnas språkstöd.
3. Arbetsuppgifterna i schemat ska vara sorterade efter tid. Det som ska göras först ska alltså synas längst upp i listan.
4. Applikationen ska dels visa tidpunkt för när en arbetsuppgift ska göras, men när man är klar med en arbetsuppgift ska även tidpunkten för när man väl började utföra uppgiften och när man var klar med den sparas.
5. Man ska kunna felrapportera genom att ta en bild med mobilen för att snabbt och enkelt skicka iväg den till den man önskar via exempelvis e-post.
6. Man ska även kunna spara sina egna kontakter i applikationen. Dessa ska man även kunna ta bort, redigera, kunna ringa och skicka e-post till dessutom.
7. Applikationen ska vara kopplad till samma yttre databas som hemsidan.

1.1.3 Avgränsningar

När detta system är komplett ska man kunna rapportera och överföra information mellan hemsidan och applikationen. Men eftersom hemsidan inte är riktigt färdig och är under utveckling så kommer det bli en utmaning att både utveckla applikationen och erhålla en hållbar kommunikation mellan oss och de som utvecklar hemsidan. Vårt mål är att få detta att fungera, men om omständigheterna kräver det kommer vi istället att utveckla en prototyp som simulerar data med en intern databas.

DISA vill främst visa sina kunder hur applikationen kommer att fungera och att det faktiskt fungerar. I framtiden kommer ju applikationen att kommunicera med en yttre databas som hemsidan som servicechefen kommer använda även kommer göra.

Språkstöd ska finnas men applikationen ska för närvarande endast vara på engelska och mer funktioner utöver de som finns i kraven kommer att läggas till vid vidareutvecklingen. Just nu avgränsar vi oss till att endast utveckla de viktigaste

funktionerna och det gör vi genom att uppfylla kraven som uppdragsgivaren specificerar.

2 Arbetsmetod

2.1 Översikt

Detta kapitel beskriver vår arbetsmetod vilket innefattar utvecklingsmodell, testning, implementering och informationsinhämtning.

Här beskriver vi även hur vi gått tillväga för att utveckla en prototyp som stämmer överrens med kundens förväntningar.

2.2 Utvecklingsmodell

Under utvecklingens gång har det inte strikt följts någon specifik traditionell utvecklingsmodell, dels för att anpassa oss till projektet och dels för att manarbetar bäst och effektivast på det sätt man är van att arbeta vid. Man kan ändå säga att modellen som följdes är en iterativ metod. I princip användes följande modell under utvecklingens gång:

1. Undersökning av problemet som Disa upplever och varför de vill ha en applikation och hur de vill ha den.

2. När vi sedan förstod problemet använde vi oss av en prototyp för att förstå och analysera vad kunden förväntade sig av oss. Denna prototyp byggde vi vidare på vår slutgiltiga prototyp och kunde dessutom lägga till lämpliga funktioner som önskades. Det som inte behövdes togs bort.

3. Utveckling av designen.

4. Implementera funktionalitet till designen.

5. Testning.

Om det upptäcktes att något inte fungerade som det skulle, gick vi tillbaka till punkt 4 för att felsöka problemet och implementera en fungerande funktion.

Testningen var kontinuerlig. Allteftersom något nytt lades till i applikationen testades det på nytt. Vi jobbade väldigt iterativt på så sätt att utvecklingen och testningen var

kontinuerliga. Utvecklingen fortsatte inte förrän testning av den skrivna kodenutfördes och så att man på så sätt kunde vara säker på att det fungerade som det skulle.

2.3 Arbetsmetoder och källkritik

Arbetsmetoderna som använts under projektet var följande:

1. Litteratursökning
2. Kundkontakt
3. Utveckling
4. Testning
5. Verifiering.

Android är ganska nytt men vi har i huvudsak använt oss av följande två böcker som varit till hjälp:

1. Beginning Android 4 av Grant Allen, 2012, Apress.

2. Android for programmers av Paul J. Deitel, Harvey M. Deitel och Abbey Deitel, 2011, Prentice Hall.

Annars har vi använt oss av internet som informationskälla och i huvudsak när det kommer till utvecklingen. Vi har inte använt oss av en internetkälla förrän vi varit säkra på att det som står där är korrekt. Detta gör man bland annat genom att antingen själv testköra en kod som man hittar på nätet och se att den verkligen fungerar och har det med mer teoretiska områden att göra så undersöktes andra källor för att man därför kunde vara säker på att de bekräftar samma sak.

Kontakt med kunden har vi haft ett par gånger för att säkerställa att vi är på rätt spår. Då har vi dels visat upp en prototyp och även haft vanliga möten och diskuterat hur applikationen ska fungera.

Eftersom vi är två personer så har vi under utvecklingen både utvecklat tillsammans på en dator och ibland har vi även delat upp utvecklingen i olika delar där var och en av oss sköter sitt område.

Under testningen har vi noggrant testat att den skrivna koden fungerar som den ska genom att undersöka om eventuella buggar finns i programmet.

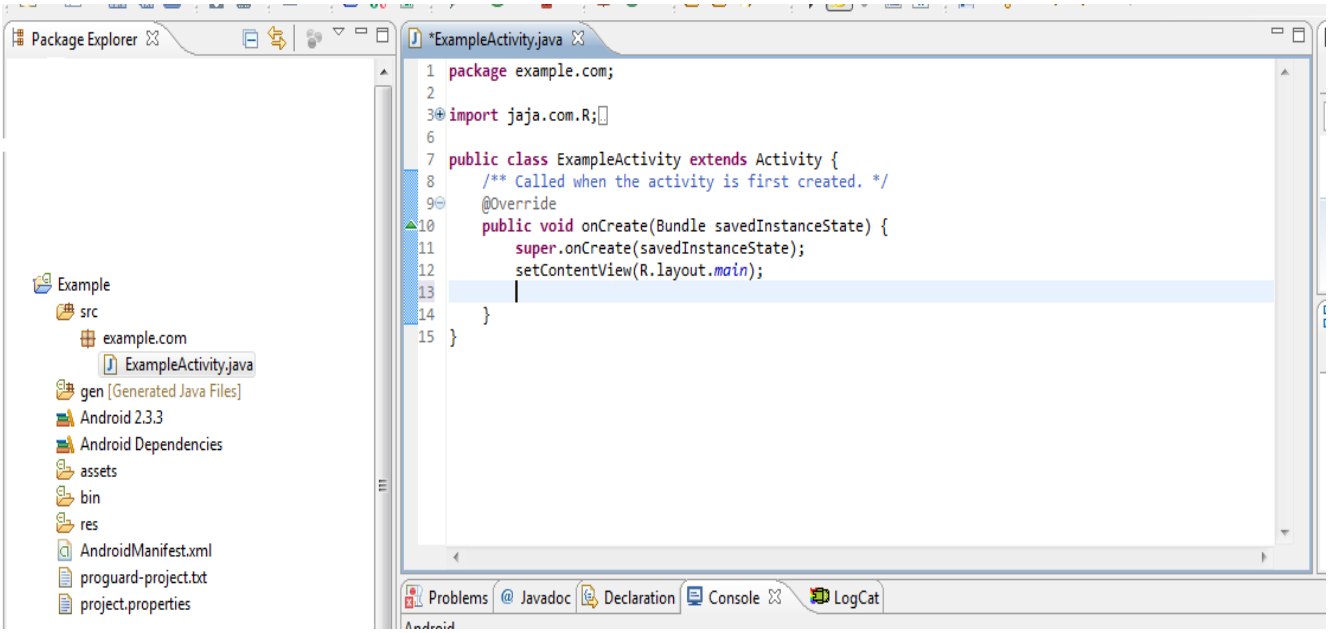
Till sist försöker vi analysera om det vi utvecklat verkligen stämmer överrens med det som DISA vill ha i applikationen och då försöker vi hela tiden anpassa applikationen till deras förväntningar.

2.4 Utvecklingsmiljö

Applikationen utvecklar vi i Eclipse. Det är det vanligaste alternativet och det vi fann enklast att arbeta i. Det är ju dessutom helt gratis att använda.

När man ska köra ett androidprojekt så är filerna indelade i olika mappar. Man kan dela in det i Javafiler, XML-filer och övriga filer. Övriga filer kan vara allt från bilder, ljudfiler och annat.

Man lägger in det i separata mappar och det ser ut såhär när du skapar ett nytt projekt:



Figur 2.4: *Ett nytt androidprojekt*

I *src-mappen* har man som vanligt Javafilerna. De andra mapparna som *res – mappen* är till för XML – filerna och andra användbara filer man kan behöva nå i applikationen såsom bilder och ljud.

När man utvecklar androidprojekt är det vanligast att man designar layouten i XML–filerna för att sedan sköta själva funktionaliteten i Javakod.

3 Teknisk bakgrund

3.1 Java

Java är ett objektorienterat programspråk som konstruerades år 1991-1995 av bland annat James Gosling. Java är plattformsoberoende vilket innebär att det kan köras och utvecklas i olika plattformar och operativsystem. Detta uppnås genom att Java – koden kompileras till bytekod istället för maskinkod. Denna bytekod körs sedan i ett plattformsoberoende program, även kallat ”virtuell maskin”.

Det finns olika versioner av Java. Exempel på dessa är:

Java ME (Micro Edition): Detta är en Java-plattform avsedd för inbyggda system, som exempelvis mobiltelefoner och industriella styrsystem.

Java SE (Standard Edition): Denna version används främst för skrivbords– och klientprogram.

Java EE (Enterprise Edition): Det vanligaste tillämpningsområdet för tillfället. Denna version består av inbyggda ramverk som löser vanliga problem i exempelvis avancerade serverapplikationer. [1]

3.2 XML

XML står för eXtensible *MarkupLanguage* och används för att bearbeta data. Det är ett uppmärkningsspråk och liknar till stor del HTML. Man använder sig av taggar i båda språken där taggarnas placering i koden har en innebörd. En skillnad dock mellan HTML och XML är att HTML används för att presentera data på skärmen med fokus på hur det ser ut medan XML mer används för att ”hålla uppe” data. [2]

```
23      <TextView
24          android:id="@+id/textPassword"
25          android:layout_width="wrap_content"
26          android:layout_height="wrap_content"
27          android:text="Password"
28          android:textAppearance="?android:attr/textAppearanceMedium" />
29
30      <EditText
31          android:id="@+id/password"
32          android:layout_width="match_parent"
33          android:layout_height="wrap_content"
34          android:ems="10"
35          android:inputType="textPassword" />
36
37      <Button
38          android:id="@+id/login"
39          android:layout_width="wrap_content"
40          android:layout_height="wrap_content"
41          android:text="Login" />
```

Figur 3.2.1: *Ett kodexempel*

Ovanstående kodexempel som är en del av applikationen består av en text som heter *Password,* ett textfält där man skriver in sitt lösenord samt en knapp som heter *Login.* Ovanstående kod ger följande gränssnitt:

Figur 3.2.2: *Gränssnittet*

3.3 SQLite

SQlite är en inbyggd SQL – databasmotor. Till skillnad från andra databaser så har SQLite inte någon separat serverprocess som de flesta SQL databaser använder.

Databaser som är implementerade enligt serverprocessmetoden fungerar som så att kommunikationen från och till databasservern sker genom en interprocess oftast med TCP/IP som protokoll. SQLite skriver och läser direkt till en specifik fil på hårddisken utan någon separat mellanprocess.

Det finns både fördelar och nackdelar med att använda en databas utan någon som helst serverprocess. En av föredelarna är att man slipper installera och konfigurera massor av olika filer bara för att sätta upp en server. Det är både sparsamt och effektivt för mindre projekt. Nackdelen är att en icke serverprocess databas är mindre säker och framförallt så uppstår det mer buggar i realtid (concurrency på engelska) vid hantering av större datamängder.

I Android är SQLite ett inbyggt klassbibliotek. Därför är det vanligaste att man använder SQLite när man utvecklar i Android.

Fördelen med SQLite är att den är väldigt snabb och sparsam när det kommer till utrymme, vilket gör denna databasmotor till ett optimalt alternativ för applikationer till mobila enheter. [3]

3.4 Android

Android är ett mobilt operativsystem. Det är ett öppet och gratis operativsystem som numera ägs av Google sedan 2005. Innan dess ägdes den av företaget Inc. Företaget köptes sedan upp av Google.

Operativsystemet körs på många olika enheter som mobilenheter, surfplattor och även TV–apparater numera. Företag som HTC, Sony och Samsung använder Android som operativsystem vilket gör operativsystemet bland de största på marknaden.

Den första generationen av Android släpptes i oktober 2008. Försäljningen ökade enligt Gartner med 707% det första kvartalet år 2010 jämfört med samma period föregående år. Under mars 2010 hade Android37% av marknaden när det kommer till smartphones, jämfört med Apples iPhone som endast hade 27% och Blackberry som hade 22%. I augusti 2010 aktiverades över 200 000 Androidtelefoner varje dag. Endast två månader tidigare låg denna siffra på ungefär 100 000. I Juni 2011 hade denna siffra ökat till 500 000.

Det som är bra med att utveckla i Android är att plattformen är öppen och operativsystemet är *open source* och gratis. Programmeringsspråket som oftast används är Java.

Det finns olika versioner av Android. Bland de första versionerna som släpptes var version 1.5 (Cupcake) som släpptes i april 2009, sedan 1.6 (Donut) som släpptes i september samma år och så vidare. Den version som är störst i marknaden just nu är version 2.3 (Gingerbread) som släpptes i december 2010. I februari 2011 släppte

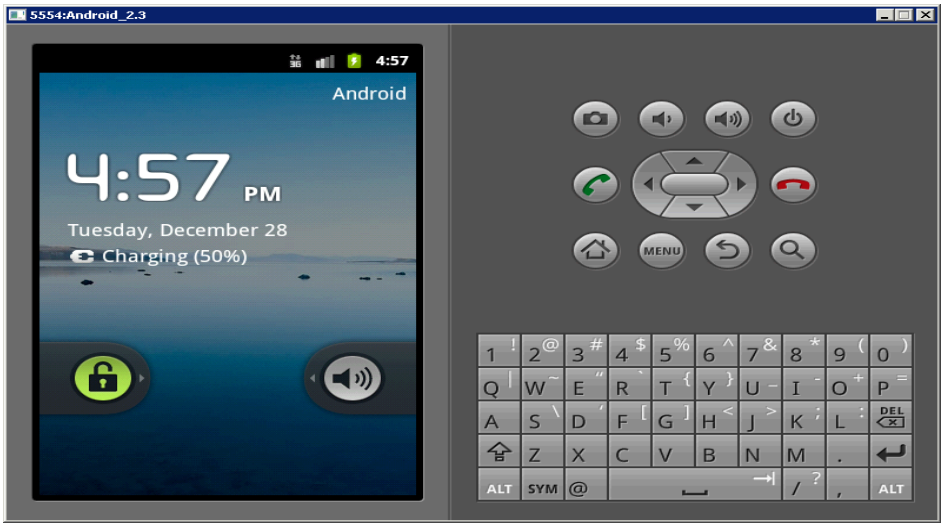
man version 3 (Honeycomb) som är en version enbart för surfplattor. Version 4 (Ice Cream Sandwich) som släpptes i oktober 2011 är en version för alla Androids enheter, vare sig det rör sig om surfplattor eller telefoner. [4]

4 Genomförande och utveckling

4.1 Bli bekanta med Android

Efter att ha laddat ner Eclipse var det första som gjordes att ladda ner och installera nödvändiga paket för Androids utvecklingsmiljö. Det som sedan gjorde var att vi testade koda enkla småprojekt som var relevanta för detta projekt som exempelvis en enkel GUI med fungerande inloggningsfunktion, vilket även användes i applikation som utvecklades i detta projekt. På så sätt kom man igång att lära sig programmera i Android.

När man utvecklar ett Android – projekt kan man alltid testköra applikationen på datorn med hjälp av en emulator.



Figur 4.1: *Emulator*

Det finns dock ett annat alternativ och det är att köra applikationen på ens egen Androidtelefon. Det är bättre av flera anledningar. För det första går det mycket snabbare att köra programmet på ens egen telefon medan emulatorn först måste initiera vilket tar en viss tid. Sedan fungerar inte alla funktioner som e-post funktioner, kamera och samtal på emulatorn. Vill man därför vara säker om man programmerat rätt så får man prova applikationen på telefonen.

4.2 Prototyp

Vi började med att skissa själva designen först så som vi uppfattade kundens behov i form av en bildprototyp utan funktionalitet. Därefter visade vi denna prototyp

förkunden som då fick bättre insyn på vad som var nödvändigt respektive onödigt att ha med i applikationen. På så sätt underlättade det för oss under utvecklingensgång eftersom man fick bättre förståelse för kundens krav och önskemål.
Figurerna 4.2.1 och framåt illustrerar vår första prototyp med kort förklaring.

Figur 4.2.1 Prototypen visar gränssnittet innan och efter inloggning. Med fliken ”Schedule” som aktiv. Här ser man sina arbetsuppgifter direkt efter inloggning.

Figur 4.2.2: Bilden illustrerar hur man går tillväga för att rapportera.

Figur 4.2.3: Bilden till vänster visar kontaktuppgifter till de som man lagt till. Den högra bilden visar möjligheten att ta en bild till rapportering.

4.3 Kravhantering

Efter att ha visat upp prototypen för DISA fick vi många förslag och önskemål som lade grunden för utvecklingen utav applikationen. Vi fick bland annat reda på följande:

1) Schemat ska vara sorterat efter tid närmast deadline och inte efter antal gjutningar som gjorts. När man startar applikationen ska man alltså kunna se den arbetsuppgift som ska göras tidigast längst upp. Sedan den som skall göras nästtidigast efter och så vidare. Detta gör det enklare för användaren att prioritera det som behöver göras först och vad gäller antal gjutningar så är det endast servicechefen som är användaren av webbapplikationen som matar in hur många gjutningar som gjorts.

2) Man ska kunna utföra en arbetsuppgift tillsammans med någon annan och då skall man även kunna se vem man ska göra den med.Det är alltså inte så att en arbetsuppgift görs individuellt alltid. Hänsyn skall tagas åt att flera användare kan behöva utföra en och samma arbetsuppgift.

3) Applikationen ska kunna logga när en arbetsuppgift påbörjades och när den blev klar. Detta ska sparas i databasen och ska därför kunna visas i historiken, dock endast av servicechefen i webbapplikationen. Men användaren ska trycka på en knapp när han påbörjar en arbetsuppgift samt när han är färdig. På så sätt kan servicechefen lätt kunna se hur lång tid en viss arbetsuppgift behöver ta.

4) Man ska kunna ta ett foto för att därefter skicka den till en önskad person snabbt och effektivt. Ibland vill man nämligen inte bara rapportera en viss sak med enbart text utan man vill även kunna skicka en bild i samband med en viss felrapportering som enkelt ska kunna göras genom en knapptryckning från applikationen.

5) Det ska finnas språkstöd för applikationen eftersom den ska användas av DISAs kunder i olika delar av världen. Därför ska applikationen lätt kunna användas på flera olika språk.

6) Applikationen ska vara användarvänlig. Det ska alltså inte behöva vara någon som förklarar hur applikationen används. Några korta minuter ska räcka för att användaren ska känna sig bekväm med att använda applikationen.

Dessa krav kom som sagt efter prototyppresentationen. Kraven i kapitel 1.1.2 och dessa krav är i princip samma förutom att det vid detta senare tillfälle upptäcktes att det kommer vara ont om tid att utveckla en yttre databas till applikationen som kommunicerar med hemsidan. Därför togs detta krav bort här.

4.4 Databas

Databasen är en central bit i projektet då detta utgör hela kommunikationen mellan servicechefen och användaren av applikationen. När servicechefen tilldelar en uppgift till en person, uppdateras databasen med denna information som sedan läses i applikationen.

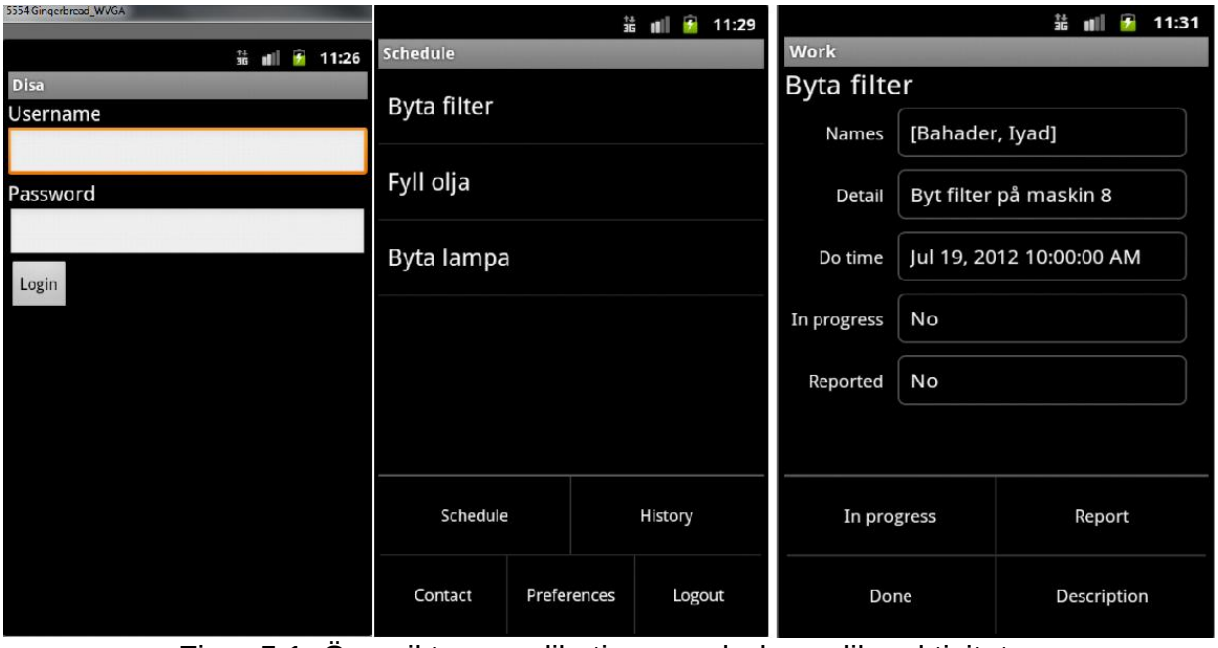
I Android är det vanligaste att man använder en intern SQLite–databas, då SQLites klassbibliotek är tillgängliga. Därför går de flesta författare till Android–böcker inte så mycket igenom hur man ansluter till en yttre databas.

Eftersom den mesta kodningen sker i Java när man utvecklar en Android applikation så tror många att det är enkelt att göra en vanlig JDBC–anslutning till en server, som man gör när man skriver ett vanligt Javaprogram. Så är dock inte fallet, då Android gjort det lite mer komplicerat. Därför var alternativet att ansluta genom en så kallad API-server och lite mer detaljer gällande detta kommer längre fram.

Eftersom hemsidan använder en SQLite databasmotor hade vi inte stora valmöjligheter, utan fick använda SQLite. Det fanns även brist på information om hur man ansluter till en yttre databas från ett androidprojekt och vi märkte att det var väldigt tidskrävande vilket gjorde att vi valde att använda en intern databas som simulerade data internt för prototypen. I framtiden kommer applikationen anslutas till en yttre databas tillsammans med hemsidan.

5 Resultat

5.1 Översikt



Figur 5.1: Översikt av applikationen och dess olika aktiviteter.

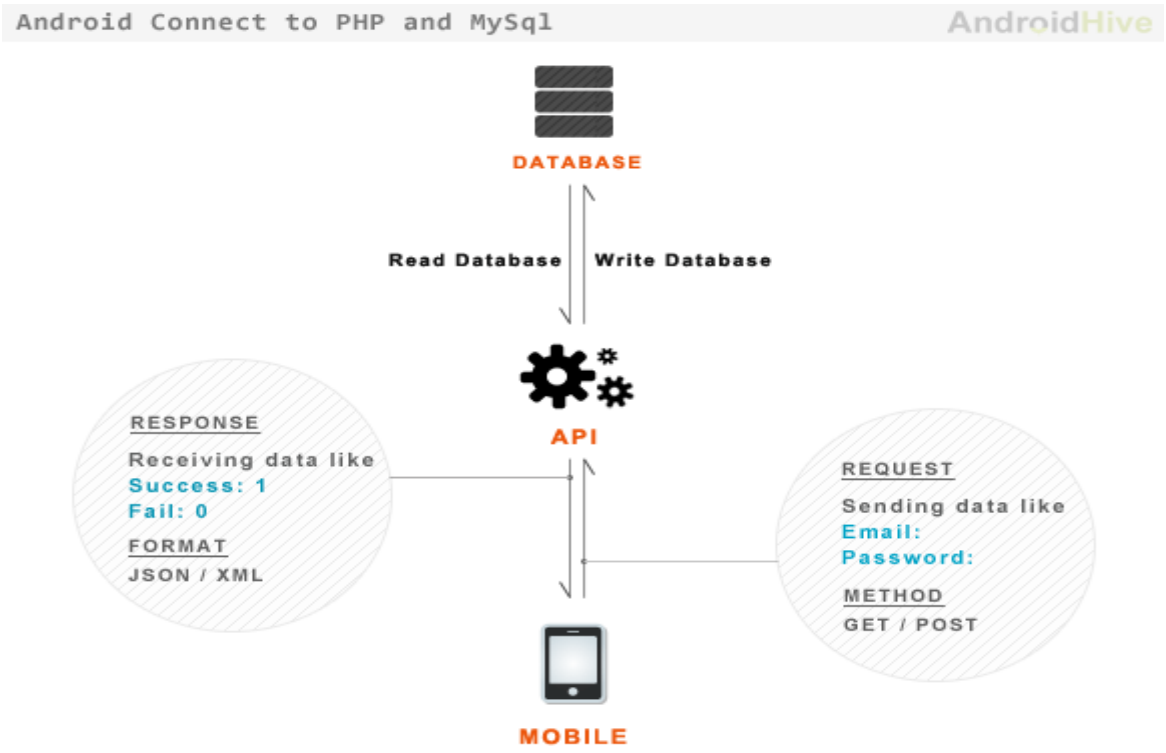
Resultatet av utvecklingen ledde till en fungerande prototyp med simulerade data från en lokal databas som Figur 3.1 visar. Den första aktiviteten man kommer till när man startar applikationen är givetvis inloggningsaktiviteten. När man har loggat in med en ”testuser” får man upp olika arbetsorder sorterad efter tidigast ”Do time” som då syftar på när arbetsordern skall påbörjas.

Man kan klicka sig in på varje arbetsorder för att ändra dess status. Det finns tre olika status för en arbetsorder:

1. *In progress* – En arbetsorder påbörjas och tidsaspekterna börjar loggas när man påbörjade ordern för att sedan utnyttjas till att se hur lång tid varje arbetsorder tar att göra.
2. *Report* – Man kan rapportera en arbetsorder men ej vara klar med den p.g.a. saknad resurs eller kompetens.
3. *Done* – Arbetsordern är färdig och åtgärdad. Då tas den bort från schemat och läggs in i historik.

5.2 Databasen

Valet av databasen var ett av problemen som vi stötte på rätt så tidigt eftersom det är där all data hämtas och sparas i vår applikation. I början var det tänkt att vi skulle använda en och samma databas som den andra gruppen för att kunna ta emot arbetsorder direkt till applikationen från servicechefens hemsida. För att kunna komma åt en extern databas från Android så behövdes en så kallad ”API”.



Figur 5.2: En grovskiss över kommunikationen mellan olika instanser.

Som extern databas valdes MySQL för att kunna använda den tillsammans med hemsidan och på så sätt synkronisera dataöverföringen
.
API står för **A**pplication **P**rogramming **I**nterface. Detta behövs när man vill komma åt en externkälla så som en databas från Android. En API används som ett interface mellan olika applikationer för att möjliggöra kommunikation. [5]

Först skrevs en API för MySQL genom att utnyttja JSON och PHP. Vi lyckades kommunicera med den externa databasen genom Android men stötte då på ett annat problem. Gruppen som hade hand om hemsidan valde att använda SQLite som hemsidans lokala databas, vilket gjorde att vi var tvungna att ändra typ av databasmotor. Nu implementerades istället en API för SQLite och därmed kunde data synkroniseras till vår applikation.

Att skriva en API för en extern SQLite databasmotor visade sig inte bara vara svårt utan även att information gällande detta saknades. Olika forum genomsöktes där användare diskuterade liknande problem och de flesta menade på att SQLite [6] inte är byggd för att användas av flera olika källor samtidigt över internet. Det kan uppstå synkroniseringsfel helt enkelt [7] och lösningen till detta är att man skriver en webservice till hemsidan för att inhämta data från den lokala databasen som används av hemsidan. Men detta är endast möjligt om hemsidan är någorlunda färdig [8].

5.2.1 Lokala databasstrukturen

Contacts	**Workorders**	**Users**	**WorkForUsers**
			work_id(ref. _id
_id	_id	username	**workOrders)**
			user_id(ref. username
Name	Title	pass	**Users)**
Email	Detail	name	
Phone	Done		
	start_time		
	do_time		
	done_time		
	Inprogress		
	reported		

Figur 5.2.1: Visar relationen mellan olika tabeller i databasen.

Vi var tvungna att skapa en egen lokaldatabas som vår applikation använde sig av vid simulering av data. Användandet av SQLite som databasvaldes därför, vilken dessutom ingår i Androids bibliotek. Ett utav kraven var att man skulle kunna se i applikationen vilka som var inne på samma arbetsorder. Svårigheter med kodningen

av denna funktion upplevdes men efter att ha kontaktat Per Andersson, som bland annat undervisar i databaser vid Lunds Universitet, kom vi fram till att man var tvungen att skapa ytterligare tabell som Figur 4.2.2 visar, nämligen ”workForUsers”. Med hjälp av den tabellen fungerade funktionen tack vare att den införde många-till-många-relationer mellan användare och arbetsorder.

Man kan fråga sig varför vi har en ”Contacts” tabell i vår databas när man redan har kontakter i sin mobil. Fördelen med detta är att det underlättar för användaren att ha jobbrelaterade kontakter i så att man lätt och effektivt hittar de man vill nå, istället för att leta bland telefonens kontakter.

6 Slutsats

Målet med examensarbetet var till en början att utveckla en applikation som direkt kommunicerar med hemsidan där servicechefen lägger upp diverse olika arbetsordar och information till applikationsanvändarna. Eftersom den andra gruppen utvecklade parallellt på hemsidan och redan hade byggt upp en databasstruktur så ledde detta till svårigheter med synkronisering mellan applikation och hemsidan. Detta ledde oss till att välja en intern databas som simulering av data.

Resultatet blev således en fungerande användarvänlig prototyp med cirka tvåtusen raders kod som uppfyllde de villkor vi hade tänkt oss från början förutom krav nummer 7 i kapitel 1.1.2.

Vad gäller krav nummer ett (kapitel 1.1.2.) så uppnåddes ett användarvänligt gränssnitt tack vare den iterativa utvecklingsmodellen med prototyper som till en början visades för DISA och som sedan vidareutvecklades.

Vad gäller kravnummer två så stödjer Android språkfiler, vilket gör att språkstöd finns. När sedan kravnummer tre skulle uppfyllas, gjordes det genom en färdig funktion som redan finns inbyggd i Androids klassbibliotek.

Man kan nämligen själv välja i vilken ordning man vill hämta ut respektive information från en viss databastabell. När man därför väljer att hämta ut informationen sorterat efter datum, lyckades man sortera efter den tid man önskade.

Det som också var viktigt i applikationen var att man skulle kunna ta hänsyn till start- och sluttid, dvs. när en viss arbetsuppgift påbörjades och när den avslutades. Här kommer kravnummer fyra in i bilden och det lättaste var att lösa detta genom att spara den nuvarande tiden när man trycker på knappen *In progress* som sedan sparas i databasen vilket ger starttiden. När man är färdig med en arbetsuppgift och klickar på *Done* så sparas den nuvarande tiden på telefonen på samma sätt som

starttiden i databasen. På så sätt uppfylldes kravet.

Resten av kraven var inte så svåra att uppfylla, trots att det var mycket kodning bakom resultatet. Det finns många inbyggda funktioner och i Android som underlättar programmeringen.

Lärdomen av allt detta är att det alltid lönar sig att först bygga upp en enkel prototyp för att verkligen uppfatta användarens behov och önskemål på ett förståeligt sätt. Genom dessa prototyper kan man med hjälp av intressenterna analysera huvudsyftet och behoven som utgör kärnan i utvecklingen av applikationen. Samtidigt besvaras frågan varför applikationen är nödvändig för intressenterna. Testning och verifikation av applikationen är något vi kunde ha gjort mer på för att garantera och säkerhetsställa kvalitén. De tester vi körde på applikationen var mest funktionella tester för att gardera oss mot oväntade och slumpvisa tillstånd så som krascher och märkliga beteenden.

Med det sagt så betyder det inte att vår applikation brister på något sätt gällande kvalitén. Vi har lyckats upprätthålla en någorlunda bra kvalité på applikationen tack vare samarbetet med DISA. När den ena av oss utvecklande så granskades den andra koden samtidigt och på så sätt lyckades vi få till en välfungerande applikation trots de få tester vi körde.

Säkerhet är en annan aspekt av utvecklingen som vi inte prioriterade eftersom applikationen inte körs mot omvärlden och information gällande användarna och innehåll i databasen utgör inte lika stor fara i att exponeras. Applikationen kommer självklart att köras i användarens personliga smartphone vilket betyder att säkerheten är sårbar ifall användaren blir av med sin mobil eller dylikt.

7 Vidareutveckling

Eftersom det vi utvecklat är en prototyp så är det tänkt att applikationen ska vidareutvecklas vilket det finns massor av möjligheter till. Prototypen har fångat ändamålet med projektet och är en utmärkt prototyp att utgå ifrån när det kommer till vidareutveckling.

Målet är att applikationen ska användas av DISAs kunder som befinner sig på olika platser i världen och talar olika språk. Därför är möjligheterna när det kommer till att utöka språkstödet som redan finns inbyggt i Android.

Vad gäller databasen så är det ett projekt i sig att integrera den mot hemsidan och det är bäst att den utvecklas av endast en person eller projektgrupp som känner till uppbyggnaden av både hemsidan och applikationen.

Det finns många möjligheter till vidareutveckling av denna applikation på alla plan, framförallt underhåll och uppgraderingar av applikationen till nästa stora version av Android nämligen Android 4.1 som också kallas för ”Jelly Bean”. I den nya uppdateringen av operativsystemet i Android har förbättringar gjorts vad gäller hastigheten och vissa funktionaliteter i systemet och vissa bibliotek har ändrats om medan andra har tagits bort.[9]

8 Terminologi

API

Application Programming Interface är ett gränssnitt mellan mjukvaror för att de skall kunna kommunicera med varandra.

HTML

Hypertext Markup Language är ett märkspråk och webbstandard för strukturering av text, hypertext, media och inbyggda objekt på exempelvis webbsidor och i e-postmeddelanden.

JSON

JavaScript Object Notation är ett kompakt och textbaserat format för datorer som används för att utbyta data.

XML

eXtensible Markup Language är ett utvidgbart märkspråk som används för att lagra och transportera information.

PHP

HyperText Preprocessor (rekursiv akronym) är ett populärt skriptspråk som främst körs på webbservrar för att driva internetsajter med dynamiskt innehåll.

9 Referenser

[1] Flanagan, Java in a Nutshell,A Desktop Quick Reference, O'Reilly Media, 2005, ISBN13: 9780596007737.

[2] Learning Android, Marko Gargenta, 2011, ISBN: 978-1449390501.

[3]http://mobile.tutsplus.com/tutorials/android/android-sqlite/ http://www.sqlite.org/serverless.html[7 september 2012]

[4]http://www.android.com/about/jelly-bean/ [7 september 2012]

Android for programmers – an app-driven approach, Paul Deitel, 2012, ISBN13: 978-0-13-212136-1.

[5] http://www.makeuseof.com/tag/api-good-technology-explained/ [26 augusti 2012]

[6]http://groups.google.com/group/android-developers/browse_thread/thread/af357db58a506796#databaser[26 augusti 2012]

[7] http://www.sqlite.org/wal.html/[26augusti 2012]

[8] Android Essentials, Chris Haseman, ISBN13: 978-1430210641.

[9]http://www.android.com/about/jelly-bean/ [7 september 2012]

[10] Beginning Android 4 av Grant Allen, 2012, Apress. ISBN13: 9781430239840.

[11]Android for programmersavPaul J. Deitel, Harvey M. Deitel ochAbbey Deitel, 2011, Prentice Hall.ISBN13: 9780132121361.

[12] Databasteknik, Thomas Padron-McCarthy och Tore Risch, ISBN13: 9789144044491.