UPPSALA
UNIVERSITET

TVE 15 057 juni
Examensarbete 15 hp
Juni 2015

Utveckling av applikation till plattformen Android

Arvid Bäärnhielm

UPPSALA
UNIVERSITET

Teknisk- naturvetenskaplig fakultet
UTH-enheten

Besöksadress:
Ångströmlaboratoriet
Lägerhyddsvägen 1
Hus 4, Plan 0

Postadress:
Box 536
751 21 Uppsala

Telefon:
018 – 471 30 03

Telefax:
018 – 471 30 00

Hemsida:
http://www.teknat.uu.se/student

Abstract

Development of application for the Android platform

Arvid Bäärnhielm

Målet med detta projekt var att utveckla en applikation för Android för lagring av tillagade matlådor. Utifrån beräknad hållbarhetstid för matlådan skulle en notifikation ges innan matlådan blev dålig. Då idén byggde på att hållbarhetstider skulle beräknas utifrån ingående ingredienser så kunde projektet inte färdigställas när det visade sig att detta inte var ett genomförbart sätt att beräkna hållbarhetstid. En till stora delar fungerande applikation utifrån grundidén har ändå skapats.

Handledare: Simon Gustafsson
Ämnesgranskare: Martin Sjödin
Examinator: Martin Sjödin
ISSN: 1401-5757, TVE 15 057 juni

Innehåll

1 Introduktion

1.1 Bakgrund

1.1.1 Smarta telefoner

Det finns ingen riktig definition på vad som krävs för att en telefon skall anses vara smart. Därför är det svårt att säga exakt när den första smarta telefonen tillverkades och släpptes till försäljning. Det riktigt avgörande steget i den smarta telefonens utveckling kom dock när Apple släppte sin första version av iPhone 2007 [1]. Sedan dess har väldigt mycket hänt. Fler och fler saker blir smarta, inte bara telefoner, utan även klockor, kylskåp, bilar och mycket annat. Gränssnittet för interaktion med dessa uppkopplade, smarta, saker utgörs av applikationer. Den som äger en smart telefon eller en surfplatta är väl bekant med begreppet och det är även för telefoner och surfplattor som marknaden för applikationer är som störst.

1.1.2 Android

Android är ett operativsystem för smarta telefoner och surfplattor. Det började utvecklas 2003 av Andy Rubin. Google köpte 2005 operativsystemet Android av företaget Android Inc [2] och operativsystemet släpptes på marknaden 2007 [3]. Året efter släppte HTC den första telefonen med operativsystemet Android. Android är numera störst på marknaden, med Apple iOS som största konkurrent [4]. Jämfört med Apple iOS så är Android ett väldigt öppet och obegränsat system, både för användare och för utvecklare [5]. Detta innebär att det är lättare för en ovan utvecklare att utveckla applikationer till Android än till iOS.

1.2 Mål

Målet med projektet är att tillägna sig kunskap om hur en applikation utvecklas till mobilplattformen Android samt omsätta denna kunskap i praktiken genom att utveckla en enkel app. Applikationens funktion är att registrera en tillagad matlåda och dess ingredienser och därefter, via en databas, ange hållbarhetstiden för denna rätt vid förvaring i kylskåp. När hållbarhetstiden är på väg att löpa ut så ska en notifikation skickas via mobilens funktion för push-notiser.

1.3 Avgränsning

Projektet avgränsas genom att applikationen programmering endast kommer att ske för operativsystemet Android. Applikationens layout kommer endast att optimeras för stående skärmläge för normalstora telefoner. Ingen hänsyn kommer att tas till om layouten passar i liggande läge eller på skärmar med större eller mindre storlek.

2 Teori

2.1 Utvecklingsmiljön Android Studio

Utvecklingsmiljön Android Studio är den officiella Integrated Development Environment (IDE) för Android [6] och är fritt tillgänglig att ladda ner. Programmet innehåller alla nödvändiga funktioner för att programmera en applikation till Android, så som Android Software Development Kit (SDK). Det är även möjligt att ladda ner endast Android SDK och använda en annan utvecklingsmiljö, så som Eclipse. Android Studio bygger på IntelliJ IDEA Community Edition, en populär IDE utvecklad av JetBrains, och har ett användarvänligt gränssnitt.

2.2 Språk

2.2.1 Java

All programmering till Android utgår från programmeringsspråket Java [7]. Java är ett objektorienterat programmeringsspråk i samma familj som C, C++ och C# [8]. Varje objekt beskrivs med en klass som innehåller instansvariabler och metoder. Dessa klasser kan ärva egenskaper från andra klasser, så att redan skriven kod kan återanvändas. Instansvariablerna representerar de egenskaper som klassen har och metoderna representerar vad klassen kan göra.

2.2.2 XML

XML (eXtensible Markup Language) är ett märkspråk som i programmering för Android främst används i skapandet av layouter [9]. Språket används för formatering av data [10]. Ett av målen med språket är att det ska vara lätt att förstå, så därför är enkelhet av stor vikt.

2.2.3 SQLite

SQLite är en databashanterare som bygger på Structured Query Language (SQL) [11]. SQLite skiljer sig från flera andra databashanterare genom att den är integrerad i användarens applikation. SQLite är ett av de mest populära valen när det kommer till databashantering för applikationer som utgår från integrerad databashantering för lokal lagring. SQLite är inkluderat som standard i Android.

2.3 Program och tjänster

2.3.1 Git och GitHub

Git är ett versionshanteringsprogram som skapades 2005 av Linus Torvalds, även känd som skapare av operativsystemet Linux [12]. Numera används det flitigt av utvecklare och programmerare över hela världen. Huvudfunktionen med Git är att en kopia av koden skapas, en så kallad gren, som programmeraren arbetar med [13]. På det sättet lämnas originalkoden orörd. När den nya koden fungerar flätas den in i originalgrenen. Det är möjligt för flera

användare att grena av koden samtidigt och arbeta parallellt med olika delar av ett utvecklingsprojekt.

Git är från början utvecklat för att användas i terminalläge, men det finns även program med användargränssnitt, där GitHub tillhandahåller ett som är väl integrerat med deras webtjänster. Android Studio tillhandahåller även en egen implementering av Git och GitHub. Det möjliggör snabb växling mellan olika grenar, överföring av ny kod mellan grenar samt jämförande av grenar direkt i Android Studio.

2.3.2 Valentina Studios

Valentina Studios är ett program för databashantering. Med ett grafiskt användargränssnitt underlättas databashanteringen genom att användaren slipper skriva SQL-kod. Valentina Studios möjliggör skapandet av databaser med förbestämda poster. Att upprätta en databas med förbestämda poster är nödvändigt för att säkerställa att den information som applikationen grundar sig på inte är godtyckligt framtagen av användaren.

2.4 Operativsystemet Android

För att få en uppfattning om hur programmering sker för operativsystemet Android så är det viktigt att ha en insikt i hur applikationer är uppbyggda och hur operativsystemet startar och avslutar applikationer Det är viktigt att förstå vad vilka rättigheter en applikation har och hur rättigheterna deklareras. Det är även viktigt att ha kännedom om hur operativsystemet har förändrats samt hur operativsystemet anpassar sig till olika typer av enheter.

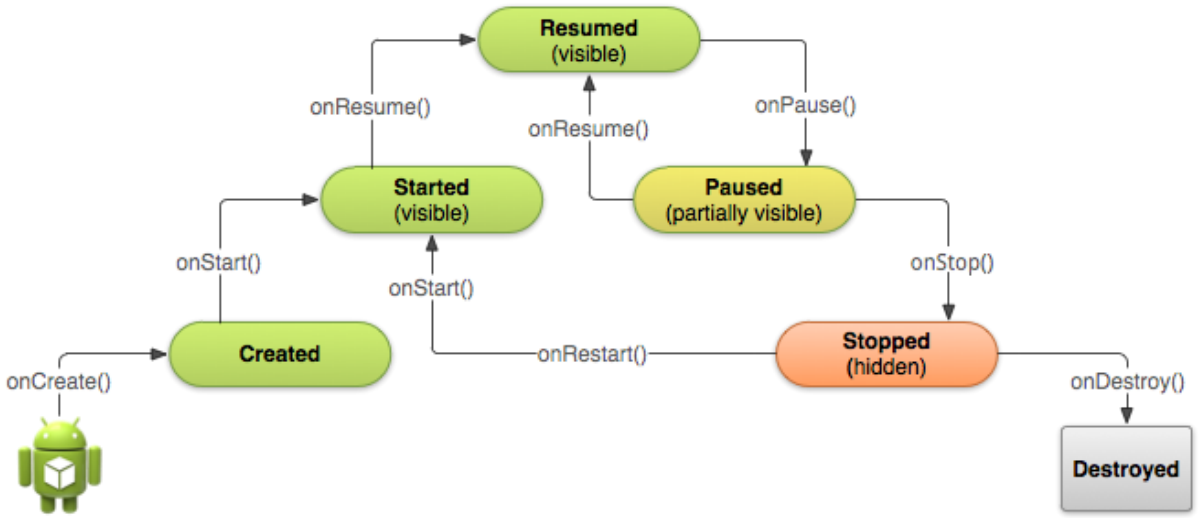
2.4.1 Applikationers uppbyggnad

En applikation för Android är uppbyggd kring aktiviteter [14]. Varje aktivitet kontrollerar vad som visas på skärmen och bygger upp det användargränssnitt som användaren möter för tillfället [15]. En applikation består vanligtvis av flera aktiviteter och växlar mellan dessa beroende på vad användaren gör. Inom varje applikation kapslas vanligtvis olika delar av funktionaliteten in i så kallade fragment [16]. Ett fragment liknar till stor del en aktivitet, men till skillnad från aktiviteter är det möjligt att ha flera fragment aktiva parallellt. En stor fördel med det är att applikationen kan bli väldigt dynamisk och fungera på enheter med olika stor skärm med väldigt lite extra kod. På en surfplatta, som har mycket större skärm än en telefon, kan flera fragment visas parallellt, som på en telefon hade behövt ta upp hela skärmen vardera.

Det är möjligt att programmera en applikation utifrån en enda aktivitet och sedan endast växla mellan fragment. Ju fler funktioner applikationen har desto viktigare blir det att använda flera aktiviteter eftersom funktionaliteten och säkerheten då blir bättre [17]. Information som applikationen hanterar ligger relativt tillgänglig så länge en och samma aktivitet körs. Hanteras känslig information är det viktigt att kontrollera att den informationen inte ligger öppen längre än nödvändigt. Det är då extra nödvändigt att använda sig av flera aktiviteter.

2.4.2 Aktiviteters livscykel

En viktig skillnad mellan aktiviteter och fragment i Android är att aktiviteter har en väl definierad livscykel, se Figur 2-1 [18]. Fragmentens livscykel styrs av den aktivitet de existerar inom [16]. I en traditionell dator har användaren full kontroll över när ett program eller applikation startar och när det ska stängas av. Den kontrollen har försvunnit i Android, åtminstone när det gäller att stänga av en applikation. Istället ligger det på operativsystemet att stänga av applikationer efter behov. En applikation är bara en uppsättning aktiviteter, så en applikation stängs av helt enkelt genom att alla dess aktiviteter stängs av.



Figur 2-1 Flödesdiagram över livscykeln för en aktivitet i en applikation.

Det är viktigt att användaren kan växla mellan applikationer och aktiviteter smidigt, men med den begränsade mängd minne som en telefon har att arbeta med så måste minne ibland frigöras för att den aktiva applikationen eller aktiviteten ska fungera tillfredsställande [18]. Genom att aktiviteter hålls startade så länge som det finns tillgängligt minne och stängs av när minnet inte räcker så kan den aktiva applikationen eller aktiviteten fungera tillfredsställande samtidigt som det går snabbt att växla till andra applikationer eller aktiviteter.

Vilka aktiviteter som har lägst prioritet avgörs av flera faktorer, där den avgörande faktorn är om aktiviteten är synlig och aktiv, delvis synlig men pausad, eller osynlig och stoppad. Alla aktiviteter som är startade lagras i en backstack och ordningen i backstacken avgör vilken prioritet en aktivitet har [19]. Det kan finnas situationer när en aktivitet inte används aktivt, men ändå behöver hållas startad. Genom att använda så kallade tjänster kan aktiviteter flyttas upp i backstacken och därmed hållas levande längre [20].

2.4.3 Manifest

Manifestet är en viktig del av applikationen där olika egenskaper och rättigheter för applikationen definieras [21]. Operativsystemet måste läsa av manifestet innan applikationen kan starta, då koden inte kan exekveras på rätt sätt om inte alla nödvändiga rättigheter har tilldelats applikationen. Dessa rättigheter gäller oftast tillgång till funktioner utanför

applikationen, såsom internet, kamera, GPS eller annat. I manifestet deklareras även alla aktiviteter och tjänster som applikationen består av.

2.4.4 Resurser

Till en applikation finns det även så kallade resurser. Bland resurserna samlas alla filer för de olika layouterna som applikationen använder sig av [22]. Här samlas även alla bildfiler till ikoner och annan design. Flera olika layouter och bildfiler skapas så att applikationen kan välja den som passar bäst utifrån storlek på skärmen.

2.4.5 Application Programming Interface (API)

Operativsystemet Android är under ständig utveckling för att optimera funktionaliteten och utnyttja den högre prestandan som kommer med utvecklingen av nya telefoner och surfplattor. Varje uppdatering av operativsystemet får ett versionsnummer, en så kallad API-nivå. Sedan 2009, då API-nivå 3 släpptes, delas dessa nivåer in i grupper med fantasirika kodnamn i bokstavsordning [23]. API-nivåerna 11, 12 och 13 går till exempel under kodnamnet Honeycomb och API-nivån 19 går under namnet KitKat. Vid programmering för Android är det viktigt att ha dessa versioner i åtanke. De högre API-nivåerna klarar av att hantera funktioner som de lägre nivåerna inte klarar av eftersom de riktar sig till enheter med högre prestanda. För att applikationen skall gå att använda på enheter som inte kan hantera operativsystem med de högre API-nivåerna måste dessa funktioner avstås ifrån. För att kunna utnyttja de funktionerna så begränsas antalet enheter som kan köra applikationen. Det är sällan motiverat att skapa två olika applikationer, en för lägre API-nivåer och en för högre API-nivåer, men det är annars ett sätt att hantera problemet på.

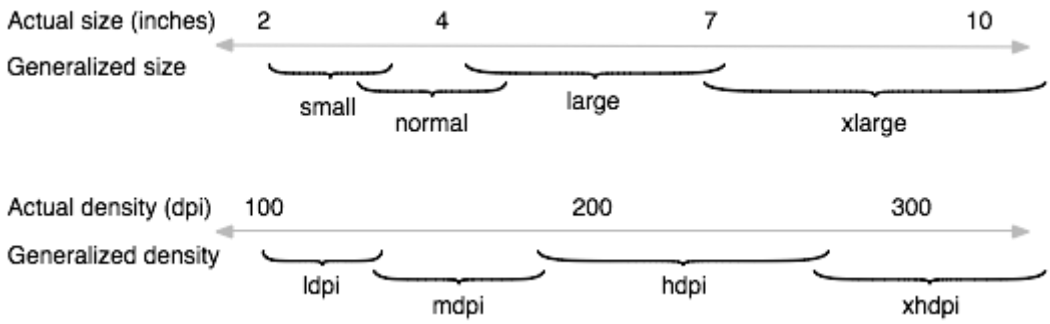
2.4.6 Skärm

Skärmstorleken skiljer sig mellan telefoner och surfplattor det är osannolikt att en och samma layout kommer att se bra ut på både telefon och surfplatta. Även mellan olika telefoner och olika surfplattor kan skärmstorleken skilja, vilket gör valet av layout ännu mer komplicerat. Android delar in skärmstorlekarna i ett antal grupper, se Figur 2-2 [24]. Vissa grupper överlappar något, så en skärmstorlek skulle tekniskt sett kunna ligga i två grupper samtidigt. Denna indelning är därför mest till för att få en känsla av storleken på en skärm.

Utöver skillnaden i storlek så skiljer även upplösningen på skärmen mellan olika enheter. Därför används oftast pixeldensitet, antal pixlar per längdenhet, som ett tydligare mått på upplösning istället för faktiskt antal pixlar [24]. Denna densitet mäts i pixlar per tum, men även måttenheten punkter per tum (”dots per inch”, dpi) används [25]. Android likställer måttenheterna men använder främst dpi för att beteckna pixeldensitet.

Även här delar Android in enheterna i ett antal grupper utifrån hur hög dpi enheten har, se Figur 2-2. Eftersom mycket av grafiken är pixelbaserat, såsom ikoner och andra bilder, så kommer de skalas upp eller ner i upplevd storlek beroende på pixeldensiteten. Därför är det viktigt att programmera så att applikationen anpassar layouten efter pixeldensiteten. Ikoner

behöver till exempel komma i flera storlekar så att den upplevda storleken är ungefär densamma på alla enheter.



Figur 2-2 Schematisk bild över grupperingen av enheter utifrån faktisk storlek (övre linjen) samt pixeldensitet (undre linjen).

För att underlätta programmeringen så används densitetoberoende pixlar, så kallade densitetspixlar (”density pixels”, dp) [26]. En dp har samma verkliga bredd oavsett upplösning på skärmen och blir därför ett tydligt mått på hur stor en bild kommer att bli på skärmen. Övergången från dp till pixlar, px sker med ekvationen $px = dp * (dpi/160)$ [24]. Genom att skapa layouter med storleken definierad i dp istället för ett givet antal pixlar så anpassas storleken efter pixeldensiteten på skärmen. Att använda dp är ett tydligare sätt att kategorisera skärmstorlek än centimeter eller tum, då det har en tydligare koppling till upplösningen, som räknas i pixlar. För att anpassa layouten efter skärmstorlek, för att till exempel visa flera fragment parallellt, så skapas därför olika layouter beroende på antal dp. Applikationen väljer därefter den högsta tillgängliga layouten utifrån skärmens dp.

3 Utförande

3.1 Programidén

Användaren ska kunna registrera att en matlåda har tillagats och markera vilka ingredienser matlådan innehåller. Ingredienserna till matlådan är tidigare lagrade i en annan databas och vid registreringen av en matlåda hämtas alla ingredienser och visas i en lista där användaren markerar vilka ingredienser som ska ingå. Vid registreringen lagras matlådan i en databas och visas på skärmen i en lista tillsammans med övriga registrerade matlådor. Det ska även vara möjligt att lägga till egna ingredienser med hållbarhetstid efter eget val. När en matlåda har registrerats ska en hållbarhetstid beräknas. En kort tid innan hållbarhetstiden löper ut skickas en notifikation så att användaren upplyses om att det är risk att maten blir dålig. Notifikationen skickas via enhetens så kallade push-notiser, ett inbyggt system i Android för notifikationer.

Applikationen ska vara lätt att använda. Det ska gå snabbt att lägga till en matlåda eller ingrediens och det ska vara enkelt att redigera tillagda matlådor och ingredienser i efterhand. En komplicerad och svårförståelig design kommer innebära att användaren inte tar sig tid att registrera matlådor då användaren kan anse att det är för jobbigt eller omständligt.

3.2 Programmering

3.2.1 Användargränssnittet

Användargränssnittet utgörs av sju separata javaklasser; två aktiviteter och fem fragment. Aktiviteten MainActivity sköter de grundläggande funktionerna för applikationen och använder sig av fragmenten BoxFragment, IngredientFragment, AddBoxDialogFragment och AddIngredientDialogFragment. Aktiviteten FoodActivity sköter detaljvyn av en matlåda och använder sig av fragmentet FoodFragment. Nedan följer en mer detaljerad beskrivning av klasserna.

3.2.1.1 MainActivity

För att göra applikationen överskådlig och lättnavigerad så läggs listan för matlådor och listan för ingredienser i varsitt fragment i en så kallad ViewPager som aktiviteten MainActivity håller reda på. En ViewPager består av ett antal sidor och användaren navigerar mellan dessa sidor genom att dra fingret i sidled över skärmen. Varje sida är kopplad till ett unikt fragment som i sin tur hanterar vad som ska visas för användaren. I denna applikation består ViewPagern av två sidor, där den första sidan är kopplad till fragmentet som hanterar listan för matlådor och den andra sidan är kopplad till fragmentet som hanterar listan för ingredienser.

3.2.1.2 BoxFragment och IngredientFragment

Både de registrerade matlådorna och de registrerade ingredienserna ska visas för användaren i form av en lista. BoxFragment är det fragment som hanterar listan för matlådor och IngredientFragment är det fragment som hanterar listan för ingredienser. Det ska också finnas en knapp för att lägga till en matlåda respektive ingrediens. Skillnaden mellan dessa två fragment är alltså väldigt liten. Den största skillnaden ligger i att det ska vara möjligt att klicka på en matlåda och få fram en detaljerad vy över vilka ingredienser som ingår i matlådan och där kunna göra ändringar. I övrigt skiljer sig fragmenten endast åt genom att de är kopplade till varsin databas samt att knappen för att lägga till en ny matlåda eller ingrediens öppnar varsin dialogruta.

3.2.1.3 FoodActivity och FoodFragment

När användaren klickar på en matlåda i listan som BoxFragment genererar så startas en ny aktivitet, FoodActivity, och ett nytt fragment, FoodFragment. Eftersom aktiviteten FoodActivity endast hanterar ett enda fragment så skapas ingen ViewPager. Fragmentet FoodFragment liknar på många sätt BoxFragment och IngredientFragment genom att en lista över ingredienser visas. De ingredienser som visas i listan är dock bara ett urval av alla ingredienser i databasen. Fragmentet skiljer sig dessutom i att det inte finns någon knapp för att lägga till ingredienser.

3.2.1.4 AddBoxDialogFragment och AddIngredientDialogFragment

BoxFragment och IngredientFragment innehåller varsin knapp för att lägga till en matlåda respektive ingrediens. Knapparna startar en typ av fragment som kallas dialog, ett fönster, som uppmanar användaren att ta ett beslut, ange ytterligare information eller att vidta någon annan åtgärd innan användaren kan gå vidare. Fönstret skymmer inte hela skärmen och den

tidigare aktiviteten är fortfarande synlig i bakgrunden [27]. AddBoxDialogFragment skapar ett fönster där användaren kan fylla i namn, lagringsplats och ingredienser till matlådan. AddIngredientDialogFragment skapar ett fönster där användaren kan fylla i namn samt hållbarhetstid i kylskåp respektive frys. När användaren har fyllt i all information och bekräftar så skickas ett anrop tillbaka till MainActivity via ett så kallat interface. MainActivity anropar därefter korrekt databas för lagring av den information som användaren har angett.

3.2.2 Databashantering

Databashanteringen delas upp i tre olika klasser som vardera sköter en del. FoodDBhelper (DB står för DataBase) sköter upprättandet av databaserna, FoodContract sköter adressering och FoodProvider sköter redigering och läsning av databaserna. Nedan följer en mer detaljerad beskrivning av de tre klasserna.

3.2.2.1 FoodDBHelper

Språket SQL har en annan syntax i sin programmering än java, så javas syntax behöver översättas till SQL för att det ska vara möjligt att skapa en databas i SQLite. Android Studio har skapat egna klasser för att hantera detta. SQLiteOpenHelper och SQLiteDatabase är sådana klasser. SQLiteOpenHelper är en klass som är skapad med syfte att underlätta skapandet av databaser. Klassen är dock inte anpassad för att kunna använda i förväg konstruerade databaser. Därför har andra programmerare skapat en klass som heter SQLiteAssetHelper [28], som ärver SQLiteOpenHelper och som underlättar vid användandet av i förväg konstruerade databaser. FoodDBHelper ärver klassen SQLiteAssetHelper och implementerar dess funktioner.

3.2.2.2 FoodContract

En viktig del vid hantering av databaser är att kunna hämta information på ett korrekt sätt. För att göra det så krävs det att applikationen vet var rätt information finns. Applikationen behöver veta namnet på databasen samt namnet på alla kolumner som ska hämtas. Applikationen behöver också veta vilka anrop som ska göras beroende på vilket urval av data som ska hämtas ifrån databasen. Denna information lagras med fördel i en separat klass, en form av kontrakt [29]. Klassen FoodContract innehåller i sin tur inre klasser för varje databas med information om namn på databasen och kolumnerna. I klassen finns även metoder för att skapa olika URI, adresser för att avgöra vilket urval av information som ska hämtas ur databasen. Genom att hålla sådan information samt genom att skapa alla sådana adresser i klassen FoodContract så elimineras risken att någon försöker hämta information ur en databas med hjälp av en felaktig adress.

3.2.2.3 FoodProvider

Centralt i hanteringen av information, vare sig informationen är lagrad i databaser i någon form eller på något helt annat sätt, är användandet av klasserna ContentProvider och ContentResolver [30]. Dessa klasser gör det möjligt att ta del av information mellan olika applikationer på ett smidigt sätt. Genom att översätta informationen så att den kan lagras i så kallade cursors skapas ett standardiserat system för överföring av information. På det sättet

kan informationen delas med andra applikationer eller inom den egna applikationen på ett smidigt sätt. FoodProvider ärver klassen ContentProvider och implementerar därmed metoder för att hämta, lagra ny, redigera samt radera information i de tre databaser som används i denna applikation.

3.2.3 Push-notiser

När en matlåda närmar sig sitt utgångsdatum, beräknat utifrån de ingående ingredienserna, så uppmärksammas användaren om det via så kallade push-notiser. Detta sker i bakgrunden utan att applikationen behöver vara aktiv och behöver därför ligga inom en så kallad Sync Adapter. En Sync Adapter kan ställas in på att köras i bakgrunden utifrån flera olika villkor, till exempel regelbundna tidsintervall eller när enheten kopplar upp sig mot ett trådlöst nätverk.

3.3 Arbetsmetodik

I första fasen av projektet ligger fokus helt och hållet på att samla kunskap om hur programmering för Android sker. Detta görs genom att följa en videokurs via Udacity som lär ut programmering för Android genom att steg för steg programmera en applikation tillsammans med den som tar del av kursen [31]. Deltagaren får löpande uppgifter för att implementera nya funktioner i applikationen och får ofta svara på kortare frågor angående programmeringsteorin. Kursen använder sig av GitHub för att göra koden tillgänglig stegvis genom kursen och deltagaren förväntas därför även lära sig hur GitHub fungerar. Kursen går igenom de flesta grundläggande teknikerna för programmering för Android: layout, databashantering, aktiviteter och fragment, laddare samt notifikationer för att nämna några.

I andra fasen av projektet ligger fokus på att programmera applikationen. Under den sker en ständig återkoppling till videokursen på Udacity samt Androids textguider [32] och träningslektioner [33] för programmering för att lösa problem som uppkommer. Även Stack Overflow, ett webforum för programmering [34], är en källa för problemlösning liksom en hel del andra websidor. Själva programmeringen sker i Android Studio, förutom upprättande av den i förväg konstruerade databasen, där Valentina Studios används. Med hjälp av GitHub är det möjligt att arbeta parallellt med flera delar av projektet samtidigt, men till största delen sker programmeringen linjärt, där varje del avslutats innan nästa påbörjats. Arbetet är inte planerat i detalj, mycket beroende på att detaljkunskapen om programmeringen inte var så hög vid starten av projektet. Det innebär att många beslut tas löpande under projektet och planeringen uppdateras efter hand som applikationen växer fram i programmeringen.

Löpande under projektets gång sker arbetet med att ta fram underlag för hållbarhetstider för ingredienser. Detta sker främst genom mailkontakt med Livsmedelsverket och Föreningen Fryst och Kyld Mat [35] samt genom att läsa de dokument som Livsmedelsverket har lagt upp på sin hemsida [36].

4 Resultat

4.1 Programidén

Ett par veckor efter att projektet påbörjats stod det klart att någon information om hållbarhetstid för råvaror och ingredienser inte var möjlig att få fram. Det finns inga riktlinjer för enskilda råvarors hållbarhetstid och i synnerhet inte för tillagade råvaror. För livsmedelsproducenter gäller att de själva har ansvar att beräkna och fastställa hållbarhetstider för sina produkter [37]. Inom restaurangbranschen finns generella riktlinjer för hur länge mat får förvaras och på vilket sätt [38]. De bygger dock inte på de specifika ingredienserna i en maträtt på det sätt som denna applikation är tänkt att göra och är därför inte direkt applicerbart på detta projekt.

När detta stod klart togs dock ändå beslutet att slutföra programmeringen utifrån originalidén då tiden inte skulle räcka till för att arbeta fram en ny idé. Detta innebär att applikationen visserligen kan fungera som idén var tänkt, men funktionaliteten har inget värde. Syftet med projektet blir därför inte längre att skapa en applikation som fungerar utan syftet blir istället att avsluta projektet och lämna en applikation som skulle kunna fortsätta att utvecklas efter projektets slut.

Efter ungefär två tredjedelar av projekttiden stod det dessutom klart att projektet hade vuxit för stort för att hinna avslutas inom tidsramen för projektet. Nya detaljer och funktioner hade lagts till under projektets gång efter hand det klarnade hur applikationen skulle behöva vara uppbyggd och till slut hade projektet vuxit sig för stort för tidsramen.

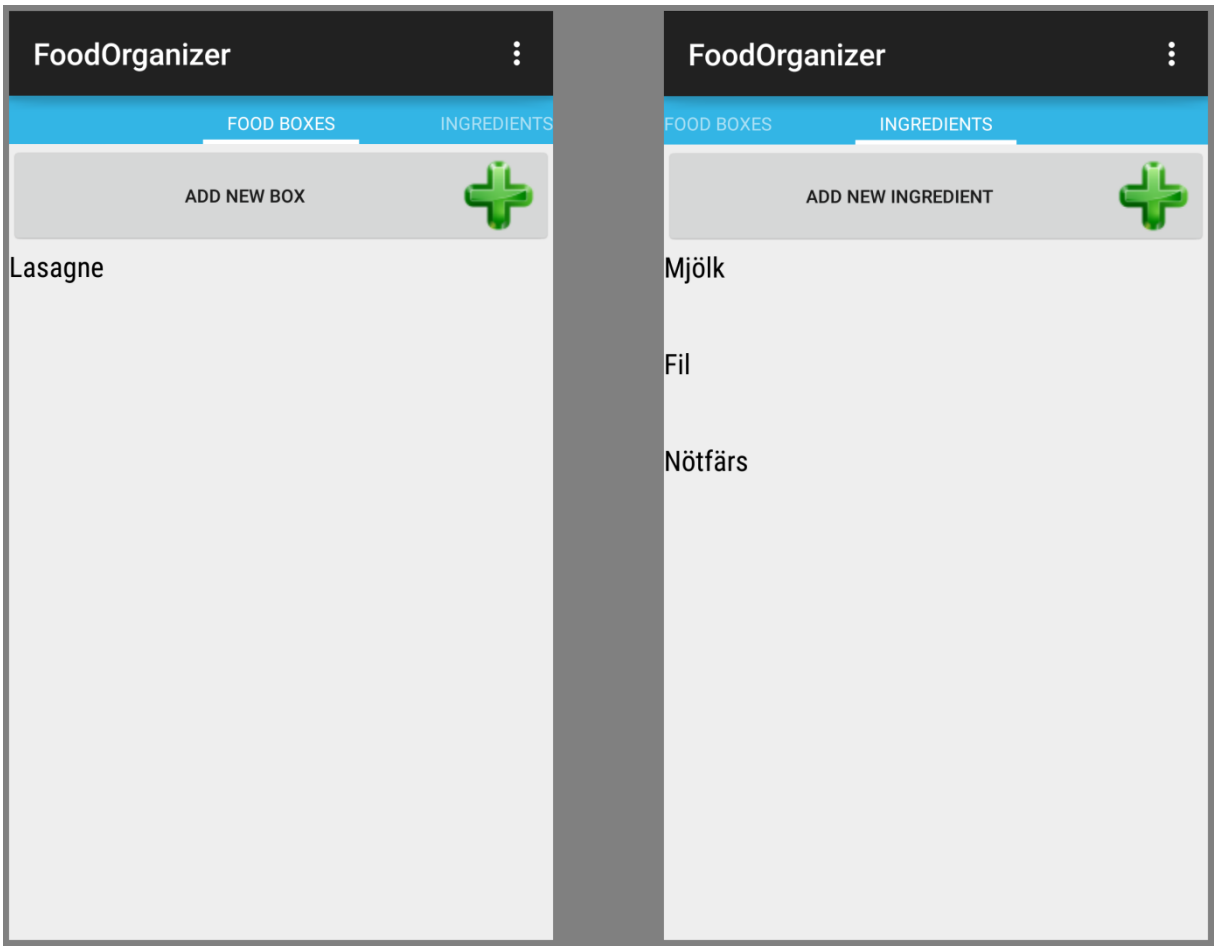
4.2 Programmering

Projektet visade sig bli för tidskrävande och därför har inte alla delar hunnit slutföras inom tidsramen för projektet. Detaljvyn är endast halvfärdig, där kopplingen till databasen ej färdigställts. Övriga delar är färdigställda till största del, men saknar de sista finjusteringarna för att upplevelsen av applikationen ska vara tillfredsställande.

4.2.1 Användargränssnittet

Vid start av applikationen möts användaren av listan över de matlådor som hittills har blivit registrerade, se Figur 4-1. Det är endast namnet på matlådan som presenteras och inte någon information om hållbarhetstid. Samma sak gäller om användaren växlar till listan över ingredienser. Även där presenteras endast namnet på ingredienserna och inte någon information om hållbarhetstid. Information om hållbarhetstid för ingredienser finns lagrade i databasen över ingredienser, men ingen koppling har skapats till matlådan.

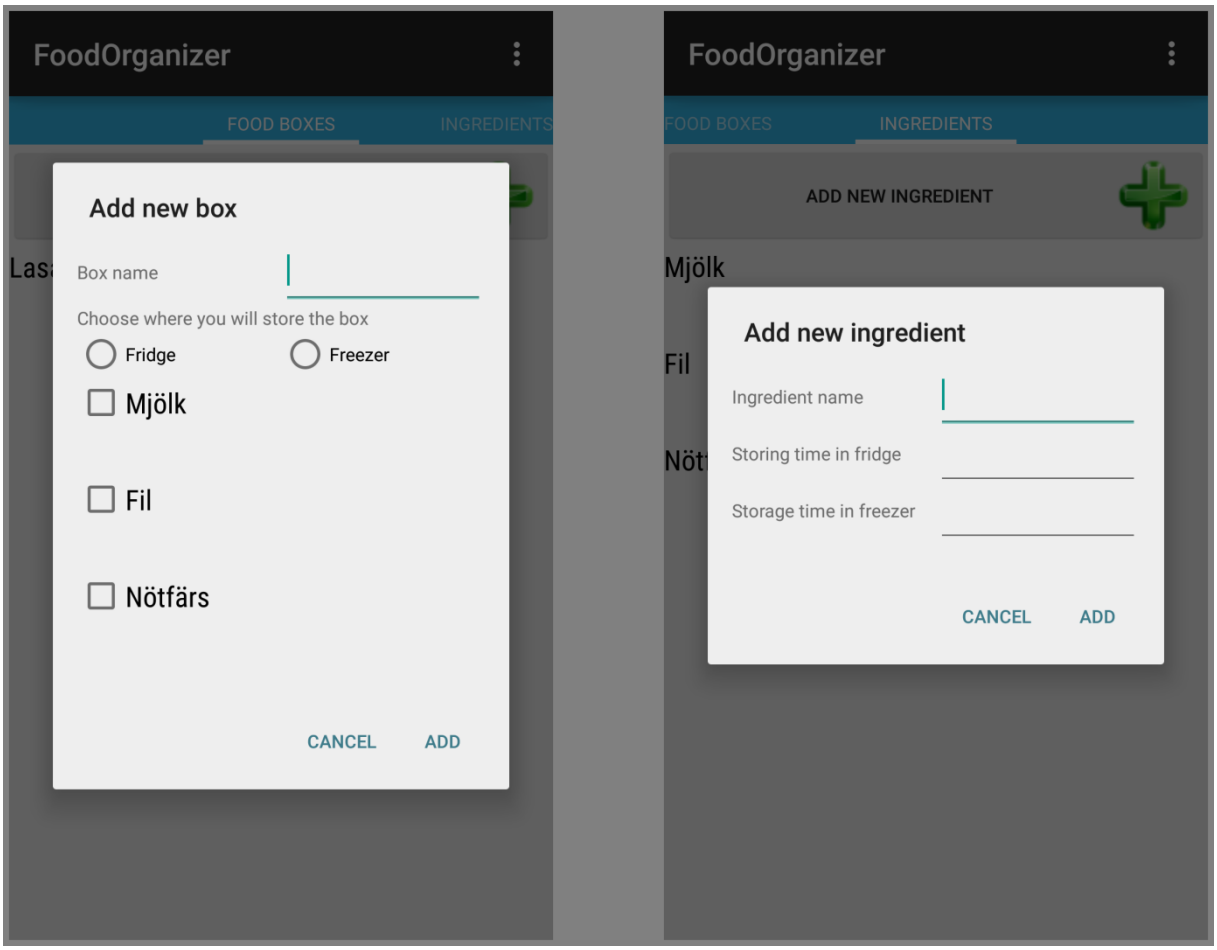
I överkant av skärmen syns namnet på applikationen samt en menyknapp, se Figur 4-1. Menyknappen har än så länge ingen funktion. Det visas en knapp för inställningar, men vid klick på knappen så händer ingenting. Strax under namnet och menyknappen syns rubrikerna på de två listorna, där rubriken för den aktiva listan är i centrum och den inaktiva är strax vid sidan av.



Figur 4-1 Översikt över de två fragmenten. Vid start visas det vänstra fragmentet och genom att svepa handen till vänster så visas det högra fragmentet. Den ljusblåa listen i övre delen av fragmenten visar de två rubrikerna. De två knapparna för att lägga till en matlåda respektive ingrediens kan också ses i överkant av båda fragmenten.

När användaren klickar på en registrerad matlåda i listan så ska en detaljvy visas över de ingående ingredienserna. Användaren får endast respons på att klicket har registrerats, men detaljvyn öppnas inte. Vid klick i listan över ingredienser ges också en respons på att klicket har registrerats. Den listan ska dock inte vara klickbar.

Ovanför listorna möts användaren av en knapp med ett grönt plustecken och texterna ”ADD NEW BOX” respektive ”ADD NEW INGREDIENT”, se Figur 4-1. Vid klick på dessa knappar öppnas respektive dialogruta, se Figur 4-2. Användaren kan sedan välja mellan att fylla i informationen för en ny matlåda eller ingrediens eller att ångra och gå tillbaka till respektive lista. Om användaren har lagt till en ny matlåda eller ingrediens så dyker den direkt upp i listan. Det finns ingen möjlighet att ta bort eller redigera en matlåda eller ingrediens. Ingen implementering av push-notiser har gjorts. En del kända buggar kvarstår dessutom, till exempel att applikationen kraschar om användaren inte fyller i hållbarhetstider för ingredienser.



Figur 4-2 De två dialogrutorna för att lägga till en ny matlåda respektive ingrediens. Dialogrutorna täcker inte hela skärmen och det går att se det tidigare fragmentet i bakgrunden. Förutom de olika fälten för redigering så finns två knappar, en för att avbryta och en för att slutföra.

4.2.2 Databashantering

SQLite kan koppla samman olika databaser genom att använda sig av en så kallad sammanbindning. En sammanbindning mellan två databaser fungerar bara om det är ett ensamt element i den ena databasen som skall kopplas samman med flera i den andra. Det sker genom att en kolumn läggs till i den andra databasen som innehåller information om vilket index i den första databasen som elementen skall sammanbindas med. Ett exempel utanför detta projekt kan vara ett företag som lagrar sina anställda i en databas och alla olika typer av befattning i en annan. För varje anställd anges sedan information om vilket index i databasen över befattningar som den anställde skall sammanbindas med. Varje anställd är bara sammanbunden med en enda befattning. Hade en anställd kunnat ha flera befattningar så hade en annan typ av sammankoppling behövt ske.

I denna applikation kan inte de två databaserna sammankopplas med en join, då varje matlåda kan innehålla flera ingredienser samtidigt som en ingrediens kan finnas i flera matlådor. Detta löses genom att en tredje databas upprättas, som använder sig av en join till vardera av de två existerande databaserna. Varje post är kopplad till en enskild matlåda och en enskild ingrediens. När en ingrediens läggs till i en matlåda så skapas en ny post i denna tredje

databas som innehåller information om både vilken matlåda och vilken ingrediens som ska höra ihop. När användaren klickar in på detaljvyn för en matlåda så söks denna databas igenom efter alla poster som är kopplade till den matlådan, det vill säga de poster som innehåller värdet för det index som matlådan är lagrad på. Sedan avläses posterna för att ta reda på de index som skall avläsas ur databasen för ingredienser. Slutligen hämtas de ingredienser som ligger lagrade på dessa index i databasen för ingredienser och presenteras i listan i detaljvyn.

Tre databaser är skapade och avläsning, skapande, redigering och radering av poster kan ske med hjälp av metoder i klassen FoodProvider. Det går ännu inte att hämta de specifika ingredienserna till en enskild matlåda. Informationen ifrån databaserna hämtas med hjälp av loaders. Så fort en ändring har skett i någon av databaserna så skickar systemet en intern notifikation som fångas upp av berörd loader och triggar igång en omstart, så att användaren ständigt möts av aktuell och uppdaterad information.

5 Diskussion

5.1 Projektplan

Projektplanen var väldigt vag vid projektets start, men utvecklades något under tiden projektet genomfördes. I praktiken skedde planeringen snarare efter behov och endast de närmaste dagarna framåt kan anses ha varit planerade vid varje enskild tidpunkt. Detta innebar såklart ett ökande stressmoment under hela projektets gång, då det var svårt att stämma av hur projektet fortlöpte i förhållande till planeringen.

Tre avstämningar hittas i planeringen. Den första var när videokursen skulle vara färdig, den andra när programmeringen skulle vara (i stor sett) färdig och den sista avstämningen var när den preliminära rapporten skulle vara färdig. Detta är alldeles för få avstämningar för att kunna bilda sig en uppfattning om hur projektet fortlöper i förhållande till planeringen.

Att planeringen blev så vag berodde till största delen på att kunskapen och insikten om vad projektet skulle innebära var väldigt liten vid projektets start. Detta borde såklart ha inneburit att den löpande uppdateringen av projektplanen skulle ha varit betydligt mer ingående. I takt med att applikationens upplägg formades så borde nya delmoment ha planerats in. En grov uppskattning om hur lång tid ett enskilt delmoment skulle kunna ta hade kunnat göras under planeringen för att på så sätt få en uppfattning om hur många delmoment som skulle vara möjligt att hinna med. Efter hand skulle denna uppskattning kunna uppdateras och bli mer detaljerad.

Då ingen uppskattning gjordes över hur lång tid delmoment skulle ta och heller ingen återkoppling gjordes under projekttiden till avslutade delmoment så utvecklades applikationen till att bli för avancerad för att rymmas inom projekttiden. Med en bättre löpande återkoppling så hade vissa av applikationens delar kunnat uteslutas och en mer fullständig applikation hade kunnat presenteras.

5.2 Hållbarhetstider

Ett par veckor in i projektets gång visade det sig att en av de grundläggande funktionaliteterna med applikationen inte skulle vara möjlig att implementera. Hållbarhetstider för enskilda råvaror går nämligen inte att beräkna och dessutom är det inte troligt att det skulle vara meningsfullt att utgå ifrån dessa hållbarhetstider när råvarorna är tillagade tillsammans med andra råvaror. Som en följd av detta fick projektets mål omarbetas. Optimalt hade informationen funnits redan vid projektets start så att ett verkligt genomförbart mål hade kunnat upprättas. Att skjuta upp starten av projektet tills informationen hade tillskansats var såklart inget alternativ. Mer tid hade dock kunnat läggas på att få informationen så tidigt som möjligt under projektet.

5.3 Programmering

5.3.1 Beslutsgång

Eftersom applikationens upplägg växte fram under projektets gång och endast en grov tanke om funktionalitet fanns vid projektets start så har de flesta besluten tagits under projektets gång. Det har till viss del inneburit att tidigare beslut har blivit tvungna att omvärderas och en del programmering har fått skrivas om. Med en bättre planering hade sådana situationer kunnat undvikas i större utsträckning.

5.3.2 Svårighetsgrad

På grund av den dåliga planeringen togs en del beslut som innebar att vissa delar av applikationen blev mer avancerade än vad tidsramen tillät. Ett exempel är beslutet att användaren skulle kunna växla mellan de två listorna genom att svepa i sidled med fingret. Det innebar att en hel del tid fick läggas på att ta reda på hur en sådan funktionalitet skulle kunna implementeras. En betydligt enklare väg, men layoutmässigt mycket sämre, hade varit att bara skapa två knappar för att öppna respektive lista. På så sätt hade mer tid kunnat läggas på att avsluta andra delmoment som nu istället inte har hunnits med.

5.3.3 Layout

Ett beslut som togs för att spara tid var att applikationen endast skulle programmeras för en layout. Ingen hänsyn skulle alltså tas till att anpassa layouten för olika skärmstorlekar eller för det fall att användaren roterar enheten så att skärmen blir liggandes istället för stående. Det var ett beslut som sparade tid för projektet, men är en dålig väg att gå om applikationen ska bli så bra som möjligt. I många fall påverkar utseendet i en viss layout hur utseendet bör vara i en annan layout.

5.3.4 Databas

Eftersom den ursprungliga tanken var att ta fram en databas med hållbarhetstider för ett stort antal råvaror och ingredienser så var det viktigt att informationen fanns med vid installation av applikationen. Det skulle heller inte vara möjligt att ändra på informationen i efterhand.

Det bästa sättet att tillhandahålla informationen var att skapa en databas i förväg som vid installation importeras till applikationen. Så som projektet utvecklade sig försvann dock behovet av att ha informationen med från början. En avvägning gjordes mellan att antingen fortsätta att skapa databasen och importera vid installation, eller att spara tid och inte inkludera funktionen. Det är inte osannolikt att funktionen kan komma att visa sig nödvändig i ett senare skede och i så fall skulle det innebära att stora delar av databashanteringen skulle behöva skrivas om ifall det inte implementeras ifrån början. Valet föll på att ändå skapa databasen i förväg och importera den vid installation. Med tanke på den tidsbrist som senare uppkom samt det faktum att det är osannolikt att applikationen lever vidare utanför detta projekt så borde valet möjligen ha blivit det omvända.

5.4 Arbetsplats

I stort sett allt arbete med projektet har skett ifrån hemmet, vilket har haft både sina för- och nackdelar. En avgörande faktor för valet av arbetsplats har varit behovet av en arbetsdator med utvecklingsmiljön Android Studio installerad. Detta var absolut enklast att lösa genom att använda en privat dator. Andra fördelar har varit att ingen tid har behövt läggas på att förflytta sig ifrån hemmet till en arbetsplats. Att arbeta i hemmet medför dock en hel del distraktion. På det hela taget så har arbetet hemifrån dock fungerat väldigt bra.

5.5 Projektgrupp

Inför projektstarten var det länge osäkert om projektgruppen skulle bestå av en eller två personer. Till slut visade det sig att den andre parten inte skulle delta i projektet. Storleken på projektet var dock sådan att detta inte behövde medföra några större förändringar i planeringen. En sak som eventuellt hade kunnat vara bättre ifall gruppen bestått av två personer är den draghjälp som den andre parten hade gett. Att hålla uppsatta tidsramar blir lättare om det inte bara är sig själv det drabbar att inte göra det. Det hade även eventuellt varit lättare att prioritera bättre om besluten inte enbart hade påverkat en persons arbete.

5.6 Handledare

Då projektet helt och hållet är en egen idé uppstod behovet av att hitta en handledare som kunde handleda projektet. När projektstarten närmade sig så hade någon handledare ännu inte hittats och projektet riskerade därför att inte kunna genomföras. I samråd med kursansvarig på universitetet beslutades dock att projektet kunde genomföras utan handledare och endast med hjälp av en mentor. Det har inneburit en stor utmaning för projektet och i vissa lägen har det varit nära inpå avgörande för projektets genomförande att ingen handledare har funnits. Projektmentorn har dock funnits till hands och fyllt många av de funktioner som en handledare annars skulle ha fyllt. Utan den ämneskunskap som en handledare hade besuttit så har vissa frågor inte kunnat besvaras av mentorn. En del av de brister som projektet har haft skulle möjligen ha kunnat undvikas om en handledare med ämneskunskap hade funnits till hands.

Källor

[1] http://thenextweb.com/mobile/2011/12/06/the-history-of-the-smartphone [läst 2015-06-06]
[2] http://www.webcitation.org/5wk7sIvVb [läst 2015-06-06]
[3] http://www.openhandsetalliance.com/press_110507.html [läst 2015-06-06]
[4] http://www.forbes.com/sites/dougolenick/2015/05/27/apple-ios-and-google-android-smartphone-market-share-flattening-idc/2 [läst 2015-06-06]
[5] http://www.howtogeek.com/217593/android-is-open-and-ios-is-closed-but-what-does-that-mean-to-you [läst 2015-06-06]
[6] https://developer.android.com/sdk/index.html [läst 2015-06-06]
[7] http://www.androidauthority.com/want-develop-android-apps-languages-learn-391008 [läst 2015-06-06]
[8] http://www.webopedia.com/TERM/J/Java.html [läst 2015-06-06]
[9] http://developer.android.com/guide/topics/ui/declaring-layout.html [läst 2015-06-06]
[10] http://www.w3.org/XML [läst 2015-06-06]
[11] https://www.sqlite.org/about.html [läst 2015-06-06]
[12] https://git-scm.com/book/en/v2/Getting-Started-A-Short-History-of-Git [läst 2015-06-06]
[13] https://git-scm.com/about [läst 2015-06-06]
[14] http://developer.android.com/reference/android/app/Activity.html [läst 2015-06-06]
[15] http://developer.android.com/guide/components/activities.html [läst 2015-06-06]
[16] http://developer.android.com/guide/components/fragments.html [läst 2015-06-06]
[17] http://developer.android.com/guide/practices/seamlessness.html [läst 2015-06-06]
[18] http://developer.android.com/training/basics/activity-lifecycle/starting.html [läst 2015-06-06]
[19] http://developer.android.com/guide/components/tasks-and-back-stack.html [läst 2015-06-06]
[20] http://developer.android.com/guide/components/services.html [läst 2015-06-06]
[21] http://developer.android.com/guide/topics/manifest/manifest-intro.html [läst 2015-06-06]
[22] http://developer.android.com/guide/topics/resources/overview.html [läst 2015-06-06]
[23] https://source.android.com/source/build-numbers.html [läst 2015-06-06]
[24] http://developer.android.com/guide/practices/screens_support.html [läst 2015-06-06]
[25] http://whatis.techtarget.com/definition/pixels-per-inch-ppi [läst 2015-06-06]
[26] http://developer.android.com/training/multiscreen/screendensities.html [läst 2015-06-06]
[27] http://developer.android.com/guide/topics/ui/dialogs.html [läst 2015-06-06]
[28] https://github.com/jgilfelt/android-sqlite-asset-helper [läst 2015-06-06]
[29] http://developer.android.com/training/basics/data-storage/databases.html [läst 2015-06-06]

[30] http://developer.android.com/guide/topics/providers/content-provider-basics.html [läst 2015-06-06]
[31] https://www.udacity.com/course/developing-android-apps--ud853 [läst 2015-06-06]
[32] https://developer.android.com/guide/index.html [läst 2015-06-06]
[33] https://developer.android.com/training/index.html [läst 2015-06-06]
[34] http://stackoverflow.com [läst 2015-06-06]
[35] http://www.frystochkyldmat.se [läst 2015-06-06]
[36] http://www.livsmedelsverket.se [läst 2015-06-06]
[37] http://www.livsmedelsverket.se/globalassets/rapporter/2011/2011_livsmedelsverket_20_forvaring_och_hallbarhet.pdf - finns även att ladda ner på http://www.livsmedelsverket.se/matvanor-halsa--miljo/miljo/ta-hand-om-maten-minska-svinnet/forvara-maten-ratt
[38] http://www.livsmedelsverket.se/globalassets/produktion-handel-kontroll/branschriktlinjer/visitas-branschriktlinjer-for-restauranger.pdf (se sida 33) - finns även att ladda ner på http://www.visita.se/branschfragor1/Livsmedel--hygien/Riktlinjer