CHALMERS

En jämförelse mellan mobila plattformar

NICLAS LUNDGREN

MARTIN NUMÉ

Examensarbete

Högskoleingenjörsprogrammet för datateknik

CHALMERS TEKNISKA HÖGSKOLA
Institutionen för data- och informationsteknik
Göteborg 2012

Summering

Utmaningen att strukturera en mobilapplikation så att den effektivt kan utvecklas till multipla plattformar är det drivande problemet. Vi utvecklade en applikation till Windows Phone för att få en djupare insikt och kunskap om plattformen och tillsammans med tidigare kunskap om Android låg de till grund för en jämförelse mellan plattformarna. Områdena som jämfördes valdes specifikt för att inom dessa existerar skillnader mellan plattformarna som kan skapa problem vid utveckling om de inte tas i beaktning. Utifrån jämförelserna arbetade vi fram ett dokument som visar på skillnader mellan plattformarna som är viktiga att uppmärksamma och ta hänsyn till vid utveckling av en applikation till både Android och Windows Phone. Dokumentet är tänkt att användas av utvecklare som ett hjälpmedel för att underlätta och effektivisera struktureringen av en applikation som ska utvecklas till både Windows Phone och Android.

Abstract

The challenge of structuring a mobile application so that it can efficiently be developed into multiple platforms is the driving problem. We developed an application for Windows Phone to get a deeper insight and knowledge of the platform and together with previous knowledge of Android formed the basis of a comparison between the platforms. The areas that were compared were selected specifically for the existence of differences between the platforms that can create problems during the development if not taken into consideration. Based on the comparisons we produced a document that shows the differences between the platforms that are important to acknowledge and take into account when developing an application for both Android and Windows Phone. The document is intended for developers as a mean to facilitate and optimize the structuring of an application that is developed for both Windows Phone and Android.

Innehållsförteckning

Förkortningar och begrepp

Activity	Utgör grunden för en applikation till Android
Android	Googles operativsystem för mobiltelefoner
App	Förkortning av applikation, ett program som körs på en mobil plattform
Data-bindning	Används i WP för att koppla samman kod från gränssnitt till övrig applikationskod
Gränssnitt	Applikationens grafiska gränssnitt som visas på skärmen
IDE	Integrated Development Environment, mjukvara som används för programutveckling
iOS	Apples operativsystem för mobiltelefoner
Marketplace	Service som tillhandahålls av Microsoft där användare kan ladda ned applikationer
Markup extension	Utbyggnad av attributer i XAML
MVC	Förkortning för Model View Controller. Ett designmönster som används vid mjukvaruutveckling
MVVM	Förkortning för Model-View- ViewModel. Ett designmönster som används vid mjukvaruutveckling
Page	Utgör grunden för en applikation till WP
UI	User Interface, det grafiska gränssnittet som användare ser.
Wireframes	Visuell presentation av en applikations utseende och funktioner
WP	Förkortning av Windows Phone. Microsofts operativsystem för Mobiltelefoner
XAML	eXtension Application Markup Language
XML	eXtensible Markup Language

1. Inledning

1.1 Bakgrund

Under de senaste åren har det blivit allt mer populärt med så kallade smartphones och idag äger en stor del av befolkningen en telefon baserad på Android eller iOS. Även utvecklingen till dessa plattformar har exploderat. Företagen har hoppat på den så kallade "apptrenden" och många nya företag har startats medan andra har bytt inriktning mot mobila plattformar.

Ett problem för utvecklare idag är att det inte räcker med att släppa sina applikationer för bara en plattform. Båda plattformarna är stora och genom att bara släppa applikationen till en plattform går man miste om ett stort antal användare. Eftersom de två plattformarna skiljer sig så pass mycket är det inte helt trivialt att utveckla samma applikation till Android och iOS. Är den inte väl planerad och strukturerad när man skapar den från början kan det i princip innebära att det blir en helt ny applikation när man ska utveckla den till en annan plattform.

I slutet av 2011 gjorde Microsoft intåg på den mobila marknaden med Windows Phone 7 vilket komplicerar situationen ytterligare för utvecklare när applikationer ska skapas till flera plattformar. Med tre olika plattformar som en applikation potentiellt ska kunna publiceras till krävs väl strukturerad och avskild kod.

1.2 Problem

För många utvecklare är det ett problem att skapa en applikation till flera plattformar. Vilka råd och riktlinjer kan ges till utvecklare för att underlätta detta?

1.3 Syfte

Syftet med denna rapport är att jämföra utvecklingen av applikationer mellan plattformarna Android och Windows Phone. Detta gör vi för att få en djupare kunskap och förståelse av utvecklingsprocessen och plattformarnas grundläggande struktur.

1.4 Avgränsningar

Av hänsyn till den tidsbegränsning vi har kommer bara en jämförelse mellan Android och Windows phone att göras. Vi kommer inte beröra andra plattformar, så som iOS. Jämförelsen kommer endast göras utifrån en utvecklares perspektiv och aspekter som till exempel användarvänlighet kommer inte inkluderas i denna rapport. Rapporten kommer inte behandla området spelutvecking då programflöde och struktur kan skilja sig markant.

1.5 Rapportöversikt

I avsnitt 3 finnes den tekniska bakgrunden som är en summering av den efterforskning vi gjort samt det som ligger till grund för vårt resultat. Utan tidigare kunskaper om mobila plattformar och/eller programmering kan detta avsnitt upplevas relativt tungläst. I avsnitt 4 presenteras företaget vi samarbetade med under detta examensarbete samt produkten vi skapade. Vårt resultat som innehåller jämförelser mellan Android och Windows Phone samt riktlinjer och tips vid utveckling till dessa plattformar hittas under avsnitt 5. Avsnitt 6 innehåller slutsatsen, metodkritik samt våra tankar inför framtiden.

2. Metod

För att få mer kunskap om utveckling av applikationer till mobila plattformar kontaktade vi ett företag som inriktar sig på detta vid namn MobisleApps. Under ett möte kom vi överens om att utveckla deras antecknings-applikation till Windows Phone. Utvecklingen av denna applikationen tillsammans med våra tidigare kunskaper inom Androidutveckling ska ligga till grund för jämförelsen av de två plattformarna. Under utvecklingen av applikationen kommer skillnader och likheter noteras. Dessa punkter kommer sedan leda till ytterligare efterforskningar och ett försök att finna mer information om varför de existerar samt hur de påverkar utvecklingen av en applikation. Vi kommer under detta sökande av information bygga upp vår förståelse och insikt i hur Windows Phone fungerar samt rent generellt hur en mobil plattform är uppbyggd. Jämförelsen kommer baseras på allt ifrån grafisk design, programstruktur till underliggande funktioner i operativsystemen. Utifrån denna information och vad vi lärt oss ska vi sedan dra slutsatser och skapa riktlinjer för utveckling av applikationer till de två plattformarna. Riktlinjerna ska sammanställa punkter som är värda att tänka på för att underlätta utvecklingen av en applikation till både Android och Windows Phone.

3. Teknisk Bakgrund

3.1 Introduktion till att skapa applikationer till mobila plattformar

Applikationer till mobila plattformar måste ta hansyn till och fokusera på enkel och snabb användning. Många användare är “On the Go” och har en begränsad tid, till exempel väntar på bussen eller tar en kort rast. Applikationerna måste då vara smidiga så användarna snabbt kan navigera till funktionen de är intresserad av.[1]

3.2 Android

3.2.1 Riktlinjer för design till Android

När man skapar applikationer för Android finns det vissa designtekniska riktlinjer man ska tänka på. De riktlinjer som tas upp här gäller för senaste versionen av Android som går under kodnamnet ”Ice Cream Sandwich” men de mer generella principerna gäller även för äldre versioner. När Google skapade den nya versionen hade de tre övergripande mål som låg till grund för både systemet och dess applikationer. När utvecklare själva skapar applikationer ska dessa tas i beaktning.

- Applikationer till Android ska vara estetiskt välgjorda och tilltala användarna på flera olika nivåer. Placeringen av grafiska komponenter ska vara genomtänkt och underlätta för navigation på ett naturligt sätt. Byte mellan olika vyer ska ske på ett snabbt sätt för att behålla responsiviteten gentemot användaren. Som utvecklare ska man sträva mot att kombinera snygghet, enkelhet och syfte i applikationen.
- Applikationer till Android ska vara lätta att förstå och första gången en person använder dem ska de intuitivt få grepp om de viktigaste funktionerna. Komplexa arbeten ska förenklas så det både är lätt att förstå och vara möjligt att uträtta med touch-rörelser. Människor av alla åldrar och kulturer förväntas använda applikationen och den ska vara anpassad därefter.
- Android uppmuntrar användare och utvecklare till att testa nya saker och finna nya tillämpningar för applikationer. Genom att använda funktioner som notifieringar och delning av information mellan olika applikationer kan nya arbetsmetoder skapas.[5]

För att underlätta för användaren ska applikationen ta beslut och göra så mycket som möjligt åt denna istället för att besvära den med frågor, men också tillåta användaren att ändra sina val om så önskas. Ett ytterligare led i detta är att lära känna användaren med tiden och till exempel spara val som användaren gjort tidigare, istället för att fråga om samma sak upprepade gånger.

Gränssnittet ska försökas hållas rent då användaren lätt blir förvirrad med för mycket knappar och information på samma gång. Informationen eller uppgiften

kan istället delas upp i mindre delar så den blir lättare för användaren att ta till sig. Som ses i figur 3.1 så kan funktioner som inte används för tillfället gömmas undan så det blir lättare att få en överblick.

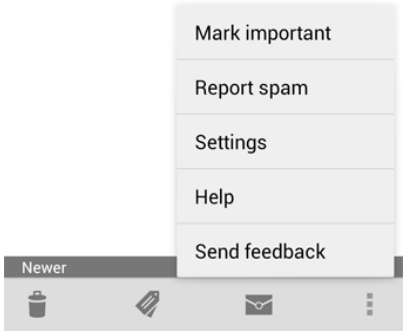


fig 3.1. Göm information som inte används för tillfället[6]

Android's designers påpekar också en annan viktig sak, en komponent som brukar användas för en viss funktion ska man inte använda för en annan. Har den samma utseende ska den också ha samma funktion. Detta gäller både inom och mellan applikationer. Detta konceptet visualiseras i figur 3.2 där tre olika applikationer använder samma komponent. Triangeln efter texten indikerar att det finns en meny som visas om man klickar på den. Detta beteende ska man inte ändra på och att göra något annat än att visa en meny skulle förvirra för användarna. [6]

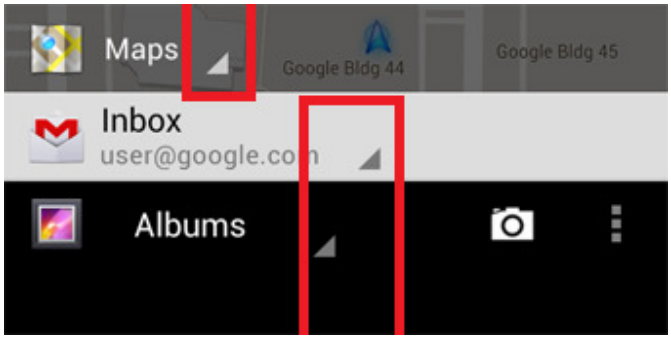


fig 3.2. Samma komponent med samma funktion i olika applikationer[6]

Android körs på miljontals enheter och många olika sorters hårdvara. Som utvecklare måste man vara medveten om vilka krav och begränsningar detta innebär och anpassa sin applikation. Det som utmärker sig mest mellan olika hårdvaror när det gäller designen är den stora bredden av skärmupplösningar.[7] När man skapar en applikation måste man ta hänsyn till detta och skapa layouter som är oberoende av antalet pixlar i upplösningen så kallad 'density independent'. Annars skulle samma komponent bli större på en lågupplöst skärm jämfört med en skärm med högre upplösning. För att att underlätta när man designar kan man dela in skillnaderna av skärmstorlek och pixeltäthet i fyra olika grupper. Det är rekommenderat att skapa resurser(bilder, knappar osv) och även extra layouter som är anpassade för de olika pixeltätheterna och skärmupplösningarna.[8]

3.3.2 Activity

En applikation för Android som ska kunna interagera och visa information för användaren är uppbyggd av en eller flera Activities. En Activity är en komponent som oftast täcker hela skärmen men kan även vara transparent eller bara täcka en viss del. Den är uppbyggd av en hierarki av komponenter så som knappar, listor och textfält.

3.3.2.1 Livscykel

Perioden från det att en Activity startas till att den avslutas eller att systemet av någon anledning behöver döda den kallas för dess livscykel. Under livscykeln befinner sig en Activity i olika tillstånd vilka är:

- **Aktiv** – när en Activity är i förgrunden och visas på skärmen
- **Pausad** – när en annan Activity är i förgrunden men täcker bara en viss del av skärmen eller är transparent så att den tidigare Activity:n syns i bakgrunden
- **Stoppad** – när en annan Activity tilldelats fokus och helt täcker den tidigare Activity:n

En Activity som är aktiv kommer få den körtid den kräver. Om den blir pausad kommer den fortfarande vara vid liv och endast dödas av systemet vid extremt hög belastning på minnet. När en Activity är stoppad finns stor risk för att systemet kommer att döda den när minne krävs av andra Activities. När en Activity byter tillstånd kommer detta meddelas i dess olika "callback methods". De olika metoder som används visas i figur 3.3. De tre viktigaste metoderna är:

- **onCreate() -** Anropas alltid när en Activity skapas för första gången. Här utförs all initial setup av variabler, lyssnare, koppling av data till listor osv.
- **onPause()** - Anropas alltid när en Activity tas bort från förgrunden. Denna metod är sista stället som med säkerhet kommer få körtid innan systemet eventuellt kommer döda Activityn. Här bör data sparas ned och arbete som kan förbruka CPU-kraft avslutas.
- **onResume()** - Anropas alltid när en Activity placeras i förgrunden. Här kan tillståndet hos en Activity återskapas med tex data som sparats ned i onPause() och krävande arbete kan återtas.

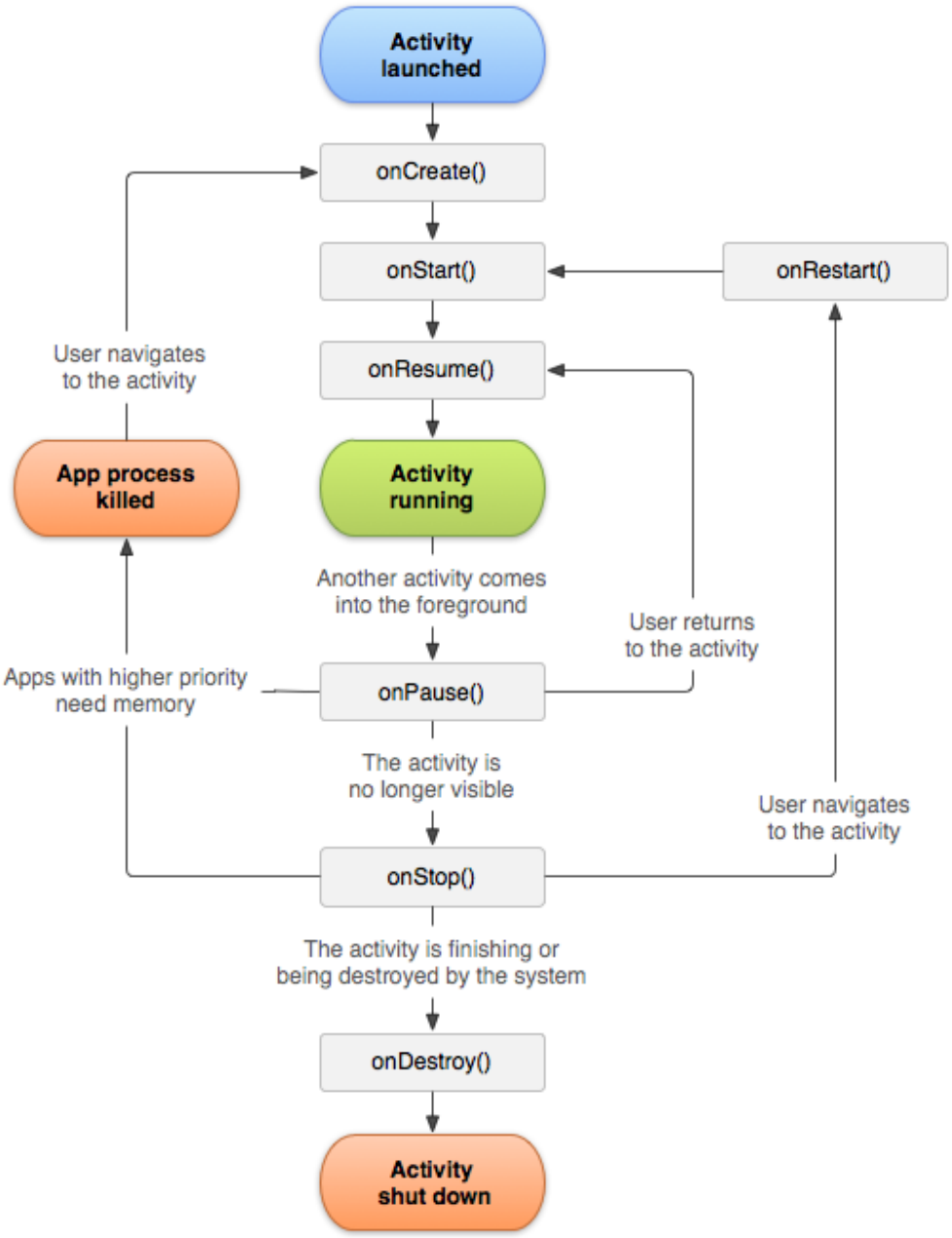


fig 3.3. Livscykeln hos en Activity och flödet hos dess "callback methods" [21].

3.3.2.2 Back stack och Task

Alla Activities som startas placeras på en "Back stack". Varje gång en ny Activity startas så kommer den tidigare att stoppas och den nya placeras överst på stacken. När en Activity stoppas så sparar systemet tillståndet på dess gränssnitt för att snabbt kunna återskapa den vid ett senare tillfälle. En Activity kan bara placeras eller plockas bort från toppen på stacken och så länge som nya Activities startas så byggs stacken på. När användaren istället trycker på "bakåtknappen" så kommer

den aktiva Activity:n plockas bort från stacken och kastas. Nästa Activity på stacken blir då aktiv. Figur 3.4 nedan visar på hur en Back stack byggs upp.

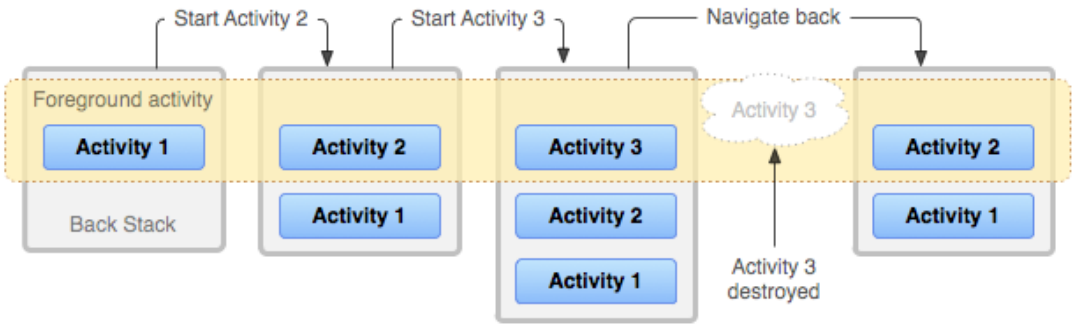


fig 3.4. Uppbyggnad av en Back stack med nya Activities [21]

En Task kan ses som en slags behållare för Activities. När man startar en ny applikation från hemskärmen så startas en ny Task och varje Task har sin egen Back stack. För varje ny Activity som startas på en Task så byggs dess Back stack upp. Om användaren trycker på "hem-knappen" så kommer den aktiva Task:en och alla dess Activities stoppas. Om användaren då väljer att starta en ny applikation kommer en ny Task skapas med en egen Back stack. Om användaren trycker på "hem-knappen" och sedan väljer att starta en applikation som redan befinner sig i en Task så kommer den istället sätta denna Task som aktiv och den Activity som befinner sig överst på stacken kommer återskapas och sättas i förgrunden. Figur 3.5 nedan visar på hur detta kan se ut.

All information i detta avsnitt som behandlar Activity och dess livscykel är hämtad från Google [21].

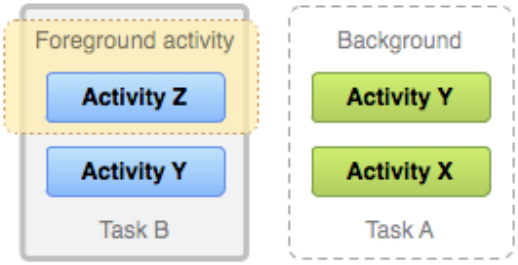


fig 3.5. Task A har stoppats och placerats i bakgrunden i väntan på att återupptas. Task B har valts som aktiv [21].

3.2.3 XML

XML står för eXtensible Markup Language och är ett plattformsoberoende språk designat för att struktuera data. XML är en vidarutveckling av SGML och blev en W3C-rekommendation i februari 1998. W3C-rekommendationen specifierar bland annat hur XML ska struktueras. XML använder märken för att avgränsa delar av data från varandra. Ett märkord avgränsas av '<' och '>'.[10] Ett element består av av en starttagg och en sluttagg och kan innehålla attribut. Element är byggstenarna i ett XML-dokument och syntaxen ser ut så här.[11]

$$<name\ attribute = "value">content</name>$$

Eftersom XML lägger till mer data i form av taggar så kommer en fil i XML vara större än motsvarande binär fil. Detta är ett medvetet val för att strukturera data så den bli läsbar för människor utan att behöva använda program som konverterar. I XML finns det inga fördefinerade taggar utan dessa måste defineras genom att skapa egna paket och moduler så kallade namnrymder.[11] För att avgöra om XML-dokument är giltliga används mallar skapade i ett 'schema'-språk. Det senaste av dessa språk heter 'XML-schema' och en mall skapad i detta språk är ett dokument som innehåller grammatiska regler för hur andra XML-dokument ska vara strukturerade.

3.2.4 XML i Android

I android använder man XML primärt för att skapa layouter. Layouter är grunden till gränssnitt och håller visuella komponenter och genom att deklarera detta i XML åstadkommer man en tydlig separation mellan gränssnitt och den övriga koden i applikationen. Komponenterna man använder i sina layouter kommer från olika namnrymder i Android och dessa måste man deklarera i filen för att kunna använda.[12] Även om man skriver sitt program i XML så är det inte detta som sen körs på telefonen. IDE:n kallar på Android's resurskompilator som bearbetar XML och konverterar till ett komprimerat binärt format. I det binära formatet är koden bättre anpassad att köras på Android som trots allt är optimerad att köra på mobila enheter med begränsningar av både minne och beräkningskraft[13].I figur 3.6 visas XML-koden för en simpel layout bestående av en textruta med texten "Hello, I am a TextView" och under den en knapp med texten "Hello, I am a Button".

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
              android:layout_width="fill_parent"
              android:layout_height="fill_parent"
              android:orientation="vertical" >
    <TextView android:id="@+id/text"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:text="Hello, I am a TextView" />
    <Button android:id="@+id/button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Hello, I am a Button" />
</LinearLayout>
```

fig 3.6. Layout för Android

3.2.4 Model View Controller (MVC)

All information som behandlar MVC-modellen i detta avsnitt är hämtad från Microsoft [19].

MVC är ett designmönster där en applikation är uppdelad i tre delar:

- Model – hanterar applikationens data och dess tillstånd.
- View – hanterar det grafiska gränssnittet och all presentation av data för användaren.
- Controller – hanterar all typ av inmatning och påverkan från användaren.

Figur 3.7 visar kopplingen mellan de olika delarna.

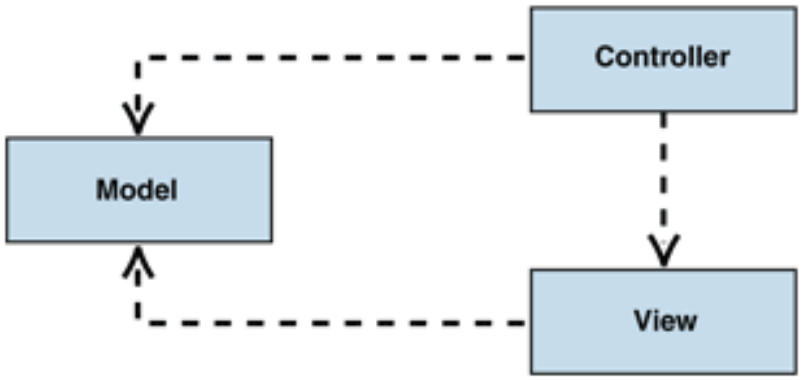


fig 3.7. De olika delarna i MVC-modellen och dess koppling till varandra

Både Controller och View känner till Model medan Model inte känner till någon annan del. På så sätt blir kopplingen låg för Model och det blir lätt att byta ut det grafiska gränssnittet för applikationen samt göra tester mot Model.

3.2.4.1 MVC i Android

Google anger ingen rekommenderad designmodell att använda när man utvecklar applikationer till Android. När man söker efter information om ämnet på internet så finns flera olika förslag men inga som direkt passar för utveckling till Android.

I en serie blogginlägg[17] beskriver Joshua Musselwhite hur MVC kan ses som en möjlig designmodell att utveckla efter. Han beskriver hur Activities kan användas som View men att detta inte är helt oproblematiskt. I Android är det Activities som ansvarar för att hantera användarens påverkan på applikationen vilket i normala fall är en uppgift för Controller. En lösning som Joshua föreslår är att skapa Controller-klasser. Dessa klasser kan Activities använda sig av genom att överlåta all hantering av händelser. Vidare beskriver han hur Model kan representeras av klasser som hanterar data och tillstånd. Med hjälp av Observer-mönstret kan View meddelas när data ändras och gränssnittet uppdateras därefter. Controller-klasserna har referenser till klasserna i Model och kan på så vis manipulera data vid användarens inmatningar. Figur 3.8 illustrerar hur hantering av händelser separeras från Activity och hanteras istället av Controller-klasser.

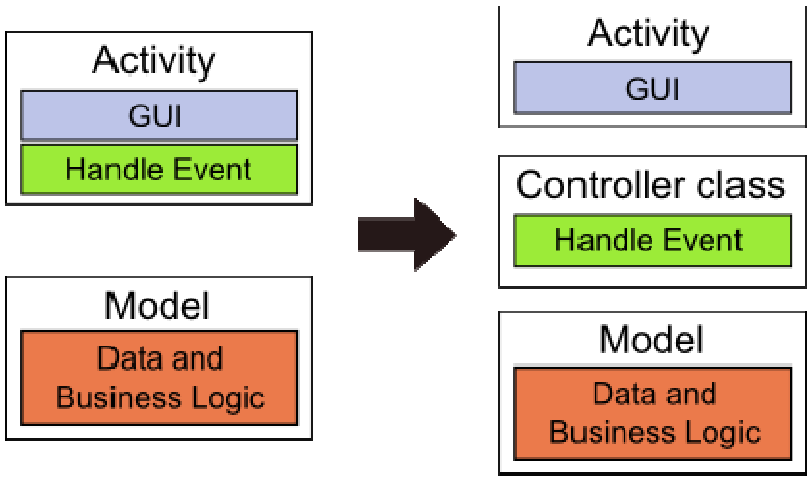


fig 3.8. Hantering av händelser överfört till separata Controller-klasser.

3.2.5 Säkerhet

All information i detta avsnitt som behandlar säkerhet i Android är hämtad från projektet Android open source [22].

Utvecklargruppen för Android har redan från början strävat efter en robust och säker plattform. Genom att bygga Android ovanpå Linux-kärnan erhålls en stabil grund som redan genomgått omfattande säkerhetstester.

3.2.5.1 Sandboxing

En av de viktigaste egenskaperna i Android vad gäller säkerhet är hur applikationer isoleras från varandra genom att utnyttja egenskaper i Linux, vilket kallas Sandboxing. Varje applikation erhåller ett unikt användarid och körs sedan som denna användare i en egen process. Varje applikation tilldelas ett minnesutrymme där bara den applikation med samma användarid som minnesutrymmet tilldelats till har rättighet att spara eller skriva. På så vis skyddas en applikation från att andra, möjligen skadliga, applikationer skulle kunna manipulera eller ta del av känslig data. Figur 3.9 visar arkitekturen för Android och de olika komponenterna som Android och applikationer för Android är uppbyggda av. Sandboxing implementeras redan i Linux-kärnan vilket betyder att all mjukvara som befinner sig över kärnan kommer att isoleras på samma vis.

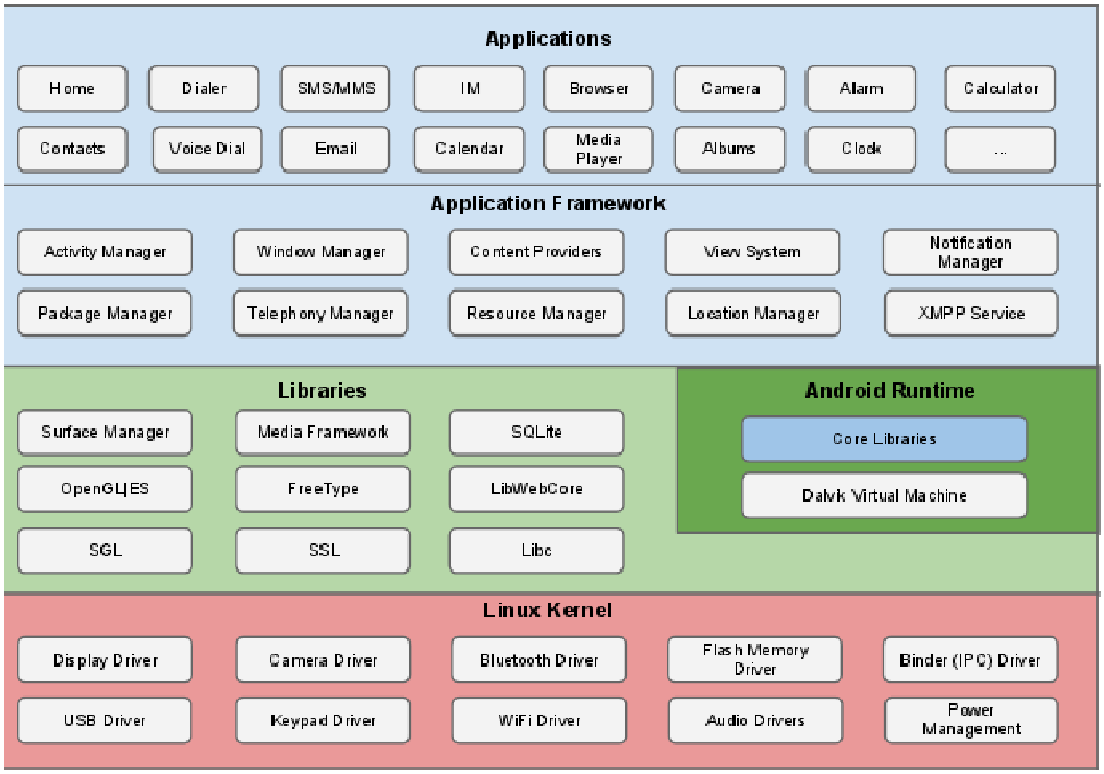


fig 3.9. Arkitekturen hos Android som visar de olika lagren och komponenter som plattformen byggs upp av [22].

3.2.5.2 Tillstånd

En annan säkerhetsåtgärd i Android är att vissa funktioner och viss data som kan vara känsliga eller kosta pengar för användaren kräver att applikationen har tillstånd från systemet för att kunna ta del av dem. Exempel på tillstånd är tillgång till internet, gps, kontakter och sms. Varje tillstånd som en applikation ber om meddelas till användaren före installation. Utan godkännande av användaren kommer inte applikationen att installeras. Om en applikation försöker använda en funktion eller ta del av data utan att ha bett om tillstånd så kommer Android avsluta applikationen.

3.2.5.3 Kommunikation mellan applikationer

Av säkerhetsskäl så har Android satt som standard att olika applikationer inte kan kommunicera med varandra. Vid vissa tillfällen kan det dock vara nödvändigt för applikationer att kunna kommunicera och ta del av en annan applikations data. Android tillhandahåller olika mekanismer som tillåter detta. Genom att en applikation kan ange olika tillstånd för andra applikationer kan de ta del av dess data eller att anropa vissa funktioner hos applikationen. Exempel på detta är att läsa användarens lista av kontakter eller att starta kamerafunktionen hos telefonen.

3.3 Windows Phone

3.3.1 Riktlinjer för design till Windows Phone

Riktlinjer för design till Windows Phone är framtagna av Microsoft för att hjälpa utvecklare att skapa bra applikationer och till fullo utnyttja vad plattformen har att erbjuda. Riktlinjerna finns för att skapa en enhetlig design av WP-applikationer och för att underlätta för användarna så de ska känna igen sig och veta hur man navigerar även om det är första gången de använder applikationen. Riktlinjerna säger ingenting om hur programmet ska fungera rent programmatiskt utan mer hur det ska se ut visuellt samt hur användaren ska kunna interagera med innehållet.

Windows Phone designsystem har kodnamnet Metro och kännetecknas av minimalistiska applikationsvyer som är fria från plottriga knappar och som går snabbt att navigera i. "*Content is the Interface*" - beskriver på ett bra sätt hur Microsoft vill att applikationer till Windows Phone ska fungera. Användaren ska interagera direkt med innehållet i form av bilder och text utan ett lager emellan för att snabba upp navigering. I fig 3.10 visas hur Microsoft själva designat sin MSN Money applikation och den är ett bra exempel på minimalistisk och modern design.[1]

Microsoft vill att applikationer till Windows Phone ska ha en känsla av responsivitet. Gränsnittet ska snabbt svara på interaktion från användaren och animationer ska ge intryck av ett "bättre flyt". Genom att visa en liten del av nästa vy, som i fig 3.10, vill Microsoft att upplevelsen ska kännas mer intuitiv.

Användaren ser då att att det finns något mer som inte får plats, och genom den naturliga rörelsen att dra åt vänster kommer denna information att visas. [2]

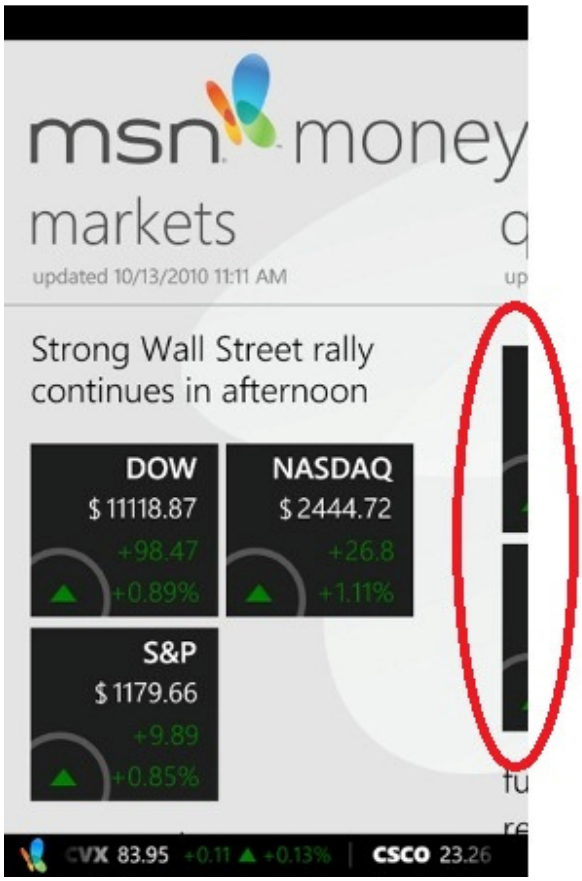


fig 3.10. MSN Money applikation till Windows Phone[1]

Ett ytterligare led i att skapa ett responsivt gränsnitt är att användaren ska få ett gensvar när något som är klickbart klickas på. Detta är för att användaren direkt ska kunna se vad som går att klicka på samt visa att klicket verkligen registrerades. Exempel på dessa gensvar är små vinklingsanimationer som kan appliceras på bilder och textfält. När dessa klickas på så vinklas objekten åt det hållet man klickat och denna till synes lilla visuella effekt är tillräcklig för att användaren ska veta att klicket registrerats.[2] Den visuella effekten illustreras på bild 3.11. Det streckade området är komponentens ursprungliga position. Den röda punkt är positionen för trycket av användaren. Den heldragna ramen av vitt visar fomen som komponenten kommer att animeras till under trycket.

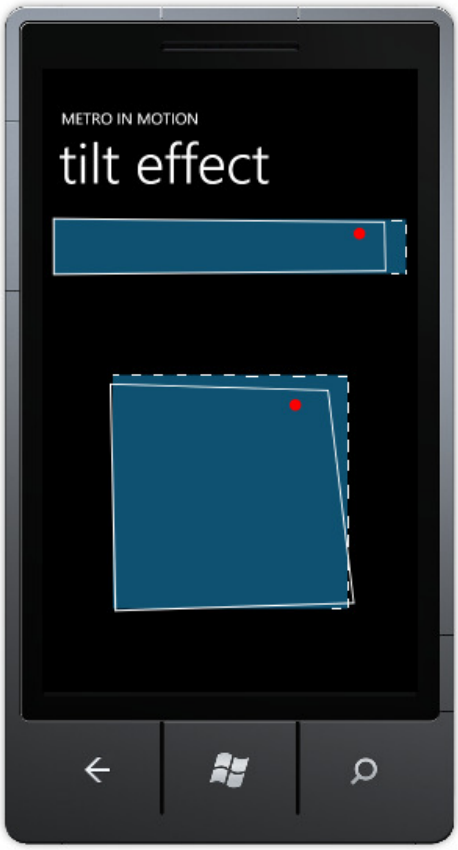


fig 3.11. Visualisering av vinklingseffekten på grafiska komponenter i Windows Phone.[9]

Utöver krav på den visuella känslan och upplevelsen ställer Microsoft mer definierade krav på komponenterna i det grafiska gränssnittet. Till exempel så ska en komponent som ska svara på beröring inte vara mindre än en kvadrat på 7x7mm, och den rekommenderade storleken är minst 9x9mm. Minimumvärdet på 7mm ska endast användas när ytan är väldigt begränsad och då rekommenderas att komponenten har en större bredd istället. Utöver detta finns det till exempel krav på hur stort mellanrummet mellan olika komponenter ska vara, vilken storlek på texten som ska användas samt hur knappar i applikationen ska placeras[3].
Alla telefoner med Windows Phone är utrustade med en “Back”-knapp och dess funktion ska även integreras med applikationen. Det är inte tillåtet att ha en egen “Back” eller “Close”- knapp utan det är endast hårdvaruknappen som ska användas till detta för att undvika förvirring för användarna. Knappar som placeras i vyerna ska även dessa följa ett visst mönster och rekommendationerna är att man ska placera så många som möjligt i en färdig menykomponent kallad “Application Bar”, istället för att placera dem fritt eller att skapa ett eget menysystem, se figur 3.12 [4]

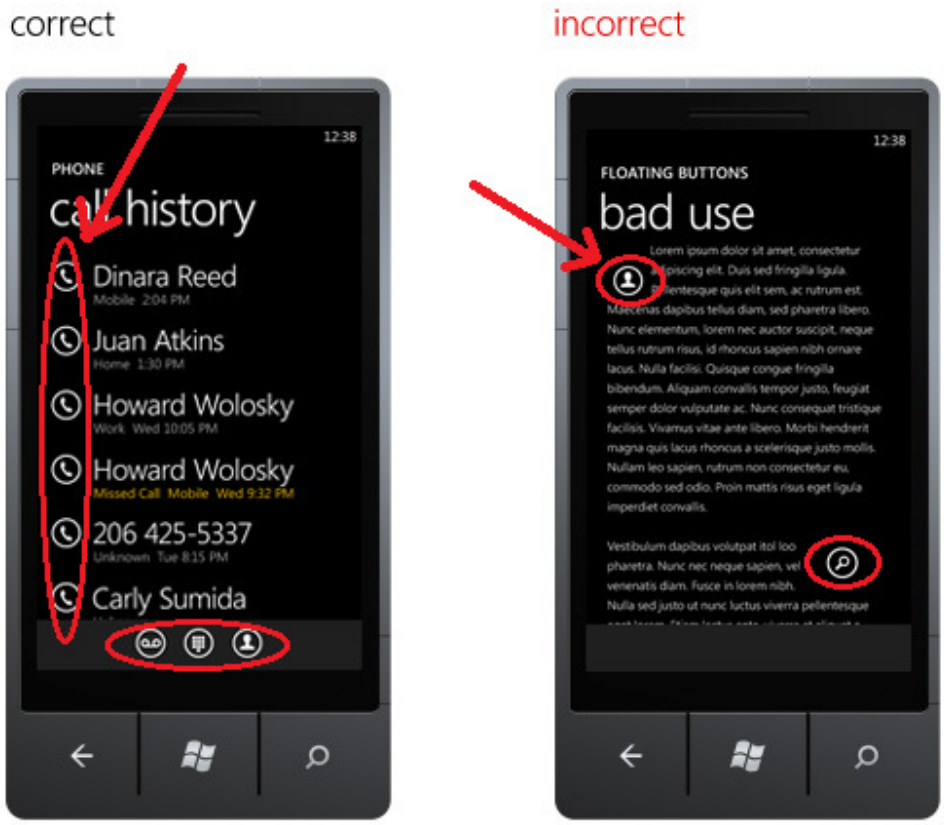


fig 3.12. Rekommenderad placering av knappar[3]

3.3.2 Page

En applikation för Windows phone är uppbyggd av en eller flera Pages. En Page är den del av en applikation som visar data och interagerar med användaren. Den är uppbyggd av en hierarki av komponenter så som knappar, listor och textboxar.

3.3.2.1 Livscykel

Perioden från det att en applikation startas tills dess att användaren väljer att avsluta den eller att systemet avslutar den för att frigöra minne, kallas för applikationens livscykel. Under en applikations livscykel kan den befinna sig i olika tillstånd. De olika tillstånden är:

- **Running** – Efter att en applikation har startats anses den vara i detta tillstånd. Applikationen befinner sig i detta tillstånd tills användaren navigerar ifrån applikationen eller när telefonens låsskärm tar över.

- **Dormant** – Om användaren navigerar ifrån applikationen kommer systemet att placera applikationen i detta tillstånd. I detta tillstånd kommer inte applikationen få någon körtid men systemet kommer att spara den i minnet. Om användaren navigerar tillbaka kommer inte applikationen att behöva återskapas.
- **Tombstoned** – I detta tillstånd har applikationen avslutats men systemet har sparat status för navigationen och även data som kan användas för att återskapa applikationens aktiva tillstånd finns lagrat. När applikationen växlar mellan dessa tillstånd anropas olika metoder där data kan hanteras. De olika metoderna är:
- **Launching** – Anropas när en ny instans av applikationen startas (en tidigare instans ligger ej som Dormant eller Tombstoned). Undvik krävande arbete här för att snabbt ladda applikationen.
- **OnNavigatedFrom** – Anropas när användaren navigerar ifrån en Page. Här bör tillståndet hos den Page man navigerar ifrån sparas för att kunna återställas om användaren återkommer.
- **Deactivated** – När användaren navigerar framåt, ifrån applikationen anropas denna metod. Här bör data som används i applikationen sparas i dess Dictionary. Om applikationen blir Tombstoned och sedan aktiverad igen så kan denna Dictionary, som finns lagrad i minnet, användas istället för att behöva läsa från fil vilket är mer tidskrävande. Eftersom systemet kan välja att helt avsluta applikationen efter denna metod så bör all data även sparas till minnet.
- **Activated** – Anropas när användaren återkommer till applikationen efter att den blivit Deactivated eller Tombstoned. Använd IsApplicationInstancePreserved hos metodens parameter för att avgöra om applikationen har varit Tombstoned och dess Dictionary behöver användas för att återställa applikationen.
- **OnNavigatedTo** – Anropas varje gång användaren navigerar till en Page eller att applikationen varit Dormant eller Tombstoned och sedan startats igen. Om det är en ny instans kan data i Dictionary:n användas för att återskapa gränssnittets tillstånd.
- **Closing** – Anropas när användaren navigerar bakåt, förbi den Page som först laddades när applikationen startade. Applikationen kommer bli helt avslutad och ingen information kommer att sparas. Här bör all data som ska lagras mellan olika instanser av applikationen sparas till minnet. Metodens körtid begränsas till 10 sekunder så inga tidskrävande uppgifter kan göras här utan data bör kontinuerligt sparas under applikationens körtid istället för att allt ska sparas i denna metod.

Figur 3.13 visar en applikations livscykel och de metoder som som anropas mellan applikationens olika tillstånd. Cirklar representerar de olika tillstånden en

applikation befinner sig i. Rektanglar representerar de metoder som anropas, både på applikationsnivå och i Page, mellan de olika tillstånden.

All information i detta avsnitt som behandlar Page och applikationens livscykel är hämtad från Microsoft [20].

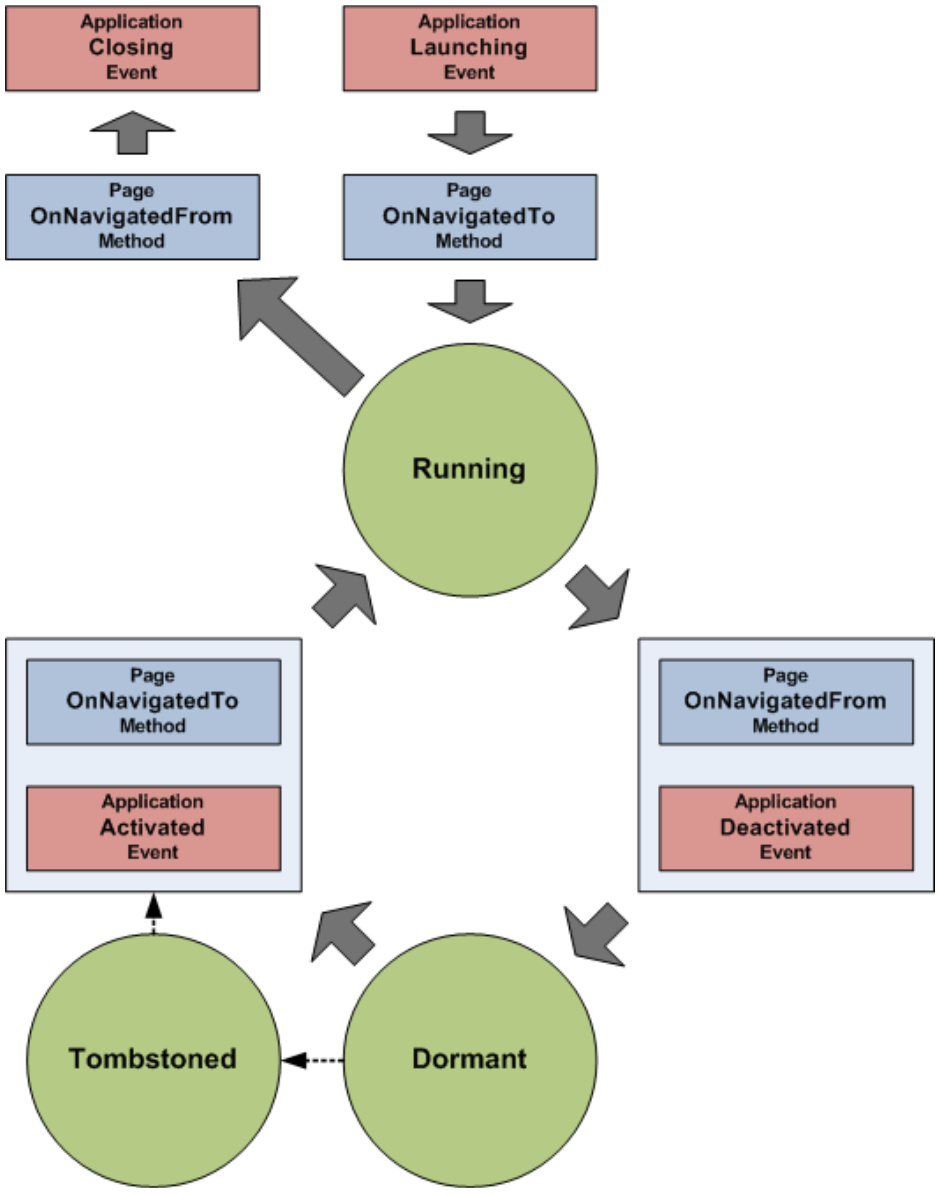


fig 3.13. Visar en applikations livscykel och de olika metoder som kallas mellan de olika tillståndens övergångar [20].

3.3.3 XAML

XAML står för 'eXtensible Application Markup Language' och är ett deklarativt märkspråk.

XAML baseras på och följer de strukturella kraven hos XML men har utökats ytterligare. Till skillnad från XML som måste tolkas och omvandlas till interpreterbar kod så *är* XAML kod som körs vid exekvering. I WP används XAML för att skapa gränssnittet och för att separera denna koden från resten av applikationskoden. En fördel med XAML som kommer med separationen av kod är att det tillåter ett arbetsätt där olika personer skulle kunna jobba på gränssnitt och applikationskod samtidigt och med olika utvecklingsvertyg[14]. Eftersom XAML bygger vidare på XML är en stor del av syntaxen lika men det finns en del nyheter i XAML som är viktiga att känna till.

3.3.3.2 Markup Extensions

Markup Extensions är en utbyggnad av attribut där man istället för att bara referera en sträng eller ett element av ett objekt kan referera till en uppbackande klass. Det som identifierar en Markup Extension är att ett attributs värde börjar med måsvingar {} och efterföljande sträng refererar till vart denna utbyggnad är placerad. Efter detta följer ett mellanslag och sen en sträng som talar om vad implementationen heter. I figur 3.14 sätts textfärgen med hjälp av Markup Extension. StaticResource refererar till klassen StaticResourceExtension som i konstruktorn tar en parameter av typen object. Strängen 'MySpecialColor' ges som argument och fungerar som nyckel till den resurs som är definerad i 3.15. Under exekveringen kommer denna resurs användas för att sätta färgen på texten[15]. Deklarationen av resursen som används ses i figur 3.15 och åtkomst till denna finns nu i hela applikationen.

```
<TextBox Text="Hello, I am a TextBox" Foreground="{StaticResource MySpecialColor}"/>
```

fig 3.14 Textruta i Windows Phone

```
<!--Application Resources-->
<Application.Resources>
    <Color x:Key="dark_grey">#ff808080</Color>
    <SolidColorBrush x:Key="MySpecialColor" Color="{StaticResource dark_grey}"/>
</Application.Resources>
```

fig 3.15. Definition av en resurs som görs åtkomlig i hela applikationen.

3.3.3.3 Data-bindning

Ett annat användningsområde för Markup Extensions som används flitigt i WP-applikationer är data-bindning. Det man vill uppnå med data-bindning är att sammankoppla data från applikation till gränssnittet genom deklaration i XAML[15]. I figur 3.16 sätts texten i ett TextBlock genom data-bindning med text genererad från koden i applikationen. Från det grafiska gränssnittet vet vi inte hur texten genereras och det spelar inte heller någon roll om den hämtas från en databas, en fil eller genom ett webanrop. Texten i TextBlocket binds till en

property i dataklass (se figur 3.17) och vid exekvering kommer denna att hämtas automatiskt. I fig 3.18 skapas ett objekt av dataklassen och referensen tilldelas till komponenten i UI:t.

```
<!--LayoutRoot is the root grid where all page content is placed-->
<Grid x:Name="LayoutRoot" Background="Transparent">
    <StackPanel Orientation="Vertical">
        <TextBlock Text="{Binding DataString}"/>
    </StackPanel>
</Grid>
```

fig 3.16. Data Binding i Windows Phone

```
public class ApplikationData
{
    private string dataString;
    public string DataString
    {
        get { return dataString; }
        set { dataString = value; }
    }
}
```

fig 3.17. Klass som omsluter data.

```
//create a new instance of ApplikationData
ApplikationData appData = new ApplikationData() { DataString = "this represents loads of data" };
//set the LayoutRoot DataContext Property
LayoutRoot.DataContext = appData;
```

fig 3.18. Koppling mellan UI och data sätts upp.

3.3.3.4 Attached Properties

'Attached Properties' är ett koncept som bygger på att egenskaper som finns i ett visst element kan sättas som attribut i ett annat element. Typen av dessa två element behöver inte överensstämma Det främsta användningsområdet är när man vill att nästlade element ska kunna rapportera information till huvudelementet utan att de på något sätt ska behöva dela någon gemensam basklass.[14]
I figur 3.19 illustreras användandet av attached properties. Klassen DockPanel har en Attached Property DockPanel.Dock som i detta exempel används för att positionera knappar inuti elementet.

```
<DockPanel>
  <Button DockPanel.Dock="Left" Width="100" Height="20">I am on the left</Button>
  <Button DockPanel.Dock="Right" Width="100" Height="20">I am on the right</Button>
</DockPanel>
```

fig 3.19. Exempel på Attached Properties i Windows Phone [14]

3.3.4 Model-View-ViewModel (MVVM)

All information som behandlar MVVM i detta avsnitt är hämtad från Microsoft [18].

MVVM är det designmönster som Microsoft rekommenderar att använda när man utvecklar applikationer för WP. I MVVM är applikationen uppdelad i tre olika delar, se figur 3.19.

3.3.2.1 Model

Model är den del som hanterar applikationens data. All applikationslogik som hanterar och bearbetar data hör till Model. Data kan till exempel hämtas från en databas eller webbtjänster.

3.3.2.2 View

View hanterar det grafiska gränsnittet och presentationen av data för användaren. Ingen data eller logik som hanterar data får placeras här. Genom data-bindning (se avsnitt 3.3.3.3) kan View kopplas till ViewModel för att presentera de data som finns i Model och på så vis kan det grafiska gränssnittet vara fristående från all logik som hanterar data. View har referenser till ViewModel men har ingen vetskap om Model.

3.3.2.3 ViewModel

ViewModel är kopplingen mellan View och Model. ViewModel tillhandahåller data som finns i Model genom olika attribut. Attributen anpassas efter den information som ska presenteras i View. Här kan även logik placeras som sammanställer data i Model på lämpligt sätt innan den presenteras för användaren. Med hjälp av de olika attributen kan ViewModel tillhandahålla den data som finns i Model utan att View vet något om bakomliggande logik. ViewModel kan även definiera olika kommandon som View kan presentera för användaren. Dessa kommandon kan till exempel vara sortering av data. Med hjälp av en knapp kan View tillhandahålla denna funktion för användaren och utföra kommandot i ViewModel vid knapptryck.

ViewModel har ingen referens till View men genom tvåvägs data-bindning kommer ändringar i de attribut som finns i ViewModel att avbildas i View.

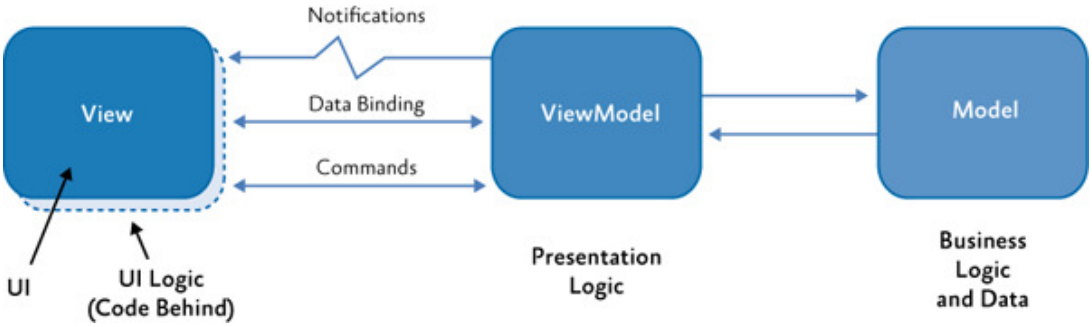


fig 3.19. visar de olika delarna i MVVM och hur de är kopplade till varandra. [18]

3.3.5 Säkerhet

All information i detta kapitel som behandlar säkerheten i Windows phone är tagen från en rapport skriven av Microsoft [23].

Microsoft anser att säkerheten hos mobiltelefoner aldrig varit viktigare och har därför designat Windows Phone med säkerhet som hög prioritet. Microsoft har strävat efter något som de kallar isolering och minsta privilegium. Med detta menar de att isolera applikationer från varandra och ge dem så få rättigheter som möjligt till systemets funktioner och data.

3.3.3.1 Förmåga

I Windows Phone avser en förmåga, en applikations tillgång till en funktion eller information som kan vara känslig eller kosta pengar för användaren. Exempel på förmågor är tillgång till telefonens geografiska position, internet och telefonens kamera. En lista över alla förmågor en applikation besitter presenteras vid installation och kan därefter inte ändras under körning.

3.3.3.2 Kammare

I Windows Phone används olika typer av kammare. Alla applikationer som installeras på telefonen körs i en egen process i en av dessa typer av kammare. Det finns fyra olika typer där tre av dem har olika nivåer av förutbestämda rättigheter. Den fjärde kammaren har minst antal rättigheter men erbjuder applikationer att använda sig av olika förmågor. Det är här alla applikationer av tredjepartsutvecklare installeras från Windows Phone Marketplace. Applikationer som installeras i fjärde kammaren anger olika förmågor för att utöka dess rättigheter för att få tillgång till vissa funktioner och data på telefonen.

3.3.3.3 Sandboxing

Windows phone använder sig av så kallad Sandboxing där varje applikation körs i en egen process i en egen isolerad kammare. Varje applikation får även en viss mängd isolerad lagringsplats. I Windows phone finns inga möjligheter för applikationer att kommunicera med varandra eller att komma åt andra

applikationers data. De enda applikationer som finns tillgänliga är de som följer med WP, till exempel kamera- och sms-applikationen. Detta minskar risken för att skadliga applikationer ska kunna manipulera data och körning av andra applikationer.

4. Genomförande

4.1 Introduktion av företaget Mobisle Apps

MobisleApps skapades 2009 och är ett svenskt mjukvaruföretag som specialiserat sig på applikationer för mobila plattformar. Primärt är det ett konsultföretag som hjälper andra företag att utveckla och färdigställa applikationer. Utöver det har de även en egen applikation vid namn Mobisle Notes som de utvecklar själva. Vi tog kontakt med dem för att vi ville utveckla en applikation och lära oss om mobila plattformar. Det visade sig att Mobisle Notes inte fanns för Windows Phone och vi kom överens om att vi skulle skapa den.

4.2 Mobisle Notes för Windows Phone

4.2.1 Kort om Mobisle Notes

Mobisle Notes är en anteckningsapplikation som fokuserar på enkelhet. Filosofin är att det inte ska vara ett jobb i sig att anteckna utan det ska gå fort att få ner sina tankar och idéer. Därför har MobisleApps gjort ett medvetet val att endast inkludera de viktigaste funktionerna. Några av funktionerna som finns är:

- Bocka i rad när en uppgift är avslutad.
- Byta mellan att se checklista eller vanlig text.
- Skapa mappar och anteckningar samt lösenordskydda dessa.
- Dela en anteckning mellan flera olika användare.
- Säkerhetskopiera och synka anteckningar mellan olika enheter.
- Webbgränssnitt för att kunna se och editera sina anteckningar på internet.

4.2.2 Skapandet av Mobisle Notes för Windows Phone

4.2.2.1 Planering

Det första vi gjorde var att skapa oss en överblick om vilka funktioner som fanns i de redan existerade applikationerna till Android och iOS. Utifrån detta bestämde vi ungefär vilka olika paket som behövdes för att skapa dessa funktioner. Resultatet, som kan ses i figur 4.1, blev en enkel typ av flödesschema som beskrev hur vi tänkte att data skulle röra sig i applikationen. Tanken med detta flödesschema var att eftersträva en bra separation mellan de olika paketen och delarna i applikationen och få en bra grund som skulle bli enkel att bygga vidare på. Databasen blev en central punkt och Pages utväxlar data endast med databasen. WebSync uppdaterar databasen med data från webben men utväxlar aldrig någon data med Pages.

fig 4.1. Flödesschema för data i applikationen.

Efter att vi skapat oss en överblick gick vi vidare med att fundera vilka olika områden vi trodde skulle kräva mycket tid och i vilken ordning det skulle vara bäst att implementera dessa. Resultatet sammanställdes och blev listan nedan:

1. Enkelt gränssnitt
2. Inloggning
3. Lokal Databas
4. Interaktivt gränssnitt skapat efter Wireframes
5. Avancerad lista med animationer när man bockar av anteckningslinjer
6. Inställningar
7. Synkronisera noteringar mellan applikation och databas på webben

4.2.2.3 Applikationen i skrivande stund

I skrivande stund har de största och viktigaste funktionerna implementerats. I detta skick är det en fullt fungerande anteckningsapplikation men den saknar fortfarande en del funktioner som finns i motsvarande Mobisle Notes för Android och iOS. I figur 4.2 till 4.4 visas applikationen och lite av dess funktioner.

Första gången man startar applikationen möts man av välkomstskärmen i figur 4.2. Där får användaren välja mellan att skapa konto, logga in eller välja bort synkronisering. Inloggning krävs för att synkronisering ska fungera. Väljer man bort synkronisering körs applikationen i “offline-mode” och fungerar som en enkel anteckningsapplikation. Då man navigerar vidare från välkomstskärmen kommer man till huvudvyn som illustreras i figur 4.3. Denna fungerar som en utgångspunkt och varje gång man sen startar applikationen kommer man hit. Den visar de senast använda anteckningarna och klickar man på någon av dem kommer man direkt till anteckningen.

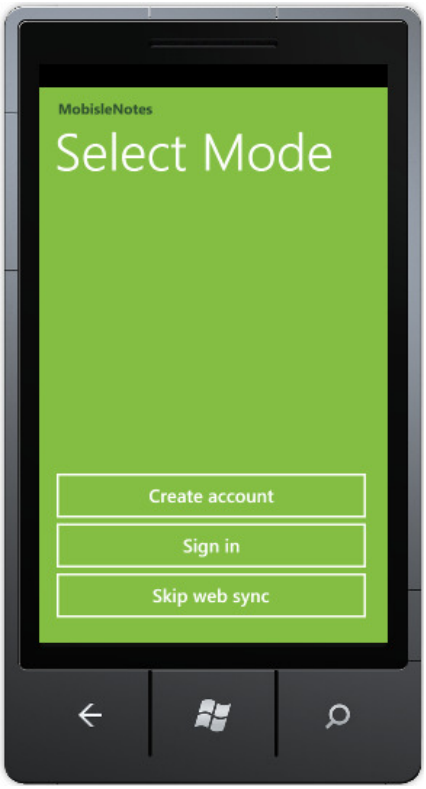


fig 4.2. Startsidan

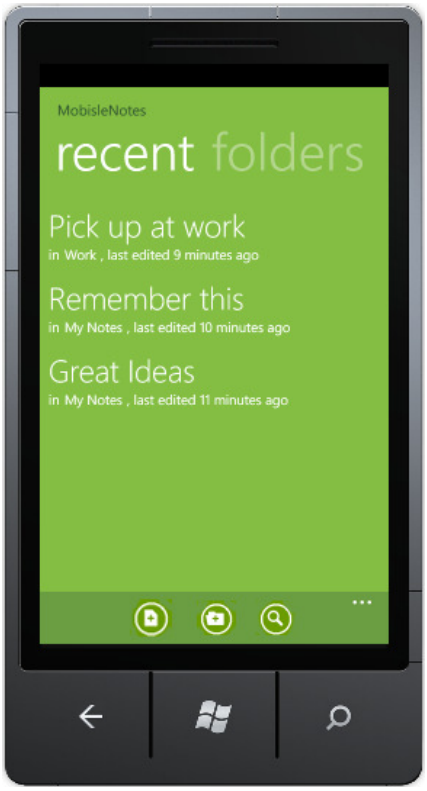
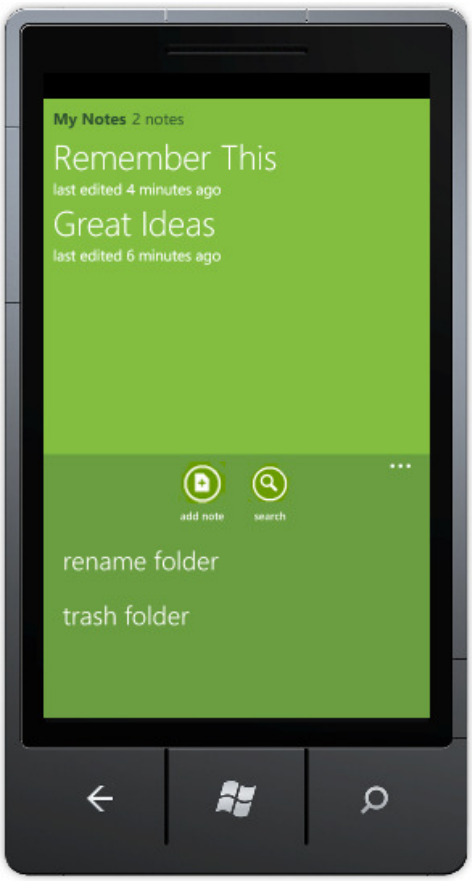


fig 4.3. Huvudvyn i Mobisle Notes WP

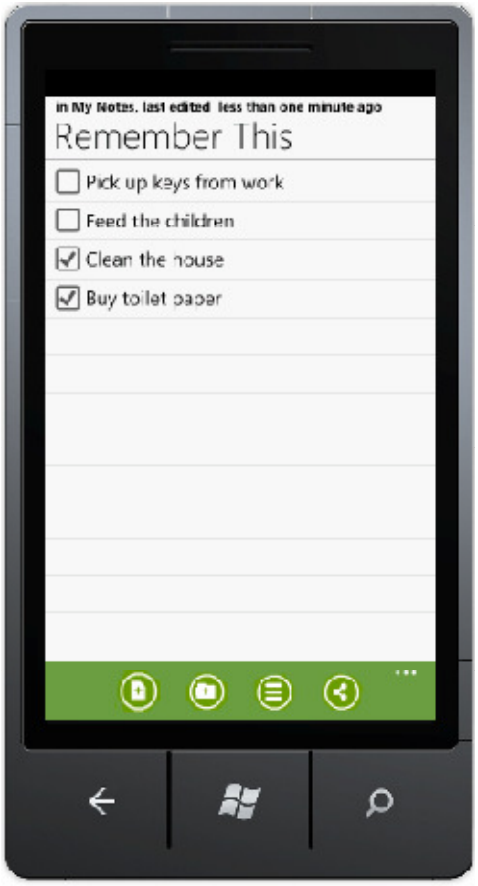
Genom att dra åt höger eller vänster navigerar man vidare till mappvyn som visas i figur 4.4. Där visas användarens mappar och hur många anteckningar dessa innehåller. Där finns också en mapp som heter ‘trash’ som kan liknas med papperskorgen i Windows där alla anteckningar som slängs hamnar. Trycker man på de tre punkterna nere i hörnet öppnas menyn som visas till höger i figur 4.4. Där kan man bland annat skapa nya anteckningar och mappar eller navigera till inställningar.

fig 4.4. Vy med användares mappar

Klickar man på en mapp navigerar man in i den mappen och dess anteckningar visas som i figur 4.5. Uppe i vänstra hörnet visas vilken mapp man är inne i samt hur många anteckningar den innehåller. I menyn finns det val för att lägga till nya anteckningar och byta namn samt ta bort den mappen man befinner sig i.
Klickar man på en anteckning kommer man till den vy som visar anteckningsrader och illustreras i figur 4.6. Längst upp visas information om vilken anteckning man befinner sig i samt när den ändrades senast. Under visas anteckningens titel och sen följer anteckningens rader. Genom att klicka på checkboxen kan man markera rader som avklarade och dessa flyttas då till botten av listan.



4.5. Vy med användares anteckningar



4.6. Exempel på en anteckning

5. Resultat

5.1 Jämförelse av riktlinjer för design

Riktlinjerna för design är skapade för att underlätta för utvecklarna att skapa applikationer som upprätthåller standarder som Microsoft och Android ställer. Många av riktlinjerna som ges är abstrakta som tex att användaren snabbt ska komma åt funktionen de letar efter, gränssnittet ska inte vara plottrigt och upplevas responsivt för användaren. Dessa riktlinjer är svåra att mäta eftersom vad som upplevs plottrigt och förvirrande för vissa är kanske inte det för andra. Men det är också detta som är meningen med riktlinjerna, att utvecklaren ska hålla dem i tanken och fundera kring dem när man skapar sin applikation. Även om riktlinjerna i dessa abstrakta punkter låter lika så blir applikationerna som skapas från dem olika.

- När man jämför det grafiska gränssnittet från WP med Android är det en sak som skiljer sig markant, och det är själva uppbyggnaden av vad användarna ser. I WP ligger fokus på enkelhet och minimalism och texter med stor fontstorlek och bilder utnyttjas som knappar i större utsträckning. I Android däremot så används knappar flitigt och texter fungerar mer för statiskt visande av information.
- Android har förhållandevis få strikta riktlinjer på hur själva innehållet ska placeras utan mer för hur användare ska uppleva det. WP är mer åt det styrande hållet och vill att användare direkt ska kunna se att applikationen är för Windows Phone, därför ska standardfonter och komponenter användas och färger utnyttjas för att skapa en personlig känsla. I Android är det mer en regel än undantag för utvecklare av applikationer att skapa egna komponenter och egna grafiska resurser till tex knappar och logos. Används standardutseendet på komponenter i Android signalerar detta att utvecklaren inte lagt ner så mycket tid eller är oerfaren.
- Både WP och Android's riktlinjer rekommenderar att man på något sätt visar en respons när användaren har vidrört en komponent som svarar på rörelser. I WP är detta små tilt-animationer som kan användas på många standardkomponenter. I Android är det inte lika definerat vad som ska hända utan det är mer upp till utvecklaren. Det allra vanligaste är att man byter bildresurs på komponenten så att till exempel en knapp ser intryckt ut.
- När det kommer till Android's riktlinjer utgör en betydande del av den hur man ska designa applikationer för att fungera till den mängd olika skärmar som existerar. Detta görs främst genom att använda densitetsoberoende pixlar och skapa bildresurser med olika upplösningar, men även genom att

skapa extra layouter. I Windows Phone är det inget problem eftersom det just nu bara finns en typ av upplösningar och pixeldensitet som används. Därför vet man att den applikation man skapar kommer se exakt lika dan ut på alla Windows Phone telefoner. Fördelen är att det är enkelt att komma igång och lätt att skapa vyer.

5.2 Jämförelse mellan Activity och Page

Både Windows Phones Page och Androids Activity är de komponenter som visas som en sida på telefonens skärm. Det är de komponenter som bygger upp en applikation och som fungerar som behållare för den hierarki av komponenter som bygger upp sidan. För en utvecklare ser en Page och Activity lika ut på ytan och de kan struktureras på samma sätt. Båda meddelas via metodanrop när navigering sker från eller till dem. Metoderna kan användas för att utföra passande arbete så som att spara data och tillstånd för att kunna återskapa komponenter när användaren återkommer. De viktigaste metoderna som bör användas är:

Android Activity	**Windows Phone Page**	**Beskrivning**
onCreate()	Konstruktor	Anropas när en Page/Activity skapas för första gången. Här kan instansvariabler initieras och annat initialt arbete utföras.
onPause()	OnNavigatedFrom()	Anropas när användaren lämnar en Page/Activity. Här bör data och tillståndet hos den aktiva Page/Activity sparas så att den kan återställas när användaren återkommer.
onResume()	OnNavigatedTo()	Anropas när användaren återkommer till en Page/Activity. Om det är en ny instans av en Page/Activity kan data som sparats användas för att återställa tillståndet så som det var när användaren lämnade.

Activities i en applikation för Android har sina egna livscykler och kan avslutas av systemet oberoende av varandra. I Windows phone pratar man istället om applikationens livscykel där alla Pages i en applikation har olika metoder som anropas när användaren lämnar eller kommer till en Page men har gemensamma metoder för hela applikationen när systemet placerar den i tillstånd som tex Dormant eller Running.

Android Activity	Windows phone applikation	Beskrivning
onStop()	Deactivated	I android sker detta när en annan Activity placeras i förgrunden. I Windows phone sker det när en annan applikation sätts som aktiv.
onStart()	Activated	I android sker det när en Activity blir aktiv efter att ha varit stoppad. I Windows phone sker detta när en applikation blir aktiv efter att den varit Dormant eller Tombstoned.

Varje gång onStop() och onStart() anropas kommer även onPause() respektive onResume() anropas. onPause() är sista metoden som garanterat får körtid innan systemet kan avsluta en Activity. Därför bör onPause() användas istället för onStop() för att avsluta uppgifter och spara data. Av de anledningarna är oftast inte onStart() och onStop() nödvändiga att använda. I Windows phone är dock Deactivated och Activated de enda metoderna där man kan avgöra när applikationen försätts i eller återupptas från tillstånd som Dormant eller Tombstoned. De är därför lämpliga metoder för att spara eller återuppta applikationsdata och återställa applikationens tillstånd.

5.3 Jämförelse av XAML och XML

- Alla XAML-dokument är giltiga XML-dokument men det omvända gäller inte.[16]
- XML är ett märkspråk som i Android används för att *beskriva* hur gränssnitt och datatyper ska se ut. XML-koden måste sen konverteras till ett annat format innan det kan tolkas.
- XAML är ett deklarativt språk och *beskriver* inte kod, utan det *är* kod. Komponenter i XAML instansieras vid exekvering och kan användas för direkt åtkomst av resurser.
- XAML för WP har bra uppbackning i form av program när det gäller skapande av gränsnitt. Microsoft Expression Blend är ett fristående program där en designer kan skapa gränssnitt samtidigt som andra utvecklar den bakomliggande koden.
- För Android finns plugin för eclipse där man kan skapa komponenter grafiskt och XML-dokument genereras automatiskt.

- XML är plattformsoberoende medan XAML endast finns implementerat för .Net och Microsofts plattformar.
- XAML är en vidareutveckling av XML och därför har språkligt många likheter när det gäller grundsyntax och uppbyggnad av dokument.
- XAML har Markup Extensions som är en utbyggnad av attribut i XML
- XAML har 'Attached Properties'

5.4 MVC och MVVM

MVVM är snarlik MVC där de båda designmönstren delar upp applikationer i tre olika delar. Både MVC och MVVM har en Model och View men i MVVM är Controller utbytt mot ViewModel. En av de tydliga skillnaderna mellan dem är hur både View och Controller i MVC är kopplade till Model och kan använda datan oberoende av varandra medan i MVVM har endast ViewModel tillgång till Model. View i sin tur använder sig av ViewModel för att presentera datan för användaren. ViewModel sammanställer och behandlar datan från Model och tillhandahåller sedan informationen genom olika attribut. Både Controller och ViewModel tillhandahåller olika kommandon som View kan representera i gränssnittet för användaren. Exempel på kommandon kan vara att skicka data till en webtjänst. View kan till exempel representera detta med en knapp som vid tryck anropar kommandot.

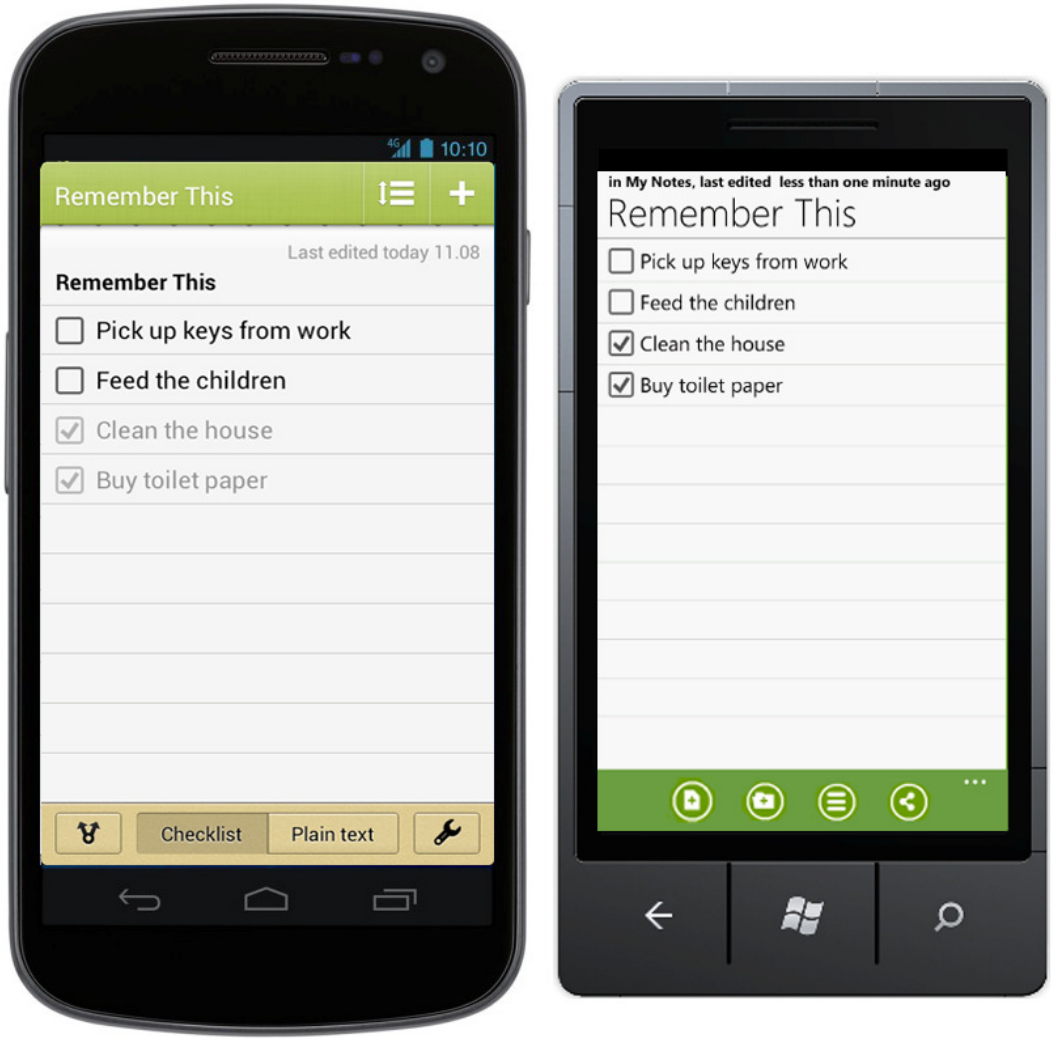
5.5 Jämförelse av gränssnitt mellan Android och Windows Phone

För att ge en klarare bild hur gränssnitt skiljer sig åt mellan Android och Windows Phone ska vi här göra en jämförelse mellan en och samma applikation som är skapad till dessa olika plattformar. Som jämförelseunderlag använder vi applikationen MobisleNotes. Androidversionen av MobisleNotes är skapad av MobisleApps och Windows Phone versionen är skapad under detta examensarbe.

I figur 5.1 visas vyn med mappar och man ser direkt att strukturen på gränssnittet skiljer sig markant. Gränssnittet till WP är avskalat och minimalistiskt i enlighet med Microsofts riktlinjer för design. Börjar man uppifrån så ser man att Androidversionen har en menykomponent som är specialgjord för just denna applikationen. Där är knapparna för att lägga till och ta bort mappar placerade medan i WP-versionen är dessa placerade i en applikationsmeny som ligger längst ned. Generellt sett används standardkomponenter i hög utsträckning på WP eftersom dessa är designade av Microsoft för att passa in. Vidare ser man att de olika listobjekten representeras på olika sätt. I Android är det en ikon och en text som representerar en mapp medan på WP är det text med hög textstorlek. Texten presenteras med en standardfont som rekommenderas av Microsoft och som används flitigt i WP-applikationer.

5.1. Jämförelse av MobisleNotes för Android och Windows Phone

I figur 5.2 visas anteckningsvyn och här har de olika applikationerna mer gemensamt. Vyn är uppbyggd på liknande sett med fokus på en lista av anteckningar. Kryssrutorna är samma och linjerna mellan de olika raderna finns där. Precis som i figur 5.1 så är det även här så att knapparna från toppmenyn i Android finns i applikationsmenyn i botten på WP-versionen. Överst i bild på WP-versionen är istället information om anteckningen placerad, bland annat vilken mapp den tillhör samt när den ändrades senast.



5.2. Jämförelse av MobisleNotes för Android och Windows Phone

5.6 Jämförelse av säkerhet i Windows Phone och Android

Både i Android och WP uttrycks hur viktig säkerheten varit i utformandet av plattformarna. De använder sig av så kallad Sandboxing där de helt isolerar applikationer från varandra. På så sätt minskar risken för att skadliga applikationer ska kunna läsa eller manipulera andra applikationers data och funktioner. Något som skiljer sig åt är att i Android finns möjligheten att kunna dela information och funktionalitet mellan applikationer om så behövs medan i WP är detta inte möjligt.

För att kunna använda sig av viss data eller vissa funktioner som kan vara känsliga eller kosta pengar för användaren krävs att applikationer på båda plattformarna får tillstånd från användaren. Tillstånden presenteras före

installation och om användaren godkänner tillstånden och installerar applikationen kommer inte dessa tillstånd kunna ändras under körning.

5.7 Riktlinjer för utveckling av en applikation till Android och Windows Phone

Dessa riktlinjer är tänkt att användas av utvecklare som ett hjälpmedel för att underlätta och effektivisera struktureringen och designen av en applikation som ska utvecklas till både Windows Phone och Android.

5.7.1 Grafiskt Gränssnitt

- Applikationer till Windows Phone och Android har utseendemässigt olika gränssnitt och detta följer direkt ur deras skiljda riktlinjer för design. Detta leder till att samma applikation som ska utvecklas till båda plattformarna kräver två visuellt olika gränssnitt. För att skapa ett liknande tema på applikationerna mellan de olika plattformarna kan med fördel samma färger, typsnitt och logos användas istället.
- I WP används ofta standardkomponenter för att skapa det speciella WP-temat. Till Android gör utvecklare i högre utsträckning egna komponenter som kan vara skräddarsydda för just den applikationen.
- Som följd av att Microsoft har hårdare riktlinjer för design kan det vara lämpligt att först designa applikationens gränssnitt till Windows Phone. Genom att gå från mer begränsad design till mindre begränsad på Android kan man enklare skapa gränssnitt till båda plattformarna som fungerar på liknande sätt och innehåller samma funktioner, utan att göra kompromisser som tummar på kvalitén.
- För att bygga gränssnitt till WP används XAML som via data-bindning kopplar samman gränssnitt och applikationskod. Detta finns inte i XML och motsvarande kopplingar i Android måste därför göras helt i bakomliggande applikationskod istället.

5.7.2 Bakomliggande applikationsstruktur

- Microsoft förespråkar användandet av MVVM som designmönster för applikationer till WP medan i Android finns det inget rekommenderat designmönster.
- Eftersom applikationer som följer MVC och MVVM delar upp kod på liknande sätt skulle MVC kunna användas vid utveckling av applikationer till Android. På så sätt kan utvecklare använda sig av liknande struktur för en applikation till båda plattformarna.
- I Android byggs applikationer upp av Activities där kopplingen mellan gränssnitt och logik som hanterar händelser är hög. För att utveckla applikationer med MVC som bakomliggande designmönster kan Activities använda sig av styrklasser som tar över ansvaret för att ta hand om olika händelser. På så sätt kan man använda Activities som View i MVC. För att sedan uppdatera View med data från Model kan man använda sig av Observer-mönstret.

- I Android finns möjligheten att låta applikationer ta del av data och funktioner från en annan applikation. Detta är inte möjligt i WP vilket bör tas i beaktning när man utvecklar en applikation så att inte funktioner som implementeras inte fungerar på den andra plattformen.

6. Slutsats

6.1 Värdering av resultat & Metodkritik

Efter att ha utvecklat en applikation till en viss plattform kan stora förändringar i kod och struktur krävas för att utveckla samma applikation till en annan plattform. Eftersom detta är ett problem för utvecklare har vi valt att göra en jämförelse mellan utveckling för Android och Windows Phone och sammanställt ritktlinjer som kan användas för att underlätta utvecklandet av en applikation för båda plattformarna. Vi valde att utveckla en applikation för WP där vi under tiden noterade skillnader mellan WP och Android som vi sedan gjorde en jämförelse på. Eftersom vi inte hade några tidigare kunskaper om utveckling för WP så var det ett bra sätt för oss att samtidigt lära oss på. För att tydligare se de skillnader som finns mellan Android och WP tror vi att det hade gett mer om vi utvecklat samma applikation till båda plattformarna. Hade vi haft tidigare kunskaper om WP hade vi dock valt ett annat tillvägagångssätt då själva utvecklandet av applikationen var tidskrävande. Vi känner att det finns mycket mer att jämföra och hade vi haft mer tid finns det flera saker vi vet redan nu att vi skulle vilja ta upp. Något som vi också tror påverkat riktlinjerna och även mängden jämförelser som skulle kunna utföras är vår, i sammanhanget, begränsade kunskap om plattformarna. Med mer kunskap skulle fler jämförelser kunna utföras och speciellt kunna leda till mer utförliga riktlinjer och tips på lösningar för att underlätta för utvecklare. Vi tycker att det skulle krävas fler riktlinjer innan dokumentet blir till någon större hjälp för utvecklare men att det är en bra början som skulle kunna kompletteras med tiden.

För att kunna utföra jämförelserna behövde vi göra mycket efterforskningar inom de olika områdena. Detta har gett oss ökade kunskaper inom både Android och WP och gjort att vi känner oss tryggare i att utveckla applikationer för de två plattformarna. Även om WP var en helt ny plattform för oss har vi under examensarbetet skapat en applikation som med ytterligare lite arbete och finjusteringar skulle vara redo för lansering på Marketplace.

6.2 Framtiden

Vårt dokument 'Riktlinjer för utveckling av en applikation till Android och Windows Phone' är inte komplett till den grad att det inte finns något att tillföra. Vi har skrapat på ytan av all information som finns om dessa plattformar och att fortsätta gå djupare skulle kunna vara ett annat examensarbete eller ett projekt för ett företag. Till exempel skulle en student som redan besitter viss kunskap inom området kunna bygga vidare på detta examensarbete för att fördjupa sig och lära sig mer ännu för att bygga vidare på våra jämförelser.

Denna rapports problemställning är att strukturera en mobilapplikation så den effektivt kan utvecklas till multipla plattformar. Orsaken till att problemet existerar är att applikationer till de olika plattformarna utvecklas i olika programmerinsspråk. Att vi idag har tre stora mobila plattformar som alla bygger

på olika språk känns onödigt och bakåtsträvande. Gemensamma standarder för både mjukvara och hårdvara är framtiden för att öka kompabiliteten mellan enheter. Kompabilitet är något som redan idag efterfrågas, och att man som konsument låser sig till ett visst system bara på grund av en speciell enhet är inte en hållbar lösning.

Som en lösning på problemet med att samma applikation måste göras till alla mobila operativsystem har det kommit så kallade 'cross-platform' ramverk. Tanken med ramverken är att man bara skapar en applikation som ska fungera på alla olika mobila operativsystem och ofta skapas applikationerna med hjälp av webspråk som tex HTML, CSS och javascript. Ett av många problem med cross-plattform ramverk är att det ofta saknas stöd för hårdvaran för specifika enheter som till exempel kamera, gps och accelerometer.

Referenser

[1] General Design Principles, Microsoft, acc 2012-05-02
http://msdn.microsoft.com/en-us/library/hh202906%28v=vs.92%29.aspx

[2] Animations and Motions, Microsoft, acc 2012-05-02
http://msdn.microsoft.com/en-us/library/hh202871%28v=vs.92%29.aspx

[3] Usability Requirements, Microsoft, acc 2012-05-02
http://msdn.microsoft.com/en-us/library/hh202889%28v=vs.92%29.aspx

[4] Essential Graphics, Microsoft, acc 2012-05-02
http://msdn.microsoft.com/en-us/library/hh202884%28v=vs.92%29.aspx

[5] Creative Vision, Android, acc 2012-05-09
http://developer.android.com/design/get-started/creative-vision.html

[6] Design Principles, Android, acc 2012-05-09
http://developer.android.com/design/get-started/principles.html

[7] Devices and Displays, Android, acc 2012-05-09
http://developer.android.com/design/style/devices-displays.html

[8] Supporting Multiple Screens, Android, acc 2012-05-09
http://developer.android.com/guide/practices/screens_support.html

[9] Tilt- Effect, The Code Project, acc 2012-05-09
http://catalog.codeproject.com/Articles/197660/Metro-in-Motion-Part-4-Tilt-Effect

[10] XML i 10 punkter, World Wide Web Consortium (W3C), acc 2012-05-09
http://www.w3c.se/resources/office/translations/XML-in-10-points_sw.html

[11] XML, W3Schools, acc 2012-05-28
http://www.w3schools.com/xml/xml_whatis.asp

[12] XML on android, Android, acc 2012-05-28
http://developer.android.com/guide/topics/ui/declaring-layout.html

[13] Pragmatic Bookshelf, Hello Android third edition, 2010

[14] XAML Overview (WPF), Microsoft, acc 2012-05-10
http://msdn.microsoft.com/en-us/library/ms752059.aspx

[15] XAML Syntax In Detail, Microsoft, acc 2012-05-28
http://msdn.microsoft.com/en-us/library/ms788723.aspx

[16] Difference between XML and XAML, acc 2012-05-28
http://www.differencebetween.net/technology/software-technology/difference-between-xml-and-xaml

[17] Android Architecture, acc 2012-05-28
http://www.therealjoshua.com/2011/11/android-architecture-part-1-intro/

[18] Implementing the MVVM pattern, Microsoft, acc 2012-05-28
http://msdn.microsoft.com/en-us/library/gg405484(v=pandp.40).aspx

[19] Model-View-Controller, Microsoft, acc 2012-05-28
http://msdn.microsoft.com/en-us/library/ff649643.aspx

[20] Execution Model Overview for Windows Phone, Microsoft, acc 2012-05-28
http://msdn.microsoft.com/en-us/library/ff817008(v=vs.92).aspx

[21] Activity, class overview, Android, acc 2012-05-28
http://developer.android.com/reference/android/app/Activity.html

[22] Android Security Overview, Android, acc 2012-05-28

http://source.android.com/tech/security/index.html

[23] Windows Phone 7.5 enterprise security and policy management, Microsoft, acc 2012-05-30
http://blogs.technet.com/b/windows_phone_for_it_pros/archive/2011/10/20/new-windows-phone-7-5-enterprise-security-and-policy-management-paper.aspx