Examensarbete

HÖGSKOLAN I SKÖVDE

KARTLÄGGNING OCH JÄMFÖRELSE AV MULTIPLATTFORMSUTVECKLINGS-VERKTYG FÖR SMARTPHONES

SURVEY AND COMPARISON OF CROSS PLATFORM MOBILE APPLICATION DEVELOPMENT TOOLS FOR SMARTPHONES

Examensarbete inom huvudområdet Datalogi
Grundnivå 30 högskolepoäng
Vårtermin 2015

Emma Jonsson

Handledare: Mikael Berndtsson
Examinator: Henrik Gustavsson

Sammanfattning

Användningen av smartphones ökar med rasande fart och problemet som app-utvecklare har är att det finns flera olika plattformar. För att nå ut till så många användare som möjligt måste alltså en app tillverkas för varje plattform. Detta kräver dels kunskap i utveckling för respektive plattform men också mycket tid och därmed pengar eftersom varje enskild app måste utvecklas i flera versioner. En lösning på detta kan dock vara så kallade multiplattformsutvecklingsverktyg där utvecklaren kan använda samma kodbas för flera olika plattformar. Men håller dessa verktyg måttet?

Denna studie undersöker dels vilka multiplattformsutvecklingsverktyg som finns att tillgå idag samt jämför hur mycket det skiljer i slutresultatet för en likvärdig app tillverkad i tre utav dessa multiplattformsutvecklingsverktyg. Resultatet visar att det skiljer en hel del.

Studien avslutas med en diskussion om resultatet och dess trovärdighet samt ger förslag på hur studien skulle kunna vidareutvecklas.

Nyckelord: multiplattformsutvecklingsverktyg, Android, iOS, smartphones, app

Innehållsförteckning

1 Introduktion

Denna studie tar upp det faktum att den mobila marknaden i form av olika smartphones växer stadigt. Detta innebär i sin tur att för att man som utvecklare ska hänga med så måste man kunna utveckla appar till flera olika plattformar, alla med sina speciella förutsättningar och krav. För att underlätta för utvecklare har därför en relativt ny typ av verktyg tagits fram, så kallade multiplattformsutvecklingsverktyg (i fortsättningen även kallade enbart utvecklingsverktyg), som kan användas för att utveckla appar till flera olika plattformar utan att för den skull behöva göra det individuellt för varje. På detta sätt kan utvecklaren spara in på både tid och pengar för uppdragsgivaren.

Det finns flera studier som gjorts på dessa multiplattformsutvecklingsverktyg men ingen utav dem kombinerar alla de utvecklingsverktyg som idag går att finna och ingen utav dem utför alla de test som dessa studier sammanlagt har undersökt. Detta är därmed det som har inspirerat till denna studie och den frågeställning som denna studie har för avsikt att besvara. Det handlar om att ta reda på vilka multiplattformsutvecklingsverktyg som finns att tillgå idag och hur dessa förhåller sig till varandra ifråga om specifikationer men också ifråga om hur de appar som utvecklingsverktygen skapar förhåller sig gentemot varandra. Hypotesen är att det skiljer mycket mellan de appar som tas fram i respektive utvecklingsverktyg. Detta trots att de borde vara mycket likvärdiga då de ska utgå från samma grund.

För att besvara den fråga som ställts planeras först en litteraturstudie, för att en sammanställning av tillgängliga multiplattformsutvecklingsverktyg ska kunna göras, och ett antal punkter som sammanställningen ska innefatta sätts upp. Detta följs sedan av planerna för att välja ut ett antal utvecklingsverktyg att jobba vidare med och utveckla en prototyp i, för att sedan kunna jämföra utvecklingsverktygen. Ett antal krav sätts även upp både på de utvecklingsverktyg som väljs ut samt den app som sedan tas fram i form av en prototyp. Detta för att resultatet från studien ska bli så representativt och informativt som möjligt. För att till sist kunna avgöra om det skiljer mellan utvecklingsverktygen och i så fall hur mycket så planeras ett antal test som ska utföras på de prototyper som tas fram.

Kartläggningen av tillgängliga multiplattformsutvecklingsverktyg resulterar i att hela 18 stycken utvecklingsverktyg hittas och sammanställs utifrån punkterna *”Specifikationer”*, *”Installation”* och *”Kompilering och testkörning av applikation”*. Därefter väljs tre utav dessa ut utifrån kraven att det ska gå att utveckla appar till Android och iOS (eftersom dessa två är de vanligaste målplattformarna), det ska vara ett representativt utvecklingsverktyg för den mängd som hittats och det ska vara lätt att testköra en app som utvecklas i utvecklingsverktyget för att den begränsade tid som denna studie skall utföras på inte ska riskera att gå åt till att i huvudsak felsöka en app som kraschar.

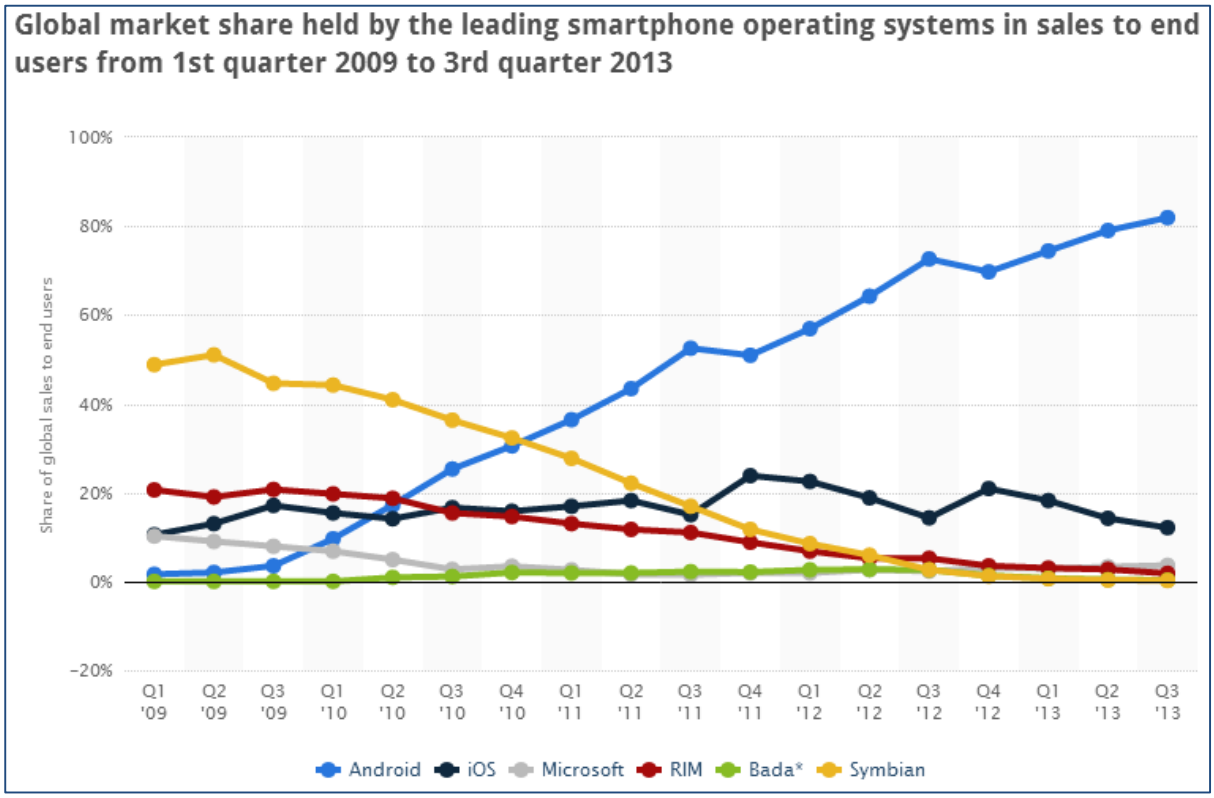
Tre utav de utvecklingsverktyg som hittats väljs ut och därefter tas en prototyp av en app fram i respektive. Appen får tre funktioner utifrån vanliga egenskaper hos appar som ligger på topplistan över populära appar på Google Play. Appen skall kunna visa på en karta var användaren befinner sig, kunna ta ett kort och visa upp det samt kunna köra en funktion och sedan returnera svaret och hur lång tid det tog att komma fram till det.

Till sist så undersöks prototyperna med tre olika test. Dels ett som kontrollerar hur mycket varje prototyp växer när en ny funktion tillförs, dels ett som kontrollerar hur mycket batteri som förbrukas när olika funktioner körs och avslutningsvis ett som kontrollerar hur lång tid det tar för vardera prototyp att utföra en funktion.

2 Bakgrund

Användningen av smartphones ökar med rasande fart och från år 2011 till år 2013 har det nästan fördubblats, från 36 % till hela 65 % av befolkningen, som använder en smartphone i Sverige. Hela två av tre svenskar använder en smartphone för att, åtminstone någon gång ibland, koppla upp sig till internet via mobilen (Findahl, 2013).

Det finns ett antal olika operativsystem som används i dessa smartphones och enligt Statista (2014) är de mest populära Android, iOS, Microsoft, RIM, Bada och Symbian. Man kan dock tydligt se på den graf som Statista (2014) publicerar på sin hemsida (se Figur 1 nedan) att Android är det operativsystem som övervägande står för den största delen av dagens mobilers operativsystem.



Figur 1 *Globala marknadsandelar för olika mobila operativsystem uppdelat på kvartal från första kvartalet 2009 till tredje kvartalet 2013. Källa: (Statista, 2014)*

2.1 Appar

De program som körs på en smartphone kallas för appar, vilket kommer ifrån ordet applikation. Generellt så finns det enligt Hui, Chieng, Ting, Mohamed och Arshad (2013) tre olika typer av appar: native-appar, webb-appar och hybrid-appar.

2.1.1 Native

En native app, eller en plattformsspecifik app som Huy & Thanh (2012) föredrar att kalla det, är en app som är speciellt framtagen för den unika plattform som den ska köras på (Android, iOS, Windows Phone etc.) och kan inte användas på någon annan plattform utan att modifieras. Exempelvis kan appar som tagits fram till en smartphone baserad på Android, inte köras på en Iphone utan dessa fungerar endast på andra enheter baserade på just Android (Huy & Thanh, 2012).

En native/plattformsspecifik app är skriven i ett specifikt programmeringsspråk (se Tabell 1 nedan) och tar hjälp av ett ramverks API (Application Programming Interface) för att komma åt enhetens hårdvara såsom kamera, GPS (Global Positioning System), accelerometer osv. (Huy & Thanh, 2012).

Tabell 1 *Nödvändiga programmeringskunskaper för olika mobila målplattformar enligt Charland & LeRoux (2011).*

Målplattform	*Nödvändiga programmeringskunskaper*
Apple iOS	C, Objective C
Google Android	Java (Harmony flavored, Dalvik VM)
RIM Blackberry	Java (J2ME flavored)
Symbian	C, C++, Python, HTML/CSS/JS
Windows Mobile	.NET
Windows 7 Phone	.NET
HP Palm webOS	HTML/CSS/JS
MeeGo	C, C++, HTML/CSS/JS
Samsung Bada	C++

2.1.2 Webb

En webb-app är en webbrowser-baserad app som laddas ner från webben (Xanthopoulos & Xinogalos, 2013). Den byggs med webbutvecklingsspråken HTML5, CSS3 och JavaScript och den kan designas så att den både ser ut och beter sig ungefär som en native app trots att den alltså befinner sig helt och hållet på nätet och istället körs genom enhetens webb-läsare (Sin, Lawson & Kannoorpatti, 2012).

En av fördelarna med en webb-app jämfört mot en native app är att den inte är plattformsspecifik. Under förutsättning att enheten har en webbläsare som klarar att tolka HTML5 och CSS3 så kan en och samma applikation köras på flera helt skilda typer av enheter. Dessutom har den förmågan att efterlikna en native app genom att placera ut en startikon på hemskärmen vilket uppmuntrar till upprepad användning på ett helt annat sätt än ett vanligt sparat bokmärke i webbläsaren gör (Sin, Lawson & Kannoorpatti, 2012). En annan fördel är att den inte behöver uppdateras i korrekt mening utan det är endast den webbplats som webb-appen bygger på som behöver uppdateras (Xanthopoulos & Xinogalos, 2013).

Några nackdelar med webb-appen är dock att man i nuläget inte kan komma åt enhetens hårdvara (man jobbar dock med att utveckla denna möjlighet i HTML5 genom ett antal API'er) samt att webb-appens innehåll är otillgängligt om det inte finns tillgång till ett nätverk (Xanthopoulos & Xinogalos, 2013).

2.1.3 Hybrid

En hybrid-app försöker kombinera fördelarna hos webb-appen med fördelarna hos en native app. Den byggs vanligtvis med HTML5 och JavaScript och någon djupare kunskap för målplattformen krävs inte (Xanthopoulos & Xinogalos, 2013).

Hybrida appar bäddar in HMTL5 i ett tunt native skal (UIWebView i iOS och WebView i Android) och precis som en webb-app så körs fortfarande appen av en webbläsare på enheten men precis som med en native app så går appen att installera på enheten och den kan dessutom komma åt viss hårdvara på enheten tack vare ett speciellt API som medföljer (Xanthopoulos & Xinogalos, 2013).

Hybrid-appen kan alltså implementeras med flera olika tekniker med hjälp av flera olika utvecklingsverktyg, men för att kunna efterlikna en native apps utseende och känsla på bästa sätt så krävs det att man använder ett utvecklingsbibliotek som exempelvis jQuery (Xanthopoulos & Xinogalos, 2013).

2.2 Multiplattformsutvecklingsverktyg

För att som app-utvecklare kunna nå ut till så många användare som möjligt behöver man kunna nå ut på så många plattformar som möjligt. För att spara både tid och pengar går därför fler och fler utvecklare över till så kallade multiplattformsutvecklingsverktyg. Dessa utvecklingsverktyg gör det möjligt att implementera appen endast en gång med hjälp av webbutvecklingsspråk såsom HTML5, CSS3 och JavaScript och sedan kompilerar utvecklingsverktyget koden till en körbar app på respektive målplattform (Dalmasso, Datta, Bonnet & Nikaein, 2013).

Fördelarna med dessa utvecklingsverktyg är att man kan spara både tid och pengar. Detta eftersom man inte behöver lägga dessa resurser på att lära sig alla de språk som respektive plattform kräver. Det räcker med att behärska de vanligaste webbutvecklingsspråken. Man behöver inte heller utveckla en app till respektive målplattform utan det räcker med att utveckla en gemensam och sedan ser alltså utvecklingsverktyget till att den går att köra på respektive målplattform som man är intresserad av. Eftersom detta går så mycket fortare än vanlig utveckling så kan man också lansera en ny applikation på väldigt kort tid. Nackdelen är dock att alla dessa fördelar sker på bekostnad av användarupplevelsen (Dalmasso m.fl., 2013).

3 Problemformulering

Som utvecklare av mobila appar finns det idag många plattformar att ta hänsyn till om man som företag vill nå ut med sin app till så många användare som möjligt. Som synes i Tabell 1 (se kapitel 2.1.1) har vi både Android, iOS, Symbian och Windows Mobile, för att bara nämna några.

För att inte slösa med företagets resurser i form av både tid och pengar så kan man, som nämns i kapitel 2.2, välja att ta hjälp av multiplattformsutvecklingsverktyg. Men hur ska man då som utvecklare veta vilket utav alla dessa utvecklingsverktyg som erbjuds på marknaden, som lämpar sig bäst för den app som man planerar att bygga?

De multiplattformsutvecklingsverktyg som finns idag erbjuder alla sin unika väg för att lösa problemet med att nå ut till så många mobila plattformar som möjligt med så lite ansträngning som möjligt. Det kan skilja både i utvecklingsmiljö och i slutprodukt, men även i support och eventuella kostnader samt till vilka plattformar som utvecklingsverktyget klarar av att leverera.

3.1 Tidigare forskning

Det finns flera studier som gjorts på det omtalade problemet tidigare och i Tabell 2 nedan sammanställs de som hittats utifrån när studien är gjord, vilka utvecklingsverktyg som på ett eller annat sätt nämns i studien, samt vilka egenskaper som har undersökts om ett experiment har utförts i studien.

Tabell 2 *En sammanställning av tidigare studier inom området.*

Referens och årtal	Utvecklingsverktyg som nämns	Utvecklingsverktyg som testats	Egenskaper som undersökts hos app
Andersson & Andreasson (2013)	- MoSync - RhoMobile Suite - PhoneGap	- PhoneGap	- Exekveringstid - Storlek
Dalmasso m.fl (2013)	- PhoneGap - Sencha Touch 2.0 - PhoneGap + Sencha Touch 2.0 - PhoneGap + jQuery Mobile - Application Craft - Appcelerator Titanium Studio 2.0	- PhoneGap - PhoneGap + jQuery Mobile - PhoneGap + Sencha Touch 2.0 - Titanium	- Batteriförbrukning - CPU-användning - Minnesanvändning
Fransson (2014)	- MoSync - jQuery Mobile - PhoneGap - Marmalade	- MoSync	N/A

Hui m.fl. (2013)	- PhoneGap - DragonRad - Rhodes - XLMVM	N/A	N/A
Palmieri, Singh & Cicchetti (2012)	- Rhodes - PhoneGap - DragonRad - MoSync	N/A	N/A
Pehrson (2011)	- Appcelerator Titanium Mobile - Corona SDK - MoSync - PhoneGap - RhoMobile Rhodes - AirPlay SDK (nu kallad Marmalade)	- PhoneGap - Corona SDK	- Exekveringstid - Minnesanvänding
Tillborg, Persson, Bentlöv (2012)	- Appcelerator Titanium Mobile - PhoneGap - jQuery Mobile	- PhoneGap + jQuery Mobile	- Användartest i form av upplevelser

Sammanfattningsvis kan man säga att de utvecklingsverktyg som funnits med i respektive studie har varit ganska blandade. Det finns ingen studie som lyckats samla alla de utvecklingsverktyg som sammanlagt listas i Tabell 2 ovan. De egenskaper som testats hos de framställda prototyp-apparna är också skilda åt. En del utvecklingsverktyg har inte blivit testade i skarpt läge överhuvudtaget.

3.2 Frågeställning och syfte

Det primära syftet med studien är att kartlägga och sammanställa de multiplattforms-utvecklingsverktyg som erbjuds idag och göra en jämförelse dem emellan. Detta för att underlätta i valet av utvecklingsverktyg för de företag eller enskilda utvecklare som är intresserade av att börja utveckla appar för flera olika plattformar på smartphones.

Den fråga som skall besvaras är:

> *Vilka multiplattformsutvecklingsverktyg finns att tillgå idag och hur mycket skiljer det ifråga om egenskaper mellan de appar som tas fram i respektive, om grunden till appen är densamma?*

Hypotesen är att det skiljer ganska mycket mellan de olika apparna ifråga om deras egenskaper. Detta trots att de borde vara mycket likvärdiga då de utgått ifrån samma grund. Anledningen till detta påstående är att eftersom utvecklarna för respektive utvecklingsverktyg troligtvis har löst tolkningen av det användaren vill skapa samt kompileringen från utvecklingskod till en native app på olika sätt, så borde det rimligtvis inte resultera i ett identiskt slutresultat.

3.3 Förväntat resultat

Det förväntade resultatet är att få fram en sammanställning av de multiplattforms-utvecklingsverktyg som finns att tillgå idag, som är så informativ och fullständig att det skulle vara av stort intresse för de företag och utvecklare som är intresserade av ämnet att ta del av den.

4 Metod

För att kunna besvara frågan *”Vilka multiplattformsutvecklingsverktyg finns att tillgå idag och hur mycket skiljer det ifråga om egenskaper mellan de appar som tas fram i respektive, om grunden till appen är densamma?”* så kommer det angreppssätt som Wohlin m.fl. (2000) förespråkar att användas. Det innefattar

- Definiera
- Planera
- Utför
- Analysera och tolka
- Presentera

Studien har i kapitel 3 definierats och i detta kapitels underkapitel kommer den att planeras. Därefter kommer den att utföras genom att den delas upp i två delar där två olika metoder kommer att användas.

I första delen av studien kommer en kvantitativ litteraturstudie att utföras för att kartlägga de multiplattformsutvecklingsverktyg som går att finna och deras respektive egenskaper. Detta kommer sedan att sammanställas. Nackdelen med en litteraturstudie är att för att korrekta slutsatser ska kunna dras så måste den vara helt komplett. Men risken finns att alla utvecklingsverktyg som existerar inte kommer att hittas vilket i så fall skulle göra litteraturstudien ofullständig. Detta skulle i sin tur innebära att den därmed inte presenterade alla de utvecklingsverktyg som finns att välja mellan, vilket ju ändå är målet. För att förhindra detta ska en så noggrann studie som möjligt utföras.

I andra delen av studien kommer en experimentell studie att genomföras genom att ett antal utvecklingsverktyg kommer att väljas ut utifrån den sammanställning som gjorts i första delen av studien. Därefter kommer dessa att undersökas ytterligare med hjälp av en prototyp av en app. Nackdelen med en experimentell studie är att risken alltid finns för att man ska göra en felaktig analys av resultatet utifrån exempelvis tillfällig data som inte egentligen ger en rättvis bild åt det man testar. För att förhindra att detta sker ska alla test upprepas så många gånger att man tydligt kan se ett mönster som bör vara det vanligtvis mest förekommande tillståndet.

Studiens resultat kommer avslutningsvis att analyseras och tolkas och därefter presenteras.

Förutom en litteraturstudie och ett experiment skulle man även kunna utföra en fallstudie men det kommer inte att vara aktuellt i denna studie eftersom det inte är hur man använder utvecklingsverktygen som ska utvärderas, utan istället hur bra utvecklingsverktygen presterar gentemot varandra.

4.1 Kartläggning

En kartläggning av de multiplattformsutvecklingsverktyg som finns att tillgå idag kommer att ske. De utvecklingsverktyg som räknas upp i Tabell 2 (se kapitel 3.1) kommer att undersökas djupare, samt en undersökning om utifall det har dykt upp ytterligare utvecklingsverktyg på marknaden kommer att utföras. Resultatet kommer sedan att sammanställas.

4.1.1 Tillvägagångssätt

Kartläggningen kommer att ske genom efterforskningar på Internet med bl.a. hjälp av Google men även med hjälp av databaser såsom IEEE Xplore Digital Library och ACM Digital Library

för att försöka sammanställa en så komplett bild som möjligt av de multiplattforms-utvecklingsverktyg som finns att tillgå idag.

Specifikationer för de utvecklingsverktyg som hittats kommer sedan att samlas in från respektive utvecklingsverktygs officiella hemsida.

4.1.2 Egenskaper/sammanställning

De uppgifter som samlats in kommer att sammanställas enligt samma egenskaper som Pehrson (2011) gjorde i sin studie, med undantag för punkten Utvecklingshjälp som innefattar dokumentation, community och exempelapplikationer. Detta beror på att målet med denna studie inte är att avgöra om utvecklingsverktyget har ett aktivt community eller om det kan klassas som ett community överhuvudtaget samt vad som kan klassas som dokumentation. De egenskaper som därmed kommer sammanställas är därför följande:

- **Specifikationer**
  - Distributionsplattformar
  - Programmeringsspråk
  - Licens
- **Installation**
  - Nödvändig programvara att installera
  - Omfattning
- **Kompilering och testkörning av applikation**
  - Typ av emulator
  - Omfattning

4.2 Val av utvecklingsverktyg

För att få fördjupad kunskap om vad dessa utvecklingsverktyg kan åstadkomma ska några utav dem väljas ut för att sedan testas i skarpt läge då en prototyp av en applikation ska skapas och sedan testas. De egenskaper som behöver finnas hos de utvecklingsverktyg som ska testas är följande:

- **Android & iOS**
  Utvecklingsverktyget ska kunna skapa appar till Android och iOS eftersom dessa är de vanligast förekommande plattformarna idag enligt Hui m.fl. (2013).
- **Representativa**
  De utvecklingsverktyg som väljs ut ska vara representativa för de utvecklingsverktyg som hittats.
- **Testkörning**
  Det ska vara enkelt att testköra appen, antingen via en emulator eller via en fysisk enhet. Detta för att det underlättar utvecklingen då det går att testköra en ny funktion rad för rad istället för att upptäcka att den nya funktionen gör att appen kraschar och då behöva hitta tillbaks till och felsöka all kod som tillkommit i och med den nya funktionen.

4.3 Appen

En prototyp av en app ska skapas med hjälp av de utvecklingsverktyg som valts ut. För att veta vilka egenskaper som är intressanta att jämföra i de olika versionerna (versioner syftar här på att de skapats av olika utvecklingsverktyg) av appen, och med andra ord vilka egenskaper som

är intressanta att ha med i appen, skall en undersökning av egenskaper hos populära appar göras.

Då tillgång till en Mac-dator inte finns och detta krävs för att kunna utveckla till iOS enligt Tillborg m.fl. (2012) så kommer appen att enbart utvecklas till Android.

4.4 Implementationsmetod

Den utvecklingsmetod som kommer användas är bottom-up. Detta innebär att utvecklingen kommer att ske med start på appens mindre delar var för sig och när alla delar fungerar kommer de att slås samman till en större del. Detta tillvägagångssätt är valt för att möjliggöra att så många funktioner som möjligt ska hinna testas under den tid som studien pågår.

4.5 Utvärdering

Tester av appens egenskaper och funktioner kommer kontinuerligt att utföras under implementationen, allt eftersom de blir färdiga. De tester som kommer utföras är inspirerade av de tester som utförts och sammanställts i Tabell 2 (se kapitel 3.1). Syftet med testerna är att få en djupare förståelse för vilka skillnader som kan uppstå när man utvecklar en app med samma förutsättningar och egenskaper men med olika multiplattformsutvecklingsverktyg. Alla tester kommer att sammanställas i form av grafer eller tabeller för att påvisa eventuella mönster.

4.5.1 Storlek

Storleken på den ”tomma” appen från vart och ett av utvecklingsverktygen kommer att undersökas för att se hur mycket det skiljer i grundläget. Detta kommer sedan att återupprepas allt eftersom appen växer med nya egenskaper och funktioner. Resultatet kommer sedan att sammanställas i en graf för att man ska få en förståelse för hur mycket extra som varje utvecklingsverktyg inkluderar när en ny funktion eller egenskap tillförs. Detta test är inspirerat av Andersson & Andreasson (2013).

4.5.2 Minnesanvändning

Minnesanvändningen skall testas på samma sätt som Dalmasso m.fl. (2013) gjorde genom att använda sig av verktyget DDMS. DDMS står för Dalvik Debug Monitor Server och är ett debugverktyg som kan användas vid utveckling av Android (Android Developers, 2014).

4.5.3 CPU-användning

Att testa CPU-användningen är även det inspirerat av Dalmasso m.fl. (2013). Testerna kommer att utföras med hjälp av ett verktyg som heter Dumpsys. Dumpsys är ett androidverktyg om körs direkt på enheten och dumpar intressant information om enhetens tillstånd (Stackoverflow, 2013a).

4.5.4 Batteriförbrukning

Appens påverkan på batteritiden kommer att mätas genom den mycket populära appen Power Tutor (Google play, 2014g) som bl.a. rapporterar batteriförbrukning för individuella appar (Zhang m.fl., 2010). Detta test är inspirerat av Dalmasso m.fl. (2013).

4.5.5 Exekveringstid

Exekveringstiden kommer att mätas genom att en tidtagning kommer att ske på exempelvis när appen byter från en vy till en annan. Samma händelse kommer naturligtvis att då

undersökas på samtliga versioner av appen. På samma sätt som Pehrson (2011) gjorde så kommer den stopwatch-teknik som Hoccer (2010) beskriver att användas. Detta innebär att tiden före och efter en händelse sparas i logcat och sedan tas mellantiden ut (Pehrson, 2011).

4.5.6 Riktlinjer

Eftersom det med största sannolikhet inte kommer att finnas tid för att intervjua användare för att undersöka hur de upplevde appen, vilket Tillborg m.fl. (2012) gjorde, så kommer inte apparnas grafiska gränssnitt att jämföras trots att detta vore en intressant studie i allra högsta grad. De viktiga är alltså att apparna får samma funktioner och inte att de har ett identiskt grafiskt utseende.

4.6 Etik

Då studien inte kommer innefatta några människor eftersom inga intervjuer eller någon fallstudie kommer att ske så kommer det inte finnas några större etiska problem med den. För att ytterligare stärka studien forskningsetiskt så kommer all producerad programkod samt information om utvecklingsmiljön att bifogas för att underlätta för en eventuell replikering av studien.

Resultatet som studien skulle kunna få på samhället är omöjlig att förutspå men inom den närmsta tiden kommer den med all sannolikhet inte att kunna påverka i någon större utsträckning.

Att använda sig av en benchmark-app för att undersöka egenskaper hos en annan app är alltid riskfyllt. Som Cai, Nerurkar och Wu (1998) skriver så finns det tillfällen då utvecklare för program skrivit in funktioner som undersöker om programmet benchmarkas och i så fall ser till att prestandan tillfälligt förbättras för att det ska se bättre ut i benchmark-resultaten. Om nu så är fallet att utvecklaren för den testenhet som används har gjort detta, bör rimligtvis denna förändring ske oavsett vilken app som undersöks och därför bör resultatet för respektive prototyp påverkas lika mycket. Om däremot utvecklaren för ett eller flera utav de utvecklingsverktyg som används i denna studie har gjort detta, finns risken för att resultaten skulle kunna bli missvisande och detta bör naturligtvis tas i beaktande då studiens resultat tolkas.

5 Genomförande

Detta kapitel tar upp hur kartläggningen av de multiplattformsutvecklingsverktyg som finns att tillgå har utförts, samt vilka och på vilka grunder ett antal utav dem har valts ut att arbeta vidare med. Den prototyp som valt att utvecklas i de utvalda utvecklingsverktygen specificeras samt implementationen av den presenteras. Kapitlet avslutas med en pilotstudie av de appar som tagits fram.

5.1 Kartläggning

Kartläggningen av multiplattformsutvecklingsverktyg har pågått under perioden 17 mars 2014 – 14 april 2014 och har skett genom omfattande efterforskningar på Internet med bl.a. hjälp av Google men även med hjälp av databaser såsom IEEE Digital Library och ACM Digital Library. Sökord som använts har bl.a. *varit mobile cross platform development, android development, ios development, hybrid app development* m.fl. Även sökning på namnen för de utvecklingsverktyg som listas i Tabell 2 (se kapitel 3.1) har utförts vilket har lett till kunskapen att utvecklingsverktyget XMLVM inte längre finns att ladda ner och därför har uteslutits ur denna studie. Flertalet utvecklingsverktyg har även hittats med hjälp av en sammanställning som Wikipedia (2014) presenterar. Utifrån detta har sedan specifikationer för respektive utvecklingsverktyg hämtats in från respektive officiella hemsida och sammanställts i detta kapitel.

5.1.1 Appcelerator Titanium Mobile

Specifikationer

Distributionsplattformar:	Nativeappar för Android, iOS, Windows Phone, Blackberry och Tizen samt mobila webbappar
Programmeringsspråk:	JavaScript, HTML, CSS, PHP, Ruby, Pydev
Licens:	Gratis (Apache License, Version 2.0)

Installation

För att kunna utveckla i Appcelerator behöver utvecklaren installera Titanium Studio samt de tredjeparts-SDK som är av intresse, alternativt räcker det med att integrera Titaniums kommandotolk i det IDE som utvecklaren föredrar att jobba med.

Kompilering och testkörning av applikation

Under utveckling av applikationen kan utvecklaren testköra den genom exempelvis emulatorn för Android eller simulatorn för iOS som medföljer respektive SDK.

5.1.2 Appception IDE

Specifikationer

Distributionsplattformar:	Hybridappar för Android och iOS
Programmeringsspråk:	HTML, CSS, JavaScript
Licens:	Helt gratis då den för tillfället är i Beta-läge.

Installation

Ingen installation är nödvändig under förutsättning att utvecklaren har en giltig webbläsare (Chrome 10+, Safari 5+ eller Firefox 3.6+) installerad eftersom all utveckling sker på Appceptions moln via en webbsida.

Kompilering och testkörning av applikation

När utvecklaren är nöjd med sin applikation kan denne förhandsgranska appen direkt på sidan och utvecklar denne för Android går det även bra att ladda ner ABug från Google Play direkt till en mobil enhet och sedan testköra samt installera applikationen där direkt som alternativ till att emulera de olika funktionerna på webbsidan. Det går även bra att ladda ner och testköra applikationen genom att läsa in den QR-kod som skapas för applikationen.

5.1.3 appery.io

Specifikationer

Distributionsplattformar:	Hybridappar för Android, iOS och Windows Phone
Programmeringsspråk:	Drag and Drop, WYSIWYG (HTML5, jQuery Mobile, JavaScript, CSS)
Licens:	Gratis att bygga en app med upp till tre sidor. Planeras utveckling i större skala finns flera betalalternativ att välja på.

Installation

Ingen installation är nödvändig eftersom all utveckling sker på appery.io's moln via deras webbsida.

Kompilering och testkörning av applikation

Eftersom utvecklaren bygger en mobil HTML5-app så kan denne testköra den direkt i webbläsaren på antigen datorn eller mobilen alternativt så kan den testköras direkt på mobilen genom applikationen Appery.io Mobile App Tester för att få tillgång till enhetsspecifika funktioner.

5.1.4 Application Craft

Specifikationer

Distributionsplattformar:	Android, iOS, Windows Mobile, BlackBerry, Symbian och Bada
Programmeringsspråk:	Drag and Drop, HTML5, CSS3, JavaScript
Licens:	45 dagars gratis prövotid med begränsningen att det inte går att ladda ner eller exportera den färdigbyggda applikationen. Fortsatt möjlighet att använda verktyget genom att utvecklaren betalar en mindre månadskostnad.

Installation

Ingen installation är nödvändig eftersom all utveckling sker på Application Craft's moln via deras webbsida.

Kompilering och testkörning av applikation

Möjlighet att testköra och debugga samt packa den färdiga applikationen finns direkt i Application Craft.

5.1.5 AppMkr

Specifikationer

Distributionsplattformar:	Nativeappar för Android och iOS samt mobila webbappar
Programmeringsspråk:	Välj egenskaper ur en lista alternativt koda HTML

Licens:	Gratis att skapa och publicera Android samt Webb-appar innehållandes reklam, alternativt finns flera betalalternativ att välja på.

Installation

Ingen installation är nödvändig eftersom all utveckling sker direkt på AppMkr's webbsida.

Kompilering och testkörning av applikation

Under pågående utveckling kan utvecklaren se direkt i en emulator hur applikationen växer fram alternativt så kan utvecklaren skicka en live-länk och öppna den direkt i en mobil enhet för att se hur utvecklingen fortskrider. När utvecklaren är nöjd kan denne direkt via AppMkr's webbsida packa och publicera appen.

5.1.6 Codename One

Specifikationer

Distributionsplattformar:	Nativeappar för Android, iOS, Windows Phone, RIM och J2ME
Programmeringsspråk:	Java
Licens:	Gratis (GPL med Classpath Exception) men det finns ytterligare tjänster att köpa till.

Installation

Codename One kommer inte med ett egen SDK utan utvecklaren kan antingen välja att installera Codename One som ett plugin till Eclipse, NetBeans eller Idea.

Kompilering och testkörning av applikation

När det är dags att bygga sin applikation bör utvecklaren förslagsvis skicka upp den till Codename One's molntjänst där applikationen packas om till en körbar applikation för respektive målplattform. Det går dock även att packa filen lokalt men då krävs ytterligare installation och Codename One erbjuder ingen guide för detta.

5.1.7 Corona SDK Starter

Specifikationer

Distributionsplattformar:	Spel till Android, iOS
Programmeringsspråk:	Lua
Licens:	Gratis med begränsad funktionalitet och möjlighet till att utöka detta genom flera betalalternativ.

Installation

För att börja arbeta med Corona SDK måste utvecklaren ladda ner och installera Corona SDK. Utvecklaren måste även installera Xcode om denne sitter på en Mac-dator eller Java JDK 6, samt valfritt textediteringsprogram.

Kompilering och testkörning av applikation

Corona SDK kommer med en egen simulator där utvecklaren kan välja mellan olika mobila enheter att testköra sin applikation i. Om utvecklaren sedan vill publicera applikationen måste denne skicka upp den till Coronas server som packar om den och skickar tillbaka en körbar programfil för respektive valt målplattform.

5.1.8 DragonRAD Designer

Specifikationer

Distributionsplattformar:	Nativeappar för Android, iOS, Windows Mobile och BlackBerry
Programmeringsspråk:	Drag and Drop, Lua
Licens:	30 dagars gratis prövotid och därefter krävs det att utvecklaren betalar för tjänsten.

Installation

För att utvecklaren ska kunna börja arbeta krävs det att denne laddar ner, installerar och aktiverar DragonRAD Designer

Kompilering och testkörning av applikation

Utvecklaren kan testköra applikationen på flera olika sätt. Antingen via en simulator eller direkt på en mobil enhet som då ska vara inkopplad via USB. För att publicera applikationen måste utvecklaren logga in på DragonRAD Host.

5.1.9 Marmalade SDK

Specifikationer

Distributionsplattformar:	Spel till Android, iOS, Windows Phone 8, BlackBerry, Tizen, Symbian, WebOS och Bada
Programmeringsspråk:	C++, Lua, Objective C, HTML5, CSS3, JavaScript
Licens:	30 dagars gratis prövotid, därefter finns flera betalalternativ.

Installation

Utvecklaren kan välja att ladda ner Marmalade integrerat i antingen Visual Studio (Windows) eller Apple Xcode (Mac) och därmed kunna arbeta i en redan bekant miljö.

Kompilering och testkörning av applikation

När utvecklaren vill testköra sina applikationer kan denne enkelt göra det via den medföljande Marmalade Simulatorn.

5.1.10 MobiOne Studio

Specifikationer

Distributionsplattformar:	Hybridappar för Android och iOS samt HTML5-appar
Programmeringsspråk:	Drag and Drop, HTML5, CSS3, JavaScript
Licens:	15 dagars gratis prövotid och därefter finns en livstids licens att köpa för $99.95

Installation

Endast MobiOne Studio måste laddas ner och installeras och sedan kan utvecklaren direkt börja arbeta.

Kompilering och testkörning av applikation

När utvecklaren vill testa sin applikation kan denne använda sig av den medföljande App Generator för att omvandla designen till en HTML5/CSS3/JavaScript-applikation som sedan går att köra i en medföljande Mobile Web Simulator. Om utvecklaren sedan vill gå vidare

använder denne sig av MobiOne's molntjänst App Center Builder för att antingen publicera en Mobile Web Application eller skapa en hybridapplikation som går att köra direkt på en mobil enhet. Molntjänsten är tyvärr begränsad till en appstorlek på 50MB.

5.1.11 MoSync

Specifikationer

Distributionsplattformar:	Hybridappar för Android, iOS, Windows Mobile, Windows Phone, BlackBerry, Symbian, MeeGo, J2ME och Moblin.
Programmeringsspråk:	HTML, CSS, JavaScript, C och C++
Licens:	Gratis för privat bruk och även för kommersiellt bruk under förutsättning att applikationen då är open source. Alternativt finns möjligheten att betala för utökade rättigheter.

Installation

Utvecklaren kan välja att antingen installera MoSync Mobile SDK och ev. ytterligare native emulatorer, alternativt MoSync Reload. Om valet faller på MoSync Reload så måste utvecklaren dels installera en UI Client och en Server på datorn samt en Reload Client på den mobila enheten och sedan samköra dessa via ett gemensamt Wi-Fi.

Kompilering och testkörning av applikation

För att kompilera en applikation kan utvecklaren antingen välja att göra det via MoRE som är den egna kompilatorn som medföljer MoSync Mobile SDK alternativt köra via en native emulator som installerats. Eller så kan utvecklaren testköra direkt på en mobil enhet under förutsättning att denne valt att installera MoSync Reload.

5.1.12 Motorola RhoMobile Suite

Specifikationer

Distributionsplattformar:	Nativeappar för Android, iOS, Windows Mobile och Windows Phone 8.
Programmeringsspråk:	Ruby, HTML, CSS och JavaScript
Licens:	Gratis (MIT License)

Installation

Om utvecklaren väljer att kunna kompilera applikationen lokalt måste denne installera respektive SDK för den plattform denne tänkt att utveckla till. Utvecklaren kan även välja att installera RhoStudio och utveckla applikationen där i, alternativt installera ett RhoMobile plugin i Visual Studio eller utveckla i valfritt verktyg.

Om utvecklaren vill kan denne istället välja att bygga applikationen i valfritt verktyg och sedan kompilera den i RhoHub, se nedan.

Kompilering och testkörning av applikation

Applikationen kan antingen kompileras och testköras lokalt direkt via RhoStudio eller kompileras via kommandotolken, eller så kan den kompileras i RhoHub som är RhoMobile's molntjänst.

5.1.13 PhoneGap

Specifikationer

Distributionsplattformar:	Android, iOS, Windows Phone, webOS, BlackBerry, Symbian, Tizen och Bada
Programmeringsspråk:	HTML, CSS och JavaScript
Licens:	Gratis (Apache License, Version 2.0) att utveckla och kompilera publika applikationer men vill utvecklaren kompilera mer än en privat applikation via PhoneGap Build krävs en prenumeration på tjänsten.

Installation

För att kunna utveckla en applikation med hjälp av PhoneGap finns de flera vägar att gå.

Antingen kan utvecklaren välja att först installera NodeJS, därefter via kommandotolken installera PhoneGap, och till sist installera de SDK till respektive plattformar som man valt att utveckla för. Först därefter kan utvecklaren kompilera en applikation med hjälp av PhoneGap som alltså kompilerar HTML, CSS och JavaScript till en körbar applikation.

Alternativt kan utvecklaren använda sig av PhoneGap Build som är PhoneGap's molntjänst dit utvecklaren skickar in sina HTML-, CSS- och JavaScript-filer och automatiskt får dem kompilerade utan att behöva installera något alls.

Eller så kan utvecklaren välja att integrera PhoneGap direkt i exempelvis Eclipse IDE eller Adobe Dreamweaver 6+.

Kompilering och testkörning av applikation

När utvecklaren är nöjd med den producerade appen i form av HTML-, CSS- och JavaScript-kod som skrivs i valfritt program, så kan denne antingen välja att kompilera appen lokalt eller via PhoneGap's molntjänst och sedan installera appen på en mobil enhet alternativt köra den i en emulator för tänkt målplattform.

5.1.14 RareWire App Creation Studio

Specifikationer

Distributionsplattformar:	Nativeappar för Android och iOS
Programmeringsspråk:	WIRE (RareWire's eget utvecklingsspråk), JavaScript
Licens:	Gratis att utveckla med men publikation av färdig app kostar. Det går då att välja mellan att betala en engångskostnad per publikation av app, alternativt betala för en prenumeration.

Installation

Ingen installation krävs eftersom utvecklaren arbetar direkt mot RareWire's molntjänst.

Kompilering och testkörning av applikation

När utvecklaren vill testköra sin app är det bara att köra appen FuseBox producerad av RareWire på den mobila enhet som ska testas med.

5.1.15 Sencha Architect med Sencha Touch

Specifikationer

Distributionsplattformar:	HTML5-appar som passar alla mobila enheter som stödjer HTML5
Programmeringsspråk:	Drag and Drop, JavaScript
Licens:	30 dagars gratis prövotid, därefter måste ett Development kit köpas.

Installation

Allt som behövs är att installera Sencha Architect och sedan kan utvecklaren direkt börja arbeta.

Kompilering och testkörning av applikation

Eftersom utvecklaren bygger en mobil HTML5-app så kan denne testköra den direkt i webbläsaren på antigen datorn eller mobilen.

5.1.16 SmartFace

Specifikationer

Distributionsplattformar:	Android och iOS
Programmeringsspråk:	Drag and Drop, JavaScript
Licens:	Gratis att använda med vissa begränsningar så som att den producerade appen antingen måste ha en splash-screen med SmartFace logga eller göra reklam för SmartFace inuti appen. Möjlighet att välja mellan flera betalalternativ för utökade funktioner.

Installation

För att kunna utveckla i SmartFace måste utvecklaren installera SmartFace App Studio samt Java SDK. Vidare om utvecklaren vill utveckla för Android så måste denne installera Android SDK, alternativt iTunes och Microsoft .NET Framework 4 för iOS.

Kompilering och testkörning av applikation

Under utveckling av applikationen kan utvecklaren testköra den med en virtuell emulator men utvecklaren rekommenderas att testköra den direkt på en mobil enhet antingen via USB (Android) eller Wi-Fi (Android och iOS).

5.1.17 ViziApps

Specifikationer

Distributionsplattformar:	Hybridappar för Android och iOS samt mobila webbappar i HTML5
Programmeringsspråk:	Drag and Drop
Licens:	30 dagars gratis prövotid men då begränsad till att de appar som skapas inte går att publicera och de går endast att testa på 1 Android-enhet och 1 iOS-enhet. Om utveckling i större skala eller under längre tid planeras finns flera betalalternativ att välja på.

Installation

Ingen installation är nödvändig under förutsättning att utvecklaren har en giltig webbläsare (Chrome 12+, Internet Explorer 9+, Firefox 10+ eller Opera 11+) installerad eftersom all utveckling sker på ViziApps moln via en webbsida.

Kompilering och testkörning av applikation

Utvecklaren kan emulera sin hybridapp direkt online eller direkt på en mobil enhet via ViziApps egen app.

5.1.18 Xamarin Studio

Specifikationer

Distributionsplattformar:	Nativeappar för Android och iOS
Programmeringsspråk:	C#, Java, Objective-C, HTML och JavaScript
Licens:	Gratis för utveckling av mindre applikationer med möjlighet att välja mellan flera betalalternativ för utökade funktioner.

Installation

För att börja arbeta med Xamarin Studio krävs det att utvecklaren laddar ner och installerar programmet. På köpet installeras de SDK's som krävs.

Kompilering och testkörning av applikation

Utvecklaren kan testköra sin applikation direkt via Xamarin Studio tack vare att nödvändiga emulatorer automatiskt har installerats och finns tillgängliga.

5.2 Val av utvecklingsverktyg

Utifrån de multiplattformsutvecklingsverktyg som tagits fram i kapitel 5.1 och utifrån de punkter som räknas upp i kapitel 4.2 så har ett antal utvecklingsverktyg valts ut att arbeta vidare med. I brist på tid har antalet begränsats till följande:

- appery.io
- SmartFace
- Xamarin

appery.io representerar de utvecklingsverktyg som inte behöver installeras utan som utvecklas med direkt på webben och som utvecklar i HTML5, CSS3 och JavaScript samt använder sig av drag-n-drop. Dessutom representerar det de utvecklingsverktyg som kan utveckla till fler målplattformar än Android och iOS och som utvecklar hybridappar.

Smartface representerar de utvecklingsverktyg som behöver installeras och som är gratis att utveckla i. Dessutom representerar det de utvecklingsverktyg som använder sig av drag-n-drop och som utvecklar nativeappar för Android och iOS.

Xamarin representerar de utvecklingsverktyg som utvecklar i andra språk än de vanligaste webbutvecklingsspråken (HTML5/CSS3/JavaScript) samt måste installeras och som utvecklar nativeappar för Android och iOS.

5.3 Appen

Valet av egenskaper för den prototyp av app som skall tas fram med de olika utvecklingsverktygen under denna studie, har baserats dels på de egenskaper som finns hos de mest populära apparna på Google Play idag (utifrån den lista som Google Play publicerar), dels på de utvärderingsmetoder som listas i kapitel 4.5. De egenskaper som planeras att finnas hos prototypen är följande:

- Karta
- RSS-flöde
- Kamera
- Något som kan undersökas med Stopwatch-teknik

5.4 Implementation

I detta kapitel beskrivs hur implementationen av prototyp-appen för respektive utvecklingsverktyg fortskridit.

Den utvecklingsmiljö som använts är följande:

En HP laptop med följande specifikationer

Produktnamn:	dv6-6001eo
Processor:	3 GHz AMD Phenom II Dual-Core N660
Processor cache:	2 MB L2 cache
Minne:	4 GB DDR3
Grafikkort:	AMD Radeon HD 6650M (1 GB DDR3)
Operativsystem:	Windows 7 Professional N

En Android mobil med följande specifikationer

Produktnamn:	Samsung Galaxy S4+
Modell-ID:	I9506
Processor:	2,3 GHz Qualcomm Snapdragon 800 MSM8974
Minne:	2 GB
Operativsystem:	Android 4.3 (Jelly Bean)

Innan arbetet med att implementera olika funktioner i appen påbörjades så kompilerades den befintliga koden till en körbar apk-fil för att det sedan skulle vara möjligt att gå tillbaka och jämföra storleken på denna fil i utgångsläget för de olika utvecklingsverktygen. När detta var gjort påbörjades implementationen. Mellan varje ny funktion som lagts till i appen har denna process återupprepats.

5.4.1 appery.io

Valet av utvecklingsverktyg att påbörja implementationen med föll på appery.io. Det var ett grafiskt verktyg av typen drag-n-drop som kördes direkt via appery.io's molntjänst på deras hemsida (version 12.1). Det krävde ytterst lite kodskrivande av användaren för att sammanställa en mobil applikation, och det var därmed väldigt smidigt att arbeta i. Se Bild 1 på nästa sida.

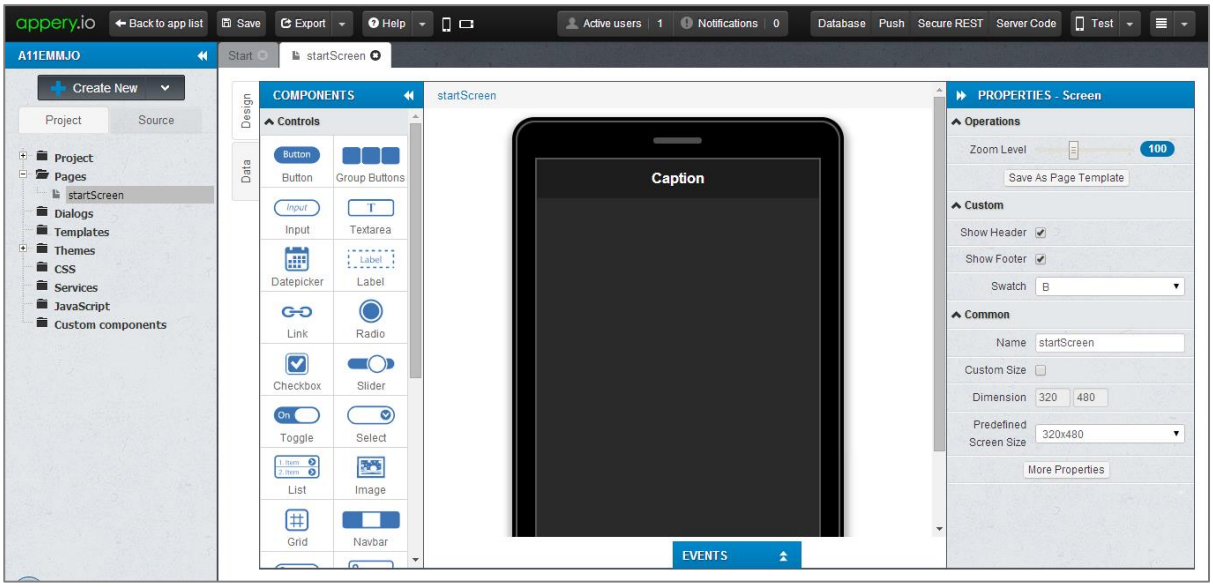


Bild 1 *Det grafiska upplägget av utvecklingsverktyget Appery.io*

Kartan

Den första funktionen som lades till i appen var att användaren skulle kunna se på en karta var denne befann sig med hjälp av den mobila enhetens inbyggda GPS. Denna funktion var inspirerad av appar som Endomondo där användaren dels kan se var denne befinner sig på en karta, dels hur denne förflyttar sig och dels en sammanställning av bland annat hur långt denne har förflyttat sig och på hur lång tid m.m. (Google play, 2014a). För att utveckla denna funktion användes en guide som fanns tillgänglig på appery.io DOCS (2014a).

Det som krävdes för att lägga in funktionen var följande steg:

1. På startsidan lades en ny knapp in. Denna knapp fick sedan funktionen att länka till en ny sida, kartsidan. Allt detta åstadkoms med drag-n-drop och genom att klicka i färdiga val i drop-down-listor. På Bild 2 nedan så syns i övre delen hur funktionen skapas genom att välja ett alternativ i tre olika drop-down-listor och sedan göra olika val för Details. Det färdiga resultatet syns på den nedre raden.

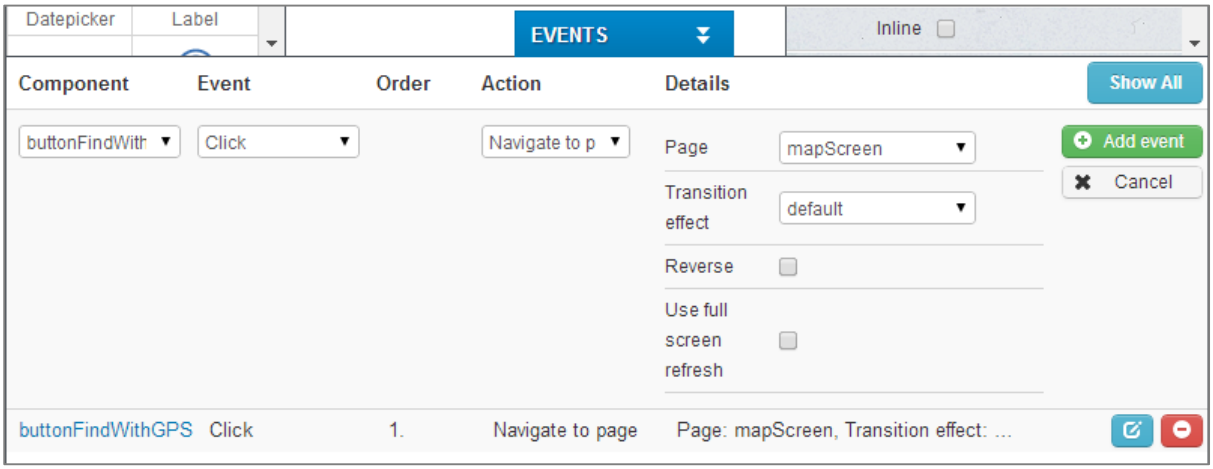


Bild 2 *Exempel på hur en enkel knappfunktion skapas i Appery.io*

2. En ny sida skapades (kartsidan) och sedan lades en Google-karta in med drag-n-drop samt en tillbaka-knapp som alltså länkar tillbaka till startsidan enligt samma princip som på Bild 2 ovan.

3. En geolocationservice lades till och sedan modifierades egenskaperna genom drag-n-drop enligt Bild 3 nedan.

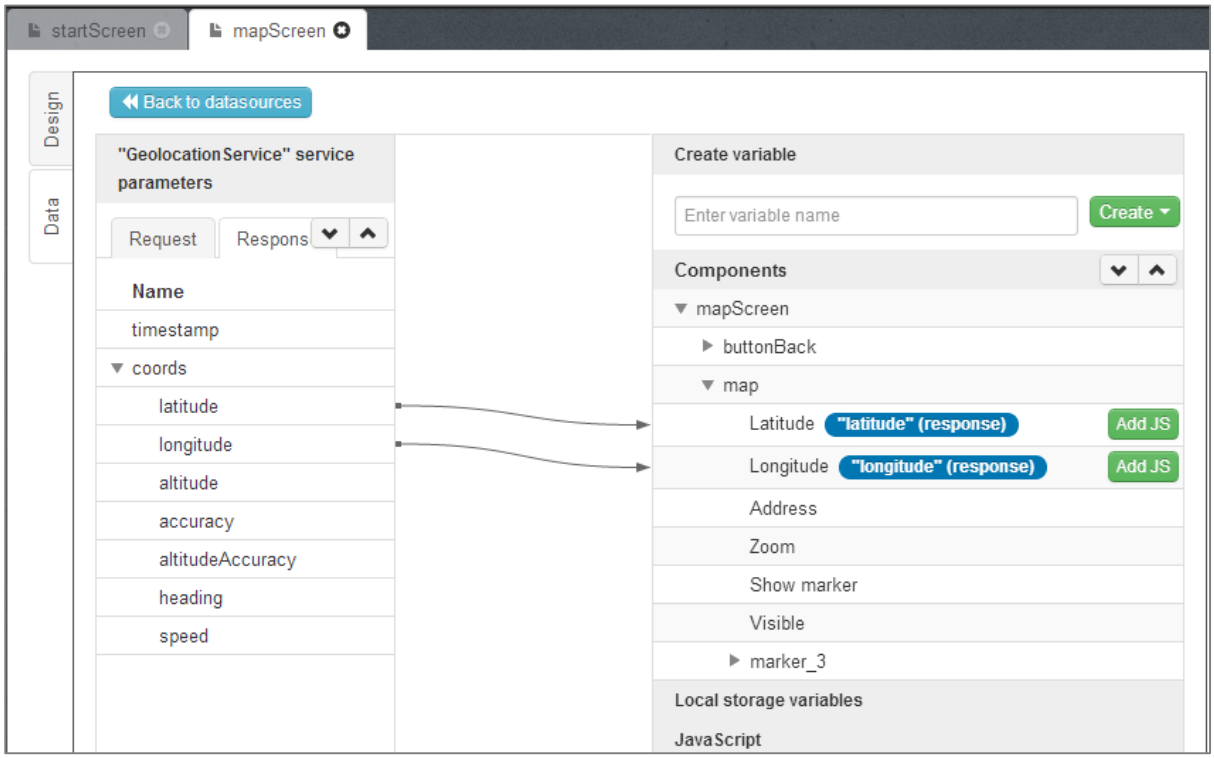


Bild 3 *Modifiering av mappning för appens geolocationservice i Appery.io*

4. Därefter lades en liten kodsnutt JavaScript in (se Bild 4 nedan) som styr så att kartan uppdateras med användarens position.

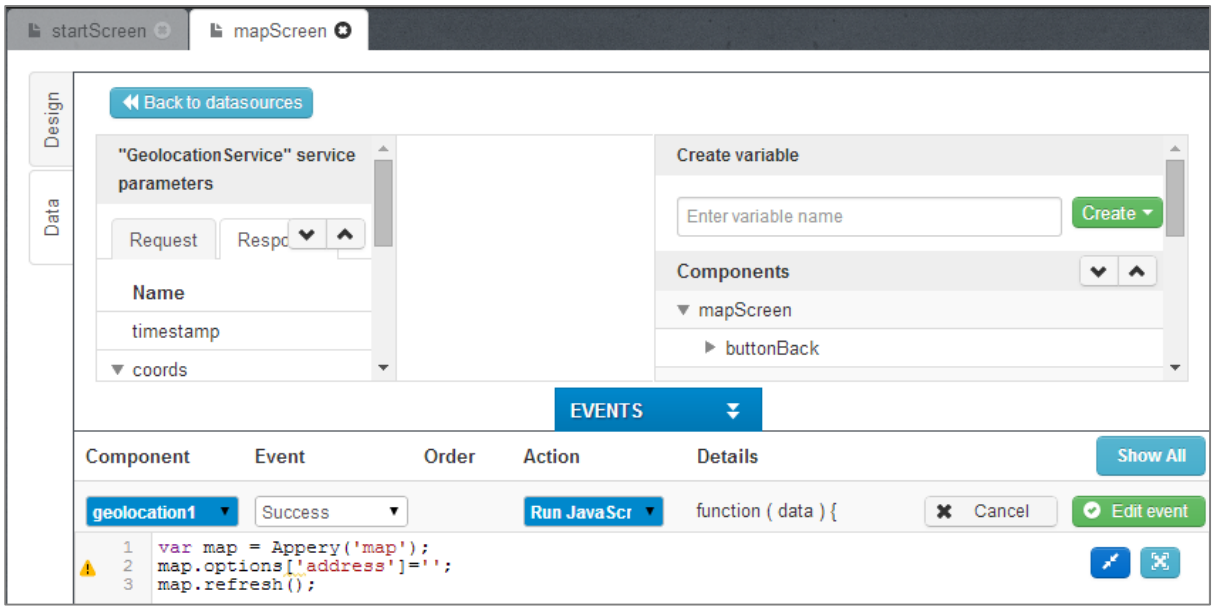


Bild 4 *Den JavaScript-kod som uppdaterar kartan med användarens position*

5. Till sist lades en funktion till för kartsidan som gör så att den skapade geolocationservicen körs när sidan laddas (se Bild 5 nedan).

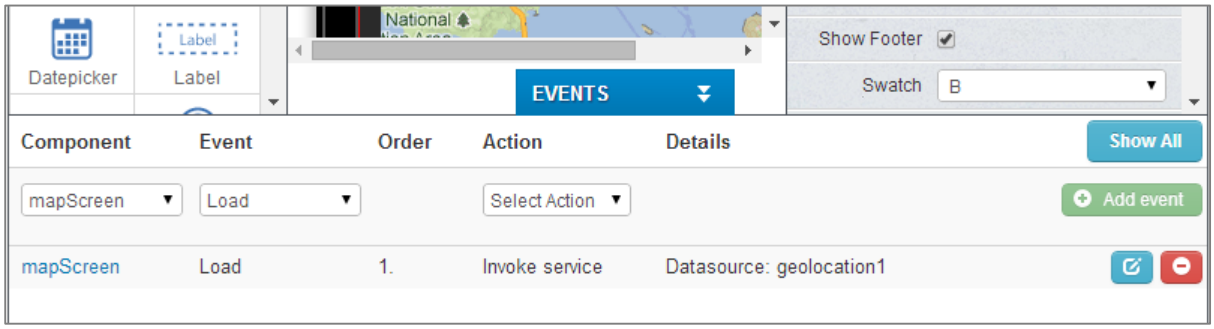


Bild 5 *Funktionen som kör JavaScript-koden (se Bild 4) och uppdaterar kartan när mapScreen laddas*

Kameran

Den andra funktionen som lades in var en kamerafunktion där användaren kan ta ett foto med den mobila enhetens kamera och sedan direkt se resultatet i appen. Denna funktion inspirerades av bland annat appar som Instagram där användaren kan ta en bild och därefter modifiera och publicera bilden i ett flöde på nätet (Google play, 2014b). Funktionen åstadkoms med hjälp av en guide på appery.io DOCS (2014b) enligt följande steg:

1. Ytterligare en knapp samt ett bildobjekt lades till på startsidan med drag-n-drop precis som tidigare.
2. En cameraservice lades till och egenskaperna modifierades enligt Bild 6 nedan.

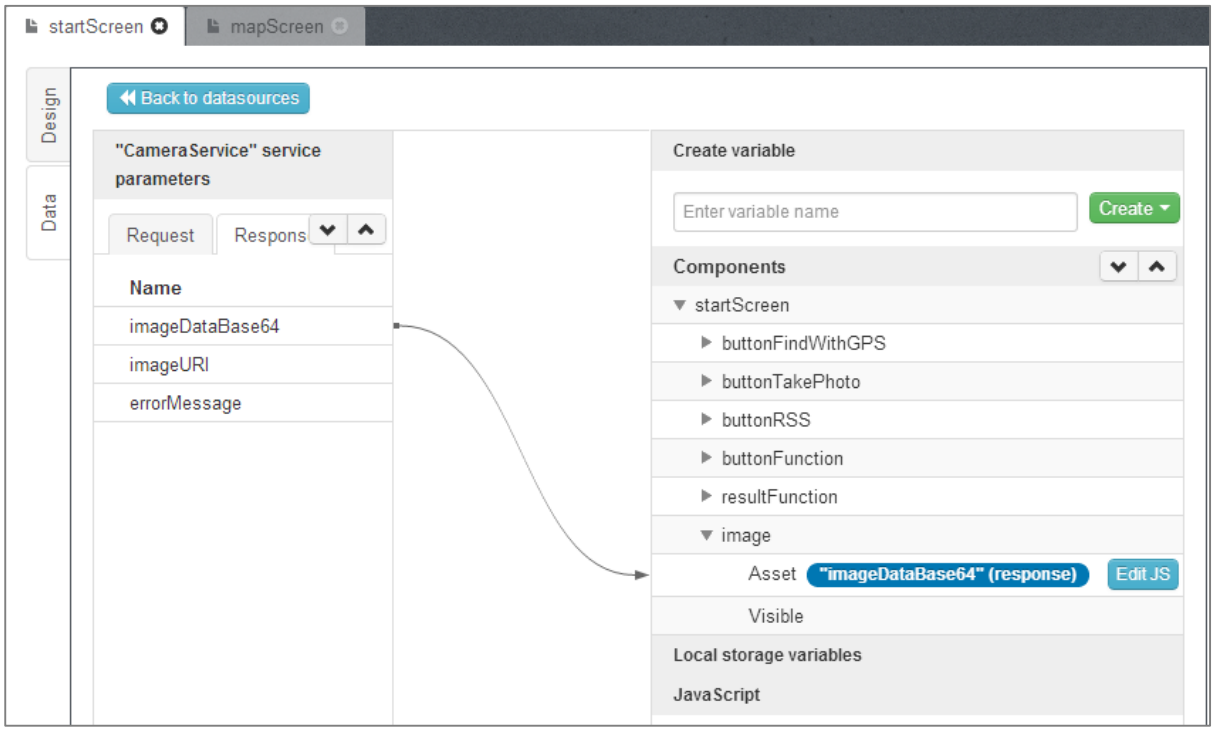


Bild 6 *Modifiering av mappning för appens cameraservice i Appery.io*

3. Därefter lades en liten kodsnutt JavaScript in (via den ljusblå knappen ”Edit JS” på Bild 6) som styr så att bilden visas i rätt proportioner och fyller ut hela appens bredd. Koden var följande:

```
Apperyio('image').css('width', '100%');
Apperyio('image').show();
```

4. Till sist lades två funktioner till enligt samma princip som Bild 2. En funktion som styr så att den skapade cameraservice körs när användaren trycker på den nya knappen, och en funktion som kör nedanstående kod när startsidan laddas. Denna kod ser till att bildobjektet inte visas förrän användaren tagit en bild:

```
Apperyio('image').hide();
```

RSS-flödet

Ett RSS-flöde inspirerat av appar som Facebook (Google play, 2014c), Twitter (Google play, 2014d) m.m. lades även till i appen enligt samma principer som ovan med hjälp av en guide på appery.io DOCS (2014c). Tyvärr gick denna funktion inte att återskapa i Smartface (se 5.4.2) utan att abonnera på en betalningsplan, så därför plockades denna funktion bort igen för att kunna utföra en så rättvis jämförelse av slutresultaten som möjligt.

Primtalsfunktionen

Den sista funktionen som lades till var en funktion där den mobila enheten gör en prestandakrävande primtals-beräkning och sedan presenterar ett svar. Denna funktion omges av en stop-watch-funktion som tar tiden på hur lång tid beräkningen tar att utföra och presenterar sedan svaret. Metoden att implementera en stop-watch-funktion inspirerades av Pehrson (2011). Funktionen åstadkoms med hjälp av tre foruminlägg (CODING TIP, 2013; Stackoverflow, 2012; Stackoverflow, 2013b) enligt följande:

1. Ytterligare en knapp samt en label lades till på startsidan.
2. Den kod som styr vad som händer när startsidan laddas (se punkt 4 för Kameran ovan) modifierades så att labeln inte syns förrän användaren valt att köra primtals-funktionen genom att följande kod lades till:

```
Apperyio('resultFunction').hide();
```

3. För den nya knappen lades den kod som återfinns i Appendix A in enligt samma princip som Bild 2. Detta innebär alltså att när användaren trycker på knappen utförs en beräkning och svaret presenteras i labeln.

Slutresultatet

Det slutliga resultatet i appen blev enligt Bild 7 nedan. Från vä till hö synes startläget (startsidan), kartsidan som visar kartan och användarens position, startsidan när ett foto tagits samt startsidan när primtalsfunktionen har körts.

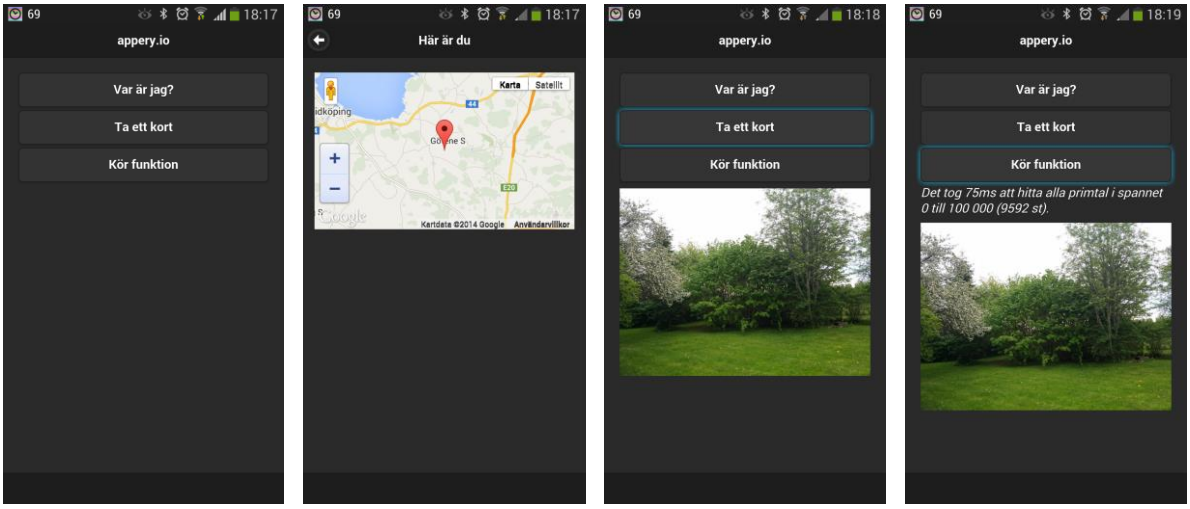


Bild 7 *Slutresultatet för prototyp-appen av Appery.io*

5.4.2 Smartface

Nästa utvecklingsverktyg att arbeta med föll på Smartface. Även detta var ett grafiskt verktyg av typen drag-n-drop men det krävde lite mer av användaren ifråga om att skriva kod. För att kunna arbeta med Smartface krävdes det att Smartface's egen programvara laddades ner och installerades (Smartface App Studio 4.2.0) och på Bild 8 nedan kan man se hur det såg ut vid uppstart av ett nytt projekt.

I Smartface utvecklades en egen version av samma prototyp-app som tagits fram med appery.io.

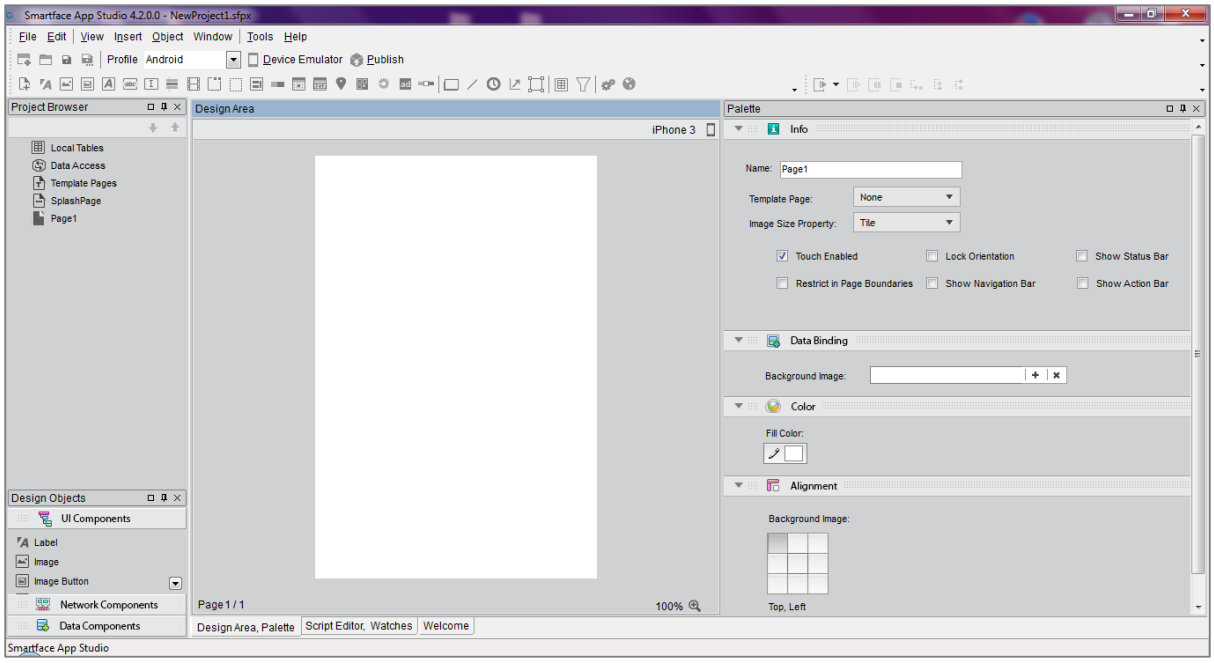


Bild 8 *Det grafiska upplägget av utvecklingsverktyget Smartface*

Kartan

För att lägga till kartfunktionen krävdes följande steg:

1. På startsidan placerades en container ut och i den en label som fick utgöra header samt en knapp för att kunna växla till kartsidan. Detta utfördes precis som i appery.io med drag-n-drop. Alla eventuella mindre modifieringar av objektens egenskaper gjordes i den högra menyn (se Bild 8 på föregående sida).
2. En ny sida lades till och på den placerades förutom en container och en tillbakaknapp, även en mapView ut.
3. Följande kod lades in i script-editorn (se menyn i nederkant på Bild 8 på föregående sida samt Bild 9 nedan) för startsidan för att knappen skulle länka till kartsidan:

```
function startScreen_buttonFindWithGPS_OnPressed(e){
   Pages.mapScreen.show();
}
```

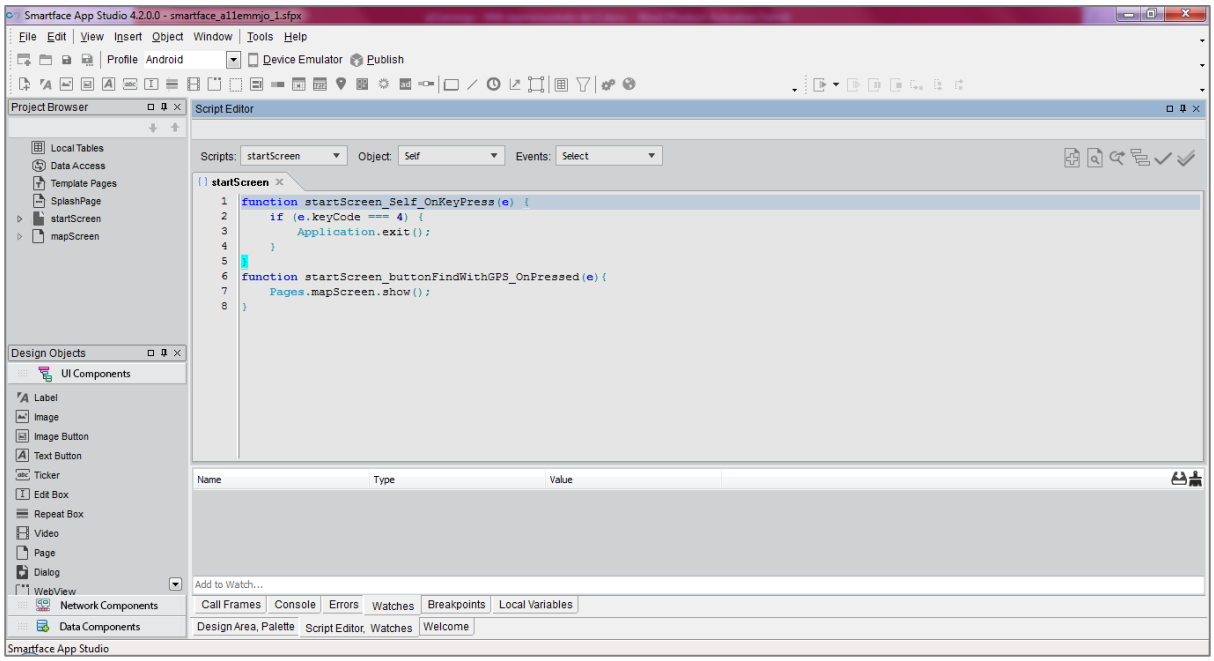


Bild 9 *Script Editorn i Smartface*

4. Följande kod lades sedan in i script-editorn för kartsidan enligt dokumentation på Smartface (Smartface, 2014a; Smartface, 2014b). Detta för att bakåtknappen, både den fysiska och grafiska, skulle länka tillbaka till startsidan samt att kartan skulle uppdateras och visa användarens position när kartsidan laddades:

```
function mapScreen_buttonBack_OnPressed(e){
   Pages.back();
}
function mapScreen_Self_OnKeyPress(e){
   if (e.keyCode === 4) {
       Pages.back();
   }
}
function mapScreen_Self_OnShow(e){
   Pages.mapScreen.containerMap.map.showUserLocation = true;
}
```

Kameran

För att lägga till kamerafunktionen krävdes följande steg:

1. På startsidan placerades ytterligare en knapp ut för att kunna starta kameran, samt ett image-objekt som visar det tagna fotot.
2. Därefter lades kod in i script-editorn för startsidan (se Appendix B), enligt dokumentation på Smartface (2014c), som såg till att kameraknappen startade den mobila enhetens kamera och gjorde det möjligt för användaren att ta ett foto samt för detta foto att sedan presenteras i appen.

Primtalsfunktionen

För att lägga till primtalsfunktionen krävdes följande steg:

1. På startsidan placerades ytterligare en knapp ut för att kunna starta primtalsfunktionen, samt en label där primtalsfunktionens resultat presenteras.
2. Därefter lades en modifierad version på källkoden för den primtalsfunktion som användes i appery.io (se Appendix A) in i script-editorn för startsidan (se Appendix C).

Slutresultatet

Det slutliga resultatet i appen blev enligt Bild 10 nedan. Från vä till hö synes startläget (startsidan), kartsidan som visar kartan och användarens position, startsidan när ett foto tagits samt startsidan när primtalsfunktionen har körts.

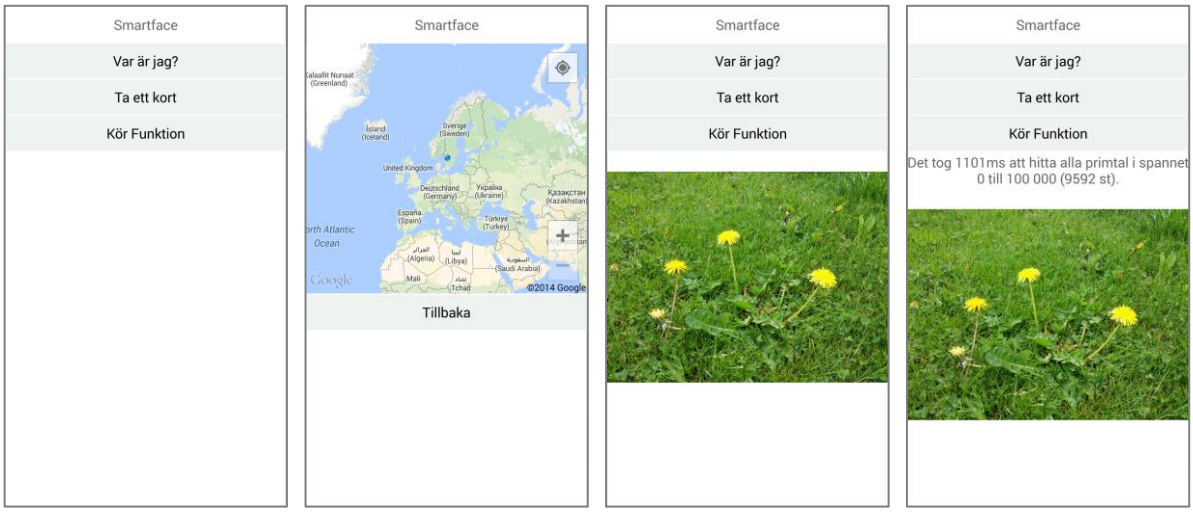


Bild 10 *Slutresultatet för prototyp-appen av Smartface*

5.4.3 Xamarin

Nästa utvecklingsverktyg att arbeta med föll på Xamarin. Detta var ett verktyg där man i första hand skrev kod i C#, men detta kombinerades även med ett grafiskt drag-n-drop för appens utseende. För att kunna arbeta med Xamarin krävdes det att Xamarin's egen programvara laddades ner och installerades (Xamarin Studio 5.8.3) och på Bild 11 på nästa sida kan man se hur detta verktyg såg ut vid uppstart av ett nytt projekt.

I Xamarin utvecklades en egen version utav samma prototyp-app som tagits fram med appery.io och Smartface.

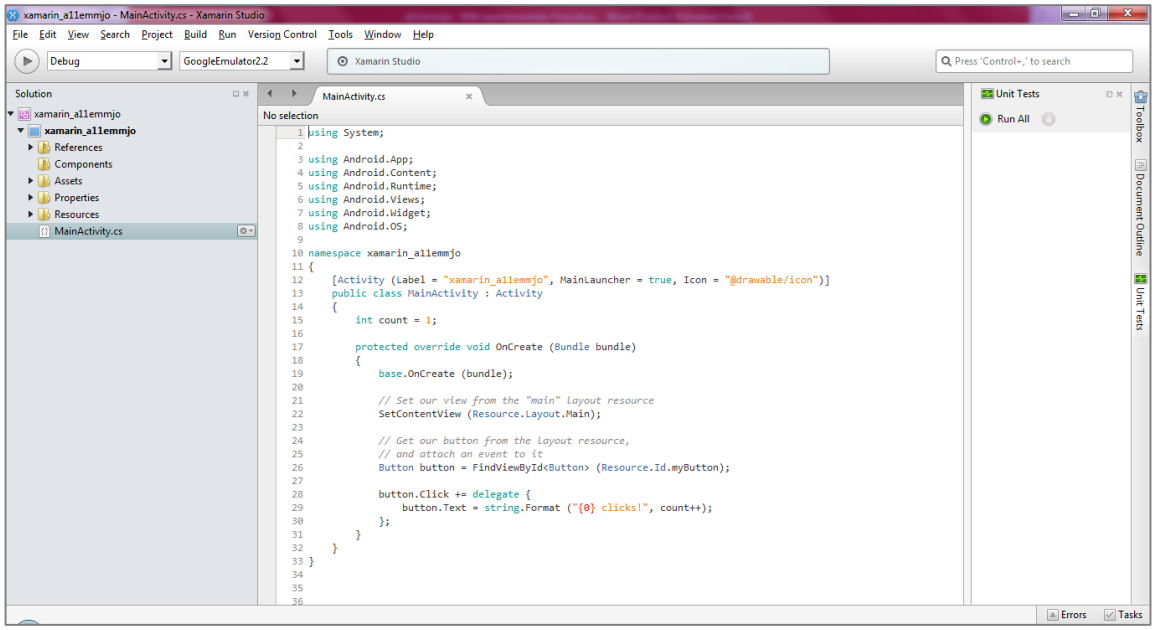


Bild 11 *Det grafiska upplägget av utvecklingsverktyget Xamarin*

Kartan

För att lägga till kartfunktionen krävdes följande steg:

1. För att kunna implementera en karta krävdes det först att tredjepartbiblioteket Google Play Service inkluderades. Detta gjordes enligt den kart-guide som Xamarin tillhandahöll (Xamarin, 2015a).
2. Därefter specificerades de permissions som krävdes för att kartan skulle fungera i dokumentet AndroidManifest.xml enligt guiden (Appendix D).
3. När ovanstående förberedelser var avklarade lades en knapp till på startsidan med hjälp av drag-n-drop. Knappens funktion skulle vara att växla över till kartsidan. Knappens egenskaper modifierades i den högra menyn under Properties (se Bild 12). Samtidigt som förändringar gjordes i verktyget med drag-n-drop, lades dessa till som kod i dokumentet Main.axml (Appendix E) och gick att läsa och modifiera via knappen Source (se nederkant på Bild 12 nedan).

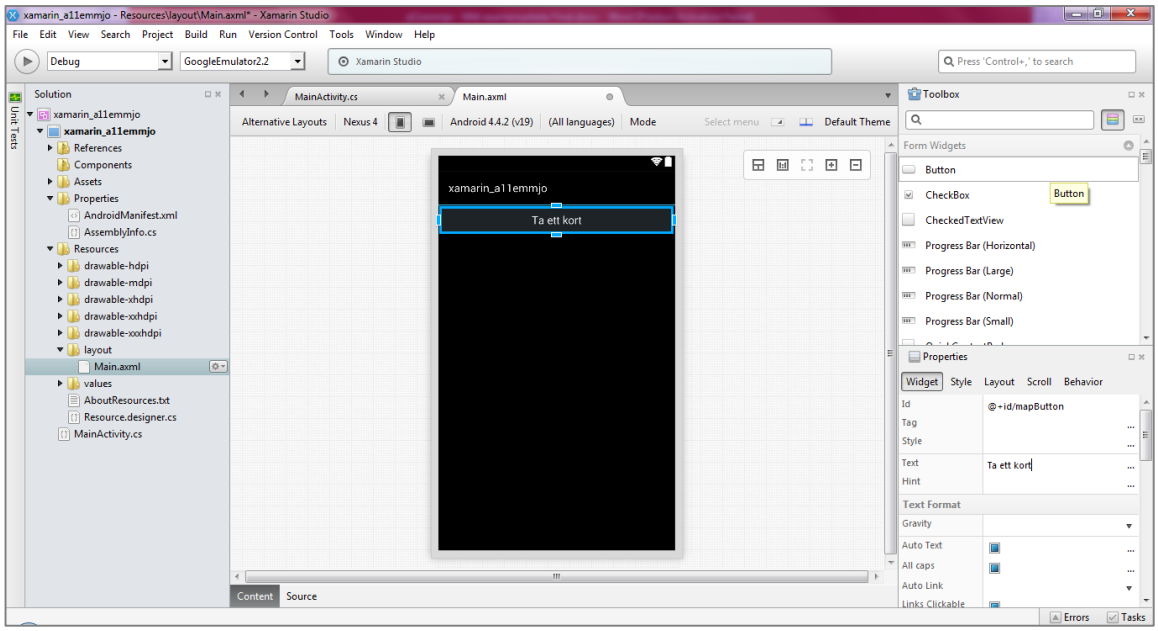


Bild 12 *En knapp läggs till med drag-n-drop i Xamarin*

4. Funktionen som gör att knappen på startsidan öppnar kartsidan lades till i metoden OnCreate() i MainActivity.cs (se Appendix F).
5. Därefter skapades en ny layout (Map.axml) och en frameLayout som skulle innehålla kartan samt en tillbaka-knapp lades till. (se Appendix G).
6. En ny activity (MapActivity.cs) skapades och funktionen som gör att knappen på kartsidan länkar tillbaka till startsidan lades till i metoden OnCreate() (se Appendix H).
7. För att kunna visa kartan i frameLayouten i Map.axml när sidan laddas så gjordes en sammankoppling mellan frameLayouten och ett GoogleMap-objekt via SupportMapFragment i MapActivity.cs enligt guiden.
8. Enligt ytterligare en guide (Xamarin, 2015b) implementerades Fused Location Provider i MapActivity.cs för att användarens position skulle markeras på skärmen när kartan visades.
9. Slutligen deklarerades knapparnas text i filen Strings.xml (se Appendix I) och tilldelades sedan via deras respektive properties.

Kameran

För att lägga till kamerafunktionen krävdes följande steg:

1. Enligt en kamera-guide (Xamarin, 2015c) så lades ytterligare ett permission till i AndroidManifest.xml (se Appendix D).
2. Därefter lades enligt guiden den statiska klassen App till i MainActivity.cs (se Appendix F).
3. På startsidan (Main.axml) lades ytterligare en knapp samt en imageview till med hjälp av drag-n-drop (se Appendix E).
4. MainActivity.cs modifierades enligt guide så att när användaren klickar på knappen startas enhetens kamera, och när ett foto tagits och sparats presenteras detta i imageview'n.
5. För att appen skall kunna hantera fotot utan att krascha enligt guiden, måste fotot krympas. Därför lades enligt guiden klassen BitmapHelpers till i projektet (se Appendix J).
6. Slutligen lades funktionen OnResume() till i MainActivity.cs. När enhetens kamera startas så pausas appen och för att appen sedan skall uppdatera vyn och visa det tagna fotot i imageview'n måste detta utföras i metoden OnResume() som alltså körs när appen återupptas.

Primtalsfunktionen

1. På startsidan (Main.axml) placerades ytterligare en knapp ut samt en textview som skulle presentera resultatet av primtalsfunktionen (se Appendix E).
2. Därefter lades en modifierad version av den primtalsfunktion som användes i appery.io (se Appendix A) och Smartface (se Appendix C) in i MainActivity.cs (se Appendix F).

Slutresultatet

Det slutliga resultatet i appen blev enligt Bild 13 nedan. Från vä till hö synes startläget (startsidan), kartsidan som visar kartan och användarens position, startsidan när ett foto tagits samt startsidan när primtalsfunktionen har körts.

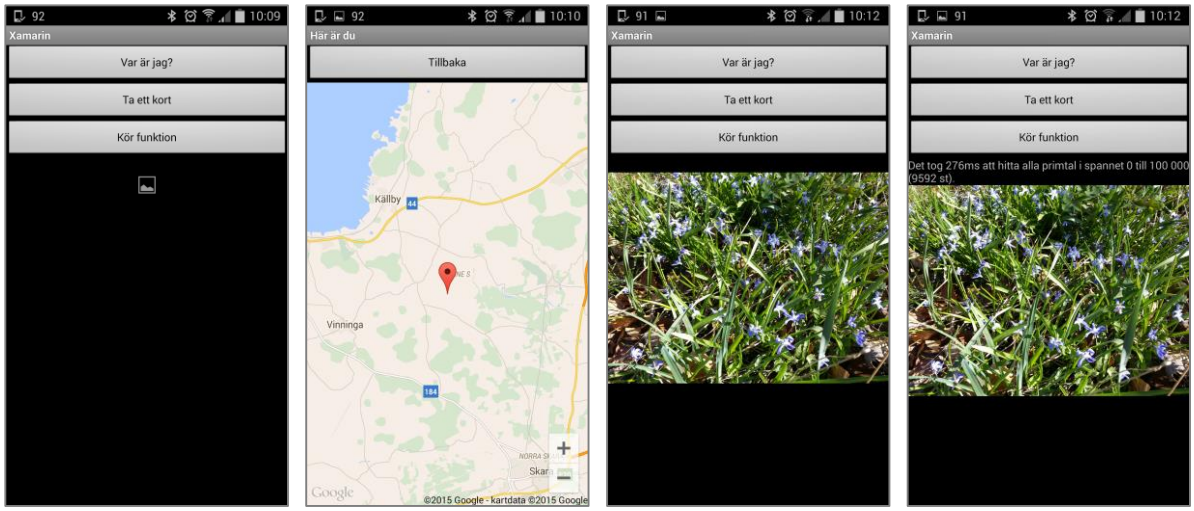


Bild 13 *Slutresultat för prototyp-appen av Xamarin*

5.5 Pilotstudie

För att säkerställa att det som tagits fram är utvärderingsbart har en pilotstudie av de punkter som listas i kapitel 4.5's underkapitel utförts. I detta kapitel listas resultatet för varje test. Varje test har enbart körts en gång för vardera prototyp för att påvisa att det är möjligt, och resultaten som presenteras i detta kapitel är därmed inte ett genomsnittligt värde utan ett enskilt värde som uppstått vid det enskilda testtillfället.

5.5.1 Storlek

Då varje version av prototyp-appen först har sparats på den lokala datorn som arbetet utförts på, innan den sedan har överförts till testenheten, så är det lätt att se storleken på installations-filen via filhanteraren. Då det skiljer ytterst lite mellan versionerna har valet på att notera resultatet i bytes valts. Storleken för respektive prototyp är i grundutförandet, dvs. innan några funktioner lagts till, följande:

appery.io	2 478 757 byte	(2,36 MB)
Smartface	11 765 287 byte	(11,2 MB)
Xamarin	3 416 104 byte	(3,25 MB)

Storleken på samma prototyp som ovan när den väl installerats på testenheten är enligt nedan. Det värde som använts är det som finns att hitta under testenhetens programhanterare och är endast värdet för Program och inkluderar alltså inte någon ytterligare data.

appery.io	3,42	MB
Smartface	34,04	MB
Xamarin	6,33	MB

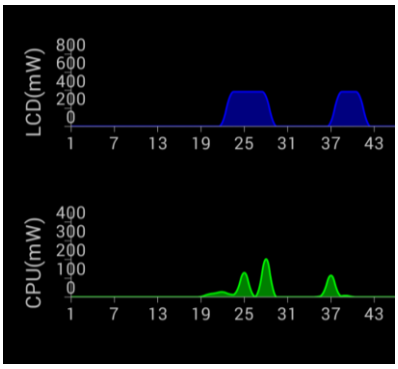
5.5.2 Exekveringstid

För att testa exekveringstiden har den primtalsfunktion som implementerats i prototypen (se delen om Primtalsfunktionen i kap 5.4.1) använts. För att undvika att eventuella andra tjänster på testenheten stör testet har testenheten startats om i flygplansläge och därefter har appen startats och primtalsfunktionen direkt körts. Testet har utförts i den sista och kompletta versionen av appen och resultatet för vardera prototyp blev enligt följande:

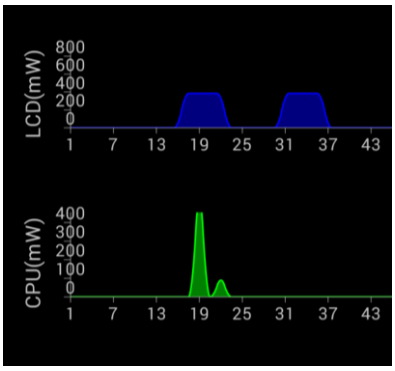
appery.io	31	ms
Smartface	861	ms
Xamarin	194	ms

5.5.3 Batteriförbrukning

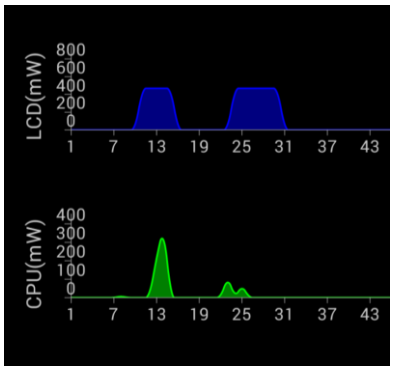
För att testa batteriförbrukningen har appen Power Tutor v1.4 använts (se 4.5.4), som efter installation startar direkt vid uppstart av testenheten. För att undvika att eventuella andra tjänster på testenheten stör testet har även här testenheten startats om i flygplansläge och därefter har prototypen startats och kallat på kamerafunktionen. Så fort fotot har sparats och visas i appen har primtalsfunktionen körts. Därefter har testenheten växlats över till Power Tutor och navigerat till *Application Viewer ⇨ [prototypens namn]* och under *Chart View* kan nedanstående grafer skådas för vardera prototyp:



Figur 2 *appery.io's batteriförbrukning*



Figur 3 *Smartface's batteriförbrukning*



Figur 4 *Xamarins batteriförbrukning*

Testet har utförts i den sista och kompletta versionen av appen och graferna som presenteras ovan är urklippta ur en printscreen som tagits på Power Tutor. Som synes på graferna presenterar Power Tutor batteriförbrukningen för enhetens skärm samt CPU för den valda appen under de senaste 60 sek. Med detta kan man alltså tydligt se hur vardera funktion som körts i prototypen påverkat enhetens batteri. Vid detta testtillfälle ser man att CPU'n inte påverkat batteriet alls när kamerafunktionen körts, men däremot primtalsfunktionen har fått CPU'ns batteriförbrukning att skjuta i höjden på Smartface-appen. Även i Xamarin-appen har primtalsfunktionen dragit betydligt mer batteri än vad kamerafunktionen gjort. Man kan också se att skärmens batteriförbrukning är väldigt likvärdig för alla apparna.

5.5.4 CPU-användning

Ett test för CPU-användning har tyvärr inte kunnat utföras. Detta beror dels på att det inte funnits tillräckligt med tid för att sätta sig in i verktyget DDMS för att sedan kunna plocka ut och tolka den data som det resulterar i, dvs. det test som alltså Dalmasso m.fl. (2013) använt sig av och inspirerat till att utföra. Dels beror det också på att försöket till att istället hitta en fungerande app, liknande Power Tutor (Google play, 2014g) men för CPU-användning, har misslyckats. Ett antal har testats (Google play, 2014e) men den mest lovande (Google play,

2014f) klarade inte av att visa någon förändring på grafen för Smartface-prototypen och därför förkastades även denna liksom de övriga.

5.5.5 Minnesanvändning

Ett test för minnesanvändning har inte heller kunnat utföras av samma anledning som för CPU-användning ovan.

6 Utvärdering

Detta kapitel börjar med att presentera resultatet från den undersökning som utförts på de protyper som tagits fram, och avslutas sedan med en analys på detta.

6.1 Presentation av undersökning

Inför varje test som gjorts har prototypens data rensats via inställningar ⇨ programhanteraren ⇨ [prototypens namn] på testenheten. Att rensa en apps data i android kan liknas vid att återställa appens data till grundläget, ungefär som att avinstallera och återinstallera appen igen, fast snabbare (Android Central, 2011).

Värt att nämna är att redan här skiljer sig apparna åt. Den data som finns att rensa bort efter att appen startats en gång, utan att göra något annat än att visa startsidan, är för appery.io-prototypen: 156 kB i ren data och 76 kB i cache medan det för Smartface-prototypen är 144 kB data och 20 kB cache. För Xamarin-prototypen är det endast 20 kB data och 12 kB cache.

6.1.1 Exekveringstiden

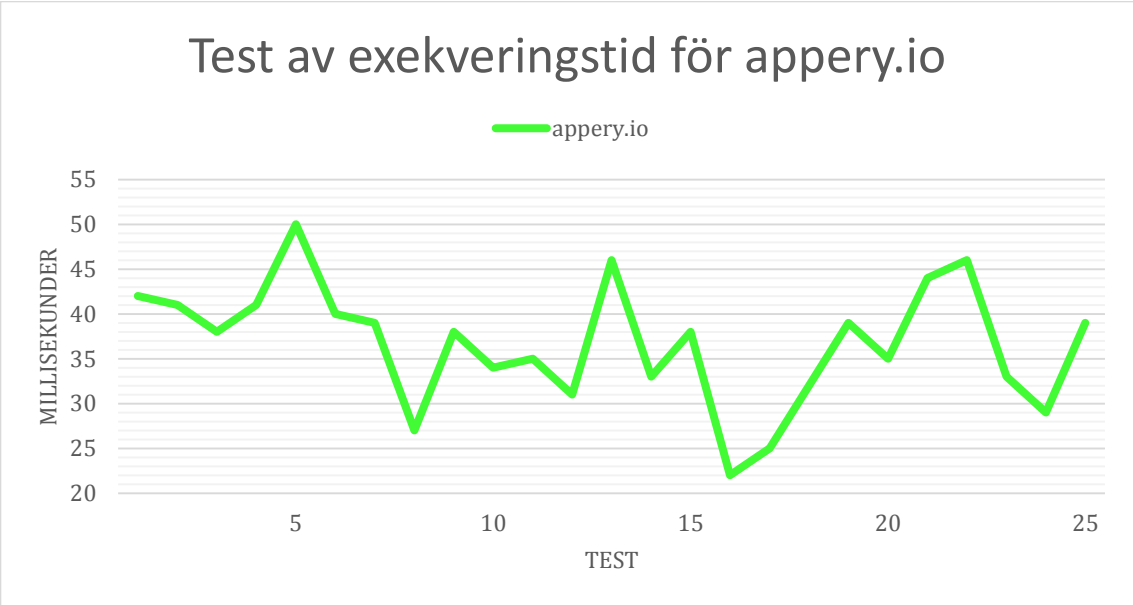
För att testa exekveringstiden har precis som nämns i kapitel 5.5.2 den primtalsfunktion som implementerats i prototypen använts. Testenheten har startats i flygplansläge och därefter har primtalsfunktionen körts direkt när appen har startat. Resultatet har antecknats och därefter har appen avslutats och dess data har rensats. Testet har upprepats 25 ggr och följande resultat har givits för vardera prototyp.

appery.io

Följande data för appery.io har registrerats. Värdena har grupperats i grupper om fem för att bli mer lättlästa och siffrorna syftar på tid i millisekunder för testenheten att köra primtalsfunktionen och presentera ett svar.

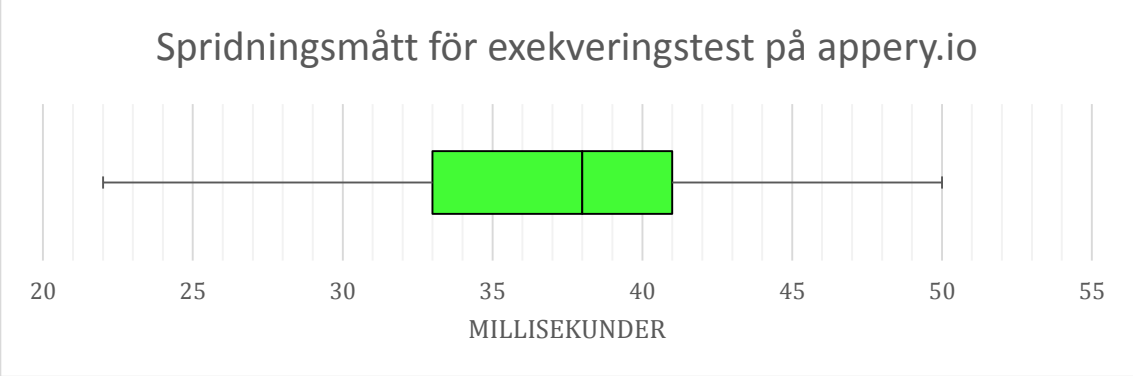
Test 1-5:	42	41	38	41	50
Test 6-10:	40	39	27	38	34
Test 11-15:	35	31	46	33	38
Test 16-20:	22	25	32	39	35
Test 21-25:	44	46	33	29	39

I Figur 5 nedan har resultatet sammanställts i ett linjediagram.



Figur 5 *Test av exekveringstid för appery.io*

Spridningen av de värden som registrerats (nedre kvartilen, medianen och övre kvartilen) visas i Figur 6 nedan.



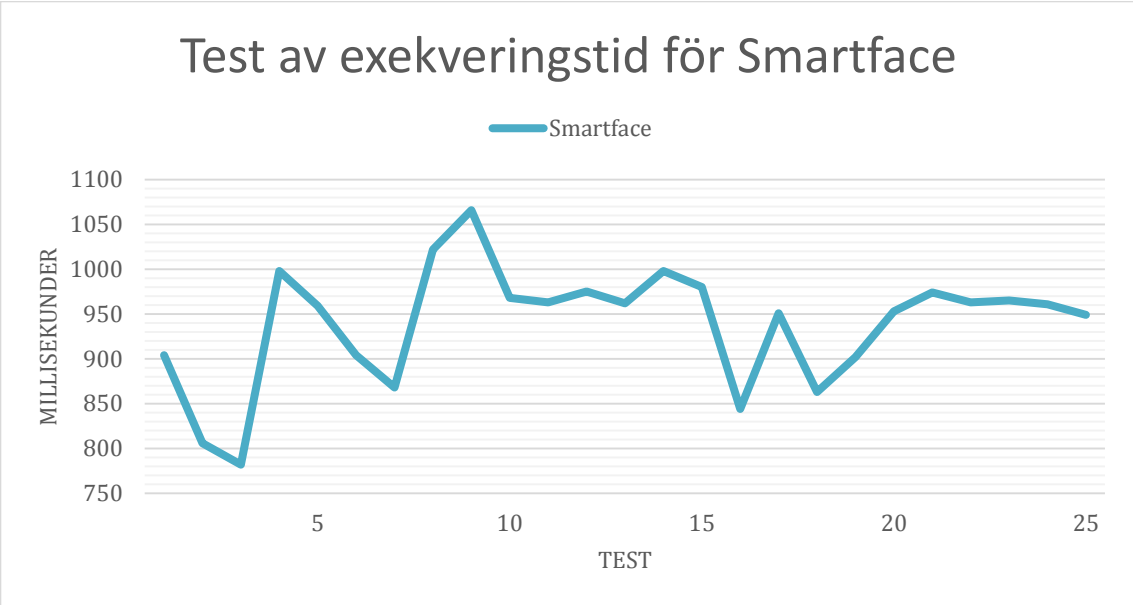
Figur 6 *Spridningsmått för exekveringstest på appery.io*

Smartface

Följande data för Smartface har registrerats. Värdena har grupperats i grupper om fem för att bli mer lättlästa och siffrorna syftar på tid i millisekunder för testenheten att köra primtalsfunktionen och presentera ett svar.

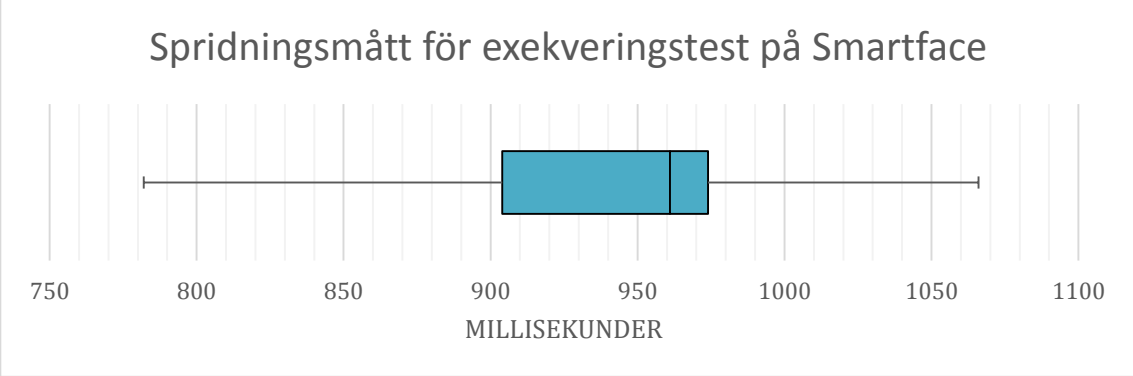
Test 1-5:	904	806	782	998	959
Test 6-10:	904	868	1022	1066	968
Test 11-15:	963	975	962	998	980
Test 16-20:	844	951	863	902	953
Test 21-25:	974	963	965	961	949

I Figur 7 nedan har resultatet sammanställts i ett linjediagram.



Figur 7 *Test av exekveringstid för Smartface*

Spridningen av de värden som registrerats (nedre kvartilen, medianen och övre kvartilen) visas i Figur 8 nedan.



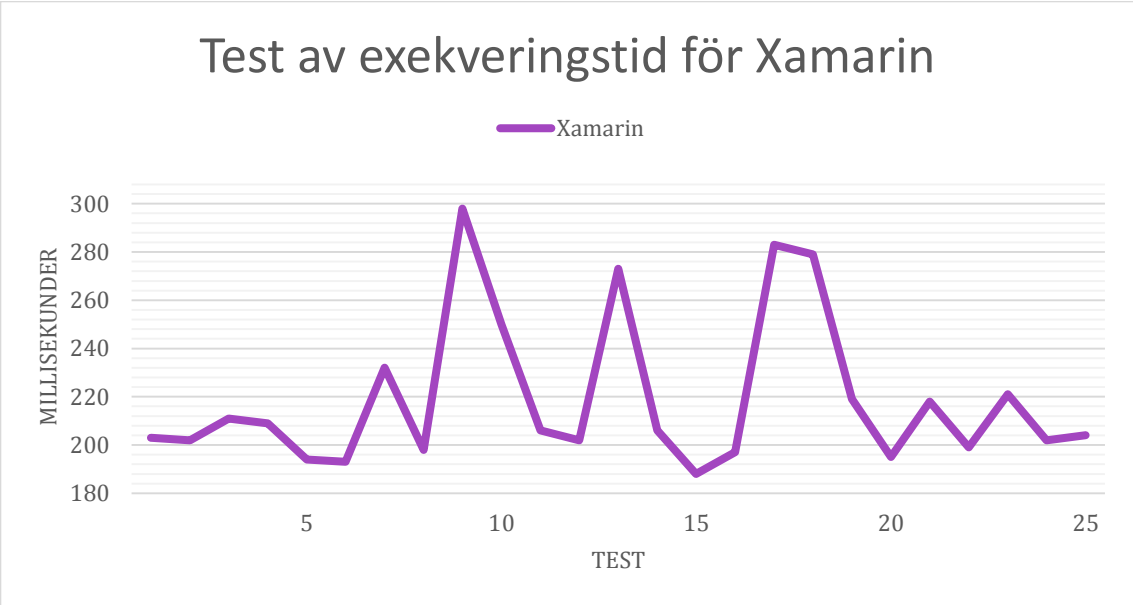
Figur 8 *Spridningsmått för exekveringstest på Smartface*

Xamarin

Följande data för Xamarin har registrerats. Värdena har grupperats i grupper om fem för att bli mer lättlästa och siffrorna syftar på tid i millisekunder för testenheten att köra primtalsfunktionen och presentera ett svar.

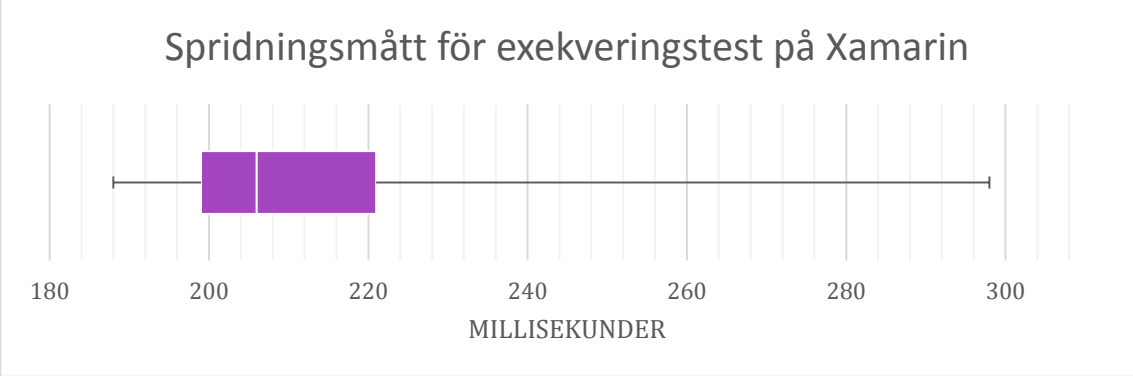
Test 1-5:	203	202	211	209	194
Test 6-10:	193	232	198	298	250
Test 11-15:	206	202	273	206	188
Test 16-20:	197	283	279	219	195
Test 21-25:	218	199	221	202	204

I Figur 9 nedan har resultatet sammanställts i ett linjediagram.



Figur 9 *Test av exekveringstid för Xamarin*

Spridningen av de värden som registrerats (nedre kvartilen, medianen och övre kvartilen) visas i Figur 10 nedan.



Figur 10 *Spridningsmått för exekveringstest på Xamarin*

6.1.2 Batteriförbrukning

För att testa batteriförbrukningen har precis som nämns i kapitel 5.5.3 appen Power Tutor använts. Testenheten har startats i flygplansläge och därefter har appen startats och kamerafunktionen körts. Så fort ett foto sparats och presenterats har primtalsfunktionen körts. Därefter har testenheten växlats över till Power Tutor och under Application Viewer ⇨ [prototypens namn] har en printscreen tagits. Slutligen har appen avlutats och dess data rensats.

Testet har upprepats 25 ggr och testernas printscreens har grupperats och publiceras på en gemensam bild i grupper om fem för att det ska bli enklare att läsa av och jämföra grafen som Power Tutor skapar. Testerna skiljs åt med ett rött vertikalt streck och grafens värden i x-led (tid i sek) har tagits bort, eftersom de annars skulle bli väldigt blandade siffror då varje test inte startat på sekunden samtidigt. Varje enskilt test är tänkt att läsas från hö till vä och den första händelsen syftar alltså på när kamerafunktionen kallas på, medan den andra händelsen syftar på när primtalsfunktionen körs. Uppehållet däremellan uppstår när enhetens inbyggda kamera-app istället körs. Originalen finns att se i Appendix K, Appendix L och Appendix M.

Det resultat som testet givit för vardera prototyp listas på nästkommande sidor.

appery.io

Följande värden för appery.io har registrerats. Av okänd anledning har anropet på kamerafunktionen inte registrerats på test 20 (se Bild 17 på nästa sida).

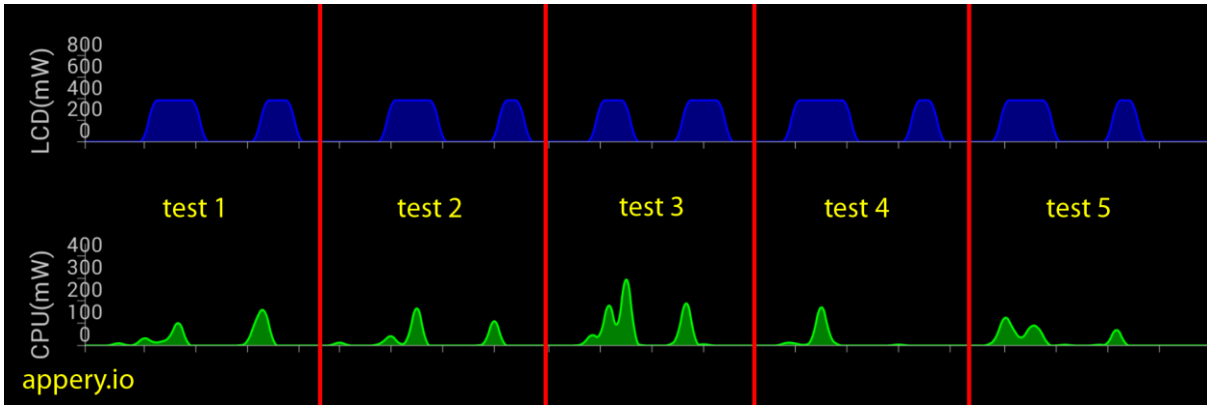


Bild 14 *Batteriförbrukning för appery.io, test 1-5*

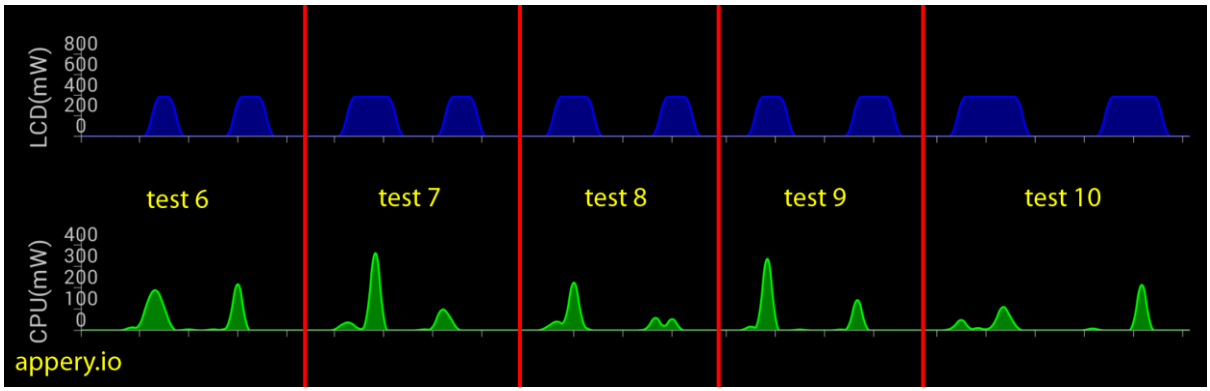


Bild 15 *Batteriförbrukning för appery.io, test 6-10*

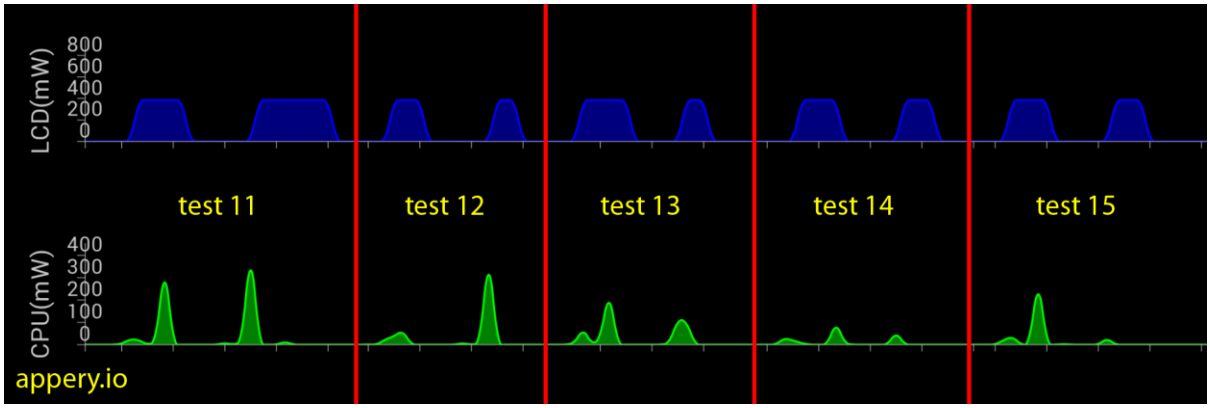


Bild 16 *Batteriförbrukning för appery.io, test 11-15*

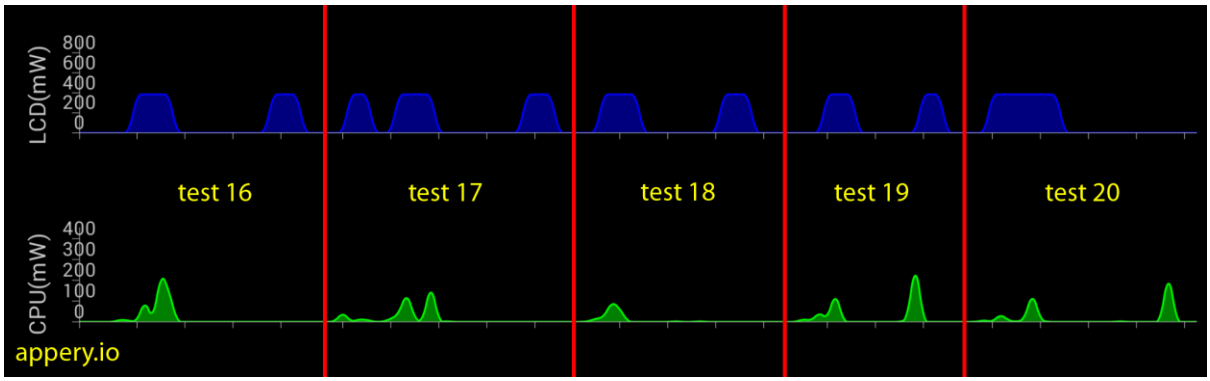


Bild 17 *Batteriförbrukning för appery.io, test 16-20*

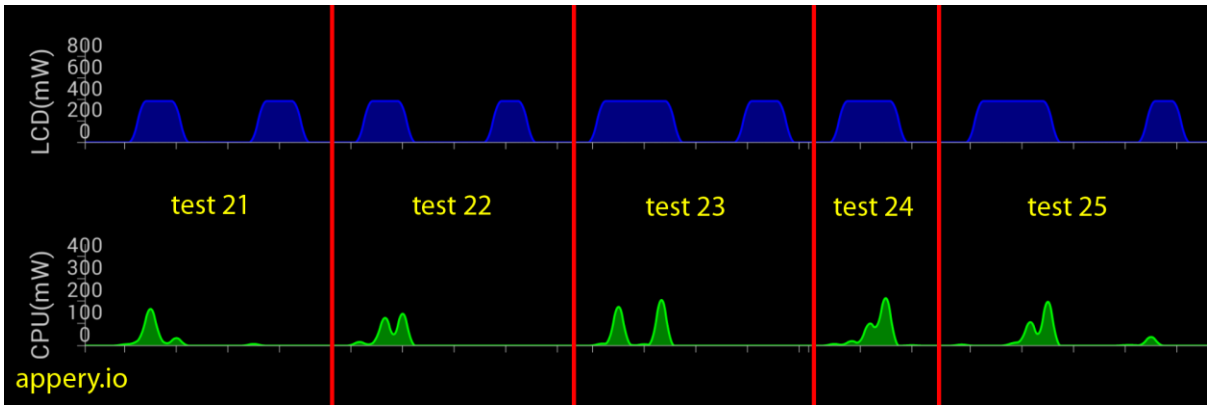
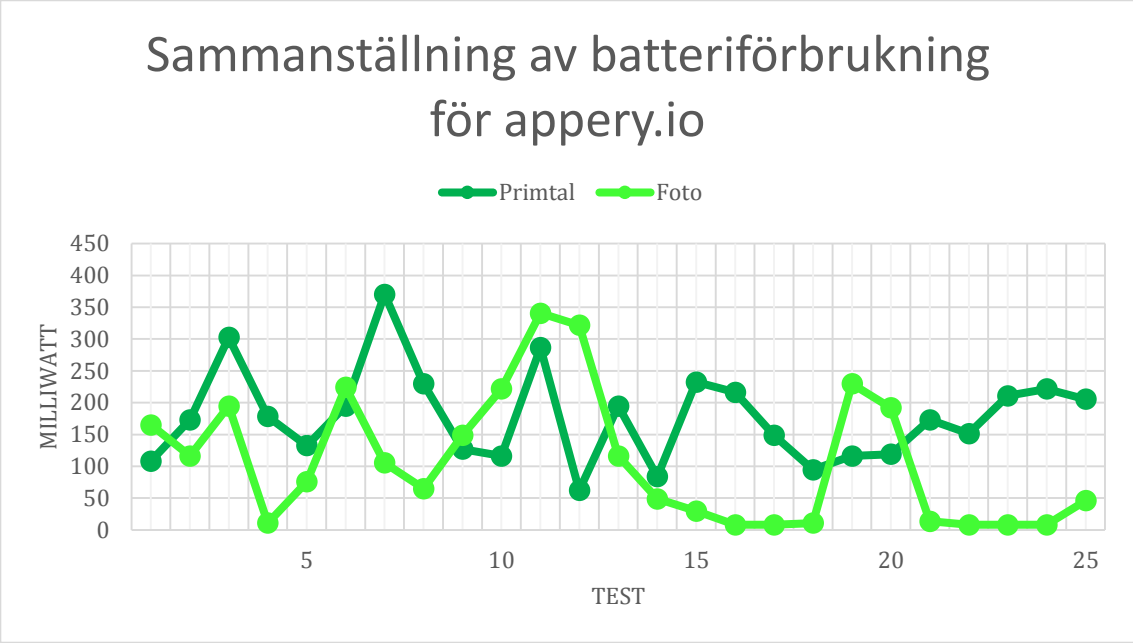


Bild 18 *Batteriförbrukning för appery.io, test 21-25*

I diagrammet på nästa sida (se Figur 11) har en sammanställning av resultatet gjorts. Ovanstående grafer har undersökts med hjälp av bildredigeringsverktyget Adobe Photoshop CS5 pixel för pixel, och en mätning och beräkning har skett för att ta reda på vad varje pixel ungefär representerar. Därefter har ett ungefärligt värde för varje gång då en funktion kallats på i appen antecknats. Det värde som antecknats är det högsta värde som kunnat läsas av under tiden som en funktion kallats.



Figur 11 *Sammanställning av batteriförbrukning för appery.io*

Smartface

Följande värden för Smartface har registrerats. Som synes på bilderna skjuter mätaren tyvärr i höjden vid ett flertal tillfällen och ett rimligt värde är därför svårt att utläsa.

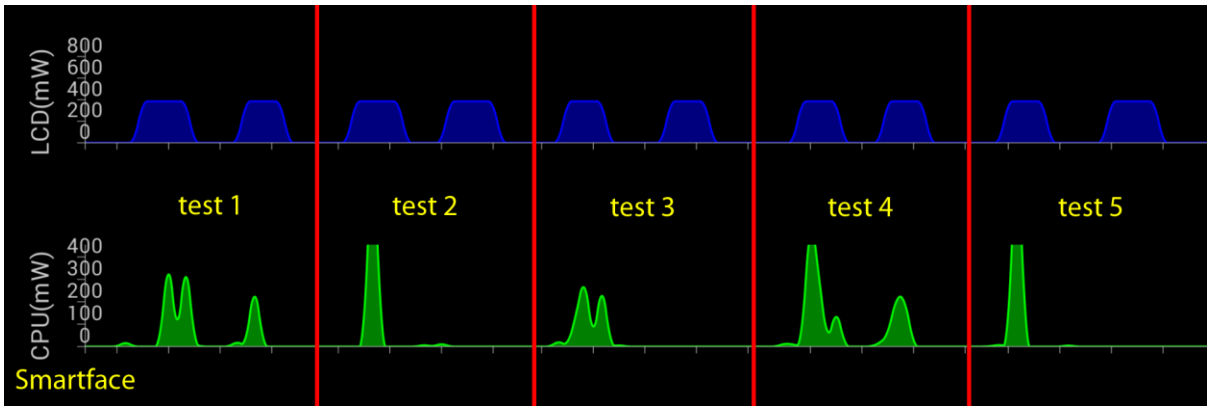


Bild 19 *Batteriförbrukning för Smartface, test 1-5*

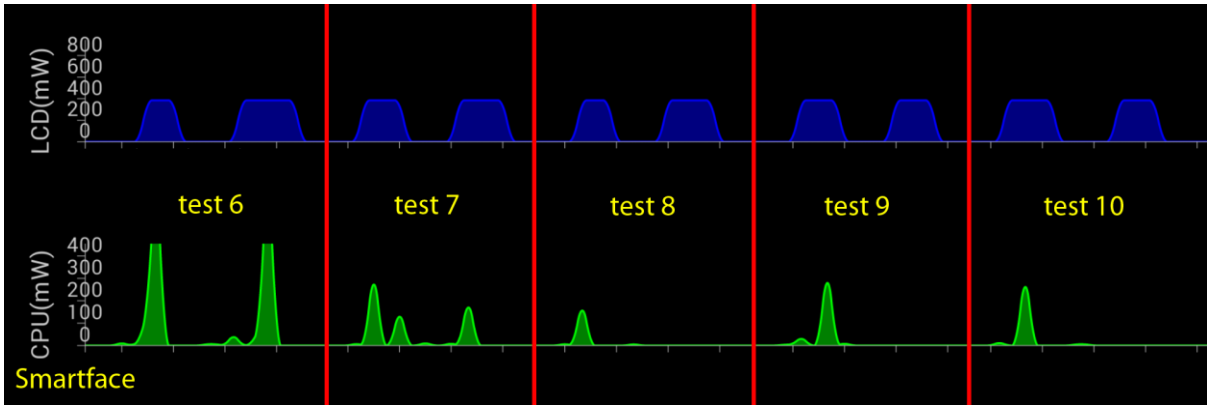


Bild 20 *Batteriförbrukning för Smartface, test 6-10*

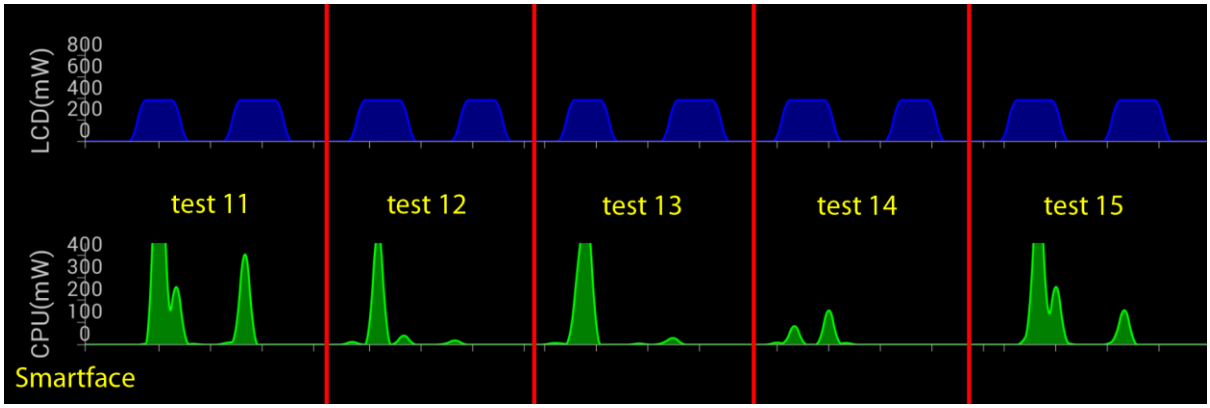


Bild 21 *Batteriförbrukning för Smartface, test 11-15*

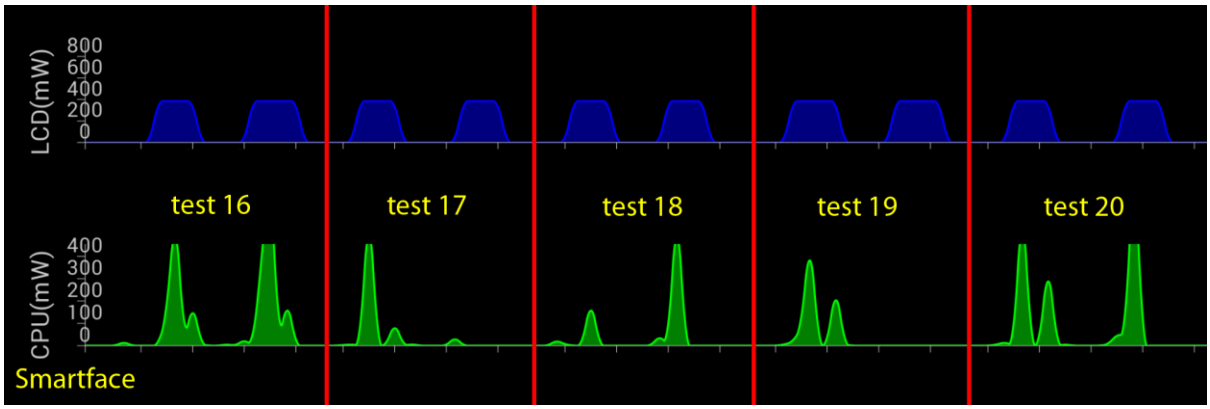


Bild 22 *Batteriförbrukning för Smartface, test 16-20*

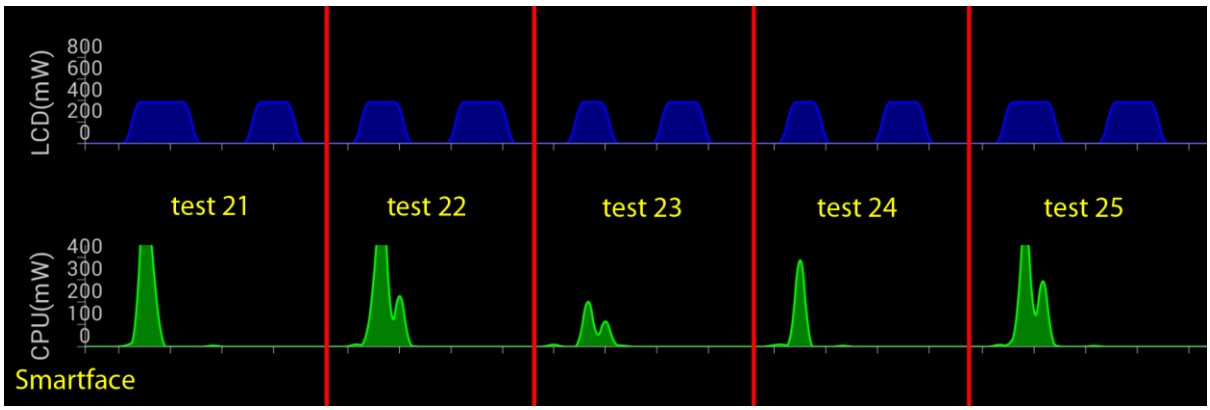
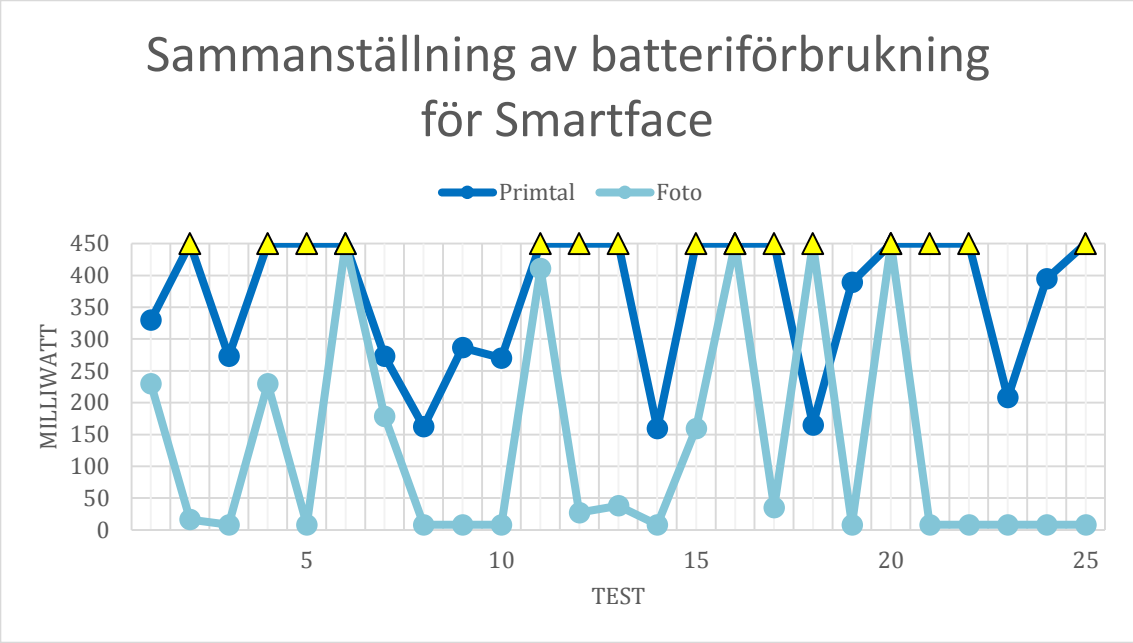


Bild 23 *Batteriförbrukning för Smartface, test 21-25*

På samma sätt som för appery.io så har en sammanställning av ovanstående grafer gjorts och resultatet visas i diagrammet på nästa sida (se Figur 12). Då det vid ett antal test registrerats ett värde som överstiger det som Power Tutor ritar ut har dessa givits värdet 450 och markerats med en gul triangel.



Figur 12 *Sammanställning av batteriförbrukning för Smartface*

Xamarin

Följande värden för Xamarin har registrerats. Lägg märke till att vid ett flertal tillfällen ser det ut som att inte bara två utan tre händelser har registrerats.

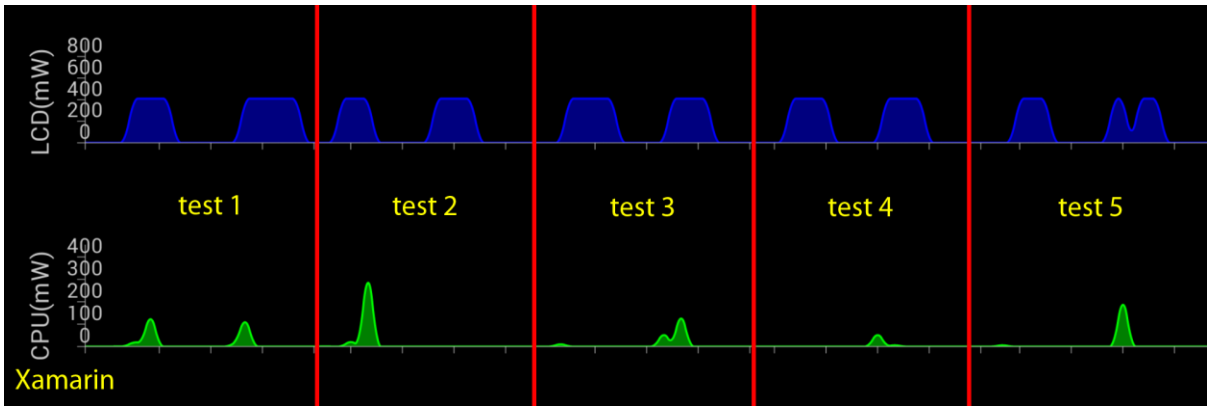


Bild 24 *Batteriförbrukning för Xamarin, test 1-5*

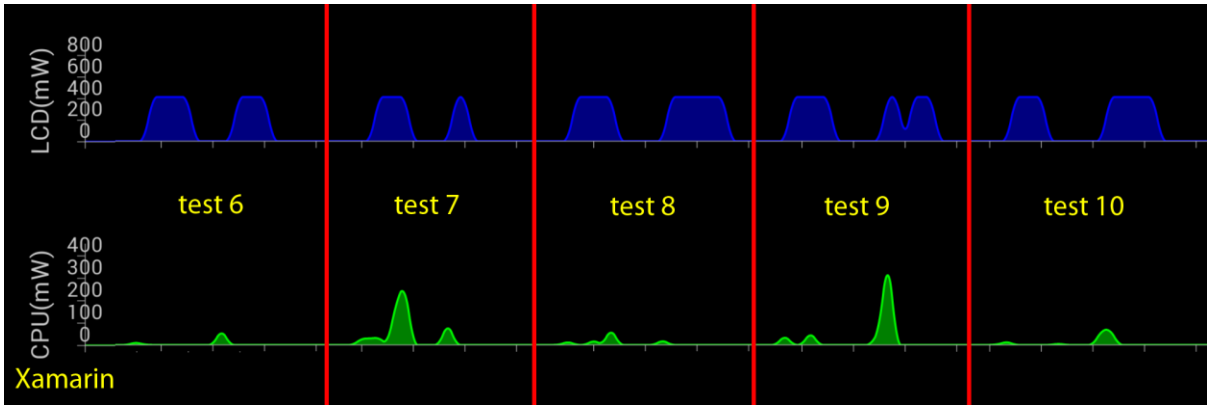


Bild 25 *Batteriförbrukning för Xamarin, test 6-10*

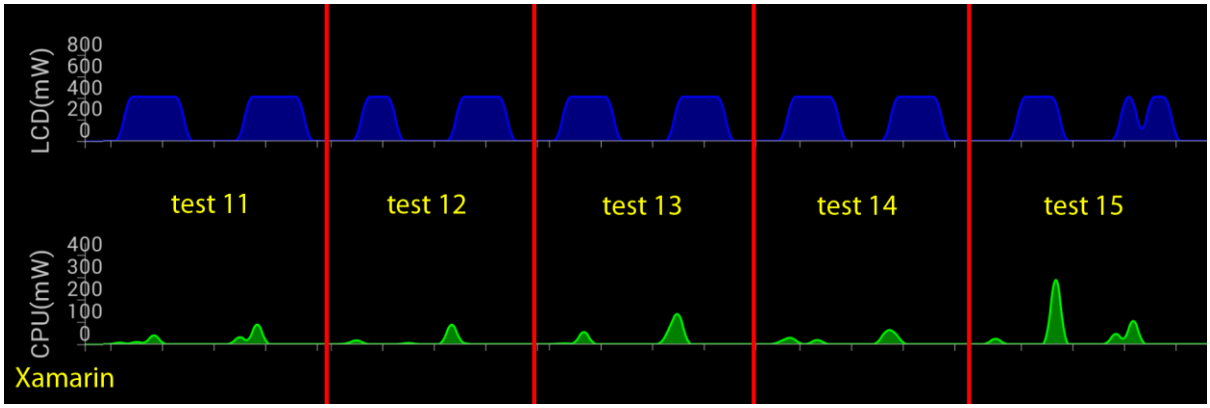


Bild 26 *Batteriförbrukning för Xamarin, test 11-15*

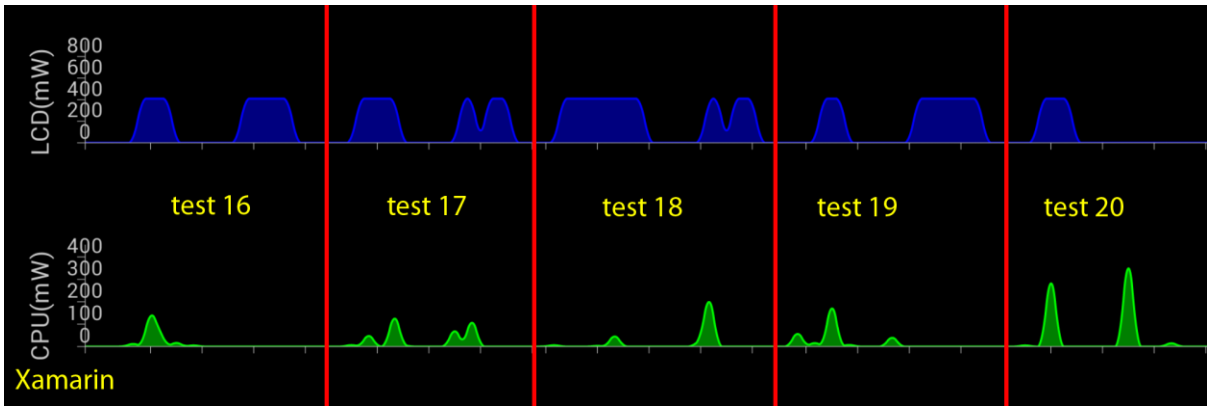


Bild 27 *Batteriförbrukning för Xamarin, test 16-20*

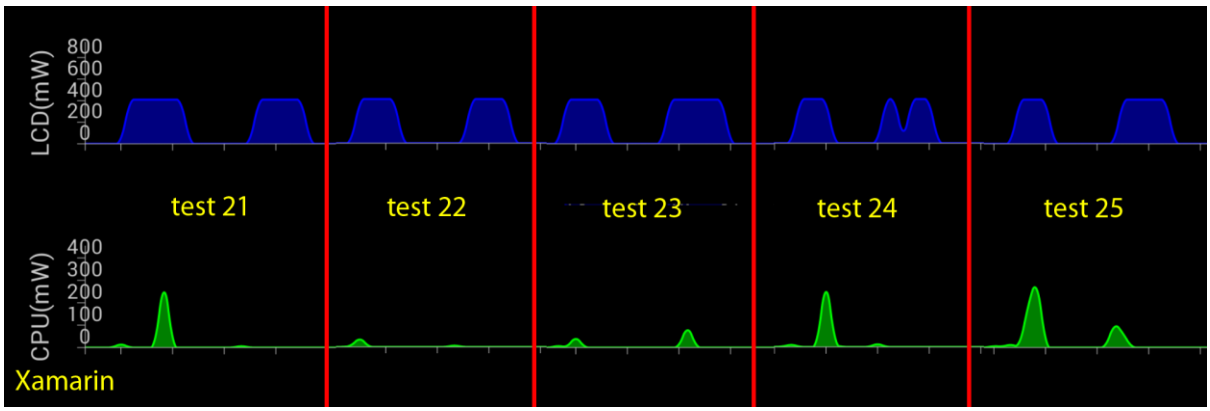
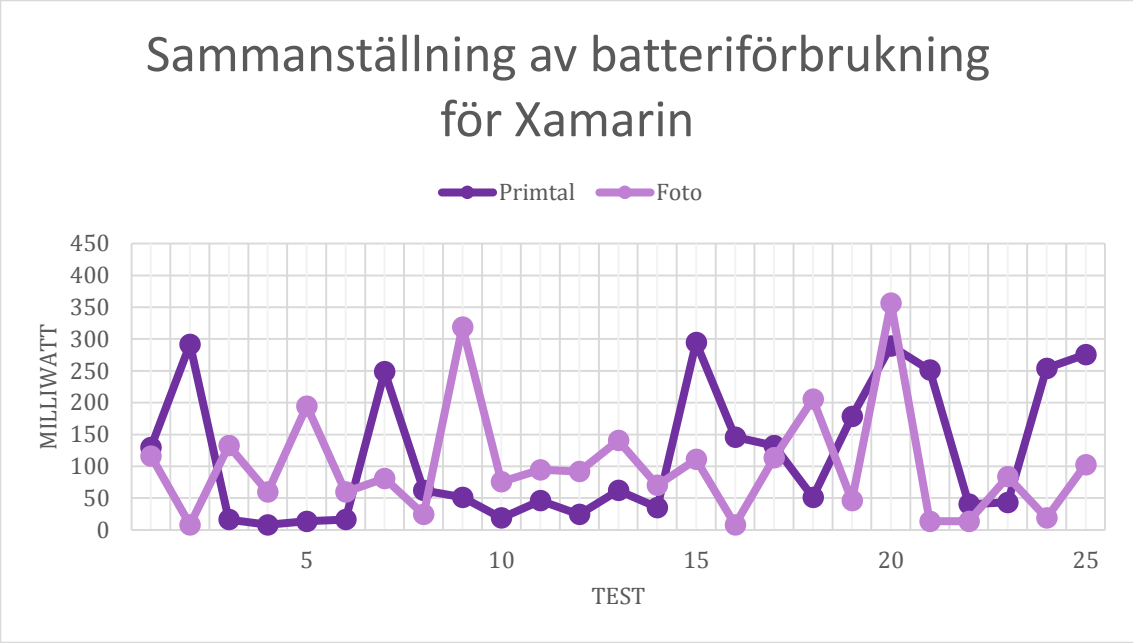


Bild 28 *Batteriförbrukning för Xamarin, test 21-25*

På samma sätt som för appery.io så har en sammanställning av ovanstående grafer gjorts och resultatet visas i diagrammet på nästa sida (se Figur 13).



Figur 13 *Sammanställning av batteriförbrukning för Xamarin*

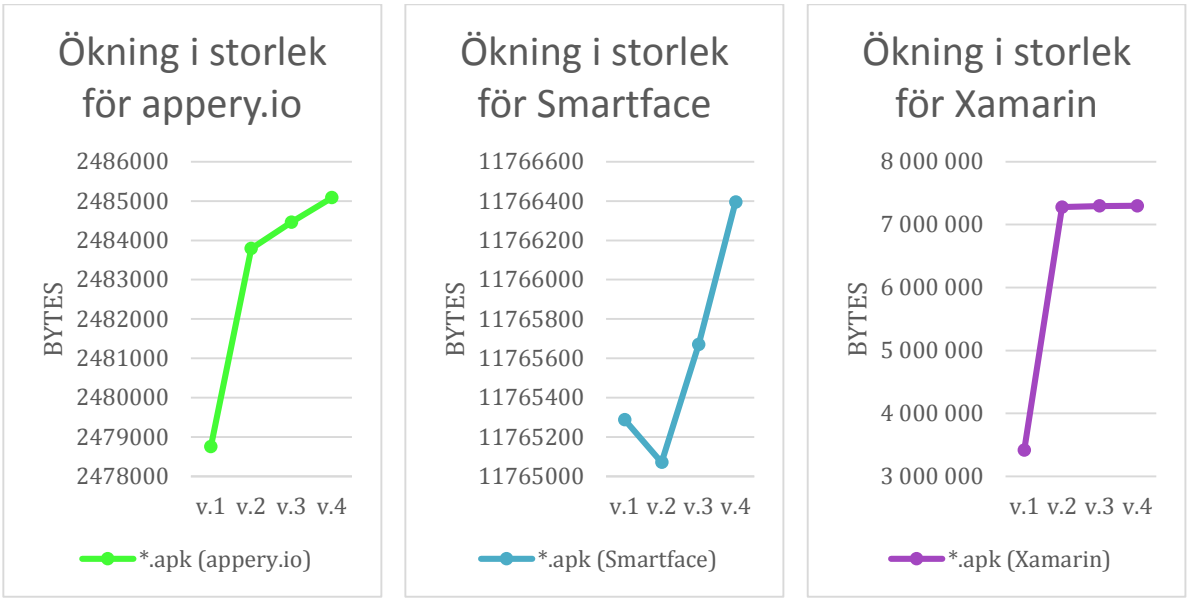
6.1.3 Storlek

Som beskrivs i kapitel 5.5.1 så har två värden för storleken på varje version av de olika prototyperna antecknats. Dels hur stor *.apk-filen är och dels hur stor testenheten säger att appen är när den väl installerats. Resultatet kan ses i Tabell 3 nedan.

Tabell 3 *Storleken på appen i respektive version för respektive prototyp*

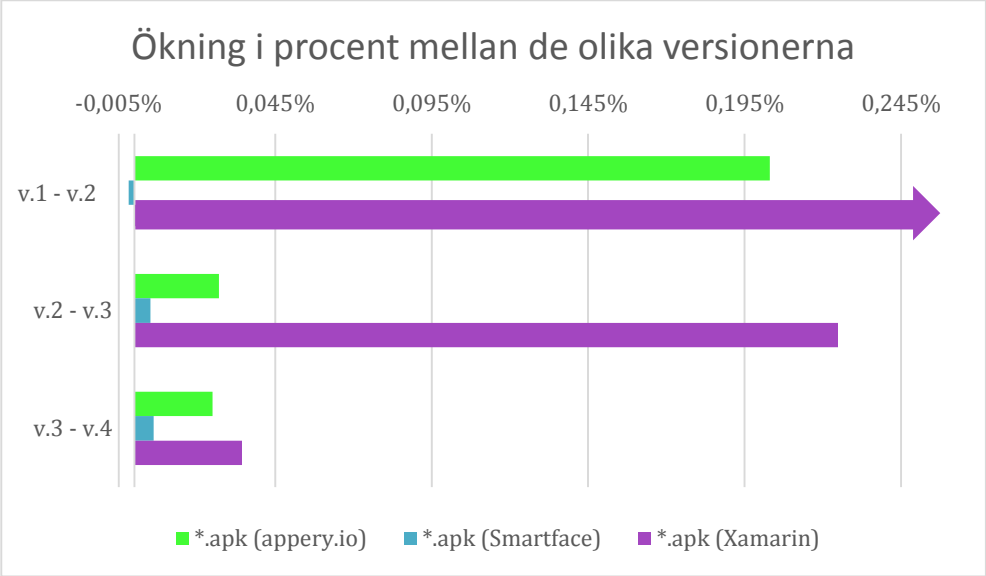
Version		**v1:** *Appen helt i grundläge*	**v2:** *Appen kan visa vart användaren befinner sig*	**v3:** *Appen kan ta ett foto*	**v4:** *Appen kan köra primtalsfunktionen*
***.apk**	**appery.io**	2 478 757 byte (2,36 MB)	2 483 790 byte (2,36 MB)	2 484 462 byte (2,36 MB)	2 485 083 byte (2,36 MB)
	Smartface	11 765 287 byte (11,2 MB)	11 765 071 byte (11,2 MB)	11 765 671 byte (11,2 MB)	11 766 395 byte (11,2 MB)
	Xamarin	3 416 104 byte (3,25 MB)	7 278 938 byte (6,94 MB)	7 295 302 byte (6,95 MB)	7 297 810 byte (6,95 MB)
Installerad	**appery.io**	3,42 MB	3,42 MB	3,42 MB	3,42 MB
	Smartface	34,04 MB	34,04 MB	34,04 MB	34,04 MB
	Xamarin	6,33 MB	11,31 MB	11,32 MB	11,33

På nästa sida visas resultatet som ett linjediagram för respektive prototyp (se Figur 14). Då det inte skiljer något i storlek för appery.io och Smartface när appen installerats, samt väldigt lite för Xamarin, så har denna information inte ritats ut. Värt att notera är att skalan på y-axeln för respektive linjediagram skiljer sig markant och diagrammet ger därför bara en individuell uppskattning om hur mycket prototypen har växt mellan versionerna.



Figur 14 *Storleksökning mellan versionerna för appery.io och Smartface*

I nedanstående stapeldiagram har en kombinerad sammanställning över hur mycket appen ökar procentuellt mellan de olika versionerna gjorts. Version 1 har satts till 0 och utgör alltså startvärdet och har därmed uteslutits ur diagrammet. Detta innebär alltså att version 2 för Smartface först minskar ifrån ursprunglig storlek i grundläget innan den sedan växer. OBS! Xamarin ökar med drygt 113 % mellan version 1 och version 2 vilket inte syns på diagrammet, men detta har markerats med en pil för att visa att denna stapel sträcker sig längre än vad som är synligt.



Figur 15 *En jämförelse i ökningen mellan de olika versionerna*

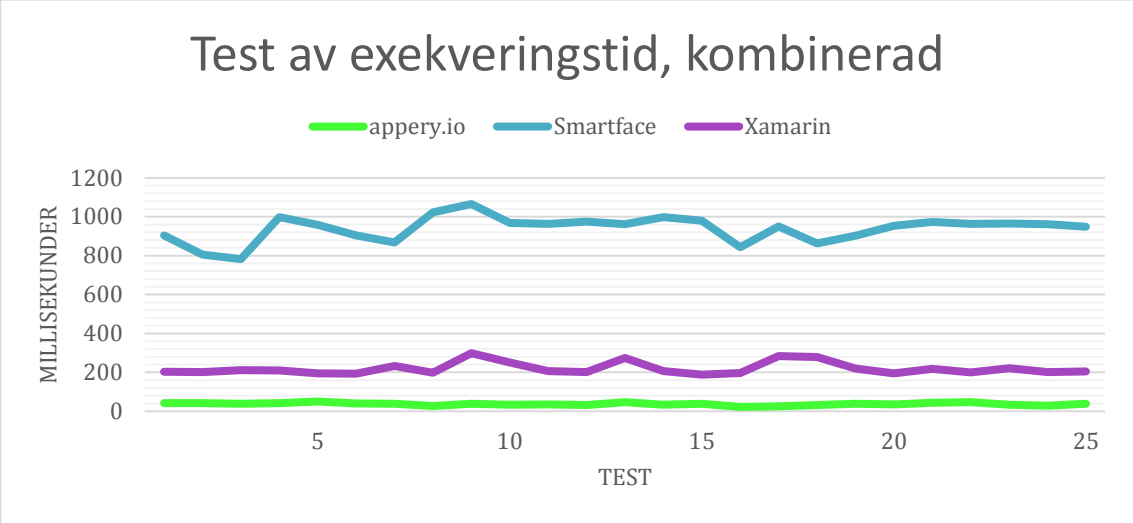
6.2 Analys

I detta kapitel analyseras det resultat som experimenten har givit.

6.2.1 Exekveringstid

Vid en första undersökning av diagrammen som representerar exekveringstiden för respektive prototyp (se Figur 5, Figur 7 och Figur 9) ser det ut som att de olika prototypernas värden varierar ungefär lika mycket, men om dessa kombineras i ett gemensamt diagram ser det helt

annorlunda ut. I Figur 16 nedan syns det tydligt att Smartface varierar väldigt mycket medan appery.io ligger väldigt stabilt. Xamarin varierar något mer än appery.io men inte mycket.



Figur 16 *Test av exekveringstid, kombinerad*

Det syns också tydligt att det skiljer väldigt mycket på hur lång tid det tar för respektive prototyp att utföra primtalsfunktionen. Om värdena i Figur 6, Figur 8 och Figur 10 kombineras i en tabell (se Tabell 4 nedan) kan man se att appery.io har en median på 38 medan Xamarin mer än fyrdubblar detta värde och ligger på 206. Smartface ligger på 961 vilket innebär mer än fyra gånger Xamarins värde och mer än 25 gånger appery.io's värde.

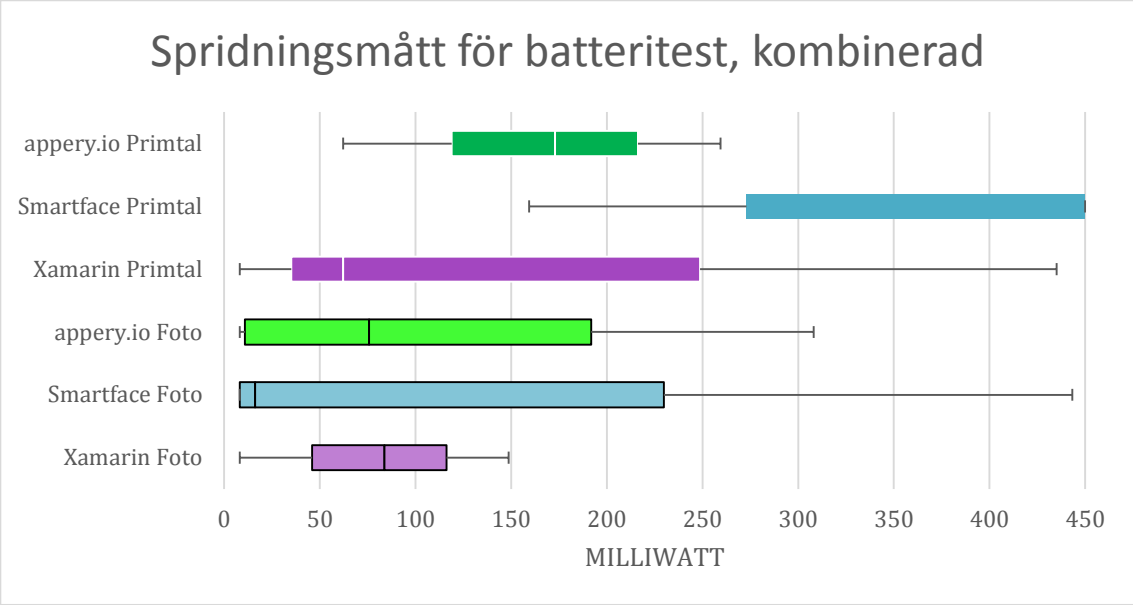
Tabell 4 *Sammanställning av median, min- och maxvärde, nedre och övre kvartil samt standardavvikelse*

Prototyp	Min	Q1	Median	Q2	Max	Standardavvikelse
appery.io	22	33	38	41	50	6,77
Smartface	782	904	961	974	1066	66,05
Xamarin	188	199	206	221	298	31,62

6.2.2 Batteriförbrukning

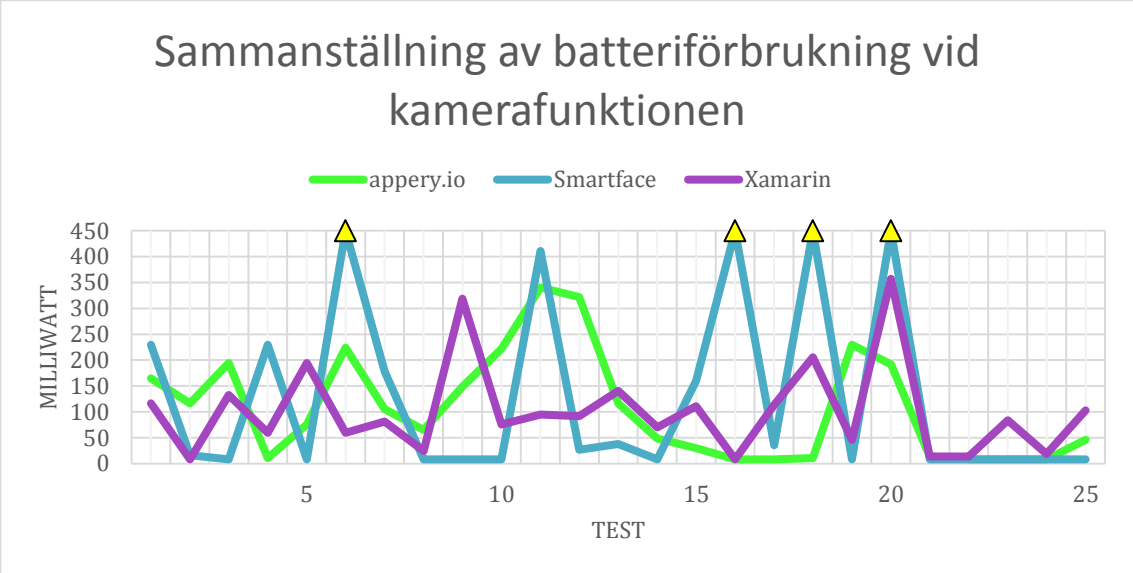
Testen för batteriförbrukningen visar att anropet på kamerafunktionen drar betydligt mindre mW (milliWatt) än när primtalsfunktionen körs för både appery.io och Smartface. Vid ett flertal tillfällen ser det till och med ut som att den inte drar något alls. Detta har dock, i de diagram som tagits fram med hjälp av mätvärden tagna via Adobe Photoshop (se Figur 11, Figur 12 och Figur 13), givits ett ungefärligt värde på drygt åtta eftersom linjen trots allt fortfarande är synlig. Primtalsfunktionen är dock betydligt mer krävande, så krävande att det tyvärr överstiger det maximala värdet (400 mW) på grafen i PowerTutor vid ett flertal tillfällen för Smartface.

Xamarin skiljer sig dock från de andra två. Här verkar istället kamerafunktionen vara mer krävande än primtalsfunktionen. Det skiljer dock inte väldigt mycket mellan batteriförbrukningen för kamerafunktionen respektive primtalsfunktionen för Xamarin utan dessa ligger båda relativt låg om man tittar på Figur 17 på nästa sida. Observera att medianen för primtalstestet för Smartface ligger utanför det faktiskt läsbara värdet.



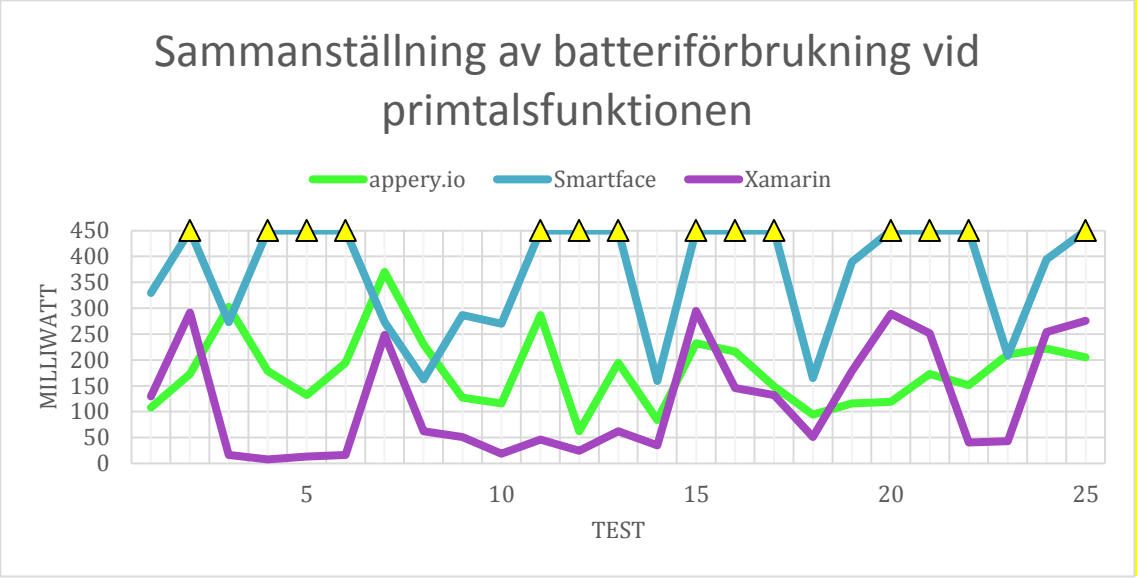
Figur 17 *Spridningsmått för batteritest, kombinerad*

Om batteriförbrukningen vid kamerafunktionen för respektive prototyp kombineras i ett gemensamt linjediagram (se Figur 18 nedan) syns det tydligt att Smartface antingen får ett så lågt värde att det knappt syns eller ett så högt värde att det inte är mätbart, medan appery.io och Xamarin ligger mer likvärdigt.



Figur 18 *Sammanställning av batteriförbrukning vid kamerafunktionen*

Om istället batteriförbrukningen vid primtalsfunktionen för respektive prototyp kombineras (se Figur 19 på nästa sida) syns det tydligt att appery.io mestadels ligger en bra bit över Xamarin som i huvudsak ligger riktigt lågt. Smartface ligger tydligt över det andra två och har dessvärre flera värden som är så höga att det inte ens går att läsa.



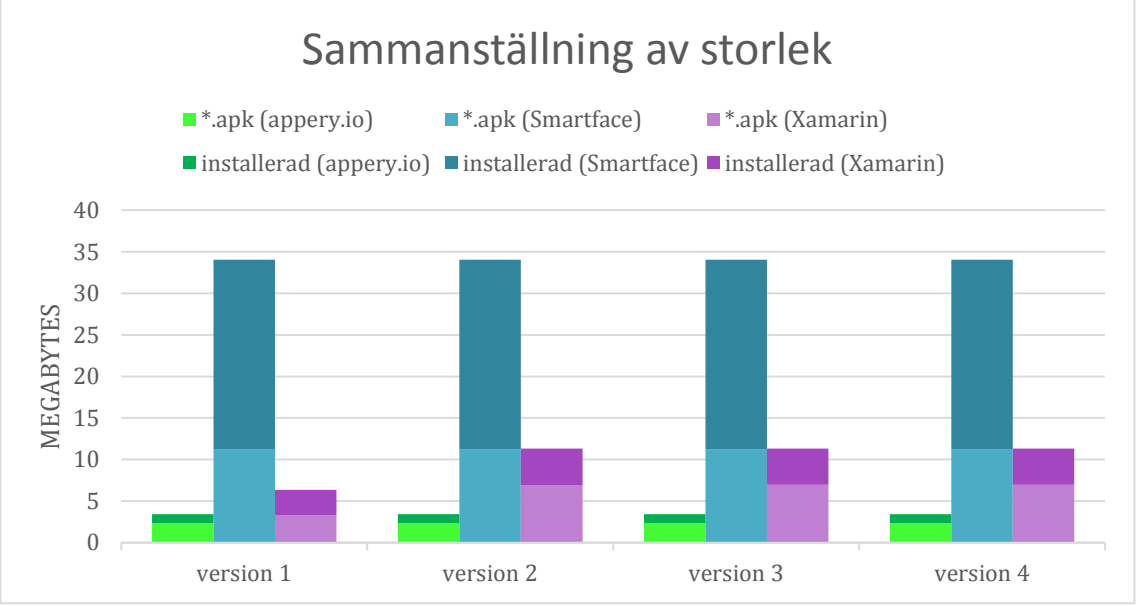
Figur 19 *Sammanställning av batteriförbrukning vid primtalsfunktionen*

6.2.3 Storlek

Att storleken skiljer sig ganska markant mellan de olika prototyperna syns tydligt i Tabell 3 (se kapitel 6.1.3). I Figur 20 nedan har denna data kombinerats i ett stapeldiagram. Som synes är Smartface nästan fem gånger så stor som appery.io, sett till *.apk-filen (de ljusa partiet i respektive stapel), medan Xamarin bara är något större i version 1. I version 2 och vidare så är dock Xamarin nästan tre gånger så stor som appery.io.

Den installerade appen (hela den kombinerade stapeln) skiljer sig även den ganska kraftigt i storlek. Som synes i diagrammet så är Xamarin nästan dubbelt så stor som appery.io i första versionen och tre gånger så stor i resterande versioner, medan Smartface är nästan tio gånger så stor i alla versioner.

När apparna installeras ökar appery.io i storlek med ca 50 % medan Smartface ökar med hela 300 % (de mörka partiet i respektive stapel). Xamarin ökar med nästan 100 % för version 1 men ligger sedan på en ökning på ca 50 % för resterande versioner.



Figur 20 *Sammanställning av storlek*

7 Avslutande diskussion

I detta kapitel sammanfattas först hela studien och därefter följer en diskussion om studiens resultat och trovärdighet samt dess samhälleliga nytta. Kapitlet avslutas med förslag på hur studien skulle kunna vidareutvecklas.

7.1 Sammanfattning

Den frågeställning som denna rapport skulle besvara var *”Vilka multiplattforms-utvecklingsverktyg finns att tillgå idag och hur mycket skiljer det ifråga om egenskaper mellan de appar som tas fram i respektive, om grunden till appen är densamma?”*

Förstudien i form av en kartläggning resulterade i att 18 multiplattformsutvecklingsverktyg hittades. Utifrån dessa valdes sedan tre stycken ut att arbeta vidare med. En prototyp togs fram i respektive utvecklingsverktyg med egenskaperna att kunna peka ut vart användaren befann sig, ta ett foto och visa detta, samt köra en primtalsfunktion och sedan returnera svaret samt hur lång tid beräkningen tog.

När prototyperna var färdiga jämfördes dessa mot varandra ifråga om storlek, exekveringstid och batteriförbrukning och slutsatsen blev att det skiljer precis som misstänkt ganska mycket mellan dem. Den första prototypen, skapad i appery.io, var relativt liten i storlek, väldigt snabb vid exekvering och drog något mer batteri för primtalsfunktionen än för kamerafunktionen. Den andra prototypen, skapad i Smartface, var betydligt större ifråga om storlek, betydligt långsammare vid exekveringstestet och drog betydligt mer batteri för primtalsfunktionen än kamerafunktionen. Den sista prototypen, skapad i Xamarin, var något större i storlek än den första, men betydligt mindre än den andra, något långsammare vid exekvering än den första, men betydligt snabbare än den andra och låg mer jämnt fördelad i batteriförbrukning för primtalsfunktionen respektive kamerafunktionen.

Hypotesen att det troligtvis skiljer ganska mycket mellan de appar som tillverkas i olika multiplattformsutvecklingsverktyg (se 3.2), trots att de utgår ifrån samma grund, kan härmed bekräftas i och med denna studie.

7.2 Diskussion

Denna studie har besvarat den första delen i frågeställningen *”Vilka multiplattforms-utvecklingsverktyg finns att tillgå idag och hur mycket skiljer det ifråga om egenskaper mellan de appar som tas fram i respektive, om grunden till appen är densamma?”* genom att samla alla de multiplattformsutvecklingsverktyg som gått att finna. Denna studie har också delvis besvarat den andra delen i frågeställningen genom att ta fram och jämföra en likvärdig prototyp från tre utav de utvecklingsverktygen som hittades. Denna studie har därmed arbetat vidare med förslaget att jämföra olika multiplattformsutvecklingsverktyg med varandra som Andersson & Andreasson (2013) samt Palmieri, Singh & Cicchetti (2012) föreslog.

7.2.1 Resultatet

Den skillnad som visar sig mellan Xamarin och de övriga i exekveringstestet skulle kunna bero på att implementationen av primtalsfunktionen kräver en alternativ lösning i C#, jämfört mot den lösning i JavaScript som används för både appery.io och Smartface. Men appery.io och Smartface använder sig av en identisk lösning och trots detta så skiljer det väldigt mycket dem emellan. Det känns inte troligt att det enda som skiljer dem åt, själva presentationen av

resultatet, skulle vara det som gör hela skillnaden. Förklaringen måste alltså med största sannolikhet ligga i hur utvecklingsverktyget översätter programmerarens kod till en native-app, dvs. hur appery.io respektive Smartface har valt att tolka JavaScript-koden.

Batteritestet ger även det väldigt skilda resultat mellan prototyperna. Sett till primtals-funktionen så förbrukar Smartface så mycket batteri att värdet hamnar utanför vad som är mätbart, dvs. vad utvecklaren av PowerTutor ansåg vara nödvändigt att kunna mäta. Detta skulle naturligtvis kunna hänga ihop med att primtalsfunktionen tar påtagligt längre tid att utföra för just Smartface.

För Xamarin är det istället kamerafunktionen som förbrukar mer batteri. En tanke är att detta kanske beror på att bilden efter att den tagits, förminskas i storlek med hjälp av BitmapHelpers-klassen (se Appendix J) innan den visas i appen. I appery.io och Smartface har detta troligtvis löst på ett annat sätt som inte kräver lika mycket batteri som det gör för Xamarin.

Att det skiljer en del på batteritestens resultat, kan naturligtvis ha att göra med att det i Android under uppstarten av testenheten sannolikt startas upp en hel del tjänster i bakgrunden som användaren inte ser, men som naturligtvis påverkar testresultatet. Med andra ord, om testet inte utförts efter exakt lika många millisekunder efter att testenheten startats om, är risken stor att resultaten påverkas olika. Trots detta så syns tydligt ett mönster för vardera prototyp som lägger dem på helt skilda medianvärden, vilket bevisar att trots denna förklaring så skiljer sig tydligt prototyperna åt.

När det gäller skillnaden i storlek mellan prototyperna är det framförallt Smartface som sticker ut. Vad är det som gör Smartface så enormt mycket större än de andra, trots att de alla har samma funktioner i den slutliga appen? En teori skulle kunna vara att Smartface valt att inkludera alla de bibliotek, som eventuellt skulle kunna behövas i en app, redan från start. Denna teori baseras på hur påtagligt mycket Xamarin växte när biblioteket Google Play inkluderades. Men hur kommer det sig då att appery.io är så otroligt liten när den har precis samma fullt fungerande funktioner? Detta beror troligtvis på att appery.io skapar en hybridapp och inte en nativeapp som de andra. Intressant är även att från och med version 2 så växer både appery.io och Xamarin med ca 50 % vid installation, men Smartface växer med otroliga drygt 100 %.

Om resultatet från denna studie jämförs emot resultatet från de tidigare studier som gjorts och som refereras till i denna studie, så kan det konstateras att dessa studier bekräftar det resultat som denna studie nått fram till och vice versa. Det vill säga, det skiljer i slutresultatet mellan prototyper skapade i olika multiplattformsutvecklingsverktyg.

7.2.2 Samhällelig nytta

Denna studie bidrar till en större förståelse för den möjlighet som ett multiplattforms-utvecklingsverktyg erbjuder.

Om fler utvecklare inser potentialen med dessa utvecklingsverktyg, ges dessa utvecklare möjligheten att välja om de vill lägga ett projekts ofta begränsade tid på att utveckla en enklare app i flera versioner (dvs. ”uppfinna hjulet på nytt” för varje målplattform), eller om de istället vill lägga tiden på att utveckla en mer avancerad app som sedan enkelt kan nå ut till flera målplattformar. För vissa projekt är det kanske viktigare att nå ut till så många som möjligt medan de för andra är viktigare att appen verkligen utnyttjar en enskild målplattforms fulla potential.

Om appens uppgift är att informera om något samhällsnyttigt som exempelvis hur vi ska sortera sopor, eller hjälper oss att sammanställa vår elförbrukning och ger tips på hur vi kan spara in på denna resurs, så är det troligtvis viktigare att nå ut till så många som möjligt. Här kan alltså ett multiplattformsutvecklingsverktyg spela stor roll för både vår miljö och vårt samhälle.

Vid utveckling av en app för människor i tredje världen kan det vara extra viktigt att nå ut till så många målplattformar som möjligt eftersom dessa människor troligtvis inte har möjlighet att inneha ”den senaste och mest populära” målplattformen utan istället får använda sig utav det som de får tag i, dvs att det kan vara en ganska blandad kompott med andra ord. Då det troligtvis inte finns obegränsat med resurser för att utveckla en app för dessa människor är det troligtvis av högsta vikt att så få utvecklare som möjligt, på så kort tid som möjligt kan utveckla appen.

Utvecklare som inte har kunskapen för att utveckla native mobilappar, ges möjligheten att ändå ta sig in på mobilapps-marknaden, tack vare dessa multiplattformsutvecklingsverktyg. Detta gör i sin tur att fler smarta idéer kan spridas. Men risken finns ju naturligtvis också för att fler mobila appar som egentligen inte håller måttet, kommer ut på marknaden.

7.2.3 Etik

Som nämns i kapitel 4.6 så är det alltid riskfyllt att undersöka egenskaper hos en app med en benchmark-app. Resultaten från batteritestet skall därför tolkas med reservation. Den kod som PowerTutor baseras på har inte kunnat undersökas och därför är det omöjligt att avgöra om PowerTutor på något sätt manipulerar resultatet. Det skulle också kunna vara så att något utav utvecklingsverktygen integrerat en funktion i dess prototyp som känner av när appen benchmarkas och då påverkar testresultatet.

Vidare finns risken att testresultatet blir ofullständigt med en benchmark-app, vilket uppstod i denna studie då flertalet av resultaten för just Smartface hamnade utanför det läsbara värdet. Detta beror på valet att begränsa det läsbara värdet till 400, som utvecklaren av appen PowerTutor har tagit. Resultatet gör att det blir omöjligt att avgöra om dessa värden hamnade precis ovanför det läsbara eller långt över det läsbara värdet. Om resultaten egentligen hamnade långt över det läsbara, innebär det att Smartface borde se helt annorlunda ut i förhållande till de andra utvecklingsverktygen när man kombinerar batteritestet.

De tre multiplattformsutvecklingsverktyg som denna studie valt att utvärdera har valts ut för att vara så representativa som möjligt (se kapitel 4.2 och 5.2) för den mängd utvecklingsverktyg som hittats. Trots det finns risken att dessa tre skiljer sig avsevärt från de utvecklingsverktyg som vart och ett är tänkt att representera. Detta innebär att det finns en stor risk för att det resultat som tagits fram inte är representativt för multiplattformsutvecklingsverktyg i allmänhet och det måste tas i beaktande. För att få en helt rättvis bild av de utvecklingsverktyg som finns att tillgå idag bör vart och ett undersökas.

Då källkoden för de flesta av utvecklingsverktygen inte finns tillgänglig, och det därför är omöjligt att veta hur respektive utvecklingsverktyg översätter programmerarens kod till en körbar app, så måste risken att okänd kod inkluderas i appen beaktas. Det finns ingenting som hindrar utvecklarna av utvecklingsverktyget från att inkludera exempelvis ett spionprogram i alla de appar som utvecklas med dess hjälp. Detta skulle kunna utnyttjas genom att exempelvis obemärkt styra olika funktioner i telefonen eller tala om vart telefonen och dess användare befinner sig.

Om exempelvis en bank-app utvecklades med ett multiplattformsutvecklingsverktyg skulle detta innebära en stor säkerhetsrisk, eftersom app-utvecklaren inte kan garantera att den kod som appen innehåller är ofarlig. Det skulle exempelvis kunna vara så att appen osynligt skickar information om konton, saldon m.m. till verktygs-utvecklaren som sedan missbrukar denna information.

Det finns också en risk för att buggar som verktygs-utvecklaren ännu inte hittat, följer med ut genom en app till en användare. Dessa buggar kan i värsta fall vara rent skadliga för enheten.

Värt att tänka på är alltså att den möjlighet som ett multiplattformsutvecklingsverktyg ger, dvs. att nå ut till så många som möjligt med så liten ansträngning som möjligt och på så kort tid som möjligt, också öppnar upp risken för att detta skulle kunna utnyttjas i onda avsikter.

Då de utvecklingsverktyg som använts i denna studie hela tiden vidareutvecklas finns risken för att denna studie inte ska gå att återupprepa trots att all producerad programkod samt information om utvecklingsmiljön bifogas till denna rapport i enlighet med kapitel 4.6. Om det är möjligt att få tag i exakt samma version av utvecklingsverktygen som använts, bör det dock inte vara några problem, men det finns en stor risk för att detta inte går att åstadkomma då verktyget appery.io inte laddas ner utan körs direkt på webben.

7.3 Framtida arbete

Om tid hade funnits till fortsatt arbete med denna studie hade det varit mycket intressant att, som det var tänkt, få med ytterligare test på prototyperna såsom CPU- och minnesanvändning. Det hade också varit väldigt intressant att återupprepa batteritestet och denna gång samtidigt utföra exekveringstestet för att se hur tidsåtgång och batteriåtgång hängde ihop. Det hade även varit intressant att utföra ett nytt batteritest på åtminstone ytterligare ett sätt för att kontrollera att de resultat som PowerTutor presenterar verkligen är trovärdigt.

Om tillgång till fler testenheter hade funnits hade ett mer komplett test på varje specifik prototyp kunnat utföras vilket hade kunna ge ett mer komplett resultat. Om testenheterna dessutom även hade haft olika versioner på operativsystemet hade resultatet blivit ännu mer komplett.

Det skulle vara väldigt intressant att göra en likvärdig prototyp i alla de utvecklingsverktyg som hittats i denna studie och för alla de olika målplattformar som finns idag. Dessa prototyper skulle sedan kunna jämföras med de test som räknas upp i denna rapport samt eventuellt ytterligare test. Jämförelsen skulle kunna utföras både mellan prototyper anpassade för samma målplattform, men också mellan prototyper anpassade för olika målplattformar. Jämförelsen skulle även kunna innefatta ett användartest där användare får testa prototyperna och sedan svara på vad de tycker. Det hade även varit väldigt givande att jämföra alla prototyper mot en native app med samma egenskaper för att testa om någon av prototyperna fungerar lika bra som en native app eller kanske till och med bättre. Rimligtvis borde ju nativeappen fungera bättre men detta kan ju såklart påverkas av hur en utvecklare av ett multiplattsformutvecklingsverktyg har löst en viss funktion jämfört mot hur en utvecklare av en nativeapp väljer att implementera samma funktion.

Att utöka protyperna med mer avancerade funktioner skulle också vara intressant att undersöka. Växer exempelvis den prototyp som skapats i Smartface (som redan är väldigt stor) lika mycket som exempelvis den skapad i Xamarin, eller kan det vara så att Smartface redan inkluderar alla de bibliotek som kan tänkas behövas i en app och därför knappt växer alls,

medan Xamarin växer rejält, precis som den gjorde mellan version 1 och version 2, eftersom den nu måste inkludera ytterligare ett stort bibliotek.

De utvecklingsverktyg som valts ut att arbeta med för denna studie kommer troligtvis att vidareutvecklas och det skulle vara väldigt intressant att upprepa denna studie om ett antal år och då se hur mycket testresultaten skiljer sig. Testen skulle kunna jämföra både om det skiljer mycket mellan den gamla och nya versionen av en prototyp, men också om det skiljer lika mycket mellan utvecklingsverktygen som det gör idag.

Den app och de funktioner som använts i denna studie har varit väldigt simpla och det skulle vara väldigt intressant att se en tyngre app utvecklas och jämföras för de olika utvecklingsverktygen. Förslagsvis något som kräver mer prestanda och grafik såsom ett spel exempelvis. Vilka utvecklingsverktyg skulle klara av den uppgiften bäst? Är det ens möjligt att utveckla ett spel, som troligtvis kräver tillgång till en hel del hårdvara, och få det att fungera lika bra utifrån samma grund oavsett målplattform, eller kräver detta en native app?

En annan vinkling vore att göra en användbarhetsanalys på utvecklingsverktygen. Det vore intressant att se hur de förhåller sig till varandra på denna punkt och om det verkligen är tidsbesparande att utveckla i dem eller om vissa utav dem kanske är så omständliga att det istället vore mer lönsamt att utveckla en native app på traditionellt vis.

Om möjligheten fanns att ändra på något för de befintliga multiplattforms-utvecklingsverktygen så skulle det vara att göra dem, eller i alla fall en version av dem, open source. Detta skulle göra det tryggare för utvecklare att arbeta med dem eftersom de skulle kunna kontrollera om skadlig kod bifogades i en app. Det skulle även göra det möjligt för en utvecklare att göra något åt en bugg som hittas i verktyget, utan att behöva vänta på att verktygs-utvecklaren skulle åtgärda den. Men att göra något open source innebär också att avslöja sitt ”recept” med risk för att andra utvecklare kan utnyttja och bygga vidare på detta och tjäna pengar på något som de själva i grunden inte byggt. Man avsäger sig även möjligheten att själv tjäna pengar på sin idé, annat än genom donationer.

En långsiktig plan för ett fortsatt arbete vore att ta hjälp av denna och vidare studier och arbeta fram en standard som gör det möjligt att på bästa sätt med ett enskilt utvecklingsverktyg ta fram appar som fungerar på alla målplattformar. Men, man måste ha i åtanke att anledningen till att multiplattforms-utvecklingsverktygen har utvecklats är ju just för att en gemensam standard inte finns, utan utvecklarna för de olika målplattformarna har alla sin egen standard. Skulle dessa kombineras till en och samma skulle detta naturligtvis medföra både för- och nackdelar. Mycket skulle troligtvis förenklas och bli bättre, men olika är bra och framförallt när det kommer till företag så krävs det konkurrens för att utvecklingen ska gå framåt till förmån för konsumenterna.

Referenser

Andersson, M. & Andreasson, J. (2013) Smartphoneutveckling för flera plattformar. Examensarbete inom information- och programvarusystem, grundnivå Högskoleingenjör, kandidatexamen, 15 hp. KTH.

Android Central (2011) *Android 201: How (and when) to clear app cache or data*. Tillgänglig på Internet: http://www.androidcentral.com/android-201-how-and-when-clear-app-cache-or-data [Hämtad 14.06.02].

Android Developers (2014) *Using DDMS*. Tillgänglig på Internet: http://developer.android.com/tools/debugging/ddms.html [Hämtad 14.02.20].

appery.io DOCS (2014a) Building an app with Google Maps & geolocation. Tillgänglig på Internet: http://docs.appery.io/tutorials/building-a-mobile-app-with-google-map-and-geolocation/ [Hämtad 14.04.22].

appery.io DOCS (2014b) Using the camera component. Tillgänglig på Internet: http://docs.appery.io/tutorials/using-the-camera-component/ [Hämtad 14.04.25].

appery.io DOCS (2014c) Building a mobile RSS app. Tillgänglig på Internet: http://docs.appery.io/tutorials/building-a-mobile-rss-app/ [Hämtad 14.04.25].

Cai, J., Nerurkar, A. & Wu, M. (1998) Making benchmarks uncheatable. I: *Computer Performance and Dependability Symposium, 1998. IPDS '98. Proceedings. IEEE International* (s. 216-226). IEEE International Computer Performance and Dependability Symposium, 7-9 september, 1998, Durham, NC, USA.

Charland, A. & LeRoux, B. (2011) Mobile Application Development: Web vs. Native. *Queue, 9*(4), 1-9.

CODING TIP (2013) *Find prime numbers in javascript*. Tillgänglig på Internet: http://codingtip.blogspot.se/2013/06/find-prime-numbers-in-javascript.html [Hämtad 14.04.27].

Dalmasso, I., Datta, S.K., Bonnet, C. & Nikaein, N. (2013) Survey, comparison and evaluation of cross platform mobile application development tools. I: *The 9th International Wireless Communications & Mobile Computing Conference* (s. 323-328). Wireless Communications and Mobile Computing Conference (IWCMC), 2013 9th International, 1-5 juli, 2013, Cagliari, Sardinien, Italien.

Findahl, O. (2013) *Svenskarna och Internet 2013* (Version 1.0 2013). Göteborg: Göteborgstryckeriet.

Fransson, S. (2014) MoSync för multi-plattformsutveckling till smartphones. Examensarbete inom datavetenskap, kandidatexamen, 16 hp. Linköpings Universitet.

Google play (2014a) Endomondo Sports Tracker. Tillgänglig på Internet: https://play.google.com/store/apps/details?id=com.endomondo.android&hl=sv [Hämtad 2014.04.22].

Google play (2014b) Instagram. Tillgänglig på Internet: https://play.google.com/store/apps/details?id=com.instagram.android&hl=sv [Hämtad 2014.04.25].

Google play (2014c) Facebook. Tillgänglig på Internet: https://play.google.com/store/apps/details?id=com.facebook.katana&hl=sv [Hämtad 2014.04.26].

Google play (2014d) Twitter. Tillgänglig på Internet: https://play.google.com/store/apps/details?id=com.twitter.android&hl=sv [Hämtad 2014.04.26].

Google play (2014e) Usemon (Cpu Usage Monitor). Tillgänglig på Internet: https://play.google.com/store/apps/details?id=com.iattilagy.usemon&hl=sv [Hämtad 2014.05.21].

Google play (2014f) CPU Monitor. Tillgänglig på Internet: https://play.google.com/store/apps/details?id=com.bigbro.ProcessProfiler&hl=sv [Hämtad 2014.05.21].

Google play (2014g) PowerTutor. Tillgänglig på Internet: https://play.google.com/store/apps/details?id=edu.umich.PowerTutor&hl=sv [Hämtad 2014.02.18].

Hoccer (2010) *Measuring performance in the Android SDK*. Tillgänglig på Internet: http://hoccer.com/2010/09/measuring-performance-with-android/ [Hämtad 14.02.20].

Hui, N.M., Chieng, L.B., Ting, W.Y., Mohamed, H.H. & Arshad, M.R.H.M. (2013) Cross-platform mobile applications for android and iOS. I: *Wireless and Mobile Networking Conference (WMNC), 2013 6th Joint IFIP* (s. 1-4). 2013 6th Joint IFIP Wireless and Mobile Networking Conference (WMNC), 23-25 april, 2013, The Palm, Dubai, United Arab Emirates.

Huy, N.P. & Thanh, D.V. (2012) Developing Apps for Mobile Phones. I: *ICCCT2012, 2012 7th International Conference on Computing and Convergence Technology (ICCIT, ICEI and ICACT)* (s. 907-912). 2012 7th International Conference on Computing and Convergence Technology (ICCIT, ICEI and ICACT), 3-5 december, 2012, Seoul, Rep. of Korea.

Palmieri, M., Singh, I. & Cicchetti, A. (2012) Comparison of cross-platform mobile development tools. I: *2012 16th International Conference on Intelligence in Next Generation Networks (ICIN)* (s. 179-186). 2012 16th International Conference on Intelligence in Next Generation Networks (ICIN), 8-11 oktober, Berlin.

Pehrson, T. (2011) Utvärdering av multiplattformsutvecklingsverktyg för smarta mobiler. Examensarbete inom datavetenskap, datavetenskapliga programmet, inriktning IT-arkitekt, kandidatexamen, 15 hp. Högskolan i Gävle.

Sin, D., Lawson, E. & Kannoorpatti, K. (2012) Mobile web apps – the non-programmer's alternative to native applications. I: *Human System Interactions (HSI), 2012 5th International Conference on Human System Interactions* (s. 8-15). 2012 5th International Conference on Human System Interactions, 6-8 juni, 2012, Perth, West Australia.

Smartface (2014a) *SMF.UI.MapView.showUserLocation Property*. Tillgänglig på Internet: http://docs.smartface.io/?topic=html/P_SMF_UI_MapView_showUserLocation.htm [Hämtad 14.04.29].

Smartface (2014b) *Android Key Codes with App Studio*. Tillgänglig på Internet: http://developer.smartface.io/hc/en-us/articles/201429686-Android-Key-Codes-with-App-Studio [Hämtad 14.04.29].

Smartface (2014c) *SMF.Multimedia.startCamera Method*. Tillgänglig på Internet: http://docs.smartface.io/?topic=html/M_SMF_Multimedia_startCamera.htm [Hämtad 14.04.29].

Stackoverflow (2012) *How to find prime numbers?* Tillgänglig på Internet: http://stackoverflow.com/questions/11966520/how-to-find-prime-numbers [Hämtad 14.04.27].

Stackoverflow (2013a) *What's Android ADB shell 'dumpsys' tool and it's benefits?* Tillgänglig på Internet: http://stackoverflow.com/questions/11201659/whats-android-adb-shell-dumpsys-tool-and-its-benefits [Hämtad 14.02.20].

Stackoverflow (2013b) *How to measure time taken by a function to execute*. Tillgänglig på Internet: http://stackoverflow.com/questions/313893/how-to-measure-time-taken-by-a-function-to-execute [Hämtad 14.04.27].

Statista (2014) *Smartphone OS: global market share 2009-2013, by quarter*. Tillgänglig på Internet: http://www.statista.com/statistics/266136/global-market-share-held-by-smartphone-operating-systems/ [Hämtad 14.02.12].

Tillborg, C., Persson, L. & Bentlöv, H.E. (2012) Utveckling av hybrid app för flera plattformar. Examensarbete inom datavetenskap, grundnivå Webbprogrammerare, kandidatexamen, 15 hp. Linnéuniversitetet.

Wikipedia (2014) *Mobile application development*. Tillgänglig på Internet: http://en.wikipedia.org/wiki/Mobile_application_development [Hämtad 14.03.17].

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B & Wesslén, A. (2000) *Experimentation in software engineering an introduction*. New York: Springer Science+Business Media New York.

Xamarin (2015a) *Part 2 - Maps API*. Tillgänglig på Internet: http://developer.xamarin.com/guides/android/platform_features/maps_and_location/maps/part_2_-_maps_api/ [Hämtad 15.04.09].

Xamarin (2015b) *Location Services - Introduction to Location Services and the Fused Location Provider*. Tillgänglig på Internet: http://developer.xamarin.com/guides/android/platform_features/maps_and_location/location/ [Hämtad 15.04.14].

Xamarin (2015c) *Take a Picture and Save Using Camera App*. Tillgänglig på Internet: http://developer.xamarin.com/recipes/android/other_ux/camera_intent/take_a_picture_and_save_using_camera_app/ [Hämtad 15.04.15].

Xanthopoulos, S. & Xinogalos, S. (2013) A Comparative Analysis of Cross-platform Development Approaches for Mobile Applications. I: *BCI '13 Proceedings of the 6th Balkan Conference in Informatics* (s. 213-220). BCI 2013 6th Balkan Conference in Informatics, 19-21 september, 2013, Thessaloniki, Greece.

Zhang, L., Tiwana, B., Qian, Z., Wang, Z., Dick, R.P., Morley Mao, Z. & Yang, L. (2010) Accurate online power estimation and automatic battery behavior based power model generation for smartphones. I: *CODES/ISSS '10 Proceedings of the eighth IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis* (s. 105-114). ESWeek '10 Sixth Embedded Systems Week, 24-29 oktober, 2010, Scottsdale, AZ, USA.

Appendix A - Primtalsfunktionen i appery.io

```
1  var start = new Date().getTime();    //Get start time
2  var numbers = loopNumbers();         //Run function
3  var end = new Date().getTime();      //Get end time
4  var time = end - start;              //Calculate
5
6  //Present
7  Appery('resultFunction').text('Det tog '+time+'ms att hitta alla primtal i
8  spannet 0 till 100 000 ('+numbers+' st).');
9  Appery('resultFunction').show();
10
11 /*-------------------------------------------------------------*/
12
13 /*
14  * Loop through 0 - 100 000
15  */
16 function loopNumbers(){
17 var result = 0;
18
19   for(var i = 0; i < 100000; i++){
20     if(isPrime(i)){
21       result++;
22     }
23 }
24 return result;
25 }
26
27 /*
28  * Check if a number is prime
29  */
30 function isPrime(numb){
31
32 if(numb%2 === 0){
33     return false;
34   }
35
36 for(var i=3; i<= Math.sqrt(numb); i=i+2){
37   if(numb%i === 0){
38     return false;
39     }
40   }
41   return true;
```

Appendix B - Kamerafunktionen i Smartface

```
1  var photoUri = ""; //destination for captured photo
2  function startScreen_Self_OnShow(e){
3    Pages.startScreen.containerStart.photoPreview.visible = false;
4  }
5  function startScreen_buttonTakePhoto_OnPressed(e){
6    SMF.Multimedia.startCamera({
7      cameraType: 1,             //rear camera
8      resolution: 1,             //medium resolution
9      autoFocus: true,
10     onStart : function () {},  //do nothing
11     onCapture : function (e) {
12       photoUri = e.photoUri;
13       Pages.startScreen.containerStart.photoPreview.image = photoUri;
14       Pages.startScreen.containerStart.photoPreview.visible = true;
15     },
16     onCancel : function () {},     //do nothing
17     onFailure : function () {}     //do nothing
18   });
19 }
```

Appendix C - Primtalsfunktionen i Smartface

```
1  var start;       // start time
2  var numbers;     // amount of prime numbers
3  var end;         // end time
4  var time;        // time difference
5
6  function startScreen_buttonFunction_OnPressed(e){
7    start = new Date().getTime();    //Get start time
8    numbers = loopNumbers();         //Run function
9    end = new Date().getTime();      //Get end time
10   time = end - start;              //Calculate
11
12   //Present
13   Pages.startScreen.containerStart.resultFunction.text    =    "Det    tog
14   "+time+"ms att hitta alla primtal i spannet 0 till 100 000 ("+numbers+"
15   st).";
16   Pages.startScreen.containerStart.resultFunction.visible = true;
17 }
18
19 /*
20  * Loop through 0 - 100 000
21  */
22 function loopNumbers(){
23   var result = 0;
24     for(var i = 0; i < 100000; i++){
25       if(isPrime(i)){
26         result++;
27       }
28     }
29     return result;
30 }
31
32 /*
33  * Check if a number is prime
34  */
35 function isPrime(numb){
36   if(numb%2 === 0){
37     return false;
38   }
39   for(var i=3; i<= Math.sqrt(numb); i=i+2){
40     if(numb%i === 0){
41       return false;
42     }
43   }
44   return;
45 }
```

Appendix D - AndroidManifest.xml, Xamarin

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3  android:versionName="1.3" package="xamarin_a11emmjo.xamarin_a11emmjo"
4  android:versionCode="3">
5    <uses-sdk android:minSdkVersion="8" android:targetSdkVersion="8" />
6    <application android:label="xamarin_a11emmjo">
7      <!-- Put your Google Maps V2 API Key here. -->
8      <meta-data android:name="com.google.android.maps.v2.API_KEY"
9        android:value="AIzaSyDugIhcN7sgDH_MEdThDoFouMdDo11lLUI" />
10   </application>
11   <permission android:name="xamarin_a11emmjo.permission.MAPS_RECEIVE"
12     android:protectionLevel="signature" />
13   <uses-permission android:name="xamarin_a11emmjo.permission.MAPS_RECEIVE"
14     />
15   <!-- We need to be able to download map tiles and access Google Play
16   Services-->
17   <uses-permission android:name="android.permission.INTERNET" />
18   <!-- Allow the application to access Google web-based services. -->
19   <uses-permission
20 android:name="com.google.android.providers.gsf.permission.READ_GSERVICES"
21 />
22   <!-- Google Maps for Android v2 will cache map tiles on external storage
23   -->
24   <uses-permission
25     android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
26   <!-- Google Maps for Android v2 needs this permission so that it may
27   check the connection state as it must download data -->
28   <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"
29     />
30   <!-- These are optional, but recommended. They will allow Maps to use
31   the My Location provider. -->
32   <uses-permission
33     android:name="android.permission.ACCESS_COARSE_LOCATION" />
34   <!-- Google Maps for Android v2 requires OpenGL ES v2 -->
35   <uses-feature android:glEsVersion="0x00020000" android:required="true"
36     />
37   <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"
38     />
39   <!-- Used for camera function -->
40   <uses-permission android:name="android.permission.CAMERA" />
41 </manifest>
```

Appendix E - Main.axml, Xamarin

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3    android:orientation="vertical"
4    android:layout_width="fill_parent"
5    android:layout_height="fill_parent">
6    <Button
7      android:id="@+id/buttonMap"
8      android:layout_width="match_parent"
9      android:layout_height="wrap_content"
10     android:text="@string/buttonMap" />
11   <Button
12     android:text="@string/buttonPhoto"
13     android:layout_width="match_parent"
14     android:layout_height="wrap_content"
15     android:id="@+id/buttonPhoto" />
16   <Button
17     android:text="@string/buttonPrime"
18     android:layout_width="match_parent"
19     android:layout_height="wrap_content"
20     android:id="@+id/buttonPrime" />
21   <TextView
22     android:textAppearance="?android:attr/textAppearanceSmall"
23     android:layout_width="match_parent"
24     android:layout_height="wrap_content"
25     android:id="@+id/label" />
26   <ImageView
27     android:src="@android:drawable/ic_menu_gallery"
28     android:layout_width="match_parent"
29     android:layout_height="wrap_content"
30     android:id="@+id/imageView"
31     android:adjustViewBounds="true" />
32 <!--
33   <Button
34     android:id="@+id/myButton"
35     android:layout_width="match_parent"
36     android:layout_height="wrap_content"
37     android:text="@string/hello" />
38 -->
39 </LinearLayout>
```

Appendix F - MainActivity.cs, Xamarin

```
1  using System;
2  using System.Collections.Generic; //IList<>
3  using Android.App;
4  using Android.Content;
5  using Android.Content.PM; //ResolveInfo, PackageInfoFlags
6  using Android.Graphics; //Bitmap
7  using Android.OS;
8  using Android.Provider; //MediaStore
9  using Android.Runtime;
10 using Android.Views;
11 using Android.Widget;
12 using Java.IO;  //File
13 using Environment = Android.OS.Environment; //Environment
14 using Uri = Android.Net.Uri; //Uri
15
16 namespace xamarin_a11emmjo
17 {
18   public static class App
19   {
20     public static File _file;
21     public static File _dir;
22     public static Bitmap _bitmap;
23   }
24
25   [Activity (Label = "Xamarin", MainLauncher = true, Icon =
26 "@drawable/icon",
27     ScreenOrientation = ScreenOrientation.Portrait)]
28   public class MainActivity : Activity
29   {
30     //fields
31     private ImageView _imageView;
32 //   int count = 1;
33
34     //creates the activity
35     protected override void OnCreate (Bundle bundle)
36     {
37       base.OnCreate (bundle);
38
39       // Set our view from the "main" layout resource
40       SetContentView (Resource.Layout.Main);
41
42       //get map button from layout
43       Button buttonMap = FindViewById<Button> (Resource.Id.buttonMap);
44       //attach an event to it that starts the map-activity
45       buttonMap.Click += (sender, e) => {
46         StartActivity (typeof(MapActivity));
47       };
48
49       //get photo button from layout
50       Button buttonPhoto = FindViewById<Button> (Resource.Id.buttonPhoto);
```

```
51          //attach an event to the button that starts the camera
52          buttonPhoto.Click += TakeAPicture;
53
54          //get imageView from layout
55          _imageView = FindViewById<ImageView> (Resource.Id.imageView);
56
57          //get primefunction button from layout
58          Button buttonPrime = FindViewById<Button> (Resource.Id.buttonPrime);
59          //attach an event to it that runs the primefunction
60          buttonPrime.Click += RunPrimeFunction;
61
62  //      // Get our button from the layout resource,
63  //      // and attach an event to it
64  //      Button button = FindViewById<Button> (Resource.Id.myButton);
65  //
66  //      button.Click += delegate {
67  //        button.Text = string.Format ("{0} clicks!", count++);
68  //      };
69      }
70
71      //runs after the activity has been paused
72      protected override void OnResume ()
73      {
74          base.OnResume (); //always call the superclass first
75
76          //if a photo is taken, fill imageView with picture
77          if (App._bitmap != null)
78          {
79              _imageView.SetImageBitmap (App._bitmap);
80          }
81      }
82
83      //runs directly after the picture is taken
84      protected override void OnActivityResult (int requestCode, Result
85          resultCode, Intent data)
86      {
87          base.OnActivityResult (requestCode, resultCode, data); //always call
88              the superclass first
89
90          //make it available in the gallery
91          Intent mediaScanIntent = new Intent
92              (Intent.ActionMediaScannerScanFile);
93          Uri contentUri = Uri.FromFile (App._file);
94          mediaScanIntent.SetData (contentUri);
95          SendBroadcast (mediaScanIntent);
96
97          //resize and load image to private variable
98          int height = Resources.DisplayMetrics.HeightPixels;
99          int width = _imageView.Height;
100         App._bitmap = App._file.Path.LoadAndResizeBitmap (width, height);
101     }
102
103     //creates a directory if needed
```

```
104         private void CreateDirectoryForPictures ()
105         {
106           App._dir = new File (Environment.GetExternalStoragePublicDirectory
107             (Environment.DirectoryPictures), "xamarin_a11emmjo");
108
109           if (!App._dir.Exists())
110           {
111             App._dir.Mkdirs ();
112           }
113         }
114
115         //starts the camera and takes a picture
116         private void TakeAPicture(object sender, EventArgs eventArgs)
117         {
118           //create a directory where we can save the picture we're about to
119             take
120           CreateDirectoryForPictures ();
121
122           Intent intent = new Intent (MediaStore.ActionImageCapture);
123
124           App._file = new File (App._dir, String.Format ("newPhoto_{0}.jpg",
125             Guid.NewGuid ()));
126
127           intent.PutExtra (MediaStore.ExtraOutput, Uri.FromFile (App._file));
128
129           StartActivityForResult (intent, 0);
130         }
131
132         //runs the prime function and presents the result
133         private void RunPrimeFunction(object sender, EventArgs eventArgs)
134         {
135           long start = SystemClock.UptimeMillis (); //start timer
136           int numbers = LoopNumbers (); //run function
137           long end = SystemClock.UptimeMillis (); //stop timer
138           long time = end - start; //get timespan
139
140           //present result
141           TextView label = FindViewById<TextView> (Resource.Id.label);
142           label.Text = String.Format
143             ("Det tog {0}ms att hitta alla primtal i spannet 0 till 100 000
144               ({1} st)."
145               , time, numbers);
146         }
147
148         //loops through 0-100 000
149         private int LoopNumbers()
150         {
151           int result = 0;
152
153           for (int i = 0; i < 100000; i++) {
154             if (IsPrime(i)) {
155               result++;
156             }
```

```
157           }
158
159           return result;
160         }
161
162         //checks if a number is prime
163         private bool IsPrime(int number)
164         {
165           if (number%2 == 0){
166             return false;
167           }
168
169           for (int i = 3; i <= Math.Sqrt(number); i=i+2) {
170             if (number%i == 0) {
171               return false;
172             }
173           }
174
175           return true;
176         }
177       }
178   }
```

Appendix G - Map.axml, Xamarin

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3    android:orientation="vertical"
4    android:layout_width="fill_parent"
5    android:layout_height="fill_parent">
6    <Button
7      android:text="@string/buttonBack"
8      android:layout_width="match_parent"
9      android:layout_height="wrap_content"
10     android:id="@+id/buttonBack" />
11   <FrameLayout
12     android:minWidth="25px"
13     android:minHeight="25px"
14     android:layout_width="match_parent"
15     android:layout_height="match_parent"
16     android:id="@+id/map" />
17 </LinearLayout>
```

Appendix H - MapActivity.cs, Xamarin

```
1  using System;
2  using Android.App;
3  using Android.Content;
4  using Android.Gms.Common; //used to support fused location provider
5  using Android.Gms.Location; //used to support fused location provider
6  using Android.Gms.Maps; //used to support map
7  using Android.Gms.Maps.Model; //used to suport map (LatLng and MarkerOptions)
8  using Android.OS;
9  using Android.Runtime;
10 using Android.Support.V4.App; //used to support map
11 using Android.Views;
12 using Android.Widget;
13
14 namespace xamarin_a11emmjo
15 {
16    [Activity (Label = "Här är du")]
17    public class MapActivity : FragmentActivity,
18                    IGooglePlayServicesClientConnectionCallbacks,
19                    IGooglePlayServicesClientOnConnectionFailedListener
20    {
21       //fields
22       private static readonly LatLng _location = new LatLng (35.44863,
23          24.20070);
24       private LocationClient _logClient;
25       private GoogleMap _map;
26       private SupportMapFragment _mapFragment;
27
28       //creates the activity
29       protected override void OnCreate (Bundle bundle)
30       {
31          base.OnCreate (bundle);
32
33          SetContentView (Resource.Layout.Map); //create view
34          _logClient = new LocationClient (this, this, this); //get location
35             client
36          InitMapFragMent (); //initiate map
37
38          //get back button from layout
39          Button buttonBack = FindViewById<Button> (Resource.Id.buttonBack);
40          //attach an event to it that ends this activity and returns to main
41          buttonBack.Click += (sender, e) => {
42             this.Finish ();
43          };
44       }
45
46       //runs after oncreate but before view is shown
47       protected override void OnStart ()
48       {
49          base.OnStart (); //always call the superclass first
50
51          _logClient.Connect (); //connect location client
```

```
52        }
53
54        //finish this activity when hardware back button is pressed
55        public override void OnBackPressed ()
56        {
57          base.OnBackPressed (); //always call the superclass first
58
59          this.Finish ();
60        }
61
62        //instantiate the map
63        private void InitMapFragMent ()
64        {
65          _mapFragment = SupportFragmentManager.FindFragmentByTag ("map") as
66            SupportMapFragment;
67
68          if(_mapFragment == null)
69          {
70            GoogleMapOptions mapOptions = new GoogleMapOptions()
71              .InvokeMapType(GoogleMap.MapTypeNormal)
72              .InvokeZoomControlsEnabled(true)
73              .InvokeCompassEnabled(true);
74
75            FragmentTransaction fragTx =
76              SupportFragmentManager.BeginTransaction();
77            _mapFragment = SupportMapFragment.NewInstance(mapOptions);
78            fragTx.Add(Resource.Id.map, _mapFragment, "map");
79            fragTx.Commit();
80          }
81        }
82
83        //populate the map
84        private void UpdateMapFragMent ()
85        {
86          if (_map == null)
87          {
88            _map = _mapFragment.Map;
89            if (_map != null)
90            {
91
92              MarkerOptions marker = new MarkerOptions();
93              marker.SetPosition(_location);
94              marker.SetTitle("Här är du :)");
95              _map.AddMarker(marker);
96
97              // We create an instance of CameraUpdate, and move the map to
98                it.
99              CameraUpdate cameraUpdate =
100               CameraUpdateFactory.NewLatLngZoom(_location, 11);
101             _map.MoveCamera(cameraUpdate);
102           }
103         }
104       }
```

```
105
106         //get last location and populate map with it
107         public void OnConnected (Bundle bundle)
108         {
109           if (_logClient.LastLocation != null)
110           {
111             //update location
112             _location.Latitude = _logClient.LastLocation.Latitude;
113             _location.Longitude = _logClient.LastLocation.Longitude;
114
115             //update map
116             UpdateMapFragMent ();
117           }
118         }
119
120         public void OnDisconnected ()
121         {
122           throw new NotImplementedException ();
123         }
124
125         public void OnConnectionFailed (ConnectionResult result)
126         {
127           throw new NotImplementedException ();
128         }
129       }
130   }
```

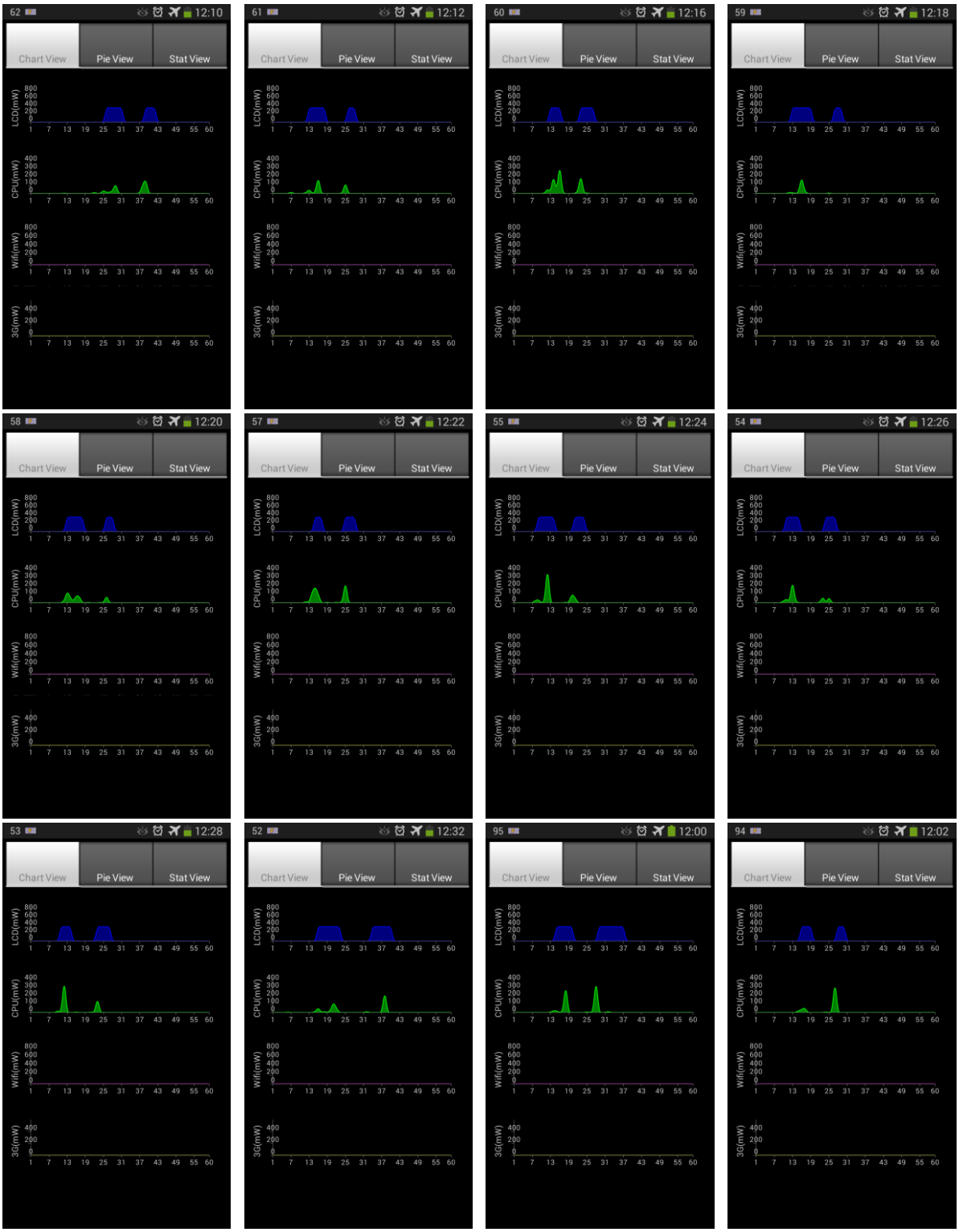
Appendix I - Strings.xml, Xamarin

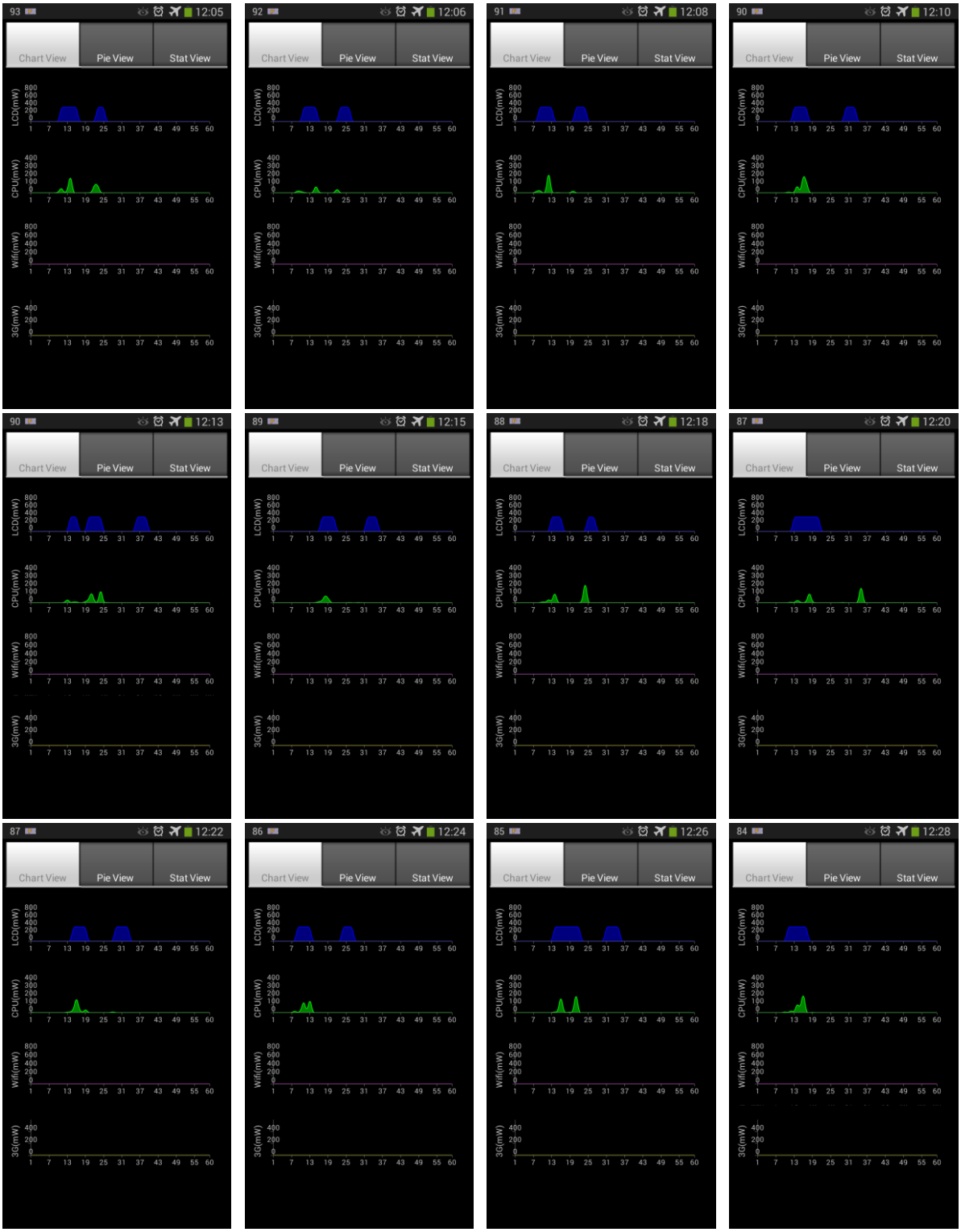
```
1  <?xml version="1.0" encoding="utf-8"?>
2  <resources>
3  <!--
4    <string name="hello">Hello World, Click Me!</string>
5  -->
6    <string name="app_name">xamarin_a11emmjo</string>
7    <string name="buttonMap">Var är jag?</string>
8    <string name="buttonBack">Tillbaka</string>
9    <string name="buttonPhoto">Ta ett kort</string>
10   <string name="buttonPrime">Kör funktion</string>
11 </resources>
```

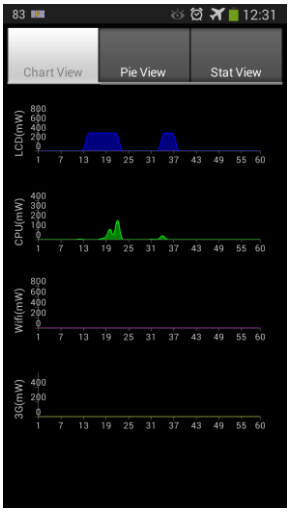
Appendix J - BitmapHelpers.cs, Xamarin

```
1  using System.IO;
2  using Android.Graphics;
3
4  namespace xamarin_a11emmjo
5  {
6    public static class BitmapHelpers
7    {
8      public static Bitmap LoadAndResizeBitmap(this string fileName, int
9        width, int height)
10     {
11       // First we get the the dimensions of the file on disk
12       BitmapFactory.Options options = new BitmapFactory.Options {
13         InJustDecodeBounds = true };
14       BitmapFactory.DecodeFile(fileName, options);
15
16       // Next we calculate the ratio that we need to resize the image by
17       // in order to fit the requested dimensions.
18       int outHeight = options.OutHeight;
19       int outWidth = options.OutWidth;
20       int inSampleSize = 1;
21
22       if (outHeight > height || outWidth > width)
23       {
24         inSampleSize = outWidth > outHeight
25           ? outHeight / height
26           : outWidth / width;
27       }
28
29       // Now we will load the image and have BitmapFactory resize it for
30         us.
31       options.InSampleSize = inSampleSize;
32       options.InJustDecodeBounds = false;
33       Bitmap resizedBitmap = BitmapFactory.DecodeFile(fileName, options);
34       return resizedBitmap;
35     }
36   }
37 }
```

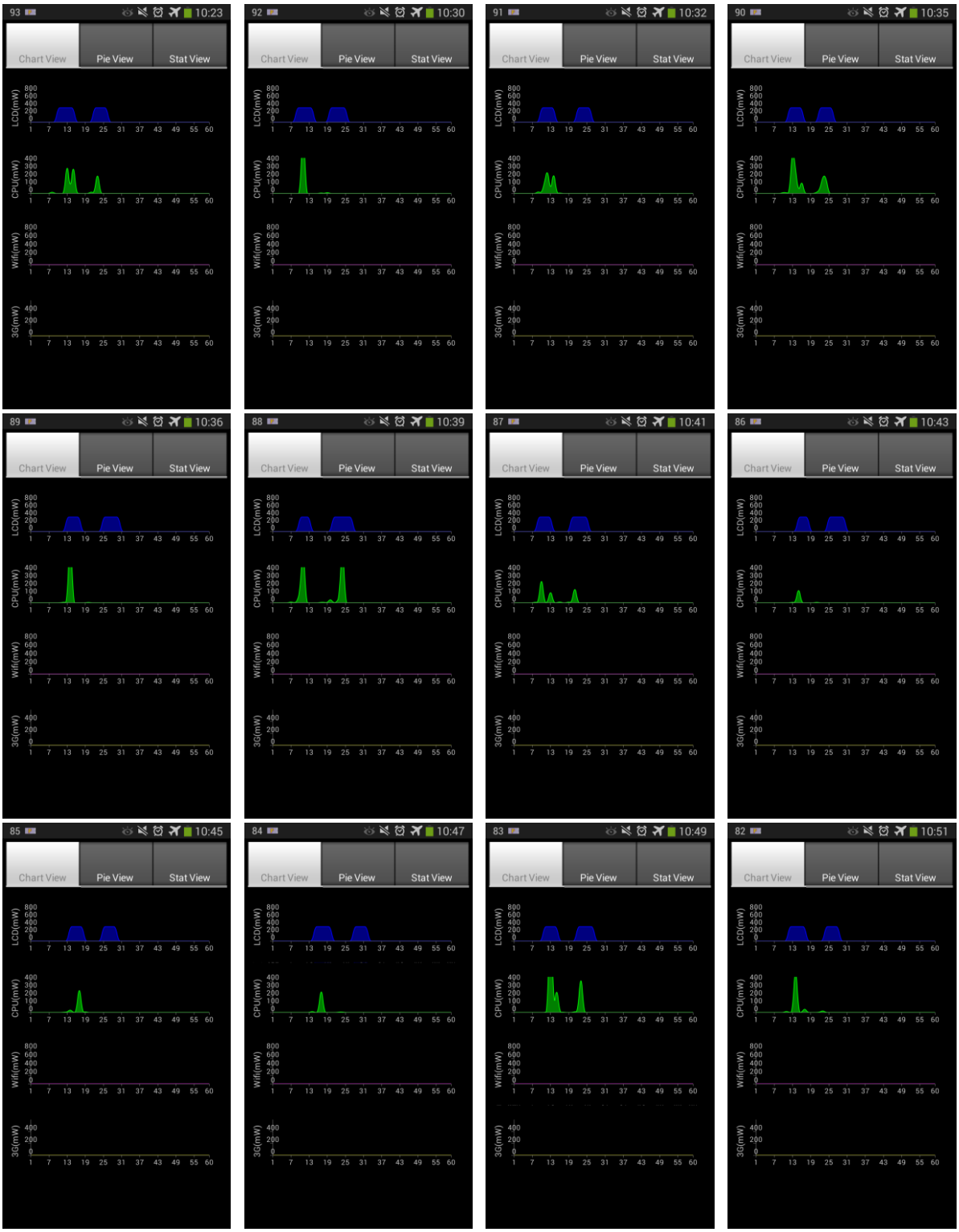
Appendix K - Printscreens från Power Tutor för appery.io

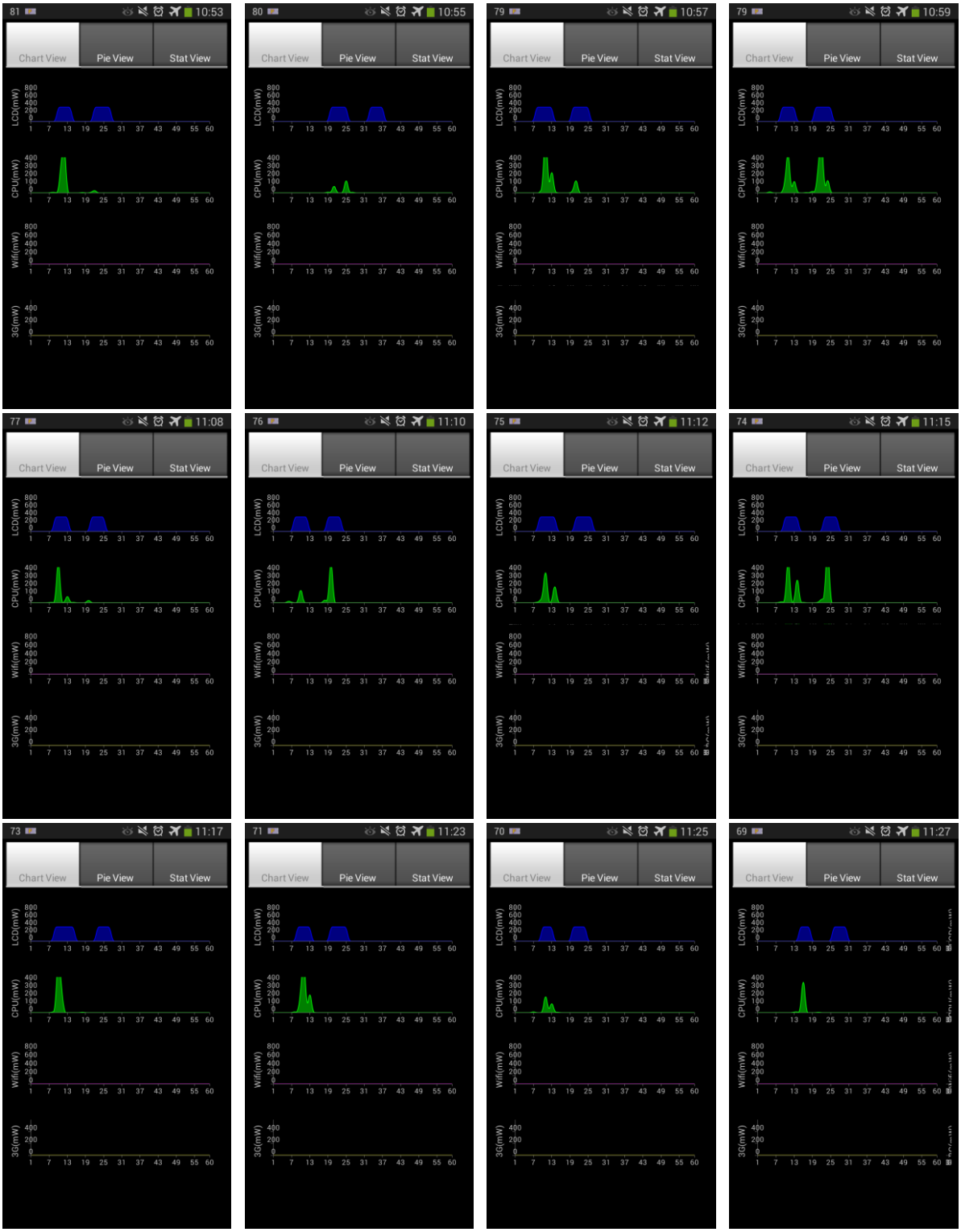


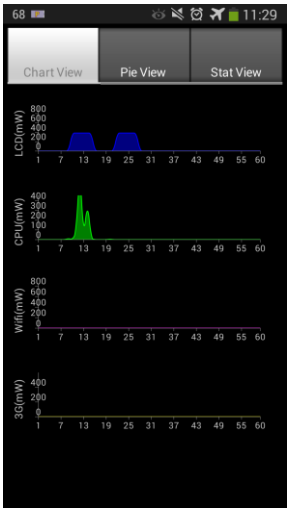

93 12:05
92 12:06
91 12:08
90 12:10
90 12:13
89 12:15
88 12:18
87 12:20
87 12:22
86 12:24
85 12:26
84 12:28
Chart View
Pie View
Stat View
LCD(mW)
CPU(mW)
Wifi(mW)
3G(mW)


83
12:31
Chart View
Pie View
Stat View
LCD(mW)
800
600
400
200
0
1 7 13 19 25 31 37 43 49 55 60
CPU(mW)
400
300
200
100
0
1 7 13 19 25 31 37 43 49 55 60
Wifi(mW)
800
600
400
200
0
1 7 13 19 25 31 37 43 49 55 60
3G(mW)
400
200
0
1 7 13 19 25 31 37 43 49 55 60

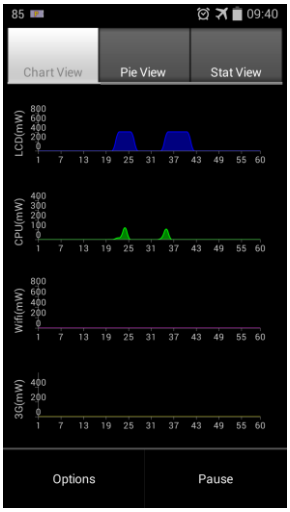
Appendix L - Printscreens från Power Tutor för Smartface

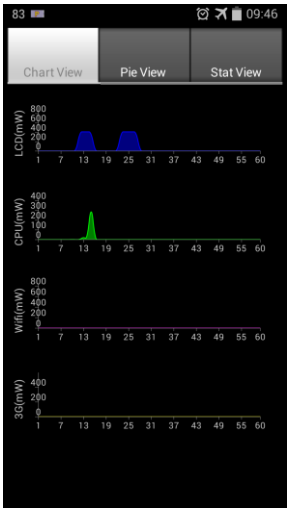


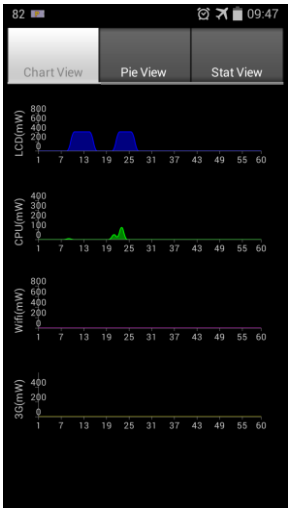

81 10:53
80 10:55
79 10:57
79 10:59
77 11:08
76 11:10
75 11:12
74 11:15
73 11:17
71 11:23
70 11:25
69 11:27
Chart View
Pie View
Stat View
LCD(mW)
CPU(mW)
Wifi(mW)
3G(mW)

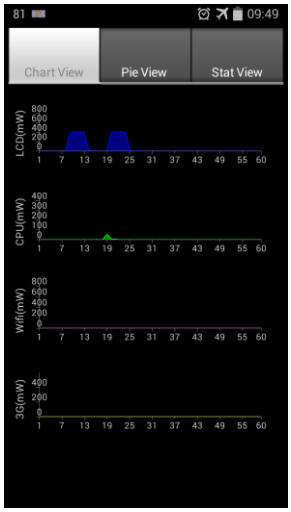

68
11:29
Chart View
Pie View
Stat View
LCD(mW)
800
600
400
200
0
1 7 13 19 25 31 37 43 49 55 60
CPU(mW)
400
300
200
100
0
1 7 13 19 25 31 37 43 49 55 60
Wifi(mW)
800
600
400
200
0
1 7 13 19 25 31 37 43 49 55 60
3G(mW)
400
200
0
1 7 13 19 25 31 37 43 49 55 60

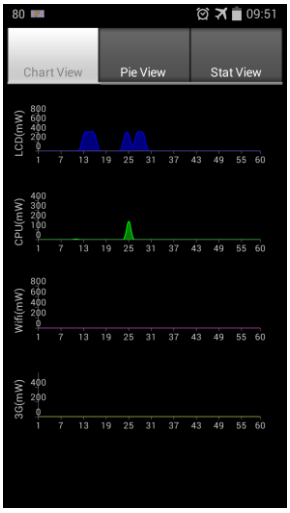
Appendix M - Printscreens från Power Tutor för Xamarin

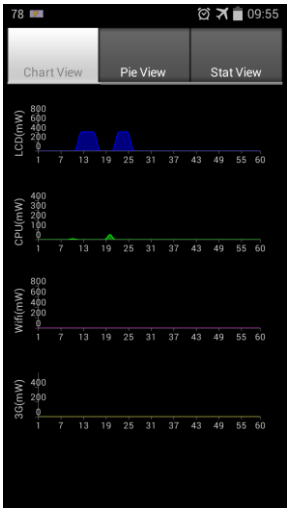


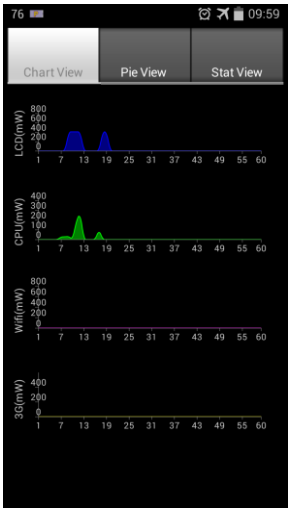


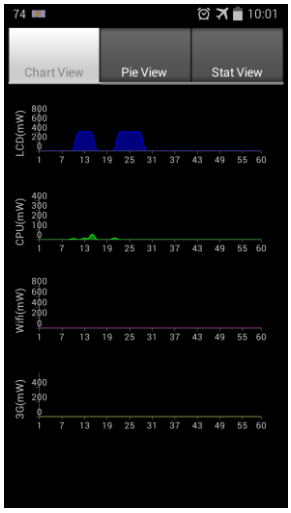


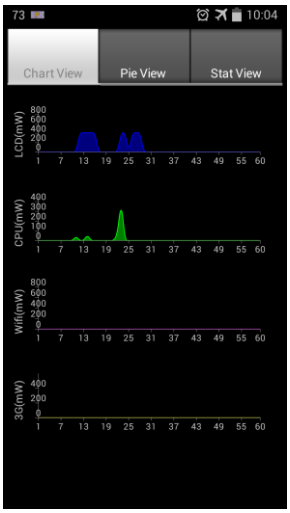


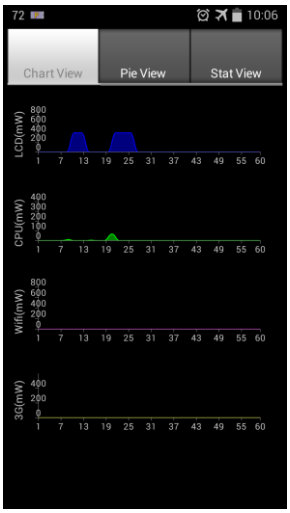


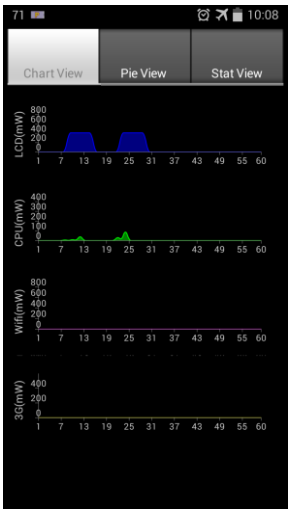


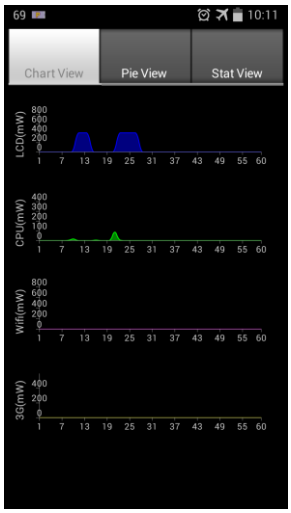


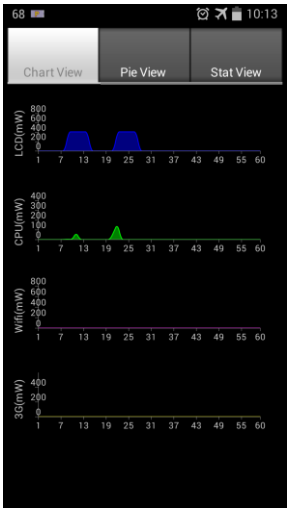


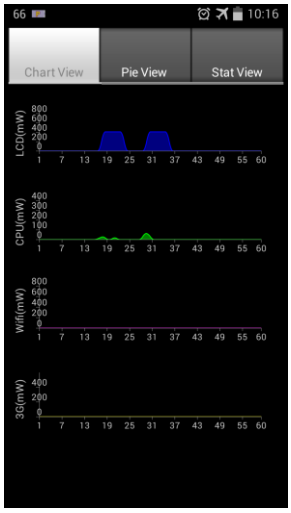


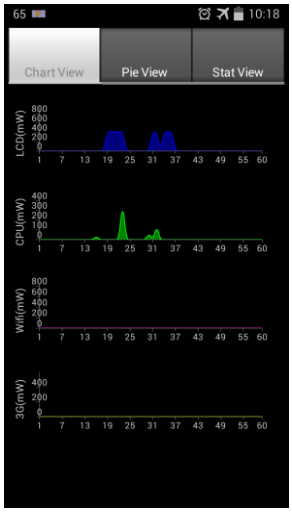


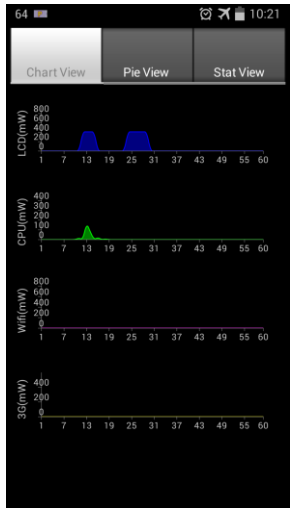


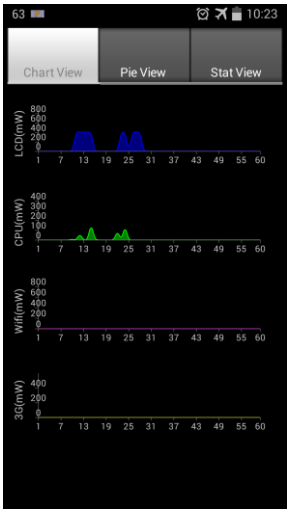


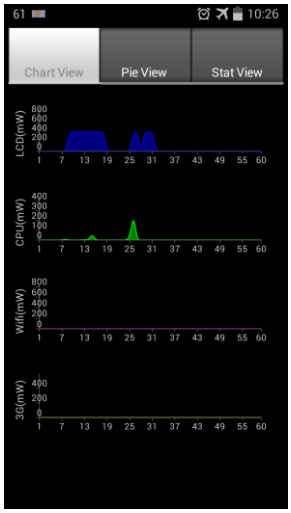

68
10:13
Chart View
Pie View
Stat View
LCD(mW)
CPU(mW)
Wifi(mW)
3G(mW)

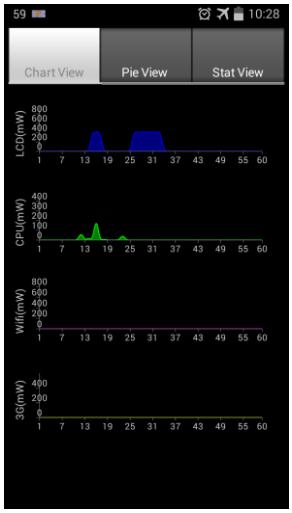

66
10:16
Chart View
Pie View
Stat View
LCD(mW)
CPU(mW)
Wifi(mW)
3G(mW)

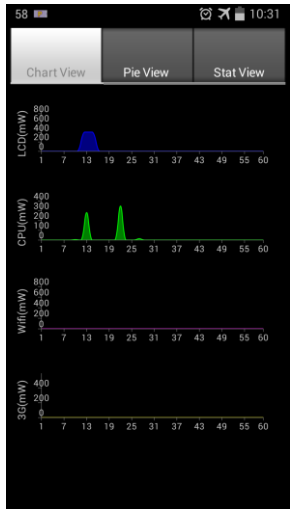

65
10:18
Chart View
Pie View
Stat View
LCD(mW)
CPU(mW)
Wifi(mW)
3G(mW)

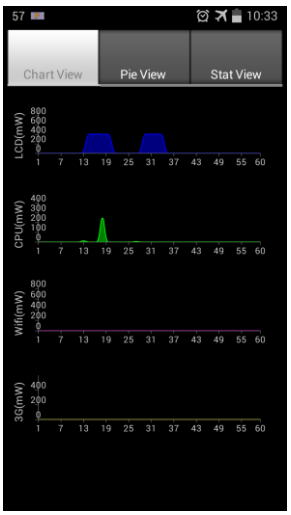

64
10:21
Chart View
Pie View
Stat View
LCD(mW)
CPU(mW)
Wifi(mW)
3G(mW)

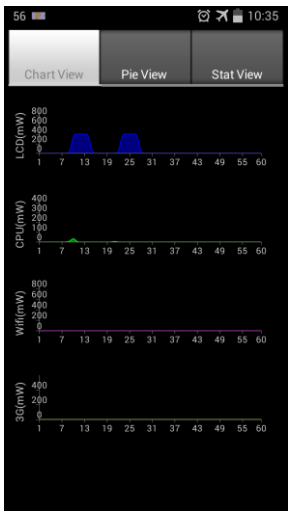

63
10:23
Chart View
Pie View
Stat View
LCD(mW)
CPU(mW)
Wifi(mW)
3G(mW)

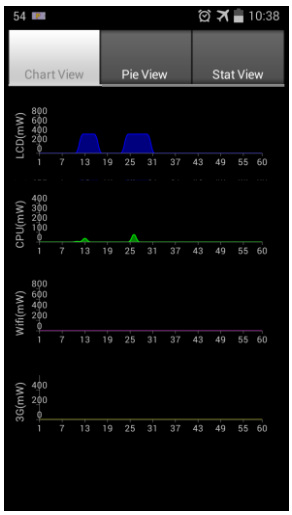

61
10:26
Chart View
Pie View
Stat View
LCD(mW)
CPU(mW)
Wifi(mW)
3G(mW)

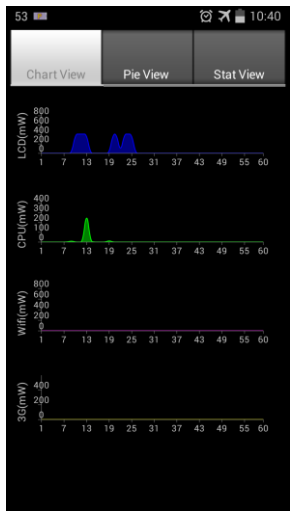

59
10:28
Chart View
Pie View
Stat View
LCD(mW)
CPU(mW)
Wifi(mW)
3G(mW)

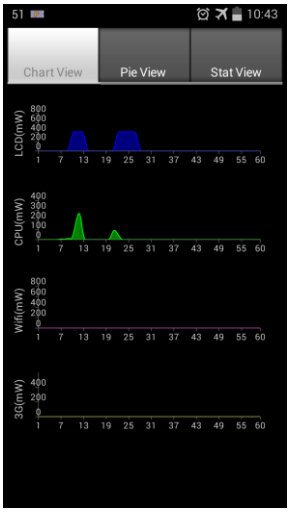

58
10:31
Chart View
Pie View
Stat View
LCD(mW)
CPU(mW)
Wifi(mW)
3G(mW)


57
10:33
Chart View
Pie View
Stat View
LCD(mW)
CPU(mW)
Wifi(mW)
3G(mW)


56
10:35
Chart View
Pie View
Stat View
LCD(mW)
CPU(mW)
Wifi(mW)
3G(mW)


54
10:38
Chart View
Pie View
Stat View
LCD(mW)
CPU(mW)
Wifi(mW)
3G(mW)


53
10:40
Chart View
Pie View
Stat View
LCD(mW)
CPU(mW)
Wifi(mW)
3G(mW)


51
10:43
Chart View
Pie View
Stat View
LCD(mW)
800
600
400
200
0
1 7 13 19 25 31 37 43 49 55 60
CPU(mW)
400
300
200
100
0
1 7 13 19 25 31 37 43 49 55 60
Wifi(mW)
800
600
400
200
0
1 7 13 19 25 31 37 43 49 55 60
3G(mW)
400
200
0
1 7 13 19 25 31 37 43 49 55 60