

CHALMERS



Ramverk för webb-baserade mobil-applikationer

Kandidatarbete inom Data- och informationsteknik

MATTIAS APPELGREN

MARKUS BERGET

ANTON MALMQUIST

ANTON WERNVIK

Institutionen för Data- och Informationsteknik
CHALMERS TEKNISKA HÖGSKOLA
Göteborg, Sverige 2012
Kandidatarbete 2012:21

Sammanfattning

Smartphone-marknaden är idag en mångmiljardindustri; för utvecklare har det skapats en helt ny marknad för mobil-applikationer. Ett problem med marknaden idag är den fragmentering som finns mellan olika mobila operativsystem, detta är ett problem då utvecklare behöver utveckla flera separata applikationer för olika mobila operativsystem.

Projektet som avhandlas i rapporten syftar till att undersöka om detta problem kan lösas med hjälp av utvecklingen av ett ramverk. Ett sådant ramverk skulle möjliggöra för utvecklare att utveckla sina applikationer med vanliga webbtekniker men samtidigt få tillgång till de funktioner som plattformsspecifika applikationer har tillgång till. Dessa kan vara vibration, kamera, kompass etc.

Grundidén var ett ramverk med två komponenter. Den ena delen skriven i Javascript som webb-applikationerna implementerar. Den andra delen av ramverket är en applikation utvecklad för det specifika mobila operativsystemet, detta för att möjliggöra tillgång till de hårdvarunära funktionerna hos enheten. Dessa två delar kommunicerar sedan via Javascript.

IOS och Android undersöktes och det visade sig möjligt att genomföra projektet på båda plattformarna. På grund av tidsramarna för projektet valdes Android som plattform att bygga vidare på. Utvecklingen skedde på ett agilt sätt med metoden *Crystal Clear*.

Det slutgiltiga ramverket tillhandahåller sju huvudsakliga komponenter till webb-utvecklare, och det producerades även en webb-applikation som implementerar alla dessa funktioner. Dessutom har tester på riktig hårdvara har genomförts framgångsrikt.

Detta projekt visar på de möjligheter det finns att integrera webb-applikationer med mer hårdvarunära funktioner. Begränsningarna ligger främst i att det behöver skapas en applikation för varje mobilt operativsystem, och eftersom denna teknik skulle göra många av de distributionsmarknader som finns redundanta så ligger det inte i operativsystemsägarnas intresse, då de även förlorar kontrollen över vilka applikationer som finns tillgängliga till deras operativsystem.

Abstract

In the last few years, the market for smartphone applications has grown vastly. It has now become a multi billion dollar industry. Despite this, there exists a few problems with the model upon which the market is based.

For developers, one of the biggest problems with the current market is the number of platforms. To reach as many users as possible a developer has to implement an application each for several platforms.

In order to find solutions to this problem, a software project was undertaken to bring native functionality of the device to the web. Examples of such functionalities are compass, vibration and camera.

The idea explored was a framework consisting of two parts, namely a Javascript framework and a native application for the smartphone. The framework would then be imported to a web application and give it access to native functionality on smartphones. To expose these functionalities to the framework, it would be necessary to write and deploy a native application that could handle incoming Javascript calls, as well as return JSON-objects to the web application.

During the project, IOS and Android were examined to evaluate if the different operating systems could accommodate this kind of framework. The examination resulted in deeming both platforms fully viable. Due to the time restraint on the project, it was decided that only an application for Android was to be developed. This development was performed with an agile method called *Crystal Clear*.

The project resulted in a working Javascript framework and Android application counterpart. The final framework provides seven different functions for web developers, and to demonstrate this a web application was developed with support for the framework.

This project proves the possibility of using web technologies for producing applications with native functionality for different mobile platforms simultaneously. The limitations of this solution lie in the fact that a native application for each supported platform must be developed before the framework can be usable on said platform. These limitations could be circumvented by implementing the technologies in the native browser for the different platforms; but due to the vested interest each platform owner has in keeping control over the market of applications on that platform, this change is less than likely.

Terminologi

API *Application Programming Interface*, ett regelverk som syftar till att möjliggöra för flera programvarukomponenter att kommunicera med varandra.

asynkron Ett asynkront meddelande är ett meddelande som skickas utan att vänta på svar [1]. Jfr. synkron.

enhet Ett vidare begrepp som används för att beskriva generella mobila enheter som kan köra tredjeparts-programvara. Inkluderar dock ej bärbara datorer.

GPS *Global Positioning System* ett allmänt tillgängligt system för satellitnavigering.

look and feel Upplevelsen och funktionen hos ett grafiskt användargränssnitt.

parsning Att tolka text för att identifiera kommandon och eventuell tillhörande data.

plattformsspecifik applikation Från engelskans *native application*, en applikation som är byggd för en specifik plattform.

proof of concept Realisering av en idé eller metod för att demonstrera att den är genomförbar.

ramverk Återanvändbar kod som kan användas och byggas ut för att tillhandahålla funktionalitet [1].

smartphone Telefon med möjlighet att installera programvara från tredje part.

synkron Samtidigt, vid kommunikation mellan två parter krävs båda parter uppmärksamhet under utbytet [1].

testbarhet I den utsträckning till vilket ett objektiv och genomförbart test kan vara utformat för att bestämma huruvida ett krav är uppfyllt [1].

webb-applikation En applikation som är skriven med webbtekniker såsom HTML, Javascript och CSS.

Innehåll

1	Inledning	1
1.1	Bakgrund	1
1.2	Syfte	2
1.3	Problem- och uppgiftsanalys	2
1.3.1	Ramverket	2
1.3.2	Mobil-applikation	2
1.3.3	Webb-applikation	3
1.4	Avgränsningar	3
2	Metod	4
2.1	Utveckling	4
2.2	Designmönster	4
2.3	Undersökningar	5
2.4	Iterationer	5
3	Omvärldsanalys	7
3.1	Befintliga tekniker	8
3.1.1	Phonegap	9
3.1.2	Appcelerator	9
3.1.3	Mozilla Boot 2 Gecko	9
3.1.4	HTML 5	9
3.1.5	Look and feel-imiterande verktyg	10
4	Analys	11
4.1	Utveckling av grafiska användargränssnitt i Android	12
4.2	Beroendeinjektion	13
4.3	Säkerhet	13
4.4	Testning av applikationen och ramverket	14
4.5	Dokumentation	15
4.6	Verktyg, mjukvarubibliotek och andra hjälpmedel	16
4.6.1	Versionshanterings-verktyg	16
4.6.2	Utvecklingsmiljö	17
4.6.3	Programmeringsspråk för ramverket	17
4.6.4	Datautbytningsformatet JSON	17
4.6.5	Gson	17
4.6.6	Google Guice och RoboGuice	17
4.6.7	Testverktyg för att simulera sensorer på Android	18
5	Kravspecifikation	19
5.1	Ramverkets funktioner	20
5.2	Android-applikationen	22
5.3	Icke funktionella krav	22
6	Design	23
6.1	Design av Ramverket	24
6.1.1	Generella designprinciper	24
6.1.2	Inkapsling av metoder	24
6.1.3	Parametrar	24

6.1.4	Första steget i valideringen	25
6.1.5	Retur av data	25
6.1.6	Asynkrona anrop	25
6.2	Design av Android-applikationen	26
6.2.1	Lyssnare	27
7	Implementering	28
7.1	Android-applikationen	28
7.1.1	Filuppladdning	28
7.2	Ramverket	28
7.3	Webb-applikationen	29
8	Diskussion	30
9	Slutsats	31
	Referenser	35
A	Use cases	36
A.1	Gå in i en webb-applikation	36
A.2	Gå ur en webb-applikation	36
A.3	Hämta godtyckliga filer	36
A.4	Ladda upp godtyckliga filer	37
A.5	Lägga till händelser i kalendern	37
A.6	Lyssna på knapptryckningar	38
A.7	Ramverket begär åtkomst till funktion på telefonen	38
B	Dokumentation	40
B.1	Example	40
B.2	Compass	41
B.3	Geolocation	42
B.4	Contacts	44
B.5	Keypress	45
B.6	Vibration	46
B.7	Calendar	46
B.8	Flashlight	47

1 Inledning

I denna rapport kommer att avhandlas ett mjukvaruutvecklingsprojekt utfört av författarna till rapporten. Projektet kretsade kring marknaden för smartphone-applikationer, vilka problem som uppstår för utvecklare i det rådande läget samt ett försök att hitta tekniska lösningar på några av dessa problem.

Rapporten fokuserar på de steg som projektet har genomgått, med omvärldsanalys, design och genomförande som fokuspunkter. Tanken med rapporten är att läsaren skall kunna följa de beslut som tagits genom processen samt få en bakgrund för de resultat som presenteras i rapport och appendix.

Under utvecklingsarbetet har arbetsnamnet för den slutgiltiga produkten varit *Cratis*, från latinets flätverk. Namnet är valt på grund av att de olika delarna hos projektet sammanflätade bildar en koherent produkt.

1.1 Bakgrund

Ett problem som utvecklare av applikationer till mobila operativsystem har idag är mängden av plattformar att utveckla till. För att nå så många användare som möjligt med sin applikation krävs det att utvecklaren vänder sig till flera plattformar och använder deras respektive ekosystem; det vill säga operativsystem, distributionskanal och utvecklingsverktyg.

Inte nog med att de olika ekosystemen skiljer sig från varandra, det finns även hårdvaruskillnader mellan olika enheter. Till exempel kan skärmupplösning, fysiska knappar, minnesmängd och processorer vara olika.

Två av de mest använda smartphone-operativsystemens ägare, Google (Android [2]) och Apple (IOS [3]), har också egna krav på funktionalitet och utseende hos applikationerna. Detta för att kunna försäkra kvaliteten och ge användaren en bra upplevelse [4, 5]. Genom att sedan också kontrollera distributionen av applikationerna genom applikationsbibliotek som Google Play och Apple App Store så kan de sedan ta en del av inkomsten av försäljningen. Ingen av de stora aktörerna har hittills tagit några större steg för att öka kompatibiliteten hos applikationer mellan operativsystemen.

När Apple släppte den första Iphonen var alla tredjepartsapplikationer tänkta att skrivas som webb-applikationer [6]. Men med efterkommande telefoner övergavs den ambitionen [7]. Ett skäl till detta kan vara att prestandaskillnaden mellan plattformsspecifika applikationer och webb-applikationer var för stor. Med dagens teknik kan dock Javascript [8] köras upp till tre gånger snabbare än vad det kunde 2009 [9] och detta ger en helt nya möjligheter för webb-baserade applikationer.

Denna ökning i prestanda möjliggör för utvecklare att byta från att utveckla applikationer för operativsystemen till att utveckla rena webb-applikationer som är anpassade för mobila enheter. Webb-applikationer kan då köras på de flesta mobila enheterna och utvecklarna behöver bara lägga tid på att utveckla en applikation istället för en till varje operativsystem.

En av bristerna hos webb-applikationer i dagsläget är att de inte når samma prestanda som plattformsspecifika applikationer och då inte är lämpliga för till exempel grafikintensiva tillämpningar. Webb-applikationerna har heller inte tillgång till de olika hårdvarufunktioner som de flesta mobila enheter har, som till exempel kompass, vibration och GPS-position.

1.2 Syfte

Syftet med projektet är att framställa ett ramverk som gör det möjligt för utvecklare att använda sig av enheters inbyggda funktioner i en webb-applikation.

1.3 Problem- och uppgiftsanalys

Under projektets gång utvecklades flera delar samtidigt för att möjliggöra meningsfull testning av de funktioner som ramverket inkluderar. Dessa delar presenteras i sin korthet härnäst.

1.3.1 Ramverket

Den största utmaningen med ett projekt som det som behandlas här är att utforma ramverket på bästa sätt för att en webb-utvecklare ska kunna utnyttja funktionerna hos de vanligaste mobila operativsystemen. Det krävs en god insikt i flera operativsystem för att komma fram till en lösning som skall fungera bra för alla dessa system. Arbetet med ramverket kom därför att involvera undersökningar för att ta reda på hur de olika systemen kan kommunicera med en webb-applikation. Arbetet kom också att bestå i att ta reda på hur man kan faktorisera dessa systems beteenden för att skapa ett ramverk som går att realisera på flera plattformar.

Tanken var att tillgodose utvecklare av webb-applikationer med ett Javascript-bibliotek utformat efter ramverkets specifikationer. Detta har krävt en god dokumentation för att det ska vara lätt för utvecklarna att ta till sig och använda ramverket. Om ramverket är för svårt att sätta sig in i finns det en stor risk för att det inte används.

1.3.2 Mobil-applikation

Utvecklingen av mobil-applikationen hade i stort sett två huvudspår; dels ett grafiskt gränssnitt för användarinteraktion och dels implementeringen av ramverket. Den viktigaste delen i det här projektet var implementeringen av ramverket. Det är där som kommunikationen mellan webb-applikationer och telefonens funktioner är implementerade. Denna implementering skedde i symbios med utformandet av ramverket och en stor utmaning låg i att inte göra ramverket alltför plattformspecifikt.

Det var viktigt att det grafiska gränssnittet på ett intuitivt sätt möjliggör för användare att spara webb-applikationer och hantera diverse inställningar.

1.3.3 Webb-applikation

Det krävdes även utveckling av en webb-applikation som ramverket kunde testas på. Webb-applikationen behövde använda sig av alla funktioner i ramverket för att inte begränsa testmöjligheterna för ramverket och mobil-applikationen.

Webb-applikationen ansågs däremot inte behöva något avancerat grafiskt gränssnitt eftersom projektets mål var att skapa ett ramverk och en mobil-applikation som använder sig av ramverket. Meningen är att de som utvecklar vanliga webb-applikationer idag skall använda sig av ramverket.

1.4 Avgränsningar

Mobil-applikationen ska ej kunna visa sidor som inte implementerar ramverket. Under detta kandidatarbete har arbetet fokuserats på enbart en mobil-plattform, även om viss undersökning har skett på andra plattformar.

2 Metod

Eftersom en stor del av projektet var utvecklingen av en mjukvara så var det viktigt att tidigt stipulera en metodik för denna process. Därför diskuteras här denna metodik.

2.1 Utveckling

Projektets tre delar utvecklades parallellt, detta eftersom delarna är avhängiga varandra. Dessa delar är applikationen till enheterna, ramverket som webb-applikationerna implementerar och en webb-applikation som skapades för att kunna testa och visa på funktionalitet hos ramverket.

Arbetet skedde iterativt, fokus låg på att alltid ha en körbar mobil-applikation som under projektets gång fick mer och mer funktionalitet. För varje funktionalitet som lades till i applikationen implementerades motsvarande funktionalitet även i ramverket. Under utvecklingens gång testades ny funktionalitet och det utfördes även användartester.

Rapporten [10] studerades för att få en övergripande skildring av för- och nackdelar med olika agila metoder. Utifrån denna rapport beslutades det att arbetssättet skulle ta sin grund i Alistair Cockburns *Crystal Clear*-metodik, då den enligt rapporten lämpar sig väl för projektets omfattning samt en grupp av den aktuella storleken.

Crystal Clear är en agil metod som fokuserar mer på utvecklarna än på kunden. Den kan sammanfattas i citatet ”*Put four to six people in a room with workstations and whiteboards and access to the users. Have them deliver running, tested software to the users every one or two months, and otherwise leave them alone*” [11].

Den största avvikelserna som gjordes ifrån Cockburns metodik är att iterations-tiden kortades ned något, vilket kan ses under rubriken ”Iterationer”. Det finns även visst stöd för parallellism i Cockburns metodik [10], vilket var av särskilt intresse då projektet syftade till att utveckla flera samspelande mjukvaror parallellt.

2.2 Designmönster

Då projektet syftade till att ta fram ett publikt ramverk kom standardiserade designmönster att vara önskvärt. Detta säkrades genom att studera andra ramverk med samma målgrupp och använda mönster liknande dessa.

Källkoden till mobil-applikationen gjordes inte publik, men använde ändå standardiserade designmönster för att underlätta samarbetet inom gruppen.

2.3 Undersökningar

Det största fokuset lades på litteraturstudier, samt en analys av vilken funktionalitet som kan vara intressant för en webb-utvecklare. Med anledning av att designen av användargränssnittet sågs som sekundärt var det inte aktuellt med användartester av denna.

2.4 Iterationer

En iteration motsvarade en vecka (sju dagar). Nedan följer en översikt av iterationernas moment.

Möte vid början av varje iteration	<p>Gå igenom tidigare vecka:</p> <ul style="list-style-type: none">• Modell<ul style="list-style-type: none">– Modellen redovisas och ändringar diskuteras.• Kod<ul style="list-style-type: none">– Tidigare veckas kod redovisas för att uppdatera samtliga medlemmar om förändringar, eventuella problem tas upp.• Tester<ul style="list-style-type: none">– Tester redovisas och eventuella buggar går igenom. <p>Kommande vecka</p> <ul style="list-style-type: none">• Modell<ul style="list-style-type: none">– Modellen anpassas för veckans nya funktioner.
Resultat av mötet	<p>Planering inför veckan:</p> <ul style="list-style-type: none">• Vilka funktioner skall implementeras<ul style="list-style-type: none">– Funktioner tas från kravspecifikationen och det bestäms vilka som skall implementeras under veckan.– Funktionerna realiserar i <i>use-cases</i>.• Buggfixar<ul style="list-style-type: none">– Buggar som uppkommit under veckan fördelas på medlemmar i gruppen.

Arbetet under iterationen	<ul style="list-style-type: none"> • Utveckling <ul style="list-style-type: none"> – Implementering av veckans nya funktioner. • Tester/användartester <ul style="list-style-type: none"> – Tester genomförs för de nya funktioner som implementerats. Tester innefattar både enhetstester samt användartester. • Rapportskrivning <ul style="list-style-type: none"> – Rapportskrivning sker kontinuerligt under arbetet för att dokumentera de framsteg som görs under projektets gång.
Resultat av arbetet	<ul style="list-style-type: none"> • Körbar kod med nya funktioner <ul style="list-style-type: none"> – I slutet av veckan finns det en körbar version av programvaran med veckans nya funktioner implementerade.

Då funktionerna som skulle implementeras var väldefinierade och tydligt avgränsade från varandra så fungerade denna arbetsplan mycket bra. Avvikelse förekom främst vid förberedande av presentationer samt arbete med rapporten.

3 Omvärldsanalys

Det finns redan idag ett antal olika tillvägagångssätt för att utveckla applikationer för mobila enheter. Tillvägagångssätten sträcker sig från rena webb-applikationer där allt byggs på webbtekniker till applikationer skrivna specifikt för operativsystemet. Det finns även tillvägagångssätt som ligger mellan dessa två på skalan.

Mobilanpassade webb-applikationer är ett alternativ som vuxit stort de senaste åren [12]. De är generella applikationer som fungerar till de flesta mobila enheter som har möjlighet att köra Javascript i sin webbläsare. Utvecklare behöver då bara utveckla applikationen en gång, för att applikationen sedan skall kunna användas av en stor del av marknaden. Detta gör utvecklingen enklare och mer kostnadseffektiv. Applikationen behöver heller inte distribueras via någon av de distributionskanaler som ägarna av operativsystemen bistår med.

Systemspecifika applikationer är utvecklade för det operativsystem de körs på. Dessa applikationer ger bäst användarupplevelse då de körs lokalt på enheten och kan därför vara mycket snabba. De brukar även vara anpassade för att passa in till den grafiska profilen som operativsystemet har. Applikationerna får även tillgång till alla de hårdvarufunktioner som finns på de mobila enheterna.

Nedan följer en tabell som jämför karaktären hos systemspecifika applikationer med den hos webb-applikationer [12, 13]. Den visar också på hur Cratis är tänkt att lösa några av de problem som finns.

Teknik	Fördelar	Nackdelar
Webb-applikationer	<ul style="list-style-type: none"> • Låg tröskel för utveckling då de bygger på befintliga webbt tekniker • Snabbt att distribuera och uppdatera • Multiplattform: billigare att utveckla en webb-applikation istället för flera plattformsspecifika applikationer 	<ul style="list-style-type: none"> • Inte samma användarupplevelse som systemspecifika applikationer • Kommer inte åt samma funktioner som systemspecifika applikationer gör • Inte lämpade för grafikintensiva applikationer
Systemspecifika applikationer	<ul style="list-style-type: none"> • Användaren känner igen <i>look and feel</i> från operativsystemet • Kommer åt fler funktioner på enheten, ger ger möjlighet till mer interaktiva applikationer • Bättre prestanda: applikationerna kan utnyttja hårdvaruaccelerationen på enheten, bättre prestanda 	<ul style="list-style-type: none"> • Utvecklare måste utveckla en applikation för varje operativsystem, detta ger ökad kostnad för utveckling, längre tid för applikationer att släppas, längre tid för uppdateringar att nå användarna samt högre kostnader för underhåll • Distributionen av applikationerna måste göras via de olika distributionskanalerna ofta ägda av skaparna av operativsystemen • Användare låser in sig i ett ekosystem med applikationer specifikt för ett operativsystem. Dessa kan inte tas med till andra ekosystem om användaren väljer att byta mobil enhet med annat operativsystem
Cratis	<ul style="list-style-type: none"> • Applikationer får tillgång till systemspecifika funktioner • Utvecklare behöver bara utveckla en applikation, som sedan fungerar till alla operativsystem som stödjer Cratis • Utvecklare styr själva över distributionen, behöver inte anpassa sig till operativsystemens ägare • Lägre kostnader för underhåll av applikationen 	<ul style="list-style-type: none"> • Användaren behöver installera en systemspecifik applikation för att få tillgång till webb-applikationer utvecklade med Cratis • Inte lämpad för grafikintensiva applikationer

3.1 Befintliga tekniker

Det finns en del alternativ för den som vill utveckla för mobila enheter utan att speciellt inrikta sig på en viss plattform. Det är också möjligt att kombinera dessa tekniker och på så sätt skapa en avancerad webb-applikation som har ett användargränssnitt som är likvärdigt det som en mobil-applikation har.

Det finns dock fortfarande problem med dessa lösningar. Om utvecklare riktar in sig och skapar en webb-applikation kommer man inte åt all funktionalitet som de mobila enheterna har. Beroende på applikation kan detta vara begränsande eller ett hinder för att en applikation skall fylla sitt syfte.

Några alternativ kan vara att använda sig av Phonegap [14] eller Appcelerator [15] som ger tillgång till många funktioner hos de mobila enheterna. Men använder utvecklare denna teknik måste de fortfarande skapa olika applikationer för olika plattformar, och om applikationen skall uppdateras måste uppdateringen ske genom att användaren laddar ner en ny version av applikationen.

En annan aspekt är de begränsningar som finns hos de olika marknadsplatserna för applikationer med olika regler. Uppfyller man inte dessa kan man inte distribuera sin applikation via dessa marknadsplatser.

Nedan följer en kort förklaring av några befintliga tekniker som delvis löser problemet med att utveckla plattformsoberoende applikationer.

3.1.1 Phonegap

Phonegap [14] är ett ramverk som tillåter utvecklare att skapa applikationer för olika mobila operativsystem genom att skriva kod i HTML, CSS och Javascript. Ramverket tar sedan denna kod och kapslar in den i en applikation. Detta innebär att man kan utnyttja många funktioner på enheten som man inte kan komma åt med en webbsida. Applikationen måste dock kompileras när man gjort ändringar i HTML-koden. Detta medför att utvecklaren inte kan uppdatera sin webb-applikation utan att användaren måste ladda ner en ny version.

3.1.2 Appcelerator

Appcelerator [15] är ett ramverk för att generera kod till olika mobila plattformar, likt Phonegap. Liksom Phonegap så måste utvecklare distribuera sina applikationer via tillgängliga marknadsplatser, trots att ramverket bygger på webb-principer. Likaledes måste användaren aktivt ladda ner nya versioner av den resulterande mjukvaran.

3.1.3 Mozilla Boot 2 Gecko

Boot 2 Gecko [16] är ett nytt operativsystem som är helt webb-baserat. Operativsystemet finns ännu inte på marknaden utan är ett projekt som drivs av Mozilla och tillkännagavs den 25 juni 2011 [17]. Detta projekt går ut på att skapa ett operativsystem som enbart bygger på applikationer som är skrivna på befintliga webb-tekniker. Detta är ett stort initiativ och innebär att nya webb-standarder måste skapas för att tillgodose de behov som dagens mobila applikationsutvecklare har. I skrivande stund har inget datum tillkännagivits när Boot 2 Gecko kommer att vara tillgängligt för slutanvändare.

3.1.4 HTML 5

HTML 5 [18] är den senaste versionen av HTML (*Hyper Text Markup Language*) som är ett märkspråk som används för att formatera innehåll på webbsidor. HTML 5 innehåller funktioner och verktyg för att skapa mer avancerade funktioner än tidigare, till exempel inbäddad video och ljud samt verktyg för mer avancerad formatering.

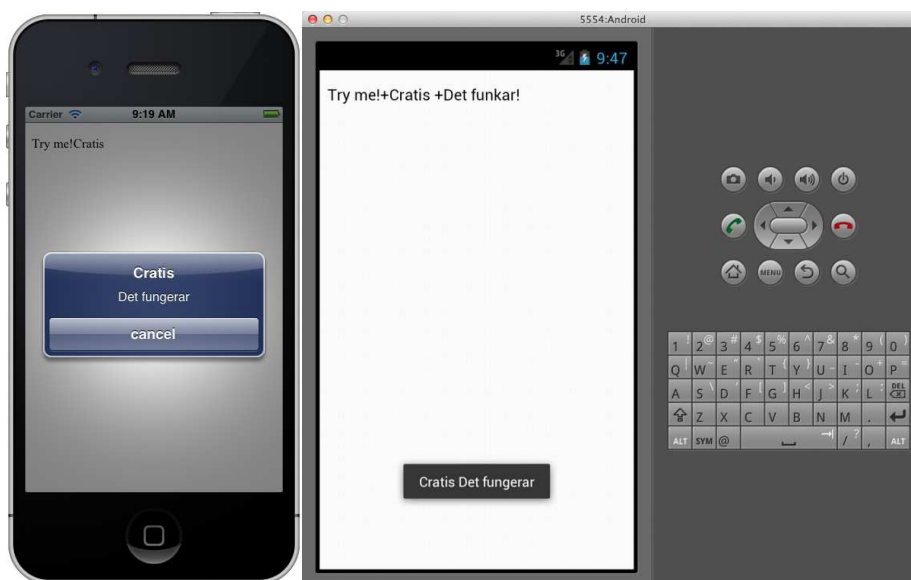
3.1.5 Look and feel-imiterande verktyg

För att berika användarupplevelsen på en webb-applikation så finns det ett antal ramverk för att efterlikna *look and feel* hos en plattformsspecifik applikation. Dessa ramverk medger inte extra funktionalitet mot den mobila enheten, utan är endast till för grafisk representation. Några exempel på sådana ramverk är: Sencha Touch [19], JQuery Mobile [20], IUI [21].

4 Analys

För att Cratis-ramverket skall fylla sitt syfte så krävs det en implementering till varje mobilt operativsystem för att komma åt de mer hårdvarunära funktionerna. Detta innebär att webb-applikationen måste kunna kommunicera med denna applikation som skrivits specifikt för det operativsystem som körs.

I början av projektet gjordes det två *proof of concept*, en till Android och en till IOS. Det byggdes en enkel mobil-applikation som bestod av en webb-vy för att kunna visa webb-applikationer. Därefter skapades en enkel webb-applikation som kunde visas av mobil-applikationen. Dessa kunde sedan kommunicera med varandra och webb-applikationen kunde kalla på funktioner i enhetens operativsystem.



Figur 1: *De initiala applikationerna för att testa kommunikation mellan webb-applikation och mobil-applikation.*

Kommunikationen mellan webb-applikationen och Android-enheten sker via Javascript. I Android finns möjligheten att binda ett objekt till Javascript så att objektet kan bli kallat på av Javascript. Detta användes för webb-applikationen att kalla på funktioner i Android-applikationen via Javascript. För att returnera data används metoden `loadUrl()`, med denna kan Javascript-funktioner exekveras via ett URL-anrop, och på så sätt kan Android-applikationen kalla på funktioner som påverkar webbsidan.

I IOS finns inte samma funktioner som hos Android, istället används omdirigering för att kommunicera med IOS-applikationen. Javascript kallar på en omdirigering med parametrar in till applikationen. Webb-vyn i applikationen behandlar dessa parametrar och parsar dem till ett metoanrop. För att svara finns det en funktion i IOS för att exekvera Javascript, och kan på så sätt kalla på funktioner som påverkar webb-applikationen.

I figur 1 syns de *proof of concept* som gjordes. En enkel webb-applikation med en länk med texten *Try me!*. När användaren trycker på denna kallas en plattformsspecifik funktion på enheten. Denna funktion visar en dialogruta för IOS och en notis för Android, som är grafiska komponenter i respektive operativsystem. Ett returvärde skickas tillbaka till webb-applikationen och det läggs till text på sidan som kommer från mobil-applikation.

Denna enkla applikation visar att kommunikationen mellan webb-applikation och enhetens operativsystem fungerar. Det sker ett metoanrop till mobil-applikationen; enheten utför en systemspecifik handling och slutligen skickas ett returvärde tillbaka till webb-applikationen.

Detta koncept visade att det var möjligt att genomföra projektet. De metoder som upptäcktes under denna studie för kommunikation mellan webb-applikation och mobil-applikation var de som låg till grund för hur ramverket senare skulle implementeras. Då projektet sträckte sig över en begränsad tid tog beslutet att fokusera på en plattform.

Plattformen som valdes var Android. Ett flertal orsaker bidrog till att denna plattform valdes. Projektgruppen hade tidigare erfarenhet från Java, programmeringsspråket som används i Android, vilket innebar att inlärningstiden skulle bli kortare. Detta eftersom det inte behövde läggas tid på att lära sig ett nytt programmeringsspråk, vilket hade varit fallet om andra plattformar hade valts. Android är också ett av de större mobila operativsystemen på marknaden [22], och att det har öppen källkod kan också underlätta för utvecklingen.

4.1 Utveckling av grafiska användargränssnitt i Android

När man utvecklar en applikation till de olika mobil-plattformarna finns det ofta krav på applikationens beteende. I Android finns det specifika krav på hur man ska implementera det grafiska användargränssnittet. Varför detta specifika krav finns och hur det påverkade utvecklingen i detta projekt presenteras nedan.

Inom forskningsområdet människa–datorinteraktion råder det sedan länge konsensus om den påverkan interaktiva responstider, den tid det tar för systemet att svara på inmatning, har på användares tillfredsställelse och produktivitet [23, 24, 25, 26]. Google och Open Handset Alliance strävar därför efter att alla applikationer i Android skall vara responsiva. Om applikationerna inte är responsiva kommer användaren tycka att enheten och operativsystemet i sig inte är tillfredsställande.

För att förebygga denna uppfattning hos användaren finns det inbyggd funktionalitet i Android som skall reglera responsiviteten i applikationer. I praktiken innebär det att om en applikation inte är tillräckligt responsiv under en viss tid kommer Androidsystemet att visa en dialog för användaren. Denna dialog kallas för Application Not Responding (ANR) dialog. ANR-dialogen är något som en utvecklare vill undvika därför att detta medför att användaren som tidigare nämnt får en negativ bild av applikationen. För att undvika ANR-

dialogen måste utvecklare undvika det synkrona begäran/svar-mönstret och utveckla användargränssnittet med asynkrona anrop istället.

I ett grafiskt användargränssnitt körs koden normalt inte hela tiden. Applikationen får anrop från operativsystemet eller fönsterhanteraren när något av intresse händer, då gör den någonting med anropet och återvänder till att vänta. Om man istället väljer att blockera användargränssnitts-tråden för att vänta på ett specifikt svar betyder detta att applikationen inte kan få någon annan indata än detta, och hela systemet är låst tills användaren svarar. I en telefon skulle detta betyda att du som programmerare genom ett modalt popup-fönster kan förhindra användaren från att ta emot samtal, eftersom telefonen är upptagen med att vänta på svar från popup-fönstret.

4.2 Beroendeinjektion

För att underlätta utvecklingen i mjukvaruprojekt finns det en stor mängd olika strategier och designprinciper man kan följa. I det här projektet har beroendeinjektion använts för att förbättra kvaliteten på koden i mobil-applikationen.

Beroendeinjektion förenklar möjligheterna att skapa kod som är utbyggbar, testbar och återanvändbar [27]. Det är ett enkelt mönster för att löst koppla ihop objekt och deras beroenden [28]. Istället för att ett objekt ska skapa sina egna beroenden finns det tre standardmetoder man kan använda för att implementera beroendeinjektion. Dessa är *constructor injection*, *setter injection* och *interface injection* [29]. Beroendeinjektion kallas också för Hollywood-principen ("ring inte oss, vi ringer dig") eller *inversion of control*.

4.3 Säkerhet

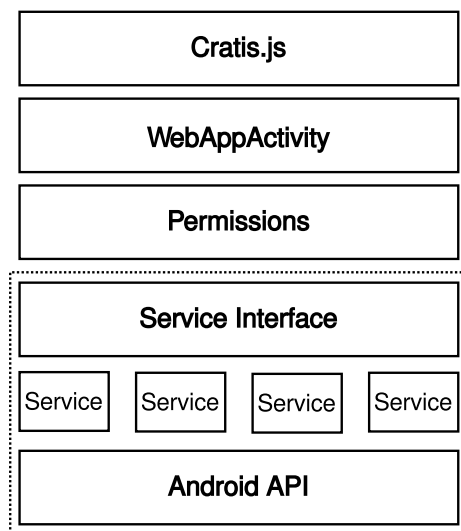
Då ramverket hanterar och delar med sig av privat information om användaren, såsom dennes geografiska position, var det från början viktigt att se till att denna delning skedde på ett för användaren säkert sätt.

För att användaren hela tiden ska ha kontroll över vad en webb-applikation får och inte får tillgång till har vi valt att implementera ett system som hanterar detta. Första gången webb-applikationen vill få tillgång till en tjänst som ramverket tillhandahåller visas ett popup-fönster. Användaren kan då välja om webb-applikationen ska få tillåtelse att använda tjänsten när som helst, bara för den innevarande sessionen eller aldrig.

Anledningen till att denna uppdelning är gjord är att en användare kan vara intresserad av att prova en applikation under en kortare tid innan denne bestämmer sig för hur applikationen skall användas framöver. Därför kan det vara bra för användaren att inte ge webb-applikationen fria tyglar att hämta känslig data när som helst. Om användaren därefter bestämmer sig för att fortsätta använda webb-applikationen kan denne spara tillåtelsen så att webb-applikationen inte visar popup-fönster varenda session.

Detta innebär i praktiken att säkerhetslagret flyttas från operativsystems nivå till applikationsnivå (se figur 2), något som är en nödvändighet då Android-

applikationen måste kunna hantera flera webb-applikationer med sinsemellan skilda tillåtelser. Det gör i sin tur att det har varit väldigt viktigt att bygga in ett säkerhetslager i Android-applikationen på ett tidigt stadium.



Figur 2: Visualisering av de olika lagren i applikationen och hur säkerhetslagret begränsar åtkomsten till Services.

Den enda egentliga nackdelen med detta tillvägagångssätt är risken för ett *phishing*-angrepp. Det innebär att en annan utvecklare, som utger sig för att vara ägare till ramverket, kan implementera ramverket i en annan applikation och på så sätt tillåta direkt åtkomst till kritiska data. Sannolikheten för ett sådant angrepp bedöms som mycket liten, och det finns egentligen inget reellt alternativ till det valda upplägget, då det hade varit tekniskt omöjligt att implementera säkerhetslagret i Javascript-ramverket.

Anledningen till att det inte skulle gå att ha säkerhetslagret på Javascript-nivå är att källkoden måste vara tillgänglig för webb-utvecklaren att se, och därför skulle denne kunna passera säkerhetslagret genom att bara skicka direkta anrop till Android-applikationen istället för att använda de fördefinierade metoderna.

4.4 Testning av applikationen och ramverket

Vikten av att kontinuerligt testa mjukvara medan den utvecklas är allmänt accepterad hos mjukvaruutvecklare och kommer inte kommenteras närmare i denna rapport. Detta projekt är inget undantag för den här uppfattningen och därför har testning genomförts regelbundet genom hela projektet.

Inom mjukvarutestning finns det generellt sett två olika strategier för testning: *black box*- och *white box*-testning [30]. *Black box*-testning innebär att programmet ses som en svart låda. Då skall fokus vara på att programmet fungerar som det är specificerat och inte hur programmet fungerar internt. Denna strategi

kallas också för data-driven eller inmatning/utmatning-driven testning. I kontrast till *black box*-strategin testar *white box*-strategin de interna funktionerna i programmet. I detta projekt har båda strategierna använts genomgående.

I enlighet med grundidéerna inom agil utveckling har det genom projektet varit viktigt att mjukvaran alltid ska vara körbar och därmed testbar. Testning av mobil-applikationen skedde till stor del genom Android-emulatorn. Mot slutet av utvecklingen testades det även på en riktigt mobiltelefon.

Vid testning av ramverket användes både *white*- och *black box*-metoderna samtidigt. Här följer en beskrivning av ett typiskt test av ramverket.

Mobil-applikationen startades, därefter skrevs adressen till webb-applikationen som implementerar ramverket in. Webb-applikationen visas upp och testaren trycker på en knapp i webb-applikationen som är avsedd för att testa en specifik funktion i ramverket. Sedan sker en manuell utvärdering av utkomsten av testet.

I detta typiska test av ramverket behandlades mobil-applikationen som en svart låda medans interna funktioner i ramverket testades, alltså *white box*-testning.

Utöver den *black box*-testningen som utfördes på mobil-applikationen vid testning av ramverket har även *white box*-testning utförts på applikationen. Detta gjordes främst genom att skriva *unit*-tester. Dessa tester är typiska *white box*-tester som testar små enheter – *units* av kod.

4.5 Dokumentation

Eftersom projektet har syftat till att utveckla ett ramverk som skall vara tillgängligt för tredjeparts-utvecklare har det varit väldigt viktigt att dokumentera själva ramverket. Inom liknande projekt med Javascript-bibliotek är det vanligt att biblioteket dokumenteras med hjälp av HTML på en webbsida, ett exempel på detta är JQuery [31]. Dokumentationen Cratis har varit liknande men i textform.

Javascript-ramverkets dokumentation går i stora drag ut på att dela upp ramverket i logiska enheter och i varje enhet dokumenteras funktionalitet inom enheten. Olika exempel på enheter kan vara lokalisering, kalender eller vibration. I varje enhet specificeras sedan vilka olika Javascript-funktioner som är tillgängliga för utvecklaren, vad den specifika funktionen har för uppgift, samt vilka parametrar som går att skicka med, såväl obligatoriska som valfria.

För den tekniskt intresserade läsaren finns dokumentationen att tillgå som appendix till denna rapport.

Vad gäller Android-applikationen har ingen extern dokumentation behövts, då källkoden är sluten. Däremot har dokumentation använts som ett internt hjälpmedel inom gruppen för att stödja den simultana utvecklingen. Eftersom applikationen är skriven i Java så föll det sig naturligt att använda Javadoc.

Javadoc utvecklades ursprungligen av Sun Microsystems, och är ett verktyg för att automatiskt generera HTML-filer med kommentarer och förklaringar till klasser, metoder och dylikt [32]. Eclipse, som är utvecklingsmiljön som

använts i det här projektet, kan visa Javadoc-dokumentationen direkt i miljön när användaren för muspekaren över till exempel ett metodnamn. På detta sätt blir det lätt att snabbt komma åt information till exempel vad en metod gör och vilka parametrar den tar emot.

4.6 Verktyg, mjukvarubibliotek och andra hjälpmedel

I följande avsnitt presenteras de hjälpmedel och verktyg som använts i utvecklandet av såväl mobil-applikationen som webb-applikationen och ramverket.

4.6.1 Versionshanterings-verktyg

Att använda versionshantering är en självklarhet i modern mjukvaruutveckling, och det finns många anledningar till detta. En uppenbar fördel med versionshantering är, som namnet antyder, att versionshanterings-mjukvaran håller reda på förändringar som görs under utvecklingen och på så sätt låter utvecklaren gå fram och tillbaka mellan revisioner, något som kommer väl till pass om buggar introduceras [33].

Den kanske största fördelen med versionshantering är dock att den tillåter samarbete mellan flera utvecklare, något som varit av största relevans i projektet. Detta åstadkoms genom att en *repository* innehåller all kod samt metadata om revisioner och förändringar. Genom att synkronisera utvecklingsmiljön med ett *repository* kan utvecklarna komma åt varandras förändringar och därmed hålla hela kodbasen synkron. Till detta kommer dessutom verktyg för att följa processen och se hur projektet fortskrider.

I projektet har versionshanterings-systemet Git [34] använts. Valet föll på Git dels på grund av gruppens tidigare erfarenhet av systemet, vilket gjorde att det inte krävdes någon egentlig inlärningsperiod. I allmänhet är inte valet av versionshanterings-system en egentlig springande punkt, olika system har olika fördelar.

Den stora skillnaden mellan Git och till exempel SVN [35], som är ett annat populärt versionshanteringssystem, är att Git är ett distribuerat system; detta innebär i praktiken att versionshanteringen faktiskt sker på varje enskild enhet som har projektet installerat. Därefter synkroniseras de olika projekten på en central server. Detta gör att varje utvecklare kan föra in ändringar i programvaran i sin lokala utvecklingsmiljö utan att behöva ha tillgång till en internetuppkoppling.

I SVN och andra centraliserade versionshanterings-system sker all versionshantering på servern, varje utvecklare laddar upp sina förändringar till servern och därefter är det serverns uppgift att hålla versionshistoriken under kontroll.

Som *repository*-tjänst har Bitbucket [36] använts. Bitbucket har använts då det är en av få tjänster som erbjuder gratis *repository* för projekt med sluten källkod, och som också tillåter flera användare av samma *repository*. Det var

självlklart också viktigt att Bitbucket har stöd för Git, då *repository*-tjänsten valdes efter versionshanterings-systemet, och inte tvärtom.

4.6.2 Utvecklingsmiljö

Utvecklandet av Android-applikationen skedde i Eclipse. Anledningen till detta är att gruppen har erfarenhet från programmet, men även det stöd som Eclipse har för Android-utveckling.

Android Developer Tools (ADT) är ett insticksprogram till Eclipse som innehåller många funktioner som underlättar utveckling av Android-applikationer [37]. ADT inkluderar inbyggd hantering av Android-projekt i Eclipse samt de funktioner som finns i Android Software Development Kit (Android SDK) [38]. Inkluderat i dessa utvecklingsverktyg finns även en Android-emulator, som använts flitigt genom projektet för att testköra de olika mjukvarorna.

4.6.3 Programmeringsspråk för ramverket

Ramverket för webb-applikationen är implementerat i Javascript. Detta för att det är ett välkänt skriptspråk för webbläsare som används på över 90% av de 10.000 mest populära webbsidorna på internet [39].

4.6.4 Datautbytningsformatet JSON

Det behövs ett datautbytningsformat som fungerar både i ramverket och i applikationen. I ramverket används JSON [40] för detta syfte. JSON är ett lättviktigt och programmeringsspråksberoende datautbytningsformat [41]. Det är lätt för människor att läsa och skriva, samtidigt som det är lätt att parse och generera [42].

4.6.5 Gson

Vid utvecklandet av mobil-applikationen användes biblioteket Gson [43]. Gson är ett Java-bibliotek som används för att konvertera Java-objekt till JSON och tvärtom. Detta bibliotek används när information skickas mellan webb-applikationen och Android-applikationen. Gson är utvecklat av Google.

4.6.6 Google Guice och RoboGuice

Ett annat bibliotek som användas är RoboGuice [44], som är en utbyggnad av Google Guice [45]. Google Guice tillhandahåller stöd för användning av beroendeinjektion med hjälp av annotationer, det vill säga nyckelord som visar var i koden injektion ska göras. RoboGuice är en utbyggnad av Google Guice som är anpassad för utveckling av Android-applikationer.

4.6.7 Testverktyg för att simulera sensorer på Android

Då ramverket kommer att tillåta att många funktioner från Android-enheten blir tillgängliga för webb-applikationen så krävdes ett systematiskt sätt att simulera de olika sensorerna i emulatorn som användes för att testa funktionerna.

För att simulera sensorerna på Android-enheten användes Sensorsimulator, ett verktyg som skapats av Openintents [46]. Detta verktyg valdes då det har öppen källkod och är gratis att använda.

Verktyget består av två delar, en applikation till Android-emulatorn samt en fristående Java applikation för att skicka värden till olika sensorer. Android-applikationen installeras på emulatorn och kopplas samman med den fristående Java-applikationen. För att det skall fungera krävs det en anslutning mellan de två. Då kan Java-applikationen simulera sensorsignaler som uppfattas av Android-enheten.

För att det skall fungera med Cratis krävs det att man använder ett bibliotek specifikt utformat för Sensorsimulator applikationen.

5 Kravspecifikation

Kraven som togs fram för detta ramverk byggdes på de funktioner som behövdes för att ramverket skulle fylla sitt syfte. Eftersom enheter från olika tillverkare har olika funktionaliteter behövdes det göra ett snitt av de mest använda funktioner på de olika mobila operativsystemen. Ett mål med kravställningen var att abstrahera bort underliggande operativsystem för applikations-utvecklare.

Phonegap, Appcelerator och andra populära applikationer på de olika plattformarna undersöktes. Detta för att få en bra bild över vilka operativsystems specifika funktioner som var vanligast att använda. Efter att detta undersökts togs en lista fram med funktioner som skulle vara önskvärda att ha med i Cratis.

Prioriteten av kraven bestämdes genom att analysera de mest populära applikationerna till Android och Iphone, för att sedan hitta funktioner som de använder sig av som inte finns tillgängliga för webb-applikationer.

Vissa av kraven finns redan implementerade i vissa mobila operativsystems webbläsare. Men alla krav står med för att få med den totala mängden funktioner för att Cratis implementeringen skall vara fullständig.

Under projektets gång utvecklades de krav som bestämts tidigare till *use-cases*. Detta gjordes för att få en klarare bild över hur funktionerna skulle fungera för användaren och var till stort stöd för utvecklaren när de skulle implementeras. Dessa *use-cases* finns som appendix till denna rapport.

Eftersom projektet är indelat i två delar finns det två olika kravspecifikationer. Det finns dock några grundläggande krav som är gemensamma för båda delarna.

Gemensamma krav:

- Metoder skall kunna anropas och data skall kunna skickas från Android-applikationen till ramverket
- Metoder skall kunna anropas och data skall kunna skickas från ramverket till Android-applikationen
- Android-applikationen skall kunna visa en webb-applikationen som implementerar ramverket.
- Det skall finnas metoder för att hantera åtkomst till de olika funktionerna i ramverket. Android-applikationen kommer att hantera de flesta av dessa funktioner.

5.1 Ramverkets funktioner

Prioritet 1:

- Ladda upp godtyckliga filer/bilder/video
 - Det skall gå att välja filer/bilder/video från enheten och ladda upp till en vanlig *upload form* på en webb-applikation.
- Hämta godtyckliga filer
 - Det skall gå att hämta hem godtyckliga filer från en webb-applikation och lägga dem i lämplig mapp på enheten.
- GPS-position
 - Webb-applikationen skall kunna få tillgång till enhetens koordinater. Webb-applikationen ska också kunna sätta sig som lyssnare på GPS:en för att kunna följa förändringar i positionen hos enheten.
- Hårdvaruknappar
 - Webb-applikationen skall kunna sätta sig som lyssnare på de hårdvaruknappar som finns tillgängliga på enheten. Webb-applikationen skall kunna lyssna på flera knappar samtidigt. Det går enbart att sätta sig på lyssnare på de knappar som finns tillgängliga på den fysiska enheten.
- Applikationen skall implementera följande säkerhetsfunktioner:
 - Applikationen skall kunna hantera åtkomst för olika funktioner
 - Användargodkännande skall ges vid åtkomst av telefonens funktioner

Prioritet 2:

- Kalender
 - Webb-applikationen skall kunna lägga till händelser på enhetens kalender. Det skall även gå att kontrollera om det finns händelser inlagda på kalendern under en viss tid.
- Kontakter från systemets adressbok
 - Webb-applikationen skall kunna lägga in kontakter i enhetens adressbok. Webb-applikationen skall även kunna hämta kontakter från enhetens adressbok.
- Spela upp ljud
 - Webb-applikationen skall kunna spela upp ljud.
- Vibration
 - Webb-applikationen skall kunna aktivera enhetens vibrator.

Prioritet 3:

- Kompassriktning/magnetometer

- Webb-applikationen skall kunna få kompassriktningen från enheten. Webb-applikationen skall även kunna sätta sig som lyssnare på kompassriktningen för att kunna följa förändring i kompassriktningen.
- Lampa
 - Webb-applikationen skall kunna tända och släcka lampan/blixten på enheten om den har en sådan. Webb-applikationen skall även kunna skicka in ett fält med mönster för blinkning.
- Accelerometer/gyrometer
 - Webb-applikationen skall kunna få värden från enhetens accelerometer/gyrometer. Den skall även kunna sätta sig på lyssnare på förändring hos dessa sensorer.
- Apparatdata
 - Webb-applikationen skall kunna begära information om den enhet som används.
- Notifikationer
 - Webb-applikationen skall kunna skicka notifikationen till enheten. Detta enbart efter att enheten har besökt webb-applikationen och lämplig information utbytt för att möjliggöra detta.
- Ljussensor
 - Webb-applikationen skall kunna få värden från ljussensorn på enheten. Webb-applikationen skall även kunna sätta sig som lyssnare på förändring hos ljussensorn.

Prioritet 4:

- Strömma upp ljud
 - Webb-applikationen skall kunna öppna en ljudström från enhetens mikrofon.
- USB
 - Webb-applikationen skall få tillgång till det USB-gränssnitt som enheten har.
- Bluetooth
 - Webb-applikationen skall få tillgång till det Bluetooth-gränssnitt som enheten har.
- Nätverk
 - Webb-applikationen skall kunna hämta information om de tillgängliga trådlösa nätverk som finns i närheten av enheten.

5.2 Android-applikationen

Följande krav är de funktionella krav som Android-applikationen skall ha för att kunna fungera som applikation. Prioritet 1:

- Visa en webb-applikation som implementerar ramverket.
- Gå ur en webb-applikation
- Implementera ramverkets funktioner (dessa efter prioritering)

Prioritet 2:

- Spara länk till webb-applikationer
- Visa sparade webb-applikationer
- Lägg till ikoner för webb-applikationer på hemskärm (widget)
- Växla mellan webb-applikationer

Prioritet 3:

- Spara inställningar/åtkomst till webb-applikationer

Prioritet 4:

- Köra webb-applikation i bakgrunden
- Ta bort webb-applikation och tillhörande filer länkar inställningar
- Kunna spara ner och köra webb-applikationer så att de kan köras offline.

5.3 Icke funktionella krav

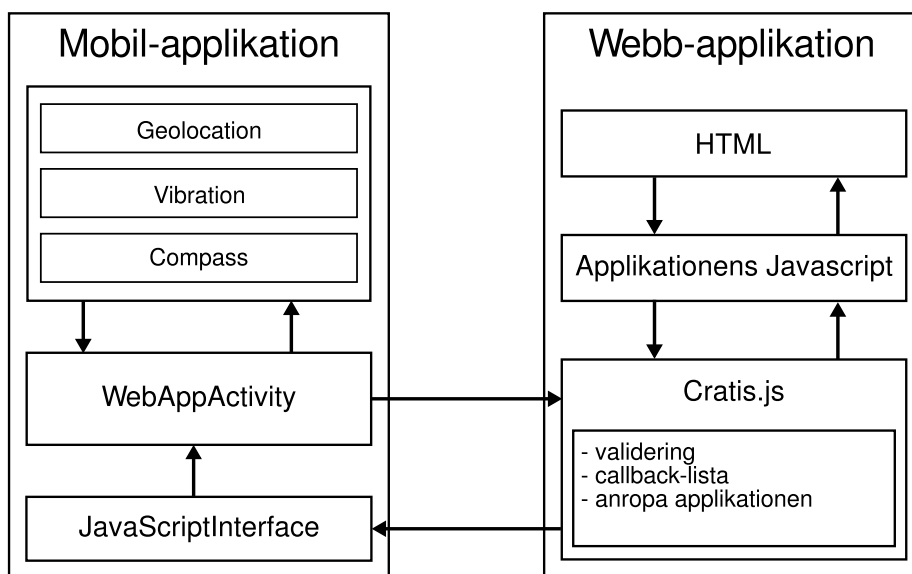
- Ramverket och Android-applikationen skall designas och implementeras så att de enkelt kan byggas ut.
- Ramverket skall designas så att det liknar andra ramverks design, för att göra det enkelt för utvecklare att implementera.
- Ramverket och Android-applikationen skall implementeras utan att göra avkall på säkerheten.
- Ramverket skall kunna gå att använda i samband med andra Javascript-ramverk.

6 Design

Mot bakgrund av analysen har Cratis designats i två olika delar, ett ramverk som används av webb-applikationer och en applikation som användaren installerar på sin mobila enhet.

Ramverket för webb-utvecklare är ett Javascript-ramverk, detta används på liknande sätt som andra populära Javascript-ramverk. Utvecklaren lägger till ramverket i sin webb-applikation och kan sedan kalla på Javascript-funktioner (se figur 3). Dessa Javascript-funktioner motsvarar vanliga systemspecifika funktioner som hade varit tillgängliga om applikationen utvecklats specifikt för ett mobilt operativsystem.

För att knyta de olika funktionerna till den mobila enheten används en applikation på enheten. Denna applikation är skriven specifikt för varje mobilt operativsystem och kan liknas med en webbläsare. Användaren använder då denna applikation för att få åtkomst till alla webb-applikationer som använder Cratis-ramverket. Denna applikation agerar även som ett säkerhetslager och ger tillgång eller nekar webb-applikationer till den mobila enhetens funktioner och data.



Figur 3: Förenklat diagram över kommunikation mellan en webb-applikation och mobil-applikationen.

Att utnyttja Cratis underlättar för utvecklare som slipper utveckla en separat applikation för varje mobilt operativsystem, men samtidigt får tillgång till vanliga systemspecifika funktioner genom att använda sig av Cratis-ramverket. Ramverket tillåter också utökade möjligheter för de webb-utvecklare som satsat på mobila webbsidor istället för separata applikationer för varje system, dessa utvecklare kan, om de använder Cratis-ramverket, utöka funktionaliteten på sina mobila webb-applikationer för att få en bättre användarupplevelse.

6.1 Design av Ramverket

En del av Cratis är det ramverk som webb-applikationerna använder för att kalla på funktioner hos de mobila enheterna. Detta ramverk är utvecklat i Javascript och utvecklaren av webb-applikationer behöver bara importera Javascript-filen till sin webb-applikation för att kunna kalla på funktioner.

6.1.1 Generella designprinciper

Ramverket är utvecklat för att likna befintliga Javascript-ramverk så mycket som möjligt, detta är gjort för att inlärningskurvan ska vara låg för utvecklare som vill använda sig av ramverket. Under designarbetet så undersöktes de mest populära Javascript-biblioteken för att hitta gemensamma designmönster. Dessa mönster har sedan i största mån används under utvecklingen av detta ramverk, dels för att dessa designprinciper innebär att användare till ramverket kommer att känna igen sig från andra ramverk. Designprinciperna följer även god praxis för Javascript-utveckling.

6.1.2 Inkapsling av metoder

Eftersom att alla funktioner i Javascript är objekt, och att det inte existerar privata funktioner på samma sätt som i andra programmeringsspråk behövs ett designmönster för att kunna kapsla in de olika funktioner som implementeras. Detta för att separera de interna funktionerna som används inom ramverket från de publika funktionerna som är tillgängliga att anropa från webb-applikationen.

För att åstadkomma detta så används ett designmönster som heter *Revealing module pattern*, detta är utvecklat av Christian Heilmann [47] och har till syfte att på ett enkelt sätt kapsla in funktioner i moduler. När funktionerna är inkapslade kan de inte anropas globalt. De funktioner och variabler som man vill ha synliga returnerar man till ett modulobjekt. Sedan kallas funktionerna som variabler på detta objekt.

Genom att använda detta designmönster så kan man enkelt separera de publika funktionerna som är tillgängliga mot webb-applikationsutvecklaren mot de interna funktionerna. Designmönstret gör också koden mycket läsvänlig vilket underlättar för de utvecklare som vill bygga vidare på ramverket i sina egna webb-applikationer.

6.1.3 Parametrar

Applikationer som använder ramverket kan skicka olika antal parametrar för olika funktioner. För att kunna hantera detta och för att validering skall göras så lätt som möjligt används en liknande metod som motsvarande i JQuery. JQuery är det mest populära Javascript-ramverket på internet idag [48]. Den enda parametern som skickas är ett objekt, detta objekt innehåller de parametrar som skall skickas. Detta gör det enkelt för användare, samtidigt som validering förenklas eftersom endast en parameter skickas till funktionen.

6.1.4 Första steget i valideringen

Ett av designmålen är att ramverket skall vara så effektivt som möjligt för att inte sätta onödig press på webb-servrar samt minska väntetid för slutanvändaren. Detta genom att göra kommunikation mellan den mobila enheten och webb-applikation så effektiv som möjligt.

En av metoderna för att göra ramverket effektivt är att utföra mindre genomgående valideringar av de parametrar som skickas vidare till mobil-applikationen. De kontroller som görs är att parametrarna finns och att de har rätt namn, däremot utvärderas inte parametervärdena. Genom att ramverket gör dessa initiala valideringar kan tid sparas om parametrarna som skickas är felaktiga, då upptäcks detta i ett tidigt skede och någon kommunikation mellan webb-applikationen och den mobila enheten behöver inte ske.

6.1.5 Retur av data

En av de fundamentala funktionerna hos ramverket är att kommunikation skall kunna ske mellan en webb-applikation och en mobil enhet. Kommunikation innefattar att anropa metoder på den mobila enheten, men det behövs även ett system för att kunna hantera retur av data.

Data returneras vid två situationer. Den första är när webb-applikationen anropar en funktion som kräver en retur, till exempel begära en kontakt från den mobila enhetens kontakter. En annan situation är då webb-applikationen har satt sig som lyssnare på något hos den mobila enheten, då skickas returer vid varje händelse som lyssnas på, till exempel vid lyssning på kompassförändringar. Det krävs även hantering om något skulle gå fel under funktionsanropet.

När en funktion i ramverket anropas är både returfunktion och en felhanteringsfunktion obligatoriska att skicka med i parameterobjektet. Felhanteringsfunktionen kommer att anropas om något går fel under exekveringen av funktionen på den mobila enheten. Denna är separerad från returneringsfunktionen för att göra det tydligare och enklare för webb-applikationsutvecklaren att hantera eventuella fel som sker under exekvering.

Returfunktionen kallas om anropen går som det skall. Till denna skickas också returdata i formen av ett JSON-objekt.

6.1.6 Asynkrona anrop

För att inte låsa webb-applikationen för användaren används asynkrona anrop för att kalla på funktioner på den mobila enheten. Detta underlättar även för webb-applikationen att sätta sig som lyssnare på den mobila enhetens funktioner.

När webb-applikationen anropar funktioner tas de först hand om av ramverket. Som tidigare nämnt sker viss validering vid detta anrop, det som också sker är att ramverket bygger en unikt ID för det specifika anropet baserat på namnet på anropet och ett indexnummer. Returfunktionen och felhanteringsfunktionen placeras i ett objekt som sedan läggs i en lista på indexplatsen enligt det ID

som genererats. Denna lista används sedan då ramverkets funktion för retur kallas.

Ramverket anropar den mobila applikationen genom Javascript och skickar med vilken funktion som ska anropas, ramverket skickar också med de unika ID för returfunktionen. Detta anrop är den första kommunikationen mellan webb-applikationen och den mobila enheten, detta anrop är även asynkront och webb-applikationen väntar inte på svar från detta anrop.

Efter att den mobila enheten har behandlat funktionsanropet kommer den mobila enheten kalla på en returfunktion för ramverket, denna funktion är generisk och gäller för samtliga funktioner. Det som skickas med till denna funktion är ett JSON-objekt med data, en indikator om anropet var lyckat eller inte, samt funktionsanropets ID som genererats av ramverket.

Ramverket söker upp objektet som är kopplat till anropets ID. Sedan kallas returfunktionen eller felhanteringsfunktionen beroende på indikatorn tillsammans med datan som returnerats.

6.2 Design av Android-applikationen

Android-applikationer utgår i grunden från en aktivitet. En aktivitet är en *controller* som visar en vy i applikationen. Tanken bakom aktiviteterna är att de skall användas vid navigation i applikationen där varje vy är en egen instans av en aktivitet.

I Cratis är `CratisActivity` den aktivitet som först startas då applikationen startar. Denna aktivitet innehåller en vy där användaren väljer vilken webb-applikation som ska startas. Efter att ha angett en sökväg till webb-applikationen skapas en instans av `WebAppActivity`. `WebAppActivity` har en vy som agerar webbläsare och visar webb-applikationer. Vid navigation till en annan webb-applikation (i en annan aktivitet) eller minimering av applikationen sparas den nuvarande aktivitetens läge. Utvecklaren definierar beteendet vid förändringar i dess livscykel. Då en aktivitet pausas och en annan visas kan den exempelvis släppa anslutningar och större objekt [49].

Vyn som visar webb-applikationen fångar de anrop som görs. De Javascript-anrop, sidladdningar, anrop till filsystemet och felmeddelanden som ska till Android-applikationen behandlas och övriga skickas vidare då det inte är något för ramverket att behandla.

Funktionalitet för att fånga Javascript anrop är inbyggt i Androids webb-vy, medan funktionalitet för att lägga till Javascript-metoder fås genom att använda ett Javascript-gränssnitt. Ett Javascript-gränssnitt är en klass skriven med Java och som kompileras precis som vilken klass som helst. Det webb-vyn gör är att tilldela klassen ett namespace och översätter anropen till `JavascriptInterface` som fångar upp anropen innehållandes strängar med information om vilken sorts anrop som gjorts. Innehållet är typ av tjänst, metod, parametrar samt ett *callback-ID*.

Det första anropet från en webb-applikation som använder Cratis kommer meddela Android-applikationen att ramverket är implementerat. Anledningen till detta är att Android-applikationen inte visar dessa webb-applikationer då de lika gärna, om inte hellre, visas i systemets webbläsare.

Efter att ha kontrollerat att webb-applikationen använder ramverket är Android-applikationen redo att ta emot och utföra metदानrop på ramverket. Därefter skickar `JavascriptInterface` vidare anropet till `WebAppActivity`, som sedan matchar strängen innehållandes tjänst mot de tjänster som är implementerade. Ett objekt av typen `CratisCall` skapas. Ett sådant objekt innehåller en referens till tjänsten, sträng med metodnamnet som anropas, lista med parametrar och ett unikt ID för att hålla koll på vart returvärdet ska skickas.

Då ramverket öppnar för många möjligheter att komma åt känslig information på enheten är det viktigt att användaren också kan neka tillåtelse till vissa tjänster. Då en service anropas första gången av `WebAppActivity` kommer därför en ruta upp som frågar användaren om denna typ av tjänst får tillåtelse att köras.

De tjänster som exponeras genom ramverket och Android-applikationen implementerar gränssnittet `Service`. En `Service` innehåller en metod, `invokeCall()`, som ser till att den metod på tjänsten man anropar anropas om den ges tillåtelse. Det är först i `invokeCall()` som strängen innehållandes metoden evalueras och den implementerade Java-metoden exekveras. Eftersom alla parametrar skickas som en lista med strängar ser metodsyntaxen likadan ut för samtliga implementerade metoder. En konsekvens av detta är att parametrarna kan vara felaktiga eller saknas vid anropet av metoden och därför måste även dessa evalueras under körning.

6.2.1 Lyssnare

Ett lyssnar-mönster har tillämpats där varje tjänst går att lyssna på. Detta för att kunna få tag på metodernas resultat eller felmeddelanden och samtidigt låta dem vara asynkrona. Lyssnaren, av typen `ServiceListener`, sätts i `WebAppActivity` och ansvarar för att samtliga returanrop utförs i tur och ordning tillbaka till ramverket och webb-applikationen.

7 Implementering

I detta avsnitt tas de resulterande produkterna – Android-applikationen och Javascript-ramverket – av projektet upp, deras utformning och funktionalitet. Dessutom avhandlas den webb-applikation som utvecklats för att testa ramverket.

7.1 Android-applikationen

Android-applikationen är, designenligt, i stort sett en webbläsare. Förutom att visa upp webbsidan är det också den som tillhandahåller information från telefonen till webbsidan. De funktioner som finns tillgängliga är:

- Lokaliseringstjänst (GPS)
- Kompass
- Lampa
- Kontakter
- Accelerometrar samt gyroskop
- Vibrationer
- Knapptryckningar

Utöver den funktionalitet som tillhandahålls för webb-utvecklare finns även inbyggd funktionalitet för att känna av om webb-applikationen som besöks använder ramverket, samt stöd för att ladda ned filer från en webb-applikation.

7.1.1 Filuppladdning

För att lägga till funktionalitet och beteende av sidladdning till Androids `WebView` kan utvecklaren definiera en egen subclass av `WebChromeClient` och sätta denna till webb-vyn.

Uppladdning av filer är något som stöds av flera webbläsare men som inte finns som standard i Androids `WebView`. I Cratis där beteendet vid körande av webb-applikationer ska efterlikna det hos den inbyggda webbläsaren i så stor utsträckning som möjligt har därför uppladdnings-funktionalitet implementerats. Vid undersökning av källkoden till klassen `WebChromeClient` fann projektgruppen att en metod, `openFileChooser()`, fanns definierad men med avsikt gömd i dokumentationen med Javadoc-annotationen `@hidden`. Denna dolda metod överskuggades ändå, dock med viss reservationer, eftersom odokumenterad funktionalitet i framtiden kan försvinna eller fungera på annat sätt.

7.2 Ramverket

Ramverkets uppgift är att såväl tillhandahålla som ta emot information från Android-applikationen och webb-applikationen. Det är ett mellanliggande lager

som sköter kommunikationen mellan webbsida och Android-applikation. Detta åstadkoms med hjälp av ett Javascript-ramverk som innehåller funktioner för att anmäla sig som lyssnare till information från Android-applikationen, men också för att skicka data till densamma.

Ramverket distribueras till utvecklare i form av en Javascript-fil, som innehåller alla funktioner, med tillhörande dokumentation. Det är denna Javascript-fil som Android-applikationen söker efter för att avgöra om en webb-applikation skall visas eller ej.

All funktionalitet och alla metoder hos ramverket finns dokumenterat i appendix till denna rapport.

7.3 Webb-applikationen

Utöver ramverket och Android-applikationen har även en webb-applikation utvecklats. Syftet med denna applikation har varit tudelat. Dels har den behövts för att testa om ramverket över huvud taget fungerar: om funktionerna beter sig som de ska och om enheten skickar rätt data. Den kan också vara användbar för att demonstrera hur ramverket är tänkt att användas för såväl potentiella utvecklare som användare.

Webb-applikationen har också varit användbar för att testa prestandan hos ramverket. För att ramverket skall kunna ses som ett reellt alternativ till andra plattformar är det viktigt att prestandan inte är inbyggt sämre än den hos andra plattformar. I den här delen av arbetet har det använts en Android-enhet för att göra enhetstester.

8 Diskussion

Webb-applikationerna som utvecklas i samband med projektet visar att man med hjälp av ramverket Cratis kan skapa webb-applikationer med samma funktionalitet som systemspecifika applikationer. Dessa webb-applikationer kan skapas utan ingående kunskap om de mobila operativsystemens funktionalitet. Det enda som krävs är kunskaper om HTML, CSS, Javascript samt Cratis.

I dagsläget skulle Cratis kunna förenkla för många utvecklare. De skulle inte behöva utveckla till flera plattformar och de skulle kunna sköta distributionen helt själva, två faktorer som är väldigt viktiga hos utvecklare [12]. De befintliga tekniker som undersöktes kan bara lösa högst ett av dessa problem.

Ramverket Cratis skulle vara redundant om skaparna av de mobila operativsystem hade valt att skapa ett liknande ramverk. Som författarna ser det finns det två möjliga anledningar till varför det inte finns ett sådant här ramverk redan, nämligen pengar från försäljning av applikationer och kontroll. Argumentet att det skulle handla om pengar vid försäljning faller dock till föga vid närmare undersökning. Mobil-applikationsförsäljningen genom Apples App Store står för ungefär 2% av Apples inkomster [50, 51, 52]. Vidare tjänar Google väldigt lite pengar på direktförsäljningen av applikationer genom Google Play [53]. Det författarna istället tror är fallet är att de vill ha kontroll. För Apples del handlar det om att erbjuda en komplett produkt; ingen applikation på App Store skall vara av undermålig kvalitet, för då kan användaren förknippa denna undermåliga kvalitet med själva telefonen och operativsystemet. Google tjänar pengar genom annonser [54], och genom att kontrollera hur applikationerna distribueras blir det lättare för Google att visa upp sina annonser.

Vid skrivandet av denna rapport var Cratis implementerat i endast en mobilplattform, Android, detta är en svaghet. Även om de implementerade funktionerna fungerar bra i Android är det enda beviset för att det fungerar i något annat mobilt operativsystem det initiala *proof of concept*-testet där det påvisades att en implementering i IOS skulle vara möjlig. Anledningen till att det bara implementerades på en plattform är att flera inte rymdes inom tidramen för detta projekt. Det är istället en möjlig framtida studie.

En annan svaghet med ramverket är att det i dagsläget har en begränsad mängd funktioner. Anledningen till att antalet funktioner är begränsad är inte att det är omöjligt att implementera mer funktioner, utan snarare att det inte fanns tid att implementera mer funktioner inom detta projektets tidsram.

9 Slutsats

Applikationer till smartphones är en miljardindustri. Att komma in på en sådan marknad och förändra den från grunden är ingen lätt uppgift. I rapporten har några problem med hur marknaden ser ut idag identifierats och ett förslag på en lösning på några av dessa problem presenterats. Men för att dessa lösningar ska kunna slå sig in och få relevans på marknaden tror författarna att det krävs stöd från någon som redan har en stor närvaro på marknaden och ett stort förtroende hos många utvecklare av applikationer till smartphones.

Utbudet av applikationer till smartphones ser idag ut som marknaden för applikationer till persondatorer gjorde på tidigt 2000-tal. Klyftan mellan olika plattformar är stor, och inga av de större plattformarna delar kodbas; detta gör det svårt att skapa applikationer som körs på flera plattformar. Men vad vi har sett de senaste tio åren är en förflyttning från plattformsspecifika programvaror till programvaror som körs i webbläsaren. Författarna menar att denna förändring är oundviklig även på smartphone-sidan; men eftersom smartphones är mycket mer medvetna om sin omgivning – detta med hjälp av sensorer, kamera och mikrofon – är det viktigt att även dessa funktioner kommer webben till del. Om så inte sker kommer plattformsspecifika applikationer att betraktas som förmer än webb-applikationer.

Projektet har visat att de tekniker som utforskats i denna rapport är fullt möjliga för smartphone-plattformarnas ägare att själva implementera. Författarna tror däremot att motivationen för en sådan förändring hos dessa ägare är liten; de incitament som finns i att behålla sina kunder på sin egen plattform är större.

Projektet som avhandlas i den här rapporten är nog inte att se som ett svar på exakt hur en omdaning av marknaden skall kunna ske, men kanske som ett intressant tekniskt koncept att dra nytta av i framtiden.

Referenser

- [1] IEEE, “Systems and software engineering – vocabulary,” *ISO/IEC/IEEE 24765:2010(E)*, pp. 1–418, december 2010. [Online]. Tillgänglig: <http://ieeexplore.ieee.org/servlet/opac?punumber=5733833>
- [2] Android. [Online]. Tillgänglig: <http://android.com>
- [3] IOS. [Online]. Tillgänglig: <http://apple.com/ios/>
- [4] Apple, “App Store Review Guidelines for iOS apps.” [Online]. Tillgänglig: <https://developer.apple.com/appstore/guidelines.html>
- [5] Google, “Android Design.” [Online]. Tillgänglig: <http://developer.android.com/design/style/metrics-grids.html>
- [6] A. Gonsalves, “Apple Launches iPhone Web Apps Directory,” *InformationWeek*, oktober 2007. [Online]. Tillgänglig: <http://www.informationweek.com/news/202401732>
- [7] B. Gardiner, “Developers Rejoice! Apple to Release SDK in February,” *Wired*, oktober 2007. [Online]. Tillgänglig: <http://www.wired.com/epicenter/2007/10/apple-to-rele-1/>
- [8] Javascript. [Online]. Tillgänglig: <https://developer.mozilla.org/en/JavaScript/>
- [9] A. L. Shimpi and B. Klug, “iPhone 4S Preliminary Benchmarks: 800MHz A5, Slightly Slower GPU than iPad 2, Still Very Fast,” *AnandTech*, 2011. [Online]. Tillgänglig: <http://www.anandtech.com/show/4951/>
- [10] A. Qumer and B. Henderson-Sellers, “An evaluation of the degree of agility in six agile methods and its applicability for method engineering,” *Information and Software technology*, vol. 50, no. 4, pp. 280–295, 2007, dOI 10.1016/j.infsof.2007.02.002.
- [11] A. Cockburn, *Agile Software Development: The Cooperative Game*, [elektronisk] 2 ed. Boston, MA: Addison-Wesley Professional, 2006.
- [12] Glocal Intelligence Agency, GIA, “Native or web application?” april 2010.
- [13] A. Charland and B. Leroux, “Mobile application development: web vs. native,” *Communications of the ACM*, vol. 54, no. 5, pp. 49–53, 2011, dOI 10.1145/1941487.1941504.
- [14] Adobe Systems Inc., “Phonegap.” [Online]. Tillgänglig: <http://phonegap.com/about>
- [15] Appcelerator Inc., “Appcelerator.” [Online]. Tillgänglig: <http://www.appcelerator.com/>
- [16] Mozilla Foundation, “Mozilla Boot to Gecko.” [Online]. Tillgänglig: <http://www.mozilla.org/en-US/b2g/>
- [17] Mozilla, “Booting to the Web.” [Online]. Tillgänglig: https://wiki.mozilla.org/Booting_to_the_Web

- [18] WHATWG, W3C, “HTML 5.” [Online]. Tillgänglig: <http://www.w3.org/TR/html5/>
- [19] Sencha Inc., “Sencha Touch.” [Online]. Tillgänglig: <http://www.sencha.com/products/touch/>
- [20] The JQuery Foundation, “Jquery mobile.” [Online]. Tillgänglig: <http://jquerymobile.com/>
- [21] IUI Open Source Project, “Iui.” [Online]. Tillgänglig: <http://www.iui-js.org/>
- [22] Nielsen, “Smartphones Account for Half of all Mobile Phones, Dominate New Phone Purchases in the US,” mars 2012. [Online]. Tillgänglig: <http://blog.nielsen.com/nielsenwire/2012/03/>
- [23] R. Miller, “Response time in man-computer conversational transactions,” in *Proceedings of the December 9-11, 1968, fall joint computer conference, part I*. ACM, 1968, pp. 267–277.
- [24] J. Guynes, “Impact of system response time on state anxiety,” *Communications of the ACM*, vol. 31, no. 3, pp. 342–347, 1988.
- [25] B. Shneiderman, “Response time and display rate in human performance with computers,” *ACM Computing Surveys (CSUR)*, vol. 16, no. 3, pp. 265–285, 1984.
- [26] A. Rushinek and S. Rushinek, “What makes users happy?” *Communications of the ACM*, vol. 29, no. 7, pp. 594–598, 1986.
- [27] H. Yang, E. Tempero, and H. Melton, “An Empirical Study into Use of Dependency Injection in Java,” in *Proceedings of the 19th Australian Conference on Software Engineering*. IEEE Computer Society, mars 2008, pp. 239–247.
- [28] J. Weiskotten, “Dependency Injection,” *Dr.Dobbs Journal*, vol. 31, no. 5, pp. 10–15, maj 2006. [Online]. Tillgänglig: <http://search.proquest.com/docview/202704524?accountid=10041>
- [29] M. Fowler, “Inversion of control containers and the dependency injection pattern,” *Actualizado el*, vol. 23, 2004.
- [30] G. J. Myers, C. Sandler, and T. Badgett, *The Art of Software Testing*. Wiley, 2011.
- [31] JQuery. [Online]. Tillgänglig: <http://jquery.com/>
- [32] D. Leslie, “Using Javadoc and XML to produce API reference documentation,” in *Proceedings of the 20th annual international conference on Computer documentation*. ACM, 2002, pp. 104–109.
- [33] A. Havewala, “The version control process,” *Dr. Dobb’s Journal, São Francisco*, no. 299, 1999.
- [34] Git. [Online]. Tillgänglig: <https://git-scm.com/>
- [35] SVN. [Online]. Tillgänglig: <http://subversion.apache.org/>
- [36] Bitbucket. [Online]. Tillgänglig: <https://bitbucket.org/>

- [37] Android Open Source Project, “Android Developer Tools.” [Online]. Tillgänglig: <http://developer.android.com/guide/developing/tools/adt.html>
- [38] —, “Android Software Development Kit.” [Online]. Tillgänglig: <http://developer.android.com/sdk/index.html>
- [39] Builtwith. [Online]. Tillgänglig: <http://trends.builtwith.com/docinfo/Javascript>
- [40] JSON. [Online]. Tillgänglig: <http://json.org/>
- [41] D. Crockford, “The application/json media type for javascript object notation (JSON),” Internet Requests for Comments, RFC Editor, RFC 4627, juli 2006. [Online]. Tillgänglig: <http://www.rfc-editor.org/rfc/rfc4627.txt>
- [42] —, “JSON: The fat-free alternative to XML,” in *Proc. of XML*, vol. 2006, december 2006. [Online]. Tillgänglig: <http://www.json.org/fatfree.html>
- [43] Gson. [Online]. Tillgänglig: <http://code.google.com/p/google-gson/>
- [44] Roboguice. [Online]. Tillgänglig: <http://code.google.com/p/roboquice/>
- [45] Google Guice. [Online]. Tillgänglig: <http://code.google.com/p/google-guice/>
- [46] Openintents, “Sensorsimulator.” [Online]. Tillgänglig: <http://code.google.com/p/openintents/wiki/SensorSimulator>
- [47] S. Stefanov, *JavaScript patterns*. O’Reilly Media, Inc., 2010.
- [48] W3Techs, “Usage of javascript libraries for websites.” [Online]. Tillgänglig: http://w3techs.com/technologies/overview/javascript_library/all
- [49] Android Open Source Project, “Android Developer Guide, Activities.” [Online]. Tillgänglig: <http://developer.android.com/guide/topics/fundamentals/activities.html>
- [50] Apple, “Apple Reports Fourth Quarter Results (2011).” [Online]. Tillgänglig: <http://www.apple.com/pr/library/2011/10/18Apple-Reports-Fourth-Quarter-Results.html>
- [51] H. Dediu, “App developers receive \$12 for each iOS device sold,” *Asymco*, 2011. [Online]. Tillgänglig: <http://www.asymco.com/2012/02/19/app-developers-get-12-for-each-ios-device-sold/>
- [52] J. Yarow and K. Angelova, “How Much Apple Is Making On The App Store,” *Business Insider*, 2011. [Online]. Tillgänglig: http://articles.businessinsider.com/2011-07-11/tech/29964545_1_app-store-gene-munster-apple
- [53] H. Dediu, “The android income statement,” *Asymco*, 2012. [Online]. Tillgänglig: <http://www.asymco.com/2012/05/14/the-android-income-statement/>

- [54] Google, “Google Announces Fourth Quarter and Fiscal Year 2011 Results.” [Online]. Tillgänglig: http://investor.google.com/earnings/2011/Q4_google_earnings.html

A Use cases

A.1 Gå in i en webb-applikation

Use Case 1	Gå in i en webb-applikation
<i>Primary Actor:</i>	Slutanvändaren
<i>Stakeholders and Interests:</i>	<ul style="list-style-type: none">• Systemet: att tillhandahålla funktionaliteten
<i>Main Success Scenario:</i>	
<ol style="list-style-type: none">1. Användaren: Skriver in en url till en webb-applikation.2. System: Skickar en begäran till webb-applikationen så att den implementerar ramverket.<ol style="list-style-type: none">1. Om ja: Webb-applikationen visas i fullskärm på enheten.2. Om nej: Felmeddelande visas.	

A.2 Gå ur en webb-applikation

Use Case 1	Gå ur en webb-applikation
<i>Primary Actor:</i>	Slutanvändaren
<i>Stakeholders and Interests:</i>	<ul style="list-style-type: none">• Systemet: att tillhandahålla funktionaliteten
<i>Main Success Scenario:</i>	
<ol style="list-style-type: none">1. Användaren: Trycker på “gå ur” knapp när en webb-applikation visas.2. System: “Stänger” webb-applikationsvyn, och visar “hemskrämen”.	

A.3 Hämta godtyckliga filer

Use Case 1	Hämta godtyckliga filer
<i>Primary Actor:</i>	Slutanvändaren
<i>Stakeholders and Interests:</i>	<ul style="list-style-type: none">• Systemet: att tillhandahålla funktionaliteten• Ramverket: hanterar kommunikationen mellan applikationen och användaren• Webb-applikation: gör en fil tillgänglig
<i>Main Success Scenario:</i>	

1. Användare: går in på den sida, eller trycker på att ladda hem en fil.
 2. Webb-applikation: Gör en fil tillgänglig.
 3. Ramverk: Genomför ett anrop till Android applikationen.
 4. System: Promptar användaren vart filen skall sparas, (i Android).
 5. Denna kommer automatiskt att ge tillåtelse om användaren trycker på "spara".
Tillåtelsen kommer ej att sparas.
 1. Om ja: Systemet hämtar filen.
 2. Om nej: ingen fil hämtas.
-

A.4 Ladda upp godtyckliga filer

Use Case 1	Ladda upp godtyckliga filer
<i>Primary Actor:</i>	Slutanvändaren
<i>Stakeholders and Interests:</i>	<ul style="list-style-type: none"> • Systemet: att tillhandahålla funktionaliteten • Ramverket: hanterar kommunikationen mellan applikationen och användaren • Webb-applikation: gör det möjligt att ladda upp en fil
<i>Main Success Scenario:</i>	
<ol style="list-style-type: none"> 1. Webb-applikationen: har en html "file upload" 2. Användare: Trycker på knappen. 3. Ramverk: skickar en förfrågan till Systemet. 4. Systemet: visar en prompt om att användaren skall välja en fil att ladda upp. 5. Användaren: väljer filen och trycker på ladda upp. <ol style="list-style-type: none"> 1. Om ja: System: Skickar filen till webb-applikationen. Via HTML-post. 2. Om nej: ingen fil hämtas. Tillåtelsen kommer ej att sparas. 	

A.5 Lägg till händelser i kalendern

Use Case 1	Lägga till händelse i kalendern
<i>Primary Actor:</i>	Slutanvändaren
<i>Stakeholders and Interests:</i>	<ul style="list-style-type: none"> • Systemet: att tillhandahålla funktionaliteten • Ramverket: hanterar kommunikationen mellan applikationen och användaren • Webb-applikation: tillhandahåller möjlighet att ladda ned en händelse
<i>Main Success Scenario:</i>	

1. Användaren: trycker på en knapp för att lägga till händelse
 2. Ramverket: skickar en förfrågan till systemet
 3. Systemet: visar en prompt för användaren att godkänna
 1. Om ja: Systemet: visar en ny prompt för att lägga till händelsen i användarens kalender.
 1. Om ja: händelse läggs till i användarens kalender.
 2. Om nej: ingen händelse hämtas.
-

A.6 Lyssna på knapptryckningar

Use Case 1	Lyssna på knapptryckningar
<i>Primary Actor:</i>	Slutanvändaren
<i>Stakeholders and Interests:</i>	<ul style="list-style-type: none"> • Systemet: att tillhandahålla funktionaliteten • Ramverket: hanterar kommunikationen mellan applikationen och användaren • Webb-applikation: tillhandahåller möjlighet att lyssna på en knapptryckning
<i>Main Success Scenario:</i>	
<ol style="list-style-type: none"> 1. Användare: öppnar webb-appen 2. Webb-app: skickar förfrågan till ramverket 3. Ramverket: vidarebefordrar förfrågan till systemet 4. Systemet: visar en prompt för användaren att godkänna 5. Systemet: skickar events till webb-appen när användaren trycker på knappar. 	

A.7 Ramverket begär åtkomst till funktion på telefonen

Med funktion så menas ett "objekt", tex GPS, med denna får man tillgång till de metoder som detta objekt har

Use Case 1	Ramverket begär åtkomst till funktion på telefonen
<i>Primary Actor:</i>	Slutanvändaren
<i>Stakeholders and Interests:</i>	<ul style="list-style-type: none"> • Systemet: att tillhandahålla funktionaliteten • Ramverket: hanterar kommunikationen mellan applikationen och användaren • Webb-applikation: tillhandahåller möjlighet att ladda ned en händelse
<i>Main Success Scenario:</i>	

1. Webb-applikation: Skickar en förfrågan till Android-applikationen.
 2. System: Tar emot förfrågan och visar ett meddelande för användare, om åtkomst.
 3. Användare: Klickar ja eller nej om webb-applikationen skall få åtkomst till tillfrågad funktion. Funktionen som webb-applikationen vill ha tillgång till skall visas.
 1. Om ja: Systemet: skickar tillbaka svar för förfrågan. Webb-applikationen får tillgång till funktionen under sessionen.
 2. Om nej: Systemet: skickar ett felmeddelande till webb-applikationen att åtkomst är nekad för önskad funktion..
-

B Dokumentation

Building a web application for using Cratis is simple. Just include `Cratis.js` in your HTML and you are set to use the implemented Cratis functions.

Start the Cratis mobile application, enter the location of your web application and see the framework in action.

B.1 Example

This is an example of using Cratis for binding a hardware button on the device to toggle the flashlight.

```
<!DOCTYPE HTML>
<html>
  <head>
    <script type="text/javascript" src="Cratis.js"></script>
    <script type="text/javascript">
      function toggleFlashlight(){
        cratis.flashlight.toggle({
          success: function (data) {
            document.getElementById("info").innerHTML = "Sucessfully toggled flashlight";
          },
          fail: function (data) {
            document.getElementById("info").innerHTML = "Failed toggling flashlight";
          }
        });
      };

      function menuListen(){
        cratis.keypress.startKeypressListening({
          menu: "true",
          dpad: "false",
          camera: "false",
          success: function(data){
            toggleFlashlight();
          },
          fail: function(){
            document.getElementById("info").innerHTML = "Failed listening";
          }
        });
      };

      function stopListen(){
        cratis.keypress.stopKeypressListening({
          success: function(data){
            document.getElementById("info").innerHTML = "Stopped listening";
          },
          fail: function(){
```

```

        document.getElementById("info").innerHTML = "Failed to stop listening";
    }
});
};
</script>
</head>
<body>
    <button onclick="menuListen()">Listen to menu button</button>
    <button onclick="stopListen()">Stop Listening</button>
    <p id="info"></p>
</body>
</html>

```

B.2 Compass

cratis.compass.getCurrentHeading(params)

Get the current compass heading in degrees.

params A set of key/value pairs that configure the call.

success(data) A function to be called if the call returns successfully.

data Data returned from the successful callback.

heading

Direction that the device is pointing in degrees from 0 to 360.

resultMessage

Detailed message about the result.

fail(data) A function to be called if the call fails. Optional.

data Data returned from the failed callback.

resultMessage

Detailed message about the result.

cratis.compass.startMonitorCompass(params)

At a fixed interval, get the compass heading in degrees.

params A set of key/value pairs that configure the call.

success(data) A function to be called if the call returns successfully.

data Data returned from the successful callback.

heading

Direction that the device is pointing in degrees from 0 to 360.

resultMessage

Detailed message about the result.

fail(data) A function to be called if the call fails. Optional.

data Data returned from the failed callback.

resultMessage

Detailed message about the result.

cratis.compass.stopMonitorCompass(params)

Stops the current monitoring of the compass.

params A set of key/value pairs that configure the call.

success(data) A function to be called if the call returns successfully.

data Data returned from the successful callback.

resultMessage

Detailed message about the result.

fail(data) A function to be called if the call fails. Optional.

data Data returned from the failed callback.

resultMessage

Detailed message about the result.

B.3 Geolocation

cratis.geolocation.getCurrentPosition(params)

Returns the device's current position.

params A set of key/value pairs that configure the call.

success(data) A function to be called if the call returns successfully.

data Data returned from the successful callback.

longitude

Longitude of the current position.

latitude

Latitude of the current position.

altitude

Altitude of the current position.

resultMessage

Detailed message about the result.

fail(data) A function to be called if the call fails. Optional.

data Data returned from the failed callback.

resultMessage

Detailed message about the result.

cratis.geolocation.startMonitorPosition(params)

Monitors the device's position until stopped.

params A set of key/value pairs that configure the call.

minTime The minimum time interval for notifications, in milliseconds. This field is only used as a hint to conserve power, and actual time between location updates may be greater or lesser than this value.

minDistance The minimum distance interval for notifications, in meters. To obtain updates as frequently as possible, set both minTime and minDistance to 0.

success(data) A function to be called if the call returns successfully.

data Data returned from the successful callback.

longitude

Longitude of the current position.

latitude

Latitude of the current position.

altitude

Altitude of the current position.

resultMessage

Detailed message about the result.

fail(data) A function to be called if the call fails. Optional.

data Data returned from the failed callback.

resultMessage

Detailed message about the result.

cratis.geolocation.stopMonitorPosition(params)

Stops the current monitoring of the device's position.

params A set of key/value pairs that configure the call.

success(data) A function to be called if the call returns successfully.

data Data returned from the successful callback.

resultMessage

Detailed message about the result.

fail(data) A function to be called if the call fails. Optional.

data Data returned from the failed callback.

resultMessage

Detailed message about the result.

B.4 Contacts

cratis.contacts.getContacts(params)

Returns one or more contacts from the device address book.

params A set of key/value pairs that configure the call.

query String filtering results that have matching name, email or telephone number. Not implemented yet.

success(data) A function to be called if the call returns successfully.

data Data returned from the successful callback.

contacts

Results as an array of contacts. Containing name, telephone numbers and email addresses.

resultMessage

Detailed message about the result.

fail(data) A function to be called if the call fails. Optional.

data Data returned from the failed callback.

resultMessage

Detailed message about the result.

cratis.contacts.addContact(params)

Adds a new contact to the device's address book.

params A set of key/value pairs that configure the call.

firstName Given name of contact.

lastName Surname of contact.

email Email address of contact.

phone Telephone number of contact.

success(data) A function to be called if the call returns successfully.

data Data returned from the successful callback.

resultMessage

Detailed message about the result.

fail(data) A function to be called if the call fails. Optional.

data Data returned from the failed callback.

resultMessage

Detailed message about the result.

B.5 Keypress

cratis.keypress.startKeypressListening(params)

Starts listening to key presses for keys of choice until stopped.

params A set of key/value pairs that configure the call.

camera Boolean. Set to true if the camera key should be listened to.

dpad Boolean. Set to true if the directional pad keys should be listened to.

menu Boolean. Set to true if the menu key should be listened to.

success(data) A function to be called if the call returns successfully.

data Data returned from the successful callback.

key

Name of the pressed key.

keycode

An integer identifier, identifying the pressed key.

resultMessage

Detailed message about the result.

fail(data) A function to be called if the call fails. Optional.

data Data returned from the failed callback.

resultMessage

Detailed message about the result.

cratis.keypress.stopKeypressListening(params)

Stops listening to key presses.

params A set of key/value pairs that configure the call.

success(data) A function to be called if the call returns successfully.

data Data returned from the successful callback.

resultMessage

Detailed message about the result.

fail(data) A function to be called if the call fails. Optional.

data Data returned from the failed callback.

resultMessage

Detailed message about the result.

B.6 Vibration

cratis.keypress.vibrate(params)

Makes the device vibrate.

params A set of key/value pairs that configure the call.

milliseconds Length of vibration.

repeat Integer with amount of times the vibration should repeat.

success(data) A function to be called if the call returns successfully.

data Data returned from the successful callback.

resultMessage

Detailed message about the result.

fail(data) A function to be called if the call fails. Optional.

data Data returned from the failed callback.

resultMessage

Detailed message about the result.

B.7 Calendar

cratis.calendar.addCalendarEvent(params)

Adds an event to de device's calendar.

params A set of key/value pairs that configure the call.

startTime Start time of the event.

endTime End time of the event.

title Name of event.

place Location of event. Optional.

description Additional description of event. Optional.

success(data) A function to be called if the call returns successfully.

data Data returned from the successful callback.

resultMessage

Detailed message about the result.

fail(data) A function to be called if the call fails. Optional.

data Data returned from the failed callback.

resultMessage

Detailed message about the result.

cratis.calendar.isUserFree(params)

Checks the device's calendar if an interval is free.

params A set of key/value pairs that configure the call.

startTime Start of interval to check if calendar is free.

endTime End of interval to check if calendar is free.

success(data) A function to be called if the call returns successfully.

data Data returned from the successful callback.

isFree

Boolean value if user is free or not.

resultMessage

Detailed message about the result.

fail(data) A function to be called if the call fails. Optional.

data Data returned from the failed callback.

resultMessage

Detailed message about the result.

cratis.calendar.getCalendarNames(params)

Get the names of the calendars on the device.

params A set of key/value pairs that configure the call.

success(data) A function to be called if the call returns successfully.

data Data returned from the successful callback.

nameList

Comma separated list of calendar names.

resultMessage

Detailed message about the result.

fail(data) A function to be called if the call fails. Optional.

data Data returned from the failed callback.

resultMessage

Detailed message about the result.

B.8 Flashlight

cratis.flashlight.turnOn(params)

Turn on the device's flashlight.

params A set of key/value pairs that configure the call.

success(data) A function to be called if the call returns successfully.

data Data returned from the successful callback.

resultMessage

Detailed message about the result.

fail(data) A function to be called if the call fails. Optional.

data Data returned from the failed callback.

resultMessage

Detailed message about the result.

cratis.flashlight.turnOff(params)

Turn off the device's flashlight.

params A set of key/value pairs that configure the call.

success(data) A function to be called if the call returns successfully.

data Data returned from the successful callback.

resultMessage

Detailed message about the result.

fail(data) A function to be called if the call fails. Optional.

data Data returned from the failed callback.

resultMessage

Detailed message about the result.

cratis.flashlight.toggle(params)

Toggle the device's flashlight on or off.

params A set of key/value pairs that configure the call.

success(data) A function to be called if the call returns successfully.

data Data returned from the successful callback.

resultMessage

Detailed message about the result.

fail(data) A function to be called if the call fails. Optional.

data Data returned from the failed callback.

resultMessage

Detailed message about the result.