Val av metod och ramverk vid utveckling av mobila applikationer

Alexander Gallen

Kandidatarbete i Datateknik

Handledare: Annamari Soini

Fakulteten för Naturvetenskaper och Teknik

Åbo Akademi

Referat

Det finns flera olika sätt att utveckla mobilapplikationer på. De vanligaste metoderna är att göra en nativ mobilapplikation, en webbapplikation eller en hybrid mobilapplikation. Inom utvecklingen av hybrida mobilapplikationer finns det flera olika populära ramverk som kan användas.

Denna avhandling fokuserar på att jämföra de vanligaste metoderna och ramverken för att ge läsaren en klar bild över de relevanta alternativen som finns, samt ge en helhetsbild över i vilka fall det kan löna sej att använda de olika alternativen.

Resultatet är en jämförelse mellan de olika metoderna och de valda ramverken för att ge en klar bild över deras för- och nackdelar för att hjälpa till vid dessa val då man börjar utveckla sin mobilapplikation.

Nyckelord

Hybrid mobilapplikation, Ionic, Appcelerator Titanium, Javascript, HTML, Webbapplikation

Abstract

There are various methods for developing applications for mobile devices, the three main ones are native applications, web applications and hybrid applications. The focus of this thesis is to give a good understanding of what type of application to develop depending on the specifications of the application. This includes the choice of what framework to use if you choose to create a hybrid application.

The most common frameworks and methods are compared to each other to give an overview of when to choose what method and framework.

Keywords

Mobile application, Ionic, Appcelerator Titanium, Web application, Hybrid application, Mobile app development

Innehållsförteckning

1. Inledning

1.1 Bakgrund

Mobila applikationer är en marknad som har ökat markant under de senaste åren [1]. I början var applikationer främst för smarta mobiltelefoner inom ett fåtal operativsystem, nu för tiden finns det applikationer för smarta klockor, mobiltelefoner, läsplattor och bilars media-system [2].

Oftast utvecklar man applikationer för flera olika plattformar eller operativsystem. I de flesta fall kommer applikationen att användas både på mobiltelefoner och läsplattor, samt möjligen smarta klockor. För att detta ska vara möjligt inom ett specifikt operativsystem måste man ta i beaktande apparaternas skärm-resolution och möjliga variationer i prestanda hos olika modeller. För att sedan vidare kunna nå majoriteten av användarna krävs det att applikationen utvecklas för flera olika operativsystem, där de vanligaste är iOS, Android och Windows Phone.

Från början har applikationer utvecklats nativt för de olika operativsystemen. Detta tillvägagångssätt innebär att man oftast behöver olika mjukvaruutveckling team för de olika plattformarna eftersom de är skriva på olika programmeringsspråk. Exempel på detta är C# för Windows Phone, Java för Android och Swift för iOS.

Utöver detta klassiska sätt att utveckla applikationer finns nuförtiden ramverk för utveckling av applikationer för flera operativsystem, så kallade hybrida mobilapplikationer. Till de mest lovande ramverken hör Ionic, Intel XDK, Appcelerator Titanium, JQuery Mobile, Kendo UI, Onsen UI, Mobile Angular UI [3] [4].

1.2 Syfte och metod

Syftet med arbetet är att ge en klar bild över hur man ska utveckla en mobilapplikation beroende på vilken typs applikation det är frågan om. Detta görs genom att gå igenom de vanligaste metoderna för utveckling av mobila applikationer, samt de mest relevanta ramverken då det kommer till utveckling av hybrid-applikationer, för att ge en klar bild över hur det lönar sig att börja utveckla applikationen. Grunden för detta arbete är relevant tidigare forskning, diskussion och artiklar.

1.3 Problemformulering

Det finns flera olika sätt att utveckla mobilapplikationer på och det kan vara svårt att hitta den lösningen som passar ens egna projekt bäst. De olika metoderna och ramverken utvecklas snabbt vilket betyder att informationen ofta är mycket utspridd eller irrelevant. Valet av metod och ramverk leder till stora skillnader i applikationerna både i utvecklingsfasen samt i den slutgiltiga produkten och dess uppehåll.

1.4 Avgränsningar

I detta arbete kommer hybrid mobilapplikationsutvecklings ramverken att avgränsas till Ionic och Appcelerator Titanium. Dessa har valts eftersom de är två populära ramverk som tekniskt sett skiljer sej mycket från varandra. Dessa kommer att jämföras mot nativa applikationer samt webapplikationer.

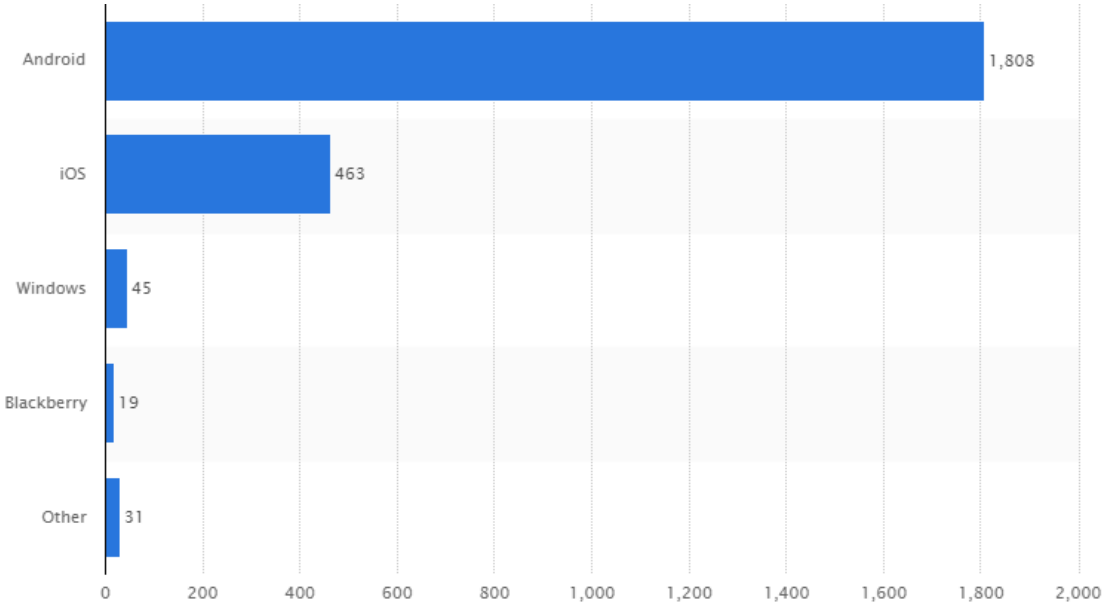
Fokuset kommer att ligga på utveckling av applikationer för den finska- och internationella marknaden. Detta innebär utveckling av applikationer för iOS, Android och Windows Phone, vilket har tagits i beaktande vid valet av ramverk.

Dessutom kommer det inte att gås noggrannare in på de olika applikationstyperna, utan de klassas som större helheter för att ge en klarare överblick över när man ska överväga de olika ramverken och mjukvaruutvecklingsmetoderna.

2. Teknisk bakgrund

2.1 Mobila plattformar

De vanligaste operativsystemen hos mobila enheter idag är iOS och Android. Globalt var 96 % av smarttelefonerna som var i bruk år 2015 antingen iOS eller Android telefoner (*figur 1*).



Figur 1: Antal mobiltelefoner i bruk (miljoner)/plattform (2015) [5].

I Finland är Windows Phone ännu vanligt förekommande, med 17 % av marknaden år 2015. Samma år hade Android 32 % och iOS 13,2 % [6]. Med detta i beaktande måste man även utveckla sina applikationer för Windows Phone för att ha en realistisk chans att slå genom på den finska marknaden.

2.1.1 Android

Android är ett operativsystem för mobila enheter som är utvecklat av Google. Android är baserad på Linuxkärnan och programmeringsspråket för nativa applikationer är Java.

Operativsystemet är designat huvudsakligen för mobila enheter med pekskärm, och finns förutom till smarta mobiltelefoner och läsplattor även tillgängligt för televisioner (Android TV), bilar (Android Auto) och smarta klockor (Android Wear) [7].

2.1.2 iOS

iOS är ett operativsystem utvecklat av Apple Inc. och finns endast tillgänglig hos Apple produkter (t.ex. iPhone, iPad).

Nativa applikationer för operativsystemet skrivs i C, C++, Objective-C eller Swift. Swift är ett programmeringsspråk som är utvecklat av Apple Inc. för att underlätta utvecklande av applikationer för iOS [8].

2.1.3 Windows Phone

Windows Phone är ett operativsystem utvecklat av Microsoft [9].

För att utveckla nativa applikationer för Windows Phone kan man använda antingen C#, Visual Basics.NET eller C++. Microsoft håller dessutom på att utveckla en

mellanprogramvara som kallas Windows Bridge, som ska möjliggöra enkel portning av Android eller iOS applikationer till Windows Phone.

2.2 Olika typer av mobilapplikationer

Det finns olika sätt att utveckla mobilapplikationer på. De tre vanliga metoderna är plattformsoberoende (nativa) mobilapplikationer, webbapplikationer och hybrida mobilapplikationer.

2.2.1 Nativa mobilapplikationer

Nativa mobilapplikationer är applikationer som utvecklas specifikt för en viss plattform. Denna typs applikationer är de snabbaste, säkraste och ger applikationen möjlighet att använda enhetens alla inbyggda funktioner [10].

För att utveckla en nativ applikation för en plattform krävs kunskaper i det operativsystemets språk. För Android krävs Java kunskaper, för iOS Objective-C eller Swift och för Windows Phone C#, Visual Basics.NET eller C++. Detta medför samtidigt den största nackdelen då det kommer till utveckling av nativa mobilapplikationer: det är frågan om skilda applikationer för de olika plattformarna. Vad detta oftast innebär i praktiken då man utvecklar en applikation för flera plattformar är att man behöver skilda utvecklings team för de olika operativsystemen. Ett team för Android som kan Java och Androids SDK, ett för iOS som behärskar Objective-C/Swift och kan deras SDK och ett team för utveckling av applikationen till Windows Phone som kan C#/Visual Basics. Med detta medföljer både ökad kostnad och tid för utvecklingen, eftersom det krävs olika team för de olika operativsystemen både under utvecklingen samt uppehållet av applikationen [11]. Priset för utvecklingen av en vanlig liten eller medelstor applikation för iOS och

Android kan vara det dubbla gentemot motsvarande hybrida applikation, detta utan att ta i beaktande tilläggskostnader för utveckling för Windows Phone [12].

Den klara fördelen hos nativa applikationer är prestanda och tillgång till all funktionalitet hos enheten. Eftersom nativa applikationer är skrivna på operativsystemens egna språk kommer prestandan alltid att vara överlägsen den hos webbapplikationer och hybrida applikationer. Detta ger även tillgång till all funktionalitet som operativsystemet och enheten har att erbjuda utvecklaren, t.ex. Kameran, notifikationer, kalendern, kontakter m.m. [10] [13] [14]

Säkerheten hos applikationer är något som blir mer relevant hela tiden i takt med den explosionsartade ökningen av applikationer och deras popularitet. Både nativa och hybrida/webbapplikationer har säkerhetsluckor, speciellt om koden bakom applikationen inte är utmärkt. Nativa applikationer har dock en klar fördel då det kommer till säkerhet eftersom det är möjligt att lagra information lokalt på enheterna. Dessutom finns det mycket dokumentation om möjliga säkerhetshot och det är i operativsystemföretagens intressen att göra deras applikationer och plattformar säkra genom att uppdatera systemet och informera utvecklare om optimal praxis då det kommer till utveckling ur ett säkerhetsperspektiv [15].

Distributionen av nativa applikationer sker via digitala distributionsplatser. De vanligaste är Google Play för Android, Apple App Store för iOS och Windows Phone Store för Windows Phone.

2.2.2 Webbapplikationer

Webbapplikationer är i sin enklaste form webbsidor som är anpassade för mobila enheter. En stor del av webbsidorna har nuförtiden mobila versioner som är anpassade för de mobila enheternas skärmstorlekar för att visa informationen på ett klarare sätt på mobila enheter och underlätta navigeringen [16].

Fördelar med webbapplikationer är att de är enklare, billigare och snabbare att utveckla. För att utveckla en webbapplikation krävs erfarenhet av vanliga webbtekniker som HTML (HyperText Markup Language), CSS (Cascading Style Sheets) och JavaScript. Detta innebär att samma utvecklings team som står bakom ett företags webbsida kan utveckla en version för mobila enheter relativt snabbt och kostnadseffektivt.

Webbapplikationer går att nås från varje plattform som har tillgång till internet och en webbläsare. Fastän detta i allmänhet är ett stort plus för webbapplikationer kan det även få negativa frekvenser. För att nå en webbapplikation måste man ha internet uppkoppling, dessutom kan applikationen se olika ut på olika plattformar beroende på vilken webbläsare plattformen i fråga använder.

Nackdelar hos webbapplikationer är att de inte har tillgång till operativsystemets SDK. Detta innebär att det inte är möjligt att komma åt enhetens kamera, notifikationer, kontakter, kalender, m.m. [10]

Webbapplikationer är dessutom långsammare än nativa applikationer eftersom största prestanda faktorn är bandbredden på det mobila nätet. Applikationen kan inte använda enhetens processor eller lagrings möjligheter på ett optimalt sätt [17]. Med detta kommer även säkerhetsrisker eftersom stora delar känslig information måste överföras med JavaScript. Till de vanligaste säkerhetsriskerna hos webbapplikationer hör Cross-site Scripting (XSS) och SQL-injektion [18].

Den märkbara skillnaden för användaren är att webbapplikationer inte hittas på digitala distributionsplatser. Det betyder att användaren måste hitta applikationen på internet, via sökmotorträffar, reklam eller sociala medier. Det gör också att användaren inte i första hand tänker på webbapplikationer som applikationer, utan som webbsidor som är utvecklade för mobila enheter.

2.2.3 Hybrida mobilapplikationer

Den nyaste metoden för utveckling av applikationer för mobila enheter är så kallade hybrida mobilapplikationer. För användaren ser de ut exakt som nativa applikationer men själva utvecklingen påminner mera den av webbapplikationer. Principen bakom hybrida applikationer är att man använder webbtekniker för att skapa en applikation som är kompatibel med flera olika plattformar.

Det finns flera olika ramverk som tillåter en att utveckla hybrida applikationer och grund principen hos dem är att de sköter om kompatibiliteten med de olika plattformarna medan utvecklaren kan fokusera på det väsentliga hos applikationen. PhoneGap/Cordova [19] gör detta genom att skapa en nativ applikation vars gränssnitt är en webbläsarvy som täcker hela skärmen på enheten och visar den utvecklade webbapplikationen för användaren [20]. Detta möjliggör användning diverse funktionaliteter hos enheten/plattformen [21] samtidigt som utvecklaren i princip endast behöver göra en webbapplikation [22]. Fördelar med en lösning som PhoneGap är flera. Om man redan har en webbsida som är mobilvänlig kan man i princip skapa en mobilapplikation direkt med hjälp av detta ramverk. Dessutom går applikationen att uppdatera när som helst utan att man måste få nya versionen till de digitala distributionsplatserna, det räcker att man uppdaterar webbsidan. Största nackdelen är att prestandan lider av samma orsaker som webbapplikationerna samt att internetuppkoppling är ett måste för att applikationen ska fungera [23].

Ett annat sätt för ett ramverk att uppnå kompatibilitet hos flera plattformar är att som PhoneGaps största konkurrent, Appcelerator Titanium, utveckla en Javascript applikation som kompileras om till de olika plattformarnas respektive programmeringsspråk [24]. Ett problem med detta från utvecklarens synvinkel är att applikationen och gränssnittet görs exklusivt med Javascript. De flesta utvecklare som bemästrat Javascript och funderar på att skapa en hybrid mobilapplikation är webbutvecklare och vana att använda Javascript tillsammans med HTML och CSS för att skapa snygga gränssnitt med smidig funktionalitet [25]. Till Titaniums fördelar

hör prestandan (i jämförelse med PhoneGap, inte nativa applikationer) samt att det alltid kommer att kännas som en nativ applikation eftersom ramverket har konverterat din Javascript kod till en äkta plattformsspecifik nativ applikation.

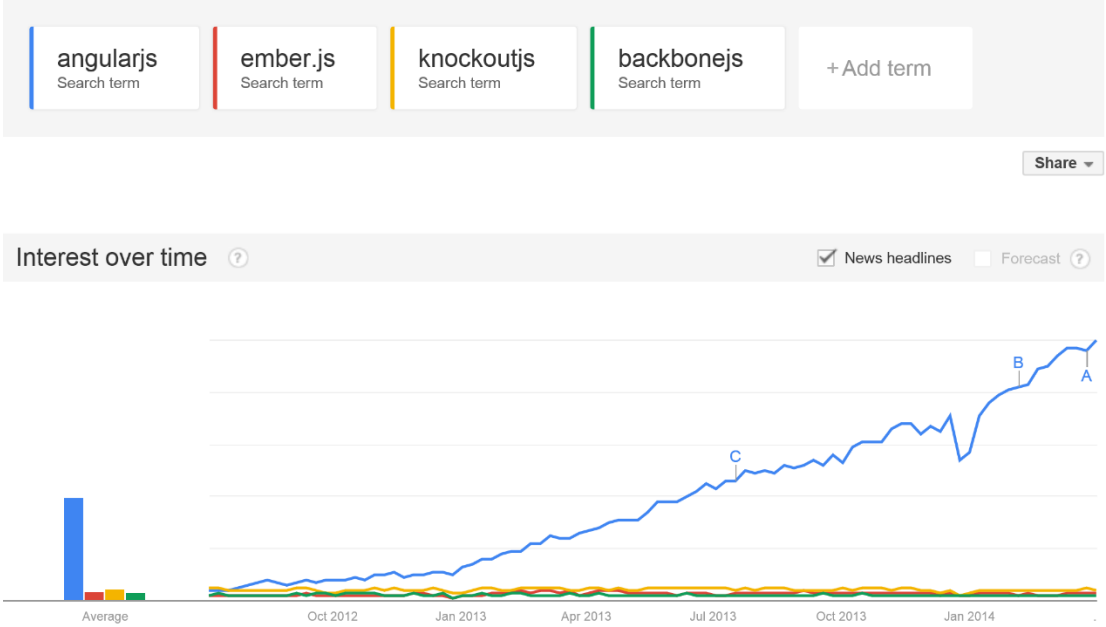
2.3 Ramverk för hybrida mobilapplikationer

Det finns flera olika ramverk då det kommer till utveckling av mobila applikationer och ett par grundprinciper gicks genom i föregående kapitel. Fokuset kommer att ligga på två ramverk, Ionic och Appcelerator Titanium, som skiljer sig mycket från varandra.

2.3.1 Ionic

Ionic ramverket lanserades som färdig produkt i Maj 2015 efter att ha varit i betatestnings skede i lite över ett år [26]. Det är alltså frågan om ett relativt nytt ramverk då det kommer till hybrida mobilapplikationer. Det är en av de populäraste ramverken för tillfället och över 2 miljoner applikationer har redan utvecklats med hjälp av Ionic, de populäraste är Sworkit, Pacifica och ChefSteps [27]. Sworkit är den populäraste applikationen utvecklad med Ionic och den har sammanlagt över 14 miljoner nerladdningar på de digitala distributionsplatserna.

Ionic är ett ramverk som låter utvecklaren skapa en webapplikation med hjälp av Javascript, HTML och CSS för att sedan ha plattformsspecifika applikationer skapade med Cordova. Det som skiljer Ionic från de flesta andra ramverk som är Cordovabaserade och litar på webbteknikkunskaper från utvecklaren, är att AngularJS ramverket används. AngularJS är ett Javascript ramverk lanserat av Google som har blivit mycket populärt under de senaste åren [28].



Figur 2: AngularJS popularitet [29].

Ionics fördelar är i princip samma som för de flesta PhoneGap/Cordova baserade ramverk. Utöver dessa är användning av AngularJS en klar fördel som sticker ut i mängden då man jämför ramverk som är baserade på utveckling med hjälp av webbtekniker. Eftersom AngularJS är så populärt är chansen stor att möjliga utvecklare redan har kunskaper om detta Javascript ramverk. AngularJS möjliggör även minimal DOM-manipulation hos Ionic applikationer vilket leder till bättre prestanda hos applikationen då hårdvaruaccelererade övergångar är möjliga [30].

Ionic har en mycket enkel inlärningskurva och själva utvecklingen är långt densamma som den hos webbsidor eller webbapplikationer. Det fungerar även som ett CSS ramverk och Javascript bibliotek vilket underlättar utvecklingen av vissa vanligt förekommande element och gränssnittsstilar som är anpassade specifikt för att se bra ut på mobila enheter och applikationer av denna typ [31].

Eftersom det är frågan om ett av de populäraste ramverken finns det förutom en omfattande dokumentation mycket stöd att få ifall möjliga problem uppstår. Den stora användargruppen gör även att utvecklingen av ramverket är kontinuerlig och

Ionic 2 är för tillfället i betatestningsskede. Med denna uppdatering av ramverket kommer stöd för AngularJS 2, förbättringar i prestandan m.m.

2.3.2 Appcelerator Titanium

Appcelerator Titanium är ett ramverk som när det lanserades 2008 var designat för att utveckla plattformsoberoende applikationer för stationära datorer. Ett år senare utvidgades projektet för att möjliggöra utveckling av mobilapplikationer och har sedan dess varit ett populärt alternativ då det kommer till val av ramverk för utveckling av mobilapplikationer.

Utvecklare som vill använda Titanium måste ha kunskap av Javascript, samt en del andra tekniker och språk som inte hör till de vanliga webbteknikerna, inkluderande XML för utformning av gränssnittet. Det har en högre inlärningskurva än Ionic eftersom de flesta Javascript utvecklare har en webbutvecklingsbakgrund där Javascript används i kombination med CSS och HTML, en praxis som Titanium inte följer eftersom det är strikt Javascript kod som kompileras.

Ramverk som Titanium har en prestanda fördel mot PhoneGap/Cordova baserade ramverk eftersom Titanium och motsvarande ramverk kompilerar den skrivna Javascript koden till nativ kod för de olika plattformarna [32].

3. Jämförelser

I detta kapitel framställs tabeller över för- och nackdelar hos både utvecklingsmetoderna och ramverken för de hybrida applikationerna.

Jämförelse av de olika metoderna för utveckling av mobila applikationer för flera plattformar:

	Metoder		
	Nativa mobilapplikationer	Webbapplikationer	Hybrida mobilapplikationer
Utseende	Bra, känns rätt för respektive plattform.	Bra, kan kännas lite fel för plattformen men märks inte då man är i webbläsaren.	Bra, vissa knappar och ikoner kan vara underligt placerade eller av oönskad storlek.
Prestanda	Optimal.	Beror på internetanslutningen. Svårt att hantera stor data.	Beror på ramverket. Sämre än nativ men kan komma nära.
Kostnad	Dyr. Dyrare ju fler plattformar applikationen ska köra på.	Billig. Webbutvecklare i genomsnitt billigare. Samma kod för alla plattformar.	Billig. Samma kod för alla plattformar, lite dyrare än webbapplikationer.
Utvecklingstid	Lång. Måste utveckla skilda applikationer.	Kort.	Kort.
Distribution	Digital distributionsplatser.	Internet.	Digitala Distributionsplatser.
Plattformsoberoende	Nej.	Jo.	Jo.
Enhetens funktionaliteter	Tillgång till allting.	Tillgång till lokaliseringsuppgifter och viss lokal lagring.	Beror på ramverket. Tillgång till största delen av funktionaliteterna.
Anslutning	Online och off-line.	Online.	Online och off-line.
Krav hos utvecklare	C#/Swift/Java/Objective-C	Javascript, HTML och CSS.	Beror på ramverk. Kan välja något man har erfarenhet av.
Stöd	Stöd av företagen bakom operativsystemen samt stora grupper utvecklare och bra dokumentation.	Mycket dokumentation och många utvecklare med år av erfarenheter av webbtekniker.	De populäraste ramverken har bra, i värsta fall kan de vara bristande.

Jämförelse av för- och nackdelar hos ramverken:

	Ramverk	
	Ionic	Appcelerator Titanium
Utseende	Bra.	Bra, nativ känsla.
Prestanda	Bra, så länge inte mycket data ska behandlas.	Bra, kan komma nära nivån av helt nativa applikationer. Komprimeringen avgör.
Kostnad	Gratis.	Gratis.
Utvecklingskostnad	Relativt låg, beror på utvecklarnas tidigare erfarenheter.	Relativt låg, beror på utvecklarnas tidigare erfarenheter.
Utvecklingstid	Kort, snabbt att göra prototyper.	Kort, snabbt att göra prototyper. Kan behöva skriva en del plattformsspecifik kod för att det ska fungera på olika plattformar.
Inlärningskurva	Enkel.	Enkel, kan kräva tid att vänja sig med strukturen på Javascripten.
Distribution	Digital distributionsplatser.	Digital distributionsplatser.
Plattformsoberoende	Jo.	Jo, de vanligaste plattformarna (inkluderar iOS, Android, Blackberry, Windows Phone)
Enhetens funktionaliteter	Tillgång till största delen av funktionaliteterna.	Tillgång till största delen av funktionaliteterna.
Anslutning	Online och off-line.	Online och off-line.
Krav hos utvecklare	Javascript, HTML, AngularJS och CSS.	Javascript, XML.
Stöd	Största användargruppen bland ramverken. Bra forum och dokumentation.	Aktiv användargrupp. Bra dokumentation

4. Sammanfattning

4.1 Val av metod

Eftersom alla tre metoder har för- och nackdelar beror valet av metod långt på typen av applikation som ska utvecklas. Webbapplikationer ses ofta inte som applikationer ur användarens synpunkt och är en mindre relevant lösning nuförtiden då hybrida mobilapplikationer håller på ta över den delen av marknaden som tidigare bestod exklusivt av webbapplikationer.

Ifall applikationen kommer att kräva mycket resurser av enheten och prestandan spelar en central roll kommer nativa applikationer alltid att vara överlägsna hybrida- och webbapplikationer. Detta märks tydligt då man undersöker de populäraste applikationerna på marknaden. Facebook, Spotify och Instagram är alla exempel på applikationer som är utvecklade nativt för de olika plattformarna.

Största orsakerna att utveckla applikationer med hjälp av ett av de flera hybrida ramverken är hastigheten med vilken man får applikationen gjord samt det låga priset i jämförelse med nativ applikationsutveckling.

Det lönar sej dock att ta i beaktande att hybrida applikationer blir vanligare hela tiden. Ramverken utvecklas konstant vilket leder till mindre skillnader i prestanda gentemot nativa applikationer.

4.2 Val av ramverk

I princip är ramverkens roll den samma oberoende om vilket ramverk man väljer. Sättet de löser kompatibilitetsproblemet varierar men slutprodukten från användarens synpunkt är mycket lika.

Det viktigaste att tänka på när man väljer ramverk för utvecklingen av sin mobilapplikation är ifall det är någon mycket specifik funktionalitet man behöver

som endast några enstaka ramverk erbjuder. Utöver det kommer det främst till vilket tillvägagångssätt man vill ta när man börjar utveckla applikationen. Ifall webbtekniker är bekanta kan det löna sej att satsa på Ionic eller något annat PhoneGap/Cordova baserat ramverk, och i annat fall kommer de andra ramverken som Titanium att ge lite bättre prestanda.

5. Slutsats

Det finns inget rätt eller fel då det kommer till val av metod och ramverk, men det finns alternativ som är bättre och sämre lämpade för utveckling av specifika typer av applikationer. Vikten hos de olika applikationskraven måste noggrant övervägas för att balansen mellan kostnad, prestanda, utseende och användarvänlighet ska överensstämma med kravspecifikationerna så bra som möjligt. Efter det återstår bara att välja den metod och det ramverk som bäst överensstämmer med detta och utvecklarnas färdigheter.

6. Referenser

[1] M. Valtari, ”Someco,” 2 12 2015. [Online]. Available: http://someco.fi/blogi/sosiaalisen-median-ja-alypuhelinten-kaytto-suomessa-vuonna-2015/. [Använd 6 4 2016].

[2] J. Tarkoma, ”Tilastokeskus,” 26 11 2015. [Online]. Available: http://www.stat.fi/til/sutivi/2015/sutivi_2015_2015-11-26_tie_001_fi.html. [Använd 6 4 2016].

[3] "Mozrif", ”The iPhone App Review,” 21 1 2016. [Online]. Available: http://www.theiphoneappreview.com/2016/01/top-hybrid-mobile-app-development-frameworks-for-2016/. [Använd 6 4 2016].

[4] R. Patil, ”Jaxenter,” 22 1 2016. [Online]. Available: http://www.theiphoneappreview.com/2016/01/top-hybrid-mobile-app-development-frameworks-for-2016/. [Använd 6 4 2016].

[5] ”Statista, The Statistics Portal,” [Online]. Available: http://www.statista.com/statistics/385001/smartphone-worldwide-installed-base-operating-systems/. [Använd 6 4 2016].

[6] P. Pitkänen, ”Digitoday,” 16 2 2016. [Online]. Available: http://www.digitoday.fi/mobiili/2016/02/16/la-tuijota-myyntilistoja--naita-puhelimia-suomi-oikeasti-kayttaa/20161762/66. [Använd 6 4 2016].

[7] ”Android,” Google, 2016. [Online]. Available: https://www.android.com/. [Använd 6 4 2016].

[8] ”Swift: Developer,” Apple Inc., 2016. [Online]. Available: https://developer.apple.com/swift/. [Använd 6 4 2016].

[9] ”Windows: Phone,” Microsoft, 2016. [Online]. Available: https://www.microsoft.com/en-us/windows/phones. [Använd 6 4 2016].

[10] M. Korf och E. Oksman, ”Salesforce Developers,” Salesforce, 4 2015. [Online]. Available: https://developer.salesforce.com/page/Native,_HTML5,_or_Hybrid:_Understanding_Your_Mobile_Application_Development_Options. [Använd 6 4 2016].

[11] M. Pronschinske, ”DZone: Mobile Zone,” 16 6 2014. [Online]. Available: https://dzone.com/articles/state-native-vs-web-vs-hybrid. [Använd 6 4 2016].

[12] B. Kohan, ”Comentum,” 26 1 2015. [Online]. Available: http://www.comentum.com/phonegap-vs-native-app-development.html. [Använd 6 4 2016].

[13] ”July Rapid: A July Systems Services Studio,” [Online]. Available: http://julyrapid.com/hybrid-vs-native-mobile-app-decide-5-minutes/. [Använd 6 4 2016].

[14] J. Badalian, ”MobileSmith: Your Quickest Path from Idea to App,” 20 5 2015. [Online]. Available: https://www.mobilesmith.com/html5-vs-native-debate-is-over/. [Använd 6 4 2016].

[15] H. Jagtap, ”Appvigil,” 11 6 2015. [Online]. Available: https://www.appvigil.co/blog/2015/06/native-vs-hybrid-apps-security-aspects-of-each-of-the-apps/. [Använd 6 4 2016].

[16] D. Das, ”Knowledge Stack,” 23 5 2015. [Online]. Available: http://www.knowstack.com/web-app-vs-native-app-vs-hybrid-app/. [Använd 6 4 2016].

[17] G. Gruman, ”InfoWorld,” 15 12 2015. [Online]. Available: http://www.infoworld.com/article/3012146/web-applications/native-apps-have-crushed-web-apps.html. [Använd 6 4 2016].

[18] ”Wikipedia: Webb Application Security,” [Online]. Available: https://en.wikipedia.org/wiki/Web_application_security. [Använd 6 4 2016].

[19] B. LeRoux, ”Adobe PhoneGap,” Adobe, 19 3 2012. [Online]. Available: http://phonegap.com/blog/2012/03/19/phonegap-cordova-and-whate28099s-in-a-name/. [Använd 10 4 2016].

[20] A. Trice, ”Tricedesigns,” 22 3 2012. [Online]. Available: http://www.tricedesigns.com/2012/03/22/phonegap-explained-visually/. [Använd 10 4 2016].

[21] ”Cordova,” Apache, [Online]. Available: http://cordova.apache.org/docs/en/3.5.0/guide/support/index.html#Platform%20Support. [Använd 10 4 2016].

[22] V. Panangipalli, ”SAP: Community Network,” 27 7 2014. [Online]. Available: http://scn.sap.com/community/developer-center/mobility-platform/blog/2014/07/27/what-is-cordova-and-how-does-it-work. [Använd 10 4 2016].

[23] ”[x]cube LABS,” 1 5 2015. [Online]. Available: http://www.xcubelabs.com/our-blog/phonegap-vs-titanium-comparison/. [Använd 10 4 2016].

[24] C. Mims, ”MIT Technology Review,” 17 6 2011. [Online]. Available: https://www.technologyreview.com/s/424328/rise-of-the-hybrid-mobile-app/. [Använd 10 4 2016].

[25] D. Gaic, ”Stackoverflow,” 30 12 2015. [Online]. Available: http://stackoverflow.com/questions/34523087/sencha-vs-ionic-vs-jquery-mobile-vs-appcelerator-titanium. [Använd 10 4 2016].

[26] ”Ionic,” [Online]. Available: http://ionic.io/about. [Använd 10 4 2016].

[27] ”Ionic Showcase,” Ionic, [Online]. Available: http://showcase.ionicframework.com/. [Använd 10 4 2016].

[28] M. Asay, ”InfoWorld,” 27 2 2015. [Online]. Available: http://www.infoworld.com/article/2890272/javascript/is-angularjs-ready-for-the-enterprise.html. [Använd 10 4 2016].

[29] ”BitNative,” [Online]. Available: http://www.bitnative.com/wp-content/uploads/2014/04/Angular-Trend1.png. [Använd 10 4 2016].

[30] ”Ionic Framework,” Ionic, [Online]. Available: http://ionicframework.com/. [Använd 10 4 2016].

[31] ”Ionic Framework: Documentation,” [Online]. Available: http://ionicframework.com/docs/api/. [Använd 10 4 2016].

[32] ”Appcelerator: Native apps. Mobile APIs. Real-time analytics. One Platform.,” Appcelerator, [Online]. Available: http://www.appcelerator.com/mobile-app-development-products/. [Använd 10 4 2016].

[33] D. Markov, ”Tutorialzine,” 13 10 2015. [Online]. Available: http://tutorialzine.com/2015/10/comparing-the-top-frameworks-for-building-hybrid-mobile-apps/. [Använd 10 4 2016].