Sem vložte zadání Vaší práce.

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra softwarového inženýrství

Bakalářská práce

Vývoj mobilních aplikací pomocí univerzálních platforem

Štěpán Heller

Vedoucí práce: Ing. Josef Gattermayer

6. května 2013

Poděkování

Mé poděkování patří zejména Bc. Dominikovi Veselému za cenné rady, které mi jako zkušený vývojář mobilních aplikací poskytl. Dále bych rád poděkoval všem, kdo mě v průběhu tvorby práce podporovali.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů, zejména skutečnost, že České vysoké učení technické v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona.

V Praze dne 6. května 2013 .....

Odkaz na tuto práci

Heller, Štěpán. *Vývoj mobilních aplikací pomocí univerzálních platforem.* Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2013.

Abstract

This thesis explores the options of mobile application cross-platform development with a special focus on those frameworks which enable to develop these applications using web technologies. The thesis provides a comparison of individual frameworks, a technological description of this approach and also a detailed analysis of a leader in this field – the PhoneGap framework. The ultimate goal of this thesis is to provide a comparison between developing applications in the native way and developing them via frameworks like PhoneGap.

Keywords mobile application, hybrid application, mobile app, hybrid app, cross-platform development, PhoneGap, hybrid framework, web technologies, iOS, Andorid, Windows Phone

Abstrakt

Práce zkoumá možnosti multiplatformního vývoje mobilních aplikací se zaměřením na frameworky, které umožňují vytvářet multiplatformní aplikace pomocí webových technologií. Práce poskytuje srovnání jednotlivých frameworků, technologický popis tohoto přístupu a také detailní analýzu frameworku PhoneGap, který je pionýrem v této kategorii. Cílem práce je poskytnout srovnání mezi vývojem nativní cestou a vývojem pomocí frameworků jako je právě PhoneGap.

Klíčová slova mobilní aplikace, hybridní aplikace, multiplatformní vývoj, PhoneGap, hybridní framework, webové technologie, iOS, Android, Windows Phone

Obsah

Úvod **1**
Motivace 2

1 Multiplatformní vývoj **3**
1.1 Co je to multiplatformní vývoj? 3
1.2 Argumenty pro multiplatformní přístup 5
1.3 Slabiny multiplatformního přístupu 7
1.4 Přístupy k multiplatformnímu vývoji mobilních aplikací . . . 10

2 Hybridní frameworky **15**
2.1 Obecně 15
2.2 Technický popis 17
2.3 Příklady hybridních frameworků 22

3 PhoneGap **27**
3.1 Obecně 27
3.2 Historie 29
3.3 Vývojářské nástroje 32
3.4 API 36
3.5 PhoneGap v reálných aplikacích 39
3.6 Budoucnost 40

4 Hybridní vs. nativní vývoj **41**
4.1 Způsob vývoje 41
4.2 Vlastní aplikace 46
4.3 Monetizace 48
4.4 Shrnutí 49

5 Demo aplikace **51**
5.1 Twitter klient (BlueBirdGap) 51
5.2 Foto aplikace (Orionids) 52
5.3 Časovací aplikace (TimerGap) 54

Závěr **57**

Literatura **61**

A Seznam použitých zkratek **67**

B Obsah přiloženého CD **69**

Seznam obrázků

1.1 Podpora HTML5 standardů v mobilních prohlížečích [40] 9
1.2 Architektura runtime frameworku Titanium [2] 13

2.1 Struktura hybridní aplikace [45] 18
2.2 Komunikace hybridní aplikace s operačním systémem 20
2.3 Architektura frameworku MoSync Wormhole [9] 24

3.1 Ripple Mobile Enviroment Emulator 34

4.1 Vývoj nabídek zaměstnání pro HTML5 vývojáře [10] 42
4.2 Vývoj nabídek zaměstnání pro PhoneGap vývojáře [10] 43
4.3 Vývoj nabídek zaměstnání pro iOS vývojáře [10] 43

5.1 BlueBirdGap, zobrazení timeline 52
5.2 Orionids, domovská obrazovka 53
5.3 TimerGap, proces odpočítávání 54

Seznam tabulek

4.1 Srovnání ceny hybridního a nativního přístupu při vývoji aplikace pro 3 platformy 44
4.2 Porovnání hybridního a nativního přístupu v klíčových oblastech 49

Úvod

V rámci této bakalářské práce se věnuji žhavé novince posledních let – hybridním mobilním aplikacím a jejich vývoji. V dnešní době stoupá poptávka po tzv. chytrých telefonech a symetricky s tím stoupá i zájem o mobilní aplikace. Vývoj pro mobilní platformy otevírá vývojářům zcela nový svět, ve kterém používají nové technologie a nové postupy.

Kromě rozvoje mobilních aplikací můžeme v posledních letech sledovat i nástup technologie HTML5, která dobývá nejen naše internetové prohlížeče pomocí moderních webových stránek, ale stále častěji jsou pomocí ní vyvíjeny i klasické desktopové či mobilní aplikace. Právě toho si všimli tvůrci univerzálních platforem a snaží se tuto technologii obohatit o chybějící funkce, sloužící zejména pro přístup k nativním elementům mobilních zařízení, jako je například fotoaparát, mikrofon nebo nativní souborový systém.

Cílem mé bakalářské práce je přinést co možná nejobjektivnější pohled na vývoj mobilních aplikací pomocí univerzálních frameworků a pomoci tím začínajícím vývojářům v rozhodování, zda-li je při vývoji té jejich aplikace hybridní přístup vhodný.

Práce nejdříve hovoří obecně o multiplatformním přístupu a posléze detailně analyzuje možnosti, které tento přístup přináší do mobilního světa. Jelikož je tato práce zaměřena na tzv. hybridní frameworky, podrobně jsem zmapoval celý tento tržní segment a představil jednotlivé zástupce a přístup, který přináší. Detailní analýze je podroben nejznámější zástupce tzv. hybridních frameworků – framework PhoneGap, u nějž jsem zmapoval historii a také rozebral jednotlivé funkce, které vývojářům poskytuje.

V kapitole 4 provádím srovnání nativního a hybridního přístupu z několika důležitých hledisek, které jsou pro rozhodnutí o technologii vývoje důležité. Nedělám si nároky na obecnou pravdu a tak je třeba k výsledkům

přistupovat s rezervou, neboť pro každý typ aplikace jsou relevantní jiná kritéria.

Aby tato práce nebyla založena jen na teoretických základech, vyzkoušel jsem si vývoj ve PhoneGap při vývoji tří elementárních aplikací, jež jsou součástí této práce.

Motivace

Toto téma jsem zvolil zejména proto, že mě vývoj mobilních aplikací vždy lákal, ale bál jsem se poměrně náročných vstupních požadavků na mé znalosti. Multiplatformní přístup mne zaujal z toho důvodu, že jsem byl zvědavý, jestli se tvůrci těchto nástrojů vyvarovali problémů, kterými trpí například Java.

Stejně tak mne zaujalo, že díky moderním multiplatformním nástrojům jako je PhoneGap, bych mohl při vývoji mobilních aplikací zúročit znalosti, které jsem načerpal při tvorbě klasických webových aplikací.

Kapitola 1

Multiplatformní vývoj

1.1 Co je to multiplatformní vývoj?

Multiplatformní vývoj je nadále žhavým tématem v diskuzích uvnitř komunit vývojářů desktopových i mobilních aplikací. Jedná se o přístup, který umožňuje vývojářům napsat program jednou, ale být jej schopen spouštět na více platformách. Pro splnění definice však již není podstatné, na kolika platformách může program běžet, pakliže je jich více než jedna. Tedy i aplikace, která dokáže běžet pouze na dvou platformách, je z hlediska této definice multiplatformní.

Na tomto místě by šlo namítnout, že i aplikace, která je napsána pro každou platformu zvlášť, je multiplatformní, neboť jí lze spouštět na více systémech. Je to do jisté míry pravda, nicméně pokud se na to podíváme z jiného úhlu, můžeme takový přístup označit za vývoj aplikací, které se podobají, ale nejedná se o programy totožné.

Multiplatformní přístup je tedy způsob, jak využít jeden zdrojový kód, který jsme schopni provozovat (s minimálními úpravami) na různých platformách a operačních systémech. Pro tento přístup se v angličtině vžilo heslo „write once, run everywhere“.

Jedním z často používaných dělících kritérií, pomocí kterých můžeme multiplatformní software rozlišovat, je kritérium způsobu jejich kompilace a následného běhu na samotné platformě.

1. Zdrojový kód je nutno zkompilovat pro každou platformu zvlášť. Poté je tedy nutno udržovat repozitář s různými instalačními balíčky pro jednotlivé platformy.

2. Zkompilování zdrojového kódu pouze jedinkrát a jeho následný běh

na uživatelově platformě pomocí nějakého interpreteru či run-time prostředí, které je součástí systému.

Příkladem multiplatformních aplikacích je vývojářské prostředí Eclipse, internetový prohlížeč Opera, Google Earth apod.

Jak již bylo řečeno, způsobů jak multiplatformně vyvíjet je mnoho. Pro tvorbu multiplatformní desktopové aplikace je možno využít standardního programovacího jazyka (pro který existuje kompilátor na více platformách) spolu s frameworkem, který umožňuje multiplatformní vývoj (nejen) GUI. Mezi takové UI toolkity můžeme zařadit například Qt, wxWidgets nebo GTK+.

Kapitola sama pro sebe je Java. Java je platformou umožňující vývoj aplikací zcela nezávislých na operačním systému. Tato technologie je skutečným zosobněním přístupu „write once, run everywhere“. Nutno však v zápětí dodat, že „everywhere“ je omezeno přítomností Java Virtual Machine. Programy napsané v Javě totiž nejsou závislé na konkrétním operačním systému, ale právě na speciálním prostředí (JVM), které zdrojový kód (Java bytekód) interpretuje.

Ačkoliv je Java velmi oblíbenou vývojářskou platformou, nestala se nikdy všudypřítomnou technologií, která překoná rozdíly mezi platformami, jak se mnozí domnívali. O důvodech můžeme spekulovat. Jedním z nich je zcela nepochybně nenativní chování javovských aplikací. Programy v Javě totiž na dané platformě málokdy působí jako nativní aplikace, což představuje značnou překážku k jejich širokému přijetí. I přes snahy o řešení (například pomocí knihovny Java Swing) se tento problém stále nepodařilo zcela překonat.

Nezřídkakdy se můžete setkat s názorem, že „HTML5 je nová Java“ [43]. Toto tvrzení vychází zejména z faktu, že HTML5, stejně jako Java, splňuje paradigma „write once, run everywhere“. Na rozdíl od Javy je však vývoj v HTML5 podstatně jednodušší a výkon aplikací se neustále zlepšuje (zejména díky výkonnějším JavaScriptovým enginům v internetových prohlížečích). Další předností HTML5 aplikací je jejich snadná dostupnost pro koncového uživatele. Nemusí instalovat víc než webový prohlížeč a znát tu správnou URL.

Zásadním argumentem, proč je HTML5 novou Javou, je však podle mě něco jiného. Je to zejména dramatický nástup mobilních zařízení. „Write once, run everywhere“ pro javovské aplikace v mobilním světě neplatí. Z předních hráčů na trhu se Java Virtual Machine vyskytuje pouze na Androidu, na ostatních platformách si tudíž javovské aplikace nespustíte. Naopak HTML5 je díky všudypřítomnosti webového prohlížeče skutečně multiplatformní technologií i z pohledu mobilního světa.

Samozřejmě by někdo mohl namítnout, že porovnávání Javy a HTML5 poněkud kulhá, neboť funkcionalita, kterou dává vývojářům k dispozici Java, je oproti HTML5 obrovská. Avšak HTML5 obsahuje (a nadále přidává) zejména ty funkce, které najdou uplatnění i v chytrých mobilních telefonech. A z těch již zase tak mnoho nechybí. HTML5 přebírá pozici Javy právě proto, že je úspěšné i na poli mobilních telefonů.

1.2 Argumenty pro multiplatformní přístup

V této části bych se rád podrobněji zaměřil na důvody, které vedou vývojáře k zahrnutí multiplatformního řešení do svých úvah. Pokusím se zde nastínit argumenty, které se v této souvislosti nejčastěji skloňují.

1.2.1 Možnost cílit na široké spektrum platforem najednou

Tento důvod byl podle výzkumu společnosti VisionMobile uváděn vývojáři nejčastěji [40]. 60 % z nich konstatovalo, že právě širší záběr, který jim multiplatformní přístup přináší, je přesvědčil k využití některého z multiplaformních nástrojů.

Velké společnosti stojící za platformami jako je Android či iOS se velmi snaží, aby vývojáři vyvíjeli exkluzivně pro jejich platformu. Přítomnost těch nejžádanějších aplikací výhradně pro jednu platformu jí totiž přináší značnou konkureční výhodu na trhu. V zájmu těchto hráčů proto není, aby bylo možné aplikace snadno vyvíjet na více platforem najednou. Naopak méně rozšířené systémy jako je Windows Phone, Samsung Bada, RIM a další mají eminentní zájem na tom, aby bylo pro jejich systém dostupné co největší množství aplikací bez zbytečného čekání. Takzvaný „time-to-market“ je totiž u těchto platforem často mnohem větší než u velkých hráčů,[c] protože nativní vývojáři se logicky snaží prvně zasáhnout co největší část trhu, která je okupována právě iOS a Androidem.

Možnost rychle zasáhnout tzv. „long tail“ je tedy pro vývojáře velice lákavá. Mohou bez zbytečných prodlev rychle oslovit prakticky celý trh a nemusí tak dávkovat marketingovou podporu podle toho, na které platformě jejich aplikace právě běží. O jejich aplikaci mohou mluvit všichni.

1.2.2 Nižší náročnost na zdroje

Náročnost na zdroje je klíčový faktor zejména pro společnosti, které aplikace vyvíjejí. Pokud chtějí být s aplikací opravdu úspěšní, musí s ní oslovit

co nejširší spektrum uživatelů. To v dnešní době znamená mít verzi pro iOS, Android a Windows Phone. Dělat nativní aplikaci pro každý z těchto systémů znamená platit tři týmy vývojářů, protože pro každou platformu se programuje jiným programovacím jazykem, systém má jiná API volání a podobně. Zároveň je velmi neobvyklé, aby jeden vývojář ovládal na solidní úrovni technologie pro všechny tři platformy.

Ale problém není jen v kódování. Vyvíjet pro tři různé platformy znamená udržovat tři různé sady zdrojových kódů, trackování bugů pro každou z nich zvlášť a podobně. Zároveň účinná synchronizace mezi týmy klade další požadavky na lidské zdroje. Zajistit, aby všechny tři verze měly stejné vlastnosti, dodržovaly stejná designová paradigmata a zároveň se neodchylovaly od časového harmonogramu, je poměrně obtížný úkol.

„Zjistili jsme, že použitím multiplatformního nástroje jsme schopni v průměru zredukovat náš ‚time to market' o 70%," tvrdí Paulius Uza, výkonný ředitel společnosti InRuntime [40]. [1]

1.2.3 Nižší finanční náročnost

S předchozím bodem přímo souvisí i nižší finanční náročnost. Je zřejmé, že když budete zaměstnávat méně vývojářů, kteří pro vás aplikaci vyvíjejí, i vaše náklady budou nižší. Není to však jen o samotném vývoji. Po vypuštění hotové aplikace na trh je nutno zajistit podporu pro její uživatele. Z hlediska rozpočtu je značný rozdíl, zda-li musíte obhospodařovat jednu sadu se zdrojovými kódy nebo tři.

Podle společnosti VisionMobile stojí vývoj pro každou další platformu v průměru 50 % původních nákladů [40]. Nejedná se tedy rozhodně o malé částky a každou rozpočtově odpovědnou firmu tento fakt nutí k zamyšlení, jak vyvíjet pro více platforem levněji.

1.2.4 Menší vstupní bariéry

K tomuto argumentu si nejprve uveďme dvě poznámky. Zaprvé, jazyky, ve kterých se vyvíjí nativní aplikace jsou poměrně obtížné na naučení. Názorným příkladem je v tomto ohledu Objective-C, který se vyznačuje nejen poměrně neobvyklou syntaxí, ale zároveň obsahuje množství značně neobvyklých návrhových vzorů a doporučení, která jsou poměrně obtížná na pochopení. I to je důvodem, proč je na trhu v současnosti převis nabídky nad poptávkou po vývojářích mobilních aplikací.

[1] "We have found that by using cross-platform tools our time to market is reduced by 70% on average," remarks Paulius Uza, CEO of development house InRuntime [40].

Neméně důležitým faktorem je poměrně široká masa webových vývojářů, která se na trhu vyskytuje. Webové technologie jsou v současnosti velmi oblíbené a to především díky strmému nárůstu popularity celého Internetu v minulé dekádě. Vývoj pomocí webových technologií je poměrně jednoduchý, neboť se při něm vývojář pohybuje na vysoké úrovni abstrakce. Díky tomu je možno vyvíjet webové aplikace rychle a levně.

Tohoto faktu šikovně využívají tvůrci multiplatformních řešení. Snaží se zasáhnout právě široké obecenstvo v řadách webových vývojářů tím, že maximálně snižují vstupní bariéry, které by jim mohly bránit ve vývoji mobilních aplikací. Drtivá většina multiplatformních nástrojů proto využívá pro vývoj mobilních aplikací právě webové technologie HTML[e], CSS a JavaScript. Pro přístup k nativním funkcím telefonu nabízí tyto nástroje jednoduchá API[g]. Že se jim tento přístup vyplací, dokládá i výzkum společnosti VisionMobile, který zjistil, že více jak 60 % vývojářů, kteří využívají některý z multiplatformních nástrojů, má více jak 5 let zkušeností s webovým vývojem. Stejně tak při pátrání po důvodech, které vedly vývojáře k využití některého z multiplatformních nástrojů, byla jako druhá nejčastější odpověď uváděna právě možnost využít již naučené technologie [40].

1.3 Slabiny multiplatformního přístupu

Pokud zmiňujeme přednosti multiplatformního přístupu, neměli bychom zapomínat ani na slabiny, které stále brání jeho širšímu rozšíření.

1.3.1 Slabší výkon

Slabší výkon je nejčastěji uváděným důvodem pro zavržení multiplatformního nástroje [40]. Vzhledem k tomu, že drtivá většina multiplatformních nástrojů závisí na nějaké mezivrstvě (ať už je to jakýsi most u hybridních aplikací, nebo runtime), není slabší výkon takto vytvořených aplikací příliš překvapivý.

1.3.2 Omezený přístup k nativním funkcím zařízení

Pravdou je, že ze své podstaty budou multiplatformní nástroje vždy o několik kroků pozadu oproti nástrojům pro nativní vývoj. Souvisí to právě s tím, že se multiplatformní nástroje snaží podporovat co nejvíce různých platforem. Z toho důvodu se musí omezovat na průnik množin funkcí, které tyto platformy nabízejí. Velice dobře tento problém popsal Steve Jobs ve svém otevřeném dopise společnosti Adobe „Thoughts on Flash“.

„Z naší strastiplné zkušenosti víme, že položením vrstvy třetí strany mezi vývojáře a platformu, nevyhnutelně získáme aplikace podprůměrné kvality a vytvoříme tím překážky pro další rozvoj platformy. Pokud se vývojáři naučí být závislí na knihovnách a nástrojích třetí strany, mohou využívat nových vylepšení platformy až ve chvíli, kdy se třetí strana rozhodne je implementovat. Nemůžeme být vydáni na milost třetí straně, až se rozhodne umožnit našim vývojářům používat vylepšení, které v platformě uděláme.

Problém se jenom zhorší, pokud třetí strana v rámci svého nástroje podporuje více platforem. Třetí strana pak nemusí implementovat nová vylepšení z jedné platformy až do chvíle, dokud jej neimplementují všechny platformy jež podporuje. Vývojáři aplikací pak mají přístup pouze k nejnižšímu společnému jmenovateli všech funkcí.“ Steve Jobs [41]
[2]

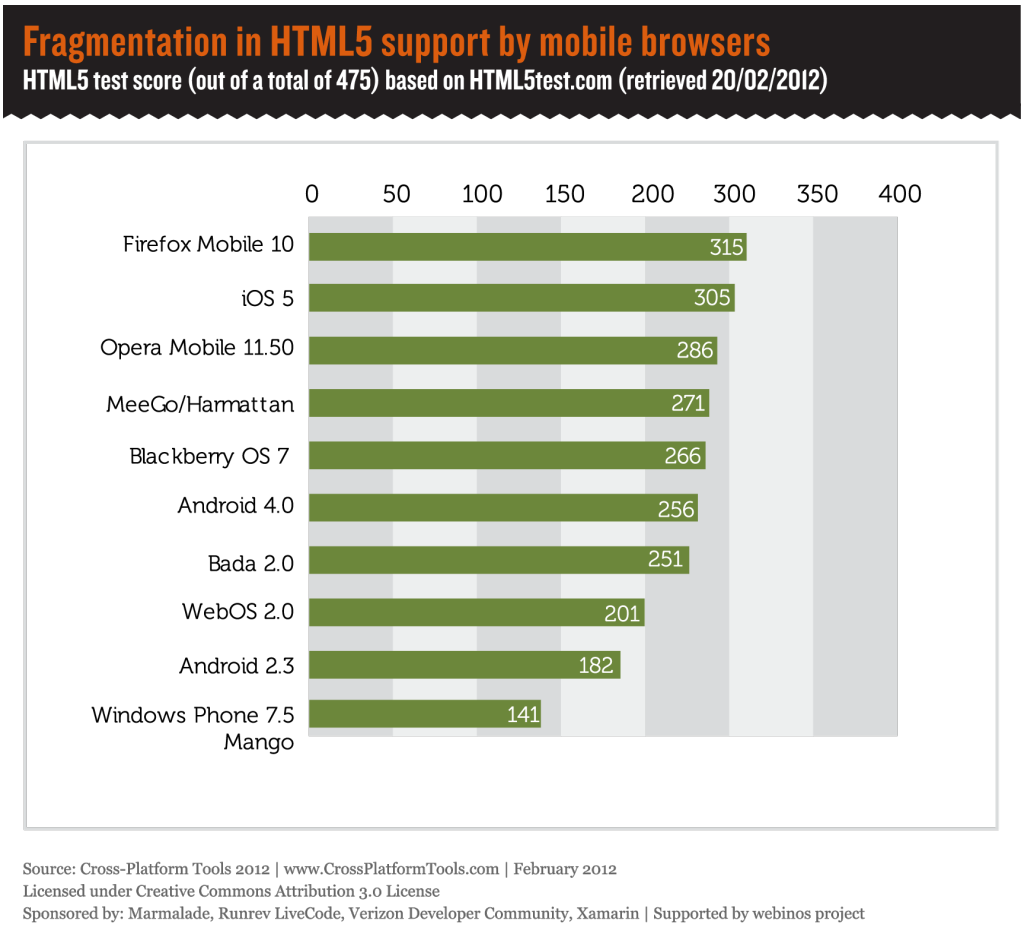
Nejedná se však pouze o přístup k nativním funkcím, jako je lokální úložiště nebo hardwarové funkce. Jde i o přístup k nativním UI elementům, který u multiplatformních nástrojů často chybí nebo je značně omezený. Vývojáři jsou tak nuceni napodobovat vzhled a chování nativních aplikací pomocí CSS a JavaScriptu, což bývá často složité a má to negativní dopad i na výkon samotných aplikací. Právě omezení v používání nativních UI elementů respondenti uváděli jako druhý nejčastější důvod k zavržení multiplatformního nástroje [40].

1.3.3 Fragmentace

Fragmentování neboli tříštění celku je v kontextu mutliplatformního vývoje především problémem webových prohlížečů. Podpora HTML5 standardů se totiž mezi jejich verzemi značně liší. Tento fakt přináší vývojářům časté komplikace, protože mnoho multiplatformních nástrojů používá pro běh aplikací právě prostředí internetového prohlížeče. Na obrázku 1.1 vidíme,

[2] "We know from painful experience that letting a third party layer of software come between the platform and the developer ultimately results in sub-standard apps and hinders the enhancement and progress of the platform. If developers grow dependent on third party development libraries and tools, they can only take advantage of platform enhancements if and when the third party chooses to adopt the new features. We cannot be at the mercy of a third party deciding if and when they will make our enhancements available to our developers.

This becomes even worse if the third party is supplying a cross platform development tool. The third party may not adopt enhancements from one platform unless they are available on all of their supported platforms. Hence developers only have access to the lowest common denominator set of features."Steve Jobs [41]



Obrázek 1.1: Podpora HTML5 standardů v mobilních prohlížečích [40]

jak moc je podpora standardů mezi prohlížeči roztříštěná. Je z něj patrné, že webový prohlížeč dodávaný se systémem Windows Phone je na tom s podporou HTML5 více než 2x hůře než prohlížeč v systému iOS 5. Problémy s optimalizací webových stránek se tedy přenesly i do světa mobilních aplikací, což vývojáře jistě nepotěší.

1.3.4 Náročnost vývoje

I když jsme v benefitech multiplatformního přístupu zmínili, že díky použití webových technologií je samotný vývoj jednodušší, průzkum společnosti VisionMobile [40] prokázal, že právě strmě stoupající náročnost vývoje je častým důvodem k zavržení multiplatformního nástroje. Zdá se tedy, že s

multiplatformními nástroji snadno vyvinete jednodušší aplikaci, ale komplexnější úkoly již představují větší výzvu.

1.4 Přístupy k multiplatformnímu vývoji mobilních aplikací

Dosud jsme hovořili o mutliplatformním vývoji, jako by se jednalo o monolitický celek. V rámci něj však existuje celá řada přístupů, které se od sebe liší technicky i filozoficky. V této části se tedy zaměříme právě na stručný popis těch nejpoužívanějších přístupů, které multiplatformní vývoj nabízí.

1.4.1 Javascriptové frameworky

Do této rodiny řadíme například známý framework jQuery Mobile či Sencha Touch. Jedná se o frameworky, které umožňují webovým vývojářům snadno vyvíjet dotyková rozhraní svých webových aplikací. Například jQuery Mobile poskytuje možnosti jak definovat vzhled stránky a jednotlivé elementy způsobem, který umožní jejich bezproblémové zobrazení na jakémkoliv dotykovém zařízení. Nesnaží se však simulovat nativní vzhled ovládacích prvků na jednotlivých platformách, narozdíl třeba od nástroje Kendo UI.

Zástupci

1. jQuery Mobile
2. Sencha Touch
3. Kendo UI
4. DHTMX Touch
5. iUI
6. Cocos2D (hry)

1.4.2 Aplikační továrny

Aplikační továrny jsou velice moderní přístupem, jak tvořit mobilní aplikace. Slovo „tvořit“ je skutečně na místě, neboť při využití těchto nástrojů není třeba žádných znalostí programovacích jazyků (tedy ani HTML, CSS a JavaScriptu). Aplikace jsou tvořeny uživatelem pomocí skládání vizuálních

elementů v nějakém (většinou webovém) WYSIWYG editoru. V případě potřeby může uživatel doplnit chybějící logiku pomocí JavaScriptu.

Jedním z nejznámějších zástupců této skupiny je Tiggzi. Tiggzi je poměrně komplexní nástroj, který v sobě kombinuje javascriptový framework jQuery Mobile, webové technologie HTML, CSS a JavaScript a hybridní framework PhoneGap, který umožňuje exportovat aplikace vytvořené v Tiggzi do binárních souborů, které je následně možno publikovat na oficiálních tržištích v rámci jednotlivých platforem.

Tiggzi nabízí uživatelům webový nástroj App builder, kde je možno si velmi jednoduše („drag & drop“[k]) poskládat uživatelské rozhraní aplikace z předdefinovaných UI komponent. Pokud by vám předdefinovné funkce nestačily, můžete si chybějící funkcionalitu dopsat pomocí JavaScriptu. Díky využití hybridního frameworku PhoneGap může uživatel ve své aplikaci přistupovat k nativním funkcím telefonu. Pro tvůrce aplikací jsou připraveny také nástroje pro snadné připojování k REST API služeb třetích stran. Vaše aplikace tak může snadno získávat data například z Twitteru. Tiggzi dále nabízí nástroje pro testování aplikací, databázové úložiště a další. Je však třeba zmínit, že za použití Tiggzi je třeba zaplatit.

Další zástupci

1. AppMkr
2. Wix mobile
3. Spot Specific
4. Games Salad

1.4.3 Hybridní frameworky

Tento druh multiplatformních nástrojů si popíšeme jen velmi stručně, neboť je mu věnována celá následující kapitola XY. Jedná se o řešení, které umožňuje vývojářům vytvářet aplikace čistě pomocí webových jazyků. Hlavní předností těchto frameworků jsou knihovny napsané v nativních jazycích dané platformy, které vývojářům poskytují tzv. most, pomocí kterého mohou přistupovat k nativním funkcím telefonu jako jsou notifikace, fotoaparát, akcelerometr a podobně. Tyto funkce jsou jim přístupné pomocí JavaScriptového API. Nejznámějším příkladem tohoto přístupu je hybridní framework PhoneGap, kterému se budeme podrobně věnovat v kapitole XY.

Další zástupci

1. Trigger.io
2. Sencha Touch (v2)
3. MoSync

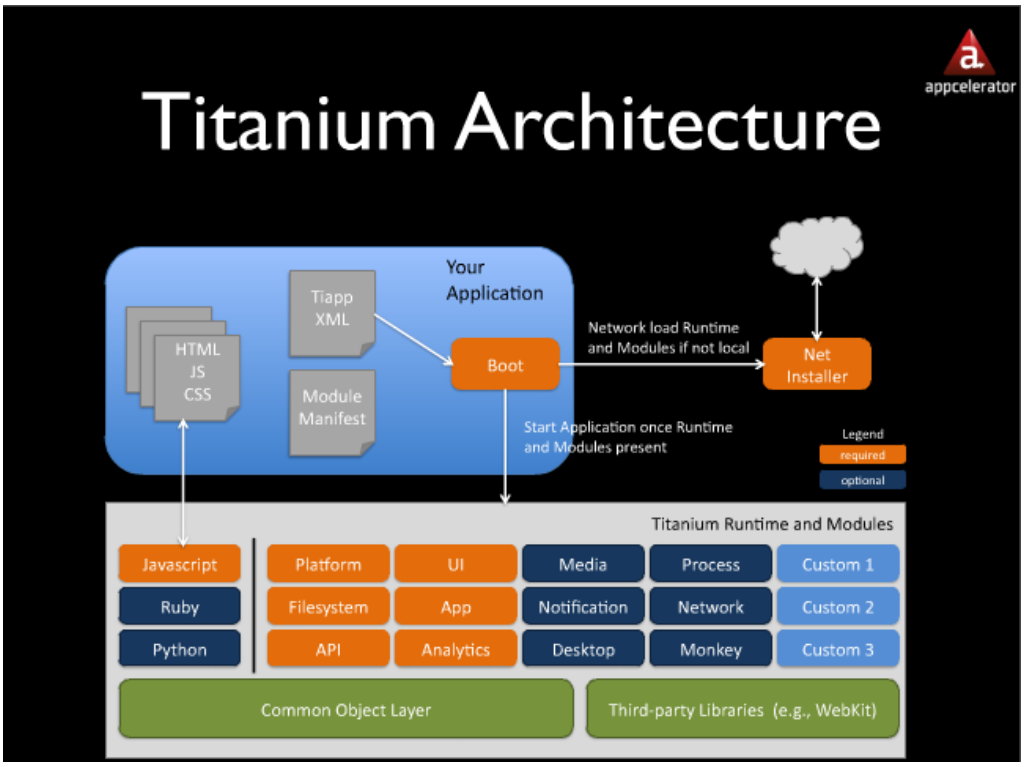
1.4.4 Runtime

Tvůrci runtime frameworků se snaží k věci přistupovat trochu jinak. Pokud využíváte takový nástroj, tak vlastně píšete nativní aplikaci, avšak místo nativního jazyka používáte JavaScript. Aplikace se spouští jako nativní, nepotřebuje k tomu prostředníka v podobě webového prohlížeče. Jednou ze zásadních výhod tohoto přístupu je, že při vývoji používáte nativní UI elementy. Není nutné tedy nic napodobovat jako v případě hybridních frameworků a výkon takto napsaných aplikací tak bývá v konečném důsledku rychlejší.

Pro správné pochopení tohoto principu si můžeme popsat, jak funguje nejznámější zástupce tohoto přístupu – nástroj Appcelerator Titanium. Uživatel v něm píše aplikaci čistě pomocí JavaScriptu, kterýžto zdrojový kód je následně zabalen do nativních binárních souborů určených pro každou z platforem. Při spuštění je použit JavaScriptový interpreter (v iOS JavaScriptCore, v Androidu a Blackberry Mozilla Rhino), který interpretuje váš kód za běhu. Pro volání nativních komponent a hardwarových funkcí telefonu je použito Titanium API, které je napsáno pomocí nativních jazyků a uživatel k němu přistupuje přes volání JavaScriptových funkcí. Appcelerator tvrdí, že 80 % kódu je možno použít při vývoji mutace aplikace pro jinou platformu [4]. Hlavním rozdílem oproti hybridním frameworkům je tedy běh aplikací bez nutnosti spouštět webový prohlížeč. O pravé nativní aplikace se však ani v tomto případě nejedná, ač se tím někteří (např. Appcelerator) rádi chlubí. V tomto případě jde spíše o marketingový termín.

Pojďme se seznámit s nástrojem Appcelerator trochu blíže. Krom open-source SDK, které poskytuje více než 5000 API volání pro přístup k nativním prvkům mobilních platforem, je součástí celého řešení i IDE na bázi nástroje Eclipse a MVC framework pro snazší vývoj. Dále je možno využít cloudových služeb pro údržbu a testování aplikací. Pomocí nich lze například posílat push-notifikace, integrovat sociální sítě, posílat e-maily přímo z aplikace apod [11]. Příjemná je i možnost statistického zobrazení dat o používání aplikace uživateli – obdoba Google Analytics.

Appcelerator se zaměřuje především na webové vývojáře, které se snaží přesvědčit zejména díky lepšímu výkonu oproti aplikacím napsaným pomocí



Obrázek 1.2: Architektura runtime frameworku Titanium [2]

hybridních frameworků jako je PhoneGap [1]. Z výzkumu [40] vyplývá, že uživatelé si vybírají Appcelerator zejména kvůli nízké ceně, přístupu k hardwarovým funkcím telefonu a širokým možnostem úpravy UI jejich aplikace. Naopak velká část z nich by ocenila ještě lepší optimalizaci orientovanou na jednotlivé platformy.

Další zástupci

1. Adobe Flex/Air
2. Corona
3. AppMobi
4. Unity (Hry)

1.4.5 Překladače zdrojových kódů

Další možností je nechat si přeložit zdrojový kód své aplikace do nativního jazyka dané platformy, případně do nějakého mezikódu či strojového jazyka. Nezřídkakdy jsou překladače zdrojových kódu používány současně s nějakým runtime nástrojem. Tyto nástroje cílí především na zkušené vývojáře, kteří potřebují vyvíjet komplexní aplikace pro více platforem s vysokou úrovní aplikační logiky.

Známým nástrojem z tohoto okruhu je například Marmalade. Marmalade umožňuje vývojářům psát aplikace v jazycích C či C++, který je následně přeložen do nativního jazyka některé z podporovaných platforem. Těch je celá řada, od iOS, Androidu přes Samsung Bada a Blackberry až po dekstopové Windows [12]. Chybí však podpora pro Windows Phone. Pro přístup k nativním funkcím telefonu poskytuje Marmalade SDK API. SDK obsahuje ale i další užitečné nástroje jako je například debugger, UI builder a další.

Podle výzkumu [40] si vývojáři vybírají Marmalade především proto, že je velmi vhodný na vývoj her. Oceňují na něm také vysoký výkon, který jde pravděpodobně na vrub zejména nízké úrovni abstrakce. Na druhou stranu je u uživatelů poptávka po více možnostech licencování. Používání Marmalade je totiž placeno.

Další zástupci

1. MoSync
2. Eqela
3. XMLVM
4. Bedrock

Kapitola 2

Hybridní frameworky

2.1 Obecně

Jak jsem již naznačil v předchozí kapitole, hybridní frameworky jsou nástroje, které umožňují vývojářům mobilních aplikací schovat svojí „webovou aplikaci“ do nativního obalu a na venkovní svět tak působit jako klasická nativní aplikace (více o tom, jak wrappery fungují si řekneme v Technickém popisu v podkapitole 2.2).

Hlavní přednosti tohoto přístupu by se daly shrnout do několika bodů:

1. Přístup k nativním funkcím zařízení (fotoaparát, akcelerometr, výběr z kontaktů, geolokace apod.)
2. Distribuce přes oficiální tržiště jednotlivých platforem
3. Nezávislost aplikací na internetovém připojení
4. Vývoj čistě pomocí webových technologií
5. Podpora mnoha platforem (znovupoužitelnost velké části zdrojového kódu)

"They are too cost effective and smart to ignore."Dominique Hazael-Massieux, lead of the Mobile Web Initiative at The W3C. [3]

Hybridní frameworky jsou tedy využívány pro vývoj mobilních aplikací, zejména pokud má vývojář v úmyslu zasáhnout více platforem ve velmi krátkém čase. Za tímto účelem mu zapouzdřovače poskytují adekvátní nástroje.

Je velice pravděpodobné, že do doby než formát HTML5 skutečně dozraje, budou hybridní postupy stále častější volbou webových vývojářů. Společnost Gartner předpovídá, že v roce 2016 bude víc než 50 % všech mobilních aplikací hybridních. [6]

„Společnosti stále častěji zjišťují, že musí podporovat více platforem, zejména když ‚BYOD' přístup nabývá na síle."[6] [3]

Taková předpověď se samozřejmě neobjeví jen tak. Stojí za ní konkrétní problémy konkrétních firem, které musí řešit při budování strategie své firmy v digitální oblasti. Zákazníci od nich očekávají služby a aplikace určené přímo pro ně, pro jejich konkrétní platformu. Nesmělé pokusy velkého množství firem, které vyprodukují verzi 1.0 své aplikace pro jednu konkrétní platformu jsou typickým příkladem toho, kde se pro hybridní frameworky trh otvírá do široka.

I z toho důvodu se pozornost vývojářů otáčí k hybridním aplikacím. Gigant na poli enterprise řešení firma SAP v květnu loňského roku oznámila, že ve spolupráci s firmami Appcelerator (Appcelerator Titanium), Sencha (Sencha Touch) a Adobe (PhoneGap), bude svým zákazníkům v rámci svých širokých služeb nabízet i hybridní řešení. Součástí dohody je také poskytnutí možností k snadnému napojení zákaznických aplikací na data a služby, které SAP obvykle poskytuje [3].

Právě problematika firemních aplikací je pro hybridní frameworky živnou půdou. Stále více totiž ve firemním prostředí sílí tzv. BOYD[a] přístup. BOYD je akronymem pro Bring Your Own Device. Tento přístup znamená, že zaměstnanci při práci používají své vlastní přístroje (notebook, mobilní telefon) a nenutí tak svého zaměstnavatele, aby jim je na své náklady poskytoval. Některé nové požadavky však na zaměstnavatele tento přístup přeci jen klade. Jde právě o začlenění těchto zařízení do chodu firemní sítě. Jedná se jak o příjem e-mailů, sdílených kalendářů apod., tak o přístup k firemním informačním systémům, kterých mohou být ve větších firmách desítky. Právě k tomu je vhodné mít širokou paletu mobilních aplikací, které budou umožňovat bezproblémové začlenění nových zaměstnanců do chodu společnosti, ať už používají jakýkoliv systém na kterékoliv platformě. Vývoj nativních aplikací pro tyto účely by však byl nákladově naprosto neúnosný. Bylo by nutné vyvinout všechny potřebné aplikace třeba ve 4-6 verzích, pro každou platformu zvlášť. To by si vyžádalo zaměstnání desítek draze placených vývojářů. A vzhledem k tomu, že se v drtivé většině nejedná o nijak

[3] "Increasingly, enterprises are finding that they need to support multiple platforms, especially as the [bring your own device] BYOD trend gains momentum."[6]

graficky náročné aplikace, ale spíše o zobrazovače dat, použití hybridního přístupu se zde přímo nabízí.

Pojďme si shrnout základní plusy a mínusy, které jsou s hybridním přístupem obvykle spojovány. Později se k nim ještě vrátíme.

+ Rychlejší „time to market“.

+ HTML5 vývojáři jsou obvykle levnější a snáze k sehnání.

+ Čím více HTML/CSS/JavaScript kódu, tím více ho můžeme znovupoužít pro port na jinou platformu.

+ Náklady na údržbu jsou obvykle nižší.

+ Délka schvalovacího procesu pro aktualizace je snížena na minimum.

- Slabší výkon.

- Hybridní frameworky a HTML5 stále nepokrývají všechny možnosti, které mají nativní vývojáři.

- Apple se hybridním aplikacím brání a do budoucna je možno očekávat problémy při schvalovacím procesu.

Všechny výše uvedené argumenty je třeba zvážit ve chvíli, kdy se firma nebo vývojář rozhoduje, jakou technologii pro vývoj své aplikace použije. Pokud je aplikace spíše jednoduššího charakteru a neklade si vysoké nároky na grafický výkon, hybridní cesta bude pravděpodobně tou pravou. Naopak pro náročné hry a graficky sofistikované aplikace zůstává nativní přístup stále jedinou možností.

„Kvalita uživatelského rozhraní, kterého může vývojář vývojář dotáhnout v nativní aplikaci, jednoduše nestojí za extra vynaložené prostředky oproti hybridní cestě, jež umožňuje vytvořit uživatelské rozhraní dostatečné kvality,“ říká Ron Perry, technický ředitel firmy Worklight [38]. [4]

2.2 Technický popis

Hybridní aplikace je složena ze tří zásadních komponent:

[4] "The slickness of the user interface a developer can achieve in the native [app] model just isn't worth the extra spending compared to the very nice level of user-interface experience they get from the hybrid option,"says Ron Perry, CTO of Worklight. [38]

Obrázek 2.1: Struktura hybridní aplikace [45]

1. Samotný kód aplikace napsaný zpravidla webovými jazyky HTML, CSS a JavaScriptem

2. Webový runtime umožňující zobrazovat HTML a interpretovat JavaScript (UIWebView na iOS, WebView na Androidu)

3. Tzv. „nativní most“, který zprostředkovává komunikaci mezi webovým runtimem (respektive samotnou aplikací) a operačním systémem.

Strukturu hybridní aplikace naznačuje obrázek 2.1.

Pojďme si nyní detailněji popsat jednotlivé části. Samotná webová aplikace nemá zcela obvyklou strukturu, na kterou jsme zvyklí z tvorby klasických webových stránek. Velice často jde o jediný HTML soubor, kde je uvnitř elementu <body> ukryta celá architektura aplikace. Tento přístup umožňuje použití některého z oblíbených JavaScriptových frameworků, jako je např. jQuery Mobile, Zepto JS nebo Sencha Touch.

Samotné přechody mezi stránkami jsou realizovány buď pomocí JavaScriptu funkcemi hide() a show() či využitím přechodů definovaných ve standardu CSS3. Výše zmíněné javascriptové frameworky většinou poskytují i CSS šablonu, ve které jsou definovány styly pro různé ovládací prvky, jakými jsou například tlačítka přívětivá pro dotyk. Vývojář si pak může tyto šablony přizpůsobit k obrazu svému přidáním vlastních pravidel.

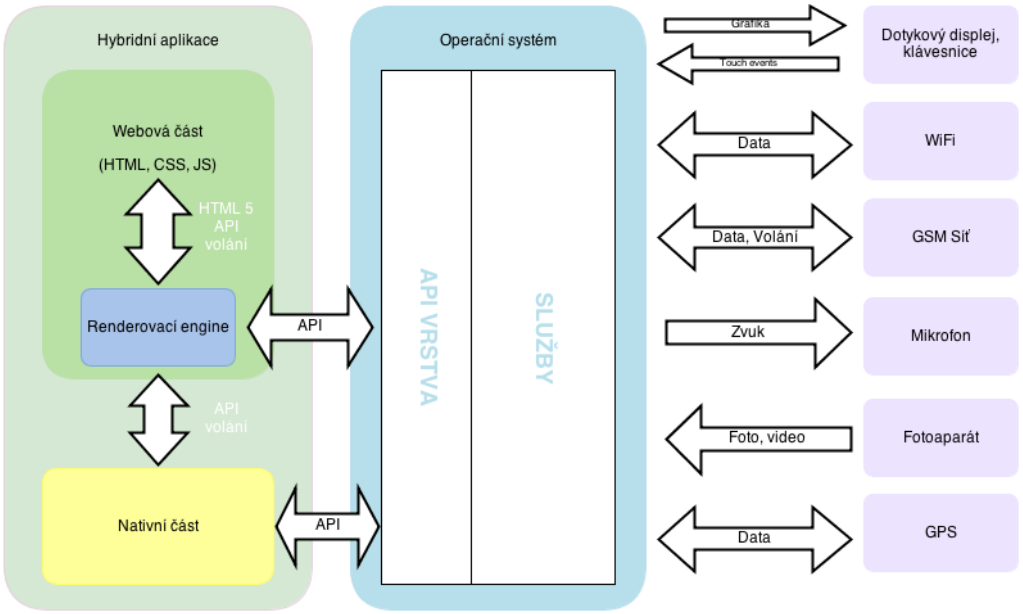
Srdcem každé hybridní aplikace je však jeho javascriptový kód, ve kterém je definována celá aplikační logika. Pro její definování je často využíván právě framework jQuery, který zjednodušuje a urychluje používání některých oblíbených javascriptových funkcí. V JavaScriptu jsou definovány všechny reakce na události, uživatelské vstupy a výpočty na pozadí.

Abychom mohli HTML zobrazit uživateli, potřebujeme webový runtime, který bude umět náš kód zobrazit a průběžně vyhodnocovat. Většinou se jedná o osekané jádro webového prohlížeče, který je jinak v systému standardně přítomen. Právě v těch omezeních je však často skrytý zádrhel pro vývojáře. Například platforma iOS používá jako runtime UIWebView, jenž má narozdíl od mobilního prohlížeče Safari značnou nevýhodu v absenci javascriptového enginu Nitro, který je díky Just In Time kompilace velmi rychlý. V praxi to znamená, že pokud vaše hybridní aplikace používá JavaScript, bude UI [c]působit na uživatele pomaleji než identické UI spuštěné v prohlížeči Safari na tom samém zařízení. [37] Některé statistiky uvádějí, že javascriptový engine Nitro, který Safari používá, je více než 3x rychlejší než engine používaný v UIWebView. [8] Bohužel pro vývojáře není jiná možnost, jak toto omezení obejít, než používat co nejméně JavaScriptu je možné a pokoušet se ho nahradit například CSS3. Což je pro hybridní aplikace životně závisející na JavaScriptu velký problém.

Webový runtime není v podstatě ničím jiným než obyčejným objektem, který v sobě ukrývá renderovací jádro (např. WebKit). Původním smyslem tohoto objektu bylo dát možnost vykreslovat HTML obsah uživatelům v rámci nativních aplikací. Například Twitter ve své oficiální aplikaci používá takový webový runtime pro otevírání odkazů z tweetů. Uživateli se tak zobrazil obsah odkazu přímo v rámci aplikace a nemusel přepínat mezi aplikací a prohlížečem.

Tvůrcům hybridních frameworků se však podařilo uzpůsobit tyto objekty tak, aby se daly použít jako wrapper pro webovou aplikaci. Tohoto uzpůsobení je dosaženo pomocí rozšíření originálních tříd (UIWebView respektive WebView) v nativním kódu. Objekty jsou obohaceny o množství funkcí, které přispívají k optimálnímu běhu hybridní aplikace pomocí specifických reakcí na rozličné události.

Nyní, když chápeme jak funguje webový runtime, si můžeme vysvětlit jak probíhá komunikace mezi webovou aplikací a nativními funkcemi



Obrázek 2.2: Komunikace hybridní aplikace s operačním systémem

telefonu. Tato komunikace může probíhat v zásadě dvěma způsoby:

1. Přímo z HTML pomocí HTML5 API.
2. Pomocí volání konkrétních javascriptových funkcí dle specifikace konkrétního frameworku.

Na prvním způsobu není nic zvláštního. V dnešní době jej mohou využívat i moderní webové stránky, pokud si je uživatel zobrazí v některém z novějších prohlížečů, jenž implementuje konkrétní funkci z HTML5 standardu. Všeobecně rozšířeným API voláním, které je možno volat přímo z HTML, je například geolokace nebo úložiště pro lokální ukládání dat. Tato volání vyhodnotí přímo webový runtime a sám se postará o zavolání konkrétní nativní funkce vhodné pro splnění požadavku.

Větší pozornost si zaslouží druhý způsob komunikace s nativními funkcemi. Při něm již využijeme tzv. nativního mostu, který nám poskytuje samotný hybridní framework. Jedná se zpravidla o sadu tříd napsanou v nativním jazyce dané platformy, které slouží k zprostředkovávání komunikace mezi javascriptovými funkcemi definovanými v samotné webové aplikaci a operačním systémem zařízení, na kterém hybridní aplikace běží. V rámci každého operačního systému je třeba tento most implementovat

odlišně. Každá platforma totiž umožňuje nějakým způsobem komunikovat mezi webovým runtimem a operačním systémem. Úkolem pro tvůrce hybridních frameworků je přijít s řešením, kterak zprostředkovat vývojáři sadu funkcí, které nezávisle na platformě poskytují aplikaci stejné služby a přitom nedat znát, že „za oponou“ probíhají zásadně odlišné procesy. Právě tato funkcionalita je klíčem k multiplatformnosti hybridních aplikací.

Logickým faktem je, že tzv. nativní most musí fungovat oboustranně, tj. nejen zpracovávat požadavky od webové aplikace, ale zároveň být schopen vrátit odpověď od operačního systému aplikaci zpět. Ač to zní jako poměrně triviální logika, každá z těchto cest je obvykle implementována velmi odlišně. Jak jsme již zmínil, implementace tohoto nativního mostu se může lišit framework od frameworku, já v rámci této sekce popíši, jak tento most funguje v nejpoužívanějším hybridním frameworku PhoneGap.

Ve PhoneGap je most mezi JavaScriptem a operačním systémem (v tomto případě Android) realizován pomocí Promptu. JavaScriptová API volání (například kamera, mikrofon apod.) jsou konvertována jako prompt příkaz, který je odposlechnut funkcí onJsPrompt definované webovým runtimem, která příkaz rozpozná a provede odpovídající nativní volání. [5]

Opačná cesta je realizována poněkud méně elegantním způsobem. Funguje na principu neustálého dotazování se nativní strany na odpověď. Aplikační strana se v intervalu 50 milisekund ptá nativní strany, zdali už je připravená odpověď na vznesený dotaz. Pokud je odpověď pozitivní, most zprostředkuje příchozí data klientské aplikaci.

V rámci operačního systému iOS vypadá schéma komunikace poněkud jednodušeji. V iOS funguje nativní most pro komunikaci směrem z klientské aplikace na principu iFrame, kdy jsou javascriptová volání nativních funkcí skladována ve frontě, ze které jsou následně čtena a prováděna pomocí nativní komponenty. Alternativou k tomuto přístupu je použití XHR dotazů. Klientská strana provede XHR dotaz na falešnou URL adresu v níž jsou umístěné příkazy, které se mají provést. Tyto příkazy jsou odposlechnuty, seřazeny a následně vykonány nativní stranou. [5]

I komunikace opačným směrem je realizována poněkud jednodušeji, než je tomu u operačního systému Android. V iOS je veškerá zpětná komunikace prováděna pomocí funkce stringByEvaluatingJavaScriptFromString zprostředkované webovým runtimem UIWebView. [5]

2.3 Příklady hybridních frameworků

2.3.1 PhoneGap

Vzhledem k tomu, že hybridnímu frameworku PhoneGap je věnována celá následující kapitola, zmíním ho zde opravdu jen velmi stručně.

PhoneGap je framework vlastněný a vyvíjený společností Adobe. Jedná se o zdaleka nejpopulárnější řešení na poli hybridních frameworků, což dokládá i fakt, že řada dalších hybridních frameworků na PhoneGapu staví a začleňuje jej do vlastních řešení.

PhoneGap v současné době oficiálně podporuje 7 mobilních platforem. V době psaní této bakalářské práce PhoneGap pokrýval 17 nativních API volání. Mezi ty nejpoužívanější dozajista patří geolokace, přístup k filesystému, fotoaparát či akcelerometr. Všechna API volání si podrobněji popíšeme v další kapitole spolu s detailním popisem celého frameworku PhoneGap.

2.3.2 Sencha Touch

Zařazení produktu Sencha Touch mezi hybridní frameworky je možná trochu odvážné. Já se toho přesto dopustím, protože Sencha Touch ve svých nejnovějších verzích poskytuje nástroje pro tvorbu hybridních aplikací, ač je jinak stále dominantně UI frameworkem.

Sencha Touch má za sebou již tři roky vývoje. Jejím primárním úkolem vždy bylo poskytovat vývojářům možnost rychle vyvíjet uživatelské rozhraní mobilních webových aplikací optimalizované pro použití na zařízeních s dotykovým displejem. Neméně důležité však je, aby zobrazované rozhraní vypadalo na dané platformě co možná nejvěrněji. V tomto přístupu se tedy příliš neliší od frameworku jQuery Mobile a dá se říci, že na poli javascriptových frameworků jsou právě jQuery Mobile a Sencha Touch těmi největšími konkurenty.

Hlavním posláním těchto UI frameworků je poskytnout vývojáři sadu vizuálních elementů jako jsou tlačítka, toolbary, seznamy, pop-up okna, formuláře, textová pole, přechody mezi stránkami atp., které mohou použít ve svých aplikacích, aniž by museli složitě pomocí CSS a JavaScriptu definovat tyto elementy sami. Zároveň jsou tyto vizuální prvky optimalizovány k maximální kompatibilitě a výkonu na co možná nejširším portfoliu webových prohlížečů.

Sencha v současné době podporuje 4 platformy (iOS, Android, Blackberry a Windows Phone 8). Použití tohoto produktu je pro většinu účelů

zdarma, platit musíte jen pokud chcete na jeho základě postavit vlastní komerční framework.

Nyní si pojďme osvětlit, proč zařazuji Sencha Touch mezi hybridní frameworky. Vývojáři tohoto produktu totiž do jeho druhé verze přidali (v rámci nástroje Sencha Touch Preview SDK Tools) možnost přístupu k některým nativním API voláním mobilních platforem. Nabízejí také možnost zabalení webové aplikace do nativních binárních souborů vhodných k publikaci na oficiálních tržištích jednotlivých mobilních platforem. Sluší se dodat, že celá tato funkcionalita je stále značně omezená, protože podporovány jsou pouze dvě platformy (iOS a Android) a dostupné jsou pouze 4 API volání (indikace připojení k internetu, nativní notifikace, orientace zařízení a přístup k fotoaparátu). Dá se však očekávat, že se vývojáři pracující na frameworku Sencha Touch budou snažit dále rozšiřovat tento nástroj, aby podporoval více API volání a více platforem.

2.3.3 MoSync Wormhole

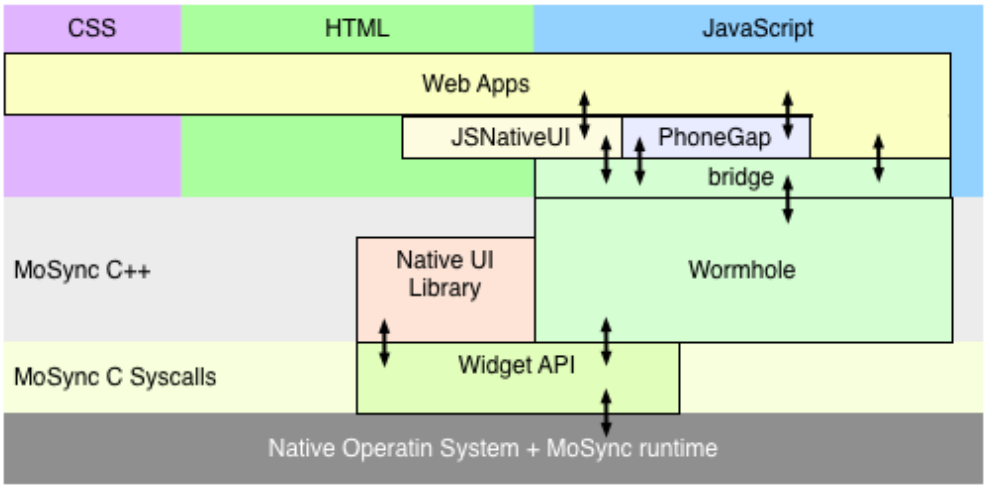
MoSync je velice zajímavý framework, který pokračuje de facto tam, kde PhoneGap končí. Nejedná se o klasický hybridní framework, neboť možnosti jeho využití jsou mnohem širší a hybridní funkcionalita do něj byla přidána až relativně nedávno.

Prapůvodním důvodem vzniku MoSync frameworku bylo umožnit vývojářům, kteří ovládají jazyky C nebo C++, vyvíjet nativní mobilní aplikace bez nutnosti měnit své programovací návyky a zároveň jim dát možnost zasáhnout více platforem najednou. Později byla přidána možnost vytvářet webové aplikace s využitím HTML5 a také hybridní aplikace s využitím obojího: HTML5 i nativního mostu napsaného v C/C++.

Vývojářům je tedy umožněno vytvářet klasické hybridní aplikace, ve kterých mohou přistupovat k nativním funkcím telefonu (geolokace, fotoaparát, filesystém apod.) a finální aplikace zabalit do nativních balíčků připravených na distribuci na oficiální tržiště. V současné době jsou pro hybridní účely podporovány tři hlavní platformy iOS, Android a Windows Phone[e].

Na obrázku 2.3 můžeme vidět, jak funguje komunikace mezi webovou aplikací a nativním operačním systémem.

Celá tato hybridní technologie se nazývá Wormhole a její javascriptová část implementuje některé technologie z frameworku PhoneGap stejně jako některé novinky z HTML5 standardu (například geolokace). Velkou výhodou, kterou MoSync oproti PhoneGap nabízí, je možnost využít ve své aplikaci nativní UI elementy jako jsou tlačítka, různé lišty atp. To může být pro vývojáře skutečně zajímavé, protože bude využívat pravé nativní prvky



Obrázek 2.3: Architektura frameworku MoSync Wormhole [9]

místo často těžkopádných (v CSS emulovaných) náhražek. Tato technologie je však logicky úkrokem stranou od klasických wrapperů, které balí webové aplikace do webového runtimu. Jelikož webový runtime neumí vykreslovat nativní elementy, je pro tyto účely využit MoSync runtime, který pracuje s hybridní aplikací na podobném principu jako framework Titanium zmíněný ve třetí kapitole.

Fakt, že MoSync implementuje funkcionalitu z frameworku PhoneGap dodává tomuto nástroji další přidanou hodnotu. Většina aplikací napsaná ve PhoneGap je s MoSync kompatibilní a tak mohou vývojáři bezbolestně rozšířit své PhoneGap aplikace o nativní UI elementy, které poskytuje nástroj MoSync.

Framework MoSync je open-source a je šířen zdarma.

2.3.4 IBM Worklight

Worklight je vývojová platforma vyvíjená pod záštitou technologického giganta IBM. Umožňuje vývoj nativních, webových i hybridních aplikací. Tato platforma může být zajímavá zejména pro využití ve firmách, neboť pro enterprise účely přináší několik zajímavých nástrojů.

IBM Worklight umožňuje vyvíjet hybridní aplikace třemi způsoby [45]:

1. Vývoj klasické hybridní aplikace. Z vašeho javacriptového kódu voláte funkce operačního systému přes nativní most, který zajišťuje nástroj PhoneGap. Jedná se tedy o identický přístup, který používá většina hybridních frameworků.

2. Vývoj částečně nativní aplikace. Vývojář může ve své aplikace využít zmenšeného web view a zbytek UI implementovat pomocí nativních UI komponent.

3. Vývoj částečně hybridní aplikace. Uživatelé vyvíjejí kompletní uživatelské rozhraní pomocí nativních jazyků, ale pokud je to potřeba, mohou využívat i web views s hybridní funkcionalitou.

IBM Worklight je celá sada nástrojů pro vývoj mobilních aplikací. Základem je IBM Worklight Studio – IDE postavené na open-source platformě Eclipse, které krom klasických funkcí nabízí i možnost tvorby uživatelského rozhraní pomocí WYSIWYG editoru. Vývojář si pomocí funkce drag & drop přetahuje jednotlivé UI elementy (podporovány jsou HTML5, jQuery Mobile a Dojo Mobile) a modeluje si tím vzhled své aplikace.

Dalším užitečným nástrojem je IBM Worklight Server, který slouží jako middleware mezi uživatelskými aplikacemi a různými back-endovými systémy či cloudovými službami. Tato služba také řídí veškerý vzdálený přístup k aplikaci. Pro firemní účely dále obsahuje platforma Worklight nástroje jako jsou IBM Worklight Application Centre, IBM Worklight Centre a IBM Worklight Device Runtime Components, které slouží ke vzdálené správě (přístup, push notifikace, analýza používání atd.) či distribuci (oficiální či privátní tržiště) aplikací.

2.3.5 FeedHenry

FeedHenry je společnost stojící za nástrojem Mobile Application Platform. Tato platforma je určena především pro enterprise segment trhu. Pro tvorbu hybridních aplikací je využíván PhoneGap, který zajišťuje propojení webové a nativní části.

Mobile Application Platform se zaměřuje především na pohodlné napojení aplikace na firemní back-end systémy a také na bezpečnost finálních aplikací, což jsou v korporátním prostředí vyhledávané benefity. FeedHenry dále poskytuje nástroje pro správu a analýzu aplikací v cloudu. Umožňuje jednoduché odeslání aplikací na privátní firemní tržiště, kde mohou firmy publikovat aplikace pro své zaměstnance či klienty.

2.3.6 Icenium

Icenium je, co se uživatelského přístupu týká, velmi inovativní nástroj. Veškerý vývoj totiž přesunul do cloudu. Nemusíte si stahovat žádné IDE či SDK. Pouze se zaregistrujete a vyvíjíte hybridní aplikace přímo v prohlížeči. Váš kód je umístěn buď ve verzovacím systému, který poskytuje samotný framework nebo můžete využít nástroje Git.

V cloudu se nachází i SDK jednotlivých platforem (v současnosti jsou podporovány pouze iOS a Android) a také knihovny nástroje Apache Cordova (PhoneGap), jehož služeb Icenium využívá. Krom vývojového prostředí se v cloudu nachází také testovací nástroje, které vývojářům umožňují real-time zobrazení uživatelského rozhraní a otestování funkčnosti samotné aplikace.

2.3.7 WYSIWYG frameworky s využitím PhoneGap

Do této skupiny řadím sobě podobné frameworky jako jsou například ApplicationCraft či Tiggzi, které vykazují společné znaky zejména v tom, že využívají pro tvorbu hybridních aplikací technologii z frameworku PhoneGap a zároveň jako primární způsob vývoje nabízí vývojářům v cloudu integrované WYSIWYG nástroje, kde je tvorba aplikací realizována pomocí mechanismu drag & drop. Vývojář (nebo spíše modelář) aplikace skládá jednotlivé vizuální či funkční elementy a postupně tak vytváří svojí aplikaci. Díky využití PhoneGap může přistupovat k nativním funkcím operačního systému a následně hotovou aplikaci zabalit do nativního balíčku a distribuovat na oficiální tržiště.

2.3.8 Srovnávací tabulka

V následující tabulce naleznete přehledné porovnání jednotlivých hybridních frameworků dle následujících kritérií:

1. Podporované platformy
2. Podporovaná nativní API volání
3. Poskytnutí vlastního IDE
4. Licence a cena

Kapitola 3

PhoneGap

3.1 Obecně

Jak již bylo v textu této práce několikrát zmíněno, PhoneGap je v současné době pravděpodobně nejpoužívanější a nejznámější hybridní framework. Vývojáři se na svém webu chlubí, že zaznamenali již více než milion stažení jejich produktu a zároveň tvrdí, že je v současné době používán více než 400 000 vývojáři pro tvorbu tisícovek aplikací.

PhoneGap umožňuje vývojářům využívat při tvorbě mobilních aplikací webové technologie HTML5, CSS3 a JavaScript a zároveň jim prostřednictvím vlastního javascriptového API poskytuje možnost komunikace s operačním systémem a využití jeho nativních součástí, jako je například fotoaparát, akcelerometr nebo například výběr ze seznamu kontaktů. To vše multiplatformně.

V rámci této kapitoly již nebudu zabíhat do technických detailů vlastního fungování frameworku PhoneGap. Tuto oblast jsem již dostatečně pokryl v předchozí kapitole 2.

„Konečným cílem PhoneGap je, aby PhoneGap již nemusel existovat.“ [34] [5]

Skupina vývojářů stojící za PhoneGap uvádí, že za jejich prací stojí následující dvě kréda:

1. Web vyřešil problém přenositelnosti.

[5] „The ultimate purpose of PhoneGap is to cease to exist.“ [34]

2. Každá technologie zastará.

S prvním krédem se dá polemizovat jen velmi těžko. Program, umožňující zobrazovat webové stránky, najdeme dnes skutečně téměř na každé platformě. Co se ne zcela povedlo Javě, se webovým technologiím daří a jejich význam roste. Ne náhodou dnes používáme mnohem více webových aplikací než dříve a spousta aplikací, u kterých to bylo dříve nemyslitelné, se přesunula do cloudu a přistupujeme k nim přes webový prohlížeč. Jedním z nejzářnějších příkladů jsou například Google Dokumenty, které přispěly i k tvorbě této práce.

Krom těchto dvou kréd uvádějí vývojáři i dva cíle, ke kterým by jejich snažení mělo přispět:

1. Vylepšit web (webové standardy), aby se stal prvotřídní vývojovou platformou.
2. Docílit takového stavu webových standardů, aby byl PhoneGap zbytečný.

Je pravdou, že různé týmy uvnitř konsorcia W3C pracují na vytvoření standardů, které umožní přístup k nativním funkcím telefonního přístroje přímo z prohlížeče bez použití jakékoliv mezivrstvy. Typickým příkladem realizace těchto snah je například standard pro využití geolokace, který již byl moderními prohlížeči implementován a je využit i v rámci PhoneGap, kde API volání pro geolokaci nedělá nic jiného, než že volá tuto standardizovanou funkci prohlížeče. Že cíle, které si tým kolem PhoneGap vytyčil, nejsou řečmi do větru, dokládá i fakt, že vývojáři stojící za tímto frameworkem jsou součástí právě těch skupin v rámci konsorcia W3C, které se starají o implementaci příslušných standardů.

3.1.1 Licence

Poté co byla firma Nitobi, která za PhoneGap původně stála, odkoupena společností Adobe, byl kód pohánějící framework PhoneGap věnován organizaci Apache Software Foundation pod názvem Apache Cordova. Nyní je tedy šířen pod licencí Apache License, Version 2. Tato licence nadále zaručuje otevřený přístup k vývoji na projektu i k použití samotného kódu.

3.1.2 PhoneGap vs. Apache Cordova

Zejména u lidí, kteří se s PhoneGap teprve seznamují, koluje mnoho nejasností ohledně rozdílu mezi PhoneGap a jeho dvojčetem Apache Cordova. Myslím, že stojí za to na tomto místě tyto nejasnosti ozřejmit.

Apache Cordova je jádro frameworku spravované Apache Software Foundation (ASF), na jehož základě je postavený framework PhoneGap. Dalo by se říci, že PhoneGap je distribucí Apache Cordova podobně, jako je operační systém Ubuntu distribucí, kterou pohání jádro Linux. Vývojáři PhoneGap nadále doplňují Apache Cordova o nové funkce a opravují nalezené chyby. K oddělení došlo zejména proto, že postupem času se dá očekávat hlubší integrace PhoneGap s nástroji Adobe, což by nebylo vhodné pro masové použití [30].

PhoneGap není jedinou distribucí Apache Cordova, ve skutečnosti jí dnes implementuje do svých nástrojů mnoho firem, jako je například Salesforce nebo Facebook [30]. Na vývoji Apache Cordova se nepodílí jen vývojáři PhoneGap, ale také mnoho dalších firem jako je například IBM či Microsoft.

3.2 Historie

„Měli jsme v hlavě jen jeden cíl. Umožnit webovým aplikacím využívat nativní funkce iPhone.“[35] [6]

Pojďme se stručně seznámit s historií PhoneGap a změnami, které vedly k frameworku, jak jej známe dnes. PhoneGap vznikl v roce 2008 na vývojářské akci iPhoneDevCamp, kde vývojáři firmy Nitobi přišli s nápadem, kterak umožnit vývojářům webových aplikací pro systém iOS (tehdy ještě iPhone OS) využívat některé oblíbené nativní funkce přístroje iPhone. V soutěži na iPhoneDevCamp sice PhoneGap neuspěl, to však firmu Nitobi nezastavilo v dalším rozvíjení svého nápadu.

V počátku své existence podporoval PhoneGap pouze systém iOS, ale multiplatformnost byla brána vývojáři na zřetel od samého počátku, a tak rychle přibyla podpora pro Android a BlackBerry. Nedlouho poté (v roce 2009) následovala podpora pro Symbian a Palm (webOS).

K začátkům frameworku PhoneGap se datuje série nepříjemností, které vznikly při schvalovacím procesu ze strany Applu, kdy Apple odmítal schvalovat a vpouštět na své oficiální tržiště aplikace napsané pomocí frameworku PhoneGap. Důvodem pro zamítání aplikací bylo porušení podmínek, které aplikace v AppStore musí splňovat. Aplikace napsané ve PhoneGap byly penalizovány, protože při jejich vývoji byl použit externí framework, což tyto podmínky zapovídají. Celá záležitost byla vyřešena domlouvou mezi vývojáři kolem frameworku PhoneGap a Applem, kdy byla stanovena „bezpečná verze“ frameworku PhoneGap (PhoneGap 0.8.0), jejíž použití bylo Applem

[6] „We went down with one goal in mind and that was to make native iPhone features available to web apps.“ [35]

oficiálně posvěceno a k aplikacím využívajícím tuto „bezpečnou verzi“ bylo během schvalovacího procestu přistupováno stejně jako k běžným nativním aplikacím. [14] + [42] + [36] + [31]

Multiplatformní záběr se stal zas o něco širším, když v roce 2010 přidal PhoneGap podporu pro Windows Phone 7 a také kvůli přechodu BlackBerry na WebKit, v šesté verzi svého operačního systému, byla podpora pro BlackBerry rozštěpena na dvě větve.

V listopadu 2010 Nitobi představilo v poloveřejné betě nástroj PhoneGap Build. Tento nástroj si podrobněji popíšeme později v této kapitole, teď jen zmíním, že se jedná o cloudovou službu, která umožňuje po nahrání zdrojových kódu vygenerovat instalační balíčky pro více platforem najednou. Vývojář tak nemusí pro každou platformu instalovat její SDK a generování binárek provádět ručně [20].

Pokračující růst PhoneGap dokládá i obliba, které se této framework těšil pouhé dva roky po svém vzniku. V listopadu 2010 zaznamenal PhoneGap již 350 000 stažení. S touto vzrůstající oblibou začalo Nitobi (původně konzultantská firma zabývající se tvorbou webů a javascriptových knihoven) na PhoneGap stále více sázet ve své obchodní strategii. Nejenže s ním pracovala ve své konzultantské činosti a doporučovala jeho používání svým zákazníkům, ale v únoru 2011 zavádí i placenou podporu pro vývojáře ve PhoneGap, v rámci které jim poskytuje webináře, přístup na privátní forum a pomoc v rámci office hours.

Dalším znakem úspěchu PhoneGap je i jeho adopce softwarovými giganty jako je Adobe či IBM, které začali tento framework integrovat do svých nástrojů, jako je Adobe DreamWeaver či IBM Worklight [27] + [15].

Prvním velkým milníkem pro PhoneGap bylo vydání verze 1.0.0, ke kterému došlo 29. 7. 2011 [25]. V té době PhoneGap již podporoval i platformu Samsung Bada a samotný framework zaznamenal 600 000 stažení s průměrem kolem 40 000 stažení měsíčně [25].

V září 2011 se PhoneGap Build přesouvá do veřejné betaverze a zároveň byly zveřejněny cenové plány pro jeho použití. Ty se pohybovali od $0 pro vývoj open-source aplikací po $900 ročně pro vývoj komerčních aplikací s uzavřeným kódem. [29]

3. září 2011 Nitobi na svém blogu oznámilo, že uzavřelo dohodu o svém převzetí firmou Adobe [24].

„PhoneGap je perfektním doplněním široké palety nástrojů pro vývojáře od společnosti Adobe (včetně Adobe AIR) a dovolí nám nadále poskytovat vývojářům ta nejlepší a nejmodernější řešení pro vytváření inovativních apli-

kací napříč platformami a zařízeními. “ [24] [7]

V rámci tohoto převzetí Nitobi oznámilo, že věnuje zdrojové kódy frameworku PhoneGap organizaci Apache Software Foundation (ASF) pod názvem Apache Callback, který byl později změněn na Apache Cordova [22] + [26].

Akvizice Nitobi firmou Adobe se podařila úspěšně dokončit 27. října 2011 [22]. Analytici se shodují, že hlavním smyslem této akvizice byla snaha Adobe posílit svojí pozici na poli vývoje aplikací v HTML5, protože jeho vlastní technologie Flash je (zvlášť v mobilním světě) na ústupu. Očekávalo se začlenění PhoneGap do vývojářských nástrojů, které Adobe poskytuje webovým vývojářům.

Nehledě na tuto velkou změnu popularita PhoneGap dále pozvolna stoupala a v červnu roku 2012 dosahoval průměrný měsíční počet stažení hodnoty 100 000 [33].

20. července 2012 byla oficiálně uvolněna verze 2.0, která (krom jiného) přinášela také Cordova WebView, což je možnost použití WebView v rámci nativní aplikace současně se všemi výhodami PhoneGap. Dále přibylo také mnoho vylepšení command-line interfacu, které umožňuje ovládat PhoneGap z konsole [16].

V září 2012 byl oficiálně spuštěn PhoneGap Build, který je nyní dostupný ve stabilní verzi [28]. Zároveň byla v listopadu 2012 vydána verze 2.2.0, která přináší podporu Windows 8 v rámci PhoneGap. V době psaní této práce se framework nachází ve verzi 2.5.0. Vydání verze 3.0.0 je naplánováno na červenec 2013 [39].

3.2.1 Ocenění

PhoneGap se za dobu své existence dočkal celé řady ocenění od všemožných institucí.

1. 2009: People's Choice award v rámci Web 2.0 Expo LaunchPad competition [32].
2. 2012: Best Cross-Platform Development Tool by CodeProject [17]
3. 2012: Technology of the Year By InfoWorld [7]

[7] "It's a perfect complement to Adobe's broad family of developer solutions, including Adobe AIR, and will allow us to continue to provide content publishers and developers with the best, cutting-edge solutions for creating innovative applications across platforms and devices." Danny Winokur, vice president and general manager, Plat- form, Adobe [24]

4. 2012: “Champion” by Info-Tech Research Group v rámci 2012 Vendor Landscape: Mobile Development Platforms [19]

5. 2012: Best Mobile Development Tool by Dr. Dobb’s Jolt Awards [18]

3.3 Vývojářské nástroje

3.3.1 Vývoj

Na začátku této sekce krátce pohovořím o nástrojích, které můžeme pro vývoj ve PhoneGap využít. Před vznikem cloudové služby PhoneGap Build byl vývoj, zvláště pokud jste chtěli aplikaci cílit na více platforem, poměrně kostrbatý. To bylo zapříčiněno několika faktory:

1. Různá SDK pro sestavení nativních binárek

2. Různá adresářová struktura projektů pro jednotlivé platformy

3. Odlišné javascriptové knihovny (PhoneGap) pro jednotlivé platformy

V důsledku to znamenalo, že pro vývoj aplikace pro systém Android jste si museli stáhnout Eclipse a Android SDK a vývoj vést v těchto nástrojích. Pokud jste následně chtěli aplikaci portovat na iOS, znamenalo to stažení iOS SDK a Xcode (funguje pouze na systému Mac OS), úpravu adresářové struktury projektu a využití knihovny PhoneGap určené pro systém iOS.

Díky cloudovému nástroji PhoneGap Build (podrobněji si ho popíšeme níže v této kapitole) se vývoj aplikací ve PhoneGap výrazně usnadnil a to zejména při cílení na více platforem. Jelikož PhoneGap Build potřebuje k vygenerování nativních binárek pouze samotné zdrojové kódy aplikace, nemusí se vývojář zabývat ani jedním z výše vyjmenovaných problémů.

Pokud vývojář není závislý na SDK konkrétní platformy, může pro vývoj použít prakticky libovolný nástroj. Může zůstat u mohutného IDE, jako je např. Eclipse nebo využít svůj oblíbený textový editor (TextMate, gedit, Sublime Text apod.).

3.3.2 Rozšíření

Pokud vyvíjíme pro konkrétní platformu nebo chceme jen přidat některé specifické funkce, které PhoneGap nativně nenabízí, můžeme sáhnout do poměrně bohaté sbírky rozšíření, která vznikla v rámci komunity, ale je oficiálně spravována projektem PhoneGap na adrese `<https://github.com/purplecabbage/phonegap-plugins>`.

Najdeme tu bohatou sbírku pluginů, které pokrývají širokou škálu funkcí. Můžeme si některé z nich uvést:

1. InAppPurchaseManager (iOS)
2. BarcodeScanner
3. GoogleAnalytics
4. PDFViewer
5. Twitter
6. iCloudKV (iOS)
7. Bluetooth
8. NFC
9. PayPalPlugin

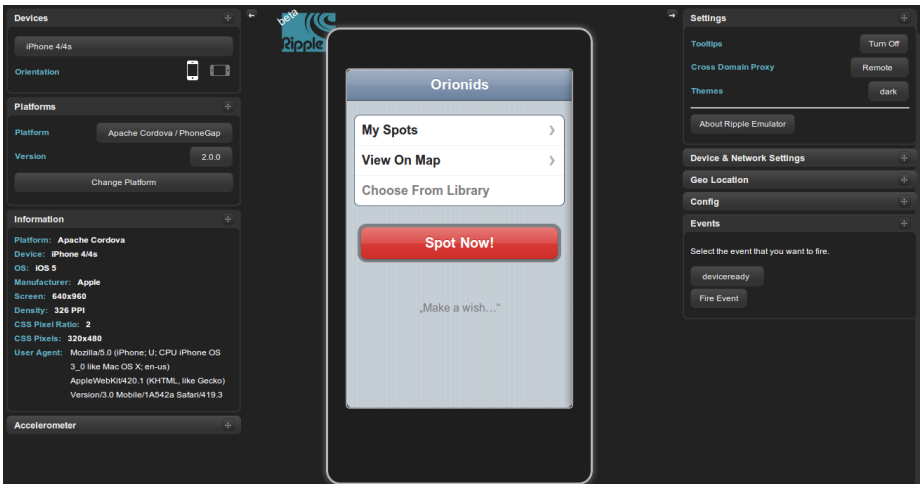
Pluginy se používají podobným způsobem jako samotný framework. Konkrétní plugin se vloží do projektu a jeho funkce jsou následně volány z javascriptového kódu vlastní aplikace. Logicky je nutné počítat s tím, že použití takového pluginu silně ochromí přenositelnost celé aplikace a před provedením portu bude třeba provést odpovídající úpravy.

3.3.3 Debugování a testování

3.3.3.1 Testování

V průběhu celého vývoje aplikace je třeba neustále testovat a ověřovat její funkčnost v prostředí co možná nejbližším produkčnímu. Pro tento účel poskytují tvůrci jednotlivých platforem simulátory (emulátory), které simulují běh aplikace v rámci daného operačního systému či dokonce přímo v rámci konkrétního zařízení.

V případě testování není žádný rozdíl mezi nativními aplikacemi a těmi hybridními napsanými ve PhoneGap. Vývojář nahraje svojí aplikaci do simulátoru a zkouší její funkčnost. To samé se týká testování aplikace na reálném zařízení, které by mělo být součástí každého vývojového plánu.



Obrázek 3.1: Ripple Mobile Enviroment Emulator

3.3.3.2 Debugování

Při vývoji čistě nativních aplikací využívá vývojář pro ladění svých aplikací debugovací nástroje, které mu poskytuje SDK a další nástroje konkrétní platformy. Problém s těmito nástroji je, že většinou nejsou připravené na ladění kódu psaného webovými jazyky. To staví vývojáře do poměrně nepříjemné pozice, kdy jsou buď nuceni protkat svůj kód voláním funkce alert() či console.log(), které používají pro vypsání ladících výpisů nebo využít některého z nástrojů třetích stran, které jsou na ladění hybridních aplikací připraveni. Dva takové nástroje si nyní uvedeme.

Ripple Mobile Enviroment Emulator Ripple Mobile Enviroment Emulator (RMEE) je náhrada klasického emulátoru z SDK jen s tím rozdílem, že je velmi odlehčený a běží jako rozšíření ve webovém prohlížeči Google Chrome či Chromium. RMEE narozdíl od holého webového prohlížeče dokáže nasimulovat i volání nativních funkcí frameworkem PhoneGap a je tak velice vhodným nástrojem pro rychlé testování mobilních aplikací bez nutnosti zdlouhavě nahrávat aplikaci do emulátoru nebo vlastního fyzického zařízení.

Výhodou použití RMEE je, že dává vývojáři k dispozici krom samotného emulátoru i veškeré vývojářské nástroje prohlížeče Google Chrome, které se používají pro ladění webových aplikací. Může tak za běhu upravovat některé parametry své aplikace a pohodlně tak ladit vlastní aplikaci.

Je jasné, že RMEE nenahradí testování v klasickém emulátoru či dokonce na fyzickém zařízení, neboť zdaleka nedokáže emulovat veškerá spe-

cifika hostitelských operačních systémů, ale stále je poměrně vhodným nástrojem pro rychlé testování aplikace zvláště v rané fázi vývoje.

RMEE je dostupný z adresy <http://emulate.phonegap.com>

Web Inspector Remote Web Inspector Remote neboli Weinre je ladící nástroj trochu jiného druhu. Nezabývá se emulací běhu aplikace, ale funguje jako vzdálený ladící nástroj. Jeho použití je poměrně jednoduché. Do kódu své aplikace vložíte skript umístěný na vzdáleném serveru (<http://debug.phonegap.com>) s unikátním označením své aplikace.

```
<script src="http://debug.phonegap.com/target/target-
    script-min.js#myapp"></script>
```

Samotné debugování probíhá v prohlížeči, do kterého vývojář zadá adresu serveru s identifikátorem laděné aplikace <http://debug.phonegap.com/client/#myapp>. Vývojáři je k dispozici javascriptová konzole a další nástroje pro ladění webových aplikací.

3.3.4 Sestavení a publikace

Pokud již máme aplikaci vytvořenou a důkladně otestovanou, je na řadě její sestavení a případná publikace například na oficiálním tržišti s aplikacemi. Za účelem sestavení aplikace nám PhoneGap umožňuje jít dvěma cestami. První z nich je sestavení nativního instalačního balíčku s využitím SDK konkrétní platformy vhodný k odeslání na tržiště. Toto je klasická cesta a je vhodná, pokud cílíme pouze na jednu platformu. Problém nastává, pokud chceme vytvořit balíčky pro více platforem najednou. Jak jsem již zmínil výše, v takovém případě jsme nuceni stáhnout SDK každé jednotlivé platformy, náš projekt upravit a poté s pomocí SDK vytvořit nativní binárku. Že toto není dobrý způsob, si uvědomovali i tvůrci frameworku PhoneGap a proto již v počátku ve svých úvahách počítali s cloudovým nástrojem, který sestavování aplikací značně usnadní.

3.3.4.1 PhoneGap Build

PhoneGap Build je nástroj dostupný z adresy <http://build.phonegap.com>. Umožňuje nám z jednoho místa a z jednoho zdrojového kódu vytvořit v jednom čase instalační balíčky pro platformy iOS, Android, Windows Phone, BlackBerry, webOS a Symbian.

Pojďme si v krátkosti popsat jak celý systém funguje. Vývojář je na začátku vyzván k propojení PhoneGap Build se svým účtem na Githubu, na němž má ve veřejném repozitáři umístěny zdrojové kódy své aplikace. Poté

se dostane do ovládacího panelu, kde vybere z nabídky správný repozitář, ve kterém se PhoneGap Build pokusí najít soubor index.html. Jakmile se mu to podaří, nabídne vývojáři možnost si vygenerovat instalační balíčky pro jednotlivé platformy pomocí jednoho kliknutí. Vyhotovení balíčků trvá několik vteřin a poté jsou uživateli k dispozici ke stažení.

3.4 API

Hlavním důvodem, proč vývojáři sahají po použití frameworku PhoneGap, je sada javascriptových API, pomocí které zprostředkovává komunikaci mezi webovou aplikací a operačním systémem na daném zařízení. Pojďme si nyní stručně popsat jednotlivá API volání.

3.4.1 Accelerometer

Akcelerometr se používá ke zjištění orientace daného přístroje. Hojně je využíván například v hrách, kde hráč ovládá hru pomocí naklopení telefonu.

Tento objekt obsahuje sadu metod, které umožňují vrátit současnou polohu přístroje pomocí trojice souřadnic x, y a z. Je také možné nastavit sledování změn polohy v zadaném intervalu. Umožňuje zrušit sledování změn vyvolané předchozí metodou.

3.4.2 Camera

Sada funkcí sloužící pro přístup k fotoaparátu mobilního zařízení. Umožňuje pořízení fotografie z daného zdroje. Zdroj je specifikován ve volání funkce getPicture, může se jednat buď o pořízení nového snímku nebo o vybrání již pořízeného snímku z fotoalba. Fotografie je vrácena buď jako URI nebo jako String zakódovaný v base64.

3.4.3 Capture

API umožňující zachycení videa, audia či fotografie. Funguje tak, že otevře výchozí aplikaci pro pořízení daného média a vrátí zpět informace o pořízených záznamech jako URI.

3.4.4 Compass

Sada metod umožňující rozpoznat, jakým směrem je zařízení namířeno. Zavoláním funkce getCurrentHeading získáme informace o aktuální orientaci

zařízeni vzhledem ke světovým stranám ve stupních od 0 do 359,99. Údaje o orientaci přístroje můžeme také snímat v pravidelném intervalu pomocí funkce watchHeading.

3.4.5 Connection

Connection je objekt, který uchovává informace o připojení daného zařízení. Umožňuje vývojáři rychle zjistit, zda je telefon připojen k WiFi, 3G či kabelovému připojení, ale také dokáže ověřit, že přístroj není k internetu připojen vůbec.

3.4.6 Contacts

Metodami tohoto objektu může vývojář přistupovat k nativní databázi kontaktů v daném mobilním zařízení. Kontakt je možno v databázi buď vyhledat či vytvořit zcela nový pomocí funkce create.

3.4.7 Device

V objektu Device vývojář nalezne důležité informace o hardware a software daného zařízení. Obsahuje informace o názvu zařízení, aktuálně používané verzi PhoneGap, operačním systému a jeho verzi a také UUID identifikátor daného přístroje.

3.4.8 Events

V aplikaci často potřebujeme reagovat na události vyvolané přímo uživatelem nebo operačním systémem. PhoneGap nám to umožňuje pomocí sady předdefinovaných událostí. Vývojář může reagovat například na stisknutí tlačítka zpět, ztišení nebo zvýšení hlasitosti, přepnutí aplikace do pozadí a mnoho dalšího. V rámci tohoto API můžeme také rozpoznat, zda-li je zařízení připojeno k internetu nebo jaký je stav baterie v přístroji.

3.4.9 File

Sada objektů implementujících W3C specifikaci API pro zápis, čtení a navigaci v nativním filesystému. Soubory či složky můžeme vytvářet, procházet, kopírovat, mazat apod.

3.4.10 Geolocation

API umožňující přístup k informacím o zeměpisné délce a šířce daného zařízení. Tyto informace mohou pocházet buď z GPS modulu nebo z jiných zdrojů jako například IP adresa či GSM připojení. Informace o poloze můžeme získat buď jednorázově či je snímat v pravidelném intervalu, což se může hodit například pro turistické či fitness aplikace.

3.4.11 Globalization

Objekt globalization obsahuje informaci o preferovaném jazyce, nastavené časové zóně a podobně.

3.4.12 InAppBrowser

Internetový prohlížeč, který se otevře v rámci aplikace po zavolání metody window.open. API nám umožňuje naslouchat událostem ze synovského prohlížeče a regovat na ně.

3.4.13 Media

Pomocí objektu Media může vývojář zajistit nahrávání a přehrávání audio souboru. Objekt obsahuje rozmanité funkce, můžeme například získat aktuální pozici v právě přehrávaném souboru, zjistit jeho délku, posunout se v něm apod. Nechybí ani standardní metody pro spuštění nebo zastavení přehrávání či nahrávání.

3.4.14 Notification

Umožňuje upozornit uživatele pomocí různých druhů notifikací. Příkladem upozornění může být vibrace, alert box nebo pípnutí.

3.4.15 Splashscreen

Pomocí tohoto API může vývojář ovlivnit zobrazování splashscreenu při startu aplikace.

3.4.16 Storage

API založené na W3C specifikaci umožňující ukládání dat do lokální databáze. Metoda openDatabase vytvoří nový objekt Database, nad kterým můžeme provádět transakce pomocí metody executeSql objektu SQL Transaction.

3.4.17 Tabulka podpory API volání v rámci platforem

Jednotlivá API volání bohužel nejsou podporována konzistentně napříč platformami. V následující tabulce najdete přehled podpory jednotlivých volání napříč mobilními platformami.

3.5 PhoneGap v reálných aplikacích

Pokud jsme dosud slyšeli jen čísla, kolikrát byl PhoneGap stažen, je čas se přesvědčit, že jej vývojáři opravdu využívají při reálném vývoji masově rozšířených aplikací. Vybral jsem 3 nejzajímavější aplikace, při jejichž vývoji byl využit PhoneGap.

3.5.1 Wikipedia

Otevřená encyklopedie Wikipedia používá PhoneGap pro vývoj oficiální mobilní aplikace, přes kterou mohou uživatelé vyhledávat, zobrazovat a ukládat jednotlivé články z nejobsáhlejší encyklopedie světa. Aplikace je v současnosti portována na platformách iOS, Android a BlackBerry Playbook.

3.5.2 BBC Olympics

Pro účely letních olympijských her v Londýně v létě 2012 vznikla pro britskou veřejnoprávní televizi BBC aplikace, která umožnila uživatelům přísun čerstvých novinek z dění na sportovištích. Aplikace nabízela například živé videopřenosy ze všech sportovišť, živý textový komentář k jednotlivým sportům, sestřihy toho nejlepšího, možnost čtení nejzajímavějších článků offline a mnoho dalšího. [21]

BBC Olympics byla dostupná na třech platformách – iOS, Android a Blackberry.

3.5.3 Fruit Salad

Fruit Salad je hra, kde se hráč snaží skládat pod sebe stejné druhy ovoce a získávat za to body. Tato hra vytvořená pomocí HTML5 je dostupná pro platformu Android.

„PhoneGap Build je skvělý nástroj pro webové vývojáře, kteří chtějí zveřejnit své HTML aplikace na App Store.“ [23] [8]

3.6 Budoucnost

Co očekávat od budoucích let v souvislosti s PhoneGap? Jednoznačně mnohem hlubší integraci s Adobe produkty. Ačkoliv se to již u některých produktů (Adobe DreamWeaver) stalo, od převzetí PhoneGap firmou Adobe se očekávalo v tomto ohledu mnohem více. V dalších letech tak můžeme očekávat zařazování PhoneGap do vývojářských nástrojů, které Adobe poskytuje vývojářům HTML5 aplikací.

Jak jsem již předeslal na začátku této kapitoly, cílem Phonegap je, aby PhoneGap nebyl třeba. Za tímto účelem se vývojáři PhoneGap účastní tvůrčích skupin v rámci konsorcia W3C pro vytváření nových webových standardů. Jakmile budou nové standardy vytvořeny, dá se očekávat jejich rychlá implementace do PhoneGap, jako je tomu již nyní u API pro geolokaci nebo u lokálního úložiště dat.

Dalším cílem bude větší a pohodlnější rozšiřitelnost frameworku. Pluginy by mělo být možné lépe vyhledávat a snadněji instalovat. Pokročilejší vývojáře jistě potěší i plánované vylepšování řádkového interface, kterým je možno framework ovládat z konsole. Zároveň se plánuje velká revize celého API a jeho pročištění a další sjednocení. Jedním z velkých cílů je, aby bylo přidání podpory pro nové platformy co možná nejjednodušší.

Když je řeč o nových platformách, bylo by dobré zmínit, které jsou další na řadě. Vývojáři ve svých plánech hovoří zejména o dvou – Tizen a Firefox OS. Tizen je open-source operační systém pro široké spektrum zařízení vyvíjený pod záštitou Linux Foundation. O Firefox OS jsem hovořil již v kapitole **??**. Jedná se o mobilní operační systém od společnosti Mozilla postavený čistě na webových technologií pomocí W3C standardů a také Mozilla WebAPI.

Vydání PhoneGap ve verzi 3.0.0 je naplánováno na červenec 2013. [39]

[8] “PhoneGap Build is a great tool for web developers who want to give App Store visibility to their HTML apps” said Fruit Salad creator, Baptiste Brunet. [23]

Kapitola 4

Hybridní vs. nativní vývoj

V rámci této kapitoly se pokusím co možná nejkomplexněji srovnat hybridní a nativní přístup. Jejich porovnání se bude týkat tří klíčových oblastí z životního cyklu aplikace.

4.1 Způsob vývoje

4.1.1 Náročnost vývoje

4.1.1.1 Lidské zdroje a finance

Tyto dvě kritéria jsem spojil do jednoho, protože spolu velmi úzce souvisí. Cena vývoje aplikace totiž naprosto dominantně záleží na lidech, kteří jí vytváří. Je zřejmé, že výsledná cena závisí na mnoha faktorech, jako je například typ aplikace (jiné náklady bude třeba vynaložit na tvorbu jednoduché To-Do aplikace, jiné na vývoj 3D hry), počet platforem, na které budeme aplikaci portovat nebo aktuální situace na trhu.

Obecně se však uvádí, že vývoj hybridní aplikace je oproti nativní variantě levnější. Hlavním důvodem je, že vývojáři znalý webových technologií stojí méně než kvalifikovaní vývojáři znalí technologií nutných pro vývoj pro konkrétní platformu. Odhaduje se, že vývoj nativní aplikace stojí mezi 20 000 a 150 000 dolary [46].

V obecné rovině také platí, že weboví vývojáři jsou na pracovním trhu poměrně lehce k sehnání a řada firem již takové vývojáře dokonce zaměstnává interně. V takovém případě je využití hybridního přístupu obzvláště lákavé.

Dle trendů na pracovním trhu, které sleduje server indeed.com, je poptávka po vývojářích ovládajících technologii HTML 5 skutečně velká. Jedná

Obrázek 4.1: Vývoj nabídek zaměstnání pro HTML5 vývojáře [10]

se dokonce o nejvíce vyžadovanou dovednost ze všech pracovních nabídek, které server pravidelně analyzuje. Strmý růst poptávky po HTML 5 vývojářích vidíme na grafu 4.1.

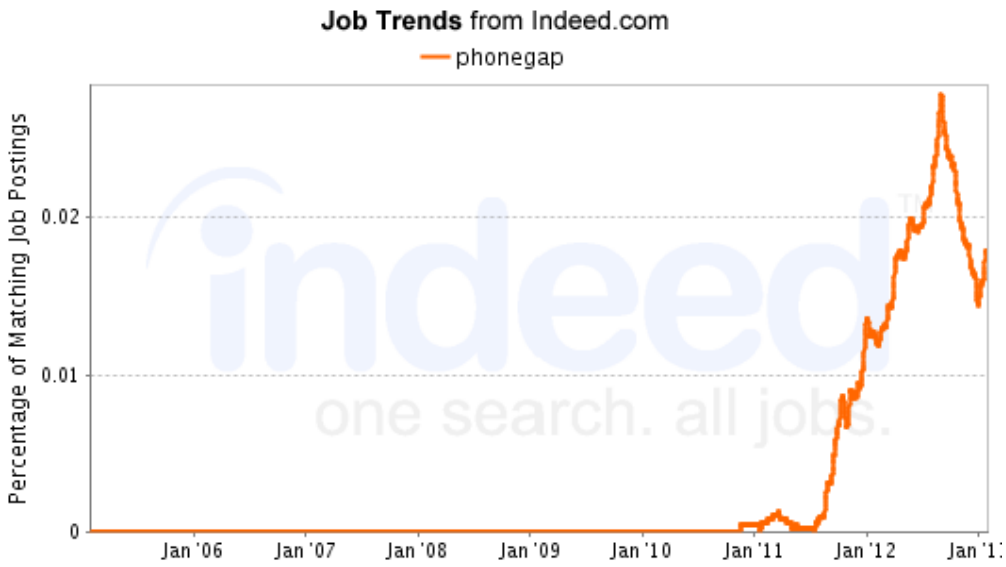
S tímto trendem se logicky svezla i poptávka po vývojářích specializovaných na přední hybridní framework PhoneGap.

Zajímavým ukazatelem je také průměrná mzda, kterou vývojáři s těmito schopnostmi dostávají. Průměrná mzda na pozicích, kde je vyžadována znalost HTML5 je 84 000 amerických dolarů ročně [10]. Zaměstnanec na pozici „HTML Developer“ pak pobírá 75 000 dolarů za rok [10]. Velice blízko těmto číslům je i průměrná mzda na pozicích, kde je jako schopnost vyžadována znalost frameworku PhoneGap, která dosahuje výše 86 000 dolarů [10].

Pokud se podíváme na nativní vývojáře, tak zjistíme, že poptávka po nich je jen nepatrně nižší.

Na datech o průměrné mzdě, která je nabízena vývojářům ovládajícím technologie pro vývoj pro iOS, je vidět, že jsou vývojáři s těmito schopnostmi nákladnější. Průměrná mzda v této oblasti dosahuje výše 89 000 dolarů za rok. Zaměstnanec na pozici „iOS Mobile Developer“ pobírá ročně 104 000 dolarů. O něco levnější je cena práce vývojáře pro platformu Android, kde si „Android Developer“ přijde na 101 000 dolarů za rok.

Dalším faktorem je také výše zmíněný počet platforem, pro které bude aplikace vyvíjena. Zatímco v případě vývoje pro jedinou platformu nemusí



Obrázek 4.2: Vývoj nabídek zaměstnání pro PhoneGap vývojáře [10]

Obrázek 4.3: Vývoj nabídek zaměstnání pro iOS vývojáře [10]

Tabulka 4.1: Srovnání ceny hybridního a nativního přístupu při vývoji aplikace pro 3 platformy [44]

Profese	Počet	Hodin [rok]	Roční náklady [USD]
iOS Architekt	1	2 000	$200 000
iOS Developer	3	6 000	$300 000
Celkem	**4**	**8 000**	**$500 000**
JavaScript Architekt	1	2 000	$200 000
JavaScript Developer	3	6 000	$300 000
iOS Developer	1	2 000	$100 000
Android Developer	1	2 000	$100 000
Windows Phone Developer	1	2 000	$100 000
Celkem	**7**	**14 000**	**$800 000**
Při vývoji pro 3 platformy			
Nativní tým	12	24 000	$1 500 000
Hybridní tým	7	14 000	$800 000
Úspory	**42 %**	**42 %**	**47 %**

být rozdíl v nákladech na vývojáře markantní, v případě cílení na dvě nebo více platforem již náklady na nativní vývojáře strmě rostou. Tuto situaci názorně dokládá následující tabulka, která ukazuje odhadované náklady na vývoj multiplatformní aplikace (iOS, Android, Windows Phone) pomocí nativní i hybridní cesty.

4.1.1.2 Složitost

Jednoznačně určit, který přístup je pro vývojáře složitější, je náročné. Pro každého jedince je subjektivně složité něco jiného.

Obecně lze říci, že vývoj nativních aplikací vyžaduje rozsáhlejší znalosti, než-li je tomu při vývoji pomocí webových technologií. I hybridní vývojář však musí velmi dobře rozumět javascriptovému frameworku, který používá. Je třeba si uvědomit, že vývoj mobilních aplikací se v mnoha ohledech výrazně liší od vývoje klasických webových stránek.

4.1.2 Rychlost vývoje

Toto kritérium je více než všechny ostatní závislé na komplexnosti vyvíjené aplikace. Velmi jednoduchá aplikace může být vyvinuta v řádu dnů až týdnů jak nativně tak hybridně. U složitější aplikace zas doba strávená jejím vývojem závisí na jiných faktorech (počet vývojářů, počet platforem apod.).

Výzkum společnosti Forrester Research udává, že vývoj nativní aplikace zabere vývojovému týmu průměrně 6 měsíců [46]. Pokud je aplikace vyvíjena pro více platforem, je jasné, že se doba vývoje prodlouží kvůli neustálé nutnosti koordinace vývojových týmů s cílem přinést konzistentní UX[c] napříč platformami.

Zřejmě tedy nebudeme daleko od pravdy, když řekneme, že v obecném slova smyslu je vývoj hybridních aplikací méně časově náročný než vývoj nativní. To je způsobeno zejména možností využití jedné sady zdrojových kódů pro port aplikace na různé platformy.

4.1.3 Nástroje

4.1.3.1 Vývojové prostředí

Volba správného vývojového prostředí je pro pohodlný a rychlý vývoj mobilní aplikace klíčová. Vývojáři nativních aplikací v drtivé většině používají vývojová prostředí dodávaná s SDK konkrétní platformy. Pro iOS tedy vyvíjejí v XCode, pro Android v Eclipse, pro Windows Phone ve Visual Studiu a tak podobně. Tato prostředí jim poskytují možnost správy projektů, tvorby zdrojového kódu a také nezbytné ladící nástroje. Pomocí těchto prostředí je rovněž možné odeslat hotovou aplikaci na oficiální tržiště dané platformy.

Vývojáři hybridních aplikací mají ve volbě svého vývojového prostředí zdánlivě větší svobodu. Od uvedení nástroje PhoneGap Build (viz. kapitola 3.3.4.1) již nejsou nuceni pro vytvoření nativních instalačních balíčků používat SDK konkrétní platformy. V zásadě nic pak vývojáři nebrání používat pro vývoj libovolné vývojové prostředí dle vlastních preferencí (například oblíbený textový editor).

4.1.3.2 Testování a debugging

V tomto případě je situace podobná jako u vývojových prostředí. Pro nativní vývojáře jsou ladící a testovací nástroje dodávány jako součást SDK. Přímo z vývojového prostředí tedy spouští svou aplikaci v emulátoru a využívají debugovací nástroje vytvořené přímo za účelem ladění mobilních aplikací pro danou platformu.

Hybridní vývojáři mají situaci obtížnější. Mohou sice také využívat nativních nástrojů z SDK, ale pouze v omezené míře, neboť tyto nejsou vhodné pro ladění aplikací napsaných webovými jazyky. Hybridní vývojáři proto nejčastěji používají kombinaci nástrojů (nativní emulátor + webový prohlížeč + Ripple + Weinre) [viz kapitola 3.3.3]. Nativní emulátor mohou spouštět buď z vývojového prostředí určeného pro vývoj nativních aplikací

(například Eclipse) nebo pomocí konsolového interface, který poskytuje například framework PhoneGap.

4.1.3.3 Podpora

Každý vývojář se v průběhu vývoje mnohokrát dostane do situace, kdy neví jak dál a potřebuje pomoci. Tuto pomoc může získat z celé řady zdrojů jako jsou například knižní příručky, placená podpora, internetové články a internetová komunitní fóra.

V této oblasti mají nativní vývojáři zatím jasně navrch, což je dáno zejména tím, že hybridní přístup je poměrně nový. Počet knih i internetových příruček zabývajících se hybridním vývojem je oproti nativnímu signifikantně nižší. Kolem vývoje nativních aplikací pro konkrétní platformu je vytvořen celý prstenec nástrojů, které mohou vývojáři používat, když potřebují pomoci. Jedná se o IRC chaty, specializované diskuzní skupiny či možnost nechat si poradit od odborníků v rámci office hours (Android). Množství zodpovězených otázek[d] na populárním komunitním diskuzním fóru stackoverflow.com vyznívá také jasně ve prospěch nativní cesty.

Hybridní vývojář má (krom oficiální dokumentace) na výběr z nízkého (ale stále rostoucího) počtu různých internetových tutoriálů, které mu pomohou v začátcích. Na trhu je také k dispozici řada knih (v angličtině) většinově zaměřených na úvod do vývoje v konkrétním hybridním frameworku (dominantně PhoneGap). Každý framework má kolem sebe také vytvořenu určitou komunitu, kde může vývojář hledat pomoc (typicky diskuzní fóra). Framework PhoneGap poskytuje i možnost placené podpory pro vývojáře.

Lze tedy říci, že v případě hybridní cesty již existuje poměrně solidní množství materiálů pro začínající vývojáře, ale v případě komplikovanějších problémů je již vývojář odkázán na pomoc komunity kolem serverů, jako je například stackoverflow.com.

4.2 Vlastní aplikace

4.2.1 Vzhled

Ve světě mobilních aplikací není vzhled jen otázkou vkusu. Uživatelé chytrých mobilních telefonů jsou zvyklí, že aplikace běžící na jejich platformě vypadají a chovají se v určitých aspektech obdobně. Díky tomu se rychle naučí s aplikací pracovat a její používání je pro ně příjemné, protože aplikace reaguje tak, jak očekávají.

Hybridní aplikace jsou pohledem tohoto kritéria jednoznačně pozadu. Hlavním problémem je, že nevyužívají nativních UI elementů a nahrazují

je pomocí CSS a JavaScriptu. Kvalifikovaný vývojář hybridních aplikací dokáže pomocí těchto technologií napodobit vzhled nativních elementů velmi věrně, nikoliv však dokonale. Méně zkušený vývojář pak často sáhne po UI elementech, které mu nabízí javascriptový framework, jakým je například jQuery Mobile. Díky tomu dokáže vytvořit uživatelské prostředí velmi rychle, jeho podoba je však od té nativní již vzdálenější.

S tímto problémem se potýkáme ve všech oblastech UX hybridní aplikace. Nejedná se jen o vzhled samotných elementů, ale i o přechody mezi jednotlivými obrazovkami či odezvu na uživatelské vstupy. Pokud chceme, aby většina uživatelů nepoznala, že aplikace kterou používají, je hybridní, je zapotřebí najmout skutečně kvalifikovaného CSS vývojáře.

Oproti tomu nativní aplikace plně využívají nativních UI elementů a dosahují tak přirozeného vzhledu i očekávaných reakcí uživatelského prostředí.

4.2.2 Odezva

Odezva přímo souvisí s výše zmíněným kritériem vzhledu. Když uživatel „tapne“[e] na tlačítko, očekává okamžitou odpovídající reakci. Nativní UI elementy toto zaručují, u napodobenin jež využíváme v hybridních aplikacích, je již situace složitější. V hybridních aplikacích jsme odkázáni na události, které definuje norma jazyka JavaScript nebo námi zvolený javascriptový framework. Hlavním problémem je, že JavaScript ve výčtu definovaných událostí příliš nepočítá s dotykovými zařízeními a tak může být odezva aplikací, jejichž uživatelské prostředí na těchto událostech závisí pomalejší, než je tomu u jejich nativních protějšků.

Dobrou ilustrací tohoto problému je událost „click“. Tato událost vznikla proto, aby dala webovým aplikacím možnost dynamicky reagovat na kliknutí myši. Javascriptový engine rozpozná událost „click“ tak, že po stisknutí pravého tlačítka čeká 300 milisekund a kontroluje, zda je po této době tlačítko stále stisknuto. Pokud ano, nastane událost „click“. Při ovládání aplikací dotykem prstu však může chování této události způsobovat problémy. „Tapnutí“ prstem je typicky velmi krátké a pokud je aplikace připravena reagovat na událost „click“, nemusí „tapnutí“ zaznamenat a uživatel je tak nucen celou akci opakovat. Tohoto problému si byli vědomi tvůrci frameworků, jako je například jQuery Mobile, a vytvořili událost „tap“, která nemá nastaveno tak dlouhé čekání a tudíž lépe reaguje na ovládání prstem.

4.2.3 Výkon

Hledisko výkonu mobilních aplikací je zmiňováno prakticky v každém seriózněji článku zabývajícím se jejich vývojem pomocí různých přístupů. V naprosté většině z nich je slabší výkon zmiňován jako hlavní nevýhoda hybridních aplikací. Dle výzkumu společnosti Vision Mobile je právě problém s výkonem nejčastějším důvodem k zavržení hybridního přístupu [40].

Obecně se udává, že hybridní přístup není vhodný pro aplikace náročné na grafiku a výkon. Důvodem výkonnostních problémů u hybridních aplikací je nativní mezivrstva a zejména pak javascriptový engine. Ačkoliv tyto enginy neustále zrychlují, stále se nedokáží vyrovnat rychlosti nativních interpretů.

Nadále tedy platí, že pro hry a jiné výkonnostně náročnější aplikace je nativní cesta prakticky jedinou možnou, pokud chceme uživateli dopřát příjemný zážitek z používání aplikace.

4.2.4 Míra integrace s OS

Ačkoliv je schopnost využívat nativních komponent systému udávána jako jedna z hlavních předností hybridního přístupu, je logické, že oproti nativním budou hybridní aplikace vždy o krok pozadu. Nejpopulárnější hybridní framework PhoneGap podporuje v současnosti 16 nativních API volání, což je samozřejmě řádově méně než mají k dispozici vývojáři nativních aplikací.

Jedním z hlavních důvodů, proč je pro vývojáře hybridních aplikací dostupný menší počet nativních API volání, je sama multiplatformnost hybridních frameworků. Tím, že se tyto frameworky snaží podporovat co nejširší množství mobilních platforem, jsou tím pádem nuceni podporovat pouze nejmenšího společného dělitele nativních komponent všech platforem, aby použití frameworku bylo konzistentní.

4.3 Monetizace

Při výběru technologie, kterou vývojář při tvorbě aplikace použije, hraje roli i něco, čemu říkáme obchodní model kolem dané aplikace. Pokud chce vývojář na aplikaci vydělat peníze, má několik možností, jak toho docílit.

1. Aplikace bude placená a uživatel za ní zaplatí při stažení na oficiálním tržišti.
2. Reklamy uvnitř aplikace.
3. Nákupy uvnitř aplikace.

Zatímco u první možnosti monetizace není mezi hybridními a nativními aplikacemi rozdíl, u zbývajících dvou již zjistíme, že hybridní aplikace na tomto poli zaostávají. Přestože existují pluginy pro reklamní sítě jako je AdMob či iAds, které tuto funkcionalitu přináší (PhoneGap), jsou vyvíjeny sporadicky a uživatelé si často stěžují na problémy. Stejná situace je i v případě nákupů uvnitř aplikace, kde vývojáři reportují nekompatibilitu s novými verzemi PhoneGap i s novými verzemi operačních systémů. Pokud chce vývojář na svých aplikacích vydělávat, měl by si použití hybridního přístupu důkladně promyslet a ověřit si, že pro jím podporovanou platformu existuje ověřené fungující řešení.

4.4 Shrnutí

Výše nastíněné srovnání je samozřejmě nutno brát s určitou rezervou. Vždy primárně záleží na typu aplikace, na jejím účelu a cílové skupině, pro kterou je určena. Silné stránky nativního přístupu jsou výkon, monetizace nebo rozsah funkcí, které můžete ve své aplikaci nabídnout. Naopak na hybridní přístup se můžete spolehnout ve chvíli, kdy vám při vývoji aplikace primárně záleží na ceně nebo multiplatformním záběru.

V tabulce 4.2 jsem se pokusil shrnout zásadní srovnávací kritéria a každý přístup v rámci nich ohodnotit. Nativní přístup dosáhl lepšího hodnocení v 5 oblastech, hybridní přístup si připsal lepší skóre ve 3 oblastech.

Tabulka 4.2: Porovnání hybridního a nativního přístupu v klíčových oblastech [13]

Oblast	Hybridní přístup	Nativní přístup
Multiplatformní záběr	**5**	1
Bezpečnost	1	**5**
Distribuce aplikace	**5**	**5**
Monetizace	1	**5**
Aktualizace	**5**	1
Cena	**5**	1
UX	1	**5**
Výkon	1	**5**
Rozsah funkcí	1	**5**

Kapitola 5

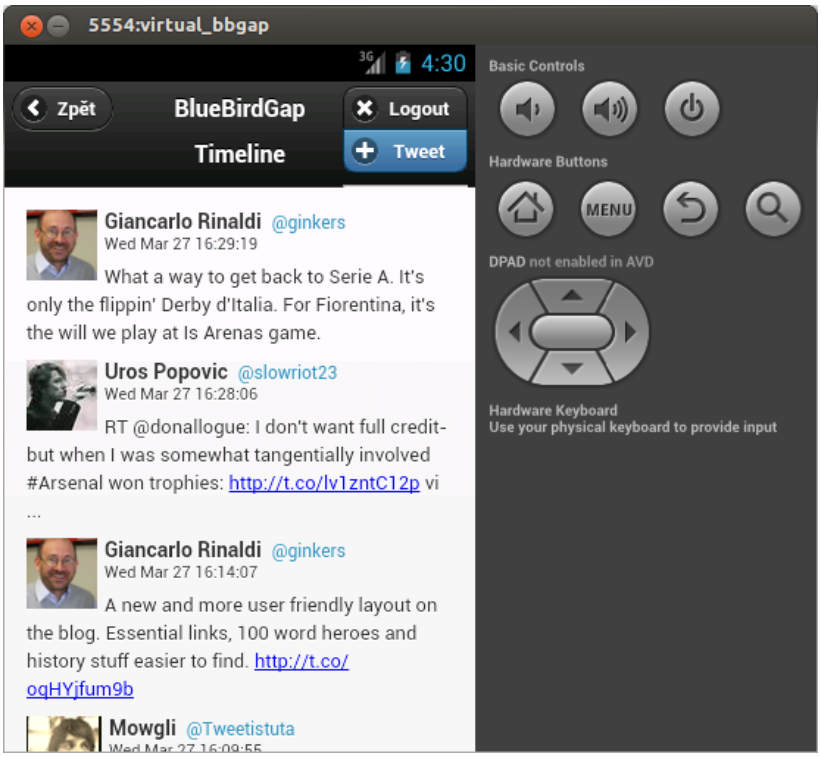
Demo aplikace

Součástí této práce jsou i tři ukázkové aplikace, které jsem vytvořil pomocí frameworku PhoneGap a slouží jako základní seznámení autora práce s frameworkem a také jako demonstrace použití základních API volání, které framework poskytuje.

5.1 Twitter klient (BlueBirdGap)

První aplikací, kterou jsem ve frameworku PhoneGap vytvořil, je jednoduchý klient pro sociální síť Twitter. Obsahuje pouze elementární funkce, kdy se může uživatel přihlásit svým uživatelským jménem a heslem, zobrazit svojí timeline a také odeslat na Twitter nový tweet. Aplikace byla původně zamýšlena jako multiplatformní, tedy tak, aby se dalo co možná největší množství kódu použít i na dalších platformách. Nakonec jí bylo z důvodu platformní závislosti pluginu ChildBrowser možné využít jen na platformě Android.

V průběhu vývoje jsem narazil na problémy zejména s implementací otevřeného protokolu OAuth, který je vyžadován k autentizaci uživatele na Twitteru. Implementace tohoto protokolu pomocí Javascriptu není přímočará záležitost, neboť tento postup skýtá bezpečnostní rizika v podobě odhalení přenášených bezpečnostních klíčů. Nakonec jsem využil knihovny jsOAuth, která nabízí abstraktní rozhraní pro využití tohoto protokolu. K tvorbě uživatelského rozhraní jsem využil framework jQuery Mobile spolu s vlastními CSS styly.



Obrázek 5.1: BlueBirdGap, zobrazení timeline

Pokud se zaměříme na unikátní vlastnosti frameworku PhoneGap, které byly v této aplikaci využity, musím na prvním místě uvést oficiální plugin ChildBrowser, který umožňuje v rámci WebView otevřít další WebView jako svého potomka. Této funkce bylo využito při identifikaci uživatele, kdy se mu otevře jako potomek další okno, kde musí ověřit své přihlašovací údaje na webové stránce Twitteru. Z dalších funkcí uvedených v API PhoneGap můžu zmínit například využití místní databáze localStorage pro uložení údajů o konkrétním uživateli.

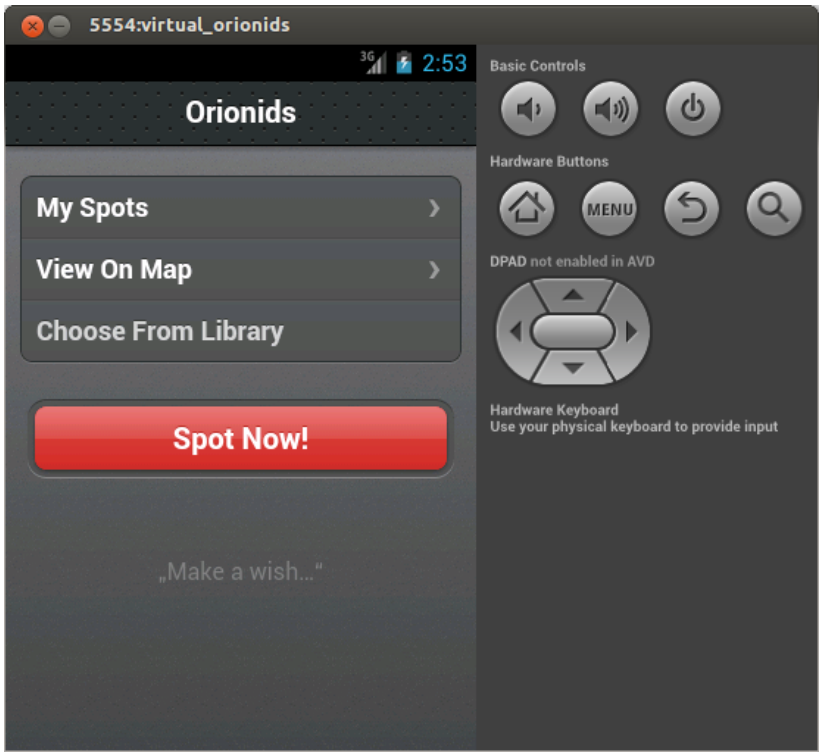
Aplikace byla otestována v Android emulátoru na verzi systému 4.2.

5.2 Foto aplikace (Orionids)

V druhé aplikaci jsem chtěl vyzkoušet patrně nejpřitažlivější funkci frameworku PhoneGap, za kterou pokládám možnost jednoduše ovládat fotoaparát daného mobilního zařízení. Vytvořil jsem za tímto účelem jednoduchou aplikaci, jejímž účelem je fotografovat a ukládat obrázky padajících hvězd. Zároveň s vyfocením fotografie je zaznamenána i geografická poloha místa, kde byla fotografie pořízena. Aplikace umožňuje pořídit fotografii

buď přímo či jí vybrat z nativní galerie. Všechny fotografie jsou v aplikaci uložené a je možno zobrazit jejich přehled a také si prohlédnout detail každé z nich.

Protože byl pro tvorbu uživatelského rozhraní použit framework jQTouch, který je závislý na jádře WebKit, funguje tato aplikace pouze na platformách Android a iOS. Pro uživatelské rozhraní jsem využil předpřipravených šablon frameworku jQTouch, které se automaticky přizpůsobí platformě, na které běží, aby více simulovaly vzhled nativních aplikací. Finální vzhled jsem upravil vlastními CSS styly. Pro zobrazení míst, kde byly fotografie pořízeny, je využito vložené Google mapy s vyznačením dotyčných poloh.



Obrázek 5.2: Orionids, domovská obrazovka

V rámci této aplikace bylo rovněž využito několik API volání, které vývojářům nabízí framework PhoneGap. Stěžejním bylo pochopitelně využití API volání Camera, které umožňuje pořídit fotografii fotoaparátem či ji vybrat z nativní fotogalerie daného zařízení. Pro ukládání informací o pořízených fotografiích jsem využil volání Storage, které zpřístupňuje ukládání do lokální databáze dle specifikace konsorcia W3C. V neposlední řadě jsem

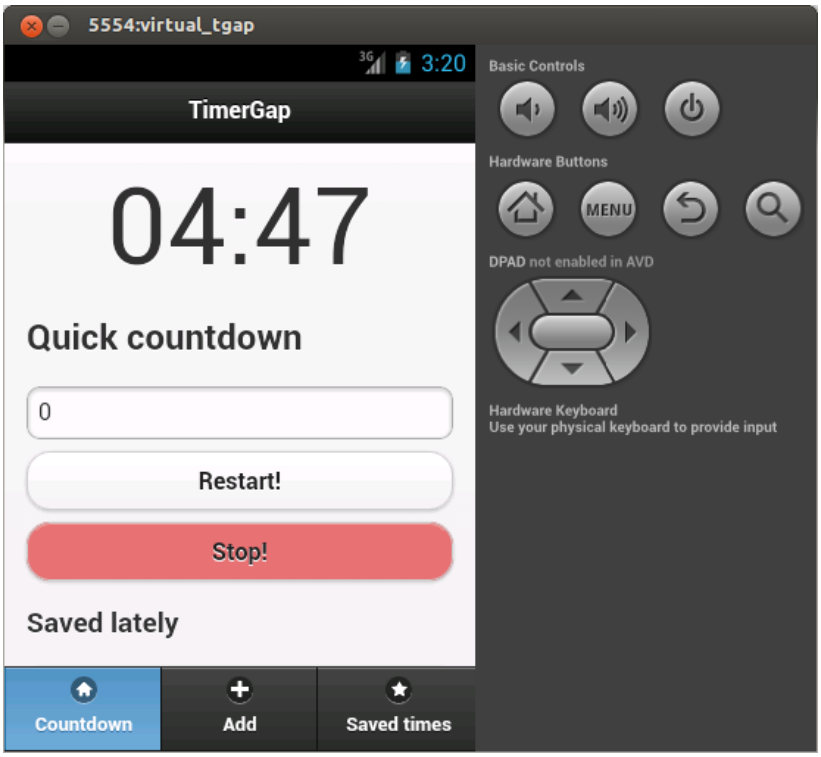
využil volání Geolocation, které zpřístupňuje údaje o zeměpisné šířce a délce místa, kde se dané zařízení právě nachází. Na závěr musím podotknout, že používání všech API bylo velmi přímočaré a bezproblémové.

Aplikace byla otestována v Android emulátoru na verzi systému 4.2.

5.3 Časovací aplikace (TimerGap)

Poslední aplikací je jednoduchý časovač, který umožňuje uživateli spustit odpočítávání času využitelné například jako kuchyňská minutka pro odměření doby potřebné pro vylouhování čaje. Aplikace umožňuje spustit rychlé odpočítávání, uložit a pojmenovat odpočítávání pro další použití a samozřejmě vybírat a spouštět již uložené časy.

V této aplikaci jsem opět využil javascriptový framework jQuery Mobile, pomocí nějž jsem vytvořil uživatelské rozhraní a také využil jeho funkcí při tvorbě funkční logiky celé aplikace. Pro realizaci vlastního odpočítávání jsem využil knihovnu jQuery Countdown, která poskytuje pro jeho implementaci velmi podařené abstraktní rozhraní. Pro zajištění uživatelsky přívětivého způsobu vybírání času jsem využil plugin jQuery Mobile - Datebox.



Obrázek 5.3: TimerGap, proces odpočítávání

V této aplikaci jsem se zaměřil především na otestování notifikačních schopností frameworku PhoneGap, který nabízí několik možností jak upozornit uživatele na nějakou událost. Z rozhraní Notification jsem využil téměř všechny nabízené možnosti: dialogy alert či confirm a v momentě vypršení odpočítávaného času také zvukové upozornění beep spolu se zavibrováním vibrate. Pro ukládání uživatelem definovaných odpočítávání jsem využil databázi localStorage.

Práce s výše zmíněnými API voláními byla poměrně bezproblémová, krom platformy Windows Phone 7, kde jsme narazil na problémy s databází localStorage, která se nad jádrem prohlížeče IE 9 chová poněkud nestandardně.

Aplikace byla otestována v Android emulátoru na verzi systému 4.2 a na mobilním telefonu Nokia Lumia 800 se systémem Windows Phone 7.8.

Závěr

V rámci této práce jsem se snažil analyzovat problematiku multiplatformního vývoje mobilních aplikací ze všech možných úhlů. Na závěr se rozhodně nechci pouštět do vynášení obecných soudů, spíše bych rád shrnul poznatky, které jsem během studia dané problematiky načerpal a také se pokusil odhadnout, jaká budoucnost hybridní frameworky čeká.

Pro hybridní přístup je na trhu bezpochyby prostor, který se bude dle mého názoru nadále zvětšovat. Již v dnešní době máme na mobilním trhu tři nebo čtyři silné hráče, kteří jsou pro tvůrce aplikací zajímaví. V dalších letech se bude tento „long tail“ nadále zvětšovat zejména díky příchodu dalších menších hráčů. Mezi nimi si můžeme uvést například Firefox OS, jehož cílem je obsadit trh s low-endovými chytrými telefony na rozvojových trzích, dále Ubuntu Phone, který přenáší prostředí nejpopulárnější Linuxové distribuce na mobilní zařízení. Sledovat se vyplatí i projekt Tizen či znovuzrození webOS.

Existuje tedy důvod domnívat se, že v přístích letech tu bude krom iOS, Androidu, Windows Phone a BlackBerry další silná skupina platforem s menším podílem, která však dohromady bude zabírat pro vývojáře zajímavou část trhu. Tento bobtnající „long tail“ tedy může být jedním z důvodů, proč bude hybridní přístup pro vývojáře stále atraktivnější. Je zajímavé si všimnout, že tito menší hráči jako Firefox OS nebo Ubuntu Phone přímo počítají s webovými/hybridními aplikacemi jakožto plnohodnotným přístupem k vývoji aplikací.

Na druhou misku vah je však třeba položit aktuální nezralost hybridních frameworků, která brání masovému přijetí tohoto přístupu mezi vývojáři. Hlavním problémem je především výkon. I na poměrně jednoduchých aplikacích lze poznat, že byly vyvinuty hybridní cestou. Především absence nativních UI elementů a nedostatečná rychlost javascriptových enginů jsou

hlavními důvody, proč hybridní aplikace výkonově zaostávají. Musím na tomto místě opět připomenout výzkum společnosti Vision Mobile, kde se 29 % vývojářů vyjádřilo, že právě problém s výkonem byl hlavním důvodem k opuštění hybridní cesty. Pro překonání tohoto problému je třeba mít hluboké znalosti CSS a javascriptových frameworků, což ovšem může výrazně zvýšit náklady na tvorbu aplikace.

Bude zajímavé sledovat, jak se s narůstajícím podílem hybridních aplikací vyrovnají tvůrci současných lídrů na trhu mobilních platforem. Jejich zájmem totiž rozhodně není, aby se aplikace tvořily jednotně pro všechny platformy. Pokud vývojář napíše aplikaci exkluzivně pro jednu platformu, dává jí to zajímavou výhodu před konkurencí. Dobrým příkladem je například populární Instagram, který měl po dlouhou dobu aplikaci pouze pro systém iOS.

Tvůrci těchto platforem přitom mají nad hybridními aplikacemi velkou moc. Nemusí jim povolit přístup na svá oficiální tržiště či jim mohou komplikovat život pomocí úprav webového runtime, který je pro většinu hybridních frameworků životně důležitý. Prvním náznakem takového omezování je přístup společnosti Apple, která poskytuje vývojářům webový runtime s pomalejším javascriptovým enginem, než jaký sama využívá ve svém vlastním mobilním prohlížeči. Pravděpodobně tedy můžeme očekávat další podobné překážky, které budou hybridním frameworkům kladeny do cesty, v rámci snahy o zachování důležité konkurenční výhody.

Velký potenciál pro hybridní přístup vidím v segmentu firemních aplikací. Takzvaný BYOD (Bring Your Own Device) přístup začíná nabývat na síle i u nás a otvírá tím hybridním aplikacím dveře dokořán. U firemních aplikací, které jsou využívány primárně zaměstnanci k přístupu k datům z informačních systémů, totiž příliš nezáleží na vzhledu ani na špičkovém výkonu mobilní aplikace. Důležitější je multiplatformní pokrytí a nízké náklady, tedy kritéria, v nichž hybridní přístup vyniká. Není náhodou, že Demokratická strana ve Spojených státech využívala hybridní aplikace ke koordinaci dobrovolníků pracujících na kampani Baracka Obamy. Důvod je zřejmý. Každý dobrovolník si mohl takovou aplikaci nainstalovat na svůj chytrý telefon nehledě na to, jaký operační systém používá. A stranu to přišlo na minimální náklady.

Budoucnost hybridních frameworků tedy skýtá mnoho příležitostí, které mohou posunout tento přístup ještě více do centra zájmu vývojářů mobilních aplikací. K tomu, aby se hybridní aplikace mohly směle poměřovat s těmi nativními, musí však urazit ještě velmi dlouhou cestu. V příštích letech se jako uživatelé budeme jistě nadále setkávat především s nativními aplikacemi. To platí zejména pro herní tituly a další výkonově náročnější aplikace. Dynamický vývoj HTML5, nárust „long tailu“ na trhu s mobilními

platformami a rozšiřování BYOD přístupu ve firmách však dává hybridnímu přístupu příležitost vychytat mnohé nedostatky a stát se v budoucnu plnohodnotnou cestou, kterou budou vývojáři vyvíjet mobilní aplikace.

Literatura

[1] Appcelerator Titanium vs. Phonegap: Which is the better Mobile Development Platform? | Mobile Zone. Dostupné z: <http://mobile.dzone.com/news/appcelerator-titanium-vs>

[2] Architecture Diagrams · TideSDK/TideSDK Wiki. Dostupné z: <https://github.com/TideSDK/TideSDK/wiki/Architecture-Diagrams>

[3] BII REPORT: How Hybrid Apps Are Accelerating HTML5 Adoption - Business Insider. Dostupné z: <http://www.businessinsider.com/bii-report-how-hybrid-apps-are-accelerating-html5-adoption-2012-11>

[4] Building Mobile Applications with Titanium - Documentation & Guides - Appcelerator Wiki. Dostupné z: <https://wiki.appcelerator.org/display/guides/Building+Mobile+Applications+with+Titanium>

[5] Dissecting Phonegap's architecture - Agiliq Blog | Django web app development. Dostupné z: <http://agiliq.com/blog/2012/09/dissecting-phonegaps-architecture/>

[6] Gartner Says by 2016, More Than 50 Percent of Mobile Apps Deployed Will be Hybrid. Dostupné z: <http://www.gartner.com/newsroom/id/2324917>

[7] InfoWorld's 2012 Technology of the Year Award winners - InfoWorld. Dostupné z: <http://www.infoworld.com/slideshow/24605/infoworlds-2012-technology-of-the-year-award-winners-183313#slide2>

[8] iPhone 5 and iOS 6 for HTML5 developers, a big step forward: web inspector, new APIs and more | Breaking the Mobile Web. Dostupné z: <http://www.mobilexweb.com/blog/iphone-5-ios-6-html5-developers>

[9] JavaScript API for native widgets on iOS, Android | MoSync Guide/Tutorial. Dostupné z: <http://www.mosync.com/documentation/manualpages/jsnativeui-library>

[10] Job Search | one search. all jobs. Indeed.com. http://www.indeed.com/. Dostupné z: <http://www.indeed.com/>

[11] Key Features | Appcelerator Cloud Services. Dostupné z: <http://www.appcelerator.com/cloud/key-features/>

[12] Mobile Applications Development, iPhone & Android App Development | Marmalade. Dostupné z: <http://www.madewithmarmalade.com/marmaladesdk/supported-platforms>

[13] Mobile Development: Web App vs. Native App PART 2 - Hybrid Approach. http://www.itexico.com/blog/bid/92633/Mobile-Development-Web-App-vs-Native-App-PART-2-Hybrid-Approach. Dostupné z: <http://www.itexico.com/blog/bid/92633/Mobile-Development-Web-App-vs-Native-App-PART-2-Hybrid-Approach>

[14] Open letter to Apple iPhone Developer Support | iOS/Web Developer's Life in Beta. Dostupné z: <http://nachbaur.com/blog/open-letter-to-apple-iphone-developer-support>

[15] PhoneGap | Adobe Dreamweaver 5.5 Supports PhoneGap. Dostupné z: <http://phonegap.com/2011/04/12/adobe-dreamweaver-5-5-supports-phonegap/>

[16] PhoneGap | Adobe PhoneGap 2.0 Released. Dostupné z: <http://phonegap.com/2012/07/20/adobe-phonegap-2-0-released.md/>

[17] PhoneGap | Adobe PhoneGap named Best Cross-Platform Development Tool by CodeProject. Dostupné z: <http://phonegap.com/2012/06/12/adobe-phonegap-named-best-cross-platform-development-tool-by-codeproject/>

[18] PhoneGap | Adobe PhoneGap named Best Mobile Development Tool by Dr. Dobb's Jolt Awards. Dostupné z: <http://phonegap.com/blog/2012/11/01/adobe-phonegap-wins-jolt-award/>

[19] PhoneGap | Adobe PhoneGap named "Champion" by Info-Tech Research Group. Dostupné z: <http://phonegap.com/2012/06/28/adobe-phonegap-named-%E2%80%9Cchampion%E2%80%9D-by-info-tech-research-group/>

[20] PhoneGap | Announcing the PhoneGap Build Beta. Dostupné z: <http://phonegap.com/2010/11/09/announcing-the-phonegap-build-beta/>

[21] PhoneGap | BBC Olympics. Dostupné z: <http://phonegap.com/app/bbc-olympics/>

[22] PhoneGap | Day 1: Nitobi joins Adobe. Dostupné z: <http://phonegap.com/2011/10/27/day-1-nitobi-joins-adobe/>

[23] PhoneGap | Fruit Salad. Dostupné z: <http://phonegap.com/app/fruit-salad/>

[24] PhoneGap | Nitobi enters into Acquisition Agreement with Adobe. Dostupné z: <http://phonegap.com/2011/10/03/nitobi-enters-into-acquisition-agreement-with-adobe/>

[25] PhoneGap | PhoneGap 1.0 Released Today at PhoneGap Day in Portland. Dostupné z: <http://phonegap.com/2011/07/29/phonegap-1-0-released-today-at-phonegap-day-in-portland/>

[26] PhoneGap | PhoneGap 1.3 Released. Dostupné z: <http://phonegap.com/2011/12/19/phonegap-1-3-released/>

[27] PhoneGap | PhoneGap and Worklight Enable Mobile Apps for the Enterprise. Dostupné z: <http://phonegap.com/2011/07/13/phonegap-and-worklight-enable-mobile-apps-for-the-enterprise/>

[28] PhoneGap | PhoneGap Build is Launched! Dostupné z: <http://phonegap.com/blog/2012/09/24/phonegap-build-is-launched/>

[29] PhoneGap | PhoneGap Build moves to Open Beta. Dostupné z: <http://phonegap.com/2011/09/30/phonegap-build-moves-to-open-beta/>

[30] PhoneGap | PhoneGap, Cordova, and what's in a name? Dostupné z: <http://phonegap.com/2012/03/19/phonegap-cordova-and-what%E2%80%99s-in-a-name/>

[31] PhoneGap | PhoneGApp Store Approval. Dostupné z: <http://phonegap.com/2009/11/20/phonegapp-store-approval/>

[32] PhoneGap | PhoneGap's Winning Pitch at Web 2.0 Expo. Dostupné z: <http://stage.phonegap.com/2009/05/01/phonegap%E2%80%99s-winning-pitch-at-web-2-0-expo/>

[33] PhoneGap | Questions and Myths About PhoneGap. Dostupné z: <http://phonegap.com/2012/06/21/questions-and-myths-about-phonegap/>

[34] PhoneGap | Rolling Releases: How Apache Cordova becomes PhoneGap and Why. Dostupné z: <http://phonegap.com/2012/04/12/rolling-releases-how-apache-cordova-becomes-phonegap-and-why/>

[35] PhoneGap | Unofficial announcement of PhoneGap. Dostupné z: <http://phonegap.com/2008/08/07/unofficial-announcement-of-phonegap/>

[36] PhoneGap officially permitted on the App Store | iOS/Web Developer's Life in Beta. Dostupné z: <http://nachbaur.com/blog/phonegap-officially-permitted-on-the-app-store>

[37] A Primer on Hybrid Apps for iOS - Cocoa Controls. Dostupné z: <https://www.cocoacontrols.com/posts/a-primer-on-hybrid-apps-for-ios>

[38] Rise of the "Hybrid"Mobile App | MIT Technology Review. Dostupné z: <http://www.technologyreview.com/news/424328/rise-of-the-hybrid-mobile-app/>

[39] RoadmapProjects - Cordova Wiki. Dostupné z: <http://wiki.apache.org/cordova/RoadmapProjects>

[40] [Survey] Which are the best cross-platform tools? | VisionMobile. Dostupné z: <http://www.visionmobile.com/blog/2011/11/new-survey-which-are-the-best-cross-platform-tools/>

[41] Thoughts on Flash. Dostupné z: <http://www.apple.com/hotnews/thoughts-on-flash/>

[42] Updates on Apple / PhoneGap | iOS/Web Developer's Life in Beta. Dostupné z: <http://nachbaur.com/blog/updates-on-apple-phonegap>

[43] Why HTML5 is the new Java | TechCentral. Dostupné z: <http://www.techcentral.co.za/why-html5-is-the-new-java/24835/>

[44] Globant: *Hybrid Mobile Development: A Cost-Effective Strategy for Building Cross-Platform Mobile Apps.*

[45] IBM: *IBM Worklight V5.0.6 Technology overview.*

[46] Michaels, ross & cole, ltd.: *Native mobile apps: The wrong choice for business?*

Příloha **A**

Seznam použitých zkratek

GUI Graphical user interface

XML Extensible markup language

Příloha **B**

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `exe` ..... adresář se spustitelnou formou implementace
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF
  - `thesis.ps` ..... text práce ve formátu PS