Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Dizertační práce

Metody vývoje aplikací s adaptivním systémem zobrazení na mobilních platformách

Methods of application development with adaptive displaying system for mobile platforms

Autor: **Ing. Radek Vala**

Studijní program: Inženýrská informatika
Studijní obor: Inženýrská informatika

Školitel: doc. Mgr. Roman Jašek, Ph.D.

Zlín, srpen 2015

OBSAH

OBSAH **2**
ABSTRAKT **5**
ABSTRACT **6**
PODĚKOVÁNÍ **7**
SEZNAM OBRÁZKŮ **8**
SEZNAM TABULEK **10**
SEZNAM ZKRATEK **11**
1 **ÚVOD** **12**
2 **SOUČASNÝ STAV** **14**
2.1 POVĚDOMÍ O TECHNOLOGII A VÝVOJ VEŘEJNÉHO MÍNĚNÍ 15
2.2 INTERNETOVÉ PORTÁLY A DISKUZNÍ FÓRA 16
2.3 KOMERČNÍ STUDIE 17
2.4 VĚDECKO-VÝZKUMNÉ PUBLIKACE 17
3 **CÍLE DIZERTAČNÍ PRÁCE** **19**
3.1 HYPOTÉZY 21
4 **TEORETICKÝ RÁMEC** **22**
4.1 PŘEDSTAVENÍ MOBILNÍCH OPERAČNÍCH SYSTÉMŮ 22
4.2 SPECIFIKA MOBILNÍCH OPERAČNÍCH SYSTÉMŮ Z HLEDISKA VÝVOJE APLIKACÍ 23
4.2.1 Omezené výpočetní možnosti *23*
4.2.2 Životní cyklus aplikace v rámci operačního systému *23*
4.2.3 Sandboxing, oprávnění aplikace, sdílení souborů *25*
4.2.4 Dotykové ovládání *26*
4.2.5 Omezená zobrazovací plocha a variabilita zobrazovacích zařízení *26*
4.2.6 Uživatelský zážitek *28*
4.2.7 Konektivita *29*
4.3 MOBILNÍ OPERAČNÍ SYSTÉMY A JEJICH ZASTOUPENÍ NA TRHU 29
4.4 KLASIFIKACE METOD VÝVOJE MOBILNÍCH APLIKACÍ 33
4.4.1 Vývoj nativních mobilních aplikací *34*
4.4.2 Vývoj mobilních webových aplikací *36*
4.4.3 Vývoj CP mobilních aplikací a klasifikace dílčích metod *36*
4.4.4 Shrnutí *39*
4.5 WEBOVÉ HYBRIDNÍ MOBILNÍ APLIKACE 40
4.5.1 Architektura webových hybridních aplikací *41*

4.5.2 Procento společného zdrojového kódu pro různé platformy ... *42*
4.5.3 Wrapper technologie ... *42*
4.5.4 Princip nativních pluginů ... *42*
4.5.5 Příklady dostupných nativních pluginů ... *45*
5 **EXPERIMENTÁLNÍ ČÁST** ... **46**
5.1 CHARAKTERISTIKA A SROVNÁNÍ ZÁKLADNÍCH METOD VÝVOJE MOBILNÍCH APLIKACÍ ... 47
5.1.1 Znalostní požadavky na vývojový tým ... *47*
5.1.2 Požadavky na vybavení vývojového týmu ... *48*
5.1.3 Jednoduchost vývoje ... *49*
5.1.4 Rychlost vývoje ... *49*
5.1.5 Návrh grafického uživatelského rozhraní ... *50*
5.1.6 Distribuční kanály ... *50*
5.1.7 Aktualizace aplikací ... *51*
5.2 KLÍČOVÉ ASPEKTY PROCESU VÝVOJE WHMA ... 52
5.2.1 WHMA z hlediska funkčních a nefunkčních požadavků ... *52*
5.2.2 WHMA z hlediska znalostí vývojového týmu ... *54*
5.2.3 WHMA z hlediska časových a finančních nákladů ... *56*
5.3 CHARAKTERISTIKA A SROVNÁNÍ ZÁKLADNÍCH METOD TVORBY UŽIVATELSKÉHO ROZHRANÍ V RÁMCI WHMA ... 59
5.3.1 Tvorba vlastního HTML5 GUI ... *60*
5.3.2 GUI pomocí HTML5 UI frameworku ... *61*
5.3.3 GUI tvořené kombinací HTML5 a nativních UI objektů ... *62*
5.3.4 Dílčí závěr – vlastnosti a srovnání metod ... *64*
5.4 NÁVRH A REALIZACE MULTIKRITERIÁLNÍ ANALÝZY VLASTNOSTÍ UI FRAMEWORKŮ ... 64
5.4.1 Rozhodovací kritéria ... *65*
5.4.2 Multikriteriální matice ... *67*
5.4.3 Výsledky vážené multikriteriální analýzy ... *69*
5.4.4 Přínosy realizované multikriteriální analýzy ... *71*
5.5 NÁVRH A REALIZACE VÝKONNOSTNÍCH TESTŮ UI FRAMEWORKŮ ... 71
5.5.1 Popis testovací aplikace ... *72*
5.5.2 Použitý hardware ... *73*
5.5.3 Metody výkonnostních testů ... *73*
5.5.4 Výsledky výkonnostních testů ... *75*
5.5.5 Přínosy realizovaných výkonnostních testů ... *78*
5.6 IDENTIFIKACE PROBLÉMŮ BĚHOVÉHO PROSTŘEDÍ APACHE CORDOVA/PHONEGAP ... 79

5.6.1 Charakteristika standardního běhového prostředí Apache Cordova/Phonegap. 79
5.6.2 Nekonzistence vlastností běhového prostředí Apache Cordova/Phonegap 79
5.6.3 Příčiny nekonzistence běhových prostředí platformy Android 81
5.6.4 Vykreslovací jádra webových technologií v rámci platformy Android..... 84
5.6.5 Návrh použití společného vykreslovacího jádra 86
5.6.6 Identifikace zobrazovacích chyb způsobených nekonzistentní podporou HTML5 standardu 87
5.7 ŘEŠENÍ PROBLÉMU NEKONZISTENCE BĚHOVÉHO PROSTŘEDÍ 89
5.7.1 Nadstandardní běhová prostředí pro Apache Cordova/Phonegap 90
5.7.2 Měření podpory HTML5 vlastností ve vybraných běhových prostředích..... 91
5.7.3 Zhodnocení výsledků měření podpory HTML5 vlastností 93
5.7.4 Měření výkonu JavaScriptu v rámci běhových prostředí..... 95
5.7.5 Zhodnocení výsledků měření výkonu JavaScriptu 96
5.7.6 Měření grafického výkonu běhových prostředí..... 100
5.7.7 Zhodnocení výsledků měření grafického výkonu 100
5.7.8 Doporučení běhového prostředí 104
5.8 APLIKACE NAVRŽENÝCH POSTUPŮ V REÁLNÝCH PROJEKTECH 106
5.9 VYHODNOCENÍ NAVRŽENÝCH POSTUPŮ POUŽITÝCH V REÁLNÉM VÝVOJOVÉM PROJEKTU APLIKACE UTB 109
5.9.1 Dotazníkový průzkum..... 109
5.9.2 Vyhodnocení úspěšnosti aplikace dle statistik Google Play 119
ZÁVĚR 122
6 **LITERATURA A OSTATNÍ ZDROJE..... 125**
7 **PUBLIKACE..... 134**
7.1 ČLÁNEK VE SBORNÍKU 134
7.2 OSTATNÍ PUBLIKACE..... 135
7.3 SOFTWARE 136
7.4 VEDENÉ DIPLOMOVÉ PRÁCE..... 136
7.5 VĚDECKOVÝZKUMNÉ AKTIVITY..... 137
ODBORNÝ ŽIVOTOPIS AUTORA 138

ABSTRAKT

Vývoj mobilních aplikací je v současné době jednou z nejaktivnějších oblastí softvérového inženýrství. Kromě standardních nativních vývojových metod založených na softvérových nástrojích dodávaných přímo výrobcem operačního systému a umožňujících sestavení aplikace právě pro konkrétní platformu, existují i nestandardní vývojové metody označované jako hybridní, které jsou velmi zajímavé z hlediska multiplatformní tvorby mobilních aplikací a právě tématem hybridních metod se tato práce zabývá. Detailněji se pak práce zaměřuje především na webové hybridní metody vývoje, jelikož vykazují nejvyšší procento zdrojového kódu společného pro více platforem, což je vlastnost v současné době vysoce žádaná, zejména z hlediska konkurenceschopnosti vývojářských společností. Cílem experimentální části je vytvoření inovativní, reprodukovatelné, časově a finančně vyvážené metody vývoje webových hybridních mobilních aplikací. Jsou zde definovány nejpalčivější problémy vývoje a doporučeny způsoby řešení vyplývající z měření a aplikačních testů. Výstupy práce jsou aplikovatelné na reálné projekty a umožňují dosáhnout stability mobilní aplikace na různých platformách a předejít zobrazovacím problémům vedoucím k navýšení finančních nákladů vynaložených při fázi testování a ladění.

Klíčová slova: mobilní aplikace, HTML5, hybridní mobilní aplikace, běhová prostředí, Apache Cordova, Phonegap, uživatelský zážitek

ABSTRACT

Mobile application development is currently one of the most active software engineering areas. Besides standard native development methods that are based on software development tools supplied by the manufacturer of the operating system, there are also non-standard development methods, so-called hybrid methods, which are very interesting in terms of cross-platform mobile applications development. Hybrid methods are currently very often discussed and connected with competitiveness of developer companies. This doctoral thesis addresses specifically the web-based hybrid mobile applications whose main advantage is the high percentage of source code, which is common to multiple platforms. The aim of the experimental part is to create innovative, reproducible, and time-consuming and financially balanced methods of web-based hybrid application development. There are defined serious development issues and on the basis of measurements and application tests, real solutions are recommended.

Keywords: mobile applications, HTML5, hybrid mobile applications, runtime, Apache Cordova, Phonegap, user experience

PODĚKOVÁNÍ

Tímto bych rád poděkoval doc. Mgr. Romanu Jaškovi Ph.D. za podporu, vstřícný přístup, lidské jednání, poskytnutí podnětných rad a odborné vedení během mého studia.

Děkuji také svým rodičům za to, že mi ukázali co je to slušnost a pokora, ale také sebevědomí a cílevědomost, tedy vlastnosti, kterým vděčím za šťastný průchod životem.

SEZNAM OBRÁZKŮ

Obrázek 1: Životní cyklus aplikace v prostředí operačního systému iOS [40] *24*
Obrázek 2: Příklad specifikace minimální dotykové oblasti společností Apple Inc. [6] *27*
Obrázek 3: Statistika rozlišení obrazovek mobilů a tabletů v ČR z období duben 2014 až duben 2015 [50] *27*
Obrázek 4: Vývoj tržních podílů mobilních operačních systémů prosinec 2008 – květen 2014 [3] [52] *30*
Obrázek 5: Zastoupení jednotlivých verzí systému Android na trhu v letech 2012-2014 [19] *31*
Obrázek 6: Aktuální graf zastoupení jednotlivých verzí Android OS [54] *31*
Obrázek 7: Zastoupení jednotlivých verzí systému iOS na trhu k červnu 2014 (vlevo) a květnu 2015 (vpravo) [55] *32*
Obrázek 8: Zastoupení jednotlivých verzí Windows Phone na trhu (březen 2014) [56] *33*
Obrázek 9: Model architektury mobilních aplikací – Nativní, Hybridní a Webové mobilní aplikace dle [66] *37*
Obrázek 10: Model „cross-platformní" architektury Xamarin aplikací [84] *39*
Obrázek 11: Klasifikace mobilních aplikací *40*
Obrázek 12: Dělení druhů mobilních aplikací dle návrhu UI *40*
Obrázek 13: Model architektury webové hybridní mobilní aplikace (Apache Cordova či Phonegap) *41*
Obrázek 14: Proces vývoje mobilní aplikace včetně klíčových parametrů výběru vývojové metody *52*
Obrázek 15: Srovnání velkosti zdrojového kódu reálné aplikace portálu property-cross.com *58*
Obrázek 16: Ukázka obrazovek simulátoru konvektomatu Retigo v rámci aplikace Retigo Vision 1.2 ..... *61*
Obrázek 17: Ukázka hlavní a detailní obrazovky grafického rozložení „master-detail" *62*
Obrázek 18: Vykreslení nativního TimePicker objektu (výběr času) pomocí pluginu DatePicker pro Apache Cordova/Phonegap na platformách iOS 8.1, Android 2.3.7 a Android 4.2.2 ..... *63*
Obrázek 19: Ukázka objektu Action Sheet (vlevo realizace pomocí HTML5, vpravo nativní objekt) *63*
Obrázek 20: Výsledky multikriteriální srovnání UI frameworků *70*
Obrázek 21: Pohled na hlavní obrazovku se seznamem (Chocolate Chip UI Framework) *72*
Obrázek 22: Pohled na obrazovku s detailními informacemi (Chocolate Chip UI Framework) *73*
Obrázek 23: Srovnání rychlosti DOM Load *76*
Obrázek 24: Srovnání dílčích aktivit načítání *76*
Obrázek 25: FPS skrolování listu o 100 položkách *77*
Obrázek 26: FPS animace přechodu stránek *78*
Obrázek 27: Výstup funkce diff (definice konstant) – porovnání balíčků com.google.android / android / 4.4_r1 / android.webkit.WebViewFactory verzí Andoid OS 4.3.1_r1 a 4.4_r1 ..... *83*
Obrázek 28: Výstup funkce diff (metoda getProvider) – porovnání balíčků com.google.android / android / 4.4_r1 / android.webkit.WebViewFactory verzí Andoid OS 4.3.1_r1 a 4.4_r1 ..... *83*
Obrázek 29: Stav webových běhových prostředí na platformě Android verze 4.0–4.4 ..... *85*
Obrázek 30: Stav webových běhových prostředí na platformě Android verze 4.4 ..... *86*
Obrázek 31: Stav webových běhových prostředí na platformě Android verze 5 ..... *86*
Obrázek 32: Návrh použití společného vykreslovacího jádra pro platformu Android *87*
Obrázek 33: Ukázka chybného pozicování ikony informace – nahoře správně v běhovém prostředí Crosswalk, dole chybně ve standardním WebView *88*

Obrázek 34: Ukázka chybného pozicování tlačítka přihlásit se – nahoře správně v běhovém prostředí Crosswalk, dole chybně ve standardním WebView *89*
Obrázek 35: Ukázka chybného pozicování horních tlačítek a výběrových lišt – nahoře správně v běhovém prostředí Crosswalk, dole chybně ve standardním WebView *89*
Obrázek 36: Přehled bodování desktopových prohlížečů v letech 2009 až 2015 dle serveru html5test.com *92*
Obrázek 37: Přehled bodování mobilních prohlížečů v letech 2009 až 2015 dle serveru html5test.com *92*
Obrázek 38: Srovnání výkonnosti JavaScriptu – Samsung Galaxy Note 10.1 (OS Android 4.1.2) *98*
Obrázek 39: Srovnání výkonnosti JavaScriptu – Motorola Nexus 6 (OS Android 5.1.1) *99*
Obrázek 40: Srovnání grafického výkonu běhových prostředí – Samsung Galaxy Note 10.1 (OS Android 4.1.2) *102*
Obrázek 41: Srovnání grafického výkonu běhových prostředí – Motorola Nexus 6 (OS Android 5.1.1) *104*
Obrázek 42: Srovnání přidané velikosti běhových prostředí v rámci instalované aplikace *106*
Obrázek 43: Ukázka UI aplikace Retigo Vision – vlevo hlavní menu konvektomatu, vpravo podmenu Easy cooking. *107*
Obrázek 44: Ukázka UI mobilní aplikace UTB na zařízení s Android OS – vlevo nástěnka po přihlášení, vpravo výpis zkouškových termínů *108*
Obrázek 45: Procentuální zastoupení pohlaví mezi respondenty *110*
Obrázek 46: Věkové zastoupení respondentů *110*
Obrázek 47: Zaměření absolvovaného studia respondentů *111*
Obrázek 48: Zájem respondentů o trendy v oblasti mobilních technologií *111*
Obrázek 49: Sebehodnocení vztahu respondentů k mobilním technologiím *112*
Obrázek 50: Srovnání distribuce subverzí OS Android mezi respondenty dotazníkového průzkumu, statistikami aplikace UTB na Google Play, statistikami v kategorii vzdělávání na Google Play a celkovými statistikami platformy Android *113*
Obrázek 51: Distribuce subverzí OS Android < 4.4. a >= 4.4 dle kategorií viz výše *114*
Obrázek 52: Hodnocení velikosti aplikace UTB *115*
Obrázek 53: Hodnocení rychlosti startu aplikace UTB *116*
Obrázek 54: Hodnocení navigace v aplikaci UTB *117*
Obrázek 55: Hodnocení plynulosti animací v aplikaci UTB *118*
Obrázek 56: Hodnocení celkového chování aplikace UTB *119*
Obrázek 57: Stav selhání a chyb ANR aplikace UTB k 25. 8. 2015 dle play.google.com *120*
Obrázek 58: Statistika aktuálního počtu instalací aplikace UTB dle play.google.com (k 28. 8. 2015) *121*
Obrázek 59: Uživatelská hodnocení aplikace UTB na Google Play k 28. 8. 2015 ..... *121*

SEZNAM TABULEK

Tabulka 1: *Aktuální podíly na trhu mobilních operačních systémů [3]* ... 29
Tabulka 2: *Distribuce jednotlivých verzí OS Android ke květnu 2015 [54]* ... 32
Tabulka 3: *Znalostní požadavky vývojového týmu* ... 47
Tabulka 4: *Srovnání požadavků na vybavení vývojového týmu* ... 48
Tabulka 5: *Srovnání jednoduchosti vývoje* ... 49
Tabulka 6: *Srovnání rychlosti vývoje* ... 50
Tabulka 7: *Srovnání technologií, používaných pro tvorbu GUI* ... 50
Tabulka 8: *51*
Tabulka 9: *Srovnání aktualizací* ... 51
Tabulka 10: *Vhodnost WHMA z hlediska funkčních požadavků* ... 53
Tabulka 11: *Vhodnost WHMA z hlediska nefunkčních požadavků* ... 54
Tabulka 12: *Měření LOC na projektu testovacích aplikací portálu property-cross.com* ... 58
Tabulka 13: *Srovnání znalostí nutných pro použití jednotlivých metod implementace UI* ... 64
Tabulka 14: *Srovnání vhodnosti metod implementace UI pro aplikace se standardním a specifickým UI* ... 64
Tabulka 15: *Jednotlivé verze frameworků použitých pro multikriteriální analýzu* ... 65
Tabulka 16: *Multikriteriální matice pro evaluaci vlastností UI frameworků* ... 67
Tabulka 17: *Normalizovaná multikriteriální matice pro evaluaci vlastností UI frameworků* ... 68
Tabulka 18: *Expertní hodnocení kritérií* ... 68
Tabulka 19: *Výsledné váhy multikriteriální matice získané na základě expertního hodnocení* ... 69
Tabulka 20: *Výsledná multikriteriální matice hodnocení UI frameworků* ... 69
Tabulka 21: *Přehled rozdílných běhových prostředí WHMA v rámci majoritních platforem* ... 81
Tabulka 22: *Srovnání podpory vlastností HTML5 na různých verzích mobilních platforem a za použití různých běhových prostředí* ... 94
Tabulka 23: *Měření podpory HTML5 vlastností na reálných zařízeních* ... 95
Tabulka 24: *Výsledky měření testu Octane 2.0 – Samsung Galaxy Note 10.1 (OS Android 4.1.2)* ... 97
Tabulka 25: *Výsledky měření testu Octane 2.0 – Motorola Nexus 6 (OS Android 5.1.1)* ... 99
Tabulka 26: *Výsledky měření grafického výkonu běhových prostředí – Samsung Galaxy Note 10.1 (OS Android 4.1.2)* ... 102
Tabulka 27: *Výsledky měření grafického výkonu běhových prostředí – Motorola Nexus 6 (OS Android 5.1.1)* ... 103
Tabulka 28: *Srovnání velikostí balíčků a instalovaných aplikací s různými běhovými prostředími, včetně srovnání velikostí samotných běhových prostředí (velikost v MB)* ... 106

SEZNAM ZKRATEK

CP	„cross-platformní” (Cross-platform)
DOM	objektový model dokumentu (Document Object Model)
GUI	grafické uživatelské rozhraní (Graphic user interface)
HMA	hybridní mobilní aplikace (Hybrid mobile application)
OS	operační system (Operating system)
SDK	balík vývojových nástrojů (Software development kit)
UI	uživatelské rozhraní (User interface)
UX	uživatelský zážitek (User experience)
WHMA	webová hybridní mobilní aplikace (web-based HMA)

1 ÚVOD

Mobilní telefony jsou běžnou součástí lidského života již řadu let. Od jejich prvního uvedení na trh prošly překotným vývojem a v posledních letech začínají promlouvat do oblastí informačních technologií, které byly dříve výhradně doménou počítačů. S příchodem tzv. chytrých telefonů (Smartphones) se možnost využití těchto zařízení podstatně rozrostla a zahrnuje původně čistě počítačové úkony, jako je například prohlížení Internetu, e-mailů, tvorba a sdílení dokumentů, komunikace na sociálních sítích apod. Dle společnosti eMarketer převýší počet chytrých mobilních telefonů už do konce roku 2018 počet klasických mobilní zařízení. V celosvětovém měřítku to znamená více jak dva a půl miliardy uživatelů chytrých mobilních zařízení. [1]

V roce 2010 zvolila American Dialect Society slovem roku slůvko „app“ (Application Software) [2], což odráží právě rapidní rozvoj nové oblasti vývoje software – vývoje mobilních aplikací. Představením operačního systému iPhone OS 2.0 společnosti Apple spolu s představením on-line obchodu s aplikacemi App Store dostali vývojáři jedinečnou možnost tvořit mobilní aplikace pro nebývale vysoký počet uživatelů. Myšlenka globálního distribučního kanálu aplikací pro konkrétní mobilní operační systém byla natolik zajímavá, že i společnosti Google a Microsoft později představily vlastní on-line obchody s aplikacemi.

Je logické, že řady vývojářských týmů se pokusily využít této jedinečné příležitosti a vrhly se na tvorbu aplikací pro jednu či více mobilních platforem. Tato oblast softwarového inženýrství však čelí mnoha technologickým výzvám. Jde zejména o různorodost prostředí jednotlivých operačních systémů a dále nepřehledné množství variant mobilních zařízení s rozdílnými výkonovými i zobrazovacími parametry. Pro mobilní trh tak v současné době v podstatě nelze vyvinout univerzální aplikaci.

Nejsilnějšími platformami na trhu jsou mobilní operační systémy Android, iOS a Windows Phone. [3] Pokud chce vývojářský tým vytvořit aplikaci pro co největší množství zákazníků, má možnost vyvíjet odděleně nativní aplikace pro každou platformu zvlášť, nebo využít některý z přístupů, jenž umožňuje vývoj pro více platforem současně (tzv. hybridní přístup). Nativní vývoj pomocí SDK (Software Development Kit) konkrétní platformy je z hlediska standardizace, výkonnosti a stability ideální možností, ovšem nutnost separátního vývoje podstatně zvyšuje časové a finanční náklady a také nároky na erudovanost vývojářského týmu, který nesmí postrádat specialisty na nativní vývoj pro tři a někdy i více koncových platforem.

V současnosti se proto stává velmi zajímavou oblastí vývoj tzv. hybridních mobilních aplikací (HMA), což je metoda, která umožňuje vyvíjet společný zdrojový kód pro více mobilních operačních systémů pomocí jednotných technických nástrojů. [4] Hlavním benefitem je pak časová a finanční úspora, protože není nutné mít zvláštní vývojový projekt či tým pro každou

mobilní platformu. Hybridní aplikace mají však své nevýhody a celou řadu specifik, které je nutno brát v potaz a které jsou výzvou k dalšímu zkoumání, dokumentaci a vylepšení metod hybridního vývoje.

Dizertační práce se tedy zabývá velmi aktuálním tématem vývoje webových hybridních mobilních aplikací, které nabízejí nejvyšší míru společného zdrojového kódu. Konkrétně se zde jedná o hybridní aplikace tvořené pomocí technologií HTML5 [5]. Vzhledem k tomu, že je v současné době na trhu velké množství různých mobilních zařízení a verzí operačních systémů, stává se příprava stabilní mobilní aplikace technologickou výzvou. Je to dáno i tím, že vývoj webových hybridních mobilních aplikací je založen na zcela nových a nestandardizovaných přístupech.

Motivací pro řešení této problematiky je tedy v prvé řadě nedostatečné povědomí o technologiích, principech a celkovém procesu vývoje webových hybridních mobilních aplikací (WHMA) mezi vývojáři mobilních aplikací, což vede k tomu, že názory publikované v komerčních studiích, ale také vědecko-výzkumných pracích, se mnohdy diametrálně liší. Dalším důvodem je pak velká fragmentace technologií, což při neexistenci standardních vývojových postupů či uceleného vývojového ekosystému činí orientaci v podpůrných vývojových softvérových rámcích velmi složitou. Bariéra adopce metody vývoje webových hybridních aplikací je pak překvapivě velká, přestože je založena na velmi rozšířených webových technologiích a kvalita výsledných produktů je mnohdy nízká, jelikož pouhá znalost postupů a metod ze světa webdesignu se v reálném procesu vývoje ukazuje jako nedostačující.

Hlavním cílem práce je proto tvorba reprodukovatelné vývojové metody s důrazem na možnost sestavení výsledné aplikace pro více platforem. Dílčí části tvořící ucelenou metodu se zaměřují především na technologické postupy pro vývoj stabilních a spolehlivých webových hybridních mobilních aplikací, které budou vykazovat konstantní výsledky i v prostředí různých subverzí mobilních platforem.

2 SOUČASNÝ STAV

Mobilní aplikace se staly nedílnou součástí každodenního života dnešní moderní společnosti. Poptávka na trhu udává velmi dynamické tempo vývoje mobilních platforem i aplikací pro ně určených. Programovací jazyky používané v rámci jednotlivých mobilních operačních systémů sice již existují mnohdy i desítky let, ale stále vznikají nové vývojářské nástroje na nich založené a celkový proces vývoje a sestavení finální aplikace se mění s příchodem aktuálních verzí operačních systémů.

Trh s mobilními operačními systémy prokázal v posledních letech značnou dynamiku (viz kapitola 4.3) a lze jen těžko předpovědět, jak úspěšné budou konkrétní mobilní platformy v horizontu několika let. Z výše uvedených důvodů se samotný vývoj mobilních aplikací stal velmi rychle se měnící oblastí obsahující množství přístupů a technologií a je velmi obtížné nalézt jejich ideální kombinaci, která umožní docílit rychlého zisku.

Jestliže má mobilní aplikace v současné době pokrývat majoritní platformy, znamená to, že musí být vytvořeny minimálně verze pro Android, iOS a Windows Phone. Nativní vývoj pak sebou nese nutnost tvorby tří naprosto odlišných projektů pro každou z výše uvedených platforem. To samozřejmě klade vysoké nároky na „know-how“ vývojového týmu (znalost jednotlivých nativních vývojových nástrojů a programovacích jazyků) a roste také časová náročnost kompletního vývoje. V tržním prostředí se manažeři vývojových týmů snaží tyto faktory zvyšující náklady na projekt minimalizovat a hledají k nativnímu vývoji alternativní metody. Zde hovoříme o tzv. hybridním vývoji, jehož označení vzešlo z architektury aplikace, která používá jak nativní kód, tak i abstraktní vrstvu mezi různými platformami sdílenou a zvyšuje tak podíl společného zdrojového kódu. Tato práce se zabývá především metodou tvorby webových hybridních mobilních aplikací (WHMA), které vykazují nejvyšší míru společného kódu, který je tvořen moderními webovými technologiemi. Vzhledem k divergenci technologií a postupů výrobců mobilních operačních systémů se tato metoda jeví jako výhodná z hlediska úspory času i nákladů právě díky sdílenému zdrojovému kódu. Je však nutné vzít v potaz ostatní faktory ovlivňující vývojový proces a vyhodnotit vhodnost metody pro konkrétní aplikaci.

V současné době se problematika hledání vhodných vývojových metod či softwarových rámců, potažmo vývoj nových technologií a postupů, stává diskutovaným tématem jak v prostředí komerční sféry, tak aplikovaného výzkumu. Tato oblast totiž skýtá celou řadu technologických výzev, od volby konkrétní metody pro vývoj mobilní aplikace, přes hodnocení kvality uživatelského zážitku, výkonu a optimalizace aplikací, až po techniky transformace uživatelského prostředí pro koncové zařízení.

Na rozdíl od hybridního vývoje je oblast vývoje nativních aplikací poměrně dobře zdokumentována a standardizována (více v kapitole 4.4.1). Je to dáno tím, že pro všechny

majoritní mobilní operační systémy existují v současné době balíky vývojových nástrojů (tzv. SDK – Software Development Kit) vydávané přímo výrobcem daného operačního systému. Kromě standardní dokumentace knihoven jsou k dispozici také šablony projektů, stejně jako nejlepší doporučení zabývající se návrhem uživatelského rozhraní. Na oblast uživatelského zážitku a samotného vzhledu aplikace se vždy nejvíce soustředila společnost Apple Inc., která již dlouhou dobu vydává aktualizovaný dokument iOS Human Interface Guidelines [6], který slouží vývojářům jako oficiální doporučení a dokumentace pro tvorbu uživatelského rozhraní. Podobná doporučení jsou však k dispozici i v rámci platformy Android [7] či Windows Phone [8].

Informační zdroje z oblasti hybridních technologií jsou však v současné době poměrně roztříštěné a nekompletní, což je dáno rychlými technologickými změnami a množstvím softvérových rámců a vývojových postupů, které lze v rámci hybridního přístupu uplatnit. Navíc společnosti jako je Apple Inc., Google Inc. či Microsoft oficiálně vydávají své vlastní SDK a nijak se tedy na podporu hybridních metod nezaměřují. Proto v této oblasti chybí vyspělé a komplexní vývojové nástroje a zdokumentované vývojové postupy či nejlepší doporučení. Přesto však hybridní metody patří k velmi diskutovaným a jak komerční sféra, tak oblast vědy a výzkumu se zabývá jejich možným potenciálem.

2.1 Povědomí o technologii a vývoj veřejného mínění

Webové hybridní mobilní aplikace zažily za dobu své poměrně krátké existence období velké popularity, její vrchol a následný pád. Zejména v období let 2011–2012 lze nalézt na Internetu celou řadu článků, které vyzdvihují výhody vývoje pro více platforem najednou a možnost využití webových technologií, avšak opomíjejí reálné nedostatky. Například v [9] jsou hybridní aplikace představeny jako technologie, umožňující snadnější vývoj mobilních aplikací pro více platforem (tehdy ještě mimo Windows Phone) a je zde vyzdvižena zejména možnost publikace pomocí oficiálních distribučních kanálů. Nikde však není zmínka o reálných problémech tohoto přístupu. Na některé nedostatky však již poukazuje [10] a dokonce zmiňuje i možné problémy v oblasti dosažení dostatečně kvalitního uživatelského zážitku – pouze však v kontextu rozdílného designu a UX cílových platforem, nikoliv například v souvislosti s reálnými zobrazovacími problémy, které se na různých subverzích OS a zařízeních projevují a proto je jim také věnována pozornost v experimentální části dizertační práce.

Na přelomu roku 2012 a 2013 přešly společnosti Facebook a LinkedIn od hybridních k nativním metodám vývoje u svých mobilních aplikací, přičemž tento krok velmi ovlivnil veřejné mínění a oblíbenost hybridních metod v IT světě podstatně klesla.

S příchodem OS Android verze 4.4 disponuje nativní internetový prohlížeč (WebView) vykreslovacím jádrem Chromium, což je z hlediska webových hybridních aplikací velký krok kupředu v podobě lepší podpory vlastností HTML5. Od verze Android 5.0 je dokonce poprvé nativní WebView umístěn v separátní aplikaci, kterou je možné aktualizovat. [11] Platforma iOS ve verzi 7 zase představila k původnímu UIWebView novou třídu WKWebView z WebKit Frameworku, která obsahuje moderní vykreslovací jádro Webkit s podporou vlastností jazyka HTML5 [12]. Tento poměrně nedávný vývoj (přelom roku 2013 a 2014) hovoří ve prospěch webových hybridních metod a může pomoci napravit veřejné mínění o jejich vhodnosti pro vývoj mobilních aplikací. Výše uvedené kvitují také vývojáři hybridních frameworků, jako je například Ross Gerbasi ze společnosti Sencha, který se v [13] zabývá právě nárůstem výkonu a podpory HTML5 u nativního prohlížeče na platformě iOS a publikuje srovnání výkonu WKWebView s původním UIWebView. Společnost Telerik (jeden z oficiálních developerů nativních pluginů pro Adobe Phonegap) dokonce vydala plugin [14] pro využití WKWebView v rámci Phonegap či Cordova projektů ještě dříve, než v květnu 2015 vyšla oficiální podpora v rámci verze cordova-ios 4.0.0 [15].

HTML5 se tak postupně stává technologií, která je konzistentně podporovaná na současných mobilních platformách a zatímco svět nativního vývoje podléhá stále větší divergenci vývojových nástrojů, běhová prostředí webových mobilních aplikací se naopak více sjednocují.

Jedním z největších rizik v rámci diskutované metody je však nedostatek vyspělých vývojových nástrojů, nejlepších doporučení, standardních postupů a informačních zdrojů. To však vytváří velký prostor pro výzkum a analýzu jednotlivých vývojových nástrojů a postupů, srovnání technologických přístupů a ustanovení nejlepších doporučení. Jedině pokud budou mít vývojáři k dispozici kompletní informace o výhodách, nevýhodách a specifikách jednotlivých přístupů, budou moci udělat kvalifikované rozhodnutí v otázce výběru metody vývoje pro konkrétní projekt.

2.2 Internetové portály a diskuzní fóra

Ze zajímavých projektů dostupných na Internetu, lze uvést například webový portál PropertyCross [16], který přináší souhrn softvérových rámců vhodných pro hybridní vývoj. Jeho tvůrci vytvořili stručné představení jednotlivých vývojových frameworků, pomocí nichž je pak implementována ukázková aplikace. Tu je možno stáhnout ve formátu pro konkrétní platformu, nainstalovat na zařízení a otestovat. Pro každý z takřka dvaceti frameworků zde autoři uvádějí orientační poměr procenta sdíleného kódu tak, jak bylo naměřeno v rámci vývoje ukázkových aplikací. Portál Mobile Frameworks Comparison Chart [17] pak umožňuje vyhledávání dle parametrů v databázi takřka padesáti vývojových frameworků. Část je však již neaktuální či

s neaktivní komunitou. Uvedené portály tedy nabízí alespoň základní přehled dostupných technologií, z nichž je ale třeba dále pečlivě vybírat.

Jako primární zdroj informací často slouží programátorům hybridních aplikací zejména anglicky psané on-line zdroje či komunitní fóra. Například fórum StackOverflow je dle [18] považováno i díky systému hodnocení kvality odpovědí za relevantní zdroj, který oproti vědecko-výzkumným publikacím nabízí možnost velmi rychle získat kvalifikovanou odpověď na velmi aktuální otázku. Svým zaměřením na praktické programování a řešení problémů systémem otázka – odpověď ovšem poskytuje pouze velmi konkrétní, negeneralizované a mnohdy i subjektivní informace.

2.3 Komerční studie

Slovní spojení multiplatformní (v anglické literatuře se lze setkat s termínem „cross-platform“) označuje v současné době v oblasti mobilních technologií velmi aktuální téma objevující se na internetových stránkách IT portálů, e-magazínů a serverů. Nejčastěji zahrnuje metody vývoje webových hybridních mobilních aplikací a komerční studie cílené zejména na CEO (Chief Executive Officer) v IT sféře obvykle přinášejí jejich představení a základní srovnání.

Rozsáhlé komerční studie zveřejňuje například server Research2guideance.com. Jedná se o informační zdroje založené na plošných dotazníkových výzkumech či rozhovorech s výkonnými pracovníky předních světových softwarových společností. Například v rámci interview v [19] odpovídá Martin Willson (CEO společnosti Appear) na to, jaké jsou největší překážky při nasazení hybridního vývoje. V rámci serveru Research2guideance jsou také publikovány výsledky dotazníkového průzkumu [20], který se zabývá otázkou, zda se vyplatí přechod z nativního na hybridní přístup, jehož se účastnilo 40 společností a 2188 vývojářů. Další hodnotné informace přináší výzkum [21] zaměřený na vývoj samotných nástrojů pro tvorbu webových hybridních aplikací, jehož výsledky jasně hovoří o tom, že se tyto produkty neustále rozvíjí, zlepšují a reagují na požadavky trhu.

Další společností, která poskytuje celou řadu analýz je Gartner, Inc. Ve studii [22] například předpovídá, že v roce 2016 bude více než 50 procent mobilních aplikací na trhu vytvořeno pomocí některého z hybridních přístupů.

2.4 Vědecko-výzkumné publikace

Vědecko-výzkumných publikací zaměřených na konkrétní mobilní platformy či jejich obecné srovnání, lze nalézt celou řadu – např. [23] [24] [25] [26].

Z oblasti hybridních mobilních aplikací je vědeckých publikací již méně. V [27] například nalezneme zevrubné srovnání nativního a hybridního vývoje a v [28] pak i výkonové srovnání

hybridních mobilních aplikací tvořených pomocí nativního „wrapperu“ PhoneGap (s čistě webovými technologiemi či s využitím UI frameworků) s frameworkem Titanium [29]. Sledovanými proměnnými jsou zde paměť využitá aplikací a dále procesorový čas, což jsou však parametry, které jen nepřímo souvisí s výkonem aplikace, plynulostí a uživatelským zážitkem. Jiné čtyři hybridní vývojářské nástroje, jež zahrnují i ty, které nejsou postavené čistě na webových technologiích, jsou srovnány v [30]. Článek se však vůbec nesoustřeďuje na výkonové aspekty a předkládá spíše základní informace o vlastnostech a architektuře jednotlivých frameworků. Další srovnání z hlediska výkonu (rychlost spuštění aplikace a množství paměti potřebné k běhu) předkládá [31]. Zároveň uvádí klasifikaci hybridních frameworků, která však nebere v úvahu některé současné moderní technologie a z toho důvodu je v rámci této práce navržena klasifikace vlastní. Autoři publikace [32] definují sadu kritérií, jejichž relevance byla v rámci této práce hodnocena za účelem návrhu vážené multikriteriální analýzy HTML5 UI frameworků. V rámci publikace [32] však bylo provedeno pouze srovnání na vyšší úrovni (nativní, hybridní, webová metoda).

Samostatnou kapitolou jsou pak publikace typu „případová studie“ (např. [33] [34] [35]), které popisují konkrétní vývoj mobilní aplikace pomocí některé z hybridních vývojových metod. Články zpravidla řeší velmi specifiké problémy, které se týkají stanovení funkčních či nefunkčních požadavků aplikace potažmo samotné implementace, nicméně nesnaží se o vyšší míru zobecnění a aplikaci poznatků na širší oblast vývojového procesu.

Z výše uvedeného vyplývá, že v současné době chybí ucelený výstup aplikovaného výzkumu zabývající se metodou webového hybridního vývoje mobilních aplikací, který by se zaměřoval na klíčové části vývojového procesu a navrhoval postupy vedoucí k efektivní tvorbě multiplatformních mobilních aplikací, jež budou vykazovat stabilní běh na různých subverzích operačních systémů.

3 CÍLE DIZERTAČNÍ PRÁCE

Hlavním cílem dizertační práce je navrhnout v rámci aplikovaného výzkumu inovativní, technologicky vyspělou, časově a finančně vyváženou metodu vývoje webových hybridních mobilních aplikací (WHMA). Jde tedy o doporučení reprodukovatelných technologických postupů pro vývoj mobilních aplikací s možností sestavení pro více platforem s důrazem na stabilitu a spolehlivost běhu a to i na starších zařízeních objevujících se na trhu. Zohledněny by měly být všechny základní fáze tvorby softvérového produktu definované normou ISO/IEC 90003 (analýza požadavků, návrh, implementace, integrace, testování, nasazení a ukončení).

Práce poukazuje mimo jiné na reálné problémy, se kterými se lze v rámci vývoje a zejména testování WHMA setkat, a diskutuje jejich minimalizaci či řešení.

K naplnění hlavního cíle je nutno uskutečnit následující výzkum, jenž bude prezentován v rámci **teoretické části** práce:

- **Kritická rešerše současného stavu, aktuálních komerčních a vědecko-výzkumných publikací pojednávajících o diskutovaném tématu** (kapitola 2).
- **Identifikace specifik mobilních operačních systémů z hlediska vývoje mobilních aplikací** (závěry získané na základě empirických znalostí z procesu implementace mobilních aplikací – kapitola 4.2).
- **Analýza a návrh klasifikace současných metod vývoje mobilních aplikací** (klasifikace vytvořená na základě rešerše aktuálních vědecko-výzkumných publikací, komparativních studií a vlastní analýzy architektury mobilních aplikací – kapitola 4.4).

V rámci **experimentální části** práce jsou pak řešeny následující dílčí kapitoly:

- **Charakteristika a srovnání základních metod vývoje mobilních aplikací** (závěry uskutečněné na základě rešerše vědecko-výzkumných publikací, kvalitativních studií, komparativních studií a empirických znalostí – kapitola 4.4)
- **Klíčové aspekty procesu vývoje WHMA** (definice a analýza klíčových aspektů z hlediska funkčních a nefukčních požadavků, znalostí vývojového týmu a časové či finanční náročnosti, vytvořená na základě empirických zkušeností z vývojových projektů a provedeného měření – kapitola 5.2)
- **Charakteristika a srovnání základních metod tvorby grafického uživatelského rozhraní v kontextu WHMA** (srovnání a závěry získané na základě aplikovaného výzkumu (včetně implementace reálných aplikací), ale i studia oficiální dokumentace jednotlivých technologií – kapitola 5.2)

- **Multikriteriální analýza vlastností UI frameworků** (návrh vážené multikriteriální analýzy pro výběr vhodného UI frameworku v rámci konkrétního vývojového týmu – kapitola 5.4)
- **Výkonnostní testování UI frameworků** (stanovení metod měření, identifikace vhodných parametrů a závěry měření testovacích aplikací implementujících vybrané UI frameworky – kapitola 5.5)
- **Implementace běhového prostředí v rámci WHMA** (závěry vyvozené ze statistických průzkumů vlastností běhových prostředí a aplikovaného výzkumu zahrnujícího implementační testy a vlastní měření – kapitola 5.4)
- **Verifikace vhodnosti navržených postupů použitých v rámci reálného vývojového projektu** (vyhodnocení dotazníkového průzkumu uživatelů reálné mobilní aplikace, zaměřené na potvrzení či vyvrácení stanovených hypotéz týkajících se zejména uživatelského zážitku)

Cílem výše uvedených dílčích kapitol a výsledné metody je generalizovat a systematizovat jinak velmi specifický a nejednotný přístup k vývoji webových hybridních mobilních aplikací. Mimo to si práce klade za úkol zodpovědět i následující otázky:

- Jaké jsou důležité parametry výběru vývojových nástrojů v oblasti WHMA?
- Jaké jsou největší problémy vývojového procesu (nestabilita, uživatelský zážitek, fragmentace nástrojů...)?
- Lze kritická místa vývojového procesu efektivně řešit?
- Jaké jsou dopady fragmentace mobilních operačních systémů a zařízení pro oblast WHMA?
- Je technologie HTML5 připravena pro vývoj moderních mobilních aplikací?
- Je možné vytvořit pomocí webového hybridního přístupu aplikaci, která je v prostředí nativních aplikací konkurenceschopná?
- Lze použitím metody webového hybridního vývoje ušetřit náklady?
- Jaká je budoucnost hybridních vývojových metod?

Důvodem k řešení popisované problematiky je fakt, že jednou z největších překážek nasazení webového hybridního vývoje je neexistence uceleného vývojového rámce (doporučení a charakteristika konkrétních technologií, vyspělé vývojové nástroje), což potvrzuje v [19] i CEO společnosti Appear Martin Wilson a upozorňuje na velké množství proprietárních API. Toto množství softwarových nástrojů a přístupů tvoří u vývojářských týmů prvotní bariéru a proces kvalifikovaného výběru vývojové technologie činí nesmírně složitým.

Teoreticko-poznávacím cílem je posun poznání v oblasti vývoje WHMA a jejich implementace na mobilních zařízeních. Zejména pak zkoumání technologií vhodných k adaptaci uživatelského rozhraní na cílovou platformu takovým způsobem, aby míra uživatelského zážitku byla co nejvyšší. Je třeba zodpovědět zejména otázku nekonzistence výkonu JavaScriptu, grafického výkonu a podpory HTML5 standardů v běhových prostředích a poskytnout možnosti řešení.

Praktickým výstupem práce jsou reálné webové hybridní mobilní aplikace UTB a Retigo Vision, jejichž vývoj probíhal dle navržené metodiky s využitím získaných znalostí a doporučených postupů.

Očekávaným **přínosem pro praxi** je významné zefektivnění procesů vývoje mobilních aplikací a to i ve formě on-line distribuovaných informačních systémů z datových center.

3.1 Hypotézy

Očekáváme, že dizertační práce naplní hypotézy, které budou v jejím závěru buď potvrzeny, nebo vyvráceny.

„Lze sestavit sadu kritérií a hodnotící metodiku univerzálně použitelnou pro výběr vývojového UI frameworku se zaměřením na konkrétní aplikaci.“

„Moderní HTML5 UI frameworky lze díky volbě vhodného běhového prostředí použít i na starších zařízeních, nepodporujících nové HTML5 standardy.“

„V rámci webových hybridních mobilních aplikací lze použít technologie HTML5 k vytvoření adaptivního uživatelského rozhraní, jež bude imitovat nativní uživatelské rozhraní platformy, pro kterou byla aplikace sestavena.“

„Pomocí vhodného výběru technologií pro vývoj hybridní mobilní aplikace lze vytvořit dostatečně výkonnou aplikaci nabízející kvalitní uživatelský zážitek skrze různé platformy a zařízení.“

4 TEORETICKÝ RÁMEC

Následující kapitola stručně vymezuje důležité technologie a principy, které jsou využívány v oblasti vývoje mobilních aplikací. V kontextu práce jde především o samotné představení mobilních operačních systémů a jejich specifik v oblasti vývoje aplikací, dále pak o přehled jejich současných tržních podílů a zejména o přehled, definice a klasifikace jednotlivých metod vývoje mobilních aplikací. Detailněji jsou v rámci této kapitoly popsány samotné webové hybridní mobilní aplikace, jakožto hlavní téma této práce.

4.1 Představení mobilních operačních systémů

Mobilní operační systémy jsou speciálně navrženy pro chytré telefony a tablety, neboli přenosná (mobilní) zařízení. Poskytují podobné služby jako operační systémy desktopové, jako například řízení zdrojů procesoru, správu paměti, spouštění programů či aplikací, řízení vstupně výstupních operací a přípravu a zobrazení uživatelského rozhraní. Mobilní operační systémy se od desktopových liší především v oblasti komplexnosti. Vzhledem k tomu, že mobilní zařízení disponují omezenými výpočetními či paměťovými zdroji a celkově jsou navržena k jednodušším úkonům než v případě desktopů, jsou i mobilní operační systémy méně komplexní, jednodušší, datově menší a úspornější. [36] Současné mobilní operační systémy jsou navrženy odlišně od desktopových zejména v oblasti správy paměti a přidělování prostředků. Hlavním cílem mobilního OS je zajistit za každých okolností běh kritických aplikací, jako je příjem hovorů, ale také níže položených aplikací jádra, jako například správa napájení. Z toho důvodu je zde přidělování paměti striktně hlídáno systémem. Například operační systém iOS nepodporuje vzhledem k omezeným hardwarovým prostředkům zařízení, na rozdíl od desktopového OS X, tzv. „backing store“. Jedná se o princip, kdy jsou data, která se zrovna nepoužívají, odstraněna systémem z operační paměti a uložena na disk. [37] V mobilním prostředí iOS jsou data pouze pro čtení, která jsou už uložena na disku, kdykoliv z operační paměti odstraněna a v případě potřeby znovu načtena. Data pro zápis však z paměti operačním systémem odstraňována nejsou. Pokud ale hladina volné paměti klesne pod určitou hranici, systém požádá běžící aplikace o uvolnění paměti (událost didRecievedMemoryWarning) [38]. Aplikace, které neuvolní dostatečné množství paměti jsou pak systémem nekompromisně ukončeny.

Aktuálně nejpoužívanějšími mobilními operačními systémy jsou: Android, iOS a Windows Phone (více v kapitole 4.3).

4.2 Specifika mobilních operačních systémů z hlediska vývoje aplikací

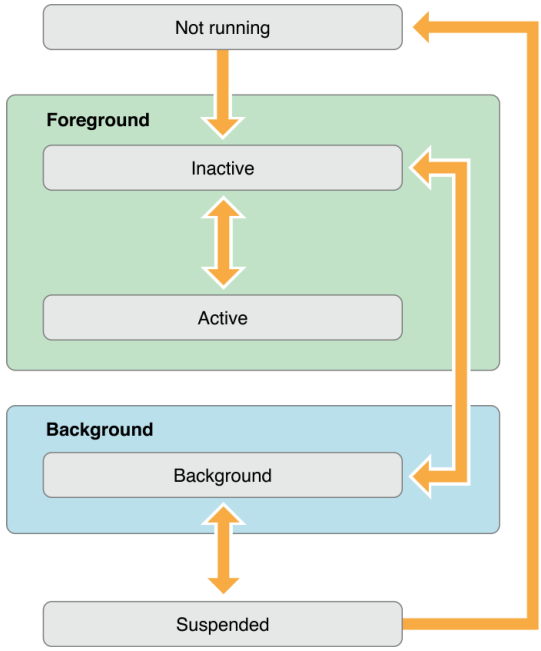
Z hlediska metod vývoje mobilních aplikací je důležité zvážit zejména následující specifika, jimiž se současné mobilní operační systémy liší od desktopových a které je nutno brát v potaz při procesu návrhu mobilní aplikace.

4.2.1 Omezené výpočetní možnosti

Při návrhu mobilních aplikací je třeba brát v úvahu zejména hardwarová omezení, která jsou dána výkonem koncového zařízení, na nichž bude mobilní aplikace provozována. Dnešní chytré telefony dosahují svým výkonem parametrů, jež byly obvyklé pro desktopové počítače před několika lety. Udávané taktování procesorů chytrých telefonů je někdy dokonce na stejné úrovni jako u některých současných notebooků či desktopů. To ovšem neznamená, že daná zařízení budou mít srovnatelný výkon. Procesor při své práci generuje teplo a to je v případě klasických počítačů poměrně dobře odváděno díky ventilaci, která je náročná na prostor. Ten však u mobilních zařízení zpravidla není k dispozici a chlazení je pouze pasivní. Přehřívání procesoru pak lze předejít například tzv. podtaktováním. Mobilní zařízení je navíc napájeno z baterií, tudíž je vyžadována co nejnižší spotřeba procesoru, což je opět řešeno zejména snížením jeho frekvence. V praxi tedy procesory mobilních telefonů zřídkakdy dosahují deklarované frekvence po delší časový úsek. [39]

4.2.2 Životní cyklus aplikace v rámci operačního systému

Životní cyklus mobilních aplikací se v některých fázích odlišuje od životního cyklu desktopových aplikací, které běží v rámci standardních operačních systémů. Cyklus je zde výrazně řízen v závislosti na přidělování prostředků operačním systémem, což je nutné brát v potaz již při vývoji samotné aplikace.



Obrázek 1: Životní cyklus aplikace v prostředí operačního systému iOS [40]

Mobilní aplikace procházejí takzvanými stavy, které definuje mobilní operační systém. Na příkladu operačního systému iOS (viz obrázek 1) lze vidět následující stavy, které se zpravidla používají i v případě ostatních diskutovaných mobilních OS (Android a Windows Phone):

- **Neběžící (Not running)** – Aplikace nebyla spuštěna, případně byla ukončena systémem.
- **V popředí (Foreground)**
  - **Neaktivní (Inactive)** – Aplikace je spuštěna v popředí, ale nepřijímá události, může však provádět jiný kód.
  - **Aktivní (Active)** – Aplikace je spuštěna v popředí a přijímá události.
- **V pozadí (Background) –** Aplikace je v pozadí a provádí kód. Zpravidla pouze v krátkém čase, než se stane suspendovanou. Aplikace však může požadovat extra čas pro provádění a zdržet se v tomto stavu déle.

Ukončená (Suspended) – Aplikace je v pozadí a nevykonává kód. Systém převádí aplikace do tohoto stavu automaticky bez upozornění. Pokud je v systému málo paměti, může být suspendovaná aplikace automaticky ukončena.

Z výše uvedeného tedy vyplývá, že v případě nedostatku zdrojů, může být mobilní aplikace kdykoliv ukončena operačním systémem, což se také v praxi, na rozdíl od desktopových

aplikací, stává. Je to mechanizmus, který má v prvé řadě zajistit schopnost vykonávat primární funkce mobilního zařízení, jako je například příjem hovoru.

Již při vývoji aplikace je tedy nutno brát v úvahu výše uvedený životní cyklus zejména v kontextu s omezenými hardwarovými prostředky, které může operační systém rozdělovat. V případě, že bude aplikace například vyžadovat nadměrné množství paměti, bude systémem nekompromisně ukončena.

O změnách stavu je aplikace operačním systémem informována pomocí tzv. notifikací. Aplikační interface pak definuje celou řadu přetížitelných metod, které lze použít jako obsluhu události změny stavu. Například při přechodu do pozadí lze přetížit v případě platformy Android metodu onPause (definováno třídou Activity rozšiřující třídu ApplicationContext) [41], v případě platformy iOS metodu applicationWillResignActive (definováno protokolem UIApplicationDelegate) [42] a v případě platformy Windows Phone metodu OnSuspending (v případě namespace Windows.UI.Xaml) [43]. Vývojář mobilní aplikace musí mít v povědomí celou řadu výše uvedených notifikací (událostí) a správně je využívat například pro náhlé uložení dočasných uživatelských dat a následné načtení z perzistentního úložiště ve chvíli, kdy je práce uživatele v aplikaci přerušena například příchozím hovorem a z důvodu nedostatku paměti je pak celá aplikace ukončena.

4.2.3 Sandboxing, oprávnění aplikace, sdílení souborů

V rámci zvýšení bezpečnosti a stability aplikací používají v současné době hlavní mobilní platformy tzv. sandboxing. Aplikace tak běží v oddělených procesech ve vlastním souborovém systému a nesdílejí data či soubory běžnou cestou, jak je zvykem v oblasti desktopových systémů. [44] [45] [46]

Za účelem zabezpečení a ochrany citlivých dat, která jsou obzvláště v mobilních zařízeních vysoce exponována, je pro mobilní aplikace zavedena celá řada dalších bezpečnostních omezení, jež se aplikací týkají. Bez nich by bylo velmi jednoduché naprogramovat například aplikaci, která bez vědomí uživatele zaznamenává jeho hovory a odesílá je pomocí datového připojení na vzdálený server. Rovněž i přístup k jiným citlivým datům, jako jsou fotografie, kontakty aktuální GPS poloha apod., vyžaduje souhlas uživatele. Zdroj [47] popisuje možnosti vývojáře v oblasti získání oprávnění aplikace pomocí tzv. manifest souboru na platformě Android. Pro platformu iOS a Windows Phone neexistuje ekvivalent manifest souboru. Jakmile vývojář použije některé z veřejných API operačního systému, jež přistupuje k citlivým údajům, povolení k přístupu uděluje samotný uživatel aplikace při jejím běhu. Zajímavé srovnání přístupu Android a iOS platformy, převážně z uživatelského hlediska, lze nalézt v [48].

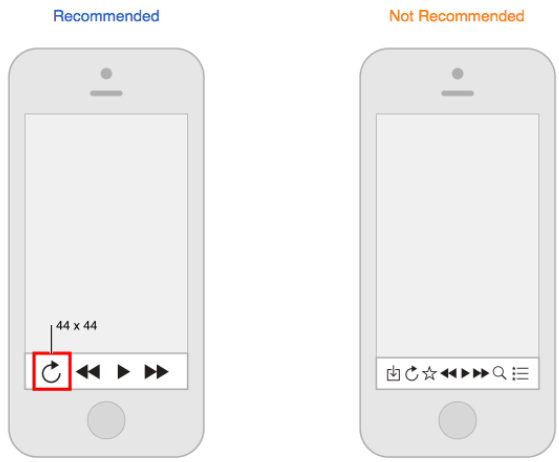
4.2.4 Dotykové ovládání

Klasické desktopové systémy nebyly v minulosti navrženy pro dotykové ovládání a lze tak říci, že programové možnosti obsluhy tzv. multi-touch (dotyk více prsty) událostí přinesly až mobilní operační systémy na přenosných zařízeních. Prvním mobilním telefonem, který podporoval multi-touch události tak, jak je dnes známe, a zaznamenal velký komerční úspěch, je iPhone od společnosti Apple Inc., představený v lednu 2007. Dnes se zaužívaná gesta používají skrze všechny hlavní mobilní platformy a v porovnání s vývojem desktopových aplikací je nutné zaměřit velkou pozornost zejména na oblast uživatelského zážitku (user experience - UX) při fázi návrhu uživatelského prostředí aplikace (user interface - UI) a jeho následném testování.

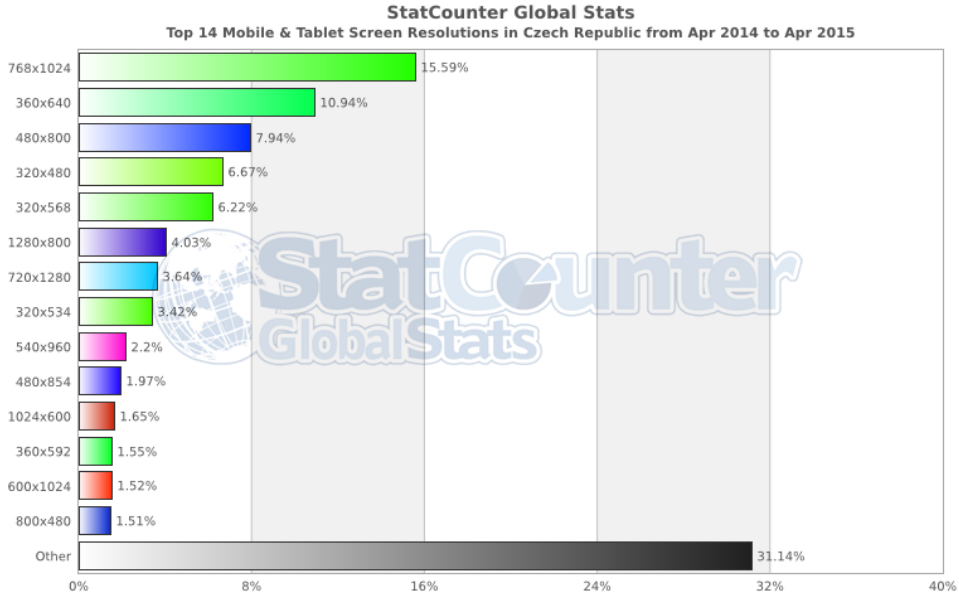
S dotykovým ovládáním, uživatelským zážitkem, mobilitou zařízení a způsobem jeho používání souvisí také celá řada animací a přechodů, které se v mobilních aplikacích na rozdíl od desktopových hojně objevují (zejména na platformě iOS) a mají za úkol zvýšit povědomí o kontextu a navigaci skrze jednotlivé části aplikace. [49] Z hlediska vývoje mobilních aplikací pomocí hybridních metod používajících webové technologie pro návrh UI je toto velmi problematická oblast, kdy nativní prvky a animace UI musí být pokud možno na co nejvyšší úrovni imitovány pomocí webových technologií. Mnohé postupy zaužívané z desktopových webových aplikací však nejsou z důvodu omezeného výpočetního výkonu a nedostatečné implementace HTML5 standardů využitelné.

4.2.5 Omezená zobrazovací plocha a variabilita zobrazovacích zařízení

Návrh grafického uživatelského rozhraní musí vždy reflektovat velikost zobrazovacího zařízení. V případě mobilních zařízení je tato velikost zpravidla podstatně menší, než v případě desktopů. Tomuto faktu je potřeba přizpůsobit UI mobilní aplikace a dále brát v potaz princip ovládání zařízení dotykem. Společnost Apple Inc. například doporučuje minimální plochu oblasti, kterou je možné zacílit dotykem lidského prstu o velikosti 44 x 44 px [6]. Apple je známý tím, že se uživatelským zážitkem v oblasti mobilních zařízení systematicky a intenzivně zabývá a poskytuje vývojářům na platformě iOS pokyny a nejlepší doporučení, jež jsou hojně přebírány a aplikovány i v rámci jiných mobilních operačních systémů.



Obrázek 2: Příklad specifikace minimální dotykové oblasti společností Apple Inc. [6]



Obrázek 3: Statistika rozlišení obrazovek mobilů a tabletů v ČR z období duben 2014 až duben 2015 [50]

V mobilním světě lze také hovořit o vysoké míře variability úhlopříček a dpi (viz obrázek 3), což je současně trend, který je poslední dobou řešen i v oblasti desktopových webových aplikací. Nativní metody návrhu UI přinášejí vlastní způsoby řešení tohoto problému pomocí základního rozlišení aplikačních layoutů na mobilní na straně jedné a vhodné pro větší displeje tabletů na straně druhé. Hybridní aplikace využívající webové technologie pro návrh

uživatelského rozhraní pak používají zejména nové technologie ze standardu CSS3 – media queries, pro aplikaci různých kaskádových stylů pro různé velikosti displejů.

4.2.6 Uživatelský zážitek

Při procesu interakce člověka s počítačem (HCI – Human Computer Interaction) můžeme zaznamenat různé úrovně kvality komunikace mezi technologickým produktem a jeho uživatelem, neboli různé úrovně uživatelského zážitku (UX – User Experience).

V současné době je trh s mobilními aplikacemi poháněn zejména módními trendy a úspěšnost aplikace je dána její oblíbeností u široké veřejnosti bez hlubších technických znalostí. Zde hraje velkou roli právě kvalitní uživatelský zážitek. Znamená to, že ovládání aplikace musí být intuitivní a vzhled musí být jednoduchý a návodný. Pokud je uživatelský zážitek nevyhovující, znamená to, že běžný uživatel není schopen s aplikací produktivně pracovat a uživatelské prostředí je pro něj matoucí. Termín User Experience (UX) se se v minulých letech nejvíce objevoval v souvislosti s webovými stránkami, což jsou v podstatě aplikace cílené na velké množství návštěvníků/uživatelů. Vezmeme-li v úvahu klíčové aspekty mobilních aplikací, jako je omezená zobrazovací plocha, či dotykové ovládání, je zřejmé, že poskytnutí intuitivního uživatelského prostředí je zde ještě důležitější.

Klíčovou vlastností, která velmi ovlivňuje míru uživatelského zážitku, jsou animace. Ty mají v rámci mobilních platforem velmi specifické využití, které je až v kontrastu se současnými desktopovými a webovými aplikacemi, jež animace používají spíše v malé míře. Při dotykovém ovládání na malých displejích pomáhají správně zvolené animace zvýšit povědomí o kontextu a zlepšují orientaci v rámci mobilní aplikace. Hovoří se zejména o přechodech obrazovek (tzv. swipe efektech), poskytnutí odezvy či navození pocitu přímé manipulace s objekty uživatelského rozhraní. [49]

Plynulost, patřičné a konzistentní použití animací, respektující doporučení dané platformy, pomáhá docílit vysoké míry uživatelského zážitku. V rámci nativního vývoje je tato oblast dobře zdokumentována, vývojáři používají standardní nativní knihovny a některé objekty uživatelského rozhraní již poskytují patřičné animace transparentně, bez nutnosti jejich implementace. Naproti tomu v rámci hybridního vývoje, kdy aplikace používají webové technologie pro tvorbu uživatelského rozhraní, není možné nativní knihovny animací jednoduše využít a příprava aplikace s dostatečnou mírou uživatelského zážitku vyžaduje speciální úsilí a zejména dobrou znalost vhodných webových technologií, jako je CSS3 či Java Script. Další možností je využití UI frameworku, který definuje základní objekty uživatelského rozhraní a má již implementované i standardní animace.

4.2.7 Konektivita

Moderní aplikace velmi často využívají různé služby dostupné jedině přes datovou síť (webové služby, cloud). V případě klasických počítačů je rychlé připojení k Internetu zpravidla dostupné. Mobilní zařízení jsou však závislá pouze na bezdrátovém připojení přes Wi-Fi potažmo datovou síť mobilního operátora. Během návrhu mobilní aplikace je proto nutné řádně otestovat její provoz nejen na rychlém Wi-Fi připojení, ale také na běžně dostupném datovém připojení operátora, včetně problematického signálu a velmi pomalých přenosů na úrovni technologie EDGE (průměrná rychlost stahování 150 Kb/s [51]).

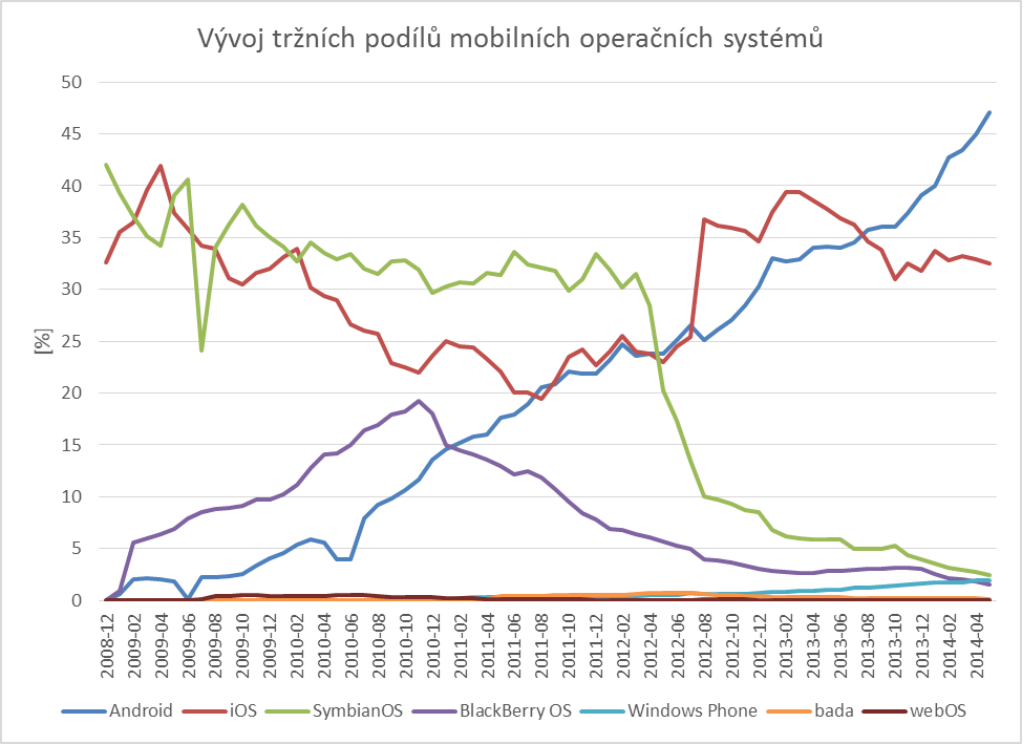
4.3 Mobilní operační systémy a jejich zastoupení na trhu

Dle statistik společnosti StatCounter (viz tabulka 1) jsou na celosvětovém trhu nejsilnějšími hráči tyto 3 mobilní operační systémy: Android (přes 58%), iOS (takřka 30%) a Windows Phone (sotva 2%). Dříve velmi rozšířený SymbianOS se již blíží pouze 1% podílu celosvětového trhu a trend tak jasně napovídá tomu, že je tato platforma odsouzena k zániku, což již potvrzují například evropské statistiky. [3]

Tabulka 1: *Aktuální podíly na trhu mobilních operačních systémů [3]*

Datum	Android	iOS	Windows Phone	SymbianOS	BlackBerry OS
2015-01	54.47	30.38	1.86	1.43	1.11
2015-02	55.3	30.78	1.87	1.26	1.02
2015-03	56.68	29.78	1.9	1.14	1.03
2015-04	58.24	27.79	1.98	1.13	1.06

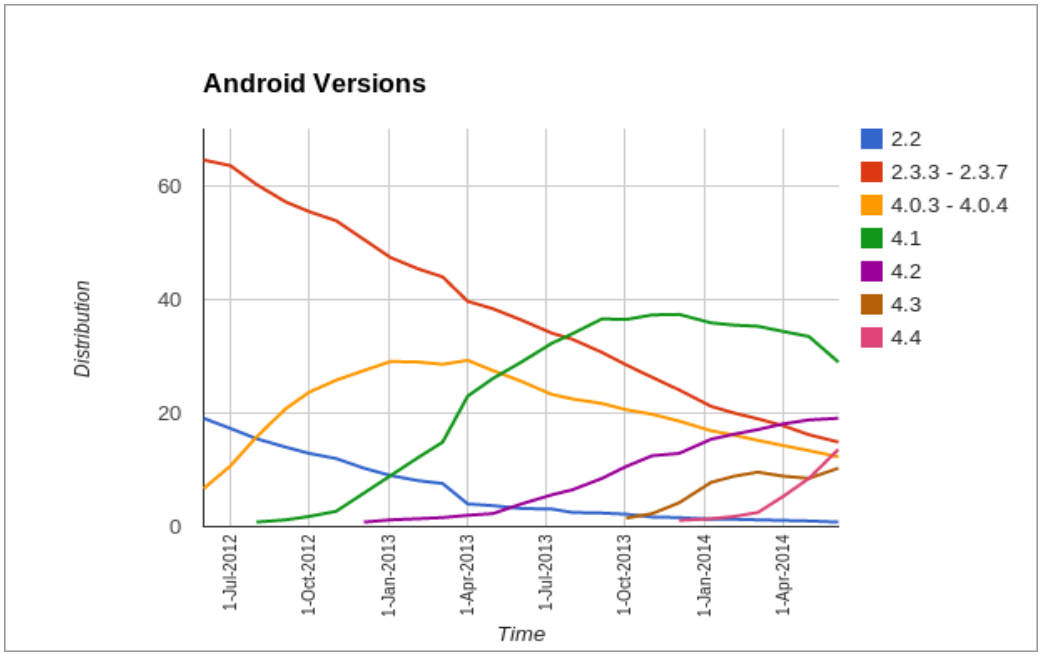
Dynamičnost mobilní platformy prokazují historické trendy zastoupení jednotlivých mobilních operačních systémů a jejich změny v čase, které jsou častější a výraznější, než je obvyklé ve světě desktopových operačních systémů. Na obrázku 4 je patrný nárůst podílu operačního systému Android na úkor ostatních operačních systémů – zejména dříve rozšířených Blackberry OS a SymbianOS. Změny zachycené v časovém úseku zhruba 5 let jsou opravdu radikální a jedinou platformou, která si zachovává stabilní zastoupení na trhu, je iOS.



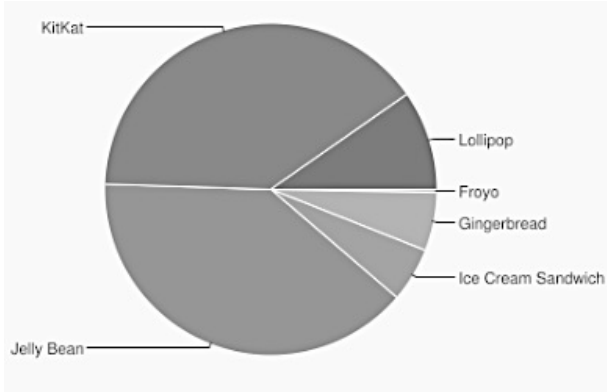
Obrázek 4: Vývoj tržních podílů mobilních operačních systémů prosinec 2008 – květen 2014 [3] [52]

Stejně tak vydávání nových verzí v rámci konkrétního operačního systému je podstatně častější, než je tomu u verzí určených pro desktopové počítače. Tento fakt reflektuje poptávku uživatelů po mobilních zařízeních a nových funkcích jejich operačních systémů. Nejvíce se tato skutečnost projevuje u platformy Android, která představila od září 2008, kdy vyšla první verze Android 1.0, do května 2015 již 34 verzí operačního systému (včetně Android Wear [53]). Průměrně je tak ročně uvedeno na trh takřka 5 verzí OS. Pro názornou ilustraci četnosti vydávání jednotlivých verzí operačních systémů a změny jejich zastoupení na trhu je na obrázku 5 znázorněno zastoupení jednotlivých verzí platformy Android v letech 2012 až 2014.

Graf (viz obrázek 6) a data uvedená v tabulce 2 pak ukazují aktuální zastoupení verzí ke květnu 2015. Zastaralé a problematické verze 2.2 - 2.3.7 už mají konečně marginální zastoupení (asi 6 %), ale verze < 4.4, které jsou problematické zejména z hlediska provozu webových hybridních mobilních aplikací, jsou zastoupeny stále takřka 45%.



Obrázek 5: Zastoupení jednotlivých verzí systému Android na trhu v letech 2012-2014 [19]

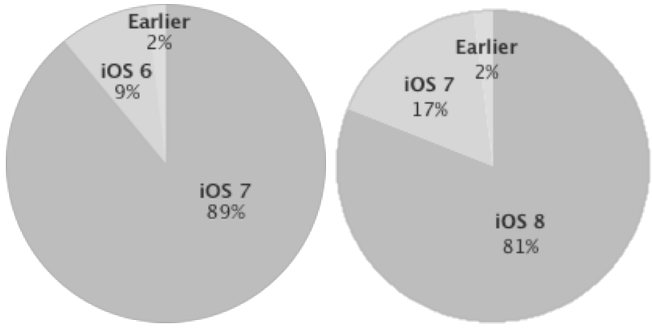


Obrázek 6: Aktuální graf zastoupení jednotlivých verzí Android OS [54]

Tabulka 2: *Distribuce jednotlivých verzí OS Android ke květnu 2015 [54]*

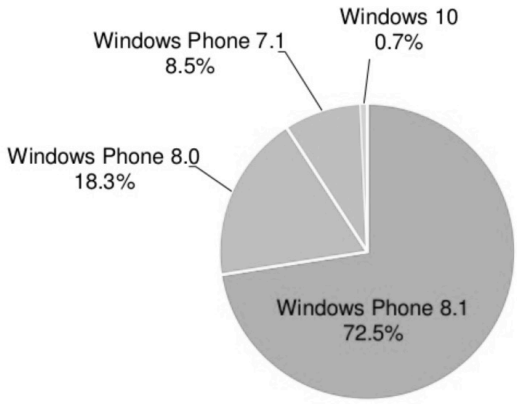
Version	Codename	API	Distribution
2.2	Froyo	8	0.3%
2.3.3 - 2.3.7	Gingerbread	10	5.7%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	5.3%
4.1.x	Jelly Bean	16	15.6%
4.2.x		17	18.1%
4.3		18	5.5%
4.4	KitKat	19	39.8%
5.0	Lollipop	21	9.0%
5.1		22	0.7%

Co se týče platformy iOS, společnost Apple představuje nové verze také poměrně často, avšak zároveň se snaží vydáváním nových zařízení, jež používají výlučně nejnovější verzi OS, a především jednoduchými aktualizacemi starších zařízení redukovat počet aktuálně se vyskytujících verzí operačního systému na trhu. To samozřejmě vývojářům třetích stran značně zjednodušuje proces vývoje mobilní aplikace, jelikož není nutné zajišťovat podporu celé řadě starších verzí používaných operačních systémů. Levý koláčový graf na obrázku 7 vyjadřuje procentuální zastoupení verzí systému iOS k červnu 2014 (aktuální verze iOS 7). Pravý graf pak zobrazuje aktuální stav, zhruba po roce, ke květnu 2015 (aktuální verze iOS 8). Z obrázku je patrné, že ekosystém jednotlivých verzí u společnosti Apple funguje dobře a díky jednoduchým aktualizacím používá většina uživatelů vždy poslední verzi operačního systému. Zpravidla se ale v mezidobí vydávání nových verzí objevují 2 rozšířenější verze OS, poměr se však velmi rychle mění ve prospěch verze novější.



Obrázek 7: Zastoupení jednotlivých verzí systému iOS na trhu k červnu 2014 (vlevo) a květnu 2015 (vpravo) [55]

Přestože společnost Microsoft byla historicky jedna z prvních společností, která vyvinula a úspěšně představila operační systém pro mobilní telefon, její aktuální verze mobilního operačního systému Windows Phone 8 dosahuje v roce 2015 sotva hranice 2% celosvětového tržního podílu. [3] Přitom první verze operačního systému Windows Mobile byla uvedena již v roce 2000, což je 7 let před představením první verze iPhone. Propad obliby systému Windows Mobile jasně ukázal, že není-li investováno dostatek úsilí do aktualizace a inovace systému, nepomůže ani historický náskok. Lze tedy usuzovat, že pokud by jedna ze silnějších společností uvedla na trh nový operační systém, který značně předčí svými možnostmi konkurenci (jako v roce 2007 iPhone se svým dotykovým ovládáním), může být aktuální zastoupení platforem během několika měsíců minulostí. Poté co Microsoft „usnul na vavřínech“, pokusil se obnovit dávnou slávu novou verzí mobilního operačního systému Windows Phone. Ten však nepřinesl žádná revoluční řešení a nejspíše i proto roste tržní podíl Microsoftu v oblasti mobilních technologií i přes slibné prognózy jen velmi pomalu. Aktuálně (květen 2015) se na trhu objevují převážně 3 různé verze OS – WP 7.1, 8.0 a 8.1. Do statistik se ovšem pomalu začíná dostávat i chystaná verze Windows 10.



Obrázek 8: Zastoupení jednotlivých verzí Windows Phone na trhu (březen 2014) [56]

4.4 Klasifikace metod vývoje mobilních aplikací

S rychlým rozvojem mobilních technologií se v posledních letech objevují stále nové přístupy k tvorbě mobilních aplikací. Různé zdroje pak uvádějí různé dělení těchto metod a to zejména v oblasti CP aplikací, kde se objevují stále nové přístupy, nástroje a vývojové frameworky. Nejčastěji se hovoří o těchto třech hlavních metodách vývoje mobilních aplikací: Vývoj nativních mobilních aplikací, vývoj mobilních webových aplikací a vývoj CP mobilních

aplikací. Klasifikace CP metod vývoje mobilních aplikací však není v současné době v literatuře jednotná a dílčí dělení a třídy budou zmíněny v kapitole 4.4.3.

4.4.1 Vývoj nativních mobilních aplikací

Nativní mobilní aplikace je aplikace navržená výlučně pro konkrétní platformu, vytvořená pomocí standardních vývojových nástrojů konkrétní platformy a na ní přímo spustitelná bez nutnosti použití další abstraktní vrstvy. Je vyvíjena pomocí softwarového vývojového balíku označovaného jako SDK – Software Development Kit, který dodává přímo vývojář dané platformy. Mezi hlavní výhody nativních aplikací patří stabilita, vysoký výkon a možnost přímého přístupu k dalším součástem mobilního operačního systému pomocí tzv. API. Lze tak využívat dalších hardwarových součástí zařízení, jako je například fotoaparát, gyroskop, GPS a další.

Pokud je ale potřeba aplikaci dodat na více platforem, je u metody nativního vývoje nutné použít pro každou platformu specifické vývojové nástroje, programovací jazyk i odlišné přístupy. Vývoj pomocí nativních nástrojů tedy bývá obecně označován jako nejnákladnější. [28] [57] [58].

Vývoj nativních aplikací pro Android

V případě platformy Android potřebují vývojáři tzv. Android SDK. Tento balík vývojových nástrojů obsahuje vývojové prostředí Android Studio (základem je IDE IntelliJ, dříve se používalo prostředí Eclipse s pluginem ADT), nástroje Android SDK, platformu Android OS aktuální verze a emulátor mobilního zařízení. [59] Vzhledem k otevřenosti platformy Android je vývojářský balík ke stažení zdarma z oficiálního webu pro vývojáře developer.android.com. Nástroje jsou k dispozici pro desktopové operační systémy Windows, Mac OS X i Linux.

Nativní aplikace pro Android jsou programovány zpravidla v jazyce JAVA (pomocí Android NDK lze určité části kódu implementovat i v C a C++ [60]). Zdrojový kód je následně sestaven do binárního balíčku s koncovkou APK, který lze distribuovat pomocí oficiálního obchodu Google Play. Android aplikace lze dokonce distribuovat mimo oficiální distribuční cesty (například poslat emailem, přes cloud úložiště, či nahrát do zařízení přes SD kartu) a tyto aplikace je také možno nainstalovat, pokud je v nastavení zařízení povolena instalace aplikací z nedůvěryhodných zdrojů.

Ze tří hlavních diskutovaných mobilních platforem je pro vývojáře OS Android nejvíce přívětivý v oblasti vydávání aplikací. Toto tvrzení se opírá zejména o fakt, že publikační proces hotové aplikace je zde nejkratší (čas od vložení APK balíčku do Google Play do doby, kdy je možno z obchodu aplikaci stáhnout). Je to dáno tím, že aplikace prochází před samotnou publikací pouze automatizovaným testem a již zde neprobíhá recenze, kterou provádí pracovník

v případě platformy iOS a Windows Phone. Aplikaci tak lze zveřejnit v řádu několika hodin, dle zkušeností autora a také uživatelů platformy Quora [61] se doba pohybuje zpravidla okolo 3 hodin.

Publikační proces vyžaduje stejně jako u ostatních dvou platforem vlastnictví vývojářského účtu. Ten je vystaven na jméno firmy či fyzické osoby a vydané aplikace jsou s ním pevně spjaty. Bez tohoto účtu, jehož roční poplatek je 25 USD (květen 2015) [62], není možné vydávat aplikace na Google Play.

Vývoj nativních aplikací pro iOS

Základním programátorským vybavením je v případě platformy iOS zejména hardware s operačním systémem Mac OS. Program XCode (obsahující SDK pro vývoj iOS aplikací) lze totiž provozovat pouze na platformě Mac OS. Existuje sice projekt Hackintosh [63], který umožňuje provozovat systém Mac OS virtualizovaně na jakékoliv platformě, avšak je to v rozporu s podmínkami používání software od společnosti Apple Inc. XCode je vývojové prostředí pro iOS i Mac OS aplikace, které obsahuje zároveň simulátor pro iPhone i iPad.

Nativní aplikace jsou programovány v jazyce Objective-C a nově je možno využít i programovací jazyk Swift, který má za úkol přivést do světa iOS více programátorů, pro které je syntaxe jazyka Objective-C překážkou (Swift se podobá spíše skriptovacímu jazyku). Aplikace jsou sestaveny do binárního balíčku s koncovkou IPA, který je možno pomocí programu XCode nahrát do webové služby iTunes Connect. Zde je potřeba vyplnit profil aplikace pro App Store a zažádat o revizi a vydání aplikace.

Platforma iOS je nejvíce problematická z hlediska publikačního procesu hotových aplikací a založení samotného vývojářského účtu. Žadatelé o vývojářský účet zde narazí na nutnost vyplnění mnoha žádostí, splnění legislativních povinností a podepsání smluv přímo se společností Apple Inc., přičemž celý proces se počítá v řádu týdnů.

Jakmile je vývojářské konto úspěšně založeno, může být publikována samotná aplikace. Ve srovnání s platformou Android je v procesu revize přítomen i lidský článek, zaměstnanec společnosti Apple Inc., který kontroluje, zda aplikace neporušuje některý z požadavků publikačního procesu. Doba od vložení aplikace a zažádání o revizi až po samotné publikování na App Store se pohybuje kolem 10 až 14 dnů. Dle [64] činí doba publikace ke konci května 2015 v průměru 11 dní. I publikace následných aktualizací trvá v průměru déle než týden a tyto lhůty je nutné brát v potaz již během vývojového procesu.

Vývoj nativních aplikací pro Windows Phone

Vyvíjet nativní aplikace pro Windows Phone lze pomocí Windows Phone SDK, které je volně stažitelné a obsahuje vývojové prostředí Windows Visual Studio Express for Windows Phone

(umožňuje i komerční vývoj) a také emulátor zařízení Windows Phone. Vývojové prostředí je dostupné pouze pro platformu Windows, tu lze však virtualizovat na libovolném hardware bez porušení licenčních podmínek.

Aplikace jsou vyvíjeny nejčastěji v programovacích jazycích C# nebo VB.NET. Dříve známé balíčky aplikací XAP jsou od verze WP 8.1 nahrazeny kontejnerem APPX.

Také společnost Microsoft umožňuje vydávání aplikací pomocí oficiálního kanálu Windows Phone Store jedině na základě registrace vývojářského účtu. Cena účtu je 365 Kč v případě programu Individual a 1720 Kč v případě programu Company (květen 2015) [65]. Co se týče samotného publikačního procesu, také v případě společnosti Microsoft do něj vstupuje lidský faktor ve formě recenzenta. Publikace je však rychlejší než u společnosti Apple Inc. Prvotní publikace trvá několik dní a následné aktualizace aplikace zpravidla 1 den.

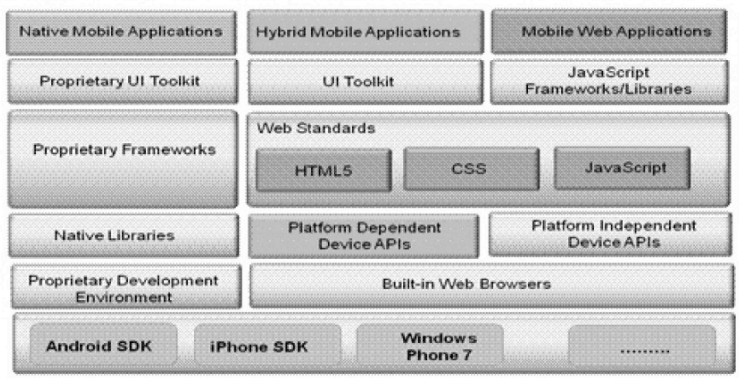
4.4.2 Vývoj mobilních webových aplikací

Mobilní webové aplikace jsou založeny čistě na webových technologiích jako je HTML, JavaScript a CSS. Ve své podstatě jsou to klasické webové stránky běžící v internetovém prohlížeči mobilního zařízení, které mají ovšem optimalizované uživatelské rozhraní pro koncovou zobrazovací jednotku. V roce 2011 se začal objevovat v souvislosti s tvorbou univerzálních „mobilně-desktopových" webových aplikací termín „responzivní layout" (Setfive – Talking to the World, 2012), který využívá novou specifikaci značkovacího jazyka HTML5 (W3C, 2014) a jazyka kaskádových stylů CSS3 (W3C, 2014) a má za úkol identifikovat technické parametry zařízení, z nějž návštěvník přistupuje. Tuto novou technologii tak lze efektivně využít pro optimalizaci uživatelského rozhraní na základě vlastností konkrétního displeje, případně existuje celá řada tzv. UI frameworků, vhodných pro tvorbu uživatelského prostředí webové mobilní aplikace. Co se týče hlubší interakce s hardwarovými senzory koncového zařízení, lze využívat pouze omezené možnosti HTML5 platformě nezávislých API.

4.4.3 Vývoj CP mobilních aplikací a klasifikace dílčích metod

Na rozdíl od metod nativního vývoje CP vývoj mobilních aplikací si klade za cíl maximalizaci procenta společného zdrojového kódu pro uživatelské rozhraní i aplikační část. V současné době lze nalézt velké množství technologií a na nich založených metod CP vývoje, lišících se procentem společného kódu. Tím, že je tato oblast velmi dynamická a rychle se rozvíjí, není klasifikace dílčích metod ve vědeckých publikacích jednotná.

Například v [66] byl publikován následující model architektury mobilních aplikací, viz obrázek 9.



Obrázek 9: Model architektury mobilních aplikací – Nativní, Hybridní a Webové mobilní aplikace dle [66]

Z uvedeného modelu architektury je zřejmé, že nativní aplikace vycházejí pouze z proprietárních nástrojů a nativních knihoven. Uživatelské rozhraní a aplikační logika v případě CP mobilních aplikací je zde pak tvořena pomocí webových standardů, podobně jako v případě klasické mobilní webové aplikace. Přestože existují i odlišné architektury, CP mobilní aplikace jsou na obrázku 9 zastoupeny skupinou hybridních mobilních aplikací, které však zohledňují pouze ty, jež jsou tvořeny pomocí webových technologií. V zahraniční literatuře lze tuto skupinu nalézt také pod označením web-to-native [67] [68] [69] [70]. Je zde však správně vyobrazen hlavní rozdíl mezi webovými hybridními a čistě webovými mobilními aplikacemi, který spočívá v tom, že hybridní aplikace mohou využívat platformě závislé API zařízení (akcelerometr, kameru, kontakty, síť…) podobně jako nativní aplikace, kdežto čistě webové mobilní aplikace mohou využít jedině platformě nezávislé API, které poskytuje nově HTML5 (např. HTML5 Geolocation, Web Storage, App Cache…).

Výše uvedený model bere v úvahu CP aplikace tvořené pomocí technologií jako je Apache Cordova [71] či Phonegap [72], avšak nezohledňuje jiné existující přístupy, jako jsou například CP interpretované či generované mobilní aplikace, jež uvádí [73] či [74].

Interpretované mobilní aplikace využívají běhového prostředí, kde je zdrojový kód interpretován za běhu aplikace a pomocí API jsou volány nativní funkce zařízení. Do této kategorie lze zařadit například Appcelerator Titanium Mobile [29], vývojový nástroj využívající JavaScript pro tvorbu aplikační logiky i generování UI, Rhodes [75], založený na webových technologiích a Ruby skriptech interpretovaných v lokálním běhovém prostředí, či Qt Mobile [76], jehož C++ kód je možné spouštět pomocí různých nativních API (např. Java Native Interface – JNI na platformě Android).

Generované mobilní aplikace jsou na rozdíl od interpretovaných kompilovány přímo do nativního kódu a není tak vyžadováno specifické běhové prostředí, jako je hostitelský kontejner

aplikace na konkrétním zařízení. Aplikace je popsána pomocí modelového jazyka, jenž definuje funkcionalitu a chování aplikace na vysoce abstraktní úrovni. Na základě modelu aplikace je pak generován nativní kód, specifický pro každou platformu. Tato vývojová metoda je poměrně málo rozšířená, zejména kvůli vysoké míře obecnosti modelu aplikace, což vede k problematické implementaci specifických funkcí či nekomfortní úpravě automaticky generovaného zdrojového kódu. Typickými zástupci jsou Applause [77] či iPhonical [78].

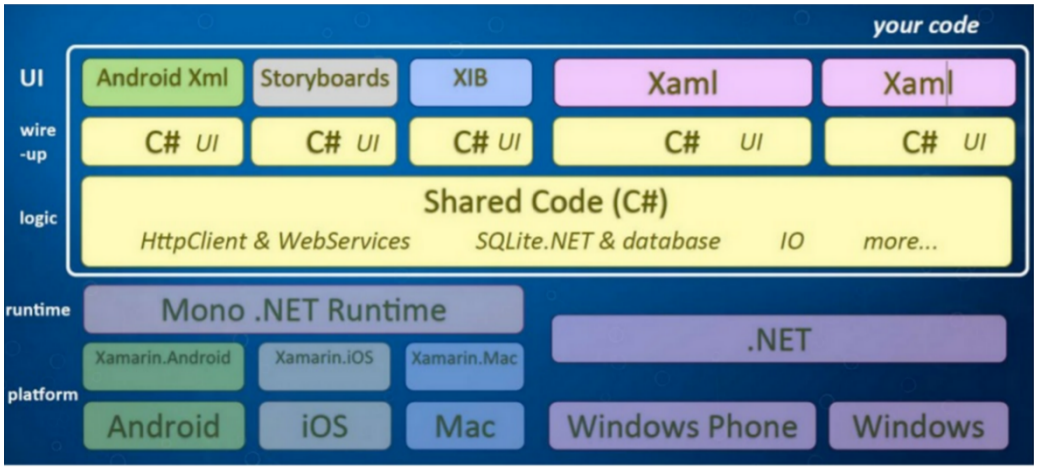
Metody vývoje CP mobilních aplikací lze dále rozdělit dle způsobu návrhu uživatelského rozhraní. Existují tyto základní druhy: Aplikace s nativním uživatelským rozhraním tvořeným pomocí nativních prvků UI, aplikace s uživatelským rozhraním „kresleným“ pomocí nativní grafické knihovny, aplikace s uživatelským rozhraním tvořeným kombinací nativních prvků a prvků kreslených pomocí webových technologií a aplikace s uživatelským rozhraním tvořeným čistě pomocí webových technologií.

Typickým zástupcem CP mobilních aplikací s nativním uživatelským rozhraním je technologie Xamarin [79]. Aplikace využívající „kreslené“ uživatelské rozhraní pomocí nativní grafické knihovny lze tvořit například pomocí technologie Appcelerator Titanium [29] či Qt Mobile [76]. UI kombinující nativní prvky s webovými technologiemi lze tvořit např. pomocí technologií Apache Cordova či Phonegap, stejně jako UI, jež je tvořeno pouze webovými technologiemi.

Při snaze o přenos kódu z jedné mobilní platformy na druhou lze také využít tzv. „cross-compiled“ metodu. Ta umožňuje díky překladačům například kompilovat aplikaci psanou pro platformu Android do jazyka C# (Windows Phone) nebo Objective-C (iOS). Přístup a samotná technologie XMLVM umožňující kompilaci jsou popsány v článku [80], jehož autoři metodu využili právě na přenos existující Android aplikace na platformy iOS a Windows Phone. Jsou zde však zmíněny také limity tohoto přístupu, jako je nemožnost přenosu funkcí API, které nejsou dostupné na cílové platformě, nebo například nemožnost ladit aplikaci na cílové platformě.

Bližší popis také jistě zaslouží dnes velmi rozšířená metoda tvorby CP aplikací pomocí technologie Xamarin [79]. Na obrázku 10 je uveden model jeho architektury, který zobrazuje jednotlivé části mobilní aplikace a technologie použité pro její implementaci a běh. Oblast uživatelského rozhraní je zde tvořena nativními knihovnami dané platformy. Uživatelské rozhraní je pak provázáno s logikou aplikace tvořenou kódem v jazyce C#, společným pro všechny platformy. Pokud je program sestaven pro platformu Windows 8, je pak spouštěn přímo pomocí Microsoft .NET frameworku v tzv. „Common Language Runtime“ (CLR) [81]. V případě sestavení pro platformy Android, iOS či Mac je pak využito mezivrstvy Mono .NET Runtime. Xamarin v podstatě implementuje knihovny, které vývojáři znali dříve pod názvem

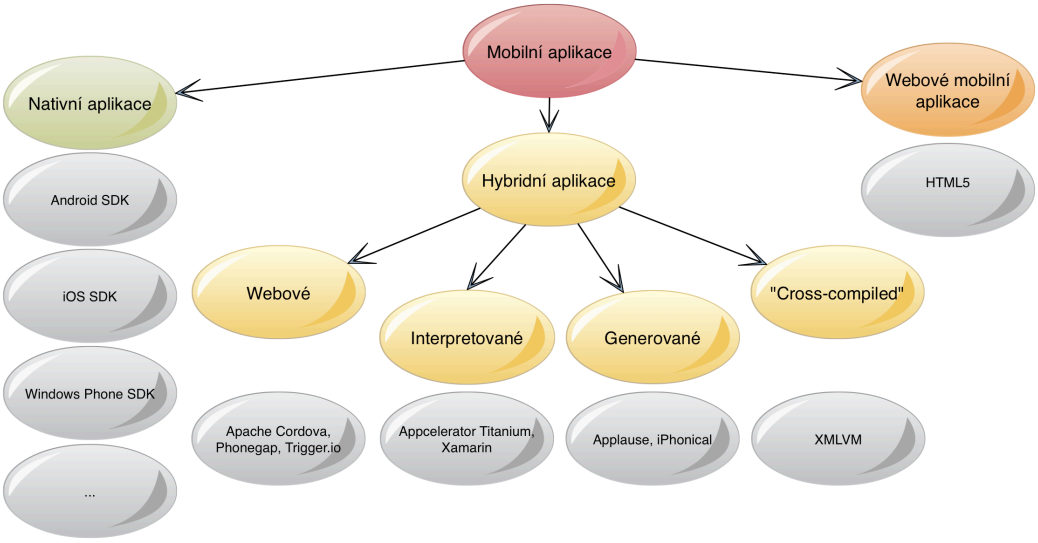
MonoDroid [82] a MonoTouch [83]. V rámci projektu Xamarin jsou dále vyvíjeny pod názvy Xamarin.Android a Xamarin.iOS [84]. Z výše uvedeného je zřejmé, že procento společného kódu je zde podstatně menší než v případě hybridních aplikací tvořených webovými technologiemi, jelikož celá část uživatelského rozhraní je zde tvořena nativními komponentami. Klasifikace technologie Xamarin z hlediska výše uvedených metod je problematická, jelikož pro platformu Android se běžně používalo kompilace JIT (Just in Time Compilation) [85], což odpovídá interpretovaným mobilním aplikacím, kdežto aplikace pro iOS byly kompilovány do nativního kódu technologií AOT (Ahead of Time Compilation) [86]. S příchodem Android OS 5.0 je však možné i Xamarin aplikace pomocí knihovny Xamarin.Android verze >5 kompilovat pomocí AOT [87].



Obrázek 10: Model „cross-platformní“ architektury Xamarin aplikací [84]

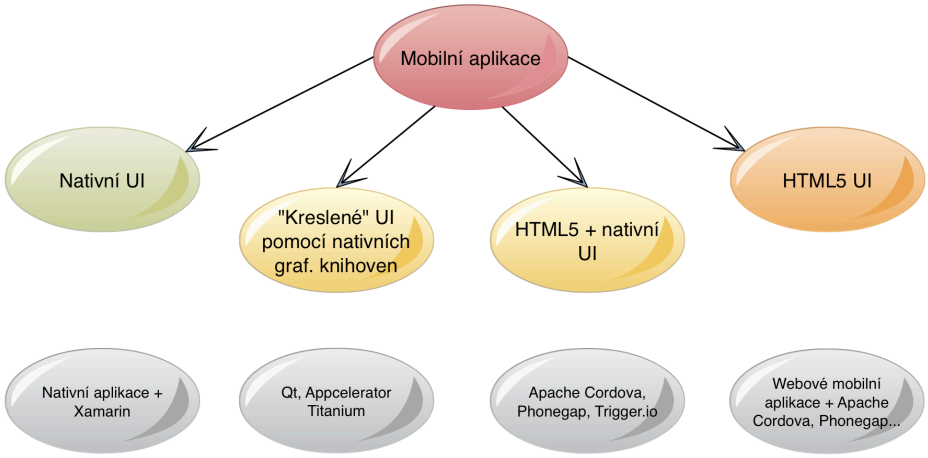
4.4.4 Shrnutí

Vzhledem k množství metod, které lze dnes pro vývoj mobilní aplikace využít, je na obrázku 11 znázorněno přehledné dělení druhů mobilních aplikací, které syntetizuje výše uvedené zdroje a poznatky. Uvedené metody lze tedy využít pro vývoj následujících druhů aplikací. Pod každým druhem jsou uvedeny příklady příslušných vývojářských nástrojů či technologií.



Obrázek 11: Klasifikace mobilních aplikací

V předchozí kapitole bylo uvedeno také dělení dle způsobu tvorby uživatelského rozhraní aplikace (viz obrázek 12). I zde jsou pod každým z druhů uvedeny konkrétní technologie, jež mohou být pro vývoj využity.



Obrázek 12: Dělení druhů mobilních aplikací dle návrhu UI

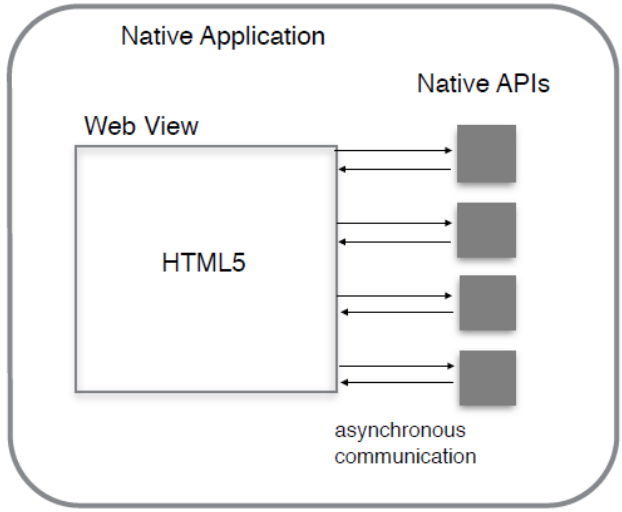
4.5 Webové hybridní mobilní aplikace

Experimentální část této dizertační práce se zaměřuje zejména na webové hybridní mobilní aplikace, které jsou založeny na webových technologiích (HTML5) a tzv. „wrapper“

technologii, neboli běhovém prostředí tvořeném mobilním internetovým prohlížečem. Toto běhové prostředí lze implementovat pomocí technologie Apache Cordova [71] či Phonegap [72].

4.5.1 Architektura webových hybridních aplikací

Webové hybridní mobilní aplikace mají mezi ostatními druhy CP mobilních aplikací (vyjma čistě webových) obecně nejvyšší míru sdílení zdrojového kódu mezi různými platformami, což je jejich největší devizou. Této míry je dosaženo právě díky webovým technologiím a existenci nativních prohlížečů webových stránek na všech rozšířených platformách. Pomocí nativního prohlížeče pak probíhá zobrazení a interpretace kódu aplikace (tvoří běhové prostředí). Hybridní mobilní aplikace je tvořena zčásti nativním kódem, který v podstatě pouze spouští okno aplikace a v něm okno nativního webového prohlížeče bez navigačních lišt a jiných ovládacích prvků (viz obrázek 13). Nativnímu prohlížeči je pak nastavena adresa vstupního HTML dokumentu aplikace. Z výše uvedeného je zřejmé, že výkon hybridních aplikací je podstatně nižší než v případě aplikací nativních. Je to dáno tím, že aplikačním kódem je zpravidla interpretovaný JavaScript a výkon je tedy dán faktickým výkonem hardware a rychlostí nativního prohlížeče daného zařízení. Pro mnohé aplikace je však výkon dostatečný.



Obrázek 13: Model architektury webové hybridní mobilní aplikace (Apache Cordova či Phonegap)

Pomocí tzv. „native-bridge“ pluginů lze navíc komunikovat z rámce aplikační logiky běžící v prohlížeči přímo s nativním kódem. Díky tomu pak lze náročnější výpočty nechat probíhat nativně. Pluginy jsou také vhodné pro komunikaci se specifickými knihovnami API, které jsou dnes nabízeny různými mobilními platformami (například Notifikace, Google Services, Health

Kit apod.) Právě díky nativním pluginům disponují hybridní aplikace takřka stejnými možnostmi v oblasti využívání hardware daného zařízení.

4.5.2 Procento společného zdrojového kódu pro různé platformy

Podstata sdílení kódu skrze různé platformy spočívá v principu multiplatformity webových technologií. Jde o možnost využívat jednu technologii, kterou lze provozovat na všech hlavních platformách. V současné době jde o technologii HTML5, kam patří značkovací jazyk HTML, jazyk kaskádových stylů a skriptovací jazyk JavaScript. Běhovým prostředím je pak nativní mobilní internetový prohlížeč (Web view), který je schopen zmíněné webové technologie interpretovat.

Podíl sdíleného zdrojového kódu aplikace pak může dosáhnout až 100% (v úvahu je brán čistě HTML5 zdrojový kód) a to i v případě, že jsou intenzivněji využívána JavaScriptová API pro komunikaci s nativními knihovnami. Komunikace přes tzv. "native-bridge" je univerzální a volají se tak totožné JavaScriptové funkce. Pokud se zaměříme na celkový zdrojový kód aplikace – včetně pluginů a zdrojového kódu nativních tříd pro inicializaci aplikace a běhového prostředí (nativního prohlížeče) – bude se pro různé platformy lišit. V úvahu je však brán pouze vyloženě aplikační kód, který je tvořen programátory aplikace.

4.5.3 Wrapper technologie

Mezi nejpoužívanější wrapper technologie patří Open Source projekt Apache Cordova a jeho komerční odnož Adobe Phonegap. Ta navíc poskytuje on-line nástroj Adobe Phonegap Build [88], který umožňuje sestavení aplikace pomocí cloudu, bez nutnosti instalovat jednotlivé vývojářské SDK příslušných platforem. Do služby Adobe Phonegap Build se nahrává pouze zdrojový kód HTML5 mobilní aplikace, který je pak dosazen do příslušných nativních projektů jednotlivých platforem. Pomocí konfiguračního souboru lze pak ovlivňovat jednotlivá nastavení nativního projektu a dále připojené pluginy. Výstupem služby je pak hotový balíček aplikace vhodný k distribuci přes oficiální distribuční kanály.

Další velmi známou technologií, zejména ze světa iOS aplikací je Trigger.io [89], který disponuje nejen vlastním javascriptovým API (Forge API) pro komunikaci s nativními knihovnami zařízení, nýbrž i funkcemi pro generování nativních UI prvků z javascriptového kódu aplikace. Trigger.io však není na rozdíl od výše zmíněných wrapper technologií Open Source a nabízí pouze placené programy.

4.5.4 Princip nativních pluginů

Jednotlivé mobilní operační systémy nabízejí vývojářům celou řadu nativních rozhraní, pomocí nichž lze komunikovat se specifickými částmi mobilního zařízení, jako je například

akcelerometr, GPS senzor, fotoaparát či kamera, souborový systém, síťové rozhraní apod. Tato rozhraní jsou však dostupná pouze v nativním programovacím jazyce příslušné platformy. Proto tvůrci wrapper technologií jako je Apache Cordova, Adobe Phonegap či Trigger.io a komunity těchto projektů, vytvářejí tzv. „native-bridge“ pluginy, jež jsou v podstatě javascriptovým rozhraním, vytvářejícím komunikační most do nativního kódu aplikace. Pomocí něj lze asynchronním systémem zpráv přistupovat přímo k funkcím nativních API konkrétní mobilní platformy.

Zdrojový kód níže uvádí příklad javascriptové části pluginu. 1. řádek zajistí připojení objektu pluginy myPlugin do objektu tzv. „global scope“ window. Přes tento objekt je plugin odkudkoliv z JavaScriptu volatelný a lze mu předat parametry str a callback. 2. řádek spouští metodu exec, kterou wrapper Apache Cordova implementuje. Metoda exec přebírá následující parametry: tzv. „Success Callback“ (funkce volaná v případě úspěšné komunikace), tzv. „Error Callback“ (funkce volaná v případě neúspěšné komunikace, výjimečných stavů apod.), tzv. „Service“ (jméno třídy, jež implementuje nativní funkcionalitu pluginu), tzv. „Action“ (jméno nativní metody v rámci tzv. „Service“, jež implementuje konkrétní funkcionalitu) a pole parametrů, které bude odesláno do metody execute implementované ve třídě „Service“ v rámci nativního prostředí (viz zdrojový kód níže).

```
1. window.myPlugin = function(strParam, callback) {
2.       cordova.exec(callback, function(err) {
3.            callback(Error state.');
4.       }, "MyPluginService", "MyPluginAction", [strParam] );
5. };
```

Výše uvedený plugin pak může být spouštěn odkudkoliv z JavaScriptu, díky napojení na globální objekt window (viz zdrojový kód níže).

```
1. window.myPlugin("String to pass", function(return) {
2.       console.log(return); // logování návratové hodnoty z
                              //  nativní části
3. });
```

Následující příklad pak demonstruje kód nativní části pluginu na platformě Android, konkrétně třídu umístěnou v balíčku src/org/apache/cordova/plugin/MyPluginService.java. Parametry metody exec volané nad objektem cordova v javascriptové části definují název třídy, která implementuje akci ke spuštění (zde jde o metodu MyPluginAction). Nativní logika může provést výpočet na návrat, poslat jakoukoliv hodnotu či data, která budou zpracována opět v javaskriptové funkci definované parametrem „Success Callback“ v případě úspěšného návratu (viz zdrojový kód níže).

```
1. package org.apache.cordova.plugin;
2. import org.apache.cordova.CordovaPlugin;
3. import org.apache.cordova.CallbackContext;
4. import org.json.JSONArray;
5. import org.json.JSONException;
6. import org.json.JSONObject;
7.
8.  /**
9.   * This class returns a string sent from JavaScript.
10. */
11.public class MyPluginService extends CordovaPlugin {
12.
13.    @Override
14.   public boolean execute(String action, JSONArray args,
   CallbackContext callbackContext) throws JSONException {
15.       if (action.equals("MyPluginAction")) {
16.            String strParam = args.getString(0);
17.            this.MyPluginAction(strParam, callbackContext);
18.            return true;
19.       }
20.       return false;
21.    }
22.
23.   private void MyPluginAction(String strParam,
   CallbackContext callbackContext) {
24.       if (strParam != null && strParam.length() > 0) {
25.           callbackContext.success(strParam);
26.       } else {
27.          callbackContext.error("Expected non-empty string.");
28.       }
29.   }
30.}
```

4.5.5 Příklady dostupných nativních pluginů

Webové hybridní mobilní aplikace jsou v současné době schopny využívat jen několik javascriptových API, které přinesla verze jazyka HTML5 a jsou podporovány mobilními webovými prohlížeči. Je to například API pro geolokaci, síťové informace, lokální úložiště, aplikační cache atd. Wrapper technologie proto přinášejí celou řadu pluginů, jež umožňují zejména komunikaci s různými specifickými senzory mobilních zařízení a dále umožňují využívat další nativní knihovny pomocí obecného javascriptového API. Apache Cordova například nabízí následující základní pluginy [71]:

- Battery Status (monitoring stavu baterie zařízení)
- Camera (využití vestavěné kamery zařízení)
- Console (přidání možnosti logování)
- Contacts (práce s databází kontaktů zařízení)
- Device (specifické informace o zařízení)
- Device Motion (práce s akcelerometrem zařízení)
- Device Orientation (získání směru natočení zařízení - kompas)
- Dialogs (nativní informační dialogy zařízení)
- FileSystem (přístup k nativnímu souborovému systému zařízení)
- File Transfer (přenos souborů v rámci nativního souborového systému)
- Geolocation (práce s GPS senzorem, získání GPS souřadnic zařízení)
- Globalization (práce s lokalizovanými formáty a jednotkami zařízení)
- InAppBrowser (možnost zobrazení URL adres v další instanci prohlížeče, v rámci webview)
- Media (možnost nahrávat a spouštět audio soubory)
- Media Capture (pořízení multimediálních souborů pomocí nativní aplikace pro zachycení audia/videa)
- Network Information (zjištění stavu datové sítě)
- Splashscreen (zobrazení a skrytí tzv. „splash“ obrazovky)
- Vibration (přístup k vibračnímu API zařízení)
- StatusBar (možnost zobrazení, skrytí a konfigurace nativního status pruhu aplikace)
- Whitelist (plugin pro povolení síťových požadavků aplikace)

5 EXPERIMENTÁLNÍ ČÁST

Experimentální část práce sestavuje pomocí aplikovaného výzkumu reprodukovatelnou metodu, kterou je možno použít při vývoji webových hybridních mobilních aplikací (WHMA). Metoda poskytuje informační oporu a pomáhá řešit jednu z největších překážek hybridního vývoje – neexistenci ověřených postupů či ucelených metod a problematický výběr z velkého množství softwarových nástrojů. Zaměřuje se především na reálné problémy vedoucí k prodlužování vývoje a zvyšování nákladů a předkládá jejich řešení, která umožňují dosáhnout časově a finančně vyváženého vývojového procesu.

Jednotlivé body experimentální části jsou strukturovány tak, aby odpovídaly na zásadní problémy jednotlivých vývojových fázi softwarového produktu dle ISO/IEC 90003 (analýza požadavků, návrh, implementace, integrace, testování, nasazení a ukončení) a to právě v kontextu WHMA.

Fáze analýzy požadavků se v úvodu zaměřuje na výběr obecného vývojového postupu – nativní, hybridní či čistě webový vývoj. V této části práce jsou uvedeny reálné výhody a nevýhody a možnosti jednotlivých metod, vycházející jak z praktických zkušeností s vývojem mobilních aplikací autora, tak z komparativních studií a vědecko-výzkumných publikací zabývajících se tímto tématem. Cílem této dílčí části je poskytnout informaci pro kvalifikovaný výběr základní vývojové metody dle konkrétních požadavků na koncovou aplikaci.

Další podkapitoly se již výhradně zabývají výběrem konkrétních technologií a softvérových nástrojů pro vývoj WHMA. Na otázku výběru implementační metody uživatelského rozhraní odpovídají dílčí závěry kapitoly 5.2. Kvalifikovaný výběr vhodného vývojového HTML5 UI frameworku je řešen reprodukovatelnou metodou multikriteriální analýzy postavené na vážených výběrových kritériích navržených a popsaných v kapitole 5.4. V současné literatuře pouze sporadicky řešenou problematikou výkonnostního testování HTML5 UI frameworků se zabývá kapitola 5.5. Stanovuje metody měření a přináší srovnání jednotlivých softvérových rámců.

Fáze implementační a integrační a dále také fáze testování přinášejí v rámci vývoje WHMA řadu problémů, na které není v současné literatuře poukázáno a vývojářské týmy tak čelí při adopci webových hybridních metod nečekaným komplikacím. Obecně jde o nekonzistenci vlastností mobilních prohlížečů, zejména v oblasti podpory HTML5 API či CSS3. Výše uvedené implementační problémy byly zaznamenány při implementačních testech provedených v kapitole 5.5 a dále také při testování běhových prostředí na reálné mobilní aplikaci UTB [90]. Kapitola 5.6 se proto věnuje jejich identifikaci a zkoumání příčin, zatímco kapitola 5.7 stanovuje na základě implementačních testů a měření nejlepší doporučení.

Navržené metody jsou pak v praxi aplikovány při vývoji mobilní aplikace UTB a jejich úspěšnost, zejména v oblasti uživatelského zážitku, je verifikována pomocí vyhodnocení dotazníkového průzkumu uživatelů aplikace (kapitola 5.9).

5.1 Charakteristika a srovnání základních metod vývoje mobilních aplikací

V rámci experimentální části bylo jako jeden z dílčích cílů zpracováno srovnání diskutované webové hybridní metody (WHMA) s nativní metodou (NA) či metodou tvorby čistě webových mobilních aplikací (WEB). V rámci zahraničních internetových zdrojů lze nalézt množství článků diskutujících zpravidla jednotlivé technologie na velmi obecné úrovni. Mezi zahraničními publikacemi však lze nalézt i detailnější srovnávací studie a dokonce i několik publikací zabývajících se přímo vývojem mobilních aplikací pro více platforem, viz kapitola 2.

Jednotlivá srovnávací kritéria, použitá v této práci, byla stanovena na základě zkušeností autora s vývojem mobilních aplikací, názorů odborníků z řad vývojářů z praxe, komunitních fór (StackOverflow, Quora) a dále výše uvedených vědeckých publikací.

5.1.1 Znalostní požadavky na vývojový tým

Každá metoda vývoje mobilních aplikací má specifické požadavky na znalosti vývojového týmu a ten musí, pro pokrytí hlavních mobilních platforem, ovládat následující technologie:

Tabulka 3: *Znalostní požadavky vývojového týmu*

NA	Android SDK + Java Apple iOS SDK + Objective-C/Swift Windows Phone SDK + C++/ C#/.NET/Visual Basic/XAML
WHMA	Webové technologie – HTML, CSS, JavaScript Wrapper technologie – Apache Cordova/Phonegap Nativní SDK + programovací jazyk (v případě programování nativních pluginů)
WEB	Webové technologie – HTML, CSS, JavaScript

Z výše uvedeného souhrnu je zřejmé, že nejméně náročné na znalost vývojového týmu je využití metody vývoje mobilních aplikací pomocí čistě webových technologií. Tato metoda je tedy také nejméně nákladná a velkou výhodou je i fakt, že aktuálně je na trhu velké množství kvalifikovaných webových kodérů.

Metoda vývoje webových hybridních aplikací vyžaduje také znalosti webových technologií, ale navíc i znalost wrapper technologie, která se stará o běh aplikace na konkrétní platformě a základní znalost SDK konkrétní platformy pro případ sestavování a ladění aplikací. Vzhledem

k tomu, že takřka veškerý kód, který vývojáři zpracovávají, je pro jednotlivé platformy společný, samotný proces vývoje je podstatně méně nákladný, než v případě nativních aplikací. Je však zároveň dražší než v případě aplikací tvořených čistě webovými technologiemi. V případě programování vlastních pluginů, pracujících na nativní úrovni, však může vyvstat potřeba znalosti nativního programovacího jazyka konkrétní platformy.

Nativní aplikace vyžadují vždy znalost konkrétního SDK a programovacího jazyka, jenž je využíván na cílové platformě. Uvažujeme-li tvorbu aplikace pro několik platforem, znamená to také větší vývojový tým, přičemž implementační fáze vývojového procesu probíhá pro každou platformu odděleně. Je nutno vzít v úvahu to, že programátorů se znalostí konkrétního mobilního SDK a jeho programovacího jazyka je v současné době na trhu podstatně méně než webových kodérů. Výše uvedené fakta činí metodu nativního vývoje nejnákladnější.

5.1.2 Požadavky na vybavení vývojového týmu

Vývojový tým musí disponovat nejen znalostmi, ale i patřičným operačním systémem, potažmo konkrétním hardware, na kterém může provozovat vývojovou platformu. Tabulka níže uvádí konkrétní požadavky na operační systém pro jednotlivé SDK.

Tabulka 4: *Srovnání požadavků na vybavení vývojového týmu*

NA	Android SDK – Windows/Linux/Mac OS Apple iOS SDK – Mac OS Windows Phone SDK – Windows
WHMA	Android SDK – Windows/Linux/Mac OS Apple iOS SDK – Mac OS Windows Phone SDK – Windows Bez SDK v případě cloudových „build“ služeb (např. Adobe Phonegap Build [88])
WEB	Webové technologie – Windows/Linux/Mac OS

Požadavky na SDK v případě nativních a hybridních aplikací jsou v podstatě stejné, jelikož oba přístupy vyžadují patřičné vybavení pro proces sestavení aplikace. U hybridních aplikací však existují i cloudové služby umožňující vzdálené sestavení aplikace bez potřebného SDK [88]. Pro sestavení aplikace pro konkrétní platformu je tedy obvykle potřebné konkrétní SDK, které však nemusí podporovat všechny běžné desktopové operační systémy. Pouze Android SDK lze provozovat na Windows, Linux i Mac OS. Největší komplikace způsobuje vývoj pro zařízení společnosti Apple, protože její SDK vyžaduje systém Mac OS, který nelze legálně

virtualizovat na jiném hardware než právě od Apple [91]. Vývoj pro platformu Windows Phone sice také vyžaduje pouze prostředí desktopového operačního systému Windows, ten však lze virtualizovat, a proto je nejčastější volbou vývojářů právě systém Mac OS, kde pak lze používat všechna zmíněná SDK.

Aplikace tvořené čistě webovými technologiemi lze vyvíjet na kterémkoliv z operačních systémů, který je podporuje.

5.1.3 Jednoduchost vývoje

Určit jednoznačně a objektivně obecné kritérium jednoduchosti vývoje jednotlivých metod není možné. V rámci jeho evaluace však může být nepřímo hodnoceno procento společného kódu aplikace v případě multiplatformního vývoje a dále nutnost znalosti různých technologií, programovacích jazyků či vývojových frameworků.

Pro hodnocení konkrétních koncových technologií je dle [92] nutno brát v úvahu kvalitu dokumentace, učící křivku, dostatek příkladového kódu, jenž popisuje problematická místa, aktivní komunitu uživatelů apod. Pokud hodnocená technologie využívá známé modely, může to podstatně urychlit proces učení.

Tabulka 5: *Srovnání jednoduchosti vývoje*

NA	Oddělený zdrojový kód pro všechny platformy / Znalost konkrétních SDK
WHMA	Veškerý kód aplikace může být společný, či tvoří většinu celkového kódu / Znalost wrapper technologií a webových technologií
WEB	Veškerý kód aplikace je společný / Znalost webových technologií

5.1.4 Rychlost vývoje

Rychlost vývoje je pro mnohé vývojářské společnosti jeden z nejdůležitějších parametrů, protože na něm závisí nejen náklady na vývoj, ale také rychlost uvedení produktu na trh. V této oblasti se jednotlivé metody vývoje opět diametrálně liší. Obecně se nejrychlejším vývojem vyznačují mobilní webové aplikace, což vychází z podstaty webových technologií, které jsou navrženy pro rychlé sdílení informací v rámci Internetu. Z toho benefitu čerpají samozřejmě i hybridní aplikace využívající webové technologie. Avšak rychlý vývoj je často následován delší testovací fází, jelikož kvůli velkému množství verzí nativních webových prohlížečů se jednotlivé prvky GUI nemusí chovat stejně v různých zařízeních. Problém ladění pro jednotlivá zařízení je obvykle menší v případě volby nativní metody vývoje. Nicméně zde je tolik odděleně probíhajících vývojových procesů, kolik je cílových platforem, což tuto metodu činní časově nejnáročnější.

Tabulka 6: *Srovnání rychlosti vývoje*

NA	Pomalejší oddělený vývoj pro jednotlivé platformy / kratší fáze testování
WHMA	Rychlejší vývoj společného kódu / delší fáze testování
WEB	Rychlejší vývoj společného kódu / delší fáze testování

5.1.5 Návrh grafického uživatelského rozhraní

Co se týče návrhu GUI, nativní aplikace používají přímo nativní knihovny pro tvorbu UI. [93] [94] [95] V případě hybridních aplikací se situace různí. Buď jsou používány webové technologie pro návrh GUI, nebo mohou být také využívány nativní knihovny pro tvorbu GUI (např. Xamarin [84]). V oblasti čistě webových mobilních aplikací pak nativní knihovny pro tvorbu GUI použít nelze.

Při návrhu vývojáři GUI obvykle preferují používání standardních objektů, jež zajišťují základní funkce interakce uživatele s aplikací. Pomocí nich lze snadno dodržet návrhové vzory dané platformy. Hodnotný uživatelský zážitek lze zajistit pouze správným návrhem GUI, což ve velké většině aplikací nutně vede k použití vzorů. Více se touto problematikou zabývá Bochers [96], který výše uvedené potvrzuje myšlenkou, že vzory jsou základem úspěšného řešení častých problémů použitelnosti interaktivních systémů.

Pokud je GUI tvořeno čistě webovými technologiemi, může být tvorba standardních ovládacích prvků, korespondujících s běžným nativním vzhledem, technicky a časově náročná. V současné době však existuje celá řada HTML/CSS/JS GUI frameworků, které přinášejí sadu standardně vyhlížejících ovládacích prvků, značně tak urychlují vývoj v oblasti hybridních mobilních aplikací a odstraňují tak zmíněnou nevýhodu.

Tabulka 7: *Srovnání technologií, používaných pro tvorbu GUI*

NA	Nativní SDK GUI knihovny
WHMA	HTML, CSS, JS / Nativní SDK GUI knihovny
WEB	HTML, CSS, JS

5.1.6 Distribuční kanály

Stejně důležitá jako fáze implementace je také fáze distribuce hotové aplikace ke koncovým zákazníkům. Její úspěšnost může hrát klíčovou roli v celkové ziskovosti projektu. Výhodou nativních a stejně tak WHMA aplikací je možnost distribuce pomocí oficiálních distribučních

kanálů jednotlivých platforem (obchody App Store, Google Play, Windows Store). Distribuční kanály jsou zároveň kanálem prodejním. Čistě webové mobilní aplikace však nejsou klasickým aplikačním balíčkem, proto je lze šířit pouze pomocí sdílení jejich URL adresy. To samozřejmě vyžaduje odlišnou marketingovou kampaň než v případě oficiální distribuce aplikačních balíčků a také vlastní systém uživatelských plateb tak, aby byla zajištěna monetizace projektu.

Tabulka 8:

NA	Oficiální distribuční kanály
WHMA	Oficiální distribuční kanály
WEB	Sdílení URL adresy, vlastní systém monetizace

5.1.7 Aktualizace aplikací

Trendem dnešního softwarového světa jsou rychlé změny a velmi často vycházející aktualizace. Oblast mobilních aplikací je toho názorným příkladem. Dle [97] je takřka 30% aplikací aktualizováno alespoň jednou za měsíc. Zde jsou samozřejmě rozdíly v nákladnosti aktualizace aplikací tvořených diskutovanými metodami a také v roztříštěnosti verzí. Zatímco u webových aplikací je samotný proces aktualizace velmi přímočarý, jelikož se aktualizuje pouze jediná verze zdrojových kódů, která je ihned po přístupu do aplikace distribuována do koncových zařízení, u hybridních a nativních aplikací se může objevit roztříštěnost verzí mezi jednotlivými uživateli. Uživatel má totiž v tomto případě nad procesem aktualizace plnou kontrolu a může nebo nemusí přechod na aktuální verzi provést.

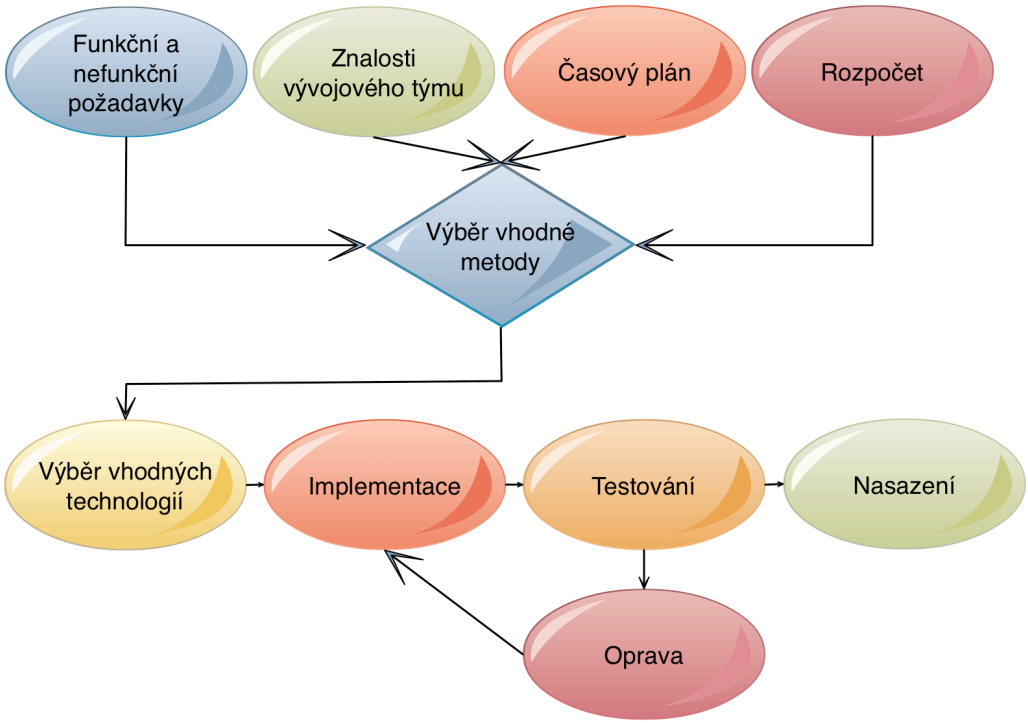
Nejméně nákladné na implementaci aktualizace jsou tedy webové mobilní aplikace, dále hybridní aplikace, a to díky tomu, že jednotlivé platformy mohou sdílet většinu vývojového kódu. Nejnákladnější jsou pak nativní aplikace, kde je nutno implementovat aktualizaci pro každou platformu odděleně.

Tabulka 9: *Srovnání aktualizací*

NA	Oddělený vývoj aktualizací pro každou platformu / možná roztříštěnost verzí mezi uživateli
WHMA	Rychlá aktualizace díky sdílenému kódu / možná roztříštěnost verzí mezi uživateli
WEB	Přímočará aktualizace společného kódu / aktuální verze se vždy stahuje ze serveru

5.2 Klíčové aspekty procesu vývoje WHMA

Před samotnou volbou metody vývoje WHMA, by měly být zváženy klíčové aspekty, výhody, nevýhody i rizika tohoto přístupu. Zjednodušený model vývojového procesu korespondující s fázemi ISO/IEC 90003 je znázorněn na obrázku 14. Vstupními parametry, které rozhodují o výběru obecné metody, jsou tedy: funkční a nefunkční požadavky na aplikaci, znalosti vývojového týmu, časový plán a rozpočet. V následujících podkapitolách budou výše uvedená kritéria vyhodnoceny z hlediska vhodnosti použití metody vývoje WHMA. Kvalitativní hodnocení bude vztaženo k metodě nativního vývoje.



Obrázek 14: Proces vývoje mobilní aplikace včetně klíčových parametrů výběru vývojové metody

5.2.1 WHMA z hlediska funkčních a nefunkčních požadavků

Funkční a nefunkční požadavky na aplikaci jsou jedním z nejdůležitějších parametrů výběru vývojové metody. Specifické požadavky na aplikaci mohou přímo některou metodu z výběru vyřadit, jelikož není pro získání konkrétního výsledku vhodná. Z toho důvodu byly identifikovány obecné funkční (viz tabulka 10) a nefunkční (viz tabulka 11) požadavky na

mobilní aplikace a následně byly ohodnoceny z hlediska použití metody vývoje WHMA. Hodnocení je v rozmezí 1–3, kdy 1 znamená nejvíce vhodné a 3 nejméně vhodné. V případě, že je tato metoda nevhodná, je doporučena i alternativní vývojová metoda, včetně odůvodnění.

Tabulka 10: *Vhodnost WHMA z hlediska funkčních požadavků*

Funkční požadavek	Hodnocení vhodnosti	Komentář
Asynchronní komunikace s webovými službami	1	Technologie AJAX pro asynchronní komunikaci je v oblasti HTML5 aplikací velmi známá a používaná. Může být plně využita pro volání existujících webových služeb i v rámci WHMA.
Podpora grafických operací	3	Standard HTML5 již sice podporuje technologii WebGL, která umožňuje využít i výpočetní schopnosti GPU mobilního zařízení, jeho implementace ve starších běhových prostředích však úplně chybí, nebo není implementována dle standardů. Pro graficky náročné aplikace se podstatně více hodí nativní přístup, kde je možno využít celou řadu nativních knihoven specializujících se na práci s grafikou.
Přístup k hardware (senzorům) zařízení	1	V případě WHMA lze říci, že přístup k hardware zařízení je bezproblémový a to díky nativním pluginům (více v kapitole 4.5.4).
Přístup ke specifickým API (Health API, iCloud API, Touch ID API)	2	V případě přístupu ke specifickým knihovnám určité platformy může nastat situace, že není dostupný žádný oficiální nativní plugin pro běhové prostředí. V takovém případě je nutno jej naprogramovat za použití nativního programovacího jazyka. Funkčně jsou však možnosti v podstatě identické, jako v případě tvorby nativní aplikace.

Tabulka 11: *Vhodnost WHMA z hlediska nefunkčních požadavků*

Nefunkční požadavek	Hodnocení vhodnosti	Komentář
Podpora více platforem	1	WHMA jsou ideálním nástrojem pro tvorbu multiplatformních aplikací a to zejména proto, že v podstatě veškerý zdrojový kód aplikační logiky včetně UI může být společný pro cílové platformy. Technologie Apache Cordova podporuje v současné době sestavení aplikace pro platformy Amazon Fire OS, Android, BlackBerry 10, Firefox OS, iOS, Ubuntu, Windows Phone 8, Windows, Tizen. [98]
Velikost	1	Obecně jsou WHMA velmi malé. V reálné situaci však velikost může narůst použitím nadstandardního běhového prostředí viz kapitola 5.4.
Výkon	2	WHMA jsou ve srovnání s nativními aplikacemi vždy méně výkonné, jelikož aplikační logika je implementována pomocí JavaScriptu a v mnoha případech tak nejsou využity reálné nativní výpočetní schopnosti zařízení. Navíc i výkon JavaScriptu je omezen kvalitou implementace v rámci konkrétního běhového prostředí. Dosažení takřka nativního výkonu je však možné pomocí nativních pluginů, které umožňují komunikaci systémem zpráv z platformě nezávislé webové části do platformě závislého zdrojového kódu.
Zabezpečení	2	Zabezpečení WHMA je ve většině aspektů shodné se zabezpečením aplikací nativních. Obecně jsou však WHMA považovány za méně bezpečné a to kvůli velmi jednoduché přístupnosti zdrojového kódu aplikační logiky. Ta je totiž součástí tzv. zdrojů aplikace, podobně jako obrázky či audio soubory, a je tedy z balíčku aplikace jednoduše extrahovatelná. Řešením zvyšujícím míru zabezpečení je tzv. obfuskace zdrojového kódu.

5.2.2 WHMA z hlediska znalostí vývojového týmu

Volba vývojových metod a s nimi souvisejících technologií se zpravidla odvíjí od znalostí vývojového týmu. Lze říci, že WHMA jsou v tomto ohledu nejpřívětivější, protože jednoduchý projekt lze v ideálních podmínkách realizovat pouze se znalostí webových technologií a wrapper technologie (např. Apache Cordova). Webových vývojářů je také na trhu práce podstatně více, než vývojářů, kteří se specializují na jazyk a vývojové nástroje konkrétní mobilní platformy.

Reálné projekty však ukazují, že při tvorbě mobilní aplikace, která má být svými uživateli kladně hodnocena, nelze v žádném případě vycházet pouze ze znalostí a zkušeností získaných z čistě webových projektů. Tým by měl velmi dobře znát architekturu, zvyklosti a nejlepší doporučení všech koncových platforem, pro které bude aplikace publikována. Tvrzení se týká zejména návrhu uživatelského rozhraní (UI) a jeho ovládání, které souvisí s uživatelským zážitkem (UX).

V případě aplikací se složitější funkcionalitou či s nutností přístupu k nativním knihovnám a zvláště pak v případech vyžadujících tvorbu vlastního nativního pluginu, je nezbytná také velmi dobrá znalost jazyk a vývojářských nástrojů cílové platformy.

V následujícím textu je uvedeno několik obecných případových studií aplikace a nutné znalosti konkrétních technologií vývojového týmu.

Jednoduchý mobilní katalog produktů

Příkladem může být jednoduchý katalog produktů, s klasickým zobrazením "master-detail", což je rozložení, kdy jedna obrazovka obsahuje seznam produktů s možností prokliku na detail produktu. Je vhodné znát doporučení týkající se grafiky a architektury navigace v rámci aplikace pro konkrétní koncovou platformu. Vhodné je využít HTML5 UI framework, pro tvorbu uživatelského rozhraní s možností změny vzhledu imitující koncovou platformu. Obecně však stačí vhodně postavenou webovou aplikaci sestavit pomocí vybrané wrapper technologie (např. Apache Cordova).

Technologie: UI doporučení koncové platformy, HTML5, HTML5 UI FW, wrapper technologie (Apache Cordova), nástroje pro sestavení

HTML5 hra

Pro tvorbu HTML5 hry lze využít některý z vývojových frameworků k tomu určených. Lze však vystačit čistě se standardními webovými technologiemi. V případě využívání grafických animací je nutné vybrat vhodné nadstandardní běhové prostředí, tak aby grafické operace probíhaly pokud možno přímo na GPU.

Technologie: HTML5, wrapper technologie (Apache Cordova), běhová prostředí (Crosswalk, Webview+)

GPS aplikace

V případě tvorby GPS aplikace je stěžejní přístup k senzoru GPS. To lze pomocí pluginu wrapper technologie, který umožňuje přímo z javascriptového kódu komunikovat s nativní knihovnou obsluhující daný sensor. Využívání standardních pluginů nevyžaduje v podstatě žádnou znalost nativního programovacího jazyka.

Technologie: UI doporučení koncové platformy, HTML5, HTML5 UI FW, wrapper technologie (Apache Cordova), běhová prostředí (Crosswalk, Webview+), standardní pluginy běhového prostředí (GPS)

Firemní klientská aplikace podnikového informačního systému

V případě složitější aplikace, která slouží jako klient existujícího informačního systému, je důležité napojení na existující webové služby. Zde lze využít opět webových technologií (např. AJAX) pro asynchronní komunikaci přes protokol HTTP či HTTPS.
Technologie: HTML5, HTML5 UI FW, wrapper technologie (Apache Cordov, běhová prostředí (Crosswalk, Webview+), webservices, standardní pluginy běhového prostředí (Práce s SSL certifikáty)

Aplikace pro pořízení a grafickou úpravu fotografií

V případě implementace funkce fotoaparátu, či přístupu k fotografiím lze využít standardního pluginu, bez nutnosti znalosti nativního programování. Pro složitější grafickou úpravu fotografie již bude nutné implementovat vlastní nativní plugin a to pro každou z koncových platforem v příslušném programovacím jazyce. Předpokladem je, že úpravy budou natolik složité, že si nevystačí s novými grafickými filtry technologie CSS3 v rámci standardu HTML5.
Technologie: HTML5, wrapper technologie (Apache Cordova), běhová prostředí (Crosswalk, Webview+), standardní pluginy běhového prostředí (Camera), nativní jazyk koncové platformy pro programování vlastních pluginů.

5.2.3 WHMA z hlediska časových a finančních nákladů

V oblasti odhadu náročnosti vývoje softwarového produktu lze dnes nalézt celou řadu přístupů. Například modely COCOMO, SLIM a SEER-SEM pro odhad nákladů na vývoj software jsou založeny na metrice LOC (Lines of Code, či SLOC (Source Lines of Code) – počet řádků zdrojového kódu). [99] Metrika LOC je zdánlivě jednoduše měřitelným parametrem, který přímo vypovídá o délce zdrojového kódu. Dnešní literatura však definuje hned několik různých způsobů měření (Měření počtu řádků obsahujících spustitelný příkaz, měření počtu řádků obsahujících spustitelný příkaz a definici dat, měření počtu řádků obsahujících spustitelný příkaz, definici dat a komentáře, měření počtu fyzických řádků na vstupní obrazovce, měření počtu řádků ukončených logickými oddělovníky). [100] Lze říci, že pomocí LOC lze velmi rychle získat představu o rozsahu zdrojového kódu, avšak srovnání je možné pouze v rámci stejného programovacího jazyka, nebo alespoň programovacího jazyka podobného druhu. Metrikou LOC nelze například srovnávat projekty psané v nízkoúrovňovém jazyce (např. Assembler) s projekty psanými v kterémkoliv z vyšších programovacích jazyků (vyšší míra abstrakce, např. C++, PHP). Současná literatura diskutuje další nevýhody této metriky pro použití odhadu náročnosti, jako je například nízká vypovídající schopnost (samotný proces programování tvoří pouze část z celkového úsilí věnovaného tvorbě SW produktu), nebo nízká

míra souvislosti s funkcionalitou (např. zkušený programátor je schopen produkovat při nižší míře LOC vyšší míru funkcionality). Mezi další limity této metriky patří také rozdílnost programovacích jazyků nebo používání GUI nástrojů a automaticky generovaného kódu v rámci moderních vývojových prostředí.

I přes výše uvedené nedostatky je počet řádků zdrojového kódu využíván jako hlavní parametr metrik pro odhad časové a tím i finanční náročnosti softwarového projektu a v rámci této práce je použit i jako metrika srovnání velikosti zdrojového kódu WHMA a nativní mobilní aplikace.

Portál property-cross.com (více v kapitole 2.1) nabízí přehled frameworků pro vývoj mobilních aplikací (zejména WHMA). V rámci praktického srovnání je zde implementována referenční mobilní aplikace (aplikace pro vyhledávání v realitách) pomocí každého ze srovnávaných frameworků. Cílem této podkapitoly je přinést srovnání náročnosti webového hybridního projektu s projektem nativním, které je založeno na metrikách vývojového projektu jednotné reálné mobilní aplikace. Aplikace musí nabízet stejnou funkcionalitu, byť je tvořena jinou metodou vývoje či odlišným frameworkem. Zdrojové kódy referenční aplikace vytvořené v rámci protálu property-cross.com jsou dostupné na webu GitHub.com.

Přímo v rámci portálu property-cross.com byl vytvořen skript pro měření LOC pro každý projekt. Primárním účelem je však pouze poskytnout informaci o míře sdíleného kódu v rámci jednotlivých vývojových projektů. Při hlubší analýze skriptu bylo zjištěno, že do statistik jsou započítávány i soubory některých knihoven, které programátor pouze k projektu připojí.

Za účelem zpřesnění hrubého odhadu pracnosti na základě počtu řádek kódu (LOC), které musí programátor reálně vytvořit, bylo měření provedeno tak, aby byly počítány pouze soubory, které je nutno v rámci projektu reálně naprogramovat. Měření probíhalo přímo na zdrojových kódech publikovaných na webu GitHub.com, pomocí příkazu GIT terminálu, pro výpis počtu řádků vybraných souborů, který vypisuje údaj o počtu fyzických řádků na vstupní obrazovce:

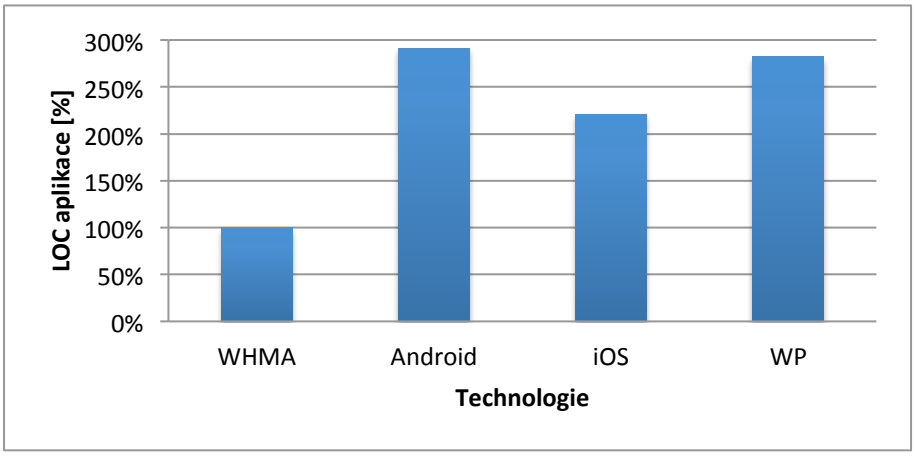
```
git ls-files | xargs wc –l
```

Výsledky srovnávají počty LOC aplikace vytvořené pomocí vybraných webových hybridních frameworků s nativní aplikací pro Android, iOS a Windows Phone. Z webových hybridních frameworků byli do měření zahrnuti následující zástupci: Emy, Famous, Intel Application Framework, Ionic, jQTouch, jQuery Mobile, Kendo UI, Lavaza, Lungo, PhoneJS, Sencha Touch.

Tabulka 12 udává v prvním řádku sloupce WHMA aritmetický průměr naměřených výsledků počtu řádků (LOC) 11-ti zkoumaných webových hybridních aplikací. Dále jsou uvedeny výsledky LOC měřené na projektech testovací aplikace pro platformy Android, iOS a Windows Phone.

Tabulka 12: *Měření LOC na projektu testovacích aplikací portálu property-cross.com*

Technologie	WHMA	Android	iOS	WP
LOC aplikace	998	2907	2205	2818
LOC aplikace [%]	100%	291%	221%	282%



Obrázek 15: Srovnání velkosti zdrojového kódu reálné aplikace portálu property-cross.com

Z výsledků (viz obrázek 15) vyplývá, že WHMA mají v průměru podstatně méně řádků zdrojového kódu, který je nutno naprogramovat, než aplikace nativní. V případě platformy Android či Windows Phone je nativní kód rozsáhlejší o takřka 300%. V praxi však vypovídací hodnota těchto údajů naráží na výše zmíněná omezení metriky LOC, jako je existence pokročilých vývojových nástrojů pro generování zdrojového kódu či GUI, kterými disponují právě vývojová prostředí nativních aplikací. Lze tedy říci, že v kontextu výše uvedených výsledků, bude jeden řádek zdrojového kódu v rámci webové hybridní aplikace představovat vyšší míru programátorského úsilí, než řádek aplikace nativní (kde se běžně využívá generování šablon tříd, případně zdrojového kódu GUI).

Praktické zkušenosti z vývoje reálných mobilních aplikací UTB či Retigo Vision a dalších ukázaly, že časová náročnost programování projektu webové hybridní aplikace, která pokrývá všechny platformy, je menší nebo nejhůře stejná jako programování nativního projektu pro jednu platformu. Předpokládáme-li tedy publikaci pro všechny tři majoritní mobilní platformy, můžeme ušetřit až dvě třetiny času, potažmo nákladů. Z výše uvedeného by se také dala odhadnout hrubá náročnost jednoho řádku zdrojového kódu webové hybridní aplikace, který by v některých případech mohl prezentovat až 3x vyšší pracnost, než v případě nativních aplikací. Přesnější stanovení míry úsilí na řádek zdrojového kódu v rámci použití různých vývojových

metod, či nástrojů, tak může být otázkou dalšího výzkumu. Ten by měl vzít v potaz následující důležité parametry ovlivňující finální časovou a tím i finanční náročnost projektu:

- **Náročnost implementace řádku zdrojového kódu** (závislost na kvalitě a vyspělosti vývojových nástrojů – WHMA vykazují oproti nativním aplikacím vyšší náročnost, způsobenou nedostatkem vyspělých vývojových nástrojů)
- **Náklady na řádek zdrojového kódu** (závislost na dostupnosti programátorů z vyžadovanou znalostí – WHMA vykazují oproti nativním nižší náklady, díky levnější pracovní síle v podobě webových kodérů)
- **Nutný počet řádků k implementaci totožné funkcionality** (závislost na zkušenosti programátora i na konkrétním programovacím jazyce)

Metrika založená na parametru LOC může být také nahrazena metrikou, která bere v úvahu náročnost implementace funkčních bodů [101]. Tento parametr však není v podstatě měřitelný zpětně na hotovém zdrojovém kódu a je nutné již během konkrétního projektu přesně vykázat míru pracnosti implementace pro každý funkční bod. Totožná aplikace, vyvíjená různými přístupy, představuje stále totožnou množinu funkčních bodů, které je nutno implementovat. Z toho důvodu může tato metrika poskytnout výsledky pro přesnější časový i finanční odhad.

5.3 Charakteristika a srovnání základních metod tvorby uživatelského rozhraní v rámci WHMA

Kvalita návrhu a realizace uživatelského rozhraní je v rámci mobilních aplikací jedním z nejdůležitějších faktorů ovlivňující úspěch či neúspěch výsledného produktu. Je to dáno zejména tím, že mobilní aplikace jsou určeny zpravidla pro širokou veřejnost, jež hodnotí spíše vnější provedení a vzhled, než technické detaily skrývající se v nitru zdrojového kódu. Jak již bylo zmíněno v kapitole 4.2 zabývající se specifiky vývoje pro mobilní zařízení, samotný návrh musí také brát v potaz omezení přenosných zařízení, jako je velikost zobrazovací plochy či dotykové ovládání.

V rámci vývoje webových hybridních mobilních aplikací se pro tvorbu uživatelského rozhraní přirozeně nabízejí technologie HTML, CSS a JavaScript, které jsou interpretovány přímo v nativním internetovém prohlížeči spouštěném uvnitř hlavní programové smyčky nativní mobilní aplikace. Lze se také setkat s přístupy, kdy jsou určité objekty GUI vytvářeny přímo pomocí nativních knihoven koncové platformy. To sebou nese výhodu použití nativních prvků, ale také nevýhodu v podobě většího množství proprietárního kódu na úkor toho společného.

Lze tedy definovat tři základní metody tvorby GUI: Tvorba vlastního GUI pomocí HTML5 technologií, použití HTML5 UI frameworku či využití kombinace HTML5 UI s nativními prvky.

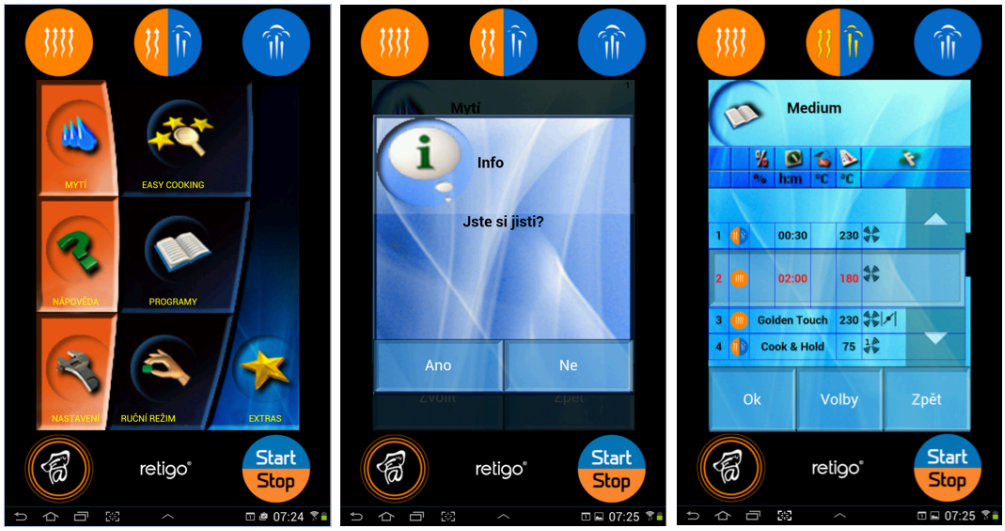
5.3.1 Tvorba vlastního HTML5 GUI

Tato metoda je založena na procesu kódování vlastního GUI dle specifického návrhu pomocí technologií HTML5. Obecně ji lze doporučit pouze pro aplikace s nestandardním uživatelským rozraním, jako jsou například hry, marketingové či propagační aplikace nebo další druhy aplikací, kde se vyžaduje specifický vzhled, který neodpovídá směrnicím koncové platformy.

Použití nestandardního vzhledu pro koncovou platformu však musí být odůvodněné a i tak se lze v případě společnosti Apple Inc. setkat se zamítnutím zveřejnění aplikace na oficiálním distribučním kanálu AppStore z důvodu nesplnění bodu 10. směrnice pro revize aplikací [102], který se týká uživatelského rozhraní. Nestandardní prvky pak zpravidla porušují body 10.1 (nekompatibilita s Apple iOS Human Interface Guidelines) a 10.3. (nesprávné použití standardních navigačních a ovládacích prvků).

Při procesu publikace marketingové mobilní aplikace ClubCube v roce 2013 (vyvíjena v rámci ústavu Informatiky a umělé inteligence na UTB ve Zlíně pro firmu MBankers s.r.o.) jsme se v praxi setkali s ovlivněním procesu revize aplikace lidským faktorem. U společnosti Apple Inc. je krom automatizované kontroly aplikací navíc prováděna revize pracovníkem. K aplikaci samotné byl třetí stranou dodán návrh vzhledu, který v několika bodech kolidoval s oficiální směrnicí společnosti Apple Inc. pro revize aplikací. Vzhled byl jednotný pro platformy Android, iOS i Windows Phone. Přes upozornění zákazníka jsme byli požádání o publikaci aplikace dle dodaného návrhu. Prvotní požadavek o zveřejnění byl schválen a aplikace byla na App Store zveřejněna. Nicméně při pokusu o zveřejnění aktualizace, byl revizní proces proveden jiným pracovníkem, který poukázal na kolidující body směrnice a aplikace byla zamítnuta, dokud nebyla problematická místa uživatelského prostředí dána do souladu s uvedenou směrnicí.

V roce 2014 pak byla v rámci spolupráce ústavu Informatiky a umělé inteligence na UTB ve Zlíně a firmy Retigo s.r.o. publikována aplikace Retigo Vision na App Store [103] a Google Play [104]. Jde o simulátor konvektomatu Retigo Blue, který disponuje dotykovým ovládáním, a může tak sloužit pro prezentační a výukové účely (viz obrázek 16). Prostředí simulátoru věrně kopíruje reálný produkt a tudíž opět není ve shodě s oficiálními směrnicemi a dokonce i základní navigace v rámci aplikace simulátoru neodpovídá uživatelskému zážitku z aplikace běžné na koncové mobilní platformě. K žádosti o publikaci na App Store tak musel být připojen popis aplikace s informací, že se jedná o simulátor reálného produktu. Díky tomu proběhly bez komplikací jak prvotní recenzní řízení, tak i publikace následných několika aktualizací.



Obrázek 16: Ukázka obrazovek simulátoru konvektomatu Retigo v rámci aplikace Retigo Vision 1.2

Z výše uvedeného lze říci, že tvorbu vlastního uživatelského prostředí lze doporučit pouze v odůvodněných případech. Zejména publikace na platformě Apple může být problematická, kvůli subjektivnímu hodnocení recenzenta.

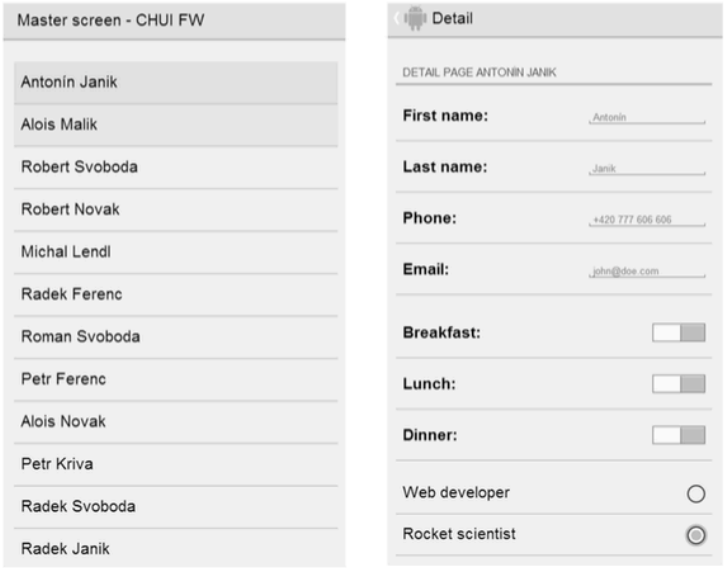
5.3.2 GUI pomocí HTML5 UI frameworku

Realizace vzhledu mobilní aplikace pomocí HTML5 technologií může skýtat celou řadu výhod. Jsou to například nižší nároky na know-how vývojářů, kdy není nutná znalost nativních UI knihoven jednotlivých mobilních platforem. Vše lze realizovat díky znalosti HTML, CSS a jazyka JavaScript, což jsou navíc velmi rozšířené kompetence na trhu práce a náklady na kodéra s uvedenými znalostmi jsou podstatně nižší než v případě nativních technologií.

Snaha vytvořit pomocí HTML5 technologií nativní uživatelský zážitek však může vývoj značně zkomplikovat a zdražit, zejména pokud vývojáři nepoužívají žádné podpůrné knihovny. Příprava jednotlivých objektů uživatelského rozhraní, jako jsou tlačítka či navigační lišty a především jejich odladění v různých verzích mobilních prohlížečů, je časově velmi náročná záležitost. Z toho důvodu lze pro návrh uživatelského rozhraní doporučit některý z mnoha HTML5 UI frameworků, z nichž některé imitují vzhled nejpoužívanějších mobilních operačních systémů a jsou zpravidla odladěné pro přední mobilní operační systémy a jejich nativní prohlížeče.

Tvorbu vlastního uživatelského prostředí tedy nelze doporučit v případě, kdy má aplikace používat standardní prvky uživatelského rozhraní koncové platformy a v podstatě tak napodobovat nativní rozhraní. Typickým příkladem jsou aplikace s grafickým rozložením typu

„master-detail“ (viz obrázek 17), kdy na první obrazovce je zpravidla posuvný seznam s možností „prokliku“ z jednotlivých položek na jejich detailní pohled. Stylování takových objektů je pak časově náročné a zvyšuje náklady jak na kodéra i následné testování, tak na opravu zobrazovacích chyb. V uvedeném případě lze spíše doporučit využití některého z HTML5 UI frameworků, který poskytuje sadu již předpřipravených objektů uživatelského rozhraní. Některé dokonce napodobují nativní vzhled pro více platforem.



Obrázek 17: Ukázka hlavní a detailní obrazovky grafického rozložení „master-detail“

5.3.3 GUI tvořené kombinací HTML5 a nativních UI objektů

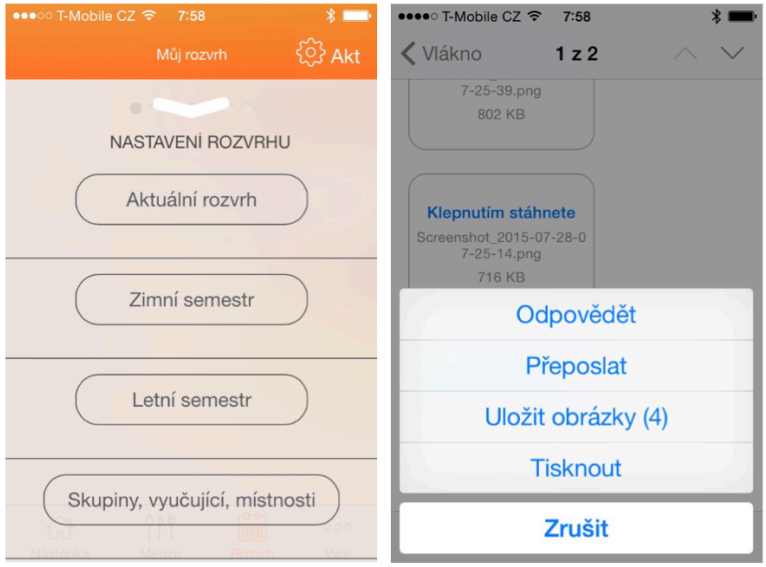
Jako nejvhodnější metoda návrhu uživatelského rozhraní se v praxi jeví kombinace HTML5 technologií a nativních UI objektů. Použití nativních objektů sice zvyšuje procento proprietárního kódu v rámci projektu, ale pokud jsou například využity pluginy wrapper technologie Apache Cordova či Phonegap, se kterými je možné komunikovat pomocí společného API v jazyce JavaScript, odpadá nutnost jakkoliv zasahovat do nativní části kódu. Výhoda nativních objektů pak tkví zejména v poskytnutí nativního zážitku při interakci s uživatelem a také nativního výkonu například při vykreslování animací.

Ukázka (viz obrázek 18) srovnává vykreslení nativního objektu TimePicker, který je volán pomocí pluginu DatePicker dostupného pro wrapper technologii Apache Cordova či Phonegap. Obrázek postupně ilustruje vykreslení objektu na mobilních zařízeních s platformami iOS 8.1, Android 2.3.7 a Android 4.2.2. Z obrázku je zřejmé, že je vždy použit nativní objekt dané verze

operačního systému, což je výhodné, protože se uživatel setkává s objekty uživatelského rozhraní, které jsou mu známé a na jejichž chování či ovládání je zvyklý.

Obrázek 18: Vykreslení nativního TimePicker objektu (výběr času) pomocí pluginu DatePicker pro Apache Cordova/Phonegap na platformách iOS 8.1, Android 2.3.7 a Android 4.2.2

Obrázek 19 naopak ilustruje imitaci nativního objektu Action Sheet [105] (list s dalšími volbami), který je specifický pro platformu iOS, pomocí frameworku Chocolate Chip UI [106]. Animace zobrazení či skrytí listu a chování v případě volby jsou v podstatě totožné s nativním objektem, v ukázce je pouze zvoleno odlišné stylování tlačítek v případě pravé obrazovky (mobilní aplikace UTB [90]). Vlevo je pak nativní Action Sheet v rámci aplikace Mail na platformě iOS. Přestože vzhled tlačítek není identický, nevnímají uživatelé ovládací prvek jako matoucí, jelikož způsob jeho zobrazení i manipulace s ním odpovídá nativnímu objektu.



Obrázek 19: Ukázka objektu Action Sheet (vlevo realizace pomocí HTML5, vpravo nativní objekt)

5.3.4 Dílčí závěr – vlastnosti a srovnání metod

Tabulka 13 uvádí rozsah minimálních znalostí vývojového týmu, jež jsou nutné pro použití konkrétní metody implementace uživatelského prostředí aplikace. Srovnání obsahuje rozpis nutné znalosti technologií a také nástrojů pro ladění aplikací.

Tabulka 13: *Srovnání znalostí nutných pro použití jednotlivých metod implementace UI*

	Technologie	Ladicí nástroje
Vlastní HTML5 UI	HTML5	Google Chrome Developer Tools
HTML5 UI pomocí FW	HTML5 + FW API	Google Chrome Developer Tools
HTML5 (FW) + nativní UI	HTML5 (FW API) + Native Plugin API	Google Chrome Developer Tools + SDK koncové platformy včetně nástrojů pro ladění

Tabulka 14 srovnává vhodnost jednotlivých metod implementace uživatelského prostředí pro konkrétní aplikace, jejichž vzhled odpovídá standardním vzorům daným směrnicí pro návrh vzhledu konkrétní platformy a těch, jejichž vzhled je specifický.

Tabulka 14: *Srovnání vhodnosti metod implementace UI pro aplikace se standardním a specifickým UI*

	Standardní UI	Specifické UI
Vlastní HTML5 UI	nevhodné	vhodné
HTML5 UI pomocí FW	vhodné	nevhodné
HTML5 (FW) + nativní UI	velmi vhodné	nevhodné

5.4 Návrh a realizace multikriteriální analýzy vlastností UI frameworků

Cílem následující kapitoly je sestavení sady kritérií, které mohou sloužit při rozhodování, který z mnoha dnešních dostupných UI frameworků bude vhodný pro použití při tvorbě konkrétní aplikace v rámci konkrétního vývojového týmu.

Přínosem této dílčí části není pouze samotné sestavení sady kritérií, které mohou být dále použity v rozhodovacích procesech týkajících se vývojových frameworků obecně. Různé vývojářské týmy mohou využít také v rámci této práce vytvořené multikriteriální analýzy a pouhým přenastavením vah získat výsledky odpovídající potřebám vlastního vývojářského týmu.

Multikriteriální analýza byla provedena pro následující webové HTML5 UI frameworky:

1) Intel App Framework (http://app-framework-software.intel.com/)
2) Emy (http://www.emy-library.org/)
3) ChocolateChip-UI (http://chocolatechip-ui.com/)
4) jQTouch (http://jqtjs.com/)
5) jQuery Mobile (http://jquerymobile.com/)
6) PhoneJS (http://phonejs.devexpress.com/)
7) TopCoat (http://topcoat.io/)

V tabulce 15 jsou uvedeny konkrétní verze, jež byly použity v rámci aplikovaného výzkumu probíhajícího v období květen–červenec 2014.

Tabulka 15: *Jednotlivé verze frameworků použitých pro multikriteriální analýzu*

FW Name	version
Emy	v1.0
ChocolateChip-UI	v3.5.5
Intel App FW	2.1.0
jQTouch	v0.99.4rc9
jQuery Mobile	1.4.2
PhoneJS	13.2.9
TopCoat	v0.8.0

5.4.1 Rozhodovací kritéria

Následující rozhodovací kritéria byla sestavena na základě praktických zkušeností s programováním mobilních aplikací, dále názorů odborníků z komunitních diskuzních fór StackOverflow a Quora a také vědecko-výzkumných publikací zmíněných v kapitole 2.4.

Použitelnost pro vývoj mobilních aplikací (MA)

Kritérium použitelnosti pro vývoj mobilních aplikací zohledňuje, zda je daný UI framework cílen v prvé řadě právě na mobilní zařízení a disponuje-li standardními ovládacími prvky a efekty. Hodnoceno bylo body 1 až 5, kde 1 znamená nejméně vhodné a 5 nejvíce vhodné. Kromě obvyklých prvků uživatelského rozhraní byla hodnocena také vhodnost layoutu pro mobilní zařízení a například také komplexnost zdrojového kódu, konkrétně DOM (Document Object Model). Vzhledem k nižšímu výkonu mobilních zařízení totiž univerzální frameworky (pro mobil i desktop) obvykle vykazují nižší úroveň uživatelského zážitku, zejména kvůli problémům s výkonem a plynulostí animací.

Použitelnost pro vývoj desktopových aplikací (DA)

Kritérium použitelnosti pro vývoj desktopových aplikací se týká zejména univerzálních UI frameworků, které jsou vhodné i pro použití v desktopových aplikacích. Opět je hodnoceno bodovou škálou od 1 do 5, kde 1 je nejméně vhodné. Při hodnocení je bráno v úvahu, zda framework obsahuje ovládací prvky vhodné pro desktop (dialogy, pokročilé formulářové prvky, tlačítka atd.). Kromě toho by však měla být k dispozici i verze UI vhodná pro zobrazení na mobilních zařízeních.

Aktuálnost (A)

Aktuálnost je velmi důležitým kritériem vývojářských nástrojů obecně. Může vypovídat o živosti projektu, rychlosti opravy chyb či reakce na nové verze mobilních operačních systémů. V rámci toho kritéria byly analyzovány informace z GitHub účtů [107] jednotlivých frameworků. Těmito účty disponují zpravidla všechny projekty, které jsou Open Source. Hodnocena byla četnost aktualizací za měsíc a datum posledního odeslání kódu (last commit). Framework, který dopadl nejlépe, obdržel 7 bodů a nejméně aktuální framework 1 bod. V případě komerčního projektu PhoneJS nebylo možné tyto informace získat, proto byla zvolena průměrná hodnota 3,5 bodu.

Licence (L)

Licenční politika je také jedním z velmi sledovaných kritérií, které udává, zda je zde možnost komerčního použití bez dalších restrikcí, nebo je nutné zakoupit komerční licenci. Hodnocení je následující: možnost neomezeného komerčního použití – 3 body, dvojí licence pro komerční a nekomerční použití – 2 body, pouze komerční licence – 1 bod.

Dokumentace (D)

Dostupnost a kvalita dokumentace přímo ovlivňuje učící křivku. Subjektivní hodnocení je založeno na vlastních zkušenostech z procesu implementace testovacích aplikací. Nejlepší hodnocení je 5 bodů, nejhorší 1 bod.

Velikost (S)

Kritérium velikosti udává minimální velikost zdrojového kódu frameworku, který musí být použit v rámci projektu. Pro bodové hodnocení bylo vytvořeno 6 klasifikačních tříd, které vyplývají z velikostí zkoumaných frameworků: < 100 kB – 6 bodů, >= 100 kB – 5 bodů, > 200 kB – 4 body, > 500 kB – 3 body, > 1 MB – 2 body, > 2 MB – 1 bod.

Nativní vzhled (NL)

Podpora nativního vzhledu jednotlivých platforem je u většiny projektů žádanou vlastností, ale nemusí být standardem. Nativně vypadající aplikace nabízejí zpravidla lepší uživatelský zážitek, jelikož díky obecně používaným vzorům uživatel interaguje se známým prostředím. Hodnocení je následující: podpora nativního vzhledu nejnovějších verzí alespoň 3 hlavních mobilních platforem (Andorid, iOS, Windows Phone) – 4 body, podpora nativního vzhledu starších verzí alespoň 3 hlavních mobilních platforem (Andorid, iOS, Windows Phone) – 3 body, základní barevná témata pro různé mobilní platformy – 2 body, univerzální vzhled – 1 bod.

Komunita (C)

Zvláště v případě open-source projektů je velmi důležitá velikost a kvalita komunity kolem projektu sdružené. Pokud je komunita dostatečně velká, dá se usuzovat, že projekt se bude dostatečně rychle rozvíjet, implementovat nové vlastnosti, opravovat dostatečně rychle chyby a že bude možné získat potřebné odpovědi na komunitním fóru. Velikost komunity byla zjištěna z GitHub účtů Open Source projektů. Na základě údajů jako je počet přispívatelů s alespoň 50ti příspěvky zdrojového kódu a počtu chyb hlášených v posledních 30 dnech, byli jednotliví kandidáti seřazeni a obodováni. 7 bodů je nejlepší a 1 bod nejhorší výsledek. Vzhledem k tomu, že u komerčního projektu PhoneJS nebyla tato data dostupná, opět byla zvolena průměrná hodnota.

5.4.2 Multikriteriální matice

Na základě evaluace výše uvedených kritérií byla sestavena multikriteriální matice pro hodnocené kandidáty (viz tabulka 16) a následně byla matice upravena do normalizovaného tvaru (viz tabulka 0).

Tabulka 16: *Multikriteriální matice pro evaluaci vlastností UI frameworků*

Framework	MA	DA	A	L	D	S	NL	C
Emy	5	1	2	3	4	6	4	2
ChocolateChip-UI	5	1	7	3	4	5	5	5
Intel App FW	5	1	5	3	4	4	4	6
jQTouch	5	1	4	3	3	6	2	5
jQuery Mobile	4	2	6	3	5	3	3	7
PhoneJS	5	1	3.5	2	4	1	5	3.5
TopCoat	4	4	1	3	3	5	1	3

Tabulka 17: *Normalizovaná multikriteriální matice pro evaluaci vlastností UI frameworků*

Framework	MA	DA	A	L	D	S	NL	C
Emy	1.00	0.00	0.17	1.00	0.75	1.00	0.75	0.00
ChocolateChip-UI	1.00	0.00	1.00	1.00	0.75	0.80	1.00	0.60
Intel App FW	1.00	0.00	0.67	1.00	0.75	0.60	0.75	0.80
jQTouch	1.00	0.00	0.50	1.00	0.50	1.00	0.25	0.60
jQuery Mobile	0.75	0.25	0.83	1.00	1.00	0.40	0.50	1.00
PhoneJS	1.00	0.00	0.42	0.50	0.75	0.00	1.00	0.30
TopCoat	0.75	0.75	0.00	1.00	0.50	0.80	0.00	0.20

Při procesu hodnocení zpravidla nemají všechna kritéria stejnou důležitost. Ta se může lišit v závislosti na konkrétním vývojovém týmu či konkrétní aplikaci a je vhodné ji popsat pomocí vah. Ty pak mohou být jednoduše změněny za účelem získání relevantních výsledků.

V rámci provedené evaluace byly zvoleny váhy, jež vycházejí z expertního hodnocení 4 vývojářů se zkušenostmi z oblasti mobilních technologií. Hodnocení kritérií probíhalo pomocí bodování následovně:

Mějme p kritérií a q expertů. Kritéria jsou hodnocena body p, $p - 1$, …, 1. Nejdůležitější kritérium je hodnoceno číslem p, a nejméně důležité číslem 1. Tabulka 18 znázorňuje výsledné hodnocení expertů.

Tabulka 18: *Expertní hodnocení kritérií*

	MA	DA	A	L	D	S	NL	C
Expert 1	7	1	6	2	4	5	8	3
Expert 2	8	1	3	7	6	2	5	4
Expert 3	7	6	5	8	4	2	1	3
Expert 4	7	6	8	2	4	1	3	5

Celkové váhy jsou pak z výsledného hodnocení expertů vypočítány následovně:

Jestliže a_{ij} je hodnocení i-tého kritéria j-tým expertem, pak váha i-tého kritéria dle j-tého experta (w_{ij}) je vypočítána pomocí (1). Výsledná celková váha i-tého kritéria (w_i) pak podle (2).

$$w_{ij} = \frac{a_{ij}}{\sum_{i=1}^{p} a_{ij}} = \frac{a_{ij}}{\frac{p(p+1)}{2}} \qquad (1)$$

$$w_i = \frac{\sum_{j=1}^{q} w_{ij}}{q} = \frac{\sum_{i=1}^{p} a_{ij}}{\frac{p(p+1)q}{2}} \qquad (2)$$

Výsledky (2) – expertní hodnocení vah, jsou uvedeny v tabulce 19. Nejdůležitějšími kritérii jsou pak použitelnost pro vývoj mobilních aplikací (MA), aktuálnost (A), licence (L) a dokumentace (D). Jako nejméně důležité pak z výše uvedeného expertního hodnocení vychází velikost (S), použitelnost pro vývoj desktopových aplikací (DA) a komunita (C).

Tabulka 19: *Výsledné váhy multikriteriální matice získané na základě expertního hodnocení*

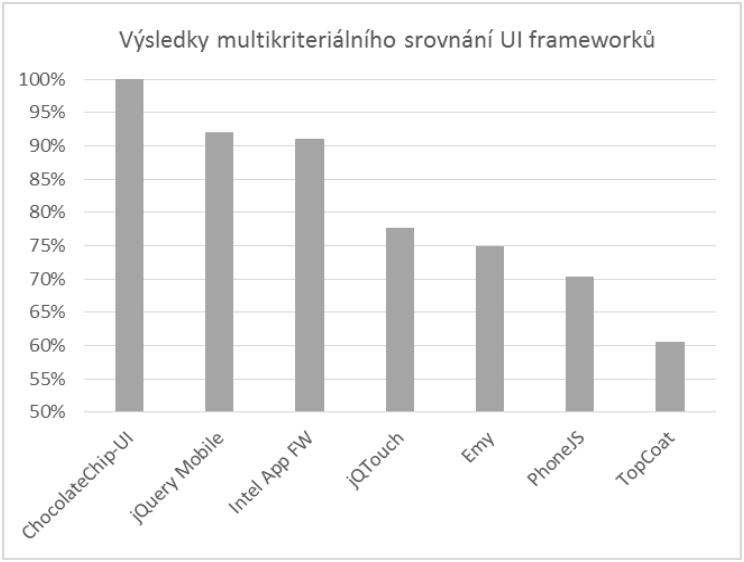
Kritérium	MA	DA	A	L	D	S	NL	C
Váha	0.20	0.10	0.15	0.13	0.13	0.07	0.12	0.10

5.4.3 Výsledky vážené multikriteriální analýzy

Z normalizované matice (viz tabulka 0) a vah (viz tabulka 19) byly vypočteny výsledky, viz tabulka 20. Nejúspěšnějším kandidátem se stal Chocolate Chip UI framework a nejméně úspěšným pak TopCoat. V grafu (viz obrázek 20) jsou pak vizualizovány výsledky dle procentuální úspěšnosti.

Tabulka 20: *Výsledná multikriteriální matice hodnocení UI frameworků*

Framework	body	procenta
ChocolateChip-UI	0.82	100%
jQuery Mobile	0.75	92%
Intel App FW	0.74	91%
jQTouch	0.63	78%
Emy	0.61	75%
PhoneJS	0.57	70%
TopCoat	0.49	61%



Obrázek 20: Výsledky multikriteriální srovnání UI frameworků

Vítězný framework Chocolate Chip UI získal dobré hodnocení zejména v kritériích použitelnosti pro vývoj mobilních aplikací (MA), aktuálnosti (A), dokumentace (D), velikosti (S) a nativního vzhledu (NL). Mezi jeho výhody patří aktivní komunita, jež se stará o potřebnou aktualizaci, která tak reaguje velmi rychle na příchod a podporu nových verzí mobilních operačních systémů. Je také velmi vhodný pro návrh UI standardních mobilních aplikací, jelikož obsahuje objekty uživatelského rozhraní, které imitují nativní komponenty. Co se týče velikosti zdrojových kódů, vejdou se nezbytně nutné soubory do 100 kB a díky nekomplikované struktuře DOM (Dom Object Model) se tak řadí mezi nejrychlejší a nejvýkonnější frameworky.

Na druhém místě se při výše uvedeném nastavení vah umístil jQuery Mobile framework, a to zejména díky nejrozsáhlejší komunitě, vysoké míře aktuálnosti a velmi dobře zpracované dokumentaci. Dle praktických zkušeností z implementace je však nutné upozornit na velkou komplexnost DOM, což na mobilních zařízeních může způsobovat výkonnostní a zobrazovací problémy.

Třetí místo pak obsadil Intel App framework, který má za sebou také silnou komunitu vývojářů, dále je velmi aktuální a obsahuje potřebné UI prvky pro návrh mobilních aplikací. Ty však nedosahují vizuální kvality frameworku Chocolate Chip UI. Intel App framework je také komplexnější a obsáhlejší, přesto však oproti například jQuery Mobile nabízí velmi dobrý výkon.

Výhodou zmíněného přístupu hodnocení je možnost jednoduché změny vah a v případě potřeby přizpůsobení konkrétnímu projektu. Lze totiž říci, že neexistuje univerzální sada vah pro

jednotlivá kritéria, která by nejlépe odpovídala všem projektům pro vývoj mobilních aplikací. V kontextu hybridních mobilních aplikací tvořených pomocí webových technologií však lze doporučit několik kritérií, na které by měl být kladen větší důraz.

5.4.4 Přínosy realizované multikriteriální analýzy

V současné době lze nalézt v rámci zahraničních internetových zdrojů několik dílčích srovnání z oblasti hybridních UI frameworků. Tato srovnání však zpravidla berou v úvahu 2 až 3 technologie a omezují se pouze na základní vlastnosti publikované jejich tvůrci.

Provedená multikriteriální analýza však definuje sadu kritérií (kapitola 5.4.1), která popisují klíčové vlastnosti preferované vývojáři mobilních technologií. Hodnocení jsou tedy reprodukovatelná a díky možnosti změny vah aplikovatelná na konkrétní projekty.

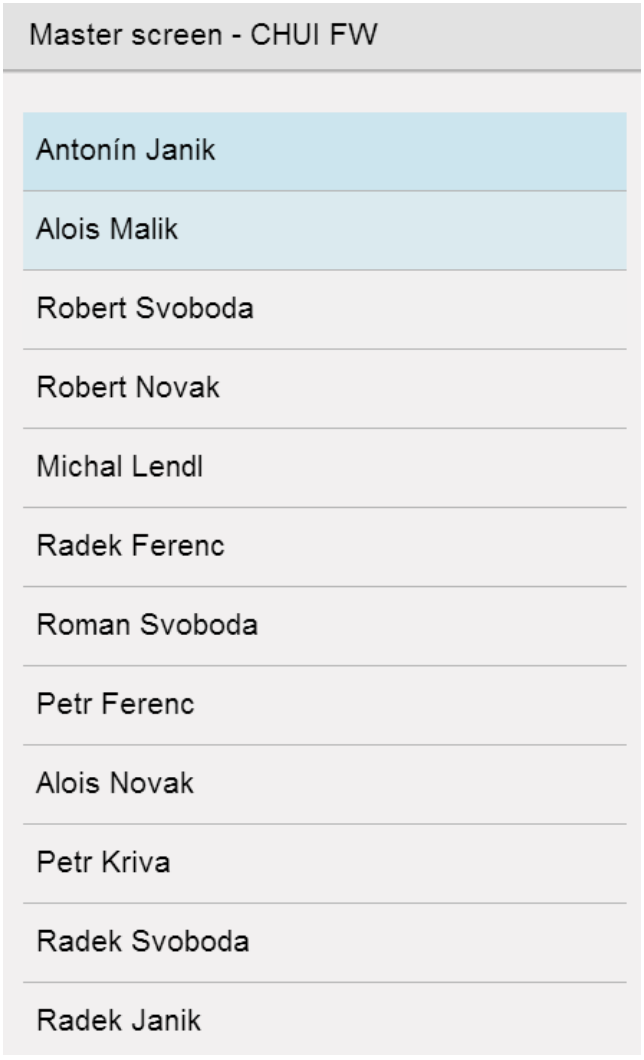
V rámci multikriteriální analýzy byly porovnány aktuální verze (k létu 2014) vybraných HTML5 frameworků a získané výsledky mohou posloužit jako znalostní báze široké oblasti vývojářů. Hodnocení bylo provedeno na základě dat, jež byla získána empiricky při procesu implementace testovacích aplikací a dále analýzou internetových zdrojů, jako je komunitní web GitHub (pro sdílení zdrojových kódů) [107] či webové stránky s dokumentací jednotlivých frameworků.

5.5 Návrh a realizace výkonnostních testů UI frameworků

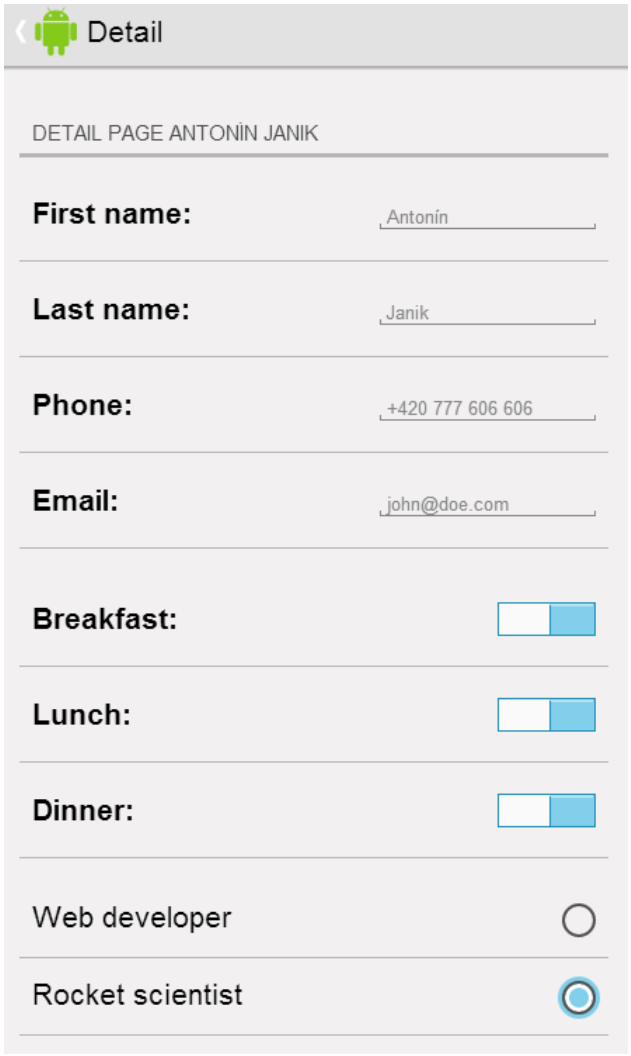
Většina dostupných srovnání UI frameworků bere v úvahu pouze jejich základní vlastnosti, jako je dostupnost standardních grafických objektů, podporu nativního vzhledu, možnost úpravy grafického tématu, možnost napojení na některou z „back-end“ technologií apod. Jeden z nejdůležitějších parametrů, zejména ve světě hybridních mobilních aplikací, je kvalita implementace ovlivňující výkon daného frameworku. Mezi jednotlivými kandidáty srovnání se objevují velké rozdíly a přitom právě výkon aplikace má přímý dopad na kvalitu uživatelského zážitku, který je základním faktorem úspěchu či neúspěchu aplikace. Porovnání výkonu na úrovni nativní/hybridní aplikace tvořené pomocí 4 odlišných technologií je publikováno v [28]. Autoři však srovnávají pouze dvě aplikace tvořené pomocí frameworku pro návrh hybridní mobilní aplikace a jako parametry měření používají paměť zabranou aplikací a vytížení procesoru. Tyto údaje však v podstatě nic nevypovídají o kvalitě uživatelského zážitku při interakci s aplikací a její plynulosti. V rámci výkonnostních testů prezentovaných v této práci proto byla zvolena odlišná metodika, viz kapitola 5.5.3. Pro účely testování byla pomocí každé testované technologie naprogramována funkčně totožná testovací aplikace, detailněji popsaná v 5.5.1.

5.5.1 Popis testovací aplikace

Za účelem měření relevantních dat pro testování výkonu byla pomocí každého z vybraných UI frameworků implementována testovací aplikace. Byla zvolena typická architektura často používaného vzoru s hlavní výčtovou obrazovkou (list) a obrazovkou s detailními informacemi jednotlivých položek (detail). List byl realizován jako seznam 100 položek. Položky byly asynchronně načítány z externího JSON souboru. Detail pak obsahuje základní formulářové prvky, aby byla vhodně simulována reálná aplikace. Formulářová pole jsou při přechodu z listu na detail nastavována konkrétními daty náležícími k vybrané položce.



Obrázek 21: Pohled na hlavní obrazovku se seznamem (Chocolate Chip UI Framework)



Obrázek 22: Pohled na obrazovku s detailními informacemi (Chocolate Chip UI Framework)

5.5.2 Použitý hardware

Výkon testovacích aplikací byl měřen na tabletu Samsung Galaxy Note 10.1 (GT-N8010) s operačním systémem Android ve verzi 4.1.2. Jako běhové prostředí aplikace byla využita aplikace Chrome ve verzi 35.0.1916.138, jelikož disponuje vývojářskými nástroji pro vzdálené ladění.

5.5.3 Metody výkonnostních testů

Provedené výkonnostní testy se zaměřují na parametry, jež vypovídají o kvalitě interakce s mobilní aplikací neboli uživatelském zážitku. Jsou to tzv. DOM (Document Object Model)

Load testy, které se zaměřují na rychlost prvotního načtení aplikace, dále testy plynulosti skrolování a přechodů stránek.

Relevantní veličiny, kterými jsou časy událostí DOM Load a počet snímků za sekundu (FPS) u testů plynulosti, byly měřeny v rámci běhového prostředí pomocí nástrojů Chrome Remote Debugging [108] a Chrome Developer Tools [109].

DOM Load testy a dílčí aktivity načítání

Rychlost prvotního spuštění aplikace je důležitým parametrem, jenž může značně ovlivnit procento lidí, kteří u dalšího využívání aplikace zůstanou. Dle průzkumu publikovaného v [110] čtyři z pěti uživatelů očekávají, že mobilní aplikace bude startovat 3 sekundy a méně. V případě hybridních mobilních aplikací lze aplikaci zpřístupnit uživateli až po té, co je kompletně sestaven DOM. Spuštění aplikace bude tedy představovat minimálně čas od prvotního požadavku do doby, než v běhovém prostředí (prohlížeči) dojde k vyvolání události Load. Tu lze použít, na rozdíl od události DOMContentLoaded, jež detekuje pouze připravenost Dom Object Modelu (DOM), pro detekci plně načtené webové aplikace, včetně obrázků a dalších externích zdrojů. [111]

Měření probíhalo vždy pro každou testovací aplikaci na testovacím zařízení a byl odečítán čas spuštění události Load pomocí nástroje Chrome Developer Tools [109]. Pro detailní analýzu byly pomocí tohoto nástroje také odečteny časy jednotlivých aktivit načítání. Jde o aktivity definované v rámci Chrome Developer Tools – Loading (načítání dat a externích zdrojů), Scripting (provádění skriptů), Rendering (příprava stromu dle DOM) a Painting (volání metody kreslení jednotlivých objektů stromu dle DOM). [112]

Testy plynulosti skrolování

Jedním z klíčových faktorů ovlivňujících uživatelský zážitek mobilních aplikací je plynulost animací. Nativní aplikace obvykle nabízejí vysokou míru uživatelského zážitku, jelikož animace jsou zde optimalizované pro běh na koncové platformě. Hybridní aplikace, kde je UI tvořeno pomocí webových technologií, však něco takového nezaručují a proto se mnozí vývojáři potýkají s výkonnostními problémy právě u standardních animací, jako je například skrolování dlouhého seznamu.

V rámci této metody výkonnostního testu je měřena veličina FPS (Frames per Second – snímků za sekundu), jež se běžně používá v herním průmyslu. Obecně se při tvorbě animací doporučuje použití frekvence 30 FPS, kterou lidské oko vnímá jako plynulou. Video formáty dokonce běžně používají nižší frekvence v rozmezí 23,976 – 30 FPS [113], aniž by byla zaznamenána ztráta kvality. Hodnota při animacích uživatelského prostředí by se tedy měla ideálně pohybovat okolo 30 FPS. V praxi však pro kvalitní uživatelský zážitek postačí rozmezí

24–30 FPS. Hodnota 24 FPS je v [114] uvedena jako ještě stále dostatečná pro vnímání plynulé animace.

Měření probíhalo vždy pro každou testovací aplikaci na testovacím zařízení a byly odečítány minimální, maximální a průměrné hodnoty FPS při sledovaných animacích pomocí nástroje Chrome Developer Tools. Průměrná hodnota FPS orientačně vypovídá o celkové plynulosti animace. Maximální hodnota FPS není relevantní, pokud převyšuje 60 FPS, což je většinou maximální hodnota frekvence snímků mobilních zařízení omezená hardwarem. Nejdůležitějším ukazatelem je minimální hodnota FPS, která pokud je nižší než 24 FPS, znamená sníženou kvalitu uživatelského zážitku. Projevuje se zpravidla určitým trháním či neplynulostí sledované animace.

Testy plynulosti přechodů stránek

Mezi nejčastější animace v mobilních zařízeních patří přechody jednotlivých stránek (obrazovek) aplikace. Metoda měření je v podstatě totožná s testy plynulosti skrolování. Opět je měřenou veličinou FPS přechodu mezi obrazovkami.

5.5.4 Výsledky výkonnostních testů

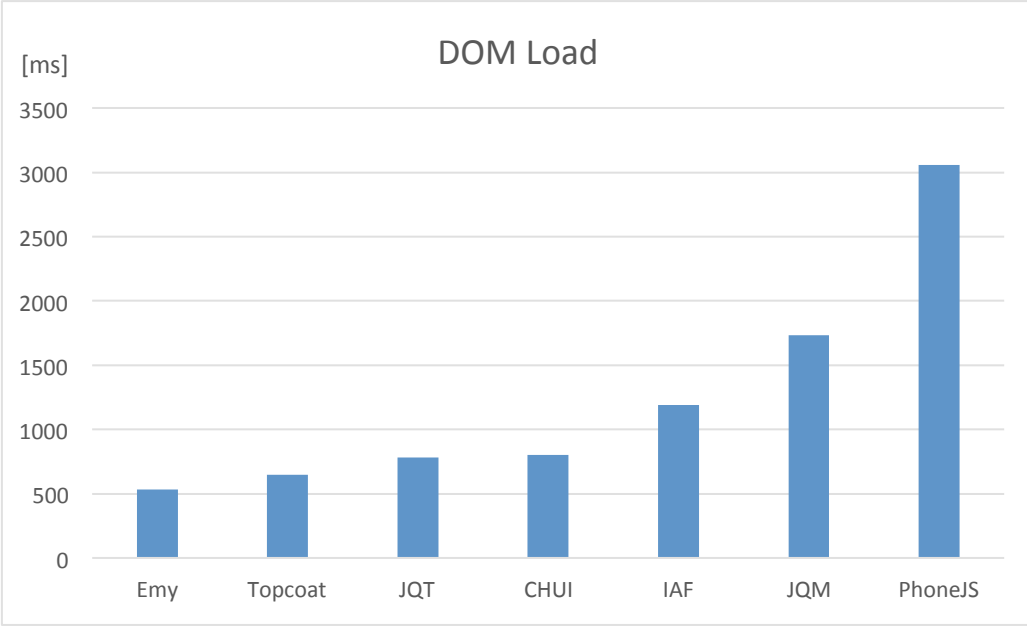
Níže uvedené výsledky výkonnostních testů nejsou zajímavé z hlediska absolutních čísel, jelikož jsou vztaženy ke konkrétnímu testovacímu hardware a také konkrétní verzi běhového prostředí Chrome. Pro účely výkonnostního srovnání testovaných frameworků jsou však naměřené hodnoty relevantní.

DOM Load testy a dílčí aktivity načítání

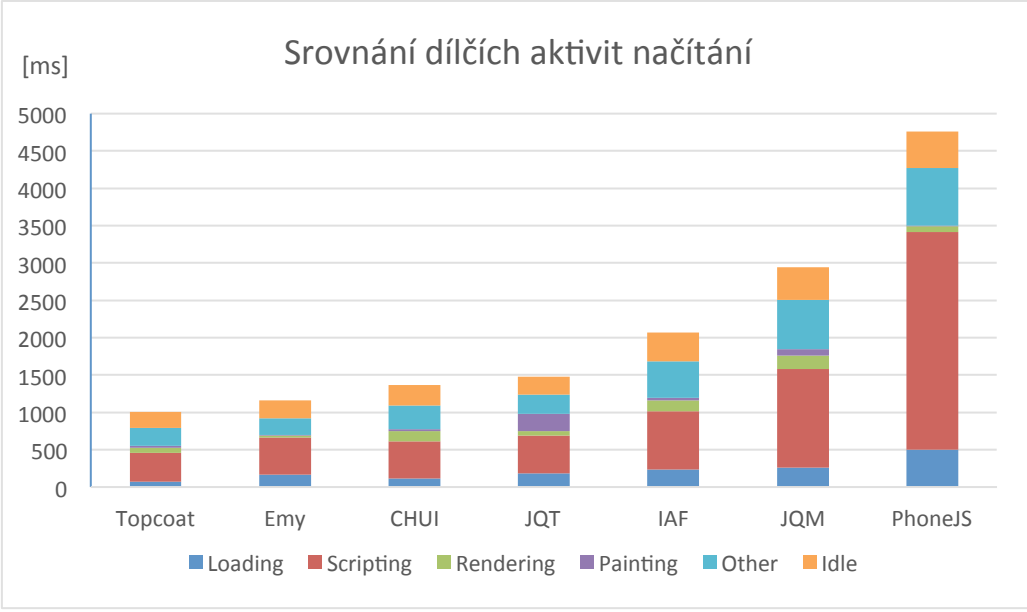
Výsledky DOM Load testů (test rychlosti prvotního načtení aplikace viz obrázek 23) ukazují poměrně velké rozdíly mezi aplikacemi používajícími různé frameworky. Ideální jsou výsledky pod 1000 ms, kam se zařadily 4 vývojové UI frameworky, jež mají také poměrně malou datovou náročnost. Intel App Framework je středně komplexním řešením s ještě akceptovatelnou rychlostí spuštění aplikace. V praxi lze také velmi často narazit na jQuery Mobile Framework, který však kvůli své velmi komplexní architektuře a složitosti DOM neposkytuje vysokou míru výkonu. Překvapivě špatně dopadl v tomto testu komerční framework PhoneJS, jehož spuštění zabralo více než 3 sekundy. Jde sice o jedno z nejkomplexnějších řešení, kde již většinou není nutné importovat další knihovny pro dosažení žádané funkcionality, ale přesto při větším množství dat v aplikaci může narůst výsledný čas spouštění i nad 5 sekund.

Dílčí aktivity načítání jsou srovnány na obrázku 24. Z výsledků je zřejmé, že dokument může být načten v takřka polovičním čase, než v součtu zaberou jednotlivé dílčí aktivity načítání. Je to dáno zejména asynchronním načítání externích zdrojů, jako jsou kaskádové styly, JavaScript či

webové fonty. Z měření dále vyplývá, že celkový čas načítání je nejvíce ovlivněn dílčím časem provádění skriptů.



Obrázek 23: Srovnání rychlosti DOM Load



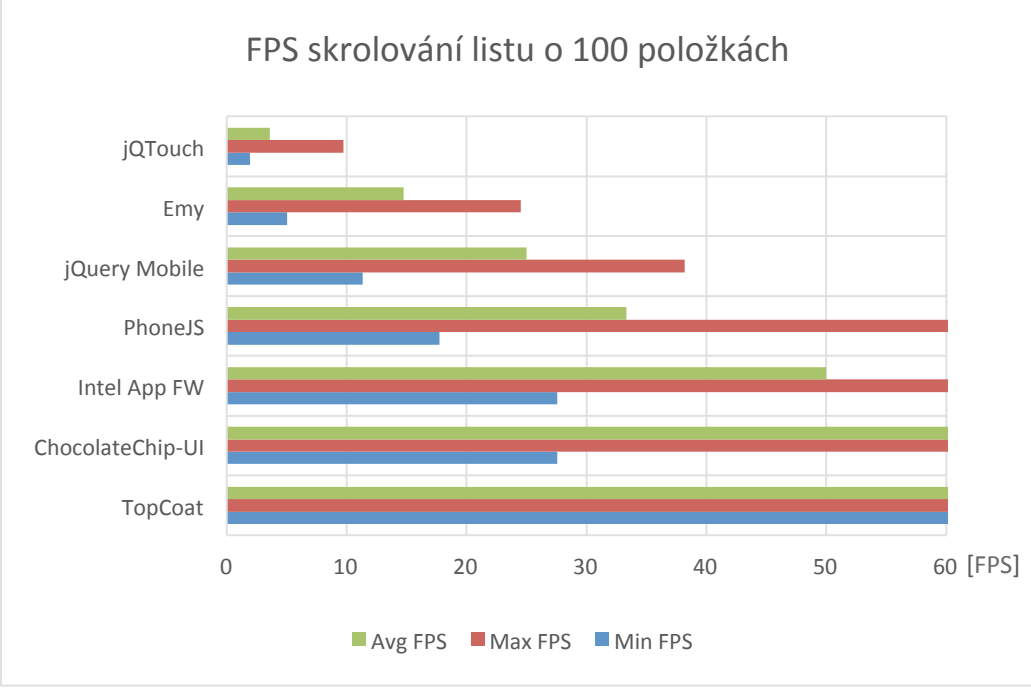
Obrázek 24: Srovnání dílčích aktivit načítání

Testy plynulosti skrolování

Pro výsledky testů plynulosti skrolování jsou nejdůležitějšími veličinami průměrná a minimální hodnota FPS. Vizualizace výsledků (viz obrázek 25) byla omezena na maximální frekvenci 60 FPS, vyšší hodnoty už nejsou z hlediska současných mobilních zařízení relevantní.

V testu se prokázalo, že nejlepší odezvu mají frameworky s co nejjednodušší strukturou DOM a minimálním prováděním skriptů. Ve všech sledovaných hodnotách FPS převyšoval hranici 60 FPS UI framework Topcoat, který je však čistě HTML&CSS frameworkem bez použití technologie JavaScript a nabízí pouze jednoduchý list bez možnosti pokročilého stylování. Velmi dobře se dále umístily frameworky Chocolate Chip UI a Intel App Framework, kdy jejich minimální FPS stále převyšovala žádanou hranici 24 snímků za sekundu.

Pro ostatní frameworky se množství 100 položek v listu ukázalo jako příliš velké a minimální hodnota FPS nižší než 24 snímků způsobovala viditelně neplynulou a trhanou animaci skrolování.

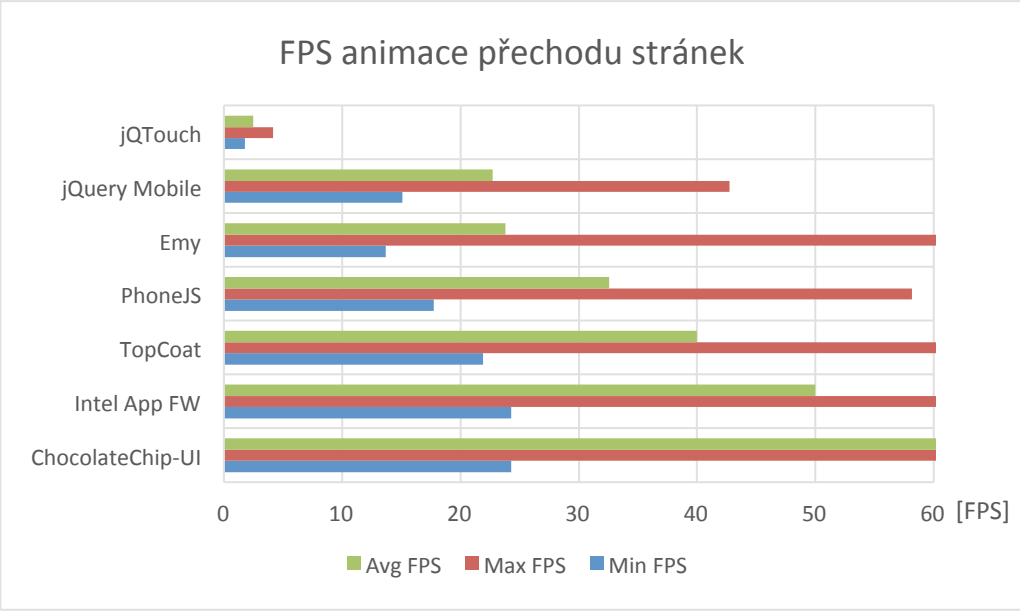


Obrázek 25: FPS skrolování listu o 100 položkách

Testy plynulosti přechodů stránek

Průměrné hodnoty FPS u přechodů mezi stránkami (obrazovkami) aplikace se kromě frameworku jQTouch, Emy a jQuery Mobile pohybovaly nad hranicí 30 FPS, což je velmi dobrý

výsledek vzhledem k tomu, že během animací bylo nutné manipulovat s dlouhým listem o 100 položkách. Přesto však plynulost animace po celou dobu jejího běhu vykazovaly pouze frameworky Chocolate Chip UI a Intel App Framework, jejichž minimální FPS se pohybovala těsně u hranice 24 snímků za sekundu. U ostatních zástupců bylo možno pozorovat neplynulost přechodu ve velmi malé či větší míře.



Obrázek 26: FPS animace přechodu stránek

5.5.5 Přínosy realizovaných výkonnostních testů

Během rešerše současného stavu v oblasti výkonového srovnání frameworků používaných v procesu návrhu mobilních aplikací bylo zjištěno, že v současné době je dostupných jen několik málo vědecko-výzkumných článků, které se soustředí na výkonnostní testování vývojových frameworků z oblasti mobilních aplikací a navíc některé z nich se zaměřují pouze na veličiny, které nemají přímou vypovídací hodnotu o kvalitě uživatelského zážitku. Měření aplikací zabrané paměti či procesorového času tak bylo v této práci nahrazeno přímým měřením počtu snímků za sekundu FPS při standardně používaných animacích, jelikož tato veličina přímo vypovídá o vnímání plynulosti animace a hodnotě uživatelského zážitku.

Relativní výsledky měření jsou pak využitelné v procesu výběru konkrétního frameworku pro tvorbu uživatelského prostředí webové hybridní aplikace.

5.6 Identifikace problémů běhového prostředí Apache Cordova/Phonegap

Během měření výkonnosti UI frameworků viz kapitola 5.5 a testování wrapper technologie Apache Cordova/Phonegap bylo zjištěno, že standardní běhové prostředí (nativním WebView), které je zde použito, vykazuje při reálné implementaci mobilní aplikace celou řadu zobrazovacích chyb. V následujících kapitolách proto bude vysvětlen jeden z nejmarkantnějších problémů webového hybridního vývoje, což je nekonzistence běhových prostředí na reálných zařízeních, a to nejen napříč různými platformami, ale také napříč jednotlivými verzemi konkrétního operačního systému. Jak bylo zjištěno testy provedenými v kapitole 5.7, nejvíce se tato nekonzistence projevuje v rozdílné podpoře HTML5 API a CSS3 vlastností, což způsobuje četné zobrazovací problémy, které se vážou ke konkrétní verzi nativního WebView. Nezanedbatelné jsou i rozdíly ve výkonu JavaScriptu a grafických animací. Kapitola se tedy zaměřuje na reálné komplikace, které často vyvolají zvýšené náklady na testovací fázi i na opravy nalezených chyb, zkoumá jejich původ a přináší návrhy řešení.

5.6.1 Charakteristika standardního běhového prostředí Apache Cordova/Phonegap

Jak již bylo popsáno v kapitole 4.5.1, WHMA je v podstatě nativní mobilní aplikací, která místo běžných objektů uživatelského rozhraní spouští v rámci své jediné obrazovky pouze objekt WebView (nativní internetový prohlížeč), který je zobrazen v celoobrazovkovém módu bez jakýchkoliv ovládacích prvků (viz obrázek 13). Úkolem WebView je načíst vstupní soubor webové hybridní aplikace (např. index.html) a zobrazit jeho obsah. Pokud obsah webové části imituje nativní uživatelské rozhraní, běžný uživatel nepozná, že právě spuštěná aplikace není aplikací nativní, nýbrž je konstruována pomocí technologií HTML5.

Samotný WebView je v podstatě objekt instanciovaný na základě standardní třídy v rámci nativního programovacího jazyka koncové platformy. Znamená to tedy, že projekt Apache Cordova/Phonegap nepoužívá vlastní implementaci prohlížeče HTML dokumentů, nýbrž standardní systémový prvek, jehož vlastnosti tedy závisí na konkrétní implementaci v rámci konkrétní verze operačního systému.

V případě platforem Android a Windows Phone je použit objekt nativního SDK shodného názvu WebView, na platformě iOS pak UIWebView.

5.6.2 Nekonzistence vlastností běhového prostředí Apache Cordova/Phonegap

Problém nekonzistence vlastností běhového prostředí znázorňuje tabulka 21. Při adopci metod webového hybridního vývoje mnozí programátoři netuší, že i v rámci mobilní platformy se setkají s rozdílností jednotlivých verzí internetových prohlížečů, která je známa ze světa webových aplikací. Jde pravděpodobně o nejpalčivější problém webového hybridního vývoje. Je

zřejmé, že zařízení s instalovaným operačním systémem Android 2.3.7 disponuje jinou verzí nativního internetového prohlížeče než zařízení s verzí 5.0. Přesto si tento fakt vývojářské týmy začínající s technologií Apache Cordova/Phonegap mnohdy nedávají do souvislosti s interpretací jejich HTML5 zdrojového kódu v rámci nativního balíčku aplikace.

Tabulka 21 sestavená na základě zjištění z implementačních testů blíže popsaných v kapitole 5.7 uvádí ve sloupci V (Varianty) počet různých nehomogenních běhových prostředí, pro které musí vývojáři zajistit bezchybný běh jejich aplikace.

Řádek označený šedě znázorňuje prvotní velmi optimistický předpoklad mnoha vývojářů či vedoucích projektů, kteří nemají hlubší zkušenosti s webovým hybridním vývojem, a sice ten, že pro pokrytí majoritních platforem budou muset aplikační kód přizpůsobit třem nehomogenním běhovým prostředím.

Při reálných implementačních testech se však ukazuje, že v rámci každé z platforem existují moderní a zastaralá, vlastnostmi nekonzistentní běhová prostředí, která se váží k verzi operačního systému. V rámci hrubého dělení můžeme identifikovat následující zlomové verze operačních systémů. V případě platformy Android je to verze < 4.0 a >= 4.0, dále verze <8 a 8 platformy iOS a verze <8 a >=8 platformy Windows Phone. V tomto případě již počítáme s šesti různými nehomogenními běhovými prostředími (viz řádek označený žlutě).

Reálné zkušenosti získané při implementaci mobilních aplikací v rámci projektů inovačních voucherů zlínského kraje (Retigo, ClubCube) však ukázaly, že tento předpoklad je mylný a je třeba vzít v úvahu minimálně 8 značně odlišných běhových prostředí (viz řádek označený žlutě). V rámci platformy Android vykazují signifikantní míru nehomogenity verze nižší než 4.0 (tyto však z důvodu zastaralosti a nízké úrovně bezpečnosti nebudou již v práci dále brány v potaz) a rovněž pak verze 4.0–4.4, které trpí častými zobrazovacími problémy, oproti verzím 4.4 a vyšším, které již implementují moderní vlastnosti HTML5 API. Platforma Windows Phone také vykazuje velké odlišnosti běhových prostředí v rámci verzí 7.5, 8 a 10.

Řádek označený oranžově navíc definuje v rámci platformy Android jednotlivé podverze 4 a 5, z nichž každá poskytuje běhové prostředí s odlišnou podporou HTML5. Zde však rozdílnosti nejsou tak markantní jako v rámci předchozího řádku, nicméně v reálné situaci znamenají zvýšený počet zobrazovacích chyb, které se váží na konkrétní podverzi operačního systému a tím také zvýšené náklady na testování a opravu. Má-li aplikace v praxi fungovat bezchybně, musí být přizpůsobena pro 13 různých běhových zařízení.

Poslední řádek tabulky označený červeně postihuje reálnou situaci, kde není nehomogenita běhového prostředí závislá pouze na verzi operačního systému, ale v případě OS Android také na jeho softwarové nástavbě, kterou dodává výrobce konkrétního mobilního zařízení. Vzniká tak

situace, kdy již není možné přesně odhadnout počet různě se chovajících běhových prostředí reálných zařízení na trhu.

Tabulka 21: *Přehled rozdílných běhových prostředí WHMA v rámci majoritních platforem*

Technologie														**V**
Android								iOS		Windows Phone				**3**
< 4.0					>= 4.0			<8	8	<8		>=8		**6**
< 4.0	<4.4				>=4.4			<8	8	7.5	8		10	**8**
< 4.0	4.0	4.1	4.2	4.3	4.4	5.0	5.1	<8	8	7.5	8		10	**13**
Verze OS + Rozšíření výrobců zařízení								<8	8	7.5	8		10	**N**

S výše uvedeným rizikem je tedy třeba počítat ještě před výběrem vývojové metody, případně později při výběru technologií pro tvorbu uživatelského rozhraní. Proto je překvapivé, že na tento fakt není nikde v dokumentaci projektu Apache Cordova/Phonegap poukázáno. Vysvětlením může být fakt, že tato tzv. wrapper technologie je zde prezentována pouze jako běhové prostředí pro HTML, CSS a JavaScript s možností komunikace s nativními API zařízení. Projekt nijak nedeklaruje standardní podporu HTML5 vlastností napříč platformami a verzemi, ani si neklade za cíl nabízet vlastní knihovny pro tvorbu uživatelského rozhraní, což je právě oblast, kde se nejvíce zobrazovacích problémů projeví.

V rámci kapitoly 5.7 je na základě implementačních testů navrženo řešení pomocí nadstandardního běhového prostředí pro Apache Cordova/Phonegap. Podstatou je minimalizace počtu nehomogenních běhových prostředí, v nichž bude aplikace reálně spouštěna.

5.6.3 Příčiny nekonzistence běhových prostředí platformy Android

Z hlediska nekonzistence běhových prostředí jsou na základě reálných měření a implementačních testů v kapitole 5.6.2 identifikovány klíčové verze operačních systémů, jejichž běhová prostředí se signifikantně liší. Za účelem porozumění naměřeným datům byla detailněji zkoumána architektura běhových prostředí v rámci platformy Android a byly položeny následující otázky, jejichž zodpovězení by mělo vést k pochopení zmíněných příčin nekonzistence:

1. Jaké vykreslovací jádro prohlížeče používá projekt Apache Cordova ve svém standardním běhovém prostředí na platformě Android?
2. Mohou se implementace standardní třídy WebView lišit v rámci odlišných verzi Android OS?
3. Jaké vykreslovací jádro používá standardní prohlížeč Internetu v rámci OS Android (aplikace Internet)?
4. Jaké vykreslovací jádro používá aplikace Chrome?

5. Lze vykreslovací jádra aktualizovat?

Jaké vykreslovací jádro prohlížeče používá projekt Apache Cordova ve svém standardním běhovém prostředí na platformě Android?

Pro zodpovězení této otázky je nutné analyzovat zdrojový kód wrapper technologie Apache Cordova, konkrétně projekt cordova-android publikovaný na portále GitHub.com. Na řádku 26 mezi importy knihoven v souboru cordova-android/framework/src/org/apache/cordova/engine/SystemWebView.java [115] lze nalézt standardní knihovnu WebView z balíčku android.webkit, která implementuje systémové vykreslovací jádro pro webové technologie. Samotná instanciace objektu WebView pak probíhá v rámci konstruktoru třídy SystemWebViewEngine (cordova-android/framework/src/org/apache/cordova/engine/SystemWebViewEngine.java) [116] tvorbou nové instance třídy SystemWebView (cordova-android/framework/src/org/apache/cordova/engine/SystemWebView.java) [117], která třídu WebView rozšiřuje (viz zdrojový kód):

```
1. public SystemWebViewEngine(Context context, CordovaPreferences
   preferences)
2. {
3.     this(new SystemWebView(context), preferences);
4. }
```

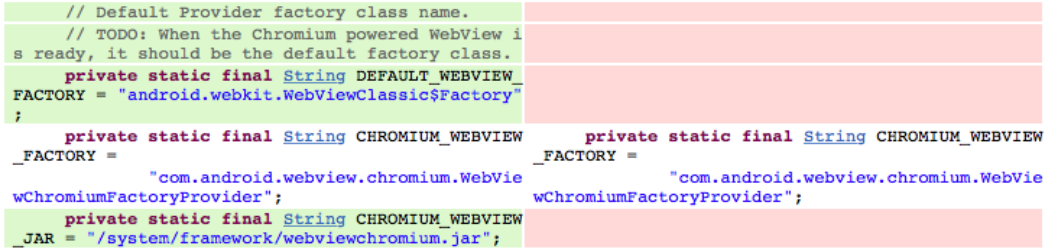
Shrnutí

Lze tedy říci, že Apache Cordova používá na platformě Android pro své běhové prostředí instanci **standardní třídy WebView**.

Mohou se implementace standardní třídy WebView lišit v rámci odlišných verzí Android OS?

Opět byla provedena analýza zdrojového kódu, který implementuje standardní třídu WebView na platformě Android. Vzhledem k tomu, že OS Android je otevřený systém, jsou jeho zdrojové kódy veřejné a lze využít například portálu GrepCode.com, kde je možné zdrojové kódy jednotlivých verzí systému stáhnout či procházet on-line.

Dle dokumentace je třída WebView součástí balíčku *android.webkit*. [118] Tento balíček byl vyhledán pro verze Android OS 4.3.1 a 4.4, které při testování vykázaly signifikantní změnu v podpoře moderních HTML5 vlastností ve prospěch novější verze. Obě verze knihovny WebView byly porovnány pomocí funkce diff. Z výstupu je zřejmá rozdílnost v konstantách, jejichž úkolem je definovat názvy balíčků pro tzv. WebViewFactoryProvider, který na jejich základě načítá příslušnou třídu pro instanciaci nativního WebView (viz obrázek 27).



```
    // Default Provider factory class name.
    // TODO: When the Chromium powered WebView i
s ready, it should be the default factory class.
    private static final String DEFAULT_WEBVIEW_
FACTORY = "android.webkit.WebViewClassic$Factory"
;
    private static final String CHROMIUM_WEBVIEW
_FACTORY =
            "com.android.webview.chromium.WebVie
wChromiumFactoryProvider";
    private static final String CHROMIUM_WEBVIEW
_JAR = "/system/framework/webviewchromium.jar";
```

```
    private static final String CHROMIUM_WEBVIEW
_FACTORY =
            "com.android.webview.chromium.WebVie
wChromiumFactoryProvider";
```

Obrázek 27: Výstup funkce diff (definice konstant) – porovnání balíčků com.google.android / android / 4.4_r1 / android.webkit.WebViewFactory verzí Andoid OS 4.3.1_r1 a 4.4_r1

```
    static WebViewFactoryProvider getProvider() {
        synchronized (sProviderLock) {
            // For now the main purpose of this functio
n (and the factory abstraction) is to keep
            // us honest and minimize usage of WebViewC
lassic internals when binding the proxy.
            if (sProviderInstance != null) return sProv
iderInstance;

            if (isExperimentalWebViewEnabled()) {
                StrictMode.ThreadPolicy oldPolicy = Str
ictMode.allowThreadDiskReads();
                try {
                    sProviderInstance = loadChromiumPro
vider();
                    if (DEBUG) Log.v(LOGTAG, "Loaded Ch
romium provider: " + sProviderInstance);
                } finally {
                    StrictMode.setThreadPolicy(oldPolic
y);
                }
            }

            if (sProviderInstance == null) {
                if (DEBUG) Log.v(LOGTAG, "Falling back
to default provider: "
                        + DEFAULT_WEBVIEW_FACTORY);
                sProviderInstance = getFactoryByName(DE
FAULT_WEBVIEW_FACTORY,
                        WebViewFactory.class.getClassLo
ader());
                if (sProviderInstance == null) {
                    if (DEBUG) Log.v(LOGTAG, "Falling b
ack to explicit linkage");
                    sProviderInstance = new WebViewClas
sic.Factory();
                }
            }
            return sProviderInstance;
        }
    }
```

```
    static WebViewFactoryProvider getProvider() {
        synchronized (sProviderLock) {
            // For now the main purpose of this functio
n (and the factory abstraction) is to keep
            // us honest and minimize usage of WebView
internals when binding the proxy.
            if (sProviderInstance != null) return sProv
iderInstance;

            Class<WebViewFactoryProvider> providerClass
;
            try {
                providerClass = getFactoryClass();
            } catch (ClassNotFoundException e) {
                Log.e(LOGTAG, "error loading provider",
e);
                throw new AndroidRuntimeException(e);
            }

            // This implicitly loads Preloader even if
it wasn't preloaded at boot.
            if (Preloader.sPreloadedProvider != null &&
                Preloader.sPreloadedProvider.getClass()
== providerClass) {
                sProviderInstance = Preloader.sPreloade
dProvider;
                if (DEBUG) Log.v(LOGTAG, "Using preload
ed provider: " + sProviderInstance);

            return sProviderInstance;
        }
```

Obrázek 28: Výstup funkce diff (metoda getProvider) – porovnání balíčků com.google.android / android / 4.4_r1 / android.webkit.WebViewFactory verzí Andoid OS 4.3.1_r1 a 4.4_r1

O dosazení příslušného názvu balíčku nativního WebView se stará metoda getProvider (viz obrázek 28), která ve verzi 4.3.1_r1 ještě obsahovala podmínku umožňující použití nového vykreslovacího jádra Chromium, avšak jedině v experimentálním módu. Oproti tomu verze 4.4_r1 obsahuje pouze volání metody getFactoryClass, která vrací jedinou definici názvu balíčku pro instanciaci vykreslovacího jádra, a to právě obsah konstanty

CHROMIUM_WEBVIEW_FACTORY – řetězec: *com.android.webview.chromium.WebViewChromiumFactoryProvider*.

Shrnutí

Dle analýzy zdrojového kódu se implementace WebView v rámci jednotlivých verzí platformy Android liší. Zlomová je verze 4.4, která jako první standardně používá knihovnu Chromium jako vykreslovací jádro webových technologií v rámci objektu WebView. Jak testy HTML5 podpory (kapitola 5.7.2), výkonu JavaScriptu či grafického výkonu (kapitola 5.7.4 a 5.7.5), tak i analýza zdrojových kódů prokázaly rozdílnost vlastností vykreslovacího jádra mezi všemi subverzemi OS Android.

Jaké vykreslovací jádro používá standardní prohlížeč Internetu v rámci OS Android (aplikace Internet)?

Jako vykreslovací jádro webových technologií používá standardní aplikace pro prohlížení Internetu na tzv. Nexus verzích OS (bez nástavby výrobce zařízení) systémový objekt WebView. Ten je pak založen na jádru Webkit či Chromium a to dle verze operačního systému (viz výše).

Reálné testy však prokázaly, že některá zařízení s nástavbou OS Android disponují vylepšenou verzí aplikace pro prohlížení Internetu, která zpravidla rozšiřuje vlastnosti standardního objektu WebView.

Jaké vykreslovací jádro používá aplikace Chrome?

Aplikace Chrome pro Android je dostupná od verze Android OS 4 a používá stejně jako WebView vykreslovací jádro Chromium. Jde však o implementace, které jsou vzájemně nezávislé.

Lze vykreslovací jádra aktualizovat?

Aplikace Chrome pro Android umožňuje pravidelné aktualizace, a proto nemusí být svými vlastnostmi konzistentní s neaktualizovaným WebView. WebView lze samostatně aktualizovat až od Android OS verze 5.

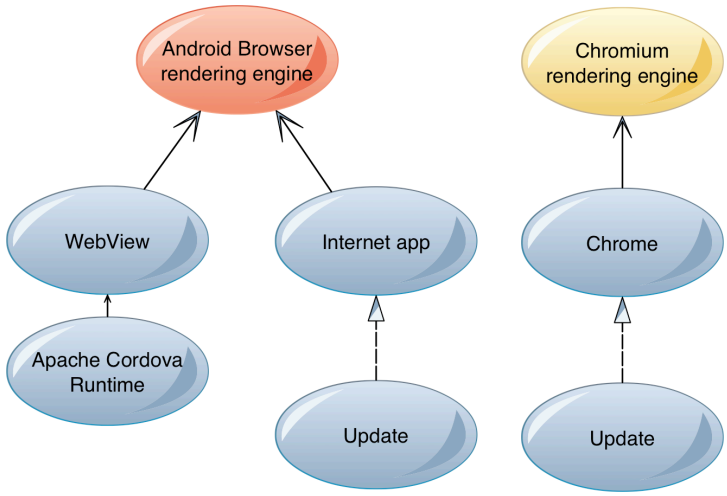
5.6.4 Vykreslovací jádra webových technologií v rámci platformy Android

Výše uvedená analýza zdrojových kódů projektu Apache Cordova a platformy Android umožnila tvorbu přehledové grafiky (mapy) vykreslovacích jader a jejich použití v rámci běhového prostředí Apache Cordova, aplikace Internet a prohlížeče Chrome pro Android. Je zohledněna také možnost samostatné aktualizace vykreslovacího jádra.

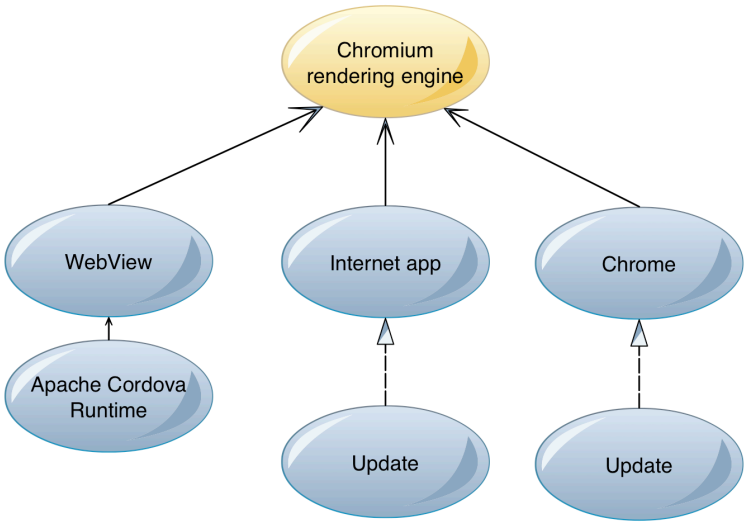
Situaci na platformách Android verze 4.0–4.4 ilustruje obrázek 29. Od verze 4.0 bylo možno instalovat do systému aplikaci Chrome pro Android. Ta již využívala vykreslovacího jádra Chromium (Chromium rendering engine) a bylo možno ji aktualizovat. Oproti tomu starší vykreslovací jádro (Android Browser rendering engine) založené na technologii Webkit je zde použito jak pro aplikaci Internet, tak pro běhové prostředí Apache Cordova, kde není možná aktualizace. V případě softvérových nástaveb některých výrobců však může být rozšířena či aktualizována aplikace Internet.

Platforma Android verze 4.4 byla díky použití jádra Chromium i v rámci nativního WebView velkým krokem kupředu, co se týče podpory moderních HTML5 vlastností a webového hybridního vývoje vůbec. Obrázek 30 ukazuje, že jednotné jádro je použito jak pro běhové prostředí Apache Cordova, tak pro aplikace Internet a Chrome pro Android. Aplikace Chrome pro Android i aplikace Internet jsou však stále samostatnými implementacemi, což umožňuje aktualizace. Naopak nemožnost aktualizace nativního WebView opět znevýhodňuje běhové prostředí Apache Cordova, které je tak závislé na vlastnostech konkrétní implementace WebView v rámci konkrétní vývojové verze OS.

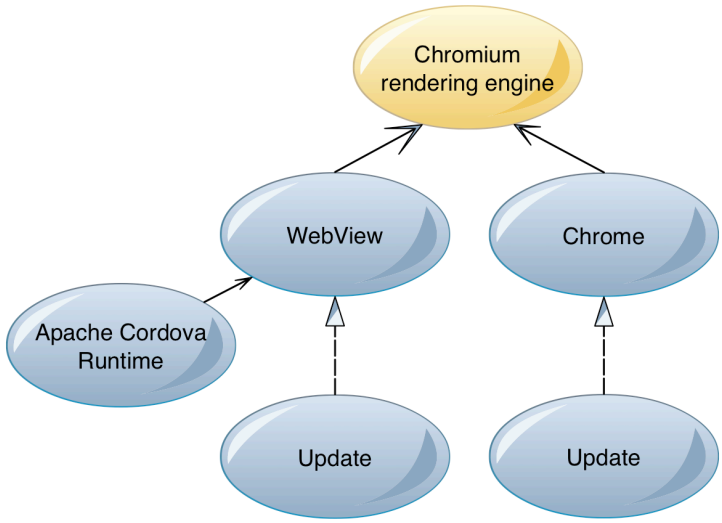
Obrázek 31 pak shrnuje stav webových běhových prostředí na platformě Android 5. Aplikace Chrome pro Android se stala oficiálním internetovým prohlížečem a zůstalo zachováno jediné vykreslovací jádro Chromium. Nově je však možné aktualizovat i nativní WebView, čímž i běhové prostředí Apache Cordova získává vysokou úroveň podpory HTML5 vlastností a může reflektovat aktuální požadavky v oblasti podpory technologií. Aplikace Chrome však zůstává samostatnou implementací a její aktualizace vycházejí zpravidla dříve a častěji než aktualizace systémového WebView.



Obrázek 29: Stav webových běhových prostředí na platformě Android verze 4.0–4.4



Obrázek 30: Stav webových běhových prostředí na platformě Android verze 4.4

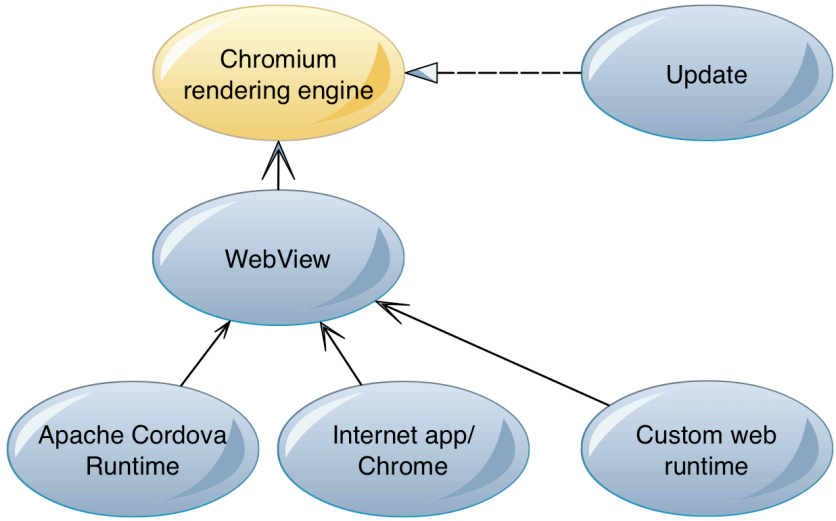


Obrázek 31: Stav webových běhových prostředí na platformě Android verze 5

5.6.5 Návrh použití společného vykreslovacího jádra

Na základě analýzy stavu implementace běhových prostředí na platformě Android byl předložen návrh použití společného vykreslovacího jádra (viz obrázek 32). Cílem je unifikovat vlastnosti běhových prostředí v rámci platformy a umožnit jejich jednoduchou aktualizaci a modernizaci. Návrh spočívá v implementaci jediného vykreslovacího jádra v rámci veškerých běhových prostředí. Jádro by bylo dodavatelem operačního systému pravidelně aktualizováno,

čímž by byl zaručen nutný předpoklad, aby splňovalo co nejvíce ze standardu HTML5. Výhodou by také byly totožné vlastnosti jednotlivých běhových prostředí.



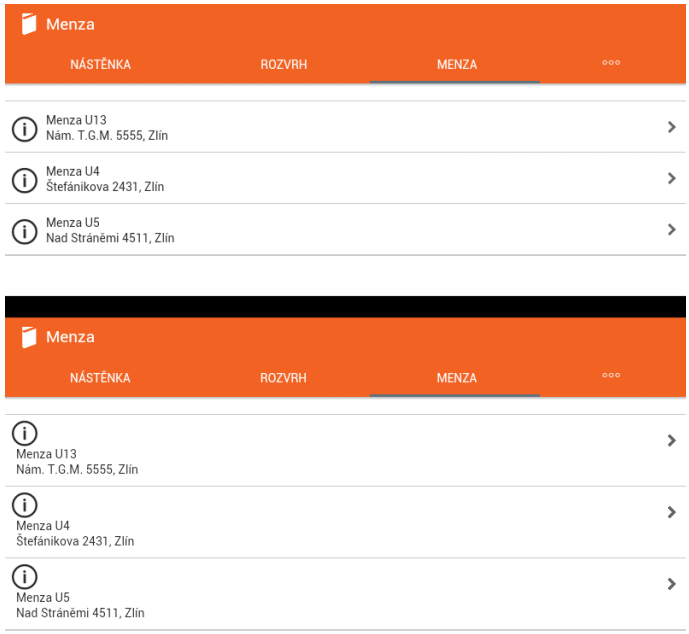
Obrázek 32: Návrh použití společného vykreslovacího jádra pro platformu Android

5.6.6 Identifikace zobrazovacích chyb způsobených nekonzistentní podporou HTML5 standardu

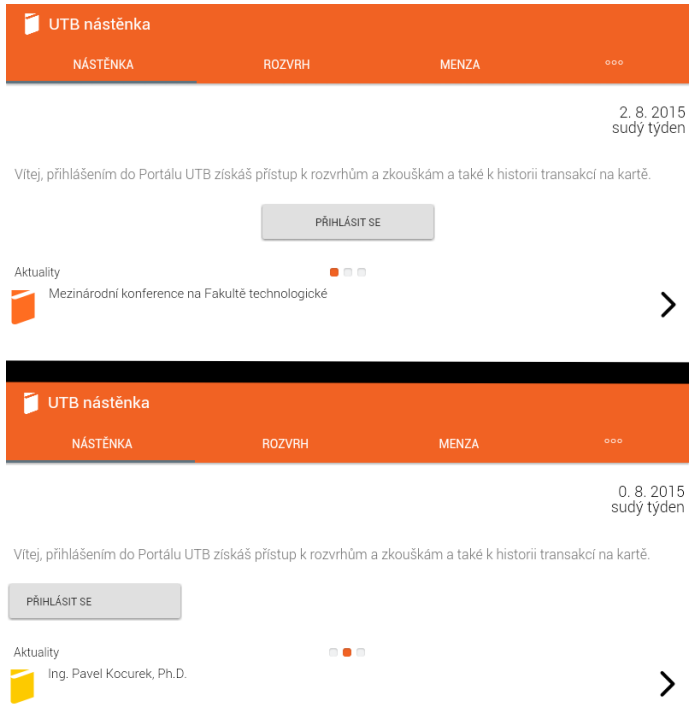
Během implementace testovacích aplikací pro komparaci HTML5 vývojových frameworků byly zaznamenány zobrazovací problémy zejména u zařízení s Android OS nižší verze než 4.4 a ty byly dále analyzovány pomocí ladicích nástrojů Remote Debugging (nástroj pro vzdálené ladění) a Google Developer Tools (vývojářské nástroje v rámci prohlížeče Google Chrome). Tento výzkum vedl k identifikaci nejčastějších zobrazovacích chyb. Jsou to: problémy s layoutem (rozmístění a pozicování prvků), nefunkční CSS feature queries [119], chybné zobrazování fontů, nefunkční SVG grafika, problém s průhledností – alfa kanálem, špatná interpretace vrstvení elementů v rámci efektů překryvu, problikávání animovaného obsahu při animacích překryvu a konečně problémy s interpretací touch (dotykových) událostí.

Obrázky 33–35 názorně demonstrují zobrazovací problémy týkající se pozicování prvků na příkladu reálné mobilní aplikace UTB. Pro implementaci uživatelského rozhraní byl použit framework Chocolate Chip UI, který nabízí sadu stylů pro platformu Android, iOS i Windows Phone. V případě použití standardního WebView v rámci wrapper technologie Apache Cordova byly zaznamenány zobrazovací problémy na platformách Android OS verze nižší než 4.4. Jak bylo zjištěno podrobnější analýzou pomocí vzdáleného ladění a nástroje Google Developer Tools, byly tyto problémy způsobeny zastaralou implementací CSS vlastností flex-box, což vede

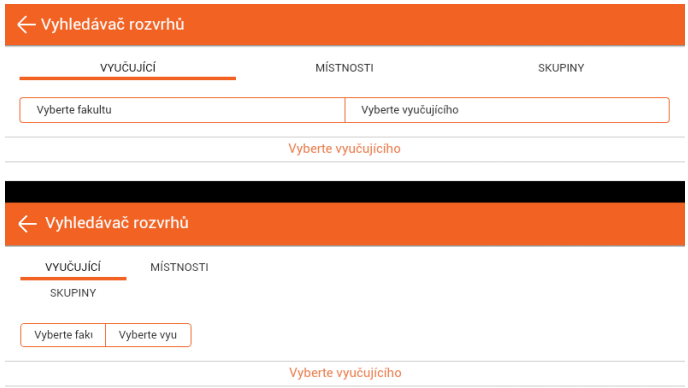
k nesprávnému pozicování objektů v rámci uživatelského rozhraní. Na obrázcích je vždy zachyceno správné zobrazení nahoře (běhové prostředí Crosswalk – Chrome 42) a chybné zobrazení dole (standardní WebView).



Obrázek 33: Ukázka chybného pozicování ikony informace – nahoře správně v běhovém prostředí Crosswalk, dole chybně ve standardním WebView



Obrázek 34: Ukázka chybného pozicování tlačítka přihlásit se – nahoře správně v běhovém prostředí Crosswalk, dole chybně ve standardním WebView



Obrázek 35: Ukázka chybného pozicování horních tlačítek a výběrových lišt – nahoře správně v běhovém prostředí Crosswalk, dole chybně ve standardním WebView

5.7 Řešení problému nekonzistence běhového prostředí

Výše uvedená nekonzistence v oblasti podpory HTML5 vlastností, ale také například výkonu JavaScriptu a podobně, vytváří jednu z největších bariér adopce metody vývoje WHMA. Lze se

dokonce setkat s takovými názory odborníků, které označují WHMA za nestabilní, neodladitelné a nevyzpytatelné, zejména co se týče zobrazování na různých verzích operačních systémů. Obecně shrnují tato tvrzení do teze, že HTML5 technologie ještě není připravena. Pravou podstatou problému je však to, že na trhu je stále velké procento zařízení, jejichž nativní běhová prostředí nenabízejí dostatečnou podporu moderních HTML5 vlastností, ani dostatečný výkon, který je nutný pro běh moderních HTML5 UI frameworků.

Řešením této situace je minimalizace počtu nehomogenních běhových prostředí (viz tabulka 21), pro které musí být mobilní aplikace odladěna. Nejjednodušší volbou je podpora pouze nejnovějších verzí mobilních OS, nicméně takové řešení by znamenalo, že mobilní aplikace nebude podporována většinou aktuálních zařízení na trhu. Bylo však nalezeno řešení této situace v podobě implementace jednotného běhového prostředí přímo do balíčku nativní aplikace. V takovém případě je nativní objekt WebView nahrazen prohlížečem, který využívá například Open Source knihovnu Chromium [120]. Ta je pak přímo součástí zdrojového kódu aplikace a zajišťuje žádanou konzistenci skrze různé verze operačních systémů a navíc také dostatečnou podporu HTML5 vlastností i na starších verzích.

V dalším textu budou představena nadstandardní běhová prostředí použitelná v rámci wrapper technologie Apache Cordova/Phonegap a budou na nich provedena měření podpory vlastností HTML5, výkonu JavaScriptu a grafických operací.

5.7.1 Nadstandardní běhová prostředí pro Apache Cordova/Phonegap

Postupně byly nalezeny následující projekty, zabývají se implementací renderovacího jádra Chromium do nativních aplikací:

ChromeView

Projekt ChromeView [121] je nativní knihovnou pro OS Android, která implementuje třídu ChromeView, jež funguje podobně jako původní WebView, ale používá kód open source knihovny Chromium. Koncem roku 2014 však byl projekt prohlášen za dále neudržovaný a původní tvůrce doporučuje použít projekt Crosswalk.

Crosswalk

Projekt Crosswalk [122], sponzorovaný společností Intel Corporation, se přímo zaměřuje na tvorbu hybridních aplikací, či možnost vestavění webového prohlížeče s jádrem Chromium přímo do aplikací nativních. Výhodou tohoto projektu je kompatibilita přímo s wrapper technologií Apache Cordova/Phonegap a s příchodem verze Apache Cordova Android 4.0.0 byl dokonce uveden oficiální plugin [123] pro rychlé vložení do projektu.

Webview+

Za vývojem projektu Webview+ [124] stojí společnost Ludei, která je známa svou specializací na optimalizaci běhových prostředí pro HTML5 hry. Webview+ implementuje jádro Chrome verze 37 a na portále GitHub je jako plugin dostupný pro Apache Cordova/Phonegap, pro platformy Android [124] a iOS [125]. Kromě jádra Chromium disponuje běhové prostředí HTML5 API Webworkers, CSS3 Transforms, Font-face, XHR2, WebGL a dalšími.

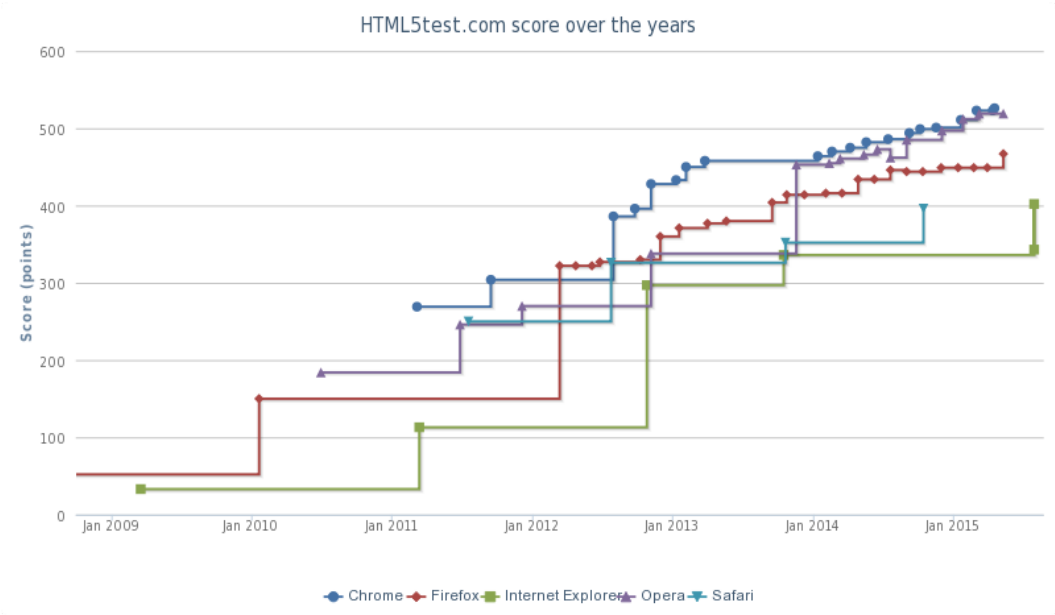
Plugin WKWebView

Na platformu iOS se zaměřuje projekt vyvíjený společností Telerik [126]. Jde o plugin pro Apache Cordova/Phonegap, který nahrazuje u zařízení, u nichž je to možné (iOS >= 8.1), objekt třídy UIWebView novějším WKWebView. Bohužel stále není z technických důvodů dostupný oficiální plugin projektu Apache Cordova (k dispozici je zatím pouze neoficiální zkušební verze [127]), jelikož WKWebView neumožňuje načítání souborů z lokálního souborového systému zařízení. Tato vlastnost je však klíčovým požadavkem WHMA, jejichž aplikační kód je právě součástí lokálního souborového systému v rámci kontejneru aplikace. Plugin společnosti Telerik řeší tento nedostatek spuštěním lokálního serveru v jednom z vláken aplikace.

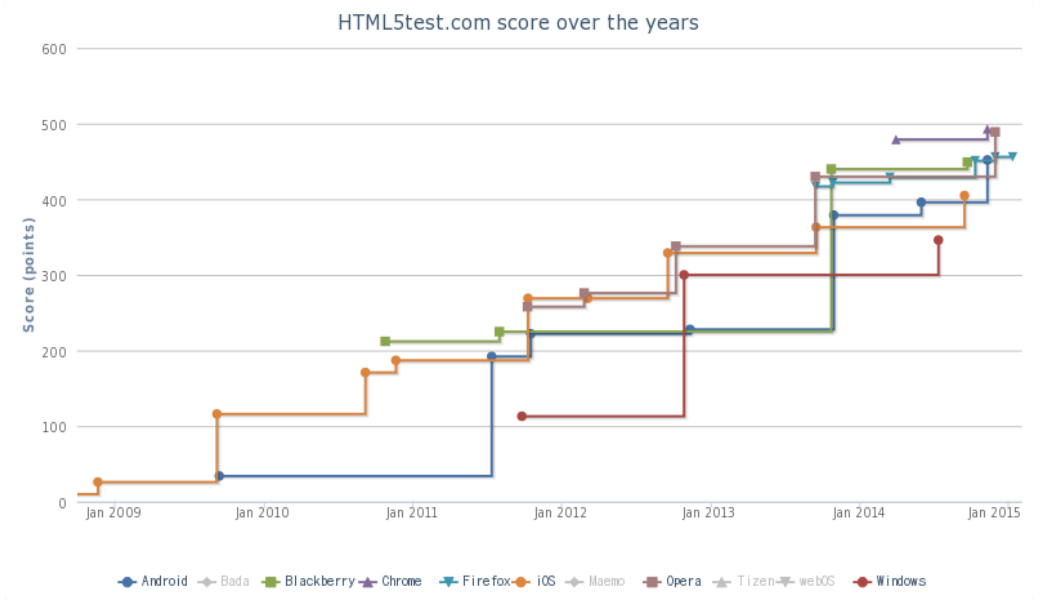
Z výše uvedených projektů pak byla vybrána běhová prostředí Crosswalk, WebView+ a Telekrik WKWebView, pro analýzu podpory HTML5 vlastností skrze jednotlivé verze operačních systémů.

5.7.2 Měření podpory HTML5 vlastností ve vybraných běhových prostředích

Pro jednotné hodnocení podpory HTML5 vlastností vybraných běhových prostředí byla zvolena testovací platforma serveru html5test.com. Samotné testování probíhá postupným testem vlastností klienta, který přistupuje na stránky s testovacími skripty a to v následujících kategoriích: sémantické značky, přístup k zařízení, konektivita, výkon a integrace, multimédia, 3D grafika a efekty, podpora offline módu a úložišť a další. Podpora vlastností v rámci kategorií je pak bodově hodnocena, přičemž maximální možný bodový zisk vyjadřující plnou podporu HTML5 vlastností je 555 bodů. Obrázek 36 ilustruje bodové zisky desktopových prohlížečů a obrázek 37 zisky mobilních prohlížečů v letech 2009–2015.



Obrázek 36: Přehled bodování desktopových prohlížečů v letech 2009 až 2015 dle serveru html5test.com



Obrázek 37: Přehled bodování mobilních prohlížečů v letech 2009 až 2015 dle serveru html5test.com

V rámci analýzy podpory HTML5 vlastností vybraných běhových prostředí byly implementovány testovací aplikace. Apache Cordova projekt byl vytvořen pomocí CLI (Command Line Interface) nástroje a dále byla upravena adresa vstupní stránky v souboru config.xml na URL adresu http://html5test.com. Podstatou testovací aplikace je zobrazit po

spuštění web html5test.com, kde proběhne test, který následně zobrazí číselné ohodnocení úrovně podpory HTML5 vlastností. Testovací aplikace jsou v podstatě totožné a liší se pouze vestavěným běhovým prostředím pro webové technologie. Pro testování byla připravena aplikace se standardním WebView (Android) respektive UIWebView (iOS), běžně používaným v rámci technologie Apache Cordova, dále aplikace, která používá běhové prostředí Crosswalk (Android), aplikace s běhovým prostředím tvořeným pomocí objektu WKWebView (iOS) a aplikace používající WebView+ (Android i iOS).

5.7.3 Zhodnocení výsledků měření podpory HTML5 vlastností

Tabulka 22 uvádí výsledky testování, přičemž výsledky jednotlivých testovacích aplikací jsou vztažené vždy ke konkrétní verzi operačního systému. Tabulka také uvádí v rámci OS Android označení tzv. API. Krom výsledků jednotlivých aplikací je uveden i výsledek nativní aplikace Internet, který zpravidla koresponduje s výsledkem standardní Apache Cordova aplikace. V obou případech je totiž použit stejný objekt nativního WebView. Aplikace byly instalovány do prostředí Android emulátoru (popřípadě iOS simulátoru) s instalovanou standardní verzí operačního systému.

Z tabulky je zřejmé, že podpora HTML5 vlastností je kontinuálně navyšována s novými verzemi operačních systémů. Přesto však standardní podpora u Android OS verze nižší než 4.4 je nedostatečná. Taktéž iOS verze 7.1 s hodnotou 363 je na hranici použitelnosti s moderními HTML5 UI frameworky.

Problematickou situaci s nedostatečnou podporou a zejména nekonzistencí v rámci různých verzí operačních systémů však řeší nestandardní běhová prostředí Crosswalk (používá zobrazovací jádro ekvivalentní s Chrome 42), plugin WKWebView od společnosti Telerik či plugin WebView+ (ekvivalentní s Chrome 37).

Crosswalk vykazuje na platformě Android nejlepší výsledky. Jeho implementací lze získat v rámci aplikace běžící na jakékoliv subverzi OS Android verze 4 hodnocení v rozmezí 485–514 bodů. Výsledky testování UI frameworků ukázaly, že jde o naprosto dostačující podporu HTML5 technologií a testované aplikace netrpěly žádnými specifickými zobrazovacími problémy spjatými s nativním WebView konkrétní verze OS.

WebView+ vykázal bodově sice nižší výsledky, což je dáno implementací renderovacího jádra Chrome ve verzi 37 (na rozdíl od novější verze 42 použité v rámci Crosswalk pluginu), nicméně i ty se po testování HTML5 UI frameworků jeví jako dostačující.

Na platformě iOS pak byl implementován plugin pro náhradu zastaralého objektu UIWebView novější implementací třídy pro zobrazování webového obsahu WKWebView. Ten prokázal navýšení oproti standardnímu nativnímu WebView na platformách iOS 8.1 a vyšších

ze standardních 387 na 405 bodů. Na platformě iOS 7.X bohužel není WKWebView dostupné, takže i přes použití pluginu k navýšení podpory HTML5 vlastností nedojde, protože standardní běhové prostředí stále používá objekt UIWebView. Za zmínku stojí, že systémová aplikace Safari na platformách iOS 8.1 a vyšších používá již nový WKWebView, v rámci projektu Apache Cordova však toto nové běhové prostředí z technických důvodů zatím použito není. Je tedy nutné implementovat výše zmíněný plugin pro dosažení vyšší úrovně podpory HTML5 vlastností v rámci WHMA.

Tabulka 22: *Srovnání podpory vlastností HTML5 na různých verzích mobilních platforem a za použití různých běhových prostředí*

Verze OS	API	Aplikace Internet	Standardní WebView	Crosswalk (Chrome 42)	WKWeb-View	WebView+ (Chrome 37)
Android 4.0.3	15	220	220	-	N/A	-
Android 4.1.2	16	228	228	514	N/A	468
Android 4.2.2	17	221	221	485	N/A	468
Android 4.3.1	18	221	221	485	N/A	468
Android 4.4.2	19	396	396	514	N/A	477
Android 5.0.1	21	480	480	514	N/A	477
iOS 7.1	N/A	363	363	N/A	363	363
iOS 8.1	N/A	405	387	N/A	405	387
iOS 8.2	N/A	405	387	N/A	405	387
iOS 8.3	N/A	405	387	N/A	405	387
iOS 8.4	N/A	405	387	N/A	405	387

Implementace zkušebních aplikací pomocí moderních HTML5 UI frameworků (květen 2015) a následné testování pomohlo identifikovat verze operačních systémů a běhových prostředí, které jsou dostatečné pro bezproblémové vykreslování UI (v tabulce označeny zeleně). Z výsledku vyplývá nutnost implementace nadstandardního běhového prostředí (Crosswalk či WebView+) v případě subverzí OS Android menších než 4.4. Lze tak zajistit konzistentní výsledky i na těchto starších verzích. Pro měření byl použit Android Emulátor s čistou instalací OS Android patřičné verze. Bohužel kvůli známé chybě běhových prostředí Crosswalk a WebView+ nebylo možné spustit testovací aplikaci na OS Android verze 4.0.3 a fyzické zařízení nebylo k dispozici. Subverze OS Android 4.4 a novější již netrpí zobrazovacími problémy ani v případě standardního WebView. Co se týče platformy iOS, verze 7 vykazuje ještě občasné zobrazovací problémy, verze 8 je však při použití běhového prostředí WKWebView bezproblémová.

Provedená měření a hlubší analýza zároveň objasnila zdánlivě nelogické výsledky, které přineslo měření na několika reálných zařízeních. Jednalo se o zařízení s OS Android verze 4.1.2, dále dvě zařízení s verzí 4.4.2. a jedno zařízení s verzí 5.1.1. Zde bylo zjištěno, že nativní aplikaci Internet není možné spolehlivě používat jako prostředí pro ladění a testování, jelikož nelze ve všech případech konstatovat, že má díky použití nativního WebView stejnou podporu HTML5 vlastností jako standardní běhové prostředí Apache Cordova. Aplikace Internet sice v rámci platformy Android využívá nativní objekt třídy WebView pro zobrazování webového obsahu, ale může obsahovat rozšiřující implementaci renderovacího jádra. Jasně to prokazují naměřené hodnoty podpory HTML5 vlastností v tabulce 23, jež se liší ve sloupcích Aplikace Internet a standardní WebView i o desítky bodů ve prospěch aplikace Internet. Ukazuje to snahu výrobců telefonů, kteří dodávají i vlastní nástavby operačního systému Android (např. Samsung Touchwiz, LG UI, HTC Sence, Sony Xperia), poskytnout uživateli mimo jiné i lepší internetový prohlížeč, než je v operačním systému standardně. Tato rozšíření však vedou ještě k markantnějšímu navýšení nekonzistence vlastností operačních systémů a mnohdy také k nepochopení souvislosti zobrazovacích problémů na různých zařízeních v různých běhových prostředích ze strany koncových vývojářů aplikací.

Tabulka 23: *Měření podpory HTML5 vlastností na reálných zařízeních*

OS, verze	API	Výrobce	Aplikace Internet	Standardní WebView	Crosswalk (Chrome 42)	WebView+ (Chrome 37)
Android 4.1.2	16	Samsung	**346**	**326**	494	477
Android 4.2.2	17	Samsung	**430**	**396**	514	477
Android 4.2.2	17	LG	**458**	**384**	514	477
Android 5.1.1	17	Motorola	**518**	**489**	514	477

5.7.4 Měření výkonu JavaScriptu v rámci běhových prostředí

Výkon WHMA je do značné míry ovlivněn implementací javascriptového interpreta v rámci běhového prostředí. Z toho důvodu bylo provedeno měření výkonu JavaScriptu v běhových prostředích standardní Apache Cordova aplikace, Crosswalk, Webview+, v nativní internetové aplikaci a také v aplikaci Chrome (verze 44.0.2403.133). Cílem měření není poskytnout absolutní výsledky, nýbrž relativní čísla, dle kterých je možné seřadit běhová prostředí z hlediska výkonu. Dále by měření mělo zodpovědět otázky rozdílnosti výkonu JavaScriptu u starších a moderních zařízení a běhových prostředí. Zjištěné výsledky také mohou ukázat trendy vývoje v této oblasti.

Měření byla provedena nejdříve na zařízení – Samsung Galaxy Note 10.1 s instalací Android OS 4.1.2. (Na zařízení je instalována nástavba operačního systému Android Samsung Touchwiz, včetně nadstavbové aplikace k prohlížení internetu verze 4.1.2.-N8010XXUCMK2.). Reprezentantem moderních zařízení pak byl mobilní telefon Motorola Nexus 6, s instalací Android OS 5.1.1 (Jedná se o čistou instalaci OS Android s aktualizovanou aplikací Chrome verze 44.0.2403.133).

Pro testování výkonu implementace JavaScriptu byl vybrán tzv. benchmark test Octane 2.0 sestávající ze 17ti dílčích testů, které reflektují požadavky na moderní webové desktopové či mobilní aplikace. Výsledné skóre testu je číslo nepřímo úměrné době jeho běhu. Vyšší číslo tedy znamená vyšší výkon a kratší čas potřebný pro výpočet. Koncové skóre je geometrickým průměrem všech dosažených výsledků. Testy jsou zaměřeny zejména na: načítání a ukládání vlastností, volání funkcí a metod, polymorfismus, regulární výrazy, matematické operace, tvorbu a destrukci objektů, výkon technologie garbage collector, manipulaci s poli, parsování a kompilaci JavaScriptu, spouštění komplexních aplikací a další [128].

Pro standardní běhové prostředí Apache Cordova, ale také pro prostředí Crosswalk a Webview+, byla implementována testovací mobilní aplikace, která po startu načte úvodní stránku testu Octane 2.0. Do měření byla dále zahrnuta nativní aplikace Internet instalovaná v zařízení a aplikace Chrome (zařízení s Android OS 5.1.1 již disponuje pouze aplikací Chrome). Po dokončení testů byly odečteny výsledky pro příslušné běhové prostředí.

5.7.5 Zhodnocení výsledků měření výkonu JavaScriptu

Výsledky měření jsou uvedeny v tabulce 24 pro zařízení Samsung Galaxy Note 10.1 (OS Android 4.1.2) a v tabulce 25 pro zařízení Motorola Nexus 6 (OS Android 5.1.1). Sloupec „Octane test“ uvádí názvy jednotlivých testů, jejichž hodnoty pro konkrétní běhové prostředí jsou uvedeny na příslušném řádku testu. Celkovým výsledkem testu Octane 2.0 je geometrický průměr dílčích testů.

Výsledky měření na zařízení Samsung Galaxy Note 10.1 (OS Android 4.1.2)

Z výsledků vychází nejlépe běhové prostředí Webview+. Skóre je však srovnatelné s výsledkem běhového prostředí Crosswalk. Lze říci, že obdobných, byť překvapivě nižších hodnot, dosahuje také aplikace Chrome (verze 44). Výsledek standardního běhového prostředí Apache Cordova je dle očekávání srovnatelný s nativní internetovou aplikací používající stejnou knihovnu WebView, avšak výkonnostně značně zaostává za běhovými prostředími s jádrem Chromium.

Největší výkonnostní rozdíl byl zaznamenán u testu „zlib“, kdy nejvyšší výkon vykázalo běhové prostředí Crosswalk a srovnatelného výsledku dosáhla také aplikace Chrome. Test se

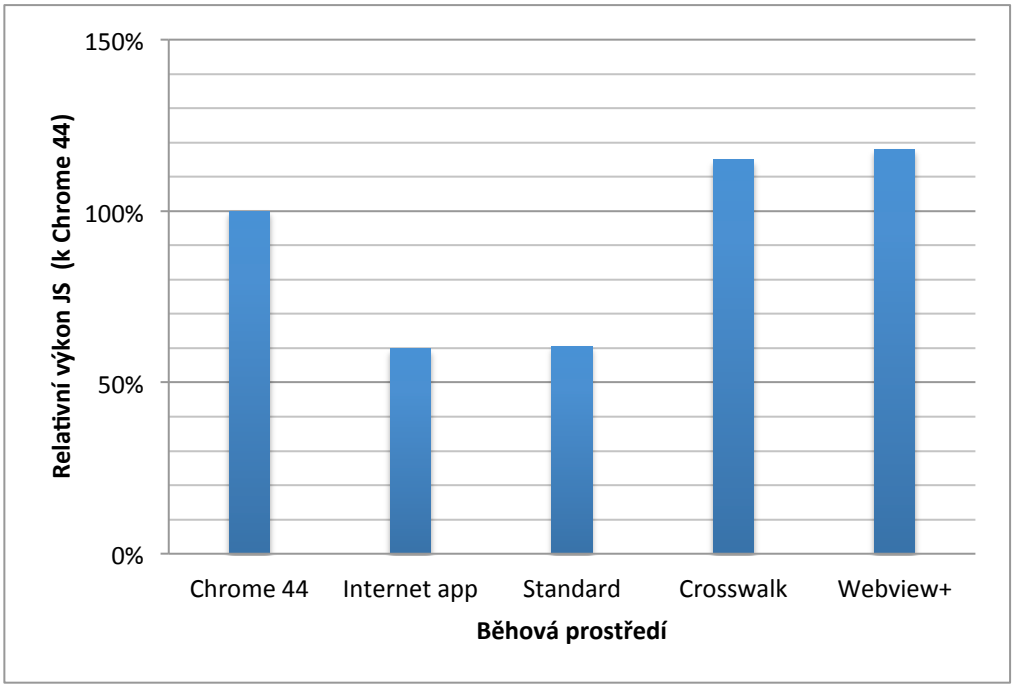
zaměřuje především na kompilaci a vykonávání JavaScriptového kódu (používá funkci eval). Naopak nejmenší rozdíly lze nalézt ve vyhodnocování a zpracování regulárních výrazů.

Celkové výsledky z tabulky 24 byly normalizovány a vztaženy k aplikaci Chrome (ta je totiž lehce instalovatelná do zařízení, lze jí tudíž využít jako testovací prostředí, ke kterému je vhodné znát možnost relativního zlepšení či zhoršení výkonu při použití různých běhových prostředí). Výše uvedené relativní výkonnosti jednotlivých běhových prostředí vztažené k aplikaci Chrome jsou uvedeny v grafu na obrázku 38. Lze z nich vyčíst, že běhová prostředí Crosswalk či Webview+ mohou nabízet ještě vyšší výkon než aplikace Chrome, naopak použijeme-li nativní Webview (aplikace Internet či standardní běhové prostředí Apache Cordova), můžeme zaznamenat snížení výkonosti až o 40%.

Uvedená konkrétní zjištění jsou aplikovatelná na širokou škálu reálných zařízení, která používají OS Android ve verzích vyšších než 4 a nižších než 4.4. Dle výsledků měření v kapitole 23 lze říci, že v případě verze 4.4 a novějších ztrácejí už nadstandardní běhová prostředí v oblasti zvýšení výkonu JS na významu, jelikož i standardní Webview zde používá jádro Chromium (od verze Android 5 dokonce aktualizované).

Tabulka 24: *Výsledky měření testu Octane 2.0 – Samsung Galaxy Note 10.1 (OS Android 4.1.2)*

Octane test	Chrome 44	Internet app	Standard	Crosswalk	Webview+
Richards	4914	2690	2687	5278	5235
Deltablue	6612	2969	2876	6539	6480
Crypto	3794	3086	3276	3855	3878
Raytrace	6438	2406	2303	6687	6540
EarlyBoyer	5802	4625	4432	5806	5724
Regexp	644	492	504	663	668
Splay	996	495	588	2437	3586
SplayLatency	1431	2206	2242	4546	4820
NavierStokes	5736	2593	2568	5447	5736
Pdf.js	1908	1946	1990	2095	2343
Mandreel	2290	1247	1267	2830	2621
MandreelLatency	1634	1430	1452	1600	1837
GB Emulator	6213	2980	3030	5067	6706
CodeLoad	2213	1974	1986	2395	2772
Box2DWeb	2675	1100	1099	2792	2973
Zlib	9983	1965	1961	10150	6173
Typescript	4855	2917	2875	4863	4113
Geometrický průměr	**3158**	**1888**	**1911**	**3629**	**3727**



Obrázek 38: Srovnání výkonnosti JavaScriptu – Samsung Galaxy Note 10.1 (OS Android 4.1.2)

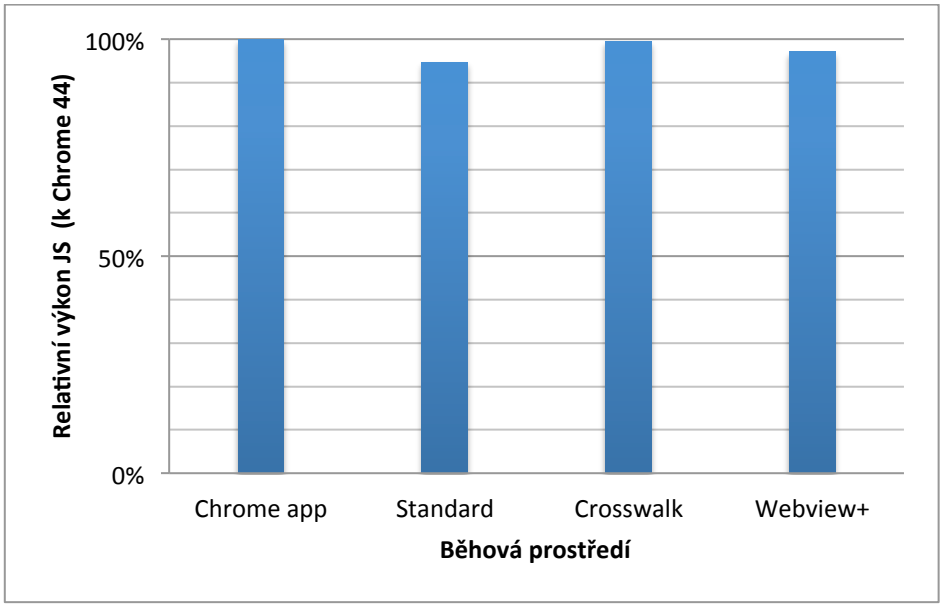
Výsledky měření na zařízení Motorola Nexus 6 (OS Android 5.1.1)

Jako zástupce moderních zařízení s OS Android byl pro realizaci měření výkonu JavaScriptu použit mobilní telefon Motorola Nexus 6 (OS Android 5.1.1). Výsledky jsou uvedeny v tabulce 25. Nejlepších výsledků zde získala aplikace Chrome, která v případě OS Android 5 a vyšší nahradila nativní aplikaci Internet. Všechny naměřené výsledky jsou srovnatelné a velmi pozitivním zjištěním je fakt, že instalace nadstandardního běhového prostředí do projektu Apache Cordova/Phonegap není v případě moderních zařízení nutná. Aplikace Chrome může být také využita jako testovací prostředí a lze očekávat stejný výkon JavaScriptu i při spuštění aplikace ve standardním prostředí Apache Cordova/Phonegap.

Graf na obrázku 39 zobrazuje procentuálně výkon jednotlivých běhových prostředí vztažený k aplikaci Chrome (ta je totiž standardním prohlížečem Internetu v této verzi OS Android). Z grafu je zřejmé, že konkrétní běhové prostředí již co se výkonu JavaScriptu týče nenabízí žádné znatelné zlepšení oproti aplikaci Chrome či standardnímu běhovému prostředí Apache Cordova/Phonegap.

Tabulka 25: *Výsledky měření testu Octane 2.0 – Motorola Nexus 6 (OS Android 5.1.1)*

Octane test	Chrome app	Standard	Crosswalk	Webview+
Richards	7353	8653	8010	7954
Deltablue	8285	7894	7471	7584
Crypto	8228	8402	8513	8461
Raytrace	8050	8945	8936	8974
EarlyBoyer	9710	9437	9437	9691
Regexp	969	990	997	1035
Splay	3411	3920	3900	3639
SplayLatency	4832	6509	5281	6350
NavierStokes	13007	11973	14428	14456
Pdf.js	2669	3020	2922	3282
Mandreel	4509	4602	4631	4607
MandreelLatency	8950	2505	2699	2670
GB Emulator	10177	11196	11106	9908
CodeLoad	3531	3353	3535	3384
Box2DWeb	2217	3442	3812	3846
Zlib	16787	9780	16819	10265
Typescript	5299	4382	4856	4615
Geometrický průměr	**5657**	**5360**	**5627**	**5494**



Obrázek 39: Srovnání výkonnosti JavaScriptu – Motorola Nexus 6 (OS Android 5.1.1)

5.7.6 Měření grafického výkonu běhových prostředí

Jedním z nejdůležitějších faktorů úspěchu mobilní aplikace je kvalitní uživatelský zážitek. Etalonem kvality uživatelského zážitku v oblasti mobilních aplikací jsou dnes úspěšné nativní aplikace. Hybridní aplikace se proto snaží o co nejvěrnější napodobení ovládání aplikace, včetně standardních animací a efektů. V tomto ohledu však v posledních letech velmi zaostávaly zejména webové hybridní aplikace a to kvůli nevýkonným či zastaralým běhovým prostředím. Běžný uživatel tento nedostatečný výkon vnímá a chování aplikace subjektivně hodnotí jako trhané, či nestandardní.

Cílem měření grafického výkonu je poskytnout informaci o výkonnosti běhových prostředí v oblasti 2D či 3D animací a CSS efektů. Měření se soustředí na parametr FPS (Frames per second – snímků za sekundu), který je přímo spojen s vnímáním plynulosti animací. Za účelem měření byly implementovány WHMA se standardním běhovým prostředím Apache Cordova a dále s běhovým prostředím Crosswalk (jádro Chrome 42) a Webview+ (jádro Chrome 37). Aplikace po svém spuštění načítají FPS benchmark test Particles Demo od společnosti Scirra (https://www.scirra.com/demos/c2/particles/). Test zobrazuje aktuální FPS, technologii použitou pro vykreslování grafických objektů (canvas2d či webgl), vytížení CPU v procentech a také počet současně zobrazovaných částic.

Měření byla provedena nejdříve na zařízení – Samsung Galaxy Note 10.1 s instalací Android OS 4.1.2. (Na zařízení je instalována nástavba operačního systému Android Samsung Touchwiz, včetně nadstavbové aplikace k prohlížení internetu verze 4.1.2.-N8010XXUCMK2.). Reprezentantem moderních zařízení pak byl mobilní telefon Motorola Nexus 6 s instalací Android OS 5.1.1 (Jedná se o čistou instalaci OS Android s aktualizovanou aplikací Chrome verze 44.0.2403.133).

5.7.7 Zhodnocení výsledků měření grafického výkonu

Výsledky měření jsou uvedeny v tabulce 0 pro zařízení Samsung Galaxy Note 10.1 (OS Android 4.1.2) a v tabulce 27 pro zařízení Motorola Nexus 6 (OS Android 5.1.1). Sledován byl parametr FPS, dále technologie použitá pro grafické výpočty, vytížení CPU a počet částic zobrazovaných na obrazovce.

Výsledky měření na zařízení Samsung Galaxy Note 10.1 (OS Android 4.1.2)

Provedená měření odhalila, že v případě graficky náročné aplikace je standardní běhové prostředí Apache Cordova/Phonegap nepoužitelné. Naměřeny byly pouze 3-4 FPS, což jsou hodnoty zdaleka nedosahující minimálních 24 FPS, při nichž člověk vnímá animace jako

plynulé [114]. V případě ostatních běhových prostředí však tato minimální hodnota dosažena byla, a to i v případě náročné animace 810–880 částic.

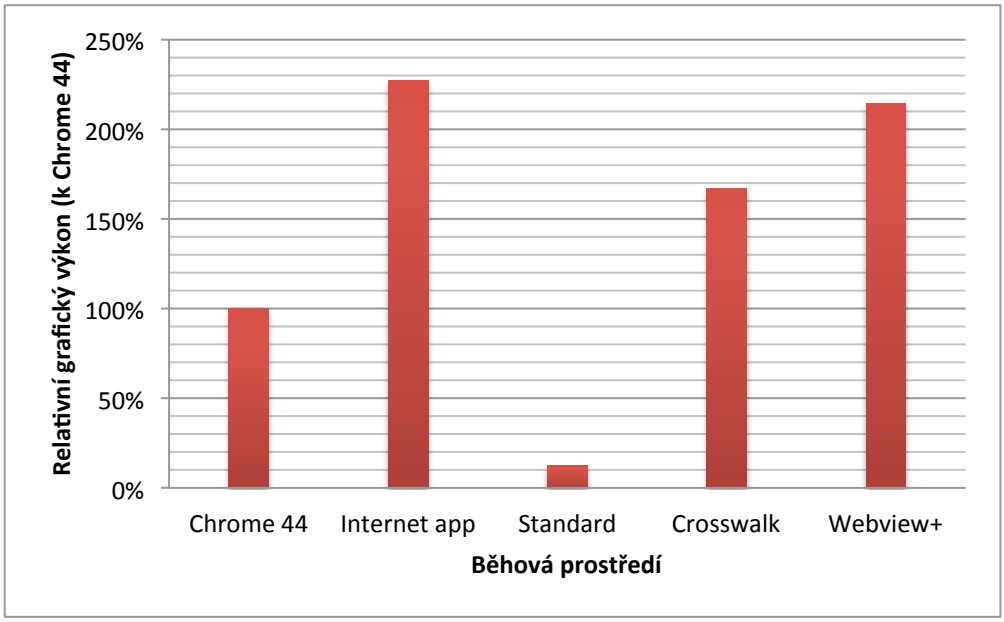
Výsledky měření grafického výkonu běhových prostředí přinesly poměrně neočekávaná zjištění. Nenaplnil se předpoklad o podobných výsledcích nativní aplikace Internet a standardního běhového prostředí Apache Cordova, tak jako tomu je v jiných srovnávaných parametrech. Nativní aplikace Internet totiž v oblasti grafického výkonu dosáhla nejlepších výsledků ze všech testovaných běhových prostředí. Je to názorná ukázka nekonzistence prostředí operačního systému Android, kdy je standardní knihovna WebView v rámci aplikace Internet dodatečně optimalizována přímo výrobcem zařízení Samsung a využívá implementaci technologie WebGL pro výpočty na grafické kartě. Tyto proprietární optimalizace však vedou k situaci, kdy nativní aplikace Internet paradoxně výkonem předčí i nadstandardní běhová prostředí či aplikaci Chrome 44, která z důvodů chybové implementace některých ovladačů v rámci technologie WebGL, tuto technologii standardně přímo zakazuje. V případě nedostupnosti WebGL je použita technologie Canvas2D, která grafické výpočty provádí přímo na CPU. V takovém případě lze plynulé animace zaznamenat pouze u těch zařízení, která mají velmi výkonné CPU, což není případ většiny běžných zařízení na trhu.

Konkrétní výsledky naměřené na výše zmíněném zařízení Samsung Galaxy Note 10.1 lze zobecnit následovně pro zařízení Andorid OS verze nižší než 4.4:

- Výkonnost grafických operací se liší dle použité technologie vykreslování
- Canvas2d využívá CPU a obecně vykazuje nižší výkon
- WebGL využívá GPU a obecně vykazuje vyšší výkon
- Nelze očekávat konzistenci grafické výkonnosti nativní aplikace Internet a standardního běhového prostředí Apache Cordova (kvůli optimalizacím či nadstandardním rozšířením ze stran výrobců)
- Na konkrétních zařízeních se lze setkat s chybějící podporou či chybnou implementací technologie WebGL (definuje WebGL blacklist [129])
- Projekt Crosswalk umožňuje sestavení s podporou WebGL i pro zařízení na blacklistu (kvůli možné nestabilitě však není doporučeno)
- Standardní běhové prostředí Apache Cordova je pro graficky náročné aplikace (např. hry) nepoužitelné a musí být nahrazeno nadstandardním běhovým prostředím

Tabulka 26: *Výsledky měření grafického výkonu běhových prostředí – Samsung Galaxy Note 10.1 (OS Android 4.1.2)*

	Chrome 44	Internet app	Standard	Crosswalk	Webview+
FPS	26-29	60-65	3-4	44-48	58-60
Technologie	Canvas2D	WebGL	Canvas2D	WebGL	WebGL
Vytížení CPU	63-67%	55-60%	80-100%	20-25%	32-41%
Počet částic	810-880	810-880	810-880	810-880	810-880



Obrázek 40: Srovnání grafického výkonu běhových prostředí – Samsung Galaxy Note 10.1 (OS Android 4.1.2)

Výsledky měření na zařízení Motorola Nexus 6 (OS Android 5.1.1)

Měření ukázalo, že zvolené zařízení disponuje dostatečně výkonným nativním běhovým prostředím, které standardně používá technologii WebGL pro grafické výpočty. Sledovaný parametr FPS tak i při náročných grafických výpočtech převyšoval minimální požadovanou hranici 24 FPS více než dvojnásobně. Standardní běhové prostředí Apache Cordova/Phonegap sice vykázalo nejnižší výkon, ten je však pro běh graficky náročných animací dostatečný a to díky tomu, že již také standardně používá výpočty pomocí WebGL. Největší naměřený rozdíl ve výkonu jednotlivých běhových prostředí tvoří pouze 14% (viz obrázek 41), což opět znamená, že není nutné použít nadstandardní běhové prostředí a testování v aplikaci Chrome

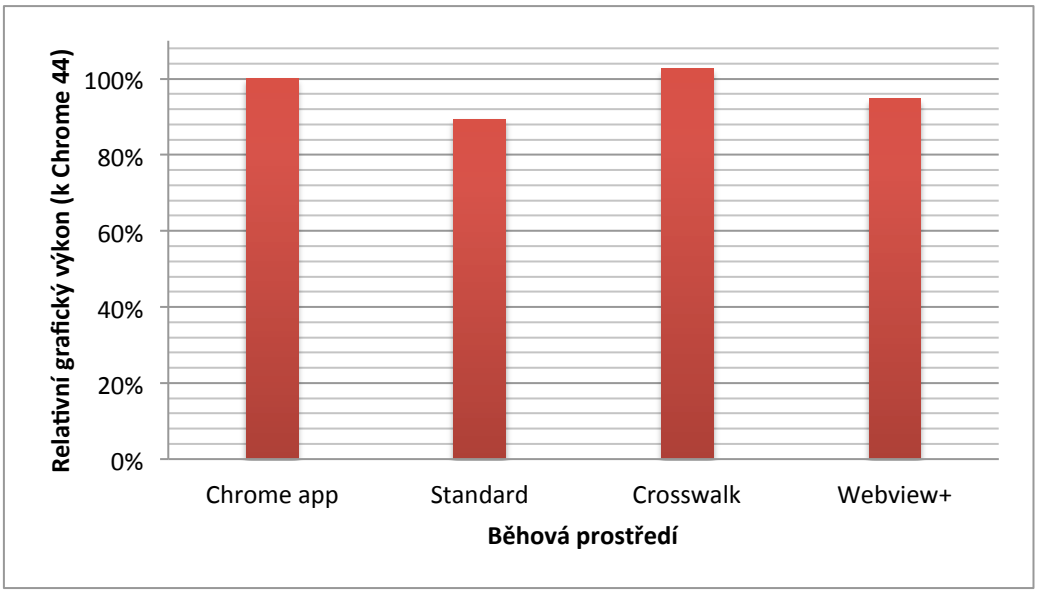
poskytne relevantní výsledky srovnatelné se standardním běhovým prostředím Apache Cordova/Phonegap.

Konkrétní výsledky naměřené na výše zmíněném zařízení Motorola Nexus 6 (OS Android 5.1.1) lze zobecnit následovně pro zařízení Andorid OS verze 4.4 a vyšší:

- Podporu WebGL lze očekávat standardně ve všech běhových prostředích
- Konzistence grafické výkonnosti aplikace Chrome a standardního běhového prostředí Apache Cordova/Phonegap je přijatelná i pro graficky náročné aplikace
- Na konkrétních zařízeních se lze setkat s chybějící podporou či chybnou implementací technologie WebGL (Definuje WebGL blacklist [129])
- Standardní běhové prostředí Apache Cordova/Phonegap je pro graficky náročné aplikace (např. hry) bez problému použitelné
- Není nutná implementace nadstandardních běhových prostředí, jelikož je vždy zachována dostatečná úroveň FPS, tudíž je toto zrychlení nepodstatné

Tabulka 27: *Výsledky měření grafického výkonu běhových prostředí – Motorola Nexus 6 (OS Android 5.1.1)*

	Chrome app	**Standard**	**Crosswalk**	**Webview+**
FPS	54-59	47-54	56-60	50-57
Technologie	WebGL	WebGL	WebGL	WebGL
Vytížení CPU	31-43	32-40	32-40	33-47
Počet částic	810-880	810-880	810-880	810-880



Obrázek 41: Srovnání grafického výkonu běhových prostředí – Motorola Nexus 6 (OS Android 5.1.1)

5.7.8 Doporučení běhového prostředí

V rámci této kapitoly bylo provedeno testování a porovnání běžného běhového prostředí v rámci wrapper technologie Apache Cordova/Phonegap s nativní aplikací Internet a dalšími dostupnými běhovými prostředími, jejichž úkolem je zvýšit úroveň podpory HTML5 a zejména konzistence skrze jednotlivé verze operačních systémů. Nadstandardní běhová prostředí navíc disponují odlišnou implementací interpreta technologie JavaScript, a proto byly provedeny benchmark testy za účelem zjištění významných rozdílů v jejich výkonnosti. Grafický výkon běhového prostředí je dalším velmi důležitým parametrem přímo ovlivňujícím uživatelský zážitek u moderních aplikací, který nezávisí pouze na výpočetních schopnostech zařízení, ale jak prokázala i měření, odvíjí se především od implementace moderních grafických knihoven.

Reálná měření na zařízeních Android OS starších než 4.4 vyvrátila teoretický předpoklad o shodných parametrech nativní aplikace Internet a standardního běhového prostředí Apache Cordova/Phonegap, protože na reálných zařízeních s OS Android byla naměřena rozdílná podpora HTML5 vlastností. Přestože by v obou případech měl být použit standardní WebView, aplikace Internet může obsahovat implementaci rozšíření výrobce konkrétního zařízení, jež používá novější či vylepšené renderovací jádro. Vývojáři by proto u reálných zařízení neměli očekávat, že aplikace Internet může sloužit k testování zobrazování či uživatelského zážitku, který bude vykazovat i finální aplikace se standardním WebView. Odlišná je však situace v případě zařízení s Android OS verze 4.4 a novější, kde je již aplikace Chrome použita jako

nativní internetový prohlížeč a lze zde tedy naměřit velmi podobné parametry jako u standardního běhového prostředí Apache Cordova/Phonegap.

Na základě výše uvedených zjištění, lze říci, že nadstandardní běhová prostředí jsou nezbytnou součástí reálné mobilní aplikace, u níž je vyžadována podpora starších verzí mobilních operačních systémů a zároveň je implementována v souladu s moderními HTML5 standardy, nebo například využívá některý z HTML5 UI frameworků.

Benchmark testy ukázaly, že nadstandardní běhová prostředí přinášejí až dvojnásobně lepší výsledky výkonnostních testů oproti standardnímu WebView a jsou také nezbytné pro graficky náročnější aplikace. Tato zlepšení lze očekávat na všech zařízeních s operačním systémem Android verze nižší než 4.4, u verzí novějších již ale nepřinášejí významné zlepšení.

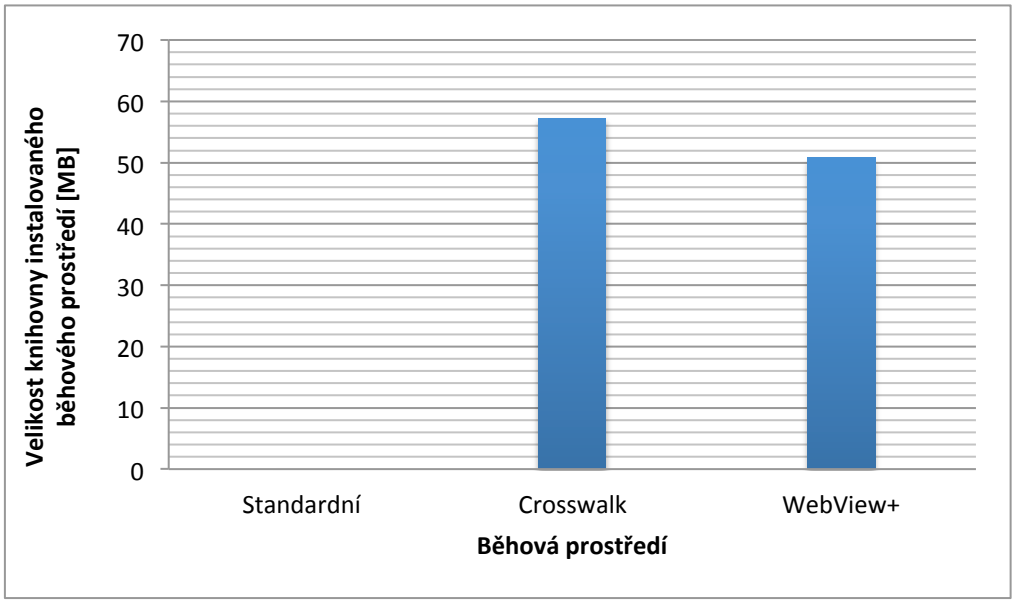
Nevýhodou implementace běhového prostředí je však navýšení velikosti jak instalačního balíčku aplikace, tak aplikace instalované v zařízení. Tabulka 28 uvádí srovnání velikostí při použití standardního běhového prostředí a pluginů Crosswalk a WebView+. Na grafu (viz obrázek 42) je pak srovnání přidané velikosti běhového prostředí v rámci instalované aplikace v zařízení.

Datově nejobjemnější je běhové prostředí projektu Crosswalk, které je tvořeno knihovnou implementující jádro Chrome 42. WebView+ je sice datově o takřka 12% úspornější, ale implementuje starší jádro Chrome 37. Navíc projekt Crosswalk přináší častější aktualizace díky aktivní komunitě i společnosti Intel, která jej podporuje a vyvíjí. V oblasti podpory HTML5 vlastností dosahuje Crosswalk lepších výsledků než WebView+ a výkon JavaScriptu je na srovnatelné úrovni. WebView+ vykazuje při testech vyšší hodnoty FPS, nicméně na úkor vyššího vytížení CPU. Pro běžné aplikace tedy lze spíše doporučit běhové prostředí Crosswalk, jehož hodnoty FPS se pohybovaly i během náročné animace dostatečně nad vyžadovanou hodnotou 24 FPS.

Závěrem je tedy doporučení běhového prostředí Crosswalk, přičemž však musí být zváženo navýšení velikosti instalačního balíčku i velikosti, kterou aplikace zabere po instalaci do zařízení (ta je více než dvojnásobná). Toto doporučení však platí pouze pro zařízení s OS Android verze nižší než 4.4. U novějších verzí lze v praxi úspěšně použít standardní běhové prostředí Apache Cordova/Phonegap a vyhnout se tak navýšení velikosti projektu.

Tabulka 28: *Srovnání velikostí balíčků a instalovaných aplikací s různými běhovými prostředími, včetně srovnání velikostí samotných běhových prostředí (velikost v MB)*

	Standardní	Crosswalk	WebView+
APK	1,8	21,9	19,4
Instalovaná aplikace	2,1	59,3	52,9
Knihovna běhového prostředí - v APK	0	20,1	17,6
Knihovna běhového prostředí - instalovaná	0	57,2	50,8



Obrázek 42: Srovnání přidané velikosti běhových prostředí v rámci instalované aplikace

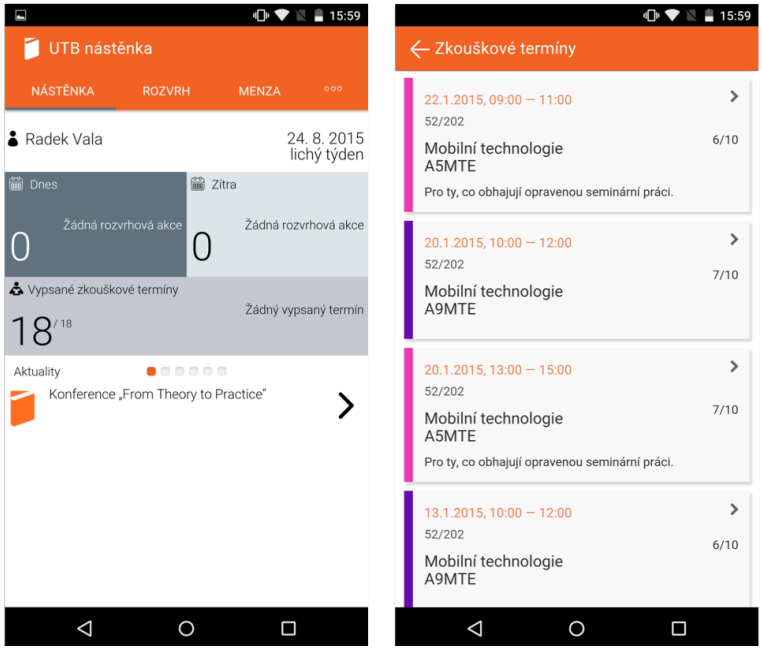
5.8 Aplikace navržených postupů v reálných projektech

Navržené postupy implementace a výběru vhodných technologií byly vyžity v rámci reálných vývojových projektů mobilních aplikací UTB [90] a Retigo Vision [104]. Obě aplikace jsou v současné době publikovány na obchodech App Store a Google Play, přičemž je plánováno i vydání aplikace UTB pro Windows Store.

V rámci aplikace Retigo Vision bylo implementováno vlastní uživatelské rozhraní pomocí HTML5 technologií bez použití vývojového frameworku. Jde totiž o přesnou simulaci ovládacího panelu reálného konvektomatu, nikoli mobilní aplikaci disponující standardním uživatelským rozhraním.

Obrázek 43: Ukázka UI aplikace Retigo Vision – vlevo hlavní menu konvektomatu, vpravo podmenu Easy cooking.

Pro implementaci uživatelského prostředí aplikace UTB byl použit přístup, který spočívá ve využití moderního HTML5 UI frameworku (definovaný v kapitole 5.3.2), jehož výběr byl uskutečněn na základě metody navržené v kapitole 5.4 a výkonnostních testů, popsaných v kapitole 5.5.



Obrázek 44: Ukázka UI mobilní aplikace UTB na zařízení s Android OS – vlevo nástěnka po přihlášení, vpravo výpis zkouškových termínů

Při vývoji aplikace UTB bylo nejprve použito standardní běhové prostředí Apache Cordova/Phonegap pro účely ověření jeho vlastností na reálné aplikaci. Během testování s 24 testery (studenti 3. a 5. ročníku FAI, UTB ve Zlíně) byly zjištěny zobrazovací problémy či problémy týkající se výkonu aplikace a plynulosti animací zaznamenané i během testování moderních UI frameworků a popsané v kapitole 5.6. Implementací nadstandardního běhového prostředí Crosswalk (viz doporučení z kapitoly 5.7.8) pro starší zařízení, používající verzi operačního systému Android OS < 4.4, byly tyto problémy eliminovány. V rámci platformy iOS pak byl použit plugin pro běhové prostředí WKWebView, popsaný v kapitole 5.7.1.

Během implementace a testování této reálné aplikace byly také potvrzeny závěry týkající se identifikace problémových verzí běhových prostředí na platformě Android a iOS, které byly vyvozeny z aplikačních testů v kapitole 5.6.2. Praktická aplikace tak prokázala nutnost implementace nadstandardního běhového prostředí v rámci podpory starších zařízení. V případě platformy Android však lze říci, že od verze OS 4.4 včetně již není nutné v praxi používat nadstandardní běhové prostředí Crosswalk a to díky dostatečně modernímu a výkonném vykreslovacímu jádru Chromium.

5.9 Vyhodnocení navržených postupů použitých v reálném vývojovém projektu aplikace UTB

Webové hybridní metody vývoje mobilních aplikací jsou přes svou vzrůstající oblíbenost stále pokládány za nestandardní a nejistou cestu k dosažení kvalitního výsledku. Jestliže za etalon kvality a míry uživatelského zážitku může být pokládána dobře udělaná nativní mobilní aplikace, vyvstává otázka, do jaké míry se může tomuto ideálu WHMA přiblížit.

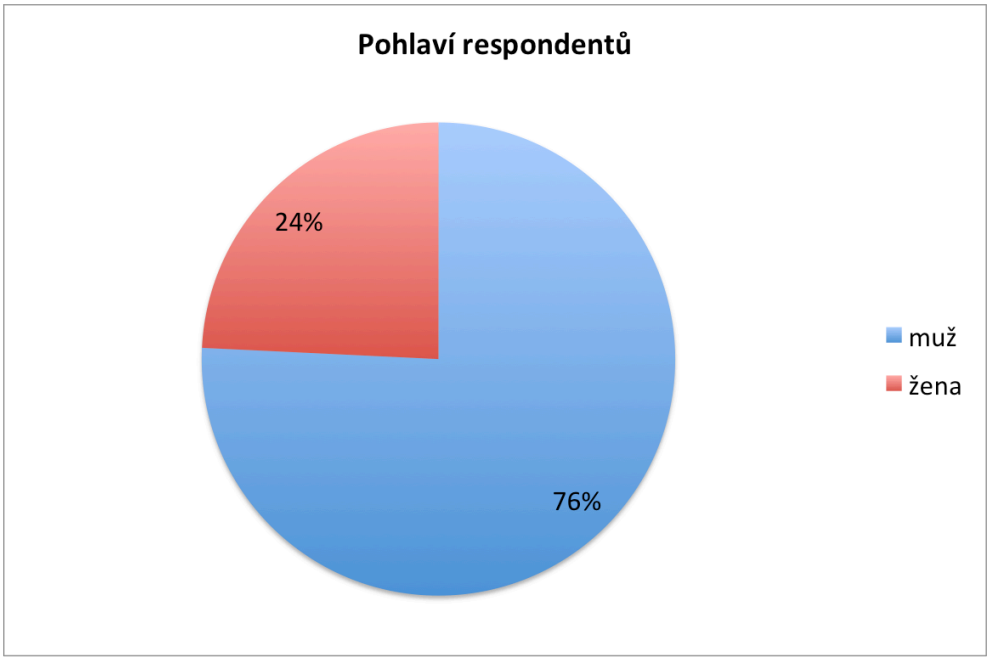
Na projektu reálné aplikace UTB byla vyhodnocena úspěšnost navržených metod vývoje a to zejména v oblasti stability aplikace, výkonu a uživatelského zážitku. Sledované parametry byly ověřeny pomocí dotazníkového průzkumu přímo u koncových uživatelů aplikace.

5.9.1 Dotazníkový průzkum

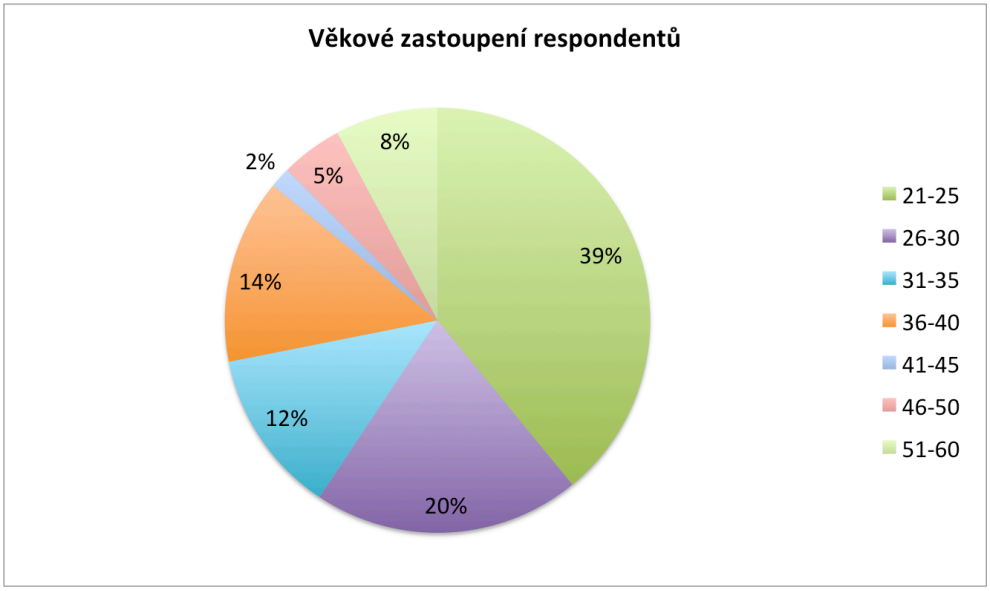
Během měsíců června až srpna 2015 byli studenti a zaměstnanci UTB ve Zlíně požádáni o vyplnění dotazníku, týkajícího se mobilní aplikace UTB a využívání mobilního zařízení obecně. Průzkumu se zúčastnilo 162 respondentů. Přesné znění dotazníku je součástí přílohy „Dotazníkový průzkum“.

Výsledky dotazníku – demografické údaje

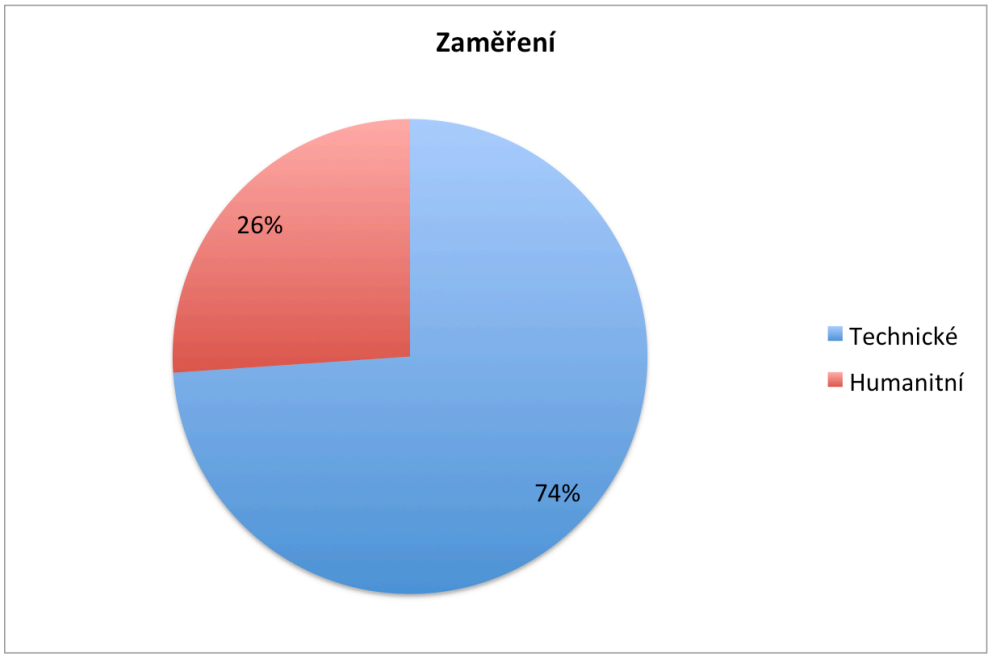
Průzkumu se zúčastnilo celkem 159 respondentů (103 mužů a 56 žen), z toho na dotazy týkající se aplikace UTB odpovídalo 52 mužů a 20 žen. Grafy na obrázcích 45 až 48 ukazují, že nejvíce byli zastoupeni účastníci z věkových skupin 21–25 let, 26–30 let a 36–40 let. Takřka tři čtvrtiny respondentů uvádí, že zaměření jimi absolvovaného studia bylo technické. Mezi respondenty bylo 64% uživatelů se zájmem o mobilní technologie (z toho 38% uživatelů s hlubším zájmem o software i hardware), 22% se zajímá pouze v oblasti pracovního využití a 16% respondentů se o mobilní technologie nezajímá vůbec. 40% respondentů uživatelů pak sama sebe označuje za běžné uživatele, 58% se označuje za pokročilé uživatele a pouhá 2% uvádí, že mají zkušenosti s programováním mobilních aplikací.



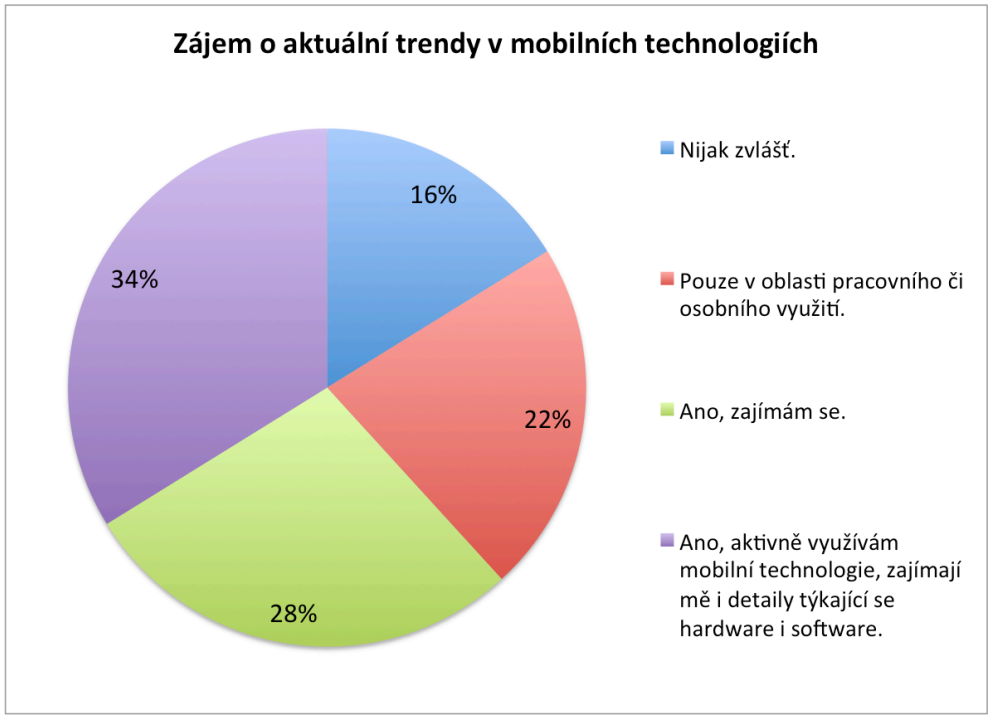
Obrázek 45: Procentuální zastoupení pohlaví mezi respondenty



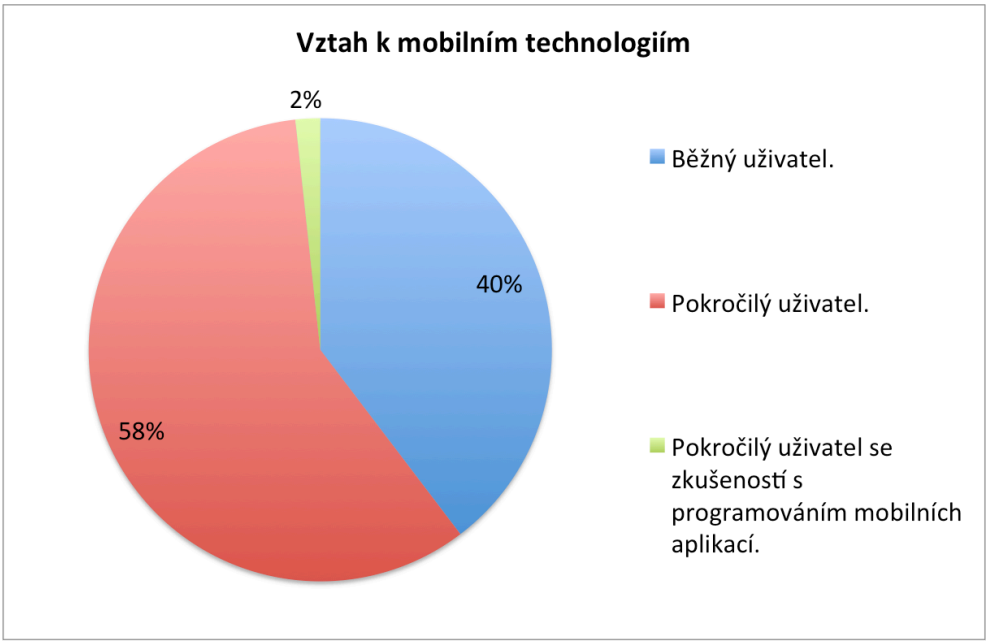
Obrázek 46: Věkové zastoupení respondentů



Obrázek 47: Zaměření absolvovaného studia respondentů



Obrázek 48: Zájem respondentů o trendy v oblasti mobilních technologií



Obrázek 49: Sebehodnocení vztahu respondentů k mobilním technologiím

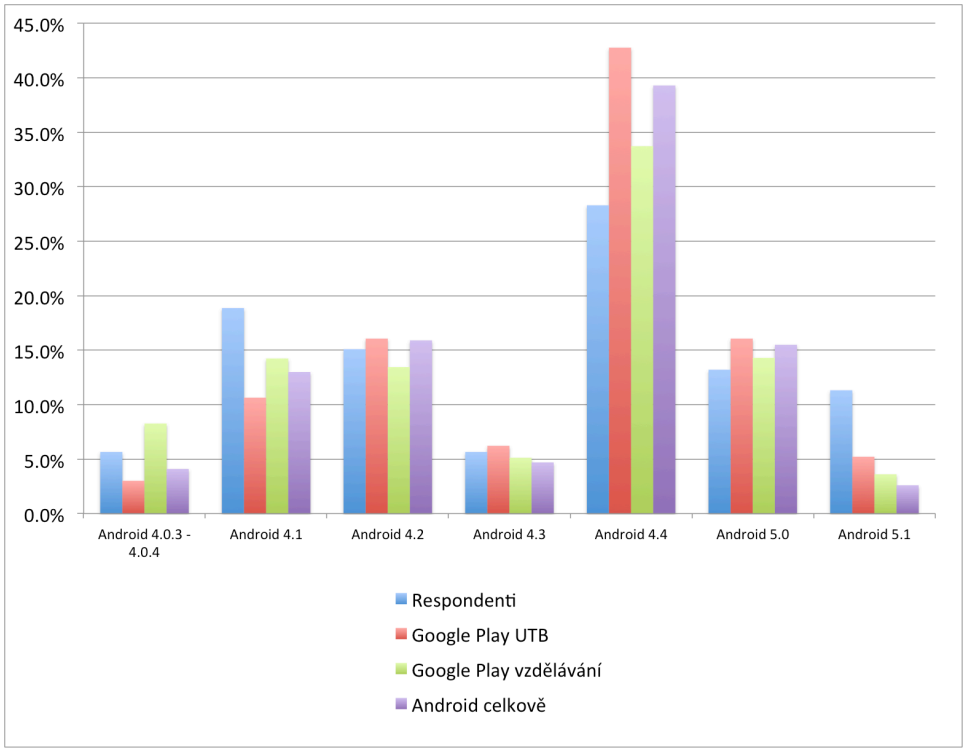
Výsledky dotazníku – statistiky subverzí operačního systému Android

Velmi zajímavým údajem je statistika subverzí OS Android, na kterých je aplikace provozována. Proto byli respondenti v rámci dotazníkového průzkumu požádáni o uvedení konkrétní subverze OS, kterou používají.

Na obrázku 50 jsou pomocí grafu srovnány výsledky dotazníkového průzkumu (sloupec respondenti) s údaji získanými z portálu Google Play. Portál nabízí data týkající se distribuce subverzí OS Android, na nichž je provozována aplikace UTB (sloupec Google Play UTB) a dále data týkající se všech aplikací zařazených na Google Play do kategorie vzdělávání (sloupec Google Play vzdělávání). Navíc jsou do srovnání přidány údaje, které odpovídají celkové distribuci subverzí OS Android na trhu (sloupec Android celkově). Data jsou aktuální k 28.8.2015.

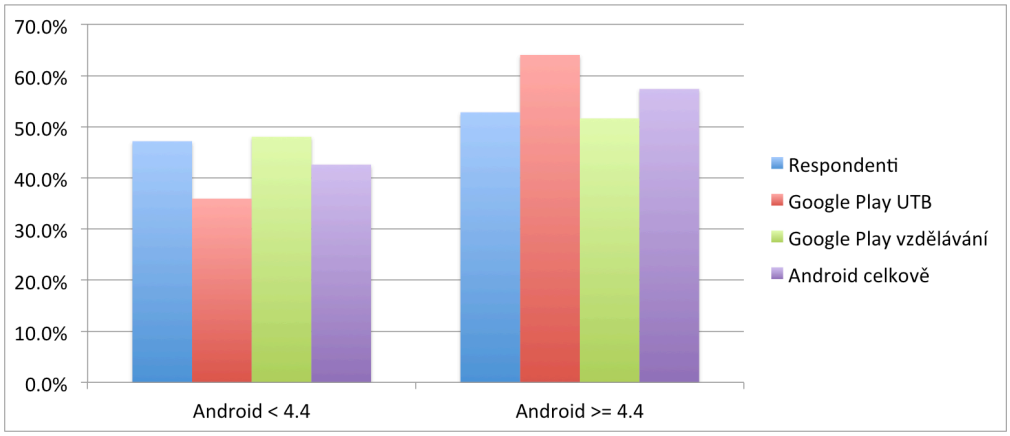
Z výsledků je patrné, že se distribuce subverzí OS Android mezi respondenty dotazníku významněji různí od poměrů odpovídajících všem aktivním uživatelům aplikace UTB zejména v případě verze 4.0.3.–4.0.4, 4.1 a 5.1, a to vyšším podílem zastoupení. Naopak v podstatně nižším poměru, než je jejich celkové zastoupení v rámci distribuce aplikace UTB, odpovídali uživatelé se subverzí 4.4. Poměry respondentů ostatních verzí korespondují s celkovými poměry uživatelů aplikace UTB.

Pokud se zaměříme jen na statistiku subverzí aplikace UTB (sloupec Google Play UTB) a celkovou statistiku distribuce subverzí platformy Android, lze říci, že poměry si v podstatě odpovídají a liší se pouze v jednotkách procent. Lze tedy konstatovat, že výsledky získané v případě reálné mobilní aplikace v podmínkách českého trhu odpovídají celosvětové statistice distribuce verzí OS Android.



Obrázek 50: Srovnání distribuce subverzí OS Android mezi respondenty dotazníkového průzkumu, statistikami aplikace UTB na Google Play, statistikami v kategorii vzdělávání na Google Play a celkovými statistikami platformy Android

Zajímavé je také srovnání poměru starších (<4.4) a novějších (>=4.4) verzí OS Android. To je zachyceno na grafu (viz obrázek 51). Data jsou opět k dispozici pro kategorii respondentů dotazníku, uživatelů aplikace UTB dle Google Play a konečně také celkové distribuce Android OS. Z grafu je zřejmé, že rozložení subverzí respondentů dotazníku v podstatě odpovídá celosvětové distribuci Android OS. Liší se pouze v jednotkách procent, jelikož respondenti častěji uváděli starší verze. Srovnáme-li však celkové rozložení subverzí mezi reálnými uživateli aplikace (sloupec Google Play UTB), zaznamenáme o zhruba 5% vyšší podíl novějších zařízení, než je celosvětovým trendem.



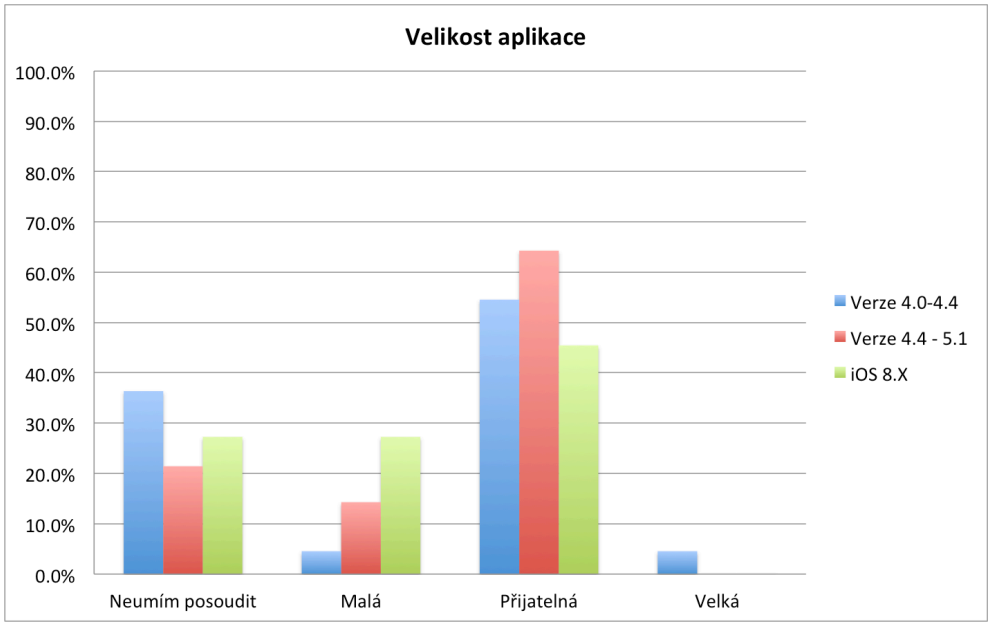
Obrázek 51: Distribuce subverzí OS Android < 4.4. a >= 4.4 dle kategorií viz výše

Výsledky dotazníku – evaluace úspěšnosti aplikace UTB

V rámci evaluace úspěšnosti mobilní aplikace, která by měla vypovídat o míře, do jaké splňuje uživatelské nároky na ovládání a UX, byly reálnými uživateli aplikace UTB subjektivně hodnoceny tyto vlastnosti aplikace: rychlost spuštění, velikost aplikace, plynulost animací, navigace v rámci aplikace a celkové chování aplikace. Výše uvedené hodnocení bylo rozděleno dle verze operačního systému. Zvlášť byly vyhodnoceny odpovědi uživatelů OS Android verze 4.0–4.4 (starší verze) a verze 4.4–5.1 (novější verze) a také OS iOS. Výsledky týkající se platformy iOS jsou limitovány a zkresleny nedostatečnou velikostí vzorku respondentů (způsobeno nízkou mírou penetrace iOS zařízení na českém trhu). Rozdělení na subverze v případě OS Android vychází z klíčových odlišností běhových prostředí těchto verzí, zjištěných a popsaných v kapitole 5.6.2, a také z faktu, že pro tyto starší verze je použito společné běhové prostředí Crosswalk, kdežto verze novější již používají standardní běhové prostředí Apache Cordova.

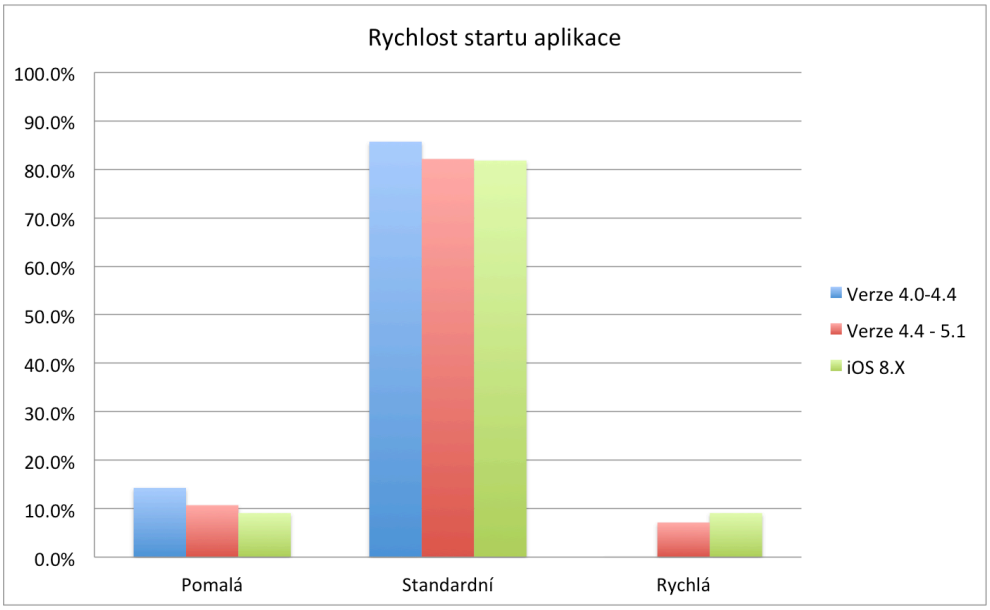
Graf na obrázku 52 zachycuje hodnocení velikosti aplikace. Uživatelský průzkum nepotvrdil předpoklad, že aplikace určená pro starší verze OS Android, která obsahuje nadstandardní běhové prostředí Crosswalk a tudíž je datově objemnější, bude hodnocena uživateli jako datově velká. Tento problém, který se zdá být o to palčivější právě na starších zařízeních bez velké interní paměti, uživatelé dle průzkumu v podstatě nevnímají. Na platformě Android byla hodnocena velikost aplikace jak uživateli starších verzí OS (a objemnější aplikace), tak uživateli novějších verzí OS (a datově velmi malé verze aplikace) většinou jako přijatelná (starší verze asi 54% a novější verze asi 64% odpovědí). Více než třetina uživatelů starších Android zařízení však uvedla, že velikost neumí posoudit. V případě uživatelů novějších zařízení je to asi 20%. Datově objemnější starší verzi aplikace hodnotily pouhé 4% uživatelů a to jako velkou.

Z výše uvedeného tedy vyplývá, že benefity, které v případě starších zařízení nadstandardní běhová prostředí přinášejí, převažují nad faktem, že výsledná aplikace je datově objemná. Koncoví uživatelé tomu totiž ve většině případů nepřikládají váhu.



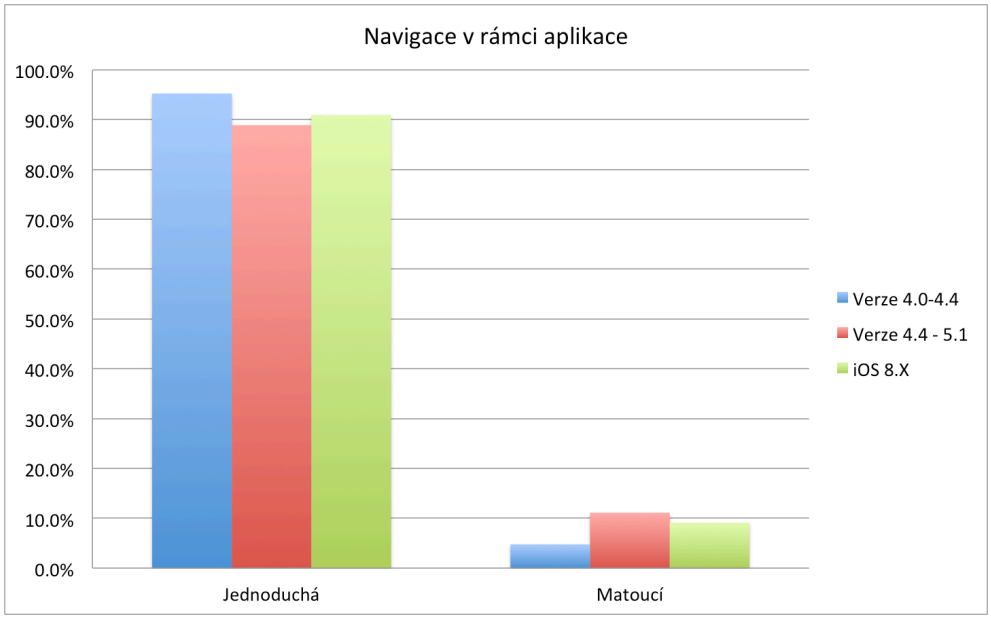
Obrázek 52: Hodnocení velikosti aplikace UTB

Rychlost startu mobilní aplikace UTB byla orientačně změřena na zařízeních Samsung Galaxy Note 10.1 (Android OS 4.1.2) a Motorola Nexus 6 (Android OS 5.1.1) - v obou případech trval start aplikace zhruba 3,5 sekundy. Pomalý start aplikace může negativně ovlivnit to, zda uživatel bude chtít aplikaci dále používat či nikoliv, a proto jde o jeden ze sledovaných parametrů UX. V rámci dotazníkového průzkumu odpovědělo více než 80% uživatelů v případě všech subverzí operačních systémů, že rychlost startu aplikace je standardní. Jako rychlou ji označilo 7% uživatelů OS Android verze 4.4–5.1 a 9% uživatelů OS iOS. Pomalou rychlost startu uvedlo 14% uživatelů platformy Android verze 4.0–4.4, necelých 11% uživatelů novějších verzí OS Android a 9% uživatelů platformy iOS.



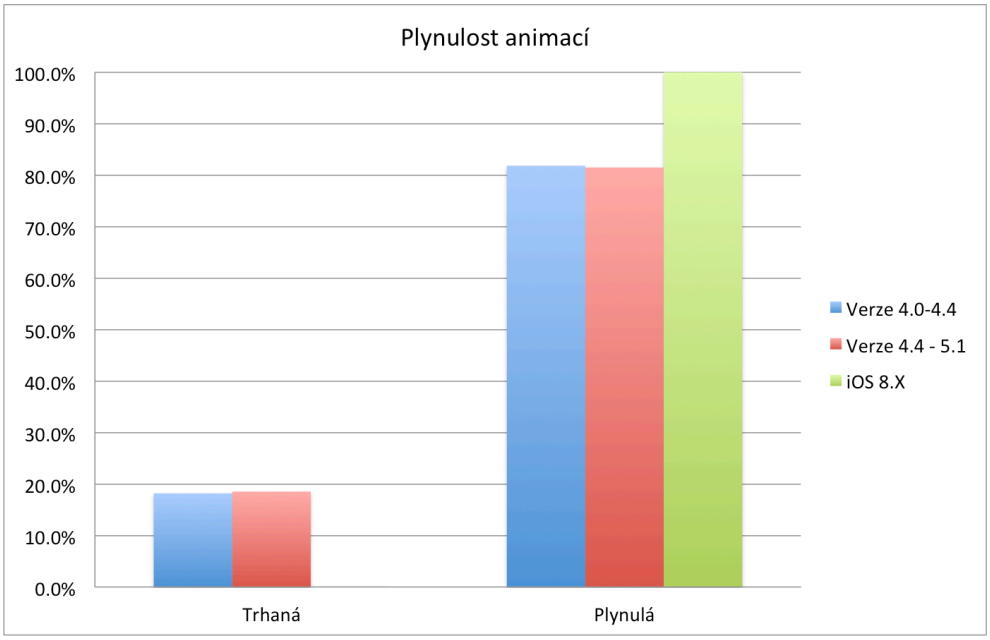
Obrázek 53: Hodnocení rychlosti startu aplikace UTB

Dalším parametrem přímo ovlivňujícím UX je kvalita navigace v rámci aplikace. Při použití vývojového UI frameworku pro návrh uživatelského rozhraní může být do jisté míry závislá na tom, jaké ovládací prvky a jaký způsob navigace framework nabízí. V aplikaci UTB byl použit navigační pruh pro hlavní menu tvořený objektem Navbar frameworku Chocolate Chip UI. Pro platformu Android byl však mírně upraven, aby se dle zvyklostí verze 5 skrýval na obrazovkách druhé úrovně. Výsledek dotazníkového průzkumu je uspokojivý, jelikož v průměru označilo zhruba 90% uživatelů navigaci za jednoduchou, tudíž lze usuzovat, že funguje dle jejich očekávání, které je utvářeno zejména vzory jiných, zpravidla nativních aplikací.



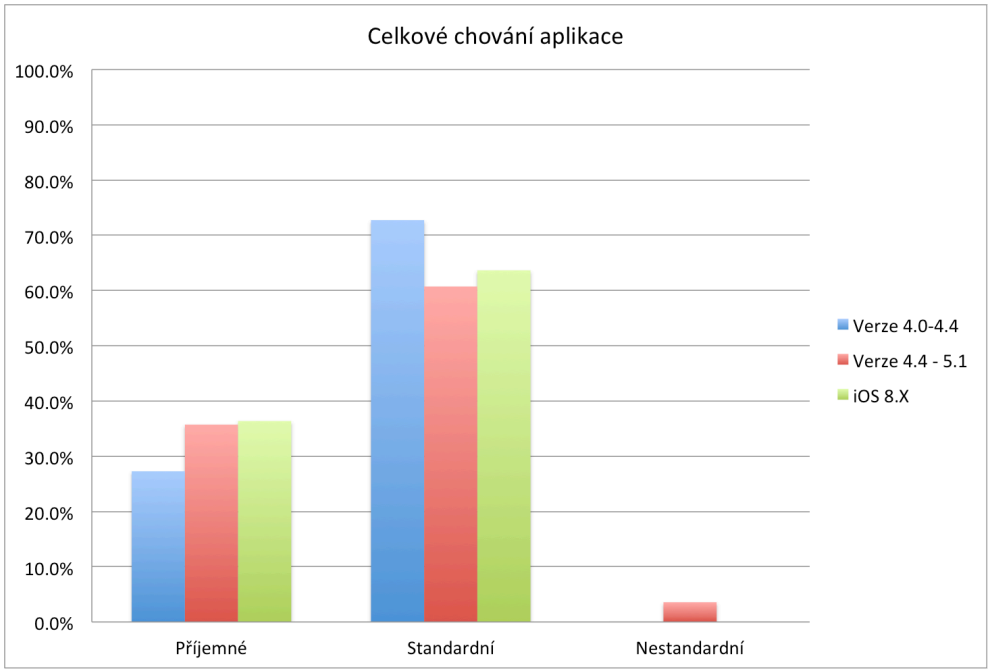
Obrázek 54: Hodnocení navigace v aplikaci UTB

Nízká úroveň plynulosti animací je jedním z nejčastějších zobrazovacích problémů WHMA a to zejména na starších zařízeních platformy Android. Dotazníkový průzkum ukázal poměrně optimistické výsledky, kdy více než 80% uživatelů platformy Android označilo animace za plynulé a to jak v případě starších verzí, tak v případě verzí novějších. Implementaci běhového prostředí Crosswalk lze tedy označit za šťastný krok, díky němuž se podařilo sjednotit vlastnosti běhových prostředí do takové míry, že i starší verze Android OS vykazují velmi dobré výsledky. Respondenti používající iOS pak shodně uváděli, že animace jsou plynulé.



Obrázek 55: Hodnocení plynulosti animací v aplikaci UTB

V rámci dotazníkového průzkumu byl také položen dotaz na celkové chování aplikace. Uživatel tak mohl odhodnotit celkový dojem z interakce s aplikací a to jako příjemný, standardní či nestandardní. Výsledky prokázaly, že vhodným postupem tvorby WHMA v kombinaci s HTML5 frameworkem lze vytvořit mobilní aplikaci, která poskytne dostatečnou míru uživatelského zážitku tak, aby byla srovnatelná s aplikacemi nativními. Na novějších verzích platformy Android a na platformě iOS označilo více než 35% uživatelů chování aplikace jako příjemné. V případě starších verzí OS Android to pak bylo takřka 28%. Většina uživatelů uvedla, že aplikace má standardní chování, čili srovnatelné s běžnými aplikacemi dostupnými na dané platformě. Jen necelá 4% uživatelů OS Android 4.4–5.1 označilo chování za nestandardní.



Obrázek 56: Hodnocení celkového chování aplikace UTB

5.9.2 Vyhodnocení úspěšnosti aplikace dle statistik Google Play

Ve statistikách obchodu s mobilními aplikacemi Google Play lze také nalézt několik zajímavých parametrů, které do jisté míry vypovídají o úspěšnosti vývojového projektu. Jsou to statistiky chyb či počtu aktivních instalací.

Statistiky chyb

Jednou z priorit vývojové metody navržené v rámci této práce je stabilita a spolehlivost koncové aplikace. Statistiky Google Play nám umožňují sledovat dva základní parametry stability aplikace – tzv. Selhání a chyby ANR (Application Not Responding – Aplikace neodpovídá). Selhání je záznam o uživatelem nahlášeném pádu aplikace (pád neboli selhání aplikace vyvolá systémový dialog, pomocí nějž lze vývojáře o události informovat). Chyba ANR nastane ve chvíli, kdy aplikace neodpovídá operačnímu systému a je vyvolán systémový dialog dotazující se uživatele, zda má být ukončena. Oba parametry jsou přímo spjaté s nativní částí kódu (obalem pro běhové prostředí webových technologií) a nesouvisí tak přímo s webovou aplikační logikou. Pokud ovšem nastane logická chyba v aplikační logice, která zapříčiní pád či zamrznutí nativní části, dojde k vyvolání jedné z výše uvedených chyb.

Dle statistiky Selhání a chyb ANR mobilní aplikace UTB, nedošlo od prvotní publikace u žádné z verzí k výskytu kteréhokoliv výše zmíněného incidentu (obrázek 57).

Obrázek 57: Stav selhání a chyb ANR aplikace UTB k 25. 8. 2015 dle play.google.com

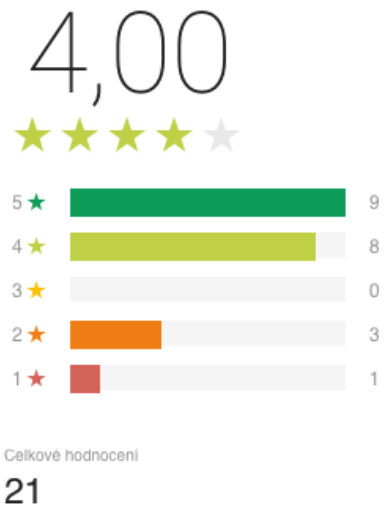
Statistiky počtu aktivních instalací

Neúspěch projektu je zpravidla dán neoblíbeností u koncových uživatelů. Uživatelé mnohdy aplikaci nainstalují, ale pokud nesplní jejich očekávání, často ji ze zařízení brzy odstraní. Aplikace UTB má co se počtu aktuálních instalací týče průměrně rostoucí trend, což svědčí o řadě spokojených uživatelů. V grafu na obrázku 58 jsou patrné 2 skokové nárůsty. První byl způsoben zveřejněním beta verze na profilu Studentské Unie UTB portálu facebook.com a druhý zveřejněním taktéž na portále facebook.com, tentokráte na profilu UTB ve Zlíně a dále na webové stránce univerzity. Slabý pokles počtu aktivních instalací způsobil pouze příchod akademických prázdnin, kdy je využití aplikace z řad studentů a zaměstnanců minimální. Aktuálně (ke dni 28.8.2015) je aplikace UTB instalována na 498 zařízeních.

Obrázek 58: Statistika aktuálního počtu instalací aplikace UTB dle play.google.com (k 28. 8. 2015)

Uživatelská hodnocení

Pro poskytnutí zpětné vazby vývojářům aplikace či ostatním uživatelům slouží také uživatelská hodnocení. Aplikace UTB obdržela na Google Play k 25. 8. 2015 21 uživatelských hodnocení. Průměrné hodnocení je 4 body z možných 5-ti, což je velmi dobrý výsledek. Byly zaznamenány také negativní hodnocení (3x 2 body a 1x 1 bod) a to během výpadku webových služeb IS-STAG provozovaných třetí stranou, kdy se nebylo možno v aplikaci přihlásit k uživatelskému účtu.



Obrázek 59: Uživatelská hodnocení aplikace UTB na Google Play k 28. 8. 2015

ZÁVĚR

Disertační práce sestavuje reprodukovatelnou metodu zaměřující se na technologické postupy pro vývoj webových hybridních mobilních aplikací. Vývojová metoda klade důraz na možnost sestavení aplikace pro více platforem a dále na stabilitu a spolehlivost běhu a to i na starších zařízeních objevujících se na trhu.

Teoretická část nejprve představuje mobilní operační systémy a důležitá specifika vývoje mobilních aplikací, dále je zde navržena klasifikace vývojových metod dle použitých technologií, která je v současné literatuře nejednotná. Dílčí kapitola je také věnována teoretickému úvodu do webových hybridních mobilních aplikací, na které se další text práce zaměřuje.

Experimentální část práce studuje především reálné problémy vedoucí k prodlužování vývoje a zvyšování nákladů a předkládá řešení, jež umožňují dosáhnout časově a finančně vyváženého vývojového procesu. Přináší charakteristiku a doporučení metody implementace uživatelského rozhraní dle použitých technologií a navrhuje sadu hodnoticích parametrů pro váženou multikriteriální analýzu vhodnou pro kvalifikovaný výběr vývojového UI frameworku. Stanovuje také metody výkonnostních testů HTML5 frameworků pro návrh uživatelského rozhraní a předkládá výsledky testování několika vybraných vývojových rámců v praxi využitelné v procesu rozhodování. Pro měření byly vybrány parametry přímo ovlivňující uživatelský zážitek. Další část se zaměřuje na zobrazovací problémy způsobené použitým běhovým prostředím v rámci WHMA. Ty tvoří nejčastější příčinu neúspěchu mobilních aplikací založených na webových technologiích. Z toho důvodu byla provedena analýza běhového prostředí Apache Cordova/Phonegap a identifikace zobrazovacích problémů a zejména problematických subverzí operačních systémů. Výsledkem analýzy je návrh řešení v podobě implementace nadstandardního běhového prostředí. V rámci verifikace byla měřena úroveň podpory HTML5 vlastností, výkon JavaScriptu a grafických operací na testovacích aplikacích implementujících odlišná běhová prostředí. Na základě výsledků je pak vhodné běhové prostředí doporučeno.

Postupy a doporučení navržené v rámci experimentální části práce byly použity v procesu reálného vývoje webových hybridních aplikací Retigo Vision a UTB. Vyhodnocení úspěšnosti navržených postupů zejména v oblasti dosažení dostatečné míry uživatelského zážitku pak bylo provedeno na základě dotazníkového průzkumu uživatelů mobilní aplikace UTB. 64% uživatelů hodnotilo celkové chování aplikace jako standardní, takřka 33% dokonce jako příjemné, oproti pouhým 3% respondentů, kteří chování označili jako nestandardní. Toto zjištění prokazuje, že vhodnou volbou postupů a softvérových nástrojů lze vytvořit webovou hybridní mobilní

aplikaci, která nebude uživateli vnímána jako cizorodá a uživatelský zážitek bude srovnatelný s aplikací nativní.

Výše uvedený aplikovaný výzkum znamená posun poznání v oblasti vývoje WHMA a jejich implementace na mobilních zařízeních, zejména v oblasti poznání technologií vhodných k adaptaci uživatelského rozhraní na cílovou platformu tak, aby míra uživatelského zážitku byla co nejvyšší.

Přínosem pro praxi je významné zefektivnění procesů vývoje webových hybridních mobilních aplikací především v oblasti minimalizace zobrazovacích problémů a následného zkrácení testovacích a opravných cyklů.

Závěry jednotlivých částí práce postupně potvrdily všechny stanovené hypotézy.

- „Lze sestavit sadu kritérií a hodnoticí metodiku univerzálně použitelnou pro výběr vývojového UI frameworku se zaměřením na konkrétní aplikaci."

Tato hypotéza byla potvrzena v rámci kapitoly 5.4, jejíž výsledky byly přijaty i vědeckou komunitou v rámci publikace na mezinárodní konferenci.

- „Moderní HTML5 UI frameworky lze díky volbě vhodného běhového prostředí použít i na starších zařízeních nepodporujících nové HTML5 standardy."

Tato hypotéza byla potvrzena výsledky měření v kapitole 5.7 a dále také při praktické implementaci mobilní aplikace UTB.

- „V rámci webových hybridních mobilních aplikací lze použít technologie HTML5 k vytvoření adaptivního uživatelského rozhraní, jež bude imitovat nativní uživatelské rozhraní platformy, pro kterou byla aplikace sestavena."

Výše uvedená hypotéza byla potvrzena dotazníkovým průzkumem cíleným na uživatele mobilní aplikace UTB, jehož výsledky jsou uvedeny v kapitole 5.9. Zejména hodnocení celkového chování aplikace prokázalo, že chování aplikace UTB považují uživatelé v naprosté většině za standardní či příjemné, tudíž srovnatelné s nativními aplikacemi, na které jsou z prostředí dané platformy zvyklí.

- „Pomocí vhodného výběru technologií pro vývoj hybridní mobilní aplikace lze vytvořit dostatečně výkonnou aplikaci nabízející kvalitní uživatelský zážitek skrze různé platformy a zařízení."

Tato hypotéza je v praxi potvrzena reálnou implementací mobilní aplikace UTB, kterou má dle statistik serveru Google Play aktuálně nainstalovánu takřka 500 uživatelů. Také na základě výsledků dotazníkového průzkumu lze říci, že metoda vývoje webových hybridních aplikací je životaschopným multiplatformním řešením vhodným pro v této práci definované druhy

mobilních aplikací. Navržená doporučení navíc pomáhají předejít implementačním a testovacím problémům a oproti nativnímu vývoji poskytují značnou časovou a finanční úsporu.

6 LITERATURA A OSTATNÍ ZDROJE

1. *eMarketer: Smartphone Users Worldwide Will Total 1.75 Billion in 2014*. 2014. s. http://www.emarketer.com/Article/Smartphone-Users-Worldwide-Will-Total-175-Billion-2014/1010536.
2. *American Dialect Society: "App" voted 2010 word of the year by the American Dialect Society (UPDATED)*. 2011. s. http://www.americandialect.org/app-voted-2010-word-of-the-year-by-the-american-dialect-society-updated.
3. *StatCounter Global Stats - Browser, OS, Search Engine including Mobile Usage Share*: StatCounter, 1999-2015. http://gs.statcounter.com/ - mobile+tablet-os-CZ-monthly-201404-201504-bar.
4. GOKHALE, P.; SINGH, S. Multi-platform strategies, approaches and challenges for developing mobile applications. In *Circuits, Systems, Communication and Information Technology Applications (CSCITA), 2014 International Conference on*. 2014. p. 289-293.
5. *W3C: HTML5 A vocabulary and associated APIs for HTML and XHTML*. 2014. s. http://www.w3.org/html/wg/drafts/html/CR/.
6. *iOS Human Interface Guidelines*: Apple Inc., 2015. https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/MobileHIG/.
7. *Design | Android Developers*: Google Inc., 2015. https://developer.android.com/design/index.html.
8. *User interface for Windows Phone 8*: Microsoft, 2015. https://msdn.microsoft.com/en-us/library/windows/apps/ff967556(v=vs.105).aspx.
9. MIMS, C. *Rise of the "Hybrid" Mobile App*, 2011. http://www.technologyreview.com/news/424328/rise-of-the-hybrid-mobile-app/.
10. WARREN, C. *The Pros and Cons of Cross-Platform App Design*, 2012. http://mashable.com/2012/02/16/cross-platform-app-design-pros-cons/.
11. *WebView for Android*: Google Inc., 2015. https://developer.chrome.com/multidevice/webview/overview.
12. *WebKit Framework Reference*: Apple Inc., 2015. https://developer.apple.com/library/ios/documentation/Cocoa/Reference/WebKit/ObjC_classic/index.html#//apple_ref/doc/uid/TP30000745.
13. GERBASI, R. *Apple Shows Love for HTML5 with iOS 8*, 2015. https://http://www.sencha.com/blog/apple-shows-love-for-html5-with-ios-8/.
14. *WKWebView*: Telerik, 2015. http://plugins.telerik.com/cordova/plugin/wkwebview.

15. ABDULLAH, S. *WKWebView and Apache Cordova*, 2015. https://shazronatadobe.wordpress.com/2015/03/03/wkwebview-and-apache-cordova/.
16. EBERHARDT, C.; PRICE, C. *PropertyCross*, 2015. https://developer.apple.com/library/ios/documentation/Cocoa/Reference/WebKit/ObjC_classic/index.html#//apple_ref/doc/uid/TP30000745.
17. FALK, M.; BÄDE, S. *Mobile Framework Comparison Chart*, 2015. http://mobile-frameworks-comparison-chart.com.
18. NASEHI, S. M.; SILLITO, J.; MAURER, F. et al. What makes a good code example?: A study of programming Q&A in StackOverflow. In Trento: IEEE, 2012. p. 25 - 34.
19. JAHNS, R.-G. *The 9 barriers for Cross-Platform App Development Tools to take off – Interview with Martin Wilson CEO of Appear*: research2guidance, 2014. http://research2guidance.com/the-9-barriers-for-cross-platform-app-development-tools-to-take-off-interview-with-martin-wilson-ceo-of-appear/.
20. POGORZELSKA, Z. *Cross-Platform Tools Benchmarking 2014 results: a big move | research2guidance*, 2014. http://research2guidance.com/cpt-benchmarking-2014-results-cross-platform-tools-are-making-a-big-progress-in-the-app-developer-community/.
21. POGORZELSKA, Z. *App developers say Cross-Platform Tools are better than native | research2guidance*, 2014. http://research2guidance.com/only-5-of-mobile-app-developers-who-use-cross-platform-tools-say-native-app-development-brings-more-value-formoney/.
22. RIVERA, J.; MEULEN, R. *Gartner Says by 2016, More Than 50 Percent of Mobile Apps Deployed Will be Hybrid*, 2015. http://www.gartner.com/newsroom/id/2324917.
23. KAUR, P.; SHARMA, S. Google Android a Mobile Platform: A Review. In Chandigarh: IEEE, 6-8 March 2014. p. 1-5.
24. GRONLI, T.-M.; HANSEN, J.; GHINEA, G. et al. Mobile Application Platform Heterogeneity: Android vs Windows Phone vs iOS vs Firefox OS. In Victoria, BC, Canada: 2014. p. 635 - 641.
25. GAVALAS, D.; ECONOMOU, D. Development Platforms for Mobile Applications: Status and Trends. In IEEE, 2011. p. 77-86.
26. GOADRICH, M. H., ROGERS, M.P. Smart smartphone development: iOS versus Android. In New York: ACM, 2011. p. 607-612.
27. SUYESH, A.; ARIANIT, K. Cross-Platform Mobile Development: Challenges and Opportunities. In Springer International Publishing, 2014. p. 219-229.

28. DALMASSO, I.; DATTA, S. K.; BONNET, C. et al. Survey, Comparison and Evaluation of Cross Platform Mobile Application Development Tools. In Sardinia: IEEE, 2013. p. 323 - 328.
29. *Titanium Mobile Application Development*: Appcelerator Inc., 2008-2014. http://www.appcelerator.com/titanium/.
30. PALMIERI, M.; INNOVATION DESIGN & ENG, M. U., VASTERAS, SWEDEN; SINGH, I. et al. Comparison of cross-platform mobile development tools. In Berlin: IEEE, 2012. p. 179 - 186.
31. OHRT, J.; HAMBURG UNIV. OF TECHNOL., H., GERMANY; TURAU, V. Cross-Platform Development Tools for Smartphone Applications. In IEEE, 2012. p. 72-79.
32. HEITKÖTTER, H.; HANSCHKE, S.; MAJCHRZAK, T. A. Evaluating Cross-Platform Development Approaches for Mobile Applications. In *8th International Conference, WEBIST 2012, Porto, Portugal, April 18-21, 2012, Revised Selected Papers*. Porto: Springer-Verlag Berlin Heidelberg, 2013. p. 120-138.
33. HUI, N. M.; CHIENG, L. B.; TING, W. Y. et al. *Cross-Platform Mobile Applications for Android and iOS.* New York: Ieee, 2013. 2013 6th Joint Ifip Wireless and Mobile Networking Conference. ISBN 978-1-4673-5616-9; 978-1-4673-5615-2.
34. ACHILLEOS, A. P.; KAPITSAKI, G. M. Enabling Cross-Platform Mobile Application Development: A Context-Aware Middleware. In Benatallah, B. et al. *Web Information Systems Engineering, Pt Ii*. Cham: Springer Int Publishing Ag, 2014. vol. 8787, p. 304-318. ISBN 0302-9743
978-3-319-11746-1; 978-3-319-11745-4.
35. CHAN, G. K.; KIONG, T. K.; NARAYANAN, A. S. et al. MEETING ROOM - a Secure Multi-Access, Cross-Platform Telemedicine Application. *2013 Australasian Telecommunication Networks and Applications Conference (Atnac),* 2013, p. 201-206.
36. PARSONS, J. J.; OJA, D. *New Perspectives on Computer Concepts 2014: Brief.* Cengage Learning, 2013. ISBN 9781285097695.
37. *About the Virtual Memory System*: Apple Inc., 2015. https://developer.apple.com/library/mac/documentation/Performance/Conceptual/ManagingMemory/Articles/AboutMemory.html.
38. *Tips for Allocating Memory*: Apple Inc., 2015.
39. *Processors: Computer vs mobile*: Pronto Marketing, 2015. http://www.techadvisory.org/2013/12/processors-computer-vs-mobile/.

40. *The App Life Cycle*: Apple Inc., 2015. https://developer.apple.com/library/ios/documentation/iPhone/Conceptual/iPhoneOSProgrammingGuide/TheAppLifeCycle/TheAppLifeCycle.html.
41. *Activity | Android Developers*: Android Inc., 2015. http://developer.android.com/reference/android/app/Activity.html.
42. *UIApplicationDelegate Protocol Reference*: Apple Inc., 2015. https://developer.apple.com/library/ios/documentation/UIKit/Reference/UIApplicationDelegate_Protocol/.
43. *Application.Suspending event - Windows app development*: Microsoft, 2015. https://msdn.microsoft.com/en-us/library/windows/apps/windows.ui.xaml.application.suspending.aspx?cs-save-lang=1&cs-lang=csharp#code-snippet-1.
44. *App Sandboxing - Apple Developer*: Apple Inc., 2015. https://developer.apple.com/app-sandboxing/.
45. *Security Tips | Android Developers*: Android Inc., 2015. http://developer.android.com/training/articles/security-tips.html.
46. *Security for Windows Phone 8*: Microsoft, 2015. https://msdn.microsoft.com/en-us/library/windows/apps/ff402533(v=vs.105).aspx.
47. *Manifest.permission | Android Developers*: Android Inc., 2015. http://developer.android.com/reference/android/Manifest.permission.html.
48. HOFFMANN, C. *iOS Has App Permissions, Too: And They're Arguably Better Than Android's*, 2013. http://www.howtogeek.com/177711/ios-has-app-permissions-too-and-theyre-arguably-better-than-androids/.
49. *iOS Human Interface Guidelines: Animation*: Apple Inc., 2015. https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/MobileHIG/Animation.html#//apple_ref/doc/uid/TP40006556-CH57-SW1.
50. *Top 14 Mobile & Tablet Screen Resolutions in Czech Republic from Apr 2014 to Apr 2015*: StatCounter, 2015. http://gs.statcounter.com/ - mobile+tablet-resolution-CZ-monthly-201404-201504-bar.
51. *02 | Jaká je rychlost připojení?* : O2 Czech Republic a.s., 2015. http://www.o2.cz/osobni/casto-kladene-otazky/172719-x3.html.
52. *Areppim: information, pure and simple: Mobile OS | worldwide market share over time.* 2014. s. http://stats.areppim.com/stats/stats_mobiosxtime.htm.
53. *Android Wear*: Android Inc., 2015. http://www.android.com/wear/.

54. INC., A. *Dashboards | Android Developers*, 2015. https://developer.android.com/about/dashboards/index.html?utm_source=suzunone.
55. *App Store Distribution*: Apple Inc., 2015. https://developer.apple.com/support/appstore/.
56. JUOZAITYTĖ, L. *AdDuplex Windows Phone Device Statistics Report for April, 2015*, 2015. http://blog.adduplex.com/2015/04/adduplex-windows-phone-device.html.
57. AMATYA, S.; KURTI, A. Cross-Platform Mobile Development: Challenges and Opportunities. In Trajkovik, V. et al. *ICT Innovations 2013*. Springer International Publishing, 2014. vol. 231, 21, p. 219-229. ISBN 978-3-319-01465-4.
58. ANDRE, C.; BRIAN, L. Mobile application development: web vs. native. *Commun. ACM,* 2011, vol. 54, no. 5, p. 49-53. ISSN 0001-0782.
59. *Download Android Studio and SDK Tools | Android Developers*: Android Inc., 2015. https://developer.android.com/sdk/index.html.
60. *Android NDK | Android Developers*, 2015. https://developer.android.com/tools/sdk/ndk/index.html.
61. GANG, W.; KONARK, G.; MANISH, M. et al. *Wisdom in the social crowd: an analysis of quora: Proceedings of the 22nd international conference on World Wide Web.* Rio de Janeiro, Brazil: International World Wide Web Conferences Steering Committee, 2013. s.
62. *Propojení účtu vývojáře Google Play s účtem Google Payments Merchant Center - Nápověda Developer Console*: Google, 2015. https://support.google.com/googleplay/android-developer/answer/3092739?hl=cs.
63. BALDWIN, P. *OSx86: Creating a Hackintosh.* Wiley, 2010. ISBN 9780470881705.
64. *Average App Store Review Times*: Shiny Development, 2015. http://appreviewtimes.com.
65. *Account types, locations, and fees - Windows app development*: Microsoft, 2015. https://msdn.microsoft.com/cs-CZ/library/windows/apps/jj863494.aspx#account_markets.
66. JITENDRA, M. Enterrise Mobility - A Future Transformation Strategy for Organizations. In Wyld, D. C. et al. Springer, 2012. p. 559-567.
67. VIKTOR, K.; PAUL, B.; SILVIA, S. et al. *Flexible development of variable software features for mobile business applications: Proceedings of the 17th International Software Product Line Conference co-located workshops.* Tokyo, Japan: ACM, 2013. s.
68. RUKH, H. S.; STEFAN, E.; ACHIM, E. Developing Mobile Apps Using Cross-Platform Frameworks: A Case Study. In Kurosu, M. *Human-Computer Interaction. Human-Centred Design Approaches, Methods, Tools, and Environments*. Springer Berlin Heidelberg, 2013. vol. 8004, 41, p. 371-380. ISBN 978-3-642-39231-3.

69. SINGH, K.; KRISHNASWAMY, V. Building Communicating Web Applications Leveraging Endpoints and Cloud Resource Service. In *Cloud Computing (CLOUD), 2013 IEEE Sixth International Conference on*. 2013. p. 486-493.
70. RIBEIRO, A.; DA SILVA, A. R. Survey on Cross-Platforms and Languages for Mobile Apps. In *Quality of Information and Communications Technology (QUATIC), 2012 Eighth International Conference on the*. 2012. p. 255-260.
71. *About Apache Cordova™*: The Apache Software Foundation, 2015. https://cordova.apache.org/.
72. *PhoneGap Home*: Adobe Systems Inc., 2015. http://phonegap.com/.
73. YANYAN, Z.; BALDWIN, J.; ANTUNNA, L. et al. Tradeoffs in cross platform solutions for mobile assistive technology. In *Communications, Computers and Signal Processing (PACRIM), 2013 IEEE Pacific Rim Conference on*. 2013. p. 330-335.
74. SPYROS, X.; STELIOS, X. *A comparative analysis of cross-platform development approaches for mobile applications: Proceedings of the 6th Balkan Conference in Informatics*. Thessaloniki, Greece: ACM, 2013. s.
75. *rhomobile/rhodes - GitHub*: GitHub, Inc., 2015. https://github.com/rhomobile/rhodes.
76. *Qt | Cross-platform application & UI development framework*: The Qt Company, 2015. http://www.qt.io.
77. *applause/applause · GitHub*: Github Inc., 2015. https://github.com/applause/applause.
78. *The iphonical Open Source Project on Open Hub*: Black Duck Software, Inc., 2015. https://http://www.openhub.net/p/iphonical.
79. *Build apps in C# for iOS, Android and Windows Phone. - Xamarin*: Xamarin Inc., 2015. http://xamarin.com/platform.
80. PUDER, A.; ANTEBI, O. Cross-Compiling Android Applications to iOS and Windows Phone 7. *Mobile Networks and Applications,* 2013, vol. 18, no. 1, p. 3-21. ISSN 1383-469X.
81. *Common Language Runtime (CLR)*: Microsoft, 2015. https://msdn.microsoft.com/cs-cz/library/8bs2ecf4(v=vs.110).aspx.
82. *MonoDevelop: Mono For Android*. 2011. s. http://monodevelop.com/Download/Mono_For_Android.
83. *InfoQ: MonoTouch: .NET Development for the iPhone*. 2009. s. http://www.infoq.com/articles/monotouch-introduction.
84. ANDERSON, B. *Introduction to Xamarin*, 2013.
85. *The Mono Runtime | Mono*: Mono Project, 2015. http://www.mono-project.com/docs/advanced/runtime/.

86. *AOT | Mono*: Mono Project, 2015. http://www.mono-project.com/docs/advanced/aot/.
87. *Xamarin.Android 5.1 - Xamarin*: Xamarin Inc., 2015. http://developer.xamarin.com/releases/android/xamarin.android_5/xamarin.android_5.1/ - AOT_Support.
88. *Adobe Phonegap Build*: Adobe Systems Incorporated, 2015. https://build.phonegap.com.
89. *Trigger.io - mobile platform for web developers*: Triggercorp, 2015. https://trigger.io.
90. VALA, R. *Mobilní aplikace UTB*, 2015. http://mobapp.utb.cz.
91. *SOFTWARE LICENSE AGREEMENT FOR OS X MAVERICKS*: Apple Inc., 2014. http://images.apple.com/legal/sla/docs/OSX109.pdf.
92. HENNING, H.; SEBASTIAN, H.; A., M. T. Evaluating Cross-Platform Development Approaches for Mobile Applications. In *8th International Conference, WEBIST 2012, Porto, Portugal, April 18-21, 2012, Revised Selected Papers*. Porto: Springer-Verlag Berlin Heidelberg, 2013. p. 120-138.
93. *Windows Phone | Dev Center: User interface for Windows Phone 8*. 2014. s. http://msdn.microsoft.com/en-us/library/windowsphone/develop/ff967556(v=vs.105).aspx.
94. *Android Developers: User Interface*. 2014. s. http://developer.android.com/guide/topics/ui/index.html.
95. *UIKit Framework Reference - iOS Developer Library*: Apple Inc., 2014. https://developer.apple.com/library/ios/documentation/uikit/reference/uikit_framework/_index.html.
96. BORCHERS, J. *A Pattern Approach to Interaction Design*. New York: John Wiley & Sons, 2001. ISBN ISBN: 978-0-471-49828-5.
97. GLEN, F. *Figuring the costs of custom mobile business app development*, 2014. http://www.formotus.com/14018/blog-mobility/figuring-the-costs-of-custom-mobile-business-app-development.
98. *Apache Cordova API Documentation*: The Apache Software Foundation, 2015. http://cordova.apache.org/docs/en/5.0.0/guide_platforms_index.md.html - Platform%20Guides.
99. BOEHM, B.; ABTS, C.; CHULANI, S. Software development cost estimation approaches—A survey. *Annals of software engineering,* 2000, vol. 10, no. 1-4, p. 177-205. ISSN 1022-7091.
100. KAN, S. H. *Metrics and Models in Software Quality Engineering*. Addison-Wesley, 2003. ISBN 9780201729153.

101. PFLEEGER, S. L.; WU, F.; LEWIS, R. et al. *Software Cost Estimation and Sizing Methods: Issues, and Guidelines.* Rand Corporation, 2005. ISBN 9780833037138.
102. *App Store Review Guidelines*: Apple Inc., 2015. https://developer.apple.com/app-store/review/guidelines/#user-interface.
103. *Retigo Vision on the App Store on iTunes*: Apple Inc., 2015. https://itunes.apple.com/us/app/retigo-vision/id863304006?mt=8.
104. *Retigo Vision – Aplikace pro Android ve službě Google Play*: Apple Inc., 2015. https://play.google.com/store/apps/details?id=cz.itech21.retigovision&hl=cs.
105. *UIKit User Interface Catalog: Action Sheets*: Apple Inc., 2015. https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/UIKitUICatalog/UIActionSheet.html.
106. *ChocolateChip-UI*: SOURCEBITS, 2015. http://chocolatechip-ui.com.
107. *GitHub - Build Software Better Together*: Github, Inc., 2014. https://github.com/.
108. *Remote Debugging on Android with Chrome: Google Chrome.* 2014. s. https://developer.chrome.com/devtools/docs/remote-debugging.
109. *Overview: Google Chrome.* 2014. s. https://developer.chrome.com/devtools/index.
110. Mobile Apps: What Consumers Really Need and Want. *Compuware*. 2012. http://offers2.compuware.com/rs/compuware/images/Mobile_App_Survey_Report.pdf.
111. *DOMContentLoaded - Event Reference: MDN.* 2014. s. https://developer.mozilla.org/en-US/docs/Web/Events/DOMContentLoaded.
112. GARSIEL, T.; IRSH, P. *How Browsers Work: Behind the scenes of modern web browsers - HTML5 Rocks*, 2011. http://www.html5rocks.com/en/tutorials/internals/howbrowserswork/.
113. DAVID, S. *Digital Cinematography: Fundamentals, Tools, Techniques, and Workflows.* Burlington: Taylor & Francis, 2014. ISBN 9781136040429.
114. IGOR, U. *iPhone Game Blueprints.* Birmingham: Packt Publishing Ltd, 2013. ISBN 9781849690270.
115. *cordova-android/SystemWebView.java at master · apache/cordova-android*: GitHub, Inc., 2015. https://github.com/apache/cordova-android/blob/master/framework/src/org/apache/cordova/engine/SystemWebView.java.
116. *cordova-android/SystemWebViewEngine.java at 9e400911f5f2cb711bd5f102d06017f4c707ea68 · apache/cordova-android*: GitHub, Inc., 2015. https://github.com/apache/cordova-android/blob/9e400911f5f2cb711bd5f102d06017f4c707ea68/framework/src/org/apache/cordova/engine/SystemWebViewEngine.java.

117. *cordova-android/SystemWebView.java at 9e400911f5f2cb711bd5f102d06017f4c707ea68 · apache/cordova-android*: GitHub, Inc., 2015. https://github.com/apache/cordova-android/blob/9e400911f5f2cb711bd5f102d06017f4c707ea68/framework/src/org/apache/cordova/engine/SystemWebView.java.
118. *android.webkit | Android Developers*: Googe Inc., 2015. http://developer.android.com/reference/android/webkit/package-summary.html.
119. *CSS Conditional Rules Module Level 3*: W3C®, 2013. http://www.w3.org/TR/css3-conditional/ - at-supports.
120. *The Chromium Projects*, 2015. https://http://www.chromium.org.
121. *pwnall/chromeview · GitHub*: GitHub, Inc., 2015. https://github.com/pwnall/chromeview.
122. *Crosswalk - build world class hybrid apps*: Intel Corporation, 2015. https://crosswalk-project.org.
123. *crosswalk-project/cordova-plugin-crosswalk-webview · GitHub*: GitHub, Inc., 2015. https://github.com/crosswalk-project/cordova-plugin-crosswalk-webview.
124. *ludei/webview-plus*: GitHub, Inc., 2015. https://github.com/ludei/webview-plus/.
125. *ludei/webview-plus-ios*: Github, Inc., 2015. https://github.com/ludei/webview-plus-ios.
126. *Telerik-Verified-Plugins/WKWebView · GitHub*: Github, Inc., 2015. https://github.com/Telerik-Verified-Plugins/WKWebView.
127. *cordova-plugins/wkwebview-engine at master · apache/cordova-plugins · GitHub*: GitHub, Inc., 2015. https://github.com/apache/cordova-plugins/tree/master/wkwebview-engine.
128. *The Benchmark | Octane | Google Developers*: Google Inc., 2015. https://developers.google.com/octane/benchmark.
129. *BlacklistsAndWhitelists - WebGL Public Wiki*, 2015. https://http://www.khronos.org/webgl/wiki/BlacklistsAndWhitelists.

7 PUBLIKACE

7.1 Článek ve sborníku

- JAŠEK, Roman, BENDA, Radek, VALA, Radek, SARGA, Libor. Launching Distributed Denial of Service Attacks by Network Protocol Exploitation. In *Proceedings of the 2nd International Conference on Applied Informatics and Computing Theory (AICT '11)*. Piraeus : WSEAS Press, 2011, s. 210-216. ISBN 978-1-61804-034-3.
- VALA, Radek, JAŠEK, Roman, MALANÍK, David. Design of a Software Tool for Mobile Application User Mental Models Collection and Visualization. In *Proceedings of teh 2014 International conference on Applied Mathematics, Computational Science and Engineering*. Craiova : Europment, 2014, s. 133-141. ISSN 2227-4588. ISBN 978-1-61804-246-0.
- VALA, Radek, JAŠEK, Roman. Proposal of Improving Web Application Security in Context of Latest Hacking Trends. In *Proceedings of the 1st International Conference on Innovative Computing and Information Processing (INCIP´13)*. Rhodes : WSEAS Press (GR), 2013, s. 107-111. ISSN 1790-5109. ISBN 978-960-474-311-7.
- VALA, Radek. Near Field Communication (NFC) v mobilních zařízeních, její bezpečnost a aplikace. In *First International Conference on Application of Modern Information Technology in Logistics*. Přerov : College of Logistics in Prerov, 2013, s. 69-75. ISBN 978-80-87179-32-1.
- ŠENKEŘÍK, Roman, PLUHÁČEK, Michal, ZELINKA, Ivan, KOMÍNKOVÁ OPLATKOVÁ, Zuzana, VALA, Radek, JAŠEK, Roman. Performance of Chaos Driven Differential Evolution on Shifted Benchmark Functions Set. In *International Joint Conference SOCO´13 - CISIS´13 - ICEUTE´13*. Heidelberg : Springer-Verlag Berlin, 2014, s. 41-50. ISSN 2194-5357. ISBN 978-3-319-01853-9.
- VALA, Radek, SARGA, Libor, BENDA, Radek. Security Reverse Engineering of Mobile Operating Systems: A Summary. In *Proceedings of the 17th WSEAS International Conference on Computers (COMPUTERS '13)*. Rhodes : WSEAS Press (GR), 2013, s. 112-117. ISSN 1790-5109. ISBN 978-960-474-311-7.
- VALA, Radek, MALANÍK, David, JAŠEK, Roman. Usability of software intrusion-detection system in web applications. In *International Joint Conference CISIS ´12-ICEUTE ´12-SOCO ´12*. Heidelberg : Springer-Verlag Berlin, 2013, s. 159-166. ISSN 2194-5357. ISBN 978-3-642-33017-9.

- SARGA, Libor, VALA, Radek. Softwarový audit. In *Optimalization and Simulation Methods*. Zlín : Fakulta aplikované informatiky Univerzity Tomáše Bati ve Zlíně, 2010, s. 1-7. ISBN 978-80-7318-989-1.
- VALA, Radek, JAŠEK, Roman. Google Hacking a rizika při uchovávání a zpracování informací on-line aplikacemi. In *Internet, bezpečnost a konkurenceschopnost organizací. Řízení procesů a využití moderních teerminálových technologií*. Zlín : Univerzita Tomáše Bati ve Zlíně, Fakulta aplikované informatiky, 2010, s. 437-443. ISBN 978-83-61645-16-0.
- ŠENKEŘÍK, Roman, VALA, Radek. Bezpečnost RFID Karet a Útoky Proti Nim. In *Optimalization and Simulation Methods*. Zlín : Univerzita Tomáše Bati ve Zlíně, Fakulta aplikované informatiky, 2010, s. 136-140. ISBN 978-80-7318-989-1.
- VALA, Radek, JAŠEK, Roman. Performance of Hybrid Mobile Application UI Frameworks. In *Proceedings of teh 2014 International conference on Applied Mathematics, Computational Science and Engineering*. Craiova : Eurompment, 2014, s. 293-306. ISSN 2227-4588. ISBN 978-1-61804-246-0.
- JAŠEK, Roman, VALA, Radek, MALANÍK, David. Security and Safety Education in the Czech Republic and eSEC-Portal User Requirements. In *Proceedings of the 11th European Conference on Information Warfare and Security*. CURTIS FARM, KIDMORE END, NR READING, RG4 9AY : Acad Conferences Ltd, 2012, s. 145-150. ISBN 978-1-908272-56-0.
- JAŠEK, Roman, MALANÍK, David, VALA, Radek. Multilevel User Verification with Intelligent Artificial Neural Network. In *Advances in Remote Sensing, Finite Differences and Information Security*. Praha : WSEAS Press, 2012, s. 192-194. ISBN 978-1-61804-127-2.
- MALANÍK, David, JAŠEK, Roman, VALA, Radek. Rogue AP with HTTPS Credentials Sniffer. In *Advances in Remote Sensing, Finite Differences and Information Security*. Praha : WSEAS Press, 2012, s. 195-198. ISBN 978-1-61804-127-2.

7.2 Ostatní publikace

- ŽOUŽELKOVÁ, Iveta, VALA, Radek. Possibilities of Storing Arrays and Their Performance Testing in the Most Common DBMS. Trilobit, 2011, 1-8.

7.3 Software

- JAŠEK, Roman, VALA, Radek. On line vyhodnocení kvality obchodních nabídek. 2014,
- JAŠEK, Roman, VALA, Radek. Ovládání konvektomatu - inteligentní vysokokapacitní kuchyně. 2013,
- JAŠEK, Roman, VALA, Radek, MALANÍK, David. Aplikace pro řízení legislativních povinností v oblasti BOZP. 2013,
- JAŠEK, Roman, VALA, Radek, MALANÍK, David. Vývoj cloud-computingového nástroje pro zprávu a sdílení dokumentací nemovitostí a infrastruktury. 2013,
- JAŠEK, Roman, VALA, Radek, MALANÍK, David. Software pro administraci. sběr a vyhodnocování dat. 2013,
- JAŠEK, Roman, VALA, Radek. Prototyp systému pro restreaming obrazu z IP kamer. 2013,

7.4 Vedené diplomové práce

- BENEDÍK, Jan. Moderní UI webové aplikace pomocí Scroll Reveal efektů. Zlín, 2015. UTB ve Zlíně.
- BEZRUČ, Martin. Konceptuální návrh ergonomie laboratorního pracoviště pro podporu výuky předmětu Kamerové systémy. Zlín, 2015. UTB ve Zlíně.
- BLAŽEK, David. Srovnání rychlosti komunikace pomocí webových služeb v prostředí iOS. Zlín, 2013. UTB ve Zlíně.
- CVRKAL, Petr. Návrh pracoviště pro měření obrazových funkčních vlastností bezpečnostních kamer. Zlín, 2015. UTB ve Zlíně.
- DORAZÍNOVÁ, Iva. Manipulace s 3D modely v mobilních aplikacích. Zlín, 2014. UTB ve Zlíně.
- FUSEK, Jakub. Interoperable Process Design Kit a jeho automatizované generování. Zlín, 2015. UTB ve Zlíně.
- GRIGAR, Jiří. Identifikace firemních objektů s využitím technologie NFC v mobilních zařízeních. Zlín, 2015. UTB ve Zlíně.
- HANZAL, Ondřej. Automatické dohledování a správa systémů s využitím instrumentačních technik. Zlín, 2012. UTB ve Zlíně.
- HOLBÍKOVÁ, Petra. Návrh responzivního uživatelského prostředí v oblasti e-komerce. Zlín, 2014. UTB ve Zlíně.

- HORŇÁK, Lukáš. Srovnání vývoje a výkonu mobilní aplikace tvořené nativně pro operační systém Android nebo pomocí HTML5. Zlín, 2013. UTB ve Zlíně.
- HRAZDIL, Martin. Projekt tvorby pracovního prostředí pro testování a školení specialistů služeb BlackBerry. Zlín, 2012. UTB ve Zlíně.
- KONČICKÝ, Martin. Konceptuální návrh technického řešení laboratorního pracoviště pro podporu výuky předmětu Kamerové systémy. Zlín, 2015. UTB ve Zlíně.
- MALIŠKA, David, Implementace cloud API pro sdílení souborů do frameworku QCubed. Zlín, 2014. UTB ve Zlíně.
- NACÍK, Jakub. Návrh klient-server fakturačního systému s podporou exportu účetních dat pomocí standardního přenosového formátu a možností použití na mobilních zařízeních. Zlín, 2014. UTB ve Zlíně.
- NEPOŽITEK, Václav. Multiplatformní aplikace pro mobilní zařízení. Zlín, 2013. UTB ve Zlíně.
- SEDLÁŘ, Martin. Návrh nízkonákladové alternativy k RFID řešením pro řízení přístupu, identifikaci, tvorbu rezervací a elektronické platby v budově fitness centra. Zlín, 2014. UTB ve Zlíně.
- STRAKA, David. Moderní webový portál školy na CMS Wordpress s integrací Google Apps. Zlín, 2015. UTB ve Zlíně.
- ŠIMÍČEK, Tomáš. Analýza vývojových frameworků vhodných pro tvorbu hybridních mobilních aplikací. Zlín, 2014, UTB ve Zlíně.

7.5 Vědeckovýzkumné aktivity

- 2013–2014 – Hlavní řešitel a spoluřešitel projektů inovačních voucherů zlínského kraje.
- 2009–2012 – člen řešitelského týmu evropského projektu Leonardo da Vinci – 502092-LLP-1-2009-SK-ERASMUS-EMHE
- Junior Researcher – Projekt CEBIA-Tech, CZ.1.05/2.1.00/03.0089

ODBORNÝ ŽIVOTOPIS AUTORA

Osobní údaje

Radek Vala, narozen 23. 11. 1984

vala@fai.utb.cz

Vzdělání

2009–současnost, Univerzita Tomáše Bati ve Zlíně, FAI, Inženýrská informatika (doktorské studium)

2007–2009, Univerzita Tomáše Bati ve Zlíně, FAI, Obor Informační technologie (navazující magisterské studium)

2004–2007, Univerzita Tomáše Bati ve Zlíně, FAI, Obor Informační technologie (bakalářské studium)

1996–2004, Gymnázium Bruntál, Dukelská 1 (osmileté gymnázium)

Pracovní zkušenosti

2012 – dosud, Asistent na Ústavu informatiky a umělé inteligence – výuka předmětů Technologie WWW, Mobilní technologie; Vedoucí vývoje mobilní aplikace UTB; Hlavní řešitel a spoluřešitel projektů smluvního výzkumu Inovační vouchery zlínského kraje 2013 a 2014 (UTB ve Zlíně)

2013–2014, externí spolupráce na implementaci a řešení zobrazovacích problémů mobilní aplikace pro bankovnictví (Invelect s.r.o.)

2011, Bezpečnostní testování a penetrační testy webového portálu mezinárodního projektu eSEC (UTB ve Zlíně)

2010, vývoj portálu Elektronické žákovské knížky (Užíváno 7 základními školami severomoravského kraje. Webový portál 4. ZŠ Bruntál www.zsbrok.cz je od roku 2013 archivován Národní knihovnou jako součást národního kulturního dědictví.)

2009–2012, Spoluřešitel mezinárodního projektu Leonardo da Vinci s aktivní účastí na zahraničních workshopech (Sicílie, Slovinsko, Slovensko, Portugalsko, Francie) (UTB ve Zlíně)

2007–2008, Poradenství při implementaci platebních bran GP WebPay (Externě pro privateguide.cz, skolmax.cz)

2006 (květen - srpen), Kódování HTML&CSS šablon pro portál AUTO.cz (Externě pro Anima Publishers, s.r.o.)

2004–2006, Kompletní vývoj portálu jobs.hw.cz (Externě pro HW server s.r.o.)

2003, Kompletní vývoj portálu continentalcup.cz (Externě pro Barum Continental spol. s r.o.)

Odborná příprava, školení

2014, Developing Microsoft Azure and Web Services (Gopas), Vývoj webových aplikací pomocí ASP.NET MVC (Gopas)

2013, Vývoj mobilních aplikací iOS (iStyle.cz)

2011, Software Summer Camp 2011 (IBA.cz), Bezpečnost Wi-fi sítí

Získané certifikáty

2014, Certifikace Apple Certified Support Professional 10.9, Certifikace Apple Certified Trainer – Mavericks 101 OS X Support Essentials 10.9, Certifikace Microsoft Web Applications, Certifikace Microsoft Programming in HTML5 with JavaScript and CSS3 Specialist, Certifikace Microsoft Certified Professional

Jazyky

Mateřský jazyk: Český

Ostatní jazyky: Anglický – středně pokročilý, Německý – mírně pokročilý

Další znalosti a dovednosti

Webdesign: HTML5, CSS3, XML, AJAX, JavaScript, jQuery, PHP, PHP frameworky, CMS systémy, MySQL, MSSQL, PostgreSQL

Mobilní technologie: Objective-C, JAVA, C, C++, C#, Apache Cordova/Phonegap, Xamarin

Vývojová prostředí: Eclipse, NetBeans, CodeBlocks, XCode

Pedagogická činnost na Univerzitě Tomáše Bati ve Zlíně v období 2009-2015

Cvičení z předmětů: Mobilní technologie, Technologie WWW, Databázové systémy, Teorie programů, Základy informatiky

Přednášky z předmětů: Mobilní Technologie, Technologie WWW

Cvičení v cizím jazyce: Mobile Technology, WWW Technology, Information technology I (Erasmus)

Ostatní: Kurz U3V - Internet a informační média