

# Moderní metody tvorby multiplatformních aplikací

Andrej Osuský

---

Bakalářská práce  
2015



Univerzita Tomáše Bati ve Zlíně  
Fakulta aplikované informatiky

---

Univerzita Tomáše Bati ve Zlíně  
Fakulta aplikované informatiky  
akademický rok: 2014/2015

## ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Andrej Osuský**

Osobní číslo: **A12199**

Studijní program: **B3902 Inženýrská informatika**

Studijní obor: **Informační a řídicí technologie**

Forma studia: **prezenční**

Téma práce: **Moderní metody tvorby multiplatformních aplikací**

Téma anglicky: **Modern Methods for the Development of Cross-platform Applications**

Zásady pro vypracování:

1. Vytvořte literární rešerši na téma moderní metody tvorby mobilních aplikací, specifika jejich vývoje, SW/HW požadavků a způsobů nasazení.
  2. Prozkoumejte možnosti a metodiku tvorby mobilních aplikací pomocí jazyka C/C++/Qml a knihovny Qt/Quick.
  3. Srovnajte implementační náročnost, možnosti a výkon takových mobilních aplikací s aplikacemi vytvořených pomocí konkurenčních nástrojů a frameworků (Xamarin, PhoneGap, ...).
  4. Demonstrujte způsob vývoje mobilních aplikací pomocí C/C++/Qml a knihovny Qt/Quick na jednoduchém příkladu a srovnajte implementační náročnost takové aplikace s některou z vybraných konkurenční technologií.
-

Rozsah bakalářské práce:

Rozsah příloh:

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

1. DIGIA. Qt Library website [online]. 2015 [cit. 2015-01-28]. Dostupné z: <http://qt-project.org/>
2. RISCHPATER, Ray. Application Development with Qt Creator, 2nd Edition. Packt Publishing – ebooks Account, 2014. 2 edition. ISBN 978-1784398675.
3. SHOTTS, Kerri. PhoneGap 3.x Mobile Application Development Hotshot. United Kingdom: Packt Publishing – ebooks Account, 2014. 2 edition. ISBN 978-1783287925.
4. REYNOLDS, Mark. Xamarin Mobile Application Development for Android. Birmingham: Packt Publishing, 2014. ISBN 978-178-3559-169.
5. BLANCHETTE, Jasmin a Mark SUMMERFIELD. C GUI Programming with Qt 4: Prentice Hall Open Source Software Development Series. 2. ed. Trolltech ASA: Prentice hall, xxi, 718 s. ISBN 978-0-13-235416-5.
6. PRATA, Stephen. Mistrovství v C++. 3. aktualiz. vyd. Překlad Boris Sokol. Brno: Computer Press, 2007, 1119 s. ISBN 978-80-251-1749-1.

Vedoucí bakalářské práce:

**Ing. Michal Bližňák, Ph.D.**

Ústav informatiky a umělé inteligence

Datum zadání bakalářské práce:

**6. března 2015**

Termín odevzdání bakalářské práce:

**22. května 2015**

Ve Zlíně dne 6. března 2015



doc. Mgr. Milan Adámek, Ph.D.  
*děkan*



L.S.



prof. Ing. Vladimír Vašek, CSc.  
*ředitel ústavu*


### **Prohlašuji, že**

- beru na vědomí, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl jsem seznámen s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – bakalářskou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

### **Prohlašuji,**

- že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně

..........  
podpis diplomanta

## **ABSTRAKT**

Hlavným cieľom tejto bakalárskej práce je poskytnúť informácie o moderných metódach tvorby multiplatformných aplikácií. V teoretickej časti sú klasifikované rôzne metódy a objasnené rozdiely medzi nimi. Taktiež sú tu predstavené niektoré z najpoužívanejších nástrojov pre multiplatformný vývoj a vysvetlené na akých technológiách sú tieto nástroje založené. V praktickej časti som demonštroval skutočný vývoj aplikácie pomocou týchto nástrojov a porovnal ich implementačné náročnosti.

Kľúčové slova: multiplatformné nástroje, mobilné aplikácie, vývoj aplikácií, aplikačný rámec, Qt, Phonegap, Xamarin

## **ABSTRACT**

Main goal of this bachelor thesis is to provide information about modern methods for development of cross-platform applications. In the theoretical section, there are classified different methods and clarified the differences between them. There are also introduced some of the most commonly used tools for cross-platform development and explained on which technologies these tools are based. In the practical part I have demonstrated the real application development using these tools and compared their implementation difficulties.

Keywords: cross-platform tools, mobile apps, application development, framework, Qt, Phonegap, Xamarin

Chcel by som na tomto mieste poďakovať vedúcemu mojej bakalárskej práce pánovi Ing. Michalovi Bližňákovi, Ph.D. za jeho ochotný a aktívny prístup ako aj za poskytnuté rady pri vypracovávaní. Taktiež ďakujem svojej rodine a blízkym za pomoc a podporu počas celej doby štúdia na vysokej škole.

# OBSAH

<b>ÚVOD</b> .....	<b>9</b>
<b>TEORETICKÁ ČASŤ</b> .....	<b>11</b>
<b>1 NATÍVNE A MODERNÉ METÓDY TVORBY APLIKÁCIÍ</b> .....	<b>12</b>
1.1 NATÍVNY VÝVOJ.....	12
1.2 MULTIPLATFORMNÝ VÝVOJ .....	13
1.2.1 Webové aplikácie .....	14
1.2.2 Hybridné aplikácie .....	15
1.2.3 Interpretované aplikácie .....	15
1.2.4 Generované aplikácie .....	16
1.2.5 Porovnanie jednotlivých prístupov k multiplatformnému vývoju .....	16
1.3 POŽIADAVKY NA VÝVOJ .....	17
1.3.1 Android .....	18
1.3.2 iOS.....	18
1.3.3 Windows Phone .....	19
1.4 SPÔSOBY NASADENIA APLIKÁCII .....	20
1.4.1 Oficiálny obchod .....	20
1.4.2 Inštalačný balíček.....	22
1.4.3 Využitie vývojového prostredia .....	24
<b>2 FRAMEWORKY</b> .....	<b>26</b>
2.1 QT FRAMEWORK.....	26
2.1.1 História a vznik Qt .....	27
2.1.2 Podporované platformy .....	27
2.1.3 Používané programovacie jazyky .....	28
2.1.3.1 Qt Widget aplikácie .....	29
2.1.3.2 Qt Quick aplikácie .....	30
2.1.4 Licenčná politika .....	33
2.1.5 Programové vybavenie inštalačného balíka .....	34
2.1.6 Požiadavky Qt pre OS Windows.....	34
2.2 PHONEGAP .....	35
2.3 XAMARIN .....	36
<b>PRAKTICKÁ ČASŤ</b> .....	<b>37</b>
<b>3 DEMONŠTRATÍVNA MOBILNÁ APLIKÁCIA</b> .....	<b>38</b>
3.1 POPIS CIEĽOVEJ VYVÍJANEJ APLIKÁCIE .....	38
3.2 TVORBA POMOCOOU QT .....	39
3.2.1 Tvorba GUI.....	40
3.2.2 Vnútná logika.....	43
3.2.3 Zhodnotenie a postrehy z vypracovania .....	44
3.3 TVORBA POMOCOOU PHONEGAP .....	45
3.3.1 Tvorba GUI.....	45
3.3.2 Vnútná logika.....	47
3.3.3 Zhodnotenie a postrehy z vypracovania .....	49
3.4 TVORBA POMOCOOU XAMARIN .....	50
3.4.1 Tvorba GUI.....	50

3.4.2	Vnútorná logika.....	53
3.4.3	Zhodnotenie a postrehy z vypracovávania.....	54
3.5	VYHODNOTENIE.....	55
<b>ZÁVER .....</b>		<b>57</b>
<b>ZOZNAM POUŽITEJ LITERATÚRY .....</b>		<b>58</b>
<b>ZOZNAM POUŽITÝCH SYMBOLOV A SKRATIEK.....</b>		<b>61</b>
<b>ZOZNAM OBRÁZKOV .....</b>		<b>62</b>
<b>ZOZNAM TABULIEK .....</b>		<b>63</b>
<b>ZOZNAM PRÍLOH.....</b>		<b>64</b>

## ÚVOD

Dnešní technická doba umožnila vznik takmer nespočetného množstva mobilných zariadení ktoré ľudia používajú každý deň. Tieto zariadenia disponujú rôznym HW a SW ktorý je pre ne špecifický. Zvyšujúce a zlepšujúce sa parametre HW prvkov ako aj postupné zvyšovanie nárokov koncových užívateľov, je dôvodom neustáleho vylepšovania a vyvíjania nových operačných systémov pre tieto zariadenia. Medzi tieto systémy patria známe platformy ako Android, iOS či Windows Phone, ale aj tie menej známe ako sú BlackBerry, Firefox OS, Sailfish OS, Tizen, Ubuntu Touch a ďalšie. Všetky platformy sa od seba vzájomne líšia z hľadiska používaných technológií pri vývoji, princípu realizácie systému, ponúkaných funkcionalít či grafického spracovania. Jednou z hlavných spoločných vlastností mobilných platforiem je však to, že užívateľovi je umožnené ich funkcionality ďalej rozširovať. To je možné vďaka inštalovateľným aplikáciám.

Mobilné aplikácie sú neoddeliteľnou súčasťou dnešných mobilných operačných systémov. Každá platforma disponuje veľkým množstvom inštalovateľných aplikácií ktoré sú vždy určené len pre konkrétny OS. To znamená, že v prípade snahy o dostupnosť určitej aplikácie na rôznych platformách je potrebné vytvoriť jej špecifickú verziu pre každú z nich. Vývojári aplikácií majú teda možnosť písať ich aplikácie zvlášť pre každú platformu v jej špecifickom jazyku alebo sa môžu rozhodnúť pre použitie niektorého z nástrojov na multiplatformný vývoj. Táto voľba so sebou nesie mnoho výhod medzi ktoré patrí pre vývojárov hlavne obrovská úspora času.

V súčasnosti je dostupný veľký počet rôznych druhov týchto nástrojov. Problémom býva však práve výber tohto nástroja, keďže každý zo sebou nesie určité výhody aj nevýhody. Je potrebné zvážiť veľa aspektov každého z nich ako je napríklad spôsob realizácie multiplatformnosti, používaný programovací, skriptovací či značkovací jazyk, rozsah možností daného nástroja a v neposlednom rade licenčnú a s ňou spojenú finančnú stránku. Z toho jasne vyplýva že správny výber takéhoto nástroja nie je vôbec triviálny, ale pre jeho správne zvolenie je potrebné rozhodovať sa na základe viacerých informácií. Vzhľadom na veľkú popularitu tejto témy je pri rozhodovaní možné obrátiť sa na veľké množstvo internetových článkov a porovnaní nástrojov na rôznych fórach. V týchto prípadoch sú však prezentované názory často založené len na teoretických znalostiach jednotlivých nástrojov, alebo bývajú silne ovplyvnené subjektívnymi preferenciami autora. Takéto zavádzajúce názory môžu byť vzhľadom na dôležitosť správneho rozhodnutia veľmi nevhodné.

K zvolení si tejto témy bakalárskej práce ma viedlo viacero dôvodov. Prvým z nich bol môj záujem o vývoj mobilných aplikácií a zvedavosť týkajúca sa súčasných moderných metód ich tvorby. Ďalej bolo mojím cieľom porovnanie rôznych nástrojov a vlastné zistenie a porovnanie ich jednotlivých výhod, keďže informácie dostupné na Internete môžu byť skreslené. Ďalším dôvodom bola snaha o to, aby zvolená téma mala aj praktický charakter, ktorého výsledky by mohli byť zaujímavé a prospešné nielen pre mňa ale aj pre čitateľov tejto práce.

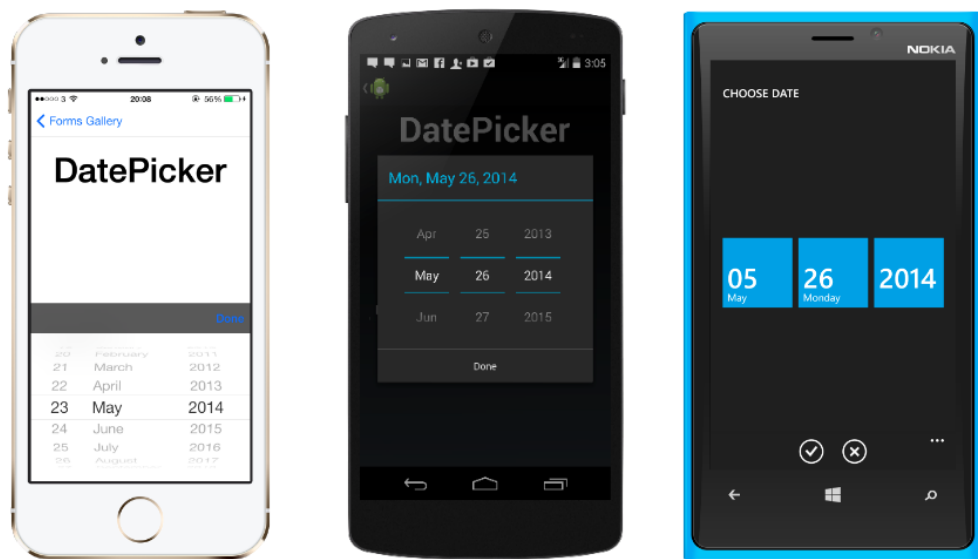
## **I. TEORETICKÁ ČASŤ**

## 1 NATÍVNE A MODERNÉ METÓDY TVORBY APLIKÁCIÍ

Prvá kapitola pojednáva o problematike tvorby aplikácii a možnostiach jej tvorby modernými metódami. Obsahuje informácie o základnom rozdelení tvorby aplikácii z hľadiska jej cieľového použitia ako aj rozdelenie rôznych metód multiplatformného vývoja. Taktiež pojednáva o jednotlivých spôsoboch nasadenia aplikácii na špecifické mobilné platformy ako aj o nevyhnutných požiadavkách každej platformy ktoré sú nutné k tomu aby bolo možné vyvíjať pre ne určené aplikácie.

### 1.1 Natívny vývoj

Za natívne aplikácie alebo programy sú považované také aplikácie, ktoré sú vyvíjané pre jednu špecifickú platformu. Vývojár teda musí vždy dopredu vedieť ktorý programovací jazyk bude používať a pre ktorý operačný systém bude aplikácia určená. Takýto spôsob tvorby nesie so sebou mnohé výhody. Spadajú do nich najmä vlastnosti aplikácii ako ich rýchlosť, prirodzený grafický vzhľad a v neposlednom rade vývojárova možnosť využívať všetky vlastnosti ponúkané daným systémom. Takto vytvorená aplikácia má teda schopnosť používať špecifický software a hardware a tak jednoducho využívať najnovšie vlastnosti daného zariadenia. Takéto riešenie má však jeden veľký nedostatok. V prípade že sa vývojár rozhodne poskytovať aplikáciu aj na inom ako pôvodnom systéme, narazí na problém vzájomnej nekompatibility. Vďaka takejto nekompatibilite zdrojových kódov nie je v drvivej väčšine prípadov možné na inej platforme aplikáciu ani len spustiť. Prípadná úprava zdrojových kódov môže dokonca v určitých prípadoch vyžadovať zásah do jadra samotného programu. Pre zaistenie korektného fungovania by bolo teda nutné vykonať zásadné zmeny zdrojového kódu, čo je často vysoko neefektívne. Pri takomto spôsobe vývoja aplikácii je teda nutné samostatne spravovať sady zdrojových kódov osobitne pre každý z cieľových operačných systémov. To má taktiež za následok radikálne predĺženie času strávené pri vývoji aplikácii, ktoré vedie k neefektívnosti práce. V prípade že je v záujme vývojára poskytovať svoju aplikáciu na viacerých platformách je natívny vývoj ako riešenie krajne nevhodné. [7]



Obr. 1. Natívny vzhľad na rôznych platformách [12]

## 1.2 Multiplatformný vývoj

Z dôvodu nekompatibility natívne písaných aplikácií s ostatnými systémami začína byť zvykom vývojárov poohliadnutie sa po vhodných knižniciach a nástrojoch na multiplatformnú aplikačnú tvorbu. Takouto tvorbou je myslený vývoj aplikácií so zdieľanou časťou zdrojových kódov, najčastejšie vnútornej logiky programu. Zo spoločného zdrojového kódu je následne vytvorený spustiteľný alebo inštalateľný súbor pre zvolené operačné systémy. Použitie takýchto nástrojov má pre tvorcu aplikácie za následok výrazné zrýchlenie práce a teda aj ušetrenie času stráveného pri vývoji. Vďaka takto urýchlenému vývoju sa môže daná aplikácia rýchlejšie dostať k užívateľom a prípadne predbehnúť konkurenciu. Výhodným je taktiež fakt že v prípade aktualizácie či nutnosti opravy, nie je potrebné vykonať zmeny v niekoľkých aplikačných projektoch. Zmeny stačí vykonať v jednom projekte a z neho vytvoriť novú verziu pre všetky cieľové platformy. Pri využívaní multiplatformných nástrojov a knižníc je ale potrebné počítať s určitými obmedzeniami. V dnešnej dobe existuje veľké množstvo rôznych platform určených pre obrovské množstvo zariadení. Z dôvodu rýchleho vývoja týchto platform, ako aj ich vzájomnej rozdielnosti, neexistuje multiplatformný nástroj ktorý by bol schopný so všetkými dokonale spolupracovať. Aj pri používaní knižníc pre multiplatformnú tvorbu sa môže stať, že bude z dôvodu odlišných platform, potrebné vykonať zmeny v kóde. Môže sa jednať o zmeny vo vnútornej logike no najčastejšie ide o editáciu užívateľského rozhrania. Obecne však platí, že vykonanie zmien býva jednoduché a úpravy sú často len kozmetické. [7,10]

V dnešnej dobe existujú viaceré druhy a spôsoby mutliplatformného vývoja. Neočakávaný nárast výskytu veľkého množstva mobilných zariadení mal za následok vytvorenie viacerých vývojových prostredí ktoré používajú rôzne metódy multiplatformnej tvorby. Takto produkované aplikácie sa delia do štyroch základných skupín a to konkrétne na webové, hybridné, interpretované a generované aplikácie. Žiadnu z týchto metód tvorby nie je možné prehlásiť za najlepšie riešený prístup k multiplatformnej tvorbe, keďže každá z nich nesie so sebou svoje klady aj zápory. [11]

### 1.2.1 Webové aplikácie

Webové aplikácie sú založené na používaní webového prehliadača. Takéto aplikácie využívajú pre svoj beh internetové technológie. Majoritne ide o značkový jazyk HTML a skriptovací jazyk JavaScript. Z toho jasne vyplýva aj jedna z hlavných nevýhod takýchto aplikácií. Tieto aplikácie majú na reálnych zariadeniach obmedzený prístup ku dátam a na hardware. Tento problém sa snaží riešiť jazyk HTML5 ktorý na HW dokáže pristupovať lepšie ako jeho predchádzajúca verzia. Ďalšou negatívnou vlastnosťou je rýchlosť aplikácie. Vždy je potrebný určitý čas na stiahnutie aplikácie z Internetu a vykreslenie web stránok v nej. Keďže sú tieto aplikácie poskytované z webového serveru a nie je fyzicky možné ich nainštalovať na zariadenie, môže vzniknúť ďalší problém. V prípade nedostupnosti servera alebo nemožnosti pripojenia sa do siete, nie je možné ani pristupovať ku aplikácii. Výhodou môže byť naopak fakt, že v tomto prípade nie je vyžadovaná inštalácia aplikácie ani jej aktualizácii. Existuje mnoho SW knižníc pre vývoj web aplikácií, ktoré sa snažia napodobniť schopnosti natívne písaných aplikácií. Medzi najznámejšie patria JQuery Mobile, Sencha Touch alebo JQTouch, no existuje mnoho ďalších. Aktuálne používaný jazyk HTML5 už zďaleka nie je len jednoduchý značkový jazyk. Bol obohatený o celú radu API ktoré boli implementované aby znižovali spotrebovanú energiu. Taktiež boli mimo iného pridané aplikačné rozhrania pre prácu s úložným priestorom, databázami, súbormi a geolokáciou . [11]

### 1.2.2 Hybridné aplikácie

Hybridné aplikácie sú primárne postavené na používaní HTML5 a JavaScriptu, vďaka čomu nie je potrebná detailná znalosť cieľovej platformy. Tieto aplikácie sa však snažia kombinovať a využívať výhody ako webových tak aj natívne písaných aplikácií. Princípom takýchto aplikácií je že aplikácie založené na používaní HTML5 sú vkladané do natívnych kontajnerov resp. obalov. U systému Android ide o WebView a u zariadení používajúcich iOS je to UIWebView. Tieto aplikácie potrebujú podobne ako webové aplikácie pre svoj chod webový prehliadač. Rozdiel nastáva pri spúšťaní zdrojového kódu. U webových aplikácií je potrebné ho najskôr stiahnuť z webového servera avšak u hybridných je zdrojový kód zapuzdrený priamo v aplikácii ktorá sa fyzicky nachádza nainštalovaná na zariadení. Pomocou špecifických API sú schopné pristupovať ku dátam a na hardware zariadenia. Jedným z najpopulárnejších nástrojov na tvorbu hybridných mobilných aplikácií je software PhoneGap. Tieto aplikácie je možné tvoriť rôznymi technológiami avšak v prípade že je od aplikácie požadovaný natívny vzhľad, nastávajú komplikácie. Je nutné použiť špeciálne vývojové knižnice ktoré dokážu pristupovať k natívnym API danej platformy alebo vzhľad simulovať vhodným nastavením štýlov. [11]

### 1.2.3 Interpretované aplikácie

Takéto aplikácie sa vyznačujú automatickým generovaním natívneho kódu. Pomocou tohto kódu je následne realizované užívateľské rozhranie. To má za následok prirodzený vzhľad aplikácie. Užívateľ teda pracuje vždy s natívnymi UI komponentmi ktoré sú vždy špecifické pre dané zariadenie. Vnútoraná logika takýchto aplikácií je realizovaná nezávisle. Môže byť realizovaná pomocou mnohých technológií ako napríklad Ruby, Java, a podobne. Efektivita natívneho užívateľského rozhrania je teda hlavnou výhodou takejto metódy vývoja. Naproti tomu nevýhodou tohto riešenie je úplná závislosť na používanom vývojovom prostredí. To sa prejavuje hlavne v prípadoch kedy môžu byť pre danú platformu k dispozícii nové špecifické funkcionality. Môže ísť napríklad o pridané funkcie UI v novej verzii systému Android. Na to aby mohla aplikácia pracovať s týmito aktualizáciami musia byť najskôr podporované používaným vývojovým prostredím. Veľmi používaným SW nástrojom na tvorby interpretovaných aplikácií, ako aj hybridných aplikácií, je v dnešnej dobe Appcelerator Titanium mobile. [11]

#### 1.2.4 Generované aplikácie

Hlavnou črtou generovaných aplikácií je že sú kompilované ako natívne. Pre každú cieľovú platformu je generovaná špecifická verzia aplikácie. Takto tvorené aplikácie dosahujú vďaka generovaniu natívneho kódu spravidla vysokého výkonu. Taktiež je teoreticky takto generovaný kód možné využiť na ďalšie úpravy v jazyku platformy a teda použiť vygenerovaný natívny kód a rozšíriť ho o vlastný natívny kód. Takéto riešenie sa však v praxi príliš nepoužíva a to z dôvodu náročnosti upravovania vygenerovaného kódu keďže ten má už vlastnú automatizovanú štruktúru. Príkladom SW pomocou ktorého sú vyvíjané generované aplikácie sú vývojové prostredia Applause alebo iPhonical. V oboch prípadoch ide o software založený na používaní akéhosi vnútorného modelu. Takýto vývoj je označovaný skratkou MDSD. Jeho zámerom je snaha o riešenie problémov pomocou tohto vnútorného modelu. Modelovací jazyk je použitý pre reprezentovanie funkcionality a správania sa aplikácie. V prípade mobilného multiplatformného vývoja sú najskôr modelované funkcionality vyvíjanej aplikácie na vysoko abstraktnej úrovni a následne je z tohto modelu generovaný natívny kód určený pre konkrétnu cieľovú platformu. [11]

#### 1.2.5 Porovnanie jednotlivých prístupov k multiplatformnému vývoju

Ako už bolo spomenuté, žiaden z uvedených prístupov k multiplatformnému vývoju nie je možné jednoznačne prehlásiť za najlepší či najhorší. Voľba zostáva na vývojárovi ktorý musí zhodnotiť jednotlivé plusy a mínusy zvoleného prístupu. Na komplexné zhodnotenie metód používaných pri vývoji mobilných aplikácií je nutné porovnávať jednotlivé metódy vývoja podľa ich špecifických charakteristík a kritérií. Pri analýze popisovaných prístupov bolo porovnávaných nasledovných päť kritérií:

- **Umiestnenie do obchodu, distribúcia** – v tomto kritériu bolo vyhodnocované, či je možné nasadzovať aplikácie vytvorené pomocou daného prístupu do obchodov s mobilnými aplikáciami. Analyzované boli špecifické obchody ako sú napríklad Google Play a Apple App Store.
- **Rozšírenosť technológií** – kritérium vyhodnocuje či je na tvorbu aplikácií danou metódou používaná široko rozšírená vývojová technológia (napr. JavaScript, HTML) ktorá je obecnne veľmi známa a používaná.
- **Prístup k dátam a na hardware** – zobrazuje schopnosť prístupu takto vytvorených aplikácií k zdrojom zariadenia. Rozdelenie do troch kategórií: bez prístupu, obmedzený a úplný prístup.

- **Vzhľad a vlastnosti užívateľského rozhrania** – toto kritérium podáva informáciu o tom či aplikácie podporujú natívne funkcionality a vzhľad komponentov užívateľského rozhrania danej platformy alebo je toto UI odlišné od natívneho vzhľadu a teda simulované pomocou použitých knižníc.
- **Užívateľovo vnímanie výkonu** – kritérium sa zaoberá skúmaním názorov koncových užívateľov na výkonnosť aplikácie. Porovnávanými parametrami sú napríklad rýchlosť načítavania či spúšťania danej aplikácie v porovnaní s natívnymi. Toto kritérium je hrubým odhadom na základe hodnotenia jednotlivých metód tvorby užívateľmi na Internete. [11]

Výsledok analýzy spôsobov multiplatformnej tvorby aplikácii podľa uvedených kritérií je prehľadne zobrazený v nasledujúcej tabuľke.

Tab. 1. Vlastnosti metód multiplatformného vývoja [11]

	WEBOVÉ	HYBRIDNÉ	INTERPRETOVANÉ	GENEROVANÉ
Umiestnenie do obchodu	Nie	Áno	Áno	Áno
Rozšírenosť technológií	Áno	Áno	Áno	Nie
Prístup k dátam a na HW	Obmedzený	Obmedzený	Obmedzený	Plný prístup
Vzhľad a vlastnosti UI	Simulovaný	Simulovaný	Natívny	Natívny
Užívateľovo vnímanie výkonu	Nízky	Stredný	Stredný	Vysoký

### 1.3 Požiadavky na vývoj

Možnosti vývoja aplikácii sú vždy špecifické pre zvolenú cieľovú platformu. Každá platforma so sebou nesie vlastné požiadavky na HW zvoleného zariadenia, ako aj požiadavky na HW a SW zariadenia na ktorom prebieha vývoj. S týmito požiadavkami je nutné počítať vždy skôr ako sa začne požadovaná aplikácia tvoriť. V opačnom prípade by sa totiž mohlo veľmi jednoducho stať, že aj keď používaný nástroj na multiplatformný vývoj v sebe podporuje danú platformu, používaný software a hardware nemusí byť schopný toho istého a vývoj tak nie je možný. V prípade že by sa vývojár rozhodol vyvíjať svoju aplikáciu pre všetky mobilné platformy, nevyhnutne na tieto obmedzenia narazí. Môže dôjsť k situácii kedy ten istý nástroj na multiplatformný vývoj bude musieť byť prítomný na rôznych zariadeniach.

### 1.3.1 Android

Hlavným rozdielom v požiadavkách oproti ďalším dvom platformám je, že Android aplikácie je možné vyvíjať na ktoromkoľvek operačnom systéme. Či už ide o Windows, Linux alebo Mac OS X, na každom z týchto systémov je možné využívať vývojové nástroje pre tvorbu Android aplikácií. Tieto vývojové nástroje v sebe obsahujú taktiež funkčný emulátor každej vydanéj verzie systému Android, preto pre začatie vývoja nie je potrebné fyzicky disponovať zariadením ktoré používa tento software. Pre vývoj v Jave, teda v natívnom programovacom jazyku pre Android, je nevyhnutné aby bol na vývojárskom zariadení prítomný nasledujúci SW:

- **Java Development Kit (JDK)** – obsahuje základné nástroje pre vývoj v jave.
- **Android software development kit (SDK)** – obsahuje vývojové nástroje a knižnice nutné pre tvorbu Android aplikácií ako aj spomínaný Android emulátor.
- **Vývojové prostredie** – momentálne je oficiálne podporovaným prostredím Android Studio. Predtým bolo používané prostredie Eclipse s doinštalovaným zásuvným modulom ADT. [14]

Ak je cieľom vyvíjať aplikácie v inom jazyku ako v natívnom, napríklad C++ je potrebný ďalší software ako **Android NDK** a **Apache Ant**. V prípade požiadavkou na hardware je nutné brať ohľad primárne na požiadavky vývojového prostredia. Pre Android Studio je potrebných minimálne 400 MB miesta na disku a približne 2 GB pamäte RAM. Pre Android SDK je doporučený aspoň 1GB miesta na disku a pre pribalený emulátor približne 2 GB operačnej pamäte. Požiadavky na zariadenia na ktorom operačný systém Android beží sa líšia podľa jeho verzie. [15,16]

### 1.3.2 iOS

Pred tým ako je možné začať s vývojom iOS aplikácií je taktiež potrebné aby boli splnené určité požiadavky na hardware a software vývojového zariadenia. Hlavnou odlišnosťou vývoja iOS aplikácií oproti vývoju Android aplikácií je že nie je možné vyvíjať aplikácie na akomkoľvek stroji či operačnom systéme. Odhliadnuc od toho, či je pri vývoji používaný určitý nástroj na multiplatformný vývoj alebo nie, pre vývoj je nevyhnutné disponovať Apple Mac počítačom s Intel procesorom.



Obr. 2. Apple Mac [17]

Pre vývoj je potrebné aby na tomto počítači bol prítomný nasledujúci software:

- **OS X vo verzii 10.9.4 alebo novšej** – tento operačný systém je štandardnou súčasťou Apple Mac počítačov. Pri vývoji aplikácii pre zariadenia iPhone a iPad je jeho prítomnosť nevyhnutná.
- **IDE Xcode** - toto prostredie umožňuje tvoriť aplikácie ako na Mac tak aj na iPhone a iPad. V aktuálnej verzii Xcode 6.3.1 môže byť zdarma stiahnuté z obchodu s aplikáciami Mac App Store. Pre vývoj je taktiež možné využiť rôzne multiplatformné nástroje a ich vývojové prostredia.
- **iOS SDK** – je vývojový nástroj ktorý je súčasťou vývojového prostredia Xcode. Nástroj obsahuje simulátor ktorý umožňuje simulovať prostredie mobilného zariadenia so systémom iOS. Tak poskytuje podobne ako Android emulátor možnosť vyvíjať a testovať aplikácie bez nutnosti reálneho zariadenia.

Vyvíjať aplikácie natívnym spôsobom je možné pomocou jazyka Objective C a taktiež pomocou nového jazyka Swift. Oba sú podporované vývojovým prostredím Xcode. Na Mac počítači je potrebné mať k dispozícii dostatok priestoru na disku pre spomínané komponenty a dostatočnú veľkosť pamäte RAM pre iOS simulátor. Zariadenia využívajúce iOS sú konštruované tak aby mali dostatočne vybavený HW pre beh systému. [17,18]

### 1.3.3 Windows Phone

Pri vývoji aplikácii Windows Phone je nutné podobne ako pri iOS disponovať špecifickým počítačom na ktorom je nainštalovaný potrebný operačný systém. Vyvíjať tieto aplikácie je možné iba na OS Windows 8.1, Windows 8 a Windows 7 v obidvoch 32 aj 64 bitových verziách. Pri Windows 7 je ešte potrebné aby bol na ňom nainštalovaný Internet Explorer

vo verzii 10 alebo vyššej. Aby sa predišlo prípadným problémom je taktiež potrebné aby boli nainštalované všetky kritické aktualizácie. Pre vývoj je potrebné aby boli nainštalované nasledujúce nástroje a to v tomto poradí:

- **Visual Studio 2013** – ide o široko rozšírené IDE určené na klasický aplikačný vývoj platformy Windows. Obsahuje nástroje pre tvorbu desktop, Windows Phone ale aj mobilných webových aplikácií. Do toho prostredia je možné doinštalovať moduly rôznych multiplatformných vývojových nástrojov alebo miesto neho využiť iné IDE ktoré môže byť súčasťou daného nástroja.
- **Windows SDK 8.1** – v súčasnej dobe je zahrnutý v balíku programu Visual Studio. Tento vývojový nástroj obsahuje knižnice a ďalšie nástroje ktoré sú používané pre vývoj na platforme Windows.
- **Windows driver kit 8.1** – tento software umožňuje vyvíjať ovládače pre zariadenia s platformou Windows. Obsahuje v sebe dokumentáciu a nástroje pre vývojárov.
- **Power Engine Plug-in** – tento modul je potrebný z dôvodu že niektoré ovládače využívajúce takzvaný systém na čipe (SoC) sú na ňom závislé.
- **Windows Phone 8.1 kit** – ide o spoločný názov pre skupinu vývojových komponentov ktorými sú Windows Phone Driver Kit, Windows Phone Adaptation Kit, Windows Phone debugger symbols a Test Schell. [19]

Bez všetkých týchto komponentov nie je možné kompilovať potrebný kód. V prípade že sa v operačnom systéme nachádzajú staršie verzie týchto nástrojov je potrebné ich odstrániť a nainštalovať znova aktuálne verzie podľa uvedeného poradia. [19]

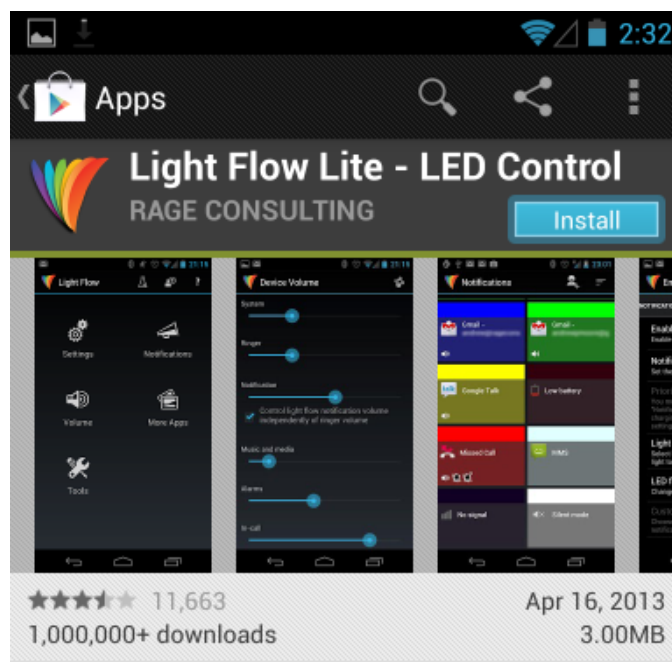
## 1.4 Spôsoby nasadenia aplikácií

Ako spôsoby nasadenia sú myslené možnosti ako dostať aplikáciu na cieľové zariadenie. Teoretický princíp spôsobov ako to docieľiť je pre Android, iOS a Windows Phone veľmi podobný. Prakticky je však možné naraziť na niektoré rozdiely. Nasadenie môže byť obecné vykonané tromi spôsobmi: z oficiálneho obchodu, z inštalačného balíčka a priamo z vývojového prostredia. [21, 22, 23, 25]

### 1.4.1 Oficiálny obchod

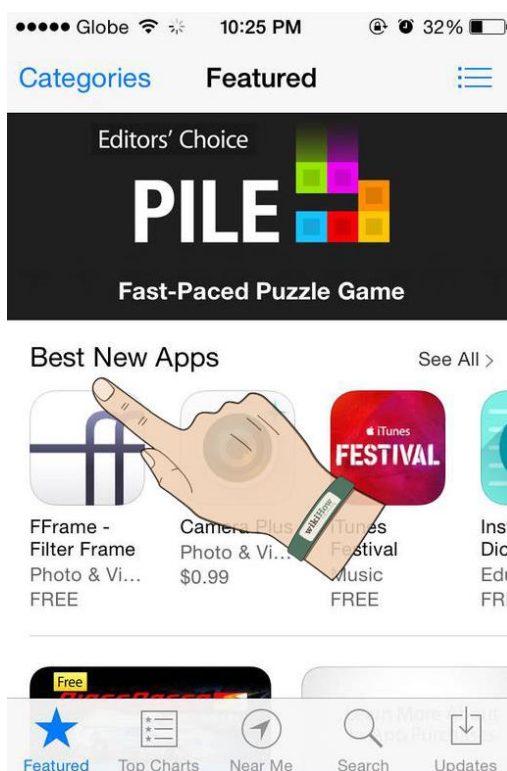
Každá platforma disponuje svojim vlastným obchodom s aplikáciami. To predstavuje najrozšírenejší spôsob distribúcie aplikácií. Pre každú platformu sú tu dostupné aplikácie za poplatok ale aj zdarma. Ide o nasledovné obchody:

- **Android** – Google Play



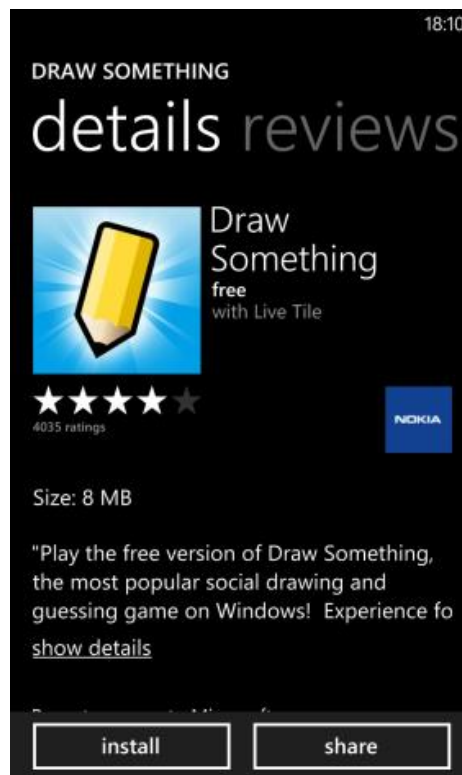
Obr. 3. Google Play [22]

- **iOS** – App Store



Obr. 4. iOS App Store [25]

- **Windows Phone** – Windows Phone Store



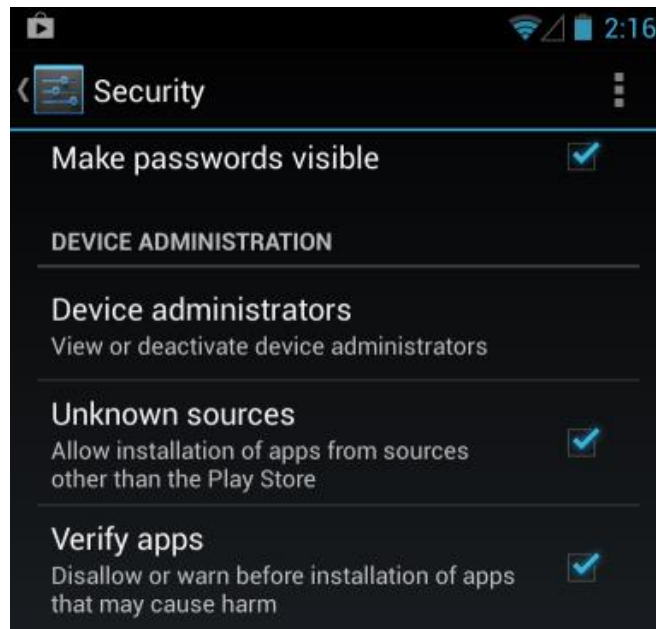
*Obr. 5. WP Store [21]*

Do týchto obchodov možno prístupovať pomocou konkrétnej aplikácie vyskytujúcej sa na každej mobilnej platforme. Pri jej použití nastane inštalácia okamžite. Existuje taktiež možnosť použiť počítač a vstúpiť do Google play a Windows Phone Store pomocou webového rozhrania. Pre inštaláciu iOS aplikácii je potrebné použiť aplikáciu iTunes. Vždy je potrebné zvoliť cieľové zariadenie. Naň sa aplikácia nainštaluje až keď bude zariadenie pripojené k Internetu. Pridávanie aplikácii do týchto obchodov vždy vyžaduje registráciu a s ňou spojený poplatok. [21, 22, 25]

#### **1.4.2 Inštalačný balíček**

Pre nasadenie aplikácie je možné využiť inštalačné balíčky. Tie sú často využívané v prípade že má mobilné zariadenie obmedzenú možnosť prístupu na Internet, v prípade využívania neoficiálnych obchodov s aplikáciami alebo na nekomerčné šírenie aplikácie za účelom testovania pred jej vydaním. Vývojár ich môže pomocou vývojového prostredia vyexportovať z aktuálneho projektu. Balíčky sú špecifické vždy pre každú platformu a nie je možné ich medzi sebou zamieňať.

Súbory s príponou **.apk** označujú inštaláčnne balíky platformy Android. Inštaláciu je možné vykonať dvoma spôsobmi. Prvým je použitie Android SDK a nainštalovanie aplikácie priamo z počítača za použitia tzv. Android Debug Bridge. Táto možnosť však vyžaduje, aby boli na mobilnom zariadení povolené neznáme zdroje, vo vývojárskych nastaveniach aktivované ladenie cez USB a aby zariadenie pripojené USB káblom malo na počítači nainštalované potrebné ovládače. Druhý spôsob je jednoduchší a stačí ak je súbor s príponou **.apk** uložený na karte alebo v internej pamäti zariadenia. V prípade že v nastaveniach sú povolené už spomínané neznáme zdroje je možné spustiť inštaláciu zo súborového prieskumníka priamo v zariadení a to poklepaním na príslušný súbor. Nie je teda potrebný žiadny dodatočný software. [23]



Obr. 6. Povolenie neznámych zdrojov [22]

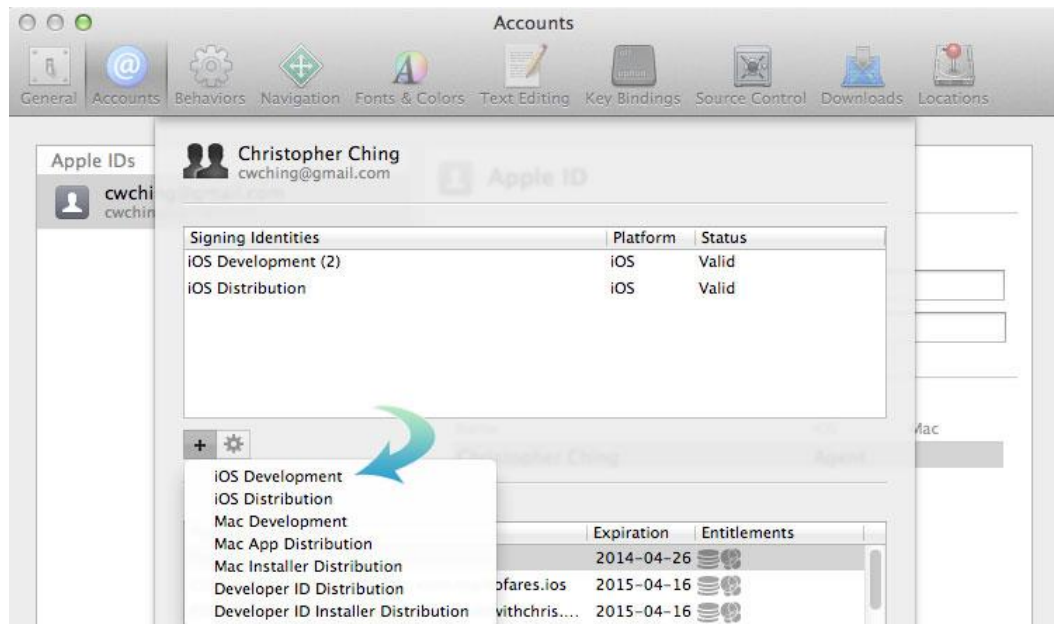
So súbormi s príponou **.ipa** sa môžu stretnúť užívatelia operačného systému iOS. Pre nainštalovanie takéhoto balíčka je potrebné pripojiť zariadenie k počítaču a spustiť Program iTunes. Pomocou neho treba otvoriť príslušný súbor, vyvolať inštaláciu a spustiť synchronizáciu. Ak všetko prebehne korektne aplikácia by sa mala objaviť v zozname aplikácii prístroja. Existuje taktiež možnosť využiť stránku [www.diawi.com](http://www.diawi.com) ktorá umožňuje nahrať **.ipa** súbor a následne vygenerovať url adresu. V prípade že sa prehliadač Safari pokúsi o načítanie stránky na vygenerovanej adrese, bude užívateľ automaticky vyzvaný na povolenie inštalácie. Tá následne prebehne bez potreby využiť ďalší software. Na takéto inštalácie však môžu byť použité len balíčky s ustanovovacím profilom ktorý obsahuje identifikačné číslo používaného zariadenia. [24, 26]

Súbory nesúce príponu **.xap** a **.appx** sú určené platforme Windows Phone. Súbory **.xap** sú staršie a súbory s príponou **.appx** sú ich nástupcami vytvorenými pre operačné systémy Windows Phone 8.1 a novšie. Ich inštaláciu na reálne zariadenie je možné vykonať pomocou nástroja Application Deployment ktorý je nainštalovaný súčasne s Windows Phone SDK. V ňom je následne treba zvoliť cieľové zariadenie a vložiť zdrojový inštalčný balíček. Program po potvrdení voľby vykoná inštaláciu. Taktiež je možné tieto balíčky nainštalovať priamo z pamäťovej karty zariadenia. Pomocou aplikácie Store je možné nájsť balíčky a vykonať inštaláciu zvolených aplikácií. [20, 21, 27]

### 1.4.3 Využitie vývojového prostredia

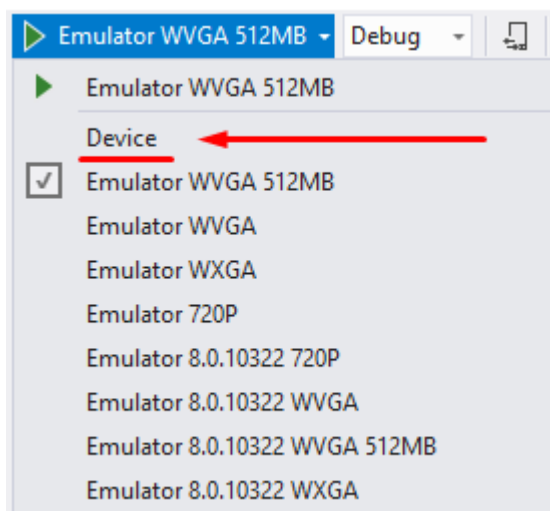
Veľmi častou možnosťou využívanou hlavne vývojármi je zavádzanie aplikácie priamo z vývojového prostredia. Vyvíjaná aplikácia tak môže byť testovaná na prípadné chyby ktoré sa v emulátore neprejavili, alebo na funkcie ako GPS či akcelerometer ktorými emulátor nedisponuje. Aby mohlo byť vykonané takéto zavádzanie priamo z IDE je pre každú platformu nutné určitým spôsobom pripraviť dané zariadenie

- **Android** – aby mohla byť zavedená aplikácia je potrebné vykonať určité nastavenia. Na reálnom zariadení je potrebné aktivovať vývojárske nastavenia. V nich je potrebné povoliť možnosť ladenia cez USB a povoliť toto ladenie konkrétnemu pripojenému počítaču. Na počítači musia byť nainštalované potrebné ovládače pre komunikáciu s mobilným zariadením. V používanom IDE následne stačí zvoliť pripojené zariadenie a pri zavedení sa aplikácia vždy automaticky nainštaluje. [22, 23]
- **iOS** – vzhľadom na snahu spoločnosti Apple o maximálnu bezpečnosť je príprava na používanie reálneho zariadenia pri ladení v tomto prípade značne zdĺhavejšia. Každý vývojár sa musí zapísať do plateného Apple iOS vývojového programu. Po zápise je získaný vývojársky účet v ktorom je možné získať vývojársky alebo distribučný certifikát. Tie sú následne používané na podpis vyvíjaných aplikácií. Predtým ako bude možné používať reálne zariadenie na testovanie je nutné ho nakonfigurovať pre vývoj a pridať ho do členského centra používaných zariadení. Následne treba vytvoriť ustanovovací profil vyvíjanej aplikácie. Pri nastavovaní tohto profilu pre vývojárske účely je potrebné špecifikovať konkrétne zariadenie na ktoré bude aplikácia inštalovaná. Až po splnení týchto náležitostí a pripojení mobilného zariadenia k počítaču je možné vykonávať testovanie aplikácie na zariadení. [24]



Obr. 7. Vytvorenie vývojárskeho a distribučného iOS certifikát [24]

- **Windows Phone** – pre ladenie na reálnom zariadení tejto platformy je taktiež potrebné vykonať určité kroky. Je potrebná registrácia vývojára ako aj registrácia zariadenia, ktoré je pri vývoji používané. Podmienkou je taktiež, že pri zavádzaní musí byť zariadenie pripojené pomocou USB kábla a jeho obrazovka musí byť zapnutá a odomknutá. Po splnení uvedeného je možné vo vývojovom prostredí zvoliť pre nasadenie aplikácie pripojené zariadenie. [20]



Obr. 8. Výber WP zariadenia [20]

## 2 FRAMEWORKY

Pri vývoji aplikácii siahajú často ich tvorcovia po nástrojoch ktoré im uľahčujú a zjednodušujú prácu. Ide o programátorské pomôcky nazývané toolkit a framework. Pod názvom toolkit je v informatike väčšinou myslená skupina nástrojov nazývaných widgety. Tieto slúžia ako základné prvky grafického užívateľského rozhrania. Ako framework alebo aplikačný rámec je nazývaná kompletná knižnica alebo skupina knižničných modulov s ďalšími podpornými nástrojmi. Jedná sa teda o akúsi softvérovú štruktúru ktorej úlohou pri vývoji aplikácii je zrýchlenie a uľahčenie programátorovej práce. Framework je možné považovať za akúsi komplexnejšiu verziu toolkitu. Aplikačný rámec so sebou nesie nie len klasické predprogramované metódy, dátové štruktúry alebo konštanty ktoré sú obecnou súčasťou knižnice, ale môže poskytovať aj niektoré návrhové nástroje a rôzne možnosti rozšírenia iného SW. [7]

### 2.1 Qt framework

Qt framework je jedným z momentálne najpoužívanejších multiplatformných UI vývojových aplikačných rámcov. Jedná sa o komplexnú knižnicu zameranú na čo najjednoduchšiu multiplatformnú tvorbu aplikácii. Hlavnou výhodou je fakt že tieto aplikácie sú aj pre rôzne platformy produkované z jedného spoločného zdrojového kódu pre vnútornú logiku ako aj pre grafické rozhranie. Framework je možné využiť napríklad ako widget toolkit čisto pre tvorbu užívateľského rozhrania aplikácii. Je tu však možné nájsť aj všetky potrebné triedy nutné k vytáraniu kompletných multiplatformných aplikácii s grafickým rozhraním. V tomto frameworku je taktiež možné vyvíjať konzolové aplikácie alebo využiť služby ktoré vyžadujú iba použitie jeho jadra. Qt poskytuje programátorovi takpovediac základné stavebné bloky vo forme pred pripravených dátových štruktúr, funkcií a tried, ktoré je možné pri tvorbe aplikácii využiť. Môžeme tu nájsť rozsiahlu množinu widgetov, štýlov a grafických vzorov ktoré vývojár môže podľa vlastnej vôle upravovať a používať pri tvorbe moderných grafických rozhraní. V Qt nechýba podpora pre prácu s multimédiami, zvukom, videom, internetom, sieťami, databázami, grafikou, animáciami či vizuálnymi efektmi ktorými konkurencia nemusí disponovať. Je tu taktiež možné nájsť napríklad integrované jadro webového prehliadača WebKit. Ten v sebe obsahuje podporu pre interakciu s webovým obsahom a jeho vykresľovaním. Taktiež podporuje mnohé technológie a štandardy, medzi ktoré patria napríklad XML alebo SVG. Tie je takto veľmi jednoduché použiť vo vyvíjanej aplikácii. [1,7]

Qt vo svojej ponuke obsahuje všetky základné nástroje, služby, komponenty a technológie potrebné a používané pri programovaní jednoduchých ale aj profesionálnych aplikácií. Používaná knižnica je ale aj navzdory tomu modulárna a flexibilná a dovoľuje používať iba tie moduly a triedy ktoré sú potrebné. Knižnica je v neustálom vývoji a do už tak rozsiahlej palety nástrojov pribúdajú stále nové užitočné triedy, funkcie a množstvo SW komponentov. Vďaka nim je neustále rozširovaná použiteľnosť a uplatnenie tohto frameworku. Qt taktiež disponuje zásuvným modulom do populárneho vývojového prostredia Visual Studio ktorý je následne možné používať pre vývoj aplikácií namiesto základného IDE, ktoré je súčasťou každého stiahnuteľného balíka Qt. Tento modul je však dostupný až v platených verziách tohto frameworku. [1,2,7]

### 2.1.1 História a vznik Qt

História tohto aplikačného rámca siaha až dvadsaťštyri rokov späť v čase. Jeho vznik je datovaný do roku 1991 pričom spočiatku sa jednalo o relatívne malý projekt ktorého autormi boli Haavarda Nord a Eirik Chambe-Eng. V tejto dobe sa ešte zďaleka nejednalo o projekt takých rozmerov ako je tomu dnes. Počas prvej dekády jeho vývoja sa ešte jednalo o multiplatformný toolkit, teda išlo o menej rozsiahly nástroj, ktorý bol dostupný len na niektorých operačných systémoch. Prvé dve verzie Qt boli určené výhradne pre Windows a X11. Koncom roku 2001 bola vydaná verzia Qt 3.0 v ktorej už bola pridaná aj podpora pre Mac OS X. Aplikačný rámec Qt sa odvtedy významne rozšíril a aktuálne je tento framework dostupný vo verzii 5.4. [1,2,9]

### 2.1.2 Podporované platformy

Žiaden z nástrojov pre multiplatformnú produkciu aplikácií nepodporuje úplne všetky existujúce systémy. Aplikačný rámec Qt je však medzi vývojármi veľmi rozšírený práve vďaka jeho širokej palete podporovaných najrôznejších platforiem. S postupom času zameriaval svoju pozornosť na platformy z ktorých niektoré už zanikli, no niektoré stále existujú a v Qt sú podporované ich najnovšie verzie. Aktuálna verzia podporuje nielen väčšinu desktopových systémov a aktuálne najpoužívanejšie mobilné platformy, ale nezaostáva ani s podporou embedded systémov. Prehľad platforiem pre ktoré je možné pomocou najnovšej verzie Qt 5.4 vyvíjať aplikácie je uvedený v nasledujúcej tabuľke. [7, 8]

Tab. 2. Podporované platformy Qt 5.4 [8]

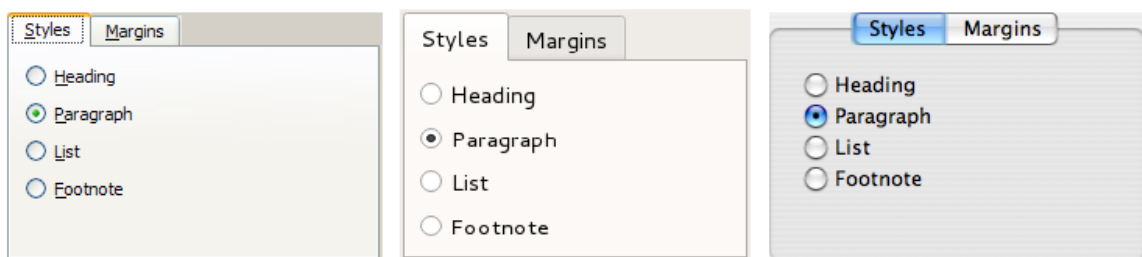
Podporované platformy Qt 5.4		
Desktopové	Mobilné	Vstavané
Windows (Vista/7/8)	WinRT (vrátane Windows Phone)	Windows Embedded 7
Linux/X11 (OpenSuSE, Red hat, Ubuntu)	Android	Embedded Linux
OS X (10.7-10.10)	iOS	Embedded Android
		QNX
		INTEGRITY

### 2.1.3 Používané programovacie jazyky

V tomto frameworku je možné používať dva programovacie jazyky a to konkrétne jazyk C++ a deklaratívny jazyk QML. Dôvod prečo sa vývojári rozhodli postaviť tento projekt práve na používaní jazyka C++ je ten, že tento jazyk predstavuje určitý štandard u aplikácií, pri ktorých je kladený veľký dôraz na vysoký výkon. To je u klasických desktopových a hlavne u mobilných aplikácií veľmi žiadané. Jazyk QML, skratka z anglického Qt Meta-object Language, je deklaratívnym jazykom slúžiacim na popis vizuálnych komponentov, z ktorých sa má skladať užívateľské rozhranie. Taktiež umožňuje definovať spôsob ako sa majú správať pri vzájomnej interakcii či interakcii s užívateľom. Z hľadiska vývojárov sa jedná o veľmi jednoducho čitateľný jazyk ktorý bol navrhnutý aby podporoval dynamický spôsob prepojenia jeho komponentov. Elementy užívateľského rozhrania dovoľuje jednoducho prispôbovať a opakovane používať. Jazyk QML taktiež disponuje podporou jazyka JavaScript, vďaka ktorému je možné implementovať jednoduchú aplikačnú logiku. Tento jazyk výraznou mierou urýchľuje prototypovanie užívateľského rozhrania, ako aj spoluprácu medzi programátormi a designermi. Qt takýmto deklaratívnym spôsobom tvorby disponuje vďaka zavedeniu modulov QtQuick a QtQML do jeho knižnice. Tie sú vo frameworku prítomné od verzie 4.7 a ich úlohou je prevod QML popisu užívateľského rozhrania v deklaratívnej podobe na jednotlivé položky grafickej scény. [2, 6, 7, 8]

### 2.1.3.1 Qt Widget aplikácie

Qt Widget aplikácie sú jedným z dvoch základných druhov aplikácii ktoré je možné pomocou frameworku Qt vyvíjať. Pri vývoji týchto aplikácii môže byť na tvorbu UI použitý vstavaný designer a jazyk XML alebo aj samotný jazyk C++ a knižničný modul Qt Widgets. Ten poskytuje množstvo elementov pre tvorbu užívateľského rozhrania. Tieto elementy dokážu zobrazovať informácie o stavoch, prijímať vstupy, poskytovať spoločný kontajnerový priestor pre ďalšie widgety a podobne. Všetky ovládacie prvky v tomto druhu projektov dedia zo základnej triedy QWidget alebo sú použité v spojení s ňou. Táto trieda poskytuje schopnosť vykresľovania elementov na obrazovku ako aj prácu so základnými udalosťami. UI elementy a základné ovládacie prvky môžu byť vykresľované pomocou rôznych štýlov. Tie sú však zamerané na klasický desktopový vzhľad ktorý sa snaží vyzeráť na týchto platformách natívne. [2,8]



Obr. 9. Rôzne desktopové štýly widget aplikácii [8]

Nasledujúci kód demonštruje vytvorenie jednoduchej Qt Widgets aplikácie. Účelom aplikácie je umiestniť na požadované miesto tlačidlo o požadovanej veľkosti. Po stlačení tlačidla je vyvolané okno informujúce o tejto udalosti. Pri tvorbe aplikácie bol využitý výlučne jazyk C++ a vygenerovaná trieda MainWindow. Do jej konštruktoru bol doplnený nasledujúci kód.

```
MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    ui->setupUi(this);

    // Vytvorenie tlačidla
    QPushButton *btn = new QPushButton("Click", this);

    // Nastavenie pozície a veľkosti tlačidla
    btn->setGeometry(QRect(QPoint(100, 100), QSize(200, 50)));

    // Prepojenie signálu tlačidla s požadovaným slotom
    connect(btn, SIGNAL(released()), this, SLOT(handleButton()));
}
```

V hlavičkovom súbore je zadaný slot reagujúci na signál vyslaný tlačidlom.

```
private slots:    void handleButton();
```

V zdrojovom súbore je pridaná obsluha v prípade prijatia signálu slotom

```
void MainWindow::handleButton()
{
    // Vyvolanie informačného okna so zadaným popisom
    QMessageBox::information(this,
                             tr("Info message"),
                             tr("Button Clicked !"));
}
```

Widget aplikácie poskytujú triedy ktoré podporujú model/view architektúru. Vďaka nej je možné určovať akým spôsobom budú dáta prezentované užívateľovi. V takýchto aplikáciách sú separované dáta a ich zobrazovanie pomocou modelov, pohľadov a delegátov. Widget aplikácie majú bohatú paletu UI elementov a ich funkcionalít. Tie sú však zväčša vhodné a používané pre statické užívateľské rozhrania. V porovnaní s modulom Qt Quick nemajú tieto aplikácie príliš vhodné vlastnosti. Obzvlášť pokiaľ ide o škálovateľnosť widgetov na zariadeniach s dotykovými obrazovkami. Nie sú vhodné ani na výrazne animované moderné užívateľské rozhrania. Voľba takéhoto typu aplikácii je vhodnou v prípade snahy vývojára o vytvorenie tradičných desktopových aplikácií. [8]

### 2.1.3.2 Qt Quick aplikácie

Druhým základným typom aplikácii ktoré je možné pomocou Qt vyvíjať sú Qt Quick aplikácie. Tieto aplikácie sú vytvárané nielen pomocou jazyka C++ ale hlavne sú postavené na využívaní deklaratívneho jazyka QML. Základom týchto aplikácii sú už spomínané moduly Qt QML a Qt Quick. Tie sú spolu vzájomne previazané. Qt QML modul poskytuje základnú SW kostru a dôležité QML typy pre aplikačný vývoj. Definuje a implementuje jazyk a infraštruktúru pre QML aplikácie. Tento modul umožňuje modulu Qt Quick a ostatným využívať jazyk QML. Základné QML typy poskytované týmto modulom sú:

- **Component** – ide o základný, opakovane použiteľný, QML typ s upraviteľným rozhraním. Tieto typy sú často definované ako súbory s príponou .qml ktoré je možné volať z ostatných qml súborov.
- **QObject** – je jednoduchý nenáročný nevizuálny typ ktorého jediná preddefinovaná vlastnosť je `objectName`, teda jeho meno. Často je používaný ako schránka pre pridávanie vlastnej skupiny vlastností. Tie sú tak dostupné pre ostatné komponenty.

- **Binding** – umožňuje previazat' akékoľvek hodnoty alebo vlastnosti. Často sú používané pri previazaní tried exportovaných z C++ do QML na ktoré klasické previazanie nefunguje.
- **Connection** – pomocou tohto objektu je možné vytvoriť spojenie s QML signálom. Pri spájaní so signálom je v QML obvyčajne vytvorený tzv. handler ktorý na daný signál následne reaguje.
- **Timer** – ide o objekt časovača ktorý môže byť použitý na spustenie určitej akcie s časovým intervalom opakovane alebo jednorázovo. [8]

Modul Qt Quick je štandardnou knižnicou pre aplikačný vývoj pomocou jazyka QML. Obsahuje všetky potrebné nástroje pre vývoj aplikácií s moderným a dynamickým UI. Dovoľuje definovať správanie a spôsob vzájomného prepojenia komponentov grafického rozhrania. Pre tieto komponenty poskytuje modul vlastné vykresľovacie plátno s vlastným súradnicovým systémom. Pre zobrazovanie aplikácií vyvíjaných pomocou Qt Quick je nevyhnutnou grafická knižnica OpenGL. Qt Quick obsahuje API pre prácu s jazykom QML ale aj s C++. QML API poskytuje nástroje pre tvorbu moderných GUI pomocou jazyka QML. C++ API slúži na rozširovanie Qt Quick aplikácie o C++ kód. Obe rozhrania majú veľkú schopnosť kooperácie. Je možné si jednoducho vzájomne odovzdávať dáta či napríklad používať v jazyku QML vlastný komponent napísaný v jazyku C++. [7,8]

Dôležité koncepty obsiahnuté v Qt Quick

- **Vizuálne plátno** – modul poskytuje vykresľovacie plátno s dvojdimenzionálnym súradnicovým systémom a zobrazovacím usporiadaním v smere tretej osi. Na vykresľovanie grafiky používaná knižnica OpenGL.
- **Užívateľské vstupy** – Qt Quick dokáže detegovať a reagovať na vstupy z klávesnice, myši, dotykovej obrazovky, či na pohybové gestá zariadenia zaznamávané pomocou zabudovaného akcelerometru.
- **Umiestňovanie** – V návrhu aplikácií je možné rozmiestňovať elementy rôznym spôsobom. Podporované je absolútne ale aj relatívne umiestňovanie. Komponenty tak môžu byť ukotvené vzhľadom na veľkosť a pozície ostatných komponentov.
- **Stavy, presuny a animácie** – Stavom je myslená skupina informácií popisujúcich kde a ako sú vizuálne komponenty zobrazované na plátno. Úlohou presunu je transformovať komponent z pôvodného stavu do iného cieľového. Na presuny medzi stavmi môžu byť aplikované rôzne animácie.

- **Dátové modely a pohľady** – Qt quick podporuje používanie model/View architektúry. Tá je založená na jednoduchom princípe že dáta sú obsiahnuté v modeli, ktorý je následne zobrazovaný pomocou pohľadu. Architektúra je výhodná v prípadoch keď je potrebné dynamicky zobrazovať dáta.
- **Grafické efekty** – Cieľom je dosiahnutie pútavejšieho užívateľského rozhrania. [8]

Základné ovládacie prvky užívateľského rozhrania sú súčasťou samostatného modulu Qt Quick Controls. Ten poskytuje nástroje na popis vlastností okna aplikácie, rôzne menu, navigácie a nástroje pre zobrazovanie iných komponentov v jednom grafickom návrhu a hlavne UI zobrazovacie a ovládacie prvky schopné reagovať na vstupy od užívateľa. [8]

Qt Quick aplikácie majú oproti Qt widget aplikáciám mnohé výhody. Qt Widgets poskytujú možnosť prispôbovať svoje komponenty prostredníctvom štýlov. Naproti tomu Qt Quick komponenty sú omnoho editovateľnejšie. Preto sú vhodnejšie pre aplikácie u ktorých nie je vyžadovaný natívny vzhľad. V Qt Quick je možné pohodlným deklaratívnym spôsobom realizovať plynulé animácie a využívať flexibilné grafické efekty. Využívať takéto funkcionality v Qt Widgets je veľmi problematické. Taktiež tento modul často pre správnu interakciu s užívateľom vyžaduje kurzor myši. Naproti tomu Qt Quick je schopný reagovať na väčšie množstvo už spomínaných vstupov dostupných či už na desktopových alebo na mobilných zariadeniach. Taktiež využíva HW grafickú akceleráciu s aplikačným rozhraním knižnice OpenGL ES 2.0. Je dokázané že takéto riešenie poskytuje najvyšší výkon aplikácie. Ten je využívaný hlavne pre grafické užívateľské rozhrania a v prípadoch kedy v sebe aplikácia integruje grafický OpenGL obsah. [8]

Nasledujúci kód demonštruje vytvorenie jednoduchej Qt Quick aplikácie, ktorá je napísaná výlučne v deklaratívnom jazyku QML. Jej účel je rovnaký ako účel demonštrovanej Qt Widgets aplikácie a to umiestniť tlačidlo určitej veľkosti na určitú pozíciu a po jeho stlačení vyvolať informačné okno.

```
Button {
    text: qsTr("Click")
    onClicked: md.open();
    x: 100;      y: 100;
    width: 200; height: 50;
}

MessageDialog {
    id: md
    title: qsTr("Info message")
    text: qsTr("Button Clicked")
}
```

### 2.1.4 Licenčná politika

Qt je poskytované v rôznych licenčných variantoch. Je dostupné pod komerčnou licenciou a taktiež aj pod GNU General Public Licence verzie 3 a GNU Lesser General Public Licence verzie 2.1. Taktiež existuje možnosť stiahnuť framework bez poplatkov no táto verzia neobsahuje všetky dostupné funkcionality a nie je určená pre komerčné využitie. Konkrétne licenčné varianty ako aj ich vlastnosti sú uvedené v nasledujúcej tabuľke. [1,2,7]

Tab. 3. Licenčné verzie frameworku Qt a ich vlastnosti [13]

	Licencie			
	Enterprise	Professional	Indie Mobile	Community
vývoj platforiem	Všetky podporované	Windows, Linux, Mac OS X, Windows Phone/WinRT, Android, iOS	Windows Phone/WinRT, Android, iOS	Všetky podporované okrem VxWorks a INTEGRITY
práva a služby	Komerčná licencia pre tvorbu aplikácií. Licenčné poplatky ročne alebo jednorázovo	Komerčná licencia pre tvorbu desktop aplikácií. Mesačné príspevky	Komerčná licencia pre tvorbu mobilných aplikácií. Mesačné príspevky	Projekty v súlade GPL, LGPL v2.1, LGPL v3
podpora Qt	Oficiálna podpora spoločnosti	Oficiálna podpora spoločnosti	Podpora komunity	Podpora komunity
služba Qt Cloud	Áno	Áno	Nie	Nie
Qt grafy	Áno	Áno	Nie	Nie
Qt vizualizácia dát	Áno	Áno	Nie	Nie
Qt Quick enterprise ovládacie prvky	Áno	Áno	Nie	Nie
Qt Quick 2D Vykreslovač	Áno	Áno	Nie	Nie
Qt Quick Kompilátor	Áno	Áno	Nie	Nie
Qt Quick Enterprise Designer	Áno	Áno	Nie	Nie
Qt Quick doplnky pre Visual studio	Áno	Áno	Nie	Nie
Podpora vstavných a RTOS platforiem	Áno	Nie	Nie	Nie

### 2.1.5 Programové vybavenie inštalačného balíka

Inštalačný balíček Qt v sebe neobsahuje len spomínanú knižnicu a moduly. Nesie zo sebou celý rad vývojových nástrojov určených na multiplatformnú tvorbu aplikácií a uľahčenie ich vývoja. Hlavnou pomôckou a základným programovým vybavením balíka je program Qt Creator. Jedná sa o integrované vývojové prostredie ktoré v sebe zapuzdruje všetky potrebné funkcionality pre vývoj klasických C/C++ aplikácií ako aj aplikácií využívajúcich knižnicu Qt. Tu prebieha kompletný proces tvorby od písania a upravovania zdrojových kódov, tvorby a vizualizácie grafického užívateľského rozhrania až po ladenie programu. Obsahuje všetky náležitosti moderného vývojového prostredia. Ďalej Qt ponúka nástroj Qt Designer ktorý slúži ako nástroj pre vizualizáciu a tvorbu návrhu užívateľského prostredia. Nájde tu taktiež program Qt Assistant slúžiaci na prácu s dokumentáciou, Qt Linguist obsahujúci internacionalizačné nástroje a multiplatformný zostavovací a prekladový systém qmake. [7]

### 2.1.6 Požiadavky Qt pre OS Windows

Balík Qt pre Windows vyžaduje pre svoje správne fungovanie splnenie určitých požiadaviek. Ide predovšetkým o určité knižnice, grafické ovládače, vývojárske utility a kompilátory bez ktorých by niektoré komponenty nemuseli fungovať korektne, alebo by nefungovali vôbec. Medzi knižnice s ktorými Qt od verzie 5 pracuje patria napríklad ICU a ANGLE. Tieto knižnice síce nie sú povinné, môžu však byť v aplikáciách použité keďže na nich závisia niektoré komponenty ako napríklad Qt WebKit. [1]

- ICU je široko používaná skupina knižníc poskytujúca podporu pre UNICODE a globalizáciu. Potrebuje ju už spomínaný Qt WebKit a aplikácie ktoré s ním pracujú. Taktiež jadro Qt môže byť nakonfigurované tak, aby sa spoliehalo namiesto Windows API na funkcionality ICU.
- Úlohou knižnice ANGLE je dovoliť používateľom OS Windows využívať aplikačné rozhranie grafickej knižnice OpenGL ES 2.0 ktoré je určené na vykresľovanie 2D a 3D grafiky. Knižnica dokáže konvertovať volania OpenGL knižnice na volania DirectX 9 alebo DirectX 11. Vďaka tomu nie je nutné inštalovať grafické ovládače na cieľové zariadenie. V prípade doinštalovania OpenGL ovládačov je možné zmeniť konfiguráciu tak, aby boli využívané miesto knižnice ANGLE [1]

## 2.2 PhoneGap

PhoneGap je ďalší známy vývojový framework určený na tvorbu mobilných aplikácií. Jeho veľká rozšírenosť pramení z používania webových technológií pri vývoji aplikácií. K jeho častému používaniu napomáha aj fakt, že všetky dnešné mobilné platformy obsahujú podporu pre webové aplikácie, resp. obsahujú webový prehliadač. PhoneGap je založený na open source projekte Apache Cordova. Ten umožňuje vyvíjať mobilné aplikácie tak, že vytvorené webové aplikácie zapuzdri do natívneho aplikačného obalu každej cieľovej platformy. I keď je tento obal odlišný pre každú platformu, vždy poskytuje základné funkcionality daného zariadenia. Každá PhoneGap aplikácia je teda v podstate len skupina web stránok. Na takýto vývoj je možné použiť technológie HTML5, CSS3 a JavaScript. Vďaka tomu môže ktokoľvek so znalosťami webových technológií jednoducho vyvíjať aj vlastné mobilné aplikácie. Vzhľadom na použitie natívneho obalu je na rozdiel od klasických webových aplikácií možné vytvorenú aplikáciu distribuovať pomocou bežných obchodov s aplikáciami. [3, 28, 29]

Phonegap je rozšírenie už spomínaného projektu Cordova. Poskytuje mierne odlišné rozhranie ale jeho hlavnou odlišnosťou je možnosť zostaviť aplikácie pomocou vzdialenej služby. Po uložení projektu na server je možné z neho exportovať aplikácie pre zvolené platformy. Nie je teda potrebné fyzicky disponovať nevyhnutným SW a HW zvlášť pre každú platformu. [3]

### Klady

- Používanie široko rozšírených webových technológií
- Použitie API umožňujúcich jednoduché zavedenie na rôzne platformy
- Podpora veľkého počtu mobilných platforiem a operačných systémov ako sú Android, iOS, Windows Phone, Blackberry, Ubuntu a Firefox OS.
- Vysoká opätovná použiteľnosť webových projektov vzhľadom na fakt že z akejkoľvek web stránky môže byť vytvorená mobilná aplikácia. [28, 29]

### Zápory

- Vzhľadom na nutnosť používania pre spúšťanie každej aplikácie webový prehliadač je výkon týchto aplikácií nižší ako je tomu u natívnych aplikácií.
- Existencia veľkého množstva knižníc a nástrojov na základnej alebo nízkej úrovni.
- Vzhľad UI sa môže líšiť v závislosti na používanom prehliadači [28, 29]

## 2.3 Xamarin

Spoločnosť Xamarin poskytuje známy, rovnomenný, multiplatformný vývojový nástroj. Framework Xamarin, pôvodne nazývaný MonoTouch, umožňuje vyvíjať aplikácie pre rôzne systémy ako sú Android, iOS, Windows Phone ale aj Windows a OS X aplikácie. Vývoj je vykonávaný pomocou jazyka C# a .NET frameworku. Tento programovací jazyk je veľmi rozšírený a jeho použitie so sebou nesie mnoho výhod. Nechýba ani vlastné IDE Xamarin Studio. Framework umožňuje tvoriť natívne sa správajúce aplikácie pre každú platformu vďaka využívaniu natívnych API a UI prvkou. Xamarin využíva open source projekt s názvom Mono aby umožnil jednoduchý vývoj aplikácii pre rôzne platformy. Ide o multiplatformnú implementáciu C# kompilátora. [4, 28, 30]

Aktuálne prišla spoločnosť Xamarin s knižnicou Xamarin.Forms. Vďaka nej je možné vyvíjať aplikácie z jedného spoločného kódu ktorý definuje vnútornú logiku ale aj kompletne užívateľské rozhranie. Spoločný kód pre UI je následne konvertovaný na natívne UI špecifické pre každú platformu. Xamarin umožňuje vývojárom využiť jej službu Xamarin TestCloud. Tá umožňuje cez Internet testovať aplikácie na stovkách rôznych zariadení. Pre rýchlejšie a efektívnejšie naučenie sa pracovať s týmto nástrojom ponúka spoločnosť internetové kurzy programu Xamarin University. [28, 30]

### Klady

- Xamarin TestCloud dovoľujúci testovať aplikácie na mnohých zariadeniach
- Knižnica Xamarin.Forms poskytujúca možnosť tvoriť natívne vyzerajúce UI z jedného spoločného kódu pre všetky platformy.
- Natívne UI a prístup k natívnym API špecifickým pre každú platformu
- Xamarin Component obchod obsahujúci pred pripravené komponenty ktoré je možné pridávať priamo cez vývojové prostredie.
- Podpora MVC a MVVM architektúry
- Podpora Android Wear a Google Glass zariadení [28, 30]

### Zápory

- Neposkytuje prístup k niektorým ovládacím prvkom UI systému Android.
- Dlhá doba načítavania vytvorenej aplikácie.
- Chýba podpora pre zdieľanie kódov pre iný napríklad natívny vývoj [28]

## **II. PRAKTICKÁ ČASŤ**

### 3 DEMONŠTRATÍVNA MOBILNÁ APLIKÁCIA

Účelom praktickej časti tejto práce je demonštrovať reálny vývoj aplikácie podľa vytvoreného zadania. Na vývoj boli použité nástroje spomínané v teoretickej časti a to konkrétne Qt, Phonegap a Xamarin. Pre každý framework bol vytvorený názorný postup vývoja krok po kroku. Popisované boli aj vývojové prostredia ak boli súčasťou nástroja.

#### 3.1 Popis cieľovej vyvíjanej aplikácie

Koncept vyvíjanej aplikácie bol vytvorený tak, aby otestoval viaceré schopnosti používateľských nástrojov. Užívateľské prostredie sa má skladať z nasledujúcich elementov:

1. Názov aplikácie – hlavný nadpis udávajúci názov používaného nástroja
2. Výberový nástroj – komponent umožňujúci výber zo zadaných možností
3. Tlačidlo – aktivačný prvok ktorý spúšťa naprogramovaný kód
4. Tabuľka – komponent zobrazujúci dáta a umožňujúci v nich scrollovať

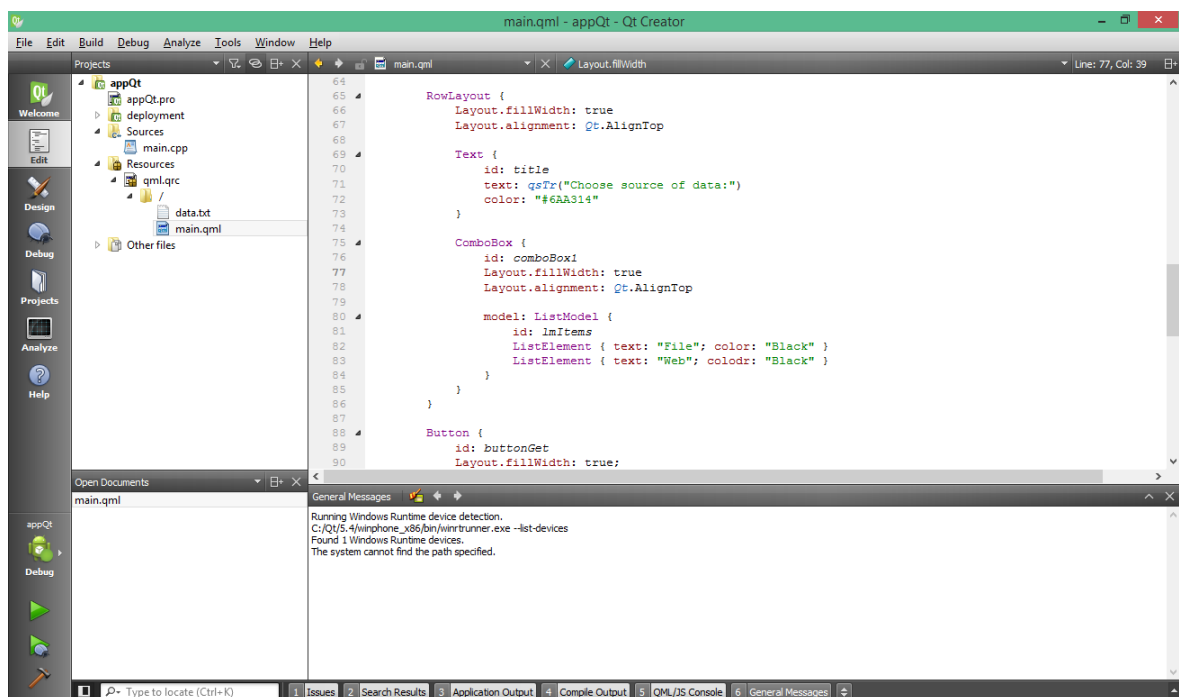


Obr. 10. Koncept aplikácie

Úlohou aplikácie je získať dáta z určitého zdroja. Poskytuje užívateľovi možnosť vybrať zdroj týchto dát ktorým môže byť súbor alebo webová stránka. Súbor je formátu .txt a je priamo súčasťou každej aplikácie. Pri zvolení web stránky sú dáta získavané z adresy [http://www.w3schools.com/website/Customers\\_MYSQL.php](http://www.w3schools.com/website/Customers_MYSQL.php). V oboch prípadoch sú zdrojom dáta serializované pomocou štandardu JSON. Po deserializovaní získaného reťazca sú následne tieto dáta vždy zobrazené do prehľadnej tabuľky a je vyvolané dialógové okno informujúce o počte načítaných záznamov ako aj o dobe trvania ich získavania. Získavanie dát zo zvoleného zdroja je aktivované stlačením tlačidla a pôvodné dáta z tabuľky sú nahradené novými. Pri vývoji každej aplikácie je cieľom dosiahnuť vzhľad užívateľského prostredia ktorý by sa čo najviac podobal natívnemu, alebo ak by to bolo možné reálne využíval natívne prvky.

### 3.2 Tvorba pomocou Qt

Na vývoj aplikácie pomocou nástroja Qt bolo použité IDE Qt Creator. Ide o prepracované vyzerajúce prostredie, poskytujúce všetko potrebné na vývoj aplikácii. Jeho inštalácia bola jednoduchá no pre testovanie na konkrétnej mobilnej platforme, v tomto prípade Android, bolo potrebné ho ešte správne nakonfigurovať. Pri vývoji je možné používať designer vstavaný do IDE umožňujúci rýchlo vytvoriť GUI aplikácie, prípadne listovať vo všetkých dostupných komponentoch UI. Tvorená aplikácia bola v tomto prípade typu Qt Quick.



Obr. 11. Qt Creator

### 3.2.1 Tvorba GUI

Pri tvorbe užívateľského rozhrania bol využívaný výhradne jazyk QML. Celé GUI bolo vytvorené vkladáním komponentov do stĺpcového elementu „*ColumnLayout*“. Tomu bolo nastavené odsadenie zo všetkých strán aby jeho vnútorné komponenty neboli vykreslené hneď od okrajov.

```
ColumnLayout {
    id: mainLayout
    anchors.fill: parent
    anchors.margins: 10
    spacing: 10

    ... //vnútorné komponenty
}
```

Prvým komponentom vo vnútri tohto stĺpcového elementu bol „*Label*“ ktorý zobrazoval nadpis aplikácie. Centrovanie textu bolo jednoducho realizovateľné pomocou dvoch pridaných parametrov. Pridaním bolo rozťahnutie na celú šírku stĺpcového elementu a následne horizontálne zarovnanie na stred.

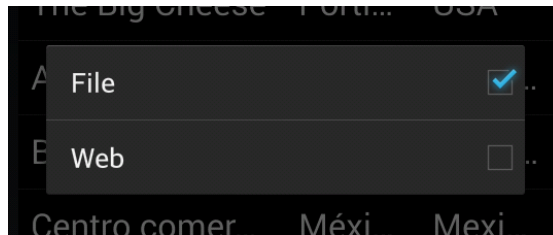
```
Layout.fillWidth: true
horizontalAlignment: Text.AlignHCenter
```

Ďalej bol pridaný riadkový element „*RowLayout*“ do ktorého bol vložený text a výberový nástroj „*ComboBox*“. Na tomto prípade je možné vidieť model/view architektúru, keďže tento komponent sa stará o zobrazenie položiek z jemu pridaného modelu. Aby bol tento výberový nástroj natiahnutý na celú zostávajúcu šírku riadkového elementu bol podobne ako pri nadpise pridaný parameter na vyplnenie šírky

```
RowLayout {
    Layout.fillWidth: true
    Layout.alignment: Qt.AlignTop

    Text {
        id: title
        text: qsTr("Choose source of data:")
        color: "#6AA314"
    }
    ComboBox {
        id: comboBox1
        Layout.fillWidth: true // vyplnenie zostávajúcej šírky
        Layout.alignment: Qt.AlignTop
        model: ListModel { // Pridanie zobrazovaného modelu
            id: lmItems
            ListElement { text: "File"; color: "Black" }
            ListElement { text: "Web"; color: "Black" }
        }
    }
}
```

Po poklepaní na výberový nástroj sa v aplikácii zobrazí nasledujúce natívne vyzerajúce výberové okno.



Obr. 12. ComboBox, Qt

Ďalším komponentom bolo tlačidlo aktivujúce získavanie dát. To je vykonané volaním vytvorenej funkcie po stlačení tlačidla.

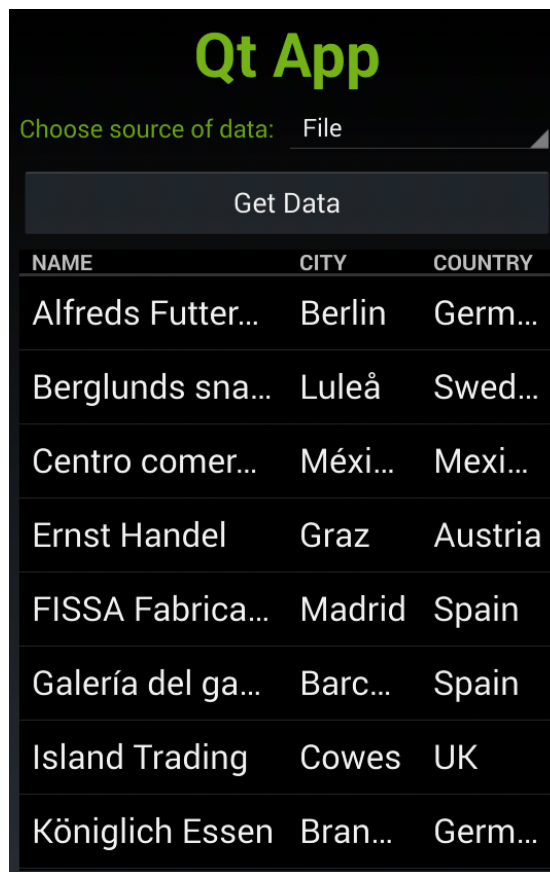
```
Button {
    id: buttonGet
    Layout.fillWidth: true;
    Layout.alignment: Qt.AlignTop
    text: qsTr("Get Data")
    onClicked: getData(); // získavanie dát
}
```

Posledným komponentom bola zobrazovacia tabuľka. Vo väčšine mobilných platformách nie je žiadny natívny tabuľkový komponent určený na zobrazovanie dát. Samozrejme je možné ho napríklad v natívnom Androide vytvoriť pomocou komponentu „*TableLayout*“ no je potrebné ďalšie upravovanie. Qt i keď sa nejedná o natívnu súčasť platformy poskytuje už pred pripravený komponent „*TableView*“ určený presne na túto úlohu. Jednoducho je možné pridať stĺpce s ich šírkami, názvami a prepojením na dáta, ktoré v nich majú byť zobrazované. Umožňuje priamo užívateľovi ťahom prsta presúvať jednotlivé stĺpce prípadne ich zväčšovať či zmenšovať. Zobrazované dáta sú opäť získavané z pridaného modelu ktorý je napĺňaný dynamicky po aktivovaní tlačidla. Tento posledný komponent je nastavený aby vyplňal celý zostávajúci horizontálny aj vertikálny priestor aplikácie.

```
TableView {
    id: tableview
    Layout.fillWidth: true
    Layout.fillHeight: true
    Layout.alignment: Qt.AlignTop
    model: ListModel {}

    TableViewColumn {
        role: "name"
        title: "Name"
        width: tableview.width/2
    }
    ... // ďalšie pridané stĺpce
}
```

Po vykonaní predošlého je už aplikácia po grafickej stránke hotová. Poslednou drobnou úpravou bola zmena používanej témy Android platformy na klasickú tmavú tému Holo keďže Qt štandardne používa svetlú Holo.Light. Táto zmena bola vykonaná priamo v IDE v nastaveniach Build & Run cieľovej platformy v súbore AndroidManifest.xml. Výsledný vzhľad aplikácie je zobrazený na nasledujúcom obrázku.

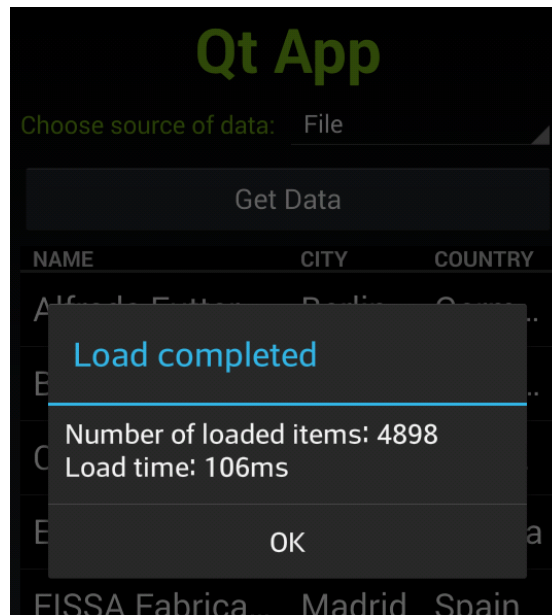


Obr. 13. Aplikácia Qt

V jazyku QML bol definovaný ešte jeden komponent a to MessageBox. Vďaka nemu je volané natívne dialógové okno s upozornením. Tento komponent je viditeľný až po jeho vyvolaní. Na zjednodušenie vyvolania okna bola pridaná vnútorná funkcia. Vyvolanie okna bolo využité po načítaní záznamov na oznámenie ich počtu a doby načítavania.

```
MessageDialog {
    id: messageDialog
    title: qsTr("Load completed")
    function show(caption) {
        messageDialog.text = caption;
        messageDialog.open();
    }
}
```





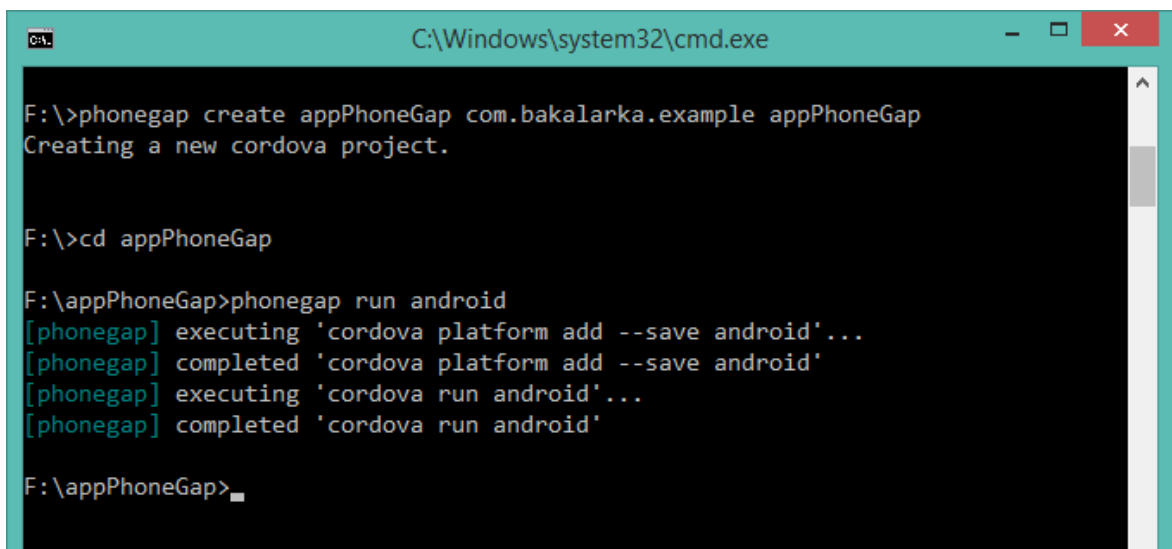
Obr. 14. MessageBox, Qt

### 3.2.3 Zhodnotenie a postrehy z vypracovávania

Tento framework bol v mnohých ohľadoch nápomocný a výrazne urýchlil prácu na aplikácii. Vstavaný designer bol výraznou pomocou pri zostavovaní GUI keďže stačilo vždy daný komponent nájsť v zozname a vložiť. Nebola potrebná žiadna prvotná znalosť jazyka QML. Následná editácia komponentov v tomto jazyku bola spravidla vždy veľmi jednoduchá. Nápona v IDE je prehľadná a na veľmi dobrej úrovni. Internetové fóra týkajúce sa tejto témy zasa až na takej dobrej úrovni neboli. Medzi kladné stránky tohto frameworku by som zaradil možnosť používať pred pripravené komponenty ktoré nie sú bežnou súčasťou cieľovej platformy, využívanie natívnych API danej platformy a hlavne možnosť jednoducho prepnúť cieľovú platformu a napríklad testovať aplikáciu na práve používanom operačnom systéme bez nutnosti emulátora. To bolo veľmi vhodné hlavne z dôvodu že zavádzanie aplikácie na cieľovú platformu je v porovnaní s konkurenciou relatívne pomalé. To má na svedomí hlavne veľkosť vyvíjanej aplikácie ktorá po nainštalovaní na testovacie zariadenie dosahuje približne 43 MB. Táto hodnota pre dnešné telefóny nie je žiadny problém no na takú jednoduchú aplikáciu je to určite dosť. Pri vývoji došlo pár krát k tomu že sa vykonané zmeny neprejavili na zavedenej aplikácii a bolo potrebné vykonať „clean“ projektu. Okrem trochu pomalšieho štartu aplikácia reaguje veľmi svižne. Pri spočítaní neprázdnych riadkov a pri udržaní určitého štandardu prehľadnosti písaného kódu bolo zaznamenaných 95 pridaných riadkov kódu v jazyku QML a ako bolo spomínané jazyk C++ nebolo vôbec potrebné využiť.

### 3.3 Tvorba pomocou PhoneGap

Pre využitie tohto nástroja bolo najskôr potrebné nainštalovať program NodeJS a následne mohol byť Phonegap stiahnutý a nainštalovaný pomocou zadania jednoduchého príkazu „*npm install -g phonegap*“ do príkazového riadku. Následne bol pomocou príkazu „*phonegap create nazovAdresara com.nazov.balicku nazovAplikacie*“ vytvorený adresár so všetkými potrebnými súbormi. Jednoduchým bolo aj zostavenie a spustenie aplikácie na cieľovej platforme. Bolo potrebné sa nastaviť do vytvoreného adresára a zadať príkaz „*phonegap run nazovPlatformy*“. Ten vykoná build aplikácie a v prípade že nie je pripojené reálne zariadenie spustí emulátor a zavedie aplikáciu. Celý postup vytvorenia a zavedenia aplikácie je na nasledujúcom obrázku.



```
C:\Windows\system32\cmd.exe

F:\>phonegap create appPhoneGap com.bakalarka.example appPhoneGap
Creating a new cordova project.

F:\>cd appPhoneGap

F:\appPhoneGap>phonegap run android
[phonegap] executing 'cordova platform add --save android'...
[phonegap] completed 'cordova platform add --save android'
[phonegap] executing 'cordova run android'...
[phonegap] completed 'cordova run android'

F:\appPhoneGap>
```

Obr. 15. Vytvorenie a zostavenie ukážkovej aplikácie PhoneGap

Vzhľadom na charakter tohto nástroja je jasné, že žiadnym vývojovým prostredím nedisponuje. Je však možné využiť akékoľvek iné IDE určené na tvorbu a editáciu web stránok. V mojom prípade to bolo prostredie Eclipse s doinštalovaným modulom PDT. Pri práci s týmto frameworkom som na niektoré úpravy pre platformu Android použil jeho oficiálne vývojové prostredie Android Studio.

#### 3.3.1 Tvorba GUI

Celé GUI je vytvorené pomocou jazyka HTML a k nemu pripojenému CSS. Pozadie aplikácie je nastavené CSS štýlom pre značku „*body*“ tak aby simulovalo tmavú Android tému. Všetky elementy rozhrania sú umiestnené vo vnútri nasledujúcich značiek „*div*“. K nim sú priradené CSS triedy zabezpečujúce napríklad odsadenie od okolitých strán.

```
<div class="wrapper" >
  <div class="app" id="app" >
    ...
  </div>
</div>
```

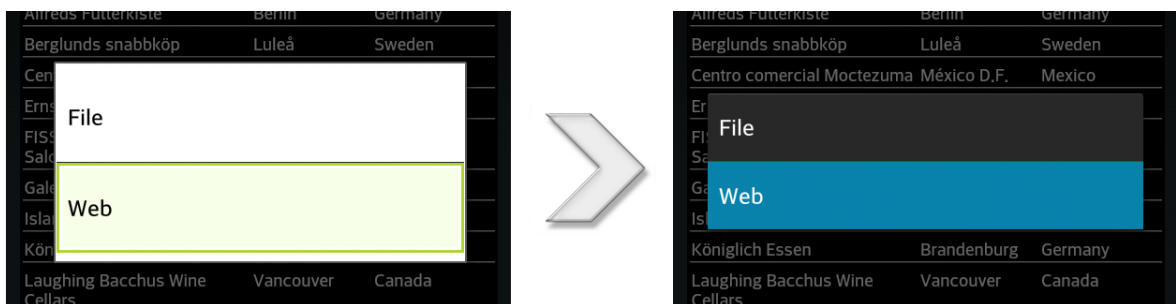
Hlavný nadpis aplikácie je vytvorený pomocou značky „h1“ a centrováný na stred je pomocou nastaveného automatického odsadenia od oboch strán v jeho CSS.

```
margin-left: auto;
margin-right: auto;
```

Ako výberový nástroj bol použitý ovládací prvok „select“ ktorý bol ešte obalený ďalším elementom div s pridanou CSS triedou. Jej úlohou bolo aby sa tento výberový prvok rozťahol vždy na celú zostávajúcu šírku obrazovky.

```
<div class="fillSelect">
  <select id="slct">
    <option value="file">File</option>
    <option value="web">Web</option>
  </select>
</div>
```

Tento element však nevyzeral vôbec natívne keďže využíval štýl pre internetový prehliadač. Po zostavení aplikácie bol jej projekt následne otvorený v IDE Android Studio. V súbore androidManifest.xml bola zmenená používaná téma opäť na Holo. Výsledok zmeny zobrazuje nasledujúci obrázok.



Obr. 16. Pôvodný a zmenený vizuálny štýl nástroja „select“, PhoneGap

Ďalším elementom bolo tlačidlo s definovanou funkciou volanou po jeho aktivácii. Taktiež je jeho šírka nastavená v CSS na 100%.

```
<button id="btn" onclick="getValues()">Get Data</button>
```

Posledným pridaným elementom bola tabuľka vytvorená pomocou značiek table. Všetky jeho elementy boli nastavené na blokové zobrazovanie, aby mohla byť tabuľka jednoducho editovaná pomocou pridávania riadkov.

```
thead, tbody, tr, td, th { display: block; }
```

Vo vnútri tabuľky boli v značkách „thead“ definované jednotlivé stĺpce s názvami a používanými CSS triedami. Tie určujú ich šírku, obtekanie a zarovnanie textu.

```
<thead>
  <tr>
    <th class="col1">NAME</th>
    <th class="col2">CITY</th>
    <th class="col3">COUNTRY</th>
  </tr>
</thead>
```

Dáta tabuľky sú pridávané medzi značky „tbody“ pomocou JavaScriptu. Z vizuálneho hľadiska je aplikácia hotová a jej vzhľad je zobrazený na nasledujúcom obrázku.



Obr. 17. Aplikácia PhoneGap

### 3.3.2 Vnútoraná logika

Funkcia spúšťaná tlačidlom je napísaná v jazyku JavaScript. Keďže bol v predchádzajúcej Qt aplikácii použitý tento jazyk, mohol byť vytvorený kód použitý aj v tejto aplikácii. Bolo však nutné vykonať určité úpravy. Taktiež bola do projektu pridaná knižnica JQuery pre zjednodušenie práce. Súbor s dátami bol jednoducho pridaný medzi súbory projektu.

Zmena nastala napríklad pri zisťovaní zadaného zdroja dát a to nasledovne

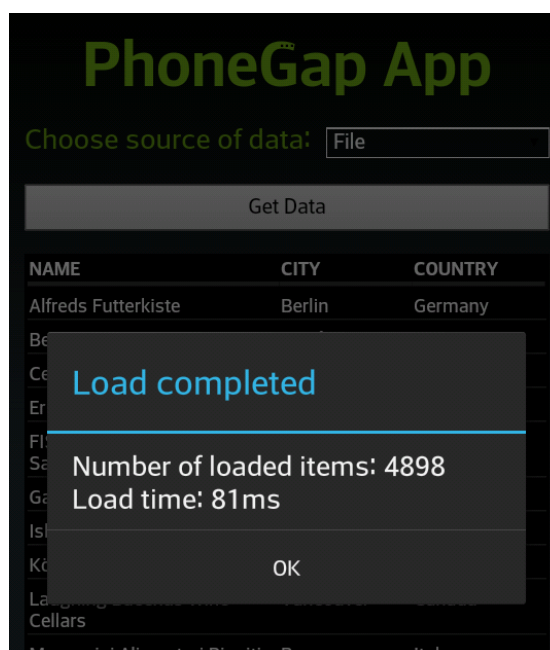
```
var selected = $('#slct').val();
```

Vo vnútornej funkcii testujúcej dokončenie získavania dát bolo nutné vykonať zmeny pri volaní dialógového okna. PhoneGap dokáže vyvolávať natívne notifikačné okná no je potrebné do adresára projektu pomocou príkazu „*cordova plugin add org.apache.cordova.dialogs*“ doinštalovať potrebný zásuvný modul. Následne je možné k nemu cez JavaScript pristupovať.

```
navigator.notification.alert(  
    'Number of loaded items: ' + arr.length + // správa  
    '\nLoad time: ' + loadTime + 'ms',  
    null, // callback funkcia  
    'Load completed', // titulok  
    'OK' // názov tlačidla  
);
```

Aby bola zobrazovaná notifikácia v nastavenej natívnej tmavej téme, bolo ešte potrebné zmeniť určité nastavenie vo vytvorenom Android projekte. V súbore `appPhoneGap/src/org.apache.cordova/dialogs/notification.java` je totiž napevno nastavená používaná téma notifikačného okna na svetlú `Holo.Light`. Bola potrebná nasledujúca zmena.

```
AlertDialog.Builder dlg = new AlertDialog.Builder(  
    cordova.getActivity(), AlertDialog.THEME_HOLO_DARK);
```



Obr. 18. AlertDialog, PhoneGap

Poslednou zmenou bolo zobrazovanie získaných dát. To bolo vykonané pomocou vytvárania vlastných HTML riadkov tabuľky.

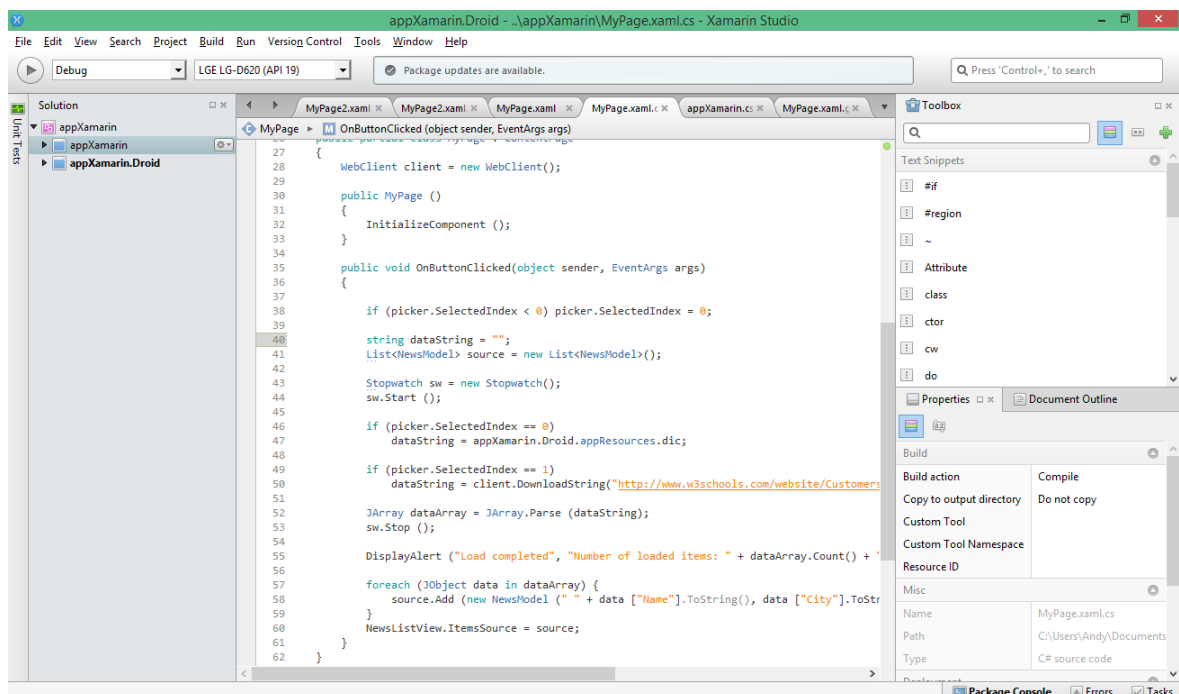
```
for(var i = 0; i < arr.length; i++) {
    stringTable += '<tr> <td class="col1">' +arr[i].Name +
                  '</td> <td class="col2">' + arr[i].City +
                  '</td> <td class="col3">' + arr[i].Country +
                  '</td> </tr>';
}
$("#stHGHT").html(stringTable);
```

### 3.3.3 Zhodnotenie a postrehy z vypracovania

Nástroj PhoneGap funguje presne tak ako sa od neho očakáva. Na vytváranie kompletných mobilných aplikácii stačí disponovať znalosťami webových technológií. Avšak vytvoriť kvalitný a responzívny layout ktorý je na mobilných zariadeniach nevyhnutný vyžaduje mať v tomto smere určitú úroveň zručnosti. Webový manuál k tomuto nástroju je trochu menej prehľadný ale dostačujúci. Veľkým plusom tohto prístupu je možnosť získať inštalčné súbory pre všetky platformy jednoducho vložení projektom na webovú stránku Phonegapu. Prínosom je taktiež možnosť testovať aplikácie priamo v prehliadači a bez nutnosti zostavovania aplikácii, ako je tomu u konkurenčných nástrojov. K dispozícii je taktiež program PhoneGap Desktop, ktorý urýchľuje prácu tým, že pri jeho používaní nie je potrebné stránku ručne obnovovať, keďže to je po uložení zmien v projekte vykonané automaticky. Môže tu však vzniknúť problém ako v tomto prípade, kedy prehliadač blokoval načítavanie dát zo súboru pomocou JavaScriptu, alebo pri zlyhaní volania notifikačného okna určeného pre mobilné platformy. V takýchto prípadoch bol potrebný build aplikácie a test na cieľovej platforme. Ten trval relatívne dlho obzvlášť pokiaľ bolo potrebné vykonať zmeny vo vytvorenom Android projekte keďže bolo na úpravu a nasadenie použité ďalšie vývojové prostredie. Kladom je určite možnosť používať niektoré natívne API ako spomínané notifikačné okná no na druhej strane je zo zvlhľadu očividné, že sa o natívnu aplikáciu nejedná. V takom prípade je potrebné využiť ďalšie nástroje schopné takýto zvlhľad u ovládacích prvkov simulovať. Aplikácia na testovacím zariadení zaberá približne 3,3 MB. Pri zobrazovaní veľkého množstva dát bolo zaznamenané výrazné omeškanie pri vykresľovaní dát tabuľky. Ich získanie síce trvalo len pár milisekúnd no ich vykreslenie asi 3 až 4 sekundy. Aplikácia vtedy pomaly reagovala aj na základné podnety ako bolo natočenie obrazovky. Pre vytvorenie tejto aplikácie bolo potrebné pridať alebo upraviť približne 30 riadkov HTML, 80 riadkov CSS a 30 riadkov JavaScriptu. Spolu teda túto aplikáciu tvorí 140 riadkov kódu.

### 3.4 Tvorba pomocou Xamarin

Framework Xamarin disponuje inštaláčnou aplikáciou ktorá sa stará o čo najjednoduchšiu inštaláciu. V podstate nebolo potrebné nič len nechať ju pracovať. Sama zistila prítomnosť všetkých prvkov a v prípade že niektoré chýbali automaticky ich stiahla. Nebola potrebná žiadna dodatočná konfigurácia. Pri vývoji bolo používané vstavané IDE Xamarin Studio. Ide o veľmi nenápadne vyzerajúce prostredie, ktoré ale disponuje všetkým potrebným pre vývoj aplikácii. Aplikácia bola vytváraná pomocou nového nástroja Xamarin.Forms. Aby mohol byť využívaný, bolo potrebné vybaviť si študentskú licenciu, keďže tento nástroj je dostupný až v platených balíkoch Xamarinu.



Obr. 19. Xamarin Studio

#### 3.4.1 Tvorba GUI

Vytvorenie grafického prostredia bolo možné realizovať pomocou jazyka C# alebo pomocou XML ktorý bol pre jeho väčšiu prehľadnosť nakoniec použitý. Všetky UI prvky boli umiestnené do vnútra komponentu „StackLayout“ ktorého orientácia bola nastavená vertikálne, aby fungoval ako stĺpcový komponent. Taktiež odsadzoval UI od okolitých okrajov.

```
<StackLayout Orientation="Vertical"
  VerticalOptions="FillAndExpand"
  HorizontalOptions="FillAndExpand"
  Padding="10,10"
  Spacing="10">
</StackLayout>
```

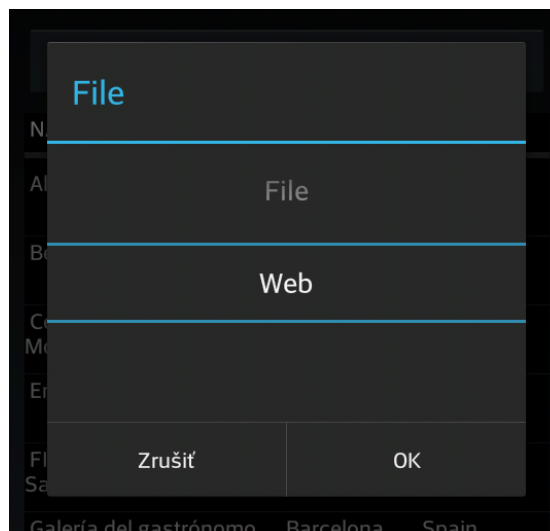
Prvým bol opäť nadpis vytvorený pomocou komponentu „Label“. Tomu boli pridané nastavenia písma a centrovanie na stred pomocou parametru horizontálnych možností.

```
HorizontalOptions="CenterAndExpand"
```

Ako výberový nástroj bol použitý komponent „Picker“ ktorý je dostupný aj ako natívny prvok cieľovej platformy. Taktiež bol pomocou horizontálnych možností nastavený aby vyplnil zostávajúci priestor s textom v spoločnom riadkovom komponente.

```
<Picker
  x:Name="picker"
  SelectedIndex="0"
  HorizontalOptions="FillAndExpand"
  Title="File">

  <Picker.Items>
    <x:String>File</x:String>
    <x:String>Web</x:String>
  </Picker.Items>
</Picker>
```



Obr. 20. Picker, Xamarin

Ďalej bolo pridané klasické tlačidlo vyvolávajúce získavanie dát.

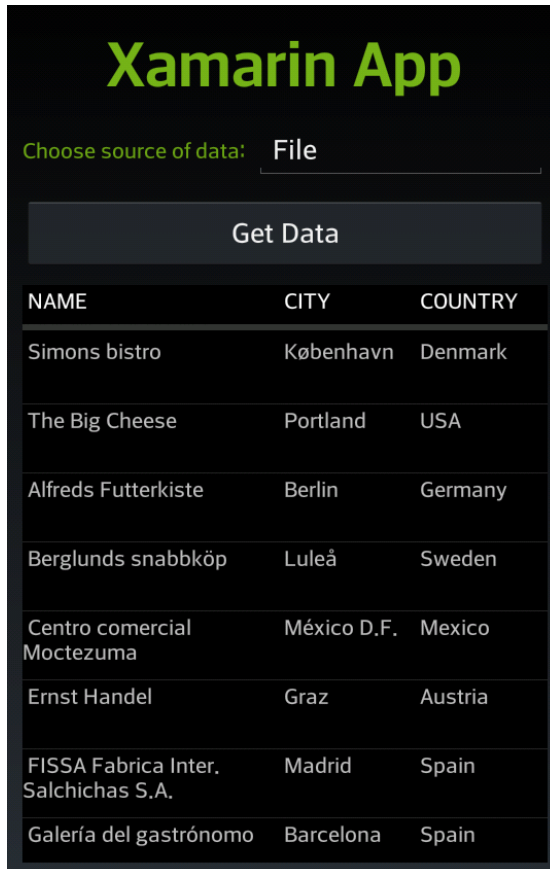
```
<Button Text="Get Data"
  HorizontalOptions="FillAndExpand"
  Clicked="OnButtonClicked" />
```

Na zobrazenie dát je opäť potrebné tabuľku vytvoriť, keďže Xamarin nedisponuje žiadnym pred pripraveným viac stĺpcovým komponentom schopným reagovať na dotykové udalosti. Komponenty „ListView“ a „TableView“ ktorým Xamarin disponuje sú dáta v súčasnosti schopné dynamicky zobrazovať len v jednom stĺpci. Pre zobrazenie tabuľky bol upravený komponent „ListView“ tak aby disponoval komponentom „StackLayout“ s horizontálnou

orientáciou. Pre vytvorenie jednoduchkej tabuľky bolo potrebné zostaviť nasledujúcu štruktúru do seba vnorených komponentov. V každom vytvorenom riadku bol text prvku „Label“ previazaný so zvoleným prvkom s dátového modelu.

```
<ListView x:Name="NewsListView">
  <ListView.ItemTemplate>
    <DataTemplate>
      <ViewCell>
        <ViewCell.View>
          <StackLayout Orientation="Horizontal"
            HorizontalOptions="FillAndExpand" >
            <Label Text="{Binding Name}"
              HorizontalOptions="Start"
              WidthRequest = "200">
          </Label>
            <Label Text="{Binding City}"
              HorizontalOptions="CenterAndExpand"
              WidthRequest = "100">
          </Label>
            <Label Text="{Binding Country}"
              WidthRequest = "100">
          </Label>
        </ViewCell.View>
      </ViewCell>
    </DataTemplate>
  </ListView.ItemTemplate>
</ListView>
```

Po vytvorení tohto komponentu bola aplikácia po vzhľadovej stránke hotová a jej výsledný vzhľad je zobrazený na nasledujúcom obrázku.



Obr. 21. Aplikácia Xamarin

### 3.4.2 Vnútorná logika

Pri vytvorení súboru .xaml s definovaným UI bol automaticky vytvorený súbor s rovnakým názvom a príponou .xaml.cs. V tomto súbore bolo možné definovať vnútornú logiku pomocou jazyka C#. Bolo potrebné vytvoriť triedu s dátovým modelom vkladaných dát.

```
public class NewsModel {
    public string Name { get; set; }
    public string City { get; set; }
    public string Country { get; set; }

    public NewsModel (string Name, string City, string Country) {
        this.Name = Name;
        this.City = City;
        this.Country = Country;
    }
}
```

Následne vo funkcii volanej po stlačení tlačidla bol zistený výber zdroja dát a bolo začaté meranie času pomocou inštalácie triedy „Stopwatch“. Vďaka pridanému názvu výberového nástroja mohla byť následne jednoducho zistená užívateľova voľba zdroja. Pre získavanie dát zo súboru bol pridaný textový súbor s dátami a súbor .resx určený na definovanie zdrojov. Po pridaní súboru s dátami medzi zdroje bolo na tento súbor následne možné pristupovať pomocou zvoleného mena. Na získavanie dát z webovej stránky bola použitá funkcia z triedy WebClient.

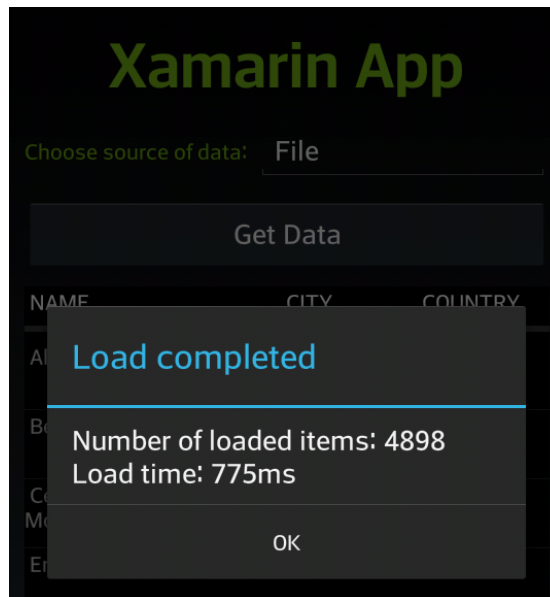
```
Stopwatch sw = new Stopwatch();
sw.Start ();

if (picker.SelectedIndex == 0)
    dataString = appXamarin.Droid.appResources.data;

if (picker.SelectedIndex == 1)
    dataString = await client.DownloadStringTaskAsync(
        "http://www.w3schools.com/website/Customers_MYSQL.php");
```

Pre deserializáciu získaných dát bolo potrebné stiahnuť externú JSON knižnicu. Tá bola stiahnutá priamo v IDE. Následne bolo z reťazca vytvorené pole JSON objektov. Hneď po dokončení deserializácie bol zastavený meraný čas a bolo vyvolané notifikačné okno informujúce o dokončení načítavania dát.

```
JArray dataArray = JArray.Parse (dataString);
sw.Stop ();
await DisplayAlert ("Load completed",
    "Number of loaded items: " + dataArray.Count() +
    "\nLoad time: " + sw.ElapsedMilliseconds + "ms",
    "OK");
```



Obr. 22. DisplayAlert, Xamarin

Do zoznamu ktorý bol typu vytvoreného modelu resp. triedy boli postupne pridané všetky dáta a následne bol tento model pridaný ako zdroj záznamov vytvorenej tabuľky.

```
foreach (JObject data in dataArray) {  
    source.Add (new NewsModel (data ["Name"].ToString(),  
                                data ["City"].ToString(),  
                                data ["Country"].ToString()));  
}  
NewsListView.ItemsSource = source;
```

### 3.4.3 Zhodnotenie a postrehy z vypracovania

Pri začiatku práce s nástrojom Xamarin bolo najskôr potrebné sa zorientovať v jeho možnostiach keďže Xamarin.Forms momentálne nedisponuje žiadnym GUI návrhovým nástrojom. Internetová dokumentácia k tomuto nástroju bola nie vždy dostatočujúca a bolo potrebné hľadať aj na iných stránkach. Naproti tomu fórum do ktorého bola potrebná registrácia bolo už na dobrej úrovni. Pri vytváraní funkcie na prácu s dátami sa prejavil jeden z hlavných plusov Xamarinu a to používanie jazyka C#. I keď bolo pri vývoji potrebné sťahovať knižnicu JSON tento jazyk poskytuje mnoho pripravených tried a funkcií pre zrýchlenie a zefektívnenie práce. Zaujímavá bola MVVM architektúra na ktorú si ale treba najskôr zvyknúť. Vývojové prostredie poskytuje množstvo nastavení a dovoľuje napríklad jednoducho vybrať ktoré súbory majú byť kompilované a ktoré majú byť považované za zdroje. Snahu o inovácie tohto nástroja bolo vidieť keďže len počas práce s ním došlo k dvom aktualizáciám. Na druhej strane inovovať je naozaj potrebné keďže Xamarin Studio sa pri práci ukázalo ako veľmi nestabilné. Počas práce došlo k jeho štyrom pádom a dvom pádom

Android emulátora okamžite po zavedení aplikácie. Pri vypracovávaní sa taktiež vyskytol problém s nefunkčnosťou release verzie danej aplikácie. Po viacerých testoch na rôznych zariadeniach a rôznych konfiguráciách sa release verziu podarilo sfunkčniť až po kompletnom preinštalovaní vývojového prostredia na najaktuálnejšiu verziu. Pozitívnym je fakt že Xamarin používa natívne ovládacie prvky no niektoré základné, ako napríklad ComboBox či DataGrid sa v jeho repertoári bohužiaľ nenachádzajú. Existuje tu možnosť stiahnuť ďalšie pred pripravené komponenty no za tie je potrebné si zaplatiť. Zavádzanie aplikácie bolo veľmi rýchle a to hlavne vďaka použitiu zdieľaného MonoRuntime. Veľkosť aplikácie v release verzii sa pohybovala na úrovni 15 MB. Pri zavádzaní sa však občas stalo že sa aplikácia po vykonaní zmien v kóde neaktualizovala. Jej spúšťanie trvalo najdlhšie oproti konkurencii. Dáta Tabuľky boli zobrazené rýchlo po ich načítaní, no pri ich rýchlejšom prezeraní bolo vidieť trhanie. Inak bola aplikácia pomerne svižná. Na jej vytvorenie bolo potrebných približne 75 riadkov v jazyku XML a 30 v jazyku C#. Spolu bolo na vytvorenie tejto aplikácie potrebných približne 105 riadkov kódu.

### 3.5 Vyhodnotenie

Jedným z hlavných posudzovaných kritérií bol výkon každej aplikácie pri práci z dátami. Porovnávané boli časy načítavania malého množstva dát z Internetu, veľkého množstva dát načítavaných z pribaleného súboru a doba ich vykresľovania. Testy boli vykonávané opakovane aby bola dosiahnutá čo najväčšia objektívnosť. Zistenia sú nasledovné:

- **Qt** – načítavanie zo súboru trvalo približne 100 až 120 ms, načítavanie z webu pri prvom spustení aplikácie asi 1,3 sekundy. Po druhom a ďalších opakovaní bola táto doba na úrovni medzi 220 až 300 ms. Vykreslenie dát nastalo takmer ihneď.
- **PhoneGap** – načítanie zo súboru bolo v časovom intervale 50 až 90 ms. Prvé získavanie dát z webu 600 ms a pri opakovaní bola doba trvania 200 až 250 ms. Veľké množstvo dát však trvalo vykresliť vždy niekoľko sekúnd.
- **Xamarin** – získavanie dát zo súboru zabralo tejto aplikácii 250 až 450 ms. Prvé získavanie webových dát trvalo priemerne asi 800 ms a pri opakovaní sa táto hodnota pohybovala v rozmedzí 220 až 300 ms. Dáta boli zobrazené takmer okamžite po ich načítaní

Z výsledkov merania je možné si všimnúť výraznú podobnosť jednotlivých časov pri opakovanom načítaní dát z webu. Z týchto hodnôt je možné prehlásiť že pri načítaní malého počtu dát je výsledný čas majoritne ovplyvnený aktuálnou rýchlosťou pripojenia.

Vo výslednom zhodnotení boli posudzované kritériá týkajúce sa nie len finálnej aplikácie ale aj procesu jej tvorby a možností použitého nástroja. Konečné hodnotenie je zobrazené vo výslednej tabuľke. Pri každom kritériu je pre konkrétny posudzovaný nástroj uvedená číselná hodnota 1, 2 alebo 3. Tá vyjadruje umiestnenie posudzovaného nástroja v rámci daného kritéria. Číslo 1 teda predstavuje prvé miesto, čiže najlepšie hodnotenie. Naopak hodnotou 3 je označený najhorší výsledok. V spodnej časti tabuľky sa nachádzajú spriemerované hodnoty umiestnení pre každý nástroj. Porovnávanými kritériami boli:

- **Zisk dát** – porovnávaná je rýchlosť získavania dát zo súboru
- **Vykresľovanie** – porovnávaná je doba nutná na vykreslenie dát
- **Doba spúšťania** – hodnotí čas potrebný na spustenie aplikácie.
- **Plynulosť** – kritérium hodnotí plynulosť zobrazovania dát pri rýchlom prezeraní
- **Natívnosť** – hodnotenie aplikácii podľa natívnosti vzhľadu a schopnosti využívať natívne API danej platformy.
- **Veľkosť** – hodnotí výslednú veľkosť aplikácie v release verzii po jej nainštalovaní na testovacie zariadenie.
- **Testovanie** – Kritérium hodnotí dobu nutnú k zavedeniu aplikácie
- **Komplexnosť** – kritérium hodnotí či bolo pri vývoji nutné dopĺňať daný framework o ďalšie komponenty alebo moduly
- **Doba vývoja** – toto kritérium sa snaží porovnať implementačnú náročnosť aplikácie. Hodnotená je doba nutná na vytvorenie aplikácie, za predpokladu že znalosti vývojára sú na rovnakej úrovni u všetkých nástrojov a používaných technológií. Pri rozhodovaní napomáhal hlavne počet riadkov kódu nutný k vytvoreniu cieľovej aplikácie.

Tab. 4. Výsledné zhodnotenie

	Qt	PhoneGap	Xamarin
Zisk dát	2	1	3
Vykresľovanie	2	3	1
Doba spúšťania	2	1	3
Plynulosť	1	3	2
Natívnosť	2	3	1
Veľkosť	3	1	2
Testovanie	2	3	1
Komplexnosť	1	3	2
Doba vývoja	1	3	2
<b>PRIEMER</b>	<b>1,78</b>	<b>2,33</b>	<b>1,89</b>

## ZÁVER

Cieľom tejto bakalárskej práce bolo porovnať a zhodnotiť rôzne prístupy k multiplatformnému vývoju mobilných aplikácií. V teoretickej časti je možné nájsť rozdelenie rôznych metód multiplatformného vývoja s objasnením ich princípu fungovania a taktiež vzájomných rozdielov. Pri porovnávaní ich vlastností bolo zistené, že ani o jednom z nich nie je možné jednoznačne prehlásiť, že ide o najlepší alebo najhorší spôsob vývoja.

V praktickej časti bol demonštrovaný reálny vývoj aplikácií pomocou najznámejších multiplatformných nástrojov. Tými boli Qt, PhoneGap a Xamarin. Aplikácie boli následne vzájomne porovnané. Toto porovnanie bolo pomerne dosť časovo náročné a pri vypracovávaní bolo nutné používať jazyky C/C++, QML, C#, XML, HTML, CSS a JavaScript.

Vo výslednom porovnaní, ktoré je zobrazené v tabuľke 4, zvíťazil framework Qt. Hlavnými prednosťami aplikácie vytvorenej pomocou tohto nástroja sú v prvom rade výkon a plynulosť. Demonštratívna aplikácia bola vytvorená v najkratšom čase a jej kód bol oproti konkurencii taktiež najkratší. To je zapríčinené veľkou komplexnosťou tohto nástroja, ktorý disponuje množstvom pred pripravených komponentov. Za pochvalu stojí aj vývojové prostredie Qt Creator a možnosť jednoduchou zmenou nastavenia vyvíjať aplikácie nielen pre mobilné, ale aj pre desktopové platformy a to z jedného spoločného kódu. Daň za všetky tieto výhody predstavuje pomalší štart a hlavne veľkosť výslednej aplikácie.

Na druhej priečke sa umiestnil nástroj Xamarin. Jeho výraznou prednosťou je využívanie natívnych prvkov UI a natívnych API. Pri vývoji bolo možné si všimnúť rýchleho zavádzania aplikácie a taktiež rýchle zobrazovanie zdrojových dát. Do plusov tohto nástroja spadá určite aj používanie jazyka C#. Aplikácia bola aj vďaka nemu vytvorená podobne krátkym kódom ako tomu bolo u Qt, no s výslednou približne tretinovou veľkosťou. Nevýhodami Xamarin aplikácie bol pomalý štart a dlhšia doba získavania dát. Pri vývoji boli dostupné iba základné UI prvky. Negatíva uzatvára nestabilné prostredie Xamarin Studio.

Tretie miesto obsadil nástroj PhoneGap. Aplikácia vytvorená týmto nástrojom predbehla svoju konkurenciu v rýchlosti načítavania dát, ako aj v rýchlosti spúšťania aplikácie. Taktiež dosahovala najmenšej veľkosti. Urýchlenie práce umožňovala možnosť použiť na testovanie aplikácie webový prehliadač. Vzhľadom na charakter tejto aplikácie bola výrazným plusom možnosť používať natívne API danej platformy. Nevýhodou bolo dlhé čakanie na vykreslenie získaných údajov a vzhľad UI prvkov, ktoré zďaleka nevyzerali natívne. Na vytvorenie tejto aplikácie bol taktiež potrebný najdlhší kód.

**ZOZNAM POUŽITEJ LITERATURY**

- [1] DIGIA. Qt Library website [online]. 2015 [cit. 2015-01-28]. Dostupné z: <http://qt-project.org/>
- [2] RISCHPATER, Ray. Application Development with Qt Creator, 2nd Edition. Packt Publishing - ebooks Account, 2014. 2 edition. ISBN 978-1784398675.
- [3] SHOTTS, Kerri. PhoneGap 3.x Mobile Application Development Hotshot. United Kingdom: Packt Publishing - ebooks Account, 2014. 2 edition. ISBN 978-1783287925.
- [4] REYNOLDS, Mark. Xamarin Mobile Application Development for Android. Birmingham: Packt Publishing, 2014. ISBN 978-178-3559-169.
- [5] BLANCHETTE, Jasmin a Mark SUMMERFIELD. C GUI Programming with Qt 4: Prentice Hall Open Source Software Development Series. 2. ed. Trolltech ASA: Prentice hall, xxi, 718 s. ISBN 978-0-13-235416-5.
- [6] PRATA, Stephen. Mistrovství v C++. 3. aktualiz. vyd. Překlad Boris Sokol. Brno: Computer Press, 2007, 1119 s. ISBN 978-80-251-1749-1.
- [7] CHROBOCZEK, Martin. *Grafická uživatelská rozhraní v Qt a C: [plně kompatibilní s Qt 5]*. 1. vyd. Brno: Computer Press, 2013, 392 s. ISBN 978-80-251-4124-3.
- [8] DIGIA. *Qt Documentation* [online]. [cit. 2015-03-25]. Dostupné z: <http://doc.qt.io/>
- [9] Qt (software). *Wikipédia: Slobodná encyklopédia* [online]. [cit. 2015-03-29]. Dostupné z: [http://en.wikipedia.org/wiki/Qt\\_\(software\)](http://en.wikipedia.org/wiki/Qt_(software))
- [10] Benefits and Disadvantages of Cross-Platform Mobile Apps. CERAIT INC. MOBILE APPS DEVELOPMENT LAB. [online]. [cit. 2015-03-31]. Dostupné z: <http://www.ceraitmobileapps.com/blog/cross-platform-mobile-apps>
- [11] XANTHOPOULOS, Spyros a Stelios XINOGALOS. *Proceedings of the 6th Balkan Conference in Informatics: A comparative analysis of cross-platform development approaches for mobile applications* [online]. S.l.: ACM, 2013, s. 213-220 [cit. 2015-04-10]. ISBN 978-145-0318-518.
- [12] ASHLEY, Kevin. CREATING BEAUTIFUL CROSS-PLATFORM APPS FOR IOS, ANDROID AND WINDOWS WITH VISUAL STUDIO. PART I. USER

- INTERFACE. *Kevin Ashley / Author, Developer @Microsoft* [online]. 2014 [cit. 2015-04-17]. Dostupné z: <http://www.kevinashley.com/2014/12/creating-beautiful-cross-platform-apps-for-ios-android-and-windows-with-visual-studio-part-i-user-interface/>
- [13] Qt Download. *Qt / Cross-platform application & UI development framework* [online]. [cit. 2015-04-25]. Dostupné z: <https://www.qt.io/download/>
- [14] Minimal Requirements. 2014. The University of Tennessee, Knoxville [online]. [cit. 2015-04-25]. Dostupné z: <http://eagle.phys.utk.edu/guidry/android/minimumMachinesAndSoftware.html>
- [15] Android Lollipop. *Android Developers* [online]. [cit. 2015-04-25]. Dostupné z: <http://developer.android.com/about/versions/lollipop.html>
- [16] Download Android Studio and SDK Tools. *Android Developers* [online]. [cit. 2015-04-25]. Dostupné z: [https://developer.android.com/sdk/index.html#Requirements – HW](https://developer.android.com/sdk/index.html#Requirements-HW)
- [17] What Hardware Will You Need to Develop iOS Apps? 2014. *iOS Development Tutorials* [online]. [cit. 2015-04-25]. Dostupné z: <http://iosdevelopmenttutorials.com/hardware-needed-to-develop-ios-apps/>
- [18] Start Developing iOS Apps Today. 2015. *Apple Developer* [online]. [cit. 2015-04-25]. Dostupné z: <https://developer.apple.com/library/ios/referencelibrary/GettingStarted/RoadMapiOS/>
- [19] Preparing for Windows Phone development. 2015. *Windows Dev Center* [online]. [cit. 2015-04-25]. Dostupné z: [https://dev.windowsphone.com/en-us/oem?contentName=docs%2FGetting\\_Started%2FPreparing\\_for\\_Windows\\_Phone\\_development](https://dev.windowsphone.com/en-us/oem?contentName=docs%2FGetting_Started%2FPreparing_for_Windows_Phone_development)
- [20] How to deploy and run an app for Windows Phone 8. *MSDN – sieť vývojárov spoločnosti Microsoft* [online]. [cit. 2015-05-02]. Dostupné z: [https://msdn.microsoft.com/en-us/library/windows/apps/ff402565\(v=vs.105\).aspx](https://msdn.microsoft.com/en-us/library/windows/apps/ff402565(v=vs.105).aspx)
- [21] How To Install Apps On a Windows Phone 8 Device. 2013. *MakeUseOf* [online]. [cit. 2015-05-05]. Dostupné z: <http://www.makeuseof.com/tag/how-to-install-apps-on-a-windows-phone-8-device/>

- [22] 5+ Ways to Install Android Apps on Your Phone or Tablet. *How-To Geek* [online]. [cit. 2015-05-02]. Dostupné z: <http://www.howtogeek.com/161366/5-ways-to-install-android-apps-on-your-phone-or-tablet/>
- [23] How to Install Applications (APK Files) on Your Android Phone. *V-Braille* [online]. [cit. 2015-05-02]. Dostupné z: [http://vbraille.cs.washington.edu/doc/how\\_to\\_install\\_apks.pdf](http://vbraille.cs.washington.edu/doc/how_to_install_apks.pdf)
- [24] How to Deploy your App on an iPhone. 2014. How To Make an iPhone App [online]. [cit. 2015-05-02]. Dostupné z: <http://codewithchris.com/deploy-your-app-on-an-iphone/>
- [25] 3 Ways to Install an iPhone Application - wikiHow. *WikiHow - How to do anything* [online]. [cit. 2015-05-02]. Dostupné z: <http://www.wikihow.com/Install-an-iPhone-Application>
- [26] Ios - How to do Ad-Hoc deployment of ipa file to my iPhone. 2014. Stack Overflow [online]. [cit. 2015-05-02]. Dostupné z: <http://stackoverflow.com/questions/23821012/how-to-do-ad-hoc-deployment-of-ipa-file-to-my-iphone>
- [27] How to install Appx (\*.appxbundle) on Windows Phone? 2014. WindowsPhoneHub [online]. [cit. 2015-05-02]. Dostupné z: <http://windowsphonehub.in/tutorials/guide-install-appx-appxbundle-windows-phone/>
- [28] PhoneGap or Titanium or Xamarin - Which Cross-Platform Framework Should You Choose? 2015. Cygnet Infotech [online]. [cit. 2015-05-05]. Dostupné z: <http://www.cygnet-infotech.com/blog/phonegap-or-titanium-or-xamarin-which-cross-platform-should-you-choose>
- [29] PhoneGap. AppIndex - app development marketplace [online]. [cit. 2015-05-05]. Dostupné z: <http://appindex.com/app-development-tools/phonegap/>
- [30] Xamarin. AppIndex - app development marketplace [online]. [cit. 2015-05-05]. Dostupné z: <http://appindex.com/app-development-tools/xamarin/>

**ZOZNAM POUŽITÝCH SYMBOLOV A SKRATIEK**

API	Application Programming Interface
CSS	Cascading Style Sheets
GUI	Graphical User Interface
HTML	HyperText Markup Language
HW	Hardware.
IDE	Integrated Development Environment
MDSD	Model Driven Software Development
obr.	Obrázok
QML	Qt Metaobject Language
resp.	Respektíve
RTOS	Real-Time Operating System
SDK	Software development kit
SW	Software
tab.	Tabuľka
tzv.	Takzvané
UI	User Interface.
WP	Windows Phone
XML	Extensible Markup Language

**ZOZNAM OBRÁZKOV**

<i>Obr. 1. Natívny vzhľad na rôznych platformách [12]</i> .....	13
<i>Obr. 2. Apple Mac [17]</i> .....	19
<i>Obr. 3. Google Play [22]</i> .....	21
<i>Obr. 4. iOS App Store [25]</i> .....	21
<i>Obr. 5. WP Store [21]</i> .....	22
<i>Obr. 6. Povolenie neznámych zdrojov [22]</i> .....	23
<i>Obr. 7. Vytvorenie vývojárskeho a distribučného iOS certifikát [24]</i> .....	25
<i>Obr. 8. Výber WP zariadenia [20]</i> .....	25
<i>Obr. 9. Rôzne desktopové štýly widget aplikácii [8]</i> .....	29
<i>Obr. 10. Koncept aplikácie</i> .....	38
<i>Obr. 11. Qt Creator</i> .....	39
<i>Obr. 12. ComboBox, Qt</i> .....	41
<i>Obr. 13. Aplikácia Qt</i> .....	42
<i>Obr. 14. MessageBox, Qt</i> .....	44
<i>Obr. 15. Vytvorenie a zostavenie ukážkovej aplikácie PhoneGap</i> .....	45
<i>Obr. 16. Pôvodný a zmenený vizuálny štýl nástroja „select“, PhoneGap</i> .....	46
<i>Obr. 17. Aplikácia PhoneGap</i> .....	47
<i>Obr. 18. AlertDialog, PhoneGap</i> .....	48
<i>Obr. 19. Xamarin Studio</i> .....	50
<i>Obr. 20. Picker, Xamarin</i> .....	51
<i>Obr. 21. Aplikácia Xamarin</i> .....	52
<i>Obr. 22. DisplayAlert, Xamarin</i> .....	54

**ZOZNAM TABULIEK**

<i>Tab. 1. Vlastnosti metod multiplatformního vývoje [11]</i> .....	17
<i>Tab. 2. Podporované platformy Qt 5.4 [8]</i> .....	28
<i>Tab. 3. Licenční verze frameworku Qt a ich vlastnosti [13]</i> .....	33
<i>Tab. 4. Výsledné zhodnotenie</i> .....	56

**ZOZNAM PRÍLOH**

P I	Projekt Qt aplikácie	appQt.zip
P II	Projekt PhoneGap aplikácie	appPhoneGap.zip
P III	Projekt Xamarin aplikácie	appXamarin.zip

Text bakalárskej práce v PDF a všetky prílohy sú vložené na priloženom CD.