Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačních technologií

Bakalářská práce

Vývoj aplikace - Android versus iOS

Pavel Kural

Čestné prohlášení

Prohlašuji, že svou bakalářskou práci "Vývoj aplikace - Android versus iOS" jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu literatury na konci práce. Jako autor uvedené bakalářské práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 10. 3. 2015 _____

Poděkování

Rád bych touto cestou poděkoval vedoucímu a konzultantovi mé bakalářské práce za odborné rady a pomoc při jejím vypracování. Dále bych chtěl poděkovat své rodině a blízkým za podporu při psaní této práce.

Vývoj aplikace - Android versus iOS

Souhrn

Předkládaná bakalářská práce na téma „Vývoj aplikace - Android versus iOS“ pojednává o základních pojmech týkajících se platforem iOS a Android a jejich vývojových prostředí Xcode a Android Studio. V práci je podrobně rozebrána jejich stavba, základní struktura, vývoj, historie, zabezpečení a typický programovací jazyk.

Teoretická část práce vychází ze studijní literatury a slouží k proniknutí do problematiky vývoje aplikací.

V praktické části je pak vytvořena demoverze aplikace ve vývojovém prostředí Android studio pro operační systém Android a ve vývojovém prostředí Xcode pro operační systém iOS.

Na základě toho je provedeno porovnání jednotlivých prostředí pomocí Saatyho metody vícekriteriální analýzy.

Keywords: Android, iOS, Android Studio, Xcode, Programování, Mobilní aplikace.

Developing application - Android vs. iOS

Summary

The Bachelor thesis on the topic "Developing application – Android platform vs. iOS" platform is about platforms Android and iOS in general, it characterizes its elements, basic structure, evolution, security and the typical programming language.

The theoretical part of Bachelor thesis is based on a literature review and use to penetrate to the issue of developing application.

The practical part of thesis is about creating demo-version application in Android studio development studio for Android operating system and about the creating same one in Xcode development studio for iOS operating system.

Based on the practical part, Android Studio and Xcode will be compared by Saaty method.

Keywords: Android, iOS, Android Studio, Xcode, Programming, Mobile application.

Obsah

Seznam obrázků

Seznam tabulek

Seznam grafů

Seznam příloh

1 Úvod

Bez mobilního zařízení si současný život nelze ani představit. 70% lidí v České republice vlastní chytrý telefon nebo tablet. S každým prodaným zařízením tak roste počet lidí používajících nové technologie a lidí závislých na neustálém spojení se světem. Dnešní uživatelé chtějí mít přístup k informacím neustále. Na trh tak musí přicházet stále lepší aplikace, které uživatelům tyto informace zpřístupní, posunou možnosti sdílení či uživatele překvapí produktem zcela novým.

Mobilní aplikace nabízí na rozdíl od webových stránek velkou řadu výhod, především z hlediska použitelnosti. Aplikaci je možné spustit odkudkoliv a internetové připojení není vždy vyžadováno. Spuštění mobilní aplikace je velice snadné, její ovládání je pro uživatele komfortní. Kromě personifikace mobilní aplikace také nabízí přístup k hardwarovým funkcím telefonu, jako je: GPS, fotoaparát, akcelerometr, mikrofon apod. Přes mobilní aplikace je rovněž možné provádět řadu výpočtů, nebo například editovat multimediální soubory. Distribuce samotných mobilních aplikací je pro mnoho uživatelů jednoduše zprostředkovaná přes mobilní markety.

S vývojem nové aplikace úzce souvisí výběr platformy a vývojového prostředí, ve kterém bude programátor aplikaci vyvíjet. V současné době jsou nejvíce rozšířené platformy Android od společnosti Google Inc. a jeho vývojové prostředí Android Studio, které nabízí řadu užitečných funkcí pro vývoj. Jejím konkurentem je platforma iOS od společnosti Apple Inc. s jedním z nejpopulárnějších vývojových prostředí Xcode. Každá platforma má své výhody i nevýhody a své specifické vlastnosti, podle kterých se aplikace dále vyvíjejí.

2 Cíl a metodika práce

2.1 Cíl práce

Cílem bakalářské práce je v teoretické části charakterizovat základní pojmy týkající se platforem IOS a Android a jejich vývojových prostředí Xcode a Android Studio. V práci je podrobně rozebrána jejich stavba, základní struktura, vývoj, historie, zabezpečení a typický programovací jazyk.

Hlavním cílem praktické části je porovnat vývoj autorem vytvořené aplikace ve vývojovém prostředí Android Studio a Xcode pomocí vícekriteriální analýzy variant Saatyho metodou.

2.2 Metodika práce

Teoretická část práce bude založena na studiu odborné literatury, porovnávání internetových zdrojů (vývojových prostředí a programovacích jazyků) pro tvorbu mobilních aplikací.

Praktická část se bude zabývat samotnou tvorbou aplikace ve vývojovém prostředí Android Studio a Xcode. Prostřednictvím vytvořené aplikace budou demonstrovány základní principy návrhu aplikací. Závěrem budou syntetizovány výhody a nevýhody vývoje aplikací pro platformu Android a platformu iOS a provedeno zhodnocení pomocí Saatyho metody vícekriteriální analýzy variant.

3 Teoretická východiska

3.1 Chytré telefony

Co z chytrého telefonu dělá telefon chytrý a z hloupého telefon „hloupý“? Definovat jednotnou a ucelenou definici pro chytré telefony není úplně jednoduché, ve skutečnosti žádná správná definice neexistuje. Jeden z mnoha zdrojů definuje chytrý telefon takto:

„A category of mobile device that provides advanced capabilities beyond a typical mobile phone. Smartphones run complete operating system software that provides a standardized interface and platform for application developers,“ (Phonescoop, 2012).

Hlavním rozdílem je otevřený operační systém, kterým chytré telefony disponují. Klasické mobilní telefony používají tzv. proprietární software, který uživateli neumožňuje instalovat libovolné aplikace jako na telefony chytré. Další rozdíly spočívají samozřejmě v hardwarové vybavenosti. Chytré telefony v dnešní době disponují například kvalitním fotoaparátem, kamerou s vysokým rozlišením, velkým dotykovým displejem, GPS[1], Bluetooth[2], Wi-fi[3]a jinými prvky. Chytrý telefon dále umožňuje přístup na internet, posílat e-mailové zprávy prostřednictvím e-mailového klienta a poskytuje přístup na sociální sítě. Dalo by se říct, že se smartphony začínají přibližovat zařízením, jako jsou například notebooky[4] či tablety[5].

Tak jako každý výrobek i smartphony mají své nevýhody. Oproti svým „hloupým“ předchůdcům mají menší výdrž baterie. Při spouštění náročnějších aplikací se výdrž baterie dramaticky sníží. Další možnou nevýhodou se jeví složitost samotných systémů.

[1] Globální polohovací systém

[2] Technologie poskytující bezdrátový přenos dat

[3] Technologie sloužící k bezdrátové komunikaci v sítích

[4] Přenosný počítač

[5] Přenosný počítač s dotykovou obrazovkou

3.1.1 Vývoj chytrých telefonů

Tabulka 1 - Vývoj chytrých telefonů

Firma	Typ	Popis	Rok
Motorola	DynaTAC	První mobilní telefon, nabízel 30minut hovoru	1983
IBM	Simon	První smartphone, disponoval vlastním OS ROM-DOS, dotykovým displejem a základními aplikacemi	1994
Nokia	9000 Communicator	Disponoval QWERTY klávesnicí, procesorem Intel 80386, displejem 640 x 200 pixelů a OS GEOS	1996
Mitsubishi	Trium Mondo	Vlastnil operačním systémem Pocket PC	2000
Samsung	SPH-I500	První smartphone s PalmOS, disponoval kamerou s rozlišením 640x480 pixelů a barevným displejem	2002
Palm	Treo 750v	První model s operačním systémem Windows Mobile	2006
Apple	Iphone 1G	První smartphone z řad generace Iphonu od společnosti Apple	2007
HTC	Dream	První smartphone s operačním systémem Android	2008
Samsung	Galaxy S	První smartphone z řad generace Galaxy S od firmy Samsung	2010
Nokia	Lumia 800	První smartphone běžící na OS Windows 7.5	2011
Sony	Xperia S	První smartphone nazvaný pouze značkou Sony	2012
Huawei	Honor 6	Smartphone nové řady Honor od firmy Huawei	2014
Apple	Iphone 6S	Zatím poslední generace z řady Iphonů	2015

Zdroj: upraveno dle (Smrček, 2011), (Zandl, 1997), (Strauss, 2010), (Kovařík, 2012).

3.2 Mobilní operační systémy

Operační systém je základní softwarové vybavení každého chytrého mobilního zařízení. Slouží k tomu, aby uživatel mohl zařízení ovládat, instalovat aplikace a spravovat hardware. OS[6] je zaveden do paměti při jeho startu a zůstává v činnosti až do jeho vypnutí. Jinými slovy operační systém je to, co dělá z normální mobilního telefonu telefon chytrý. Mobilní operační systémy kombinují funkce osobního počítače a funkce mobilního telefonu. V dnešní době jsou ovládány pomocí dotykových displejů a mají k dispozici funkce jako Bluetooth, Wi-fi, GPS, fotoaparát a podobně. Mezi dnes nejpopulárnější mobilní operační systémy patří Android od firmy Google a iOS od firmy Apple (Tanenbaum, 2008).

Tabulka 2 - Používané operační systémy pro chytré telefony

Období	Android	iOS	Windows p.	Blackberry	Další
2015	82,8%	13,9%	2,6%	0,3%	0,4%
2014	84,8%	11,6%	2,5%	0,5%	0,7%
2013	79,8%	12,9%	3,4%	2,8%	1,2%
2012	69,3%	16,6%	3,1%	4,9%	6,1%

Zdroj: upraveno dle (IDC, 2015).

[6] Operating system – Operační systém

3.2.1 Android

Android je poměrně mladý operační systém, ovšem navzdory jeho stáří je velmi úspěšný a populární. Tento mobilní operační systém byl vyroben společností Google Inc. Jedná se o plně otevřený systém postavený na monolitickém jádru operačního systému Linux a byl naprogramován v jazycích C, C+ a Java. Android se vyskytuje v mobilních zařízeních, tabletech a notebookách. V současnosti se jedná o nejvíce rozšířený operační systém, který používá 83% uživatelů (Ujbányai, 2012).

3.2.1.1 Historie a vývoj

Společnost Android Inc. byla založena v říjnu roku 2003 v Kalifornii čtyřmi zakladateli. Byli to Rich Miner, Andy Rubin, Nick Sears a Chris White. Jako začínající firma byla Android Inc. koupena v roce 2005 společností Google Inc. a stala se tak její dceřinou společností. Google v tu dobu začal vyvíjet nový operační systém v čele s Andy Rubinem a chystal se vstoupit na trh chytrých mobilních telefonů. V roce 2007 společnost získala potřebné patenty v oblasti mobilních technologií. Ve stejném roce 5. listopadu, vzniklo uskupení OHA[7] dnes zahrnující 90 společností zabývajících se výrobou mobilních telefonů, čipů a mobilních aplikací. Cílem tohoto uskupení, zahrnujícího kromě firmy Google Inc. například velké firmy jako jsou HTC, Intel, LG, Samsung a mnoho dalších, bylo vyvinout otevřený standard pro mobilní zařízení. Ve stejný den Google předal vývoj nové platformy do rukou OHA a zároveň byl ohlášen produkt Android, otevřená mobilní platforma postavená na linuxovém jádře. Android je systém navržený tak, aby byl použitelný pro co možná nejvíc zařízení, jako jsou tablety, chytré hodinky a menší notebooky (Ujbányai, 2012).

[7] Open Handset Alliance

3.2.1.2 API

API neboli Application Programming Interface je rozhraní pro programování aplikací. Jde o sbírku procedur, funkcí, tříd nebo protokolů nějaké knihovny. Toto rozhraní má několik úrovní. Jednotlivé verze operačního systému Android podporují jednu úroveň API. Vývojář tak musí určit pro jakou minimální úroveň API (a tedy verzi operačního systému) bude aplikaci vytvářet (Android, 2016).

Tabulka 3 - Verze API

Verze Androidu	API úroveň	Verze kódu	Rok
6.0	23	Marshmallow	2015
5.1	22	LOLLIPOP_MR1	2014
5.0	21	LOLLIPOP	2014
4.4	19	KITKAT	2013
4.3	18	JELLY_BEAN_MR2	2013
4.2 / 4.2.2	17	JELLY_BEAN_MR1	2012
4.1 / 4.1.1	16	JELLY_BEAN	2012
4.0.3 / 4.0.4	15	ICE_CREAM_SANDWICH_MR1	2012
4.0 / 4.0.1 / 4.0.2	14	ICE_CREAM_SANDWICH	2011
3.2	13	HONEYCOMB_MR2	2011
3.1.x	12	HONEYCOMB_MR1	2011
3.0.x	11	HONEYCOMB	2011
2.3.4 / 2.3.3	10	GINGERBREAD_MR1	2011
2.3.3 / 2.3.2 / 2.3.1	9	GINGERBREAD	2011
2.2.x	8	FROYO	2011
2.1.x	7	ECLAIR_MR1	2010
2.0.1	6	ECLAIR_0_1	2009
2.0	5	ECLAIR	2009
1.6	4	DONUT	2009
1.5	3	CUPCAKE	2009
1.1	2	BASE_1_1	2009
1.0	1	BASE	2008

Zdroj: upraveno dle (Android, 2016).

3.2.1.3 Přehled verzí

Zajímavostí Androidu je, že Google jednotlivé verze pojmenovává podle cukroví seřazených podle abecedy, můžeme se tedy setkat s verzí například „Donut“. Nejstarší verzí je Android 1.0 a nejaktuálnější verzí je Android 6 Marshmallow (Android, 2015) (Elitecsoftware, 2012).

3.2.1.3.1 Android 1.0

Úplně první verzí android, byla verze Android 1.0, která byla světu představena 23. září 2008 v USA. V České republice byla představena až v lednu roku 2009. Velmi důležitou událostí, zejména pro vývojáře, bylo poskytnutí SDK13 1.0[8] pod licencí opensource[9]. Jeho zdrojový kód tak mohl využívat naprosto každý. Navíc 22. října 2009 byl zprovozněn první Android Market[10] (dnes již Google Play) s více než 30 aplikacemi (Smrček, 2011).

3.2.1.3.2 Android 1.1

Jedná se o první upgrade systému Android, která přichází necelý půlrok po vydání verze 1.0. Aktualizace neobsahovala zásadní změny, spíše šlo výhradně o vyladění chyb a nedostatků předešlé verze 1.0 (Android, 2015).

3.2.1.3.3 Android 1.5 - Cupcake

Prvním rozsáhlejším upgradem byl právě Android 1.5 nazvaný Cupcake, který vyšel v dubnu roku 2009. Tento update založený na linuxovém jádře 2.6.27 přinesl několik významných změn. Velkou novinkou byla softwarová QWERTY[11] klávesnice, takže telefon již nepotřeboval hardwarovou klávesnici. Dalšími novinkami byla podpora widgetů[12], možnost nahráváni videí přímo na Youtube[13] a fotek na Picasu[14] (Elitecsoftware, 2012).

---

[8] Sada vývojových nástrojů

[9] Volně šiřitelná licence

[10] Obchod s aplikacemi

[11] Klávesnice s rozložením kláves v posloupnosti Q, W, E, R, T, Y

[12] Miniaplikace umístěná na hlavní ploše

[13] Webová stránka sloužící pro nahrávání videí

[14] Webová stránka sloužící pro nahrávání fotografií

3.2.1.3.4 Android 1.6 - Donut

Poslední verzí zakončující první řadu byla verze 1.6 nazvaná Donut. Byla uvolněna v září 2009 a přinesla následující změny: Aktualizaci tehdejšího Android marketu, nové rozhraní fotoaparátu, videokamery a galerie. Přibyla pokročilá funkce vyhledávání pomocí hlasu, dále pak podpora WVGA[15] a technologií CDMA/EV-DO[16], 802.1x[17], VPN[18] (Smrček, 2011).

3.2.1.3.5 Android 2.0 / 2.1 - Eclair

Update s názvem Eclair byl uvolněn v říjnu roku 2009. Nově byla přidána podpora Bluetooth verze 2.1, byla provedena optimalizace rychlosti hardware a poprvé přichází podpora HTML5[19]. Vylepšení se dočkala i softwarová klávesnice a byla zde po prvé přidána navigace Google maps (Android, 2015) (Elitecsoftware, 2012).

3.2.1.3.6 Android 2.2 - Froyo

Verze Froyo, která měla hned několik verzí, vyšla v květnu roku 2010. Novinkou byla dnes již opět neexistující podpora Adobe Flashe[20], možnost instalovat aplikace na paměťovou kartu, vylepšená správa paměti RAM[21] a možnost vytvoření z mobilního přístroje Wi-Fi hotspot[22]. Hlavním přínosem této verze je zlepšení výkonu procesoru a přidání Javascriptu[23] do základního prohlížeče (Android, 2015).

---

[15] Širokoúhlé rozlišení

[16] Rodina 3G standardů

[17] Protokol umožňující přístup do počítačové sítě

[18] Virtuální privátní síť

[19] Verze značkovacího jazyka sloužícího pro tvorbu webových stránek

[20] Software určený k prohlížení animací, filmů apod.

[21] Operační paměť

[22] Přístupový bod ke kterému se klienti připojují

[23] Skriptovací jazyk

3.2.1.3.7 Android 2.3 / 2.4 - Gingerbread

Tento update vznikl v prosinci roku 2010. Novinkou zde byla podpora 3D zobrazování v Google mapách a podpora SIP protokolu pro internetovou telefonii. Dále pak přidání strictmode[24] ladění, knihovny pro zvukové efekty, nahrazení OpenCoru Media frameworkem[25], přidání NDK[26], přidání kodeku videa VP8 a kodeku zvuku AMR. Byl přidán také formát videa WebM[27] a formát zvuku AAC[28] (Smrček, 2011).

3.2.1.3.8 Android 3.0 / 3.1 / 3.2 - Honeycomb

Verze Honeycomb vyšla v únoru 2011 a byla určena výhradně pro tablety. Hlavní změnou je podpora velkých obrazovek, dále pak podpora vícejádrových procesorů a nové uživatelské rozhraní. Z vývojářského hlediska přinesla následující změny: přidání hardwarové akcelerace 2D a 3D grafiky, kodeku zvuku FLAC, přehled síťové komunikace, kontextuální tlačítko, DRM[29] Framework, RTP streaming [30]API, API pro administraci zařízení. Dále zde byly po prvé představeny fragmenty[31] (Android, 2015), (Kilián, 2015).

3.2.1.3.9 Android 4.0 - Ice Cream Sandwich

Android 4.0 byl představen v říjnu 2011. Mezi hlavní novinky patří nové bezpečnostní funkce, funkce rozpoznávání obličeje, sdílení informací pomocí NFC a také kontrolní utilita pro mobilní data. Vývojářům byl přidán Grid[32] layout, služba pro kontrolování psaní kódu, VPN klient API, ASLR[33], služba pro zablokování kamery, ZSL[34]a přidání Flags za účelem lepší kontroly uživatelského rozhraní (Android, 2015), (Elitecsoftware, 2012).

---

[24] Vývojářská pomůcka pro lepší psaní kódu

[25] Softwarová struktura pro podporu médií

[26] Native Development Kit

[27] Video formát umožňující kompresi HTML5 videem

[28] Formát pro ztrátovou kompresi zvuku

[29] Technické metody pro kontrolu digitálních médií

[30] Sítový protokol pro odesílání audia a videa přes internetovou síť

[31] Reprezentují vlastnost nebo rozhraní uživatele

[32] Obdélníková mřížka umožňující vnořit další objekty

[33] Metoda počítačové bezpečnosti

[34] Metoda, která redukuje prodlevu mezí pořízením fotky

3.2.1.3.10 Android 4.1/4.2/4.3 - Jelly Bean

Android pokračuje v červenci 2012 verzemi 4.1 až 4.3 Jelly Bean. Hlavní novinkou bylo přepínání mezi účty a služba Google now. Z vývojářského hlediska to pak bylo přidání vertikální synchronizace[35], triple bufferingu[36], snížení odezvy doteku obrazovky, API pro skenování Wi-Fi, podpora RTL[37], podpora externího displeje, podpora OpenGL ES [38]3.0, přidání vnořených fragmentů, možnost běhu aplikace na grafickém čipu, větší možnost nastavení notifikací včetně priorit a dále byla přidána podpora Bluetooth low energy, umožňující vytvoření aplikací jako je monitor srdečního tepu (Android, 2015).

3.2.1.3.11 Android 4.4 - Kitkat

Tato verze vyšla v říjnu roku 2013. V této verzi byl přidán Framework pro tisk dokumentů. Aplikace dostaly povolení od telefonních poskytovatelů číst a psát SMS a MMS a byla přidána podpora pro bezdrátové platby pomocí NFC. Bylo vylepšeno zabezpečení aplikací, přidány nástroje pro analýzu použité paměti a monitorování zvuku a přidání klienta dálkového ovládání, umožňujícího vytvoření vlastního dálkového ovládání. Dále pak byla přidána podpora video formátu WebVTT, přidání frameworku SAF pro lepší přístup k datům a vylepšení celoobrazovkového modu (Kilián, 2015).

3.2.1.3.12 Android 5.0/5.1 - Lollipop

V listopadu 2014 přichází další verze s názvem Lollipop. Došlo k změně běhového prostředí aplikací Dalvik na prostředí ART[39], což přineslo změny jako způsob, jakým byla spravována paměť, rychlost otvírání aplikací a vylepšené bylo i ladění aplikací. Lollipop sebou přinesl velkou vizuální změnu a to design s názvem materiál. Google v této verzi pobízí vývojáře ke změně RemoteControlClient třídy na novou MediaSession API a to hlavně z důvodu podpory nových změn. Další změnou je podpora 64bitových systémů a došlo k vylepšení notifikací (Kilián, 2015).

---

[35] Technologie zajišťující plynulý obraz

[36] Technologie umožňující zrychlení snímkové frekvence za použití více než jedné vyrovnávací paměti

[37] Text zprava do leva

[38] Rozhraní pro vykreslování 2D a 3D grafiky

[39] Android Runtime, prostředí běhu aplikací

3.2.1.3.13 Android 6.0 - Marshmallow

Tato verze vyšla v říjnu 2015. Bylo zde přidáno Fingerprint[40] API a dále také možnost připojení přes USB[41] Typu C. Aplikace se nyní automaticky zálohují a je možné po jejich spuštění ověřovat uživatele podle toho, jak bylo v poslední době odemčeno mobilní zařízení. Další novinky jsou: DirectShare[42], API pro zvukovou interakci, Assist[43] API, Adoptable Storage Devices[44], Bluetooth stylus[45], 4K[46] rozlišení a další. Mobilnímu zařízení by navíc měla vydržet baterie až dvakrát déle (Android, 2015).

Obrázek 1 - Obrazovka Marshmallow

Zdroj: (Wikipedia, 2015).

[40] Rozpoznávání otisku prstu

[41] Univerzální sériová sběrnice

[42] Funkce umožňující rychlejší sdílení dat

[43] Knihovna přinášející lepší nápovědu pro aplikace

[44] Funkce, která umožnuje aby se externí paměťové zařízení chovalo jako interní

[45] Umožnuje připojení uživatelů pomocí bluetooth a následnou komunikaci pomocí stylusu

[46] Rozlišení obrazu 4096 × 3112

3.2.1.4 Programovací jazyk Java

Java je objektově orientovaný programovací jazyk společnosti Sun Microsystems, která má sídlo v Americe. Tento jazyk byl představen 23. května 1995 a je jedním z nejpoužívanějších programovacích jazyků na světě. Jedná se o nejmodernější a současně také nejrozšířenější podobu programovacího jazyka, který využívá virtuální stroj. Díky své portabilitě je používán pro programy pracující na různých platformách.

V dnešní době se setkáme s Javou skoro všude: u platebních terminálů, aplikací na ovládání sportovních tabulí až po aplikaci na našich televizních přijímačích.

Firma Google zvolila Javu jako pilíř při tvorbě svého operačního systému Android. I přesto, že byl zvolen jazyk Java, velká část jádra systému Android je napsána v jazyce C. Java je používána pro Android hlavně kvůli syntaxi nikoliv pro své knihovny, protože Android používá své vlastní knihovny. Programovací jazyk Java byl ovlivněn jazyky C, C++ a Smalltalk (Eckel, 2000).

3.2.1.4.1 Virtuální stroj

Virtuální stroj funguje následujícím způsobem: zdrojový kód je nejprve kompilátorem přeložen do mezikódu, také nazývaným bytecodem. Tento mezikód je pak, díky jeho struktuře, velmi rychle přeložen interpretem, tedy virtuálním strojem. V případě Javy se jedná o JVM[47]. Jako výstup z virtuálního stroje je strojový kód určený pro procesor.

Tato metoda tak eliminuje nevýhody interpreta a kompileru. Výhodou této metody jsou: včasné odhalení chyb ve zdrojovém kódu, stabilita, jednoduchý vývoj, rychlost, malá zranitelnost a portabilita (Eckel, 2000).

[47] Java Virtual Machine – Virtuální stroj programovacího jazyku Java

3.2.1.4.2 Dalvik vs. ART

Dalvik je název virtuálního stroje, který v systému Android vytváří běhové prostředí aplikace. Ve verzi systému 4.4 „KitKat“ byla uvedena ukázka nástupce běhového prostředí Dalvik a sice běhové prostředí ART. Běhové prostředí ART později ve verzi 5.0 Dalvik zcela nahradilo a stalo se tak výchozím virtuálním strojem pro běh aplikací.

Dalvik na rozdíl od ART funguje jako JIT[48] kompilátor. To znamená, že nejdřív vezme bytecode, který následně musí přeložit do strojového kódu, a pak spustit. Tato procedura je poměrně pomalá a proto Android přešel na běhové prostředí ART. ART používá AOT[49] kompilátor a díky tomu bytecode překládá na strojový kód hned při instalaci, tudíž jej potom jen spustí. Výhodou je menší náročnost, lepší multitasking a celkově větší plynulost prostředí. Nevýhodou je, že instalace trvá déle, právě kvůli kompilování už při instalaci, a také větší náročnost na paměť.

3.2.1.4.3 JDK

Java Development Kit neboli sada základních nástrojů pro vývoj aplikací v Javě. Jedná se o sadu, který je produktem firmy Oracle Corporation založené v roce 1977. JDK představuje rozšířenou verzi SDK. První verzí JDK byla verze 1.0 a její nejnovější verzí je zatím Java SE 8.0.

Součástí Java Development Kitu je JRE[50], sloužící pro spuštění aplikací i vývojových nástrojů, pod které patří již zmíněný virtuální stroj. Další součásti této sady jsou: zavaděč pro aplikace Java, překladač zdrojového kódu do bytecodu, nástroj pro ladění programu, nástroj pro generování programové dokumentace a další (Oracle, 2016).

[48] Just in time

[49] Ahead of time

[50] Java Runtime Environment

3.2.1.4.4 Ukázka kódu

Syntaxe Javy je do značné míry odvozena od programovacího jazyka C++. Na rozdíl od C++, byla Java postavena především jako objektově orientovaný jazyk. Veškerý kód je psán uvnitř tříd a všechno je objekt s výjimkou primitivních datových typů. Mezi primitivní datové typy jazyka Java patří byte, short, int, long, float, double, boolean a char. Dále Java nepodporuje přetěžování operátorů nebo vícenásobnou dědičnost pro třídy, což zjednodušuje celý jazyk. Při psaní zdrojového kódu je třeba si uvědomit, že programovací jazyk Java je stejně jako C++, case-sensisitive, což pro programátora znamená, že musí dodržovat malá a velká písmena (Eckel, 2000).

Obrázek 2 - Ukázka kódu Javy

```java
public class HelloWorld {
    public static void main(String[] args) {
        Pozdrav pozdrav = new Pozdrav("Ahoj světe!");
        pozdrav.print();
    }
}

class Pozdrav {
    private String text;

    public Pozdrav(String pozdrav) {
        text = pozdrav;
    }

    private String getText() {
        return text;
    }

    public void print() {
        System.out.println(getText());
    }
}
```

Zdroj: vlastní zpracování.

3.2.1.5 Android studio

Android studio je vývojové prostředí od firmy Google, které bylo představeno 16. května roku 2013. Vývojové studio bylo založeno na komerčním vývojovém prostředí IntelliJ IDEA a je určené výhradně pro rozvoj Androidu. Android Studio je volně stažitelné a přístupné pro operační systémy Windows, Mac a Linux (Android, 2016).

3.2.1.5.1 Funkce

Android studio nabízí propracovaný editor s možností navrhovat uživatelské rozhraní pomocí metody drag and drop[51], funkce pro testování kompatibility, nástroje pro zachycení výkonu, integrované nástroje pro profilování, šablony usnadňující tvorbu aplikací s běžným designem, podepisování aplikací a další (Android, 2016).

3.2.1.5.2 Životní cyklus aktivity

Aktivitu si lze představit jako základní komponentu pro vykreslení grafického návrhu aplikace. Android, jakožto OS primárně sám zodpovídá za activity a vše co s nimi souvisí. Aktivity se nacházejí v jednom ze tří stavů:

- Aktivní: aktivita běží (je v popředí), komunikuje s uživatelem, případně čeká na vstup uživatele.

- Pozastavená: jiná aktivita běží, pozastavená aktivita je stále viditelná pro uživatele. Aktivita v popředí například není zobrazena přes celý displej. Pozastavená aktivita je stále v operační paměti, ale pokud by měl systém velký nedostatek zdrojů, může ji vypnout.

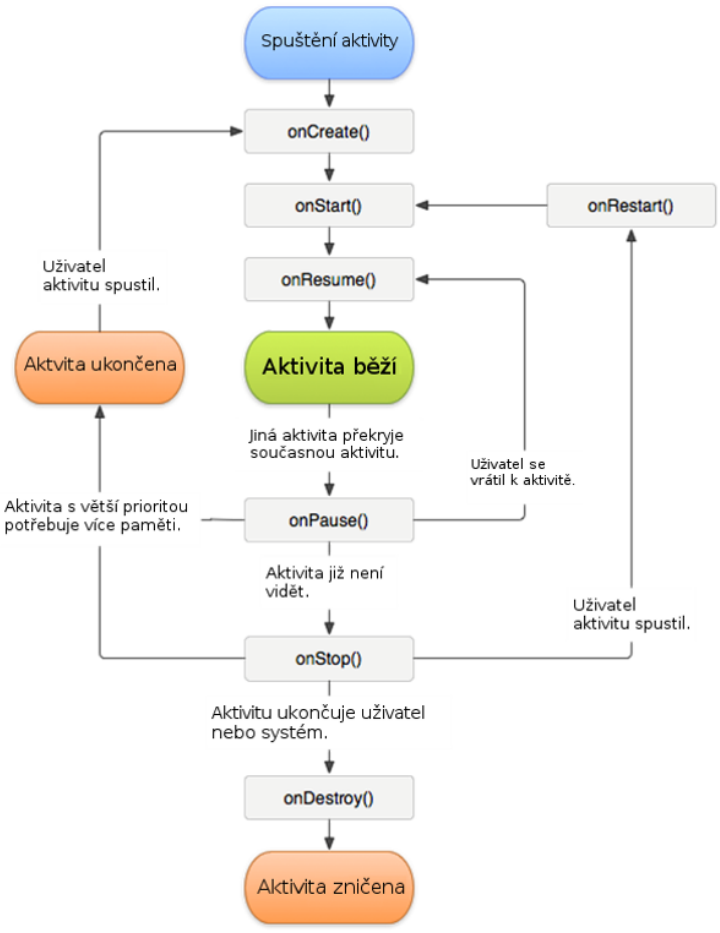
- Zastavená: aktivita je zcela v pozadí, není viditelná pro uživatele. Aktivita je v paměti, ale systém ji může kdykoliv vypnout, pokud potřebuje zdroje pro jinou činnost (Hlavík, 2015).

[51] Metoda programování, kdy se změny ve vizuální stránce projektu projeví i ve stránce kódové

Metody životního cyklu:

- onCreate() – Android v této metodě definuje vše potřebné, aby aktivita mohla běžet, například je zde definováno grafické rozhraní aplikace;
- onStart() – Metoda se volá, pokud aktivita byla poprvé spuštěna;
- onResume() – Volá se těsně předtím, než je aplikace posunuta do popředí;
- onPause() – Volá se před přechodem aplikace do pozadí;
- onStop() – Metoda volaná před zastavením aplikace;
- onDestroy() – Metoda volaná před zrušením aktivity (Hlavík, 2015).

Obrázek 3 - Životní cyklus aktivit



Zdroj: (Hlavík, 2015).

3.2.1.5.3 Minimální požadavky

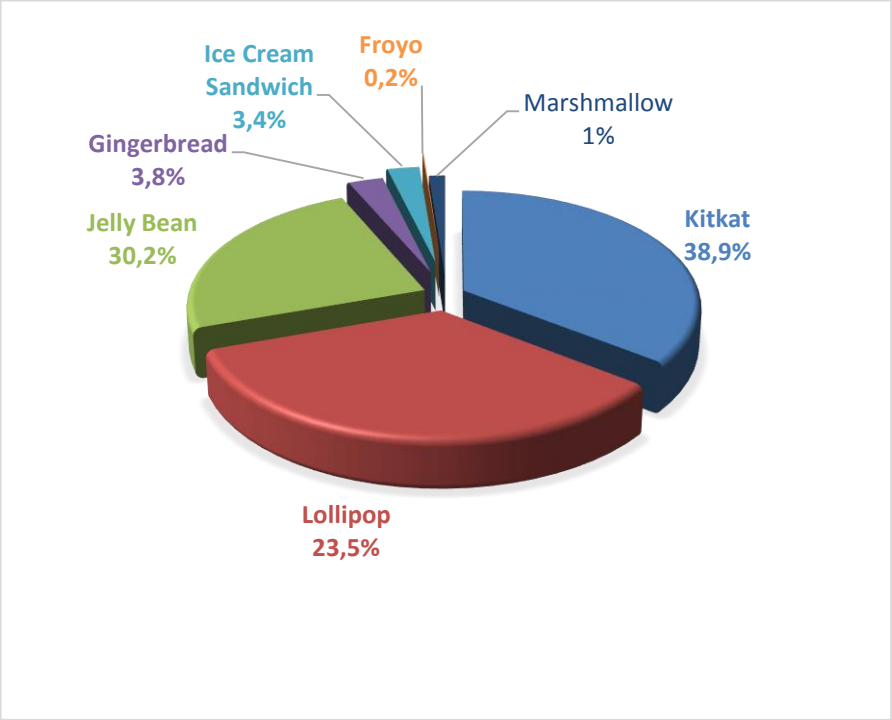
- Windows
    - Operační systém Windows 8/7/Vista;
    - 2GB[52] RAM;
    - 1,4 GB volného místa na pevném disku;
    - 1280x800 rozlišení;
    - JDK 7;
- Mac OS X
    - Operační systém Mac OS X 10.8.5 a vyšší;
    - 2GB RAM;
    - 1,4 GB volného místa na pevném disku;
    - 1280x800 rozlišení;
    - JDK 7;
    - JRE 6;
- Linux
    - Prostředí GNOME nebo KDE desktop;
    - GNU C knihovna 2.15 a novější;
    - 2GB RAM;
    - 1,4 GB volného místa na pevném disku;
    - 1280x800 rozlišení;
    - JDK 7 (Android, 2016).

[52] Gigabyte = 1000 MB

3.2.1.6 Shrnutí

Ačkoliv mobilní operační systém android je poměrně mladý, za pár let svého působení za sebou dokázal zanechat všechny konkurenty. Jeho velkou předností je otevřenost systému, díky které existuje tolik aplikací od třetích stran. Tyto aplikace jsou navíc většinou volně ke stažení. Tato velká výhoda se jeví v současné době jako velkou nevýhodou v podobě jeho bezpečnosti. Díky otevřenosti existuje i mnoho škodlivých aplikací, které mohou napáchat v systému velké škody. K rozšířenosti systému přispívá fakt, že je tento systém navržen tak, aby byl kompatibilní s co možná největším počtem zařízení, a tak se systém dostává i do cenově dostupných kategorií. Další výhodou je provázanost a synchronizace se společností Google Inc. a s jejími produkty, jako jsou například Google Maps, Google Apps či sociální síť Google+. Propojení s aplikacemi Google Apps poskytuje neomezený přístup k e-mailovému klientovi, dokumentům, kalendáři a kontaktům.

Graf 1 - Statistika používaných verzí Android



Zdroj: upraveno dle (Android, 2016).

3.2.2 iOS

Dalším populárním mobilním operačním systémem je atraktivní systém iOS, který byl vyvinut firmou Apple Inc. Jedná se o uzavřený systém postavený na hybridním jádru operačního systému Unix, byl naprogramovaný v jazycích C, C++, Objective C (Isaacson, 2014).

3.2.2.1 Historie a vývoj

Firma Apple vznikla v roce 1976 ve městě Cupertin v Kalifornii, přesněji v garáži rodičů Steva Jobse. Dnes je tato oblast známá jako Silicon Valley a považuje se za „meku" výpočetní technologie. Steve Jobs, Ronald Gerald Wayne a Steve Wozniak byli tři zakladatelé tehdejší firmy Apple Computers. První zařízení firmy Apple bylo vyrobeno ručně a pouze sto kusů se jmenovalo Apple I. Toto zařízení zaujalo mnoho lidí, některými je považováno za první osobní počítač (Isaacson, 2014). V této době o něm Wozniak řekl:

„Zmáčkl jsem několik písmen na klávesnici a byl jsem v šoku! Písmena se zobrazila na obrazovce." (Isaacson, 2014 str. 196)

Dalším model nazvaný Apple II byl představen v dubnu 1977. Jednalo se o první mikropočítač, který se začal úspěšně prodávat firemním zákazníkům a školám. Průlom přišel v 80. letech, přesněji roku 1984, s představením počítače zvaného Macintosh, zkráceně Mac. Toto značení používá firma pro své PC[53] a notebooky dodnes. V roce 2007 se firma přejmenovala z Apple Computers na Apple Inc. Důvodem pro změnu bylo to, že její portfolio již netvoří pouze osobní počítače, ale také multimediální služby a produkty (Isaacson, 2014).

Ve stejném roce Apple Inc. přichází s novinkou s prvním iPhonem, který byl představen v lednu roku 2007 a prodej byl zahájen o pár měsíců později. První verze operačního systému byla verze 1.0. Tehdy ještě systém odvozený od Mac OS X (operační systém pro PC), neměl žádný název. Od verze 2.0 se uchytil název iPhone OS, později v roce 2010 se ustálil název iOS, který je používán dodnes (Williams, 2015).

[53] Personal Computer – osobní počítač

3.2.2.2 Přehled verzí

První verzí byl iOS 1. X určený pro iPhone, iPad[54] a iPod[55]. Později se tento operační systém začal používat i u jiných zařízeních od firmy Apple, jako například u iPod Touch [56]a Apple TV[57]. Nejnovější verzí je dnes iOS 9. Apple podobně jako Google, uvolňuje nové SDK pro každou verzi iOS (Isaacson, 2014).

3.2.2.2.1 OS 1. X

Původní verze systému s názvem OS 1. X, která přišla s první verzí iPhone, byla ještě hodně nedoladěná a obsahovala mnoho chyb. Postupem času se začaly objevovat aktualizace, které tyto chyby opravovaly. Koncovým zařízením verze OS 1. X nebyl jen iPhone, ale také iPod a iPod Touch. Konečný update této série měl označení 1.1.5 (Theverge, 2013).

3.2.2.2.2 iPhone OS 2. X

Následovala verze 2. X, dostupná s příchodem iPhone 3G. Novinkou se stal mobilní market nazvaný Apple Store, který umožňuje nahrávat aplikace od třetích stran. Byly opraveny chyby, které ztěžovaly plynulost systému. SDK 2.0 přidalo do vývojového prostředí Xcode Interface Builder, který umožňuje vytvoření grafického uživatelského rozhraní pro aplikace, dále byla přidána podpora OpenGL 3D grafiky (Theverge, 2013).

3.2.2.2.3 iPhone OS 3. X

S vydáním iPhonu 3GS v roce 2009 přichází verze systému iOS 3. X. Tento upgrade ovšem neposkytoval plnou podporu všech funkcí majitelům s prvním iPhonem. Verze poskytuje mnoho vylepšení, jako například nový navigační systém, aktualizované mapy, možnost přihlášení se do účtu Youtube. Podpora HTML5 pro internetový prohlížeč Safari, rozšířené byly také podporované jazyky. Pro modely iPhone a iPod končí upgrade verzí 3.1.3. Verze 3.2 byla určená výhradně pro nové zařízení iPad (Theverge, 2013), (Costello, 2015).

---

[54] Tablet od firmy Apple

[55] Hudební přehrávač firmy Apple

[56] Hudební dotykový přehrávač

[57] Televize od firmy Apple

3.2.2.2.4 iOS 4. X

Jednalo se o první verzi přejmenovanou na „iOS“. Novinkou zde byl například FaceTime chat a poprvé přidání složek na hlavní obrazovku. Tato verze ukončuje podporu první verze Iphonu a iPodu Touch 1G. Plnou podporu všech funkcí pak tato verze neposkytuje iPhonu 3G a iPhonu Touch 2G. Ostatní zařízení využívají aktualizace v plné míře až do verze 4.3.5 (Costello, 2015).

3.2.2.2.5 iOS 5. X

Tento update již nepodporuje iPhone 3G a iPod Touch 2G. Verze byla vydána 12. října 2011 s nejnovějším modelem iPhone 4S. Poslední verze 5.1.1 je dostupná od 7. května 2012 a vychází společně s iPhone 4 GSM. Úplnou novinkou je služba iCloud, která ukládá fotografie, video, dokumenty a umožňuje tak bezdrátovou synchronizaci mezi všemi zařízeními (Costello, 2015).

3.2.2.2.6 iOS 6. X

iOS 6 byl představen v červu 2012 na již tradiční konferenci a byl vydán na podzim roku 2012. Po vzoru předchozích verzí iOS přestaly být některé starší přístroje podporovány, konkrétně to byl iPod Touch 3G a iPad 1G. Apple v této verzi řekl sbohem Google mapám a vytvořil mapy své. Další novinkou zde byla integrace populární sociální sítě Facebook a odstranění integrované aplikace YouTube. Novinkou je zde aplikace Passbook, která umožnuje na jednom místě uchovávat palubní lístky, lístky do kiny, dárkové poukazy apod. (Theverge, 2013).

3.2.2.2.7 iOS 7. X

Apple představil iOS 7 na své výroční konferenci roku 2013. Jedná se o největší změnu systému od jeho uvedení v roce 2007. Designová stránka systém prošla kompletní rekonstrukcí. Přidáno bylo rychlé ovládací centrum obsahující rychlé přepínání Wi-Fi, Bluetooth, nastavení jasu displeje a další. Předlohou se zde evidentně stal operační systém Android. Další změnou prošlo notifikační centrum, které nyní obsahuje záložky tři. Apple se distancuje od NFC technologie a přidává technologii vlastní s názvem AirDrop, který k přenosu využívá Wi-Fi a také Bluetooth. Tato verze iOS není podporována iPhonem 3GS a iPodem Touch 4G (Costello, 2015).

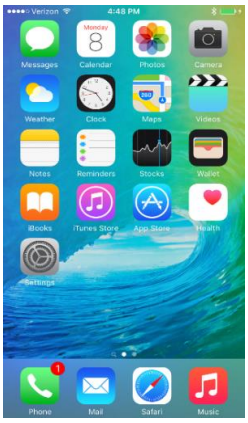
3.2.2.2.8 iOS 8. X

Následující verze iOS 8 byla představena 2. června 2014. Nový systém přináší spoustu změn. Nyní je možné emaily a další texty jednoduše nadiktovat. Rozpoznávání hlasu je na velmi vysoké úrovni a nechybí podpora českého jazyka. Vylepšená je práce s emaily, notifikačním centrem a widgety. Novinkou je zde funkce prediktivního zadávání textu QuickType, čeština však prozatím podporovaná není. Apple v této verzi konečně do svého fotoaparátu přidává samospoušť. Velkým hitem se stává funkce Continuity, která umožnuje dodělat rozdělanou práci z iPhonu na počítači a obráceně. Tentokrát byla ukončena podpora jenom jednomu zařízení a to iPhonu 4 (Theverge, 2013).

3.2.2.2.9 iOS 9. X

Nejnovější verze iOS 9 byla vydána na podzim roku 2015. Systém podporuje všechna zařízení, které podporoval iOS 8. Nová verze nám slibuje lepší stabilitu než u verze předešlé. Na zamžené obrazovce nás uvítá font San Francisco. Novinkou je zde tlačítko zpět, které vás přesune do aplikace nebo okna, z kterého jste přišli. Apple od této verze slibuje větší výdrž baterie a to až o hodinu navíc. K tomuto účelu slouží nový režim nízké spotřeby, který se automaticky nabídne, pokud baterie klesne na 20%. Další změnou jsou pak tři možnosti multitaskingu, kdy je možné pracovat s více aplikacemi najednou, toto vylepšení je možné jen na novějších iPadech. Velkou změnou prošly poznámky, které se staly z maximálně jednoduchého poznámkového bloku velice zajímavou aplikací, která může šlapat na paty zavedeným značkám jako je Evernote (Theverge, 2013), (Costello, 2015).

Obrázek 4 - Obrazovka iOS 9. X



Zdroj: (Bamburic, 2015).

3.2.2.3 Swift

Swift je multi-paradigmatický, kompilovaný programovací jazyk od společnosti Apple pro vývoj na platformách Mac OS X a iOS. Jinými slovy je to programovací jazyk, který podporuje více programovacích stylů, a jehož zdrojový kód zpracovává kompilátor. Jazyk Swift, který vznikl roku 2014, je zamýšlen jako alternativa k programovacímu jazyku Objective-C.

Jako každý jiný programovací jazyk i Swift byl ovlivněn jinými jazyky: Objective-C, Haskell, Ruby, Pythonem a jazykem D. Díky překladači LLVM může být Swift kompilován spolu s kódem jazyka C, Objective-C a Objective-C++.

Tento programovací jazyk umí spolupracovat s frameworky Cocoa a Cocoa Touch, které zajišťují běhové prostředí aplikací v operačních systémech Mac OS X a iOS. Jako vývojové prostředí pro Swift je zvolen Xcode.

Ke konci roku 2015 byla uvolněna nová verze Swift 2.0, která přináší několik novinek. Byl vylepšen model pro zachycení chyb programu, umožňující vytvořit vlastní text. Navíc je Swift rychlejší a jeho licence je nyní open source, dále došlo k malé změně syntaxe. Swift nabízí větší bezpečí díky kontrole proměnných před spuštěním. Svůj kód je možné transformovat na nejnovější verzi pomocí jednoduchého nástroje (Apple, 2016).

3.2.2.3.1 Swift vs. Objective-C

Z pohledů vlastností jazyka se jedná z větší části o obdobu Objective-C za využití moderních konceptů a syntaxe. Na rozdíl od jazyka Objective-C, jazyk Swift nevyužívá pointery, ale v případě potřeby je však možné jejich použití. Dalším rozdílem je nahrazení Smalltalkového způsobu volání metod za tečkovou notaci a jmenné prostory. Výhodou Swiftu je fakt, že by neměl dovolit takové množství chyb programátora jako u jazyku Objective-C, zejména proto, že Objective-C je jazykem dynamicky typovaným, kterému nejsou známy informace v době kompilace. Swift na rozdíl od Objective-C podporuje silné datové typy, díky čemuž zná celý typ každého objektu, a může tak lépe optimalizovat kód již během kompilace. Byla přidána podpora generik, bez kterých by se řada situací řešila obtížně. Jednou z dalších změn je změna koncovky zdrojového kódu na „.swift“ (Apple, 2016).

Obrázek 5 - Rozdíl mezi ObjC a Swiftem

Objective-C

```
const int count = 10;
double price = 23.55;

NSString *firstMessage = @"Swift is awesome. ";
NSString *secondMessage = @"What do you think?";
NSString *message = [NSString stringWithFormat:@"%@%@", firstMessage, secondMessage];

NSLog(@"%@", message);
```

Swift

```
let count = 10
var price = 23.55

let firstMessage = "Swift is awesome. "
let secondMessage = "What do you think?"
var message = firstMessage + secondMessage

print(message)
```

Zdroj: vlastní zpracování.

3.2.2.3.2 Ukázka kódu

Ve Swiftu je zvykem vytvářet raději konstanty, nežli proměnné, díky většímu výkonu a šetření paměti. Výhodou této metody je rovněž nízké riziko chyb, Swift nenechává do konstanty nic přiřadit, tudíž ji není možné přepsat na jinou hodnotu.

Základní číselné typy používané v programovacím jazyku Swift jsou: Int, UInt, Float, Double. Swift je stejně jako Objective-C case-sensitive, proto je důležité si dát pozor na velikost písmen. Avšak příkazy nemusí být odděleny středníkem jako u programovacího jazyku Objective-C (Apple, 2016).

Obrázek 6 - Ukázka kódu jazyka Swift

```
// Proměnné ve Swiftu začínají na "var", následovány jménem, typem a hodnotou
var explicitDouble: Double = 70
// Konstanty jsou definovány pomocí "let, následovány jménem, typem a hodnotou
let numberOfBananas: Int = 10
// Hodnoty proměnných mohou být použity v řetězcích následujícím způsobem
let appleSummary = "Mám \(numberOfApples) jablek."
let fruitSummary = "Mám\(numberOfApples + numberOfOranges) kusů ovoce."
// definice pole
var fruits = ["mango", "kiwi", "avocado"]
// ukázka if podmínky
if fruits.isEmpty {
    println("No fruits in my array.")
} else {
    println("There are \(fruits.count) items in my array")
}

//definice slovníky se jménem a věkem
let people = ["Anna": 67, "Beto": 8, "Jack": 33, "Sam": 25]

// ukázka získání více hodnot v jednom průběhu iterace
for (name, age) in people {
    println("\(name) is \(age) years old.")
}

// funkce a metody jsou definovány pomocí "func"
// návratový typ je definován pomocí ->
func sayHello(personName: String) -> String {
    let greeting = "Ahoj, " + personName + "!"
    return greeting
}
```

Zdroj: upraveno dle (Wikipedia, 2016).

3.2.2.4 Xcode

Xcode je vývojové prostředí od firmy Apple Inc., které bylo poprvé uvolněno roku 2003. Jedná se o profesionální soubor nástrojů pro tvorbu aplikací na platformy iOS a Mac OS X. Nejnovější stabilní verzí tohoto vývojového prostředí je verze 7.2, která je volně stažitelná, avšak pouze pro operační systémy OS X (Apple, 2016).

3.2.2.4.1 Funkce

Xcode nabízí propracovaný editor kódu, ve kterém je možné programovat aplikace pro všechny Apple zařízení, včetně chytrých hodinek Apple Watch. Tento editor skenuje text při psaní kódu. Pokud detekuje syntaktickou chybu, vyznačí řádek a zobrazí zprávu popisující problém. Xcode na rozdíl od Android Studia podporuje více programovacích jazyků, a to: C, C++, Objective-C, Objective-C++, Java, AppleScript, Python, Ruby, Rez a nově také Swift 2. Vývojové prostředí Xcode uživateli také poskytuje možnost vytvoření GUI[58], bez nutnosti psát jakýkoliv kód, díky Interface Builderu. Nechybí zde ani nástroje pro testování výkonu aplikací (Apple, 2016).

3.2.2.4.2 Životní cyklus aplikace

V životním cyklu aplikace na platformě iOS může dojít k těmto stavům:

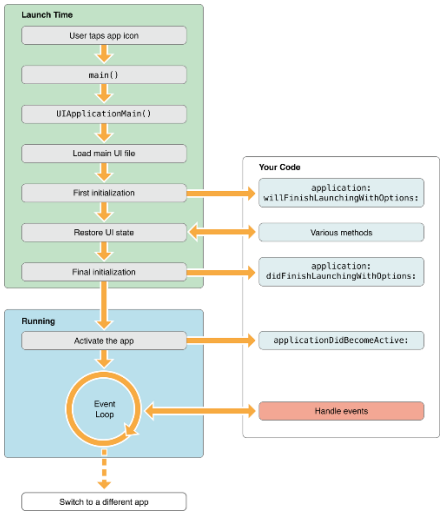
- Aplikace není spuštěna: Aplikace byla ukončena systémem, nebo zatím nebyla spuštěna.
- Aktivní aplikace: Aplikace je spuštěná v popředí a reaguje na příchozí události.
- Neaktivní aplikace: Aplikace je sice spuštěna v popředí, ale v současnosti se jí nedostává žádných vstupních událostí. V takovém případě aplikace po chvíli většinou přejde do některého z dalších stavů. V případě, že uživatel v tuto chvíli reaguje na příchozí hovor, SMS, nebo zamkne obrazovku, přechod nenastává a aplikace zůstává neaktivní.

[58] Graphical User Interface – Grafické uživatelské rozhraní

- Aplikace je v pozadí: Aplikace je v pozadí a spouští definovaný kód. Většina aplikací prochází tímto stavem cestou do neaktivního stavu. Tento stav aplikace je podporován od verze systému iOS 4 na všech zařízeních umožňujících multitasking.

- Pozastavená aplikace: Stejně jako v předchozím stavu je aplikace v pozadí. Rozdílem ale je, že nespouští žádný kód. Do tohoto stavu je aplikace ve vhodnou chvíli přemístěna systémem iOS. Nachází-li se aplikace v pozastaveném stavu, může být opět uvedena do stavu aktivního. V případě nedostatku paměti pro běh aktivních aplikací může však systém bez varování odstranit některé pozastavené aplikace z paměti. Pro přechod mezi výše jmenovanými stavy využívá systém následující metody objektu „Application Delegate“.

Obrázek 7 - Životní cyklus aplikace iOS



Zdroj: (Apple, 2015).

3.2.2.4.3 Minimální požadavky

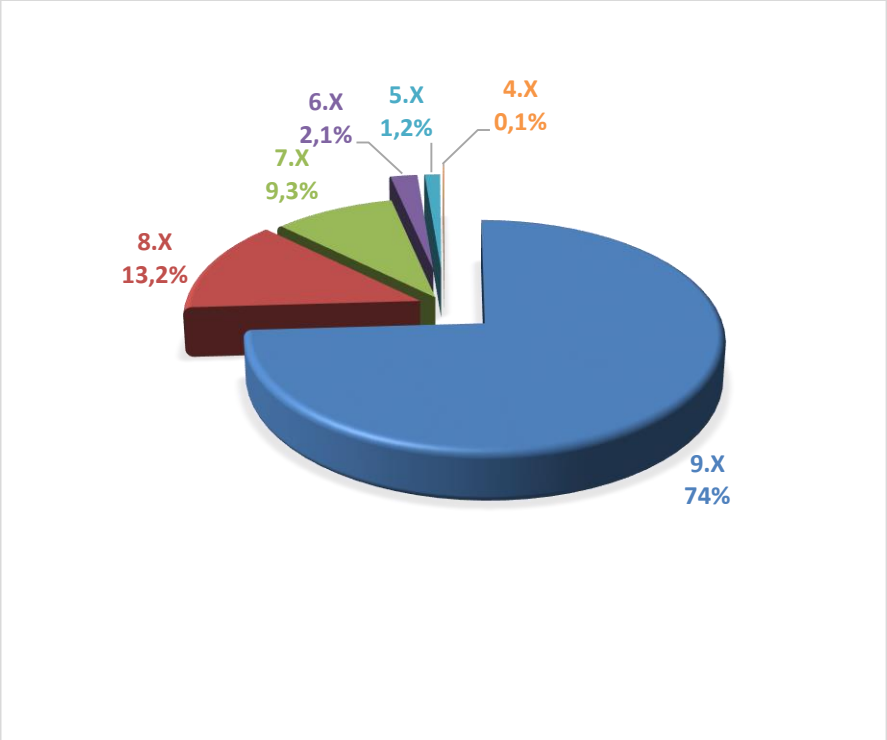
- Operační systém Mac OS X 10.10.5 Yosemite nebo starší;
- 4.41 GB volného prostoru;
- 4GB RAM (Apple, 2016).

3.2.2.5 Shrnutí

iOS je velmi oblíbený operační systém, který má mnoho příznivců, ale také mnoho odpůrců. Obecně jsou výrobky od firmy Apple Inc. považovány spíše za designovou záležitost. iOS je po grafické stránce velmi povedený a přehledný. Systém je navíc také velmi plynulý a poskytuje provázanost mezi všemi produkty Apple, což vidím jako velkou výhodu.

Nevýhodou je uzavřenost systému a kompatibilita pouze s telefony iPhone. Největším problémem je dle mého názoru jeho cena, která je mnohonásobně vyšší než produkt značky konkurenční.

Graf 2 - Statistika používaných verzí iOS



Zdroj: upraveno dle (Smith, 2016)

4 Vlastní zpracování

Cílem praktické části byla tvorba autorem navržené aplikace pomocí vývojového prostředí Android Studio a vývojové rozhraní Xcode. Aplikace nebyla vložena do žádných mobilních obchodů, protože zatím není kompletní a bude s ní dále pracováno v diplomové práci.

Níže budou podrobně rozpracovány základní charakteristiky vypracovaného projektu.

4.1 Charakteristika vytvořené aplikace

Autorem navržená aplikace je unikátní, byla navržena na základě jeho každodenních potřeb. Měla by sloužit k usnadnění pohybu a celkové orientaci v pražském metru. Uživatel nejdříve zvolí linku a stanici metra odjezdu, následně zvolí linku a stanici cílové destinace. Na základě toho bude uživateli zobrazeno číslo vagónu a dveří metra, které má použít při vstupu do dopravního prostředku tak, aby byl co nejblíže k požadovanému východu stanice metra.

Aplikace nečerpá ze žádných databázových pramenů, tudíž může fungovat i bez internetového připojení. Pro vytvoření aplikace musel autor provést náležité výpočty a zmapovat všechny stanice metra. Z důvodu omezené časové kapacity je provozuschopná pouze linka „A“. Z důvodu ochrany podnikatelského záměru bude tvorba aplikace pouze stručně nastíněna.

Grafické prvky a obrázky byly upravovány v grafickém prostředí Adobe Photoshop. Pro samotné kódování aplikace byly použity objektově orientované jazyky Java a Swift 2. Aplikace byla testována na virtuálních zařízeních, které jsou dostupné v jednotlivých vývojových prostředích. Aplikace pro Android byla poté testována na fyzickém zařízení Huawei Honor 6.

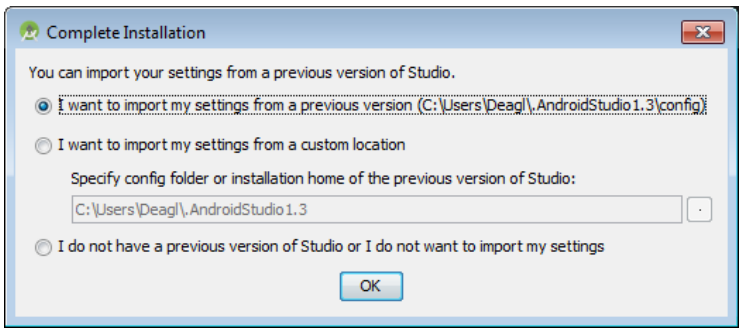
4.1.1 Android Studio

K vypracování aplikace byla autorem použita verze 1.5.1 a operační systém Windows 7. Níže budou popsány jednotlivé kroky vytváření aplikace v tomto vývojovém prostředí.

4.1.1.1 První spuštění

Po prvním spuštění nabízí program importovat nastavení z předchozích verzí studia, což výrazně zjednoduší případnou opětovnou instalaci či přenos nastavení.

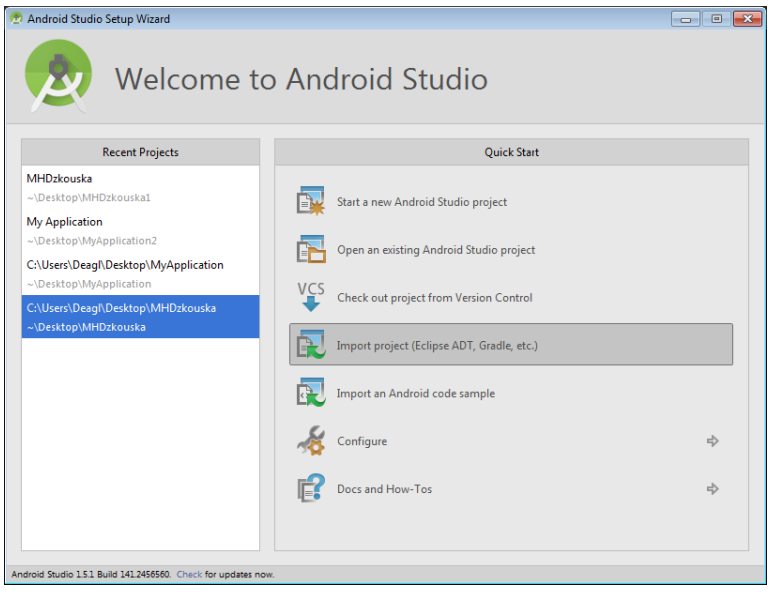
Obrázek 8 - První spuštění Android Studia



Zdroj: vlastní zpracování.

Po případném importování nastavení se vývojáři zobrazí úvodní nabídka studia. Android Studio zde nabízí migraci aplikace z vývojového prostředí Eclipse ADT, díky které je možné velice snadno změnit, Androidem již nepodporované, vývojové prostředí a přejít na prostředí novější. Stačí pouze vybrat možnost „Import project“ a najít složku s aplikací. Android Studio se pak samo o vše potřebné postará.

Obrázek 9 - Úvodní nabídka Android Studia

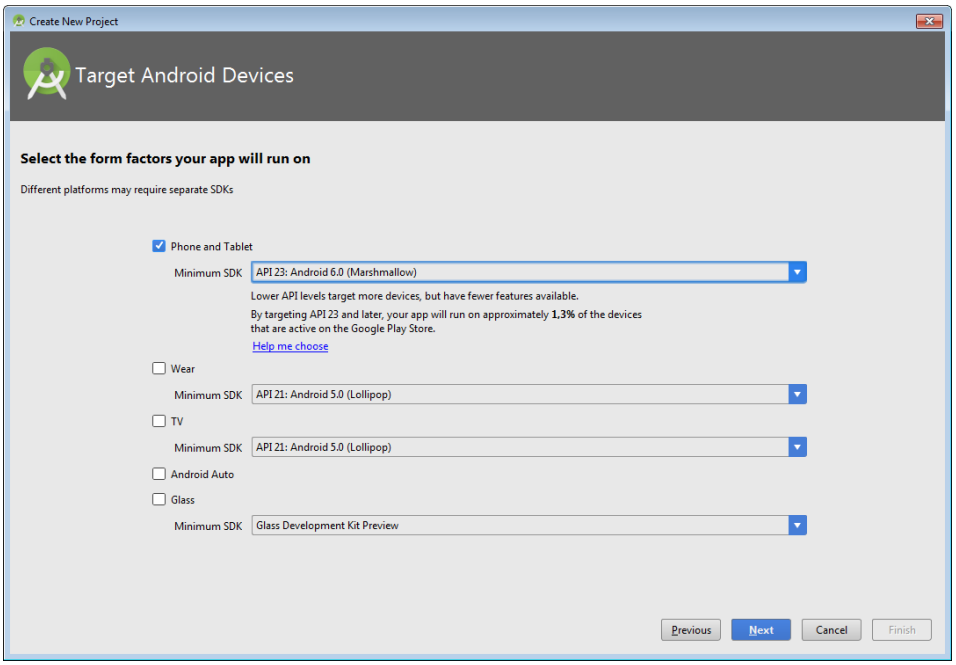


Zdroj: vlastní zpracování.

4.1.1.2 Vytvoření nové aplikace

Vytvoření nového projektu je v Android studiu jednoduché. Po vybrání položky „Start a new Android Studio project“ je potřeba zadat jméno projektu, složku uložení, případně doménu firmy. Po zadání je potřeba určit, pro jaké minimální SDK bude vývojářem naprogramovaná aplikace podporována. Tedy čím menší API úroveň, tím více potencionálních zákazníků, ale méně možností pro vývoj. API úrovně je možné vybrat i pro jiná zařízení než je mobilní telefon a to pro: chytré hodinky, TV a brýle. V případě, že programátor chce aplikaci vyvíjet pro všechna tyto zařízení, musí se API úroveň zařízení shodovat.

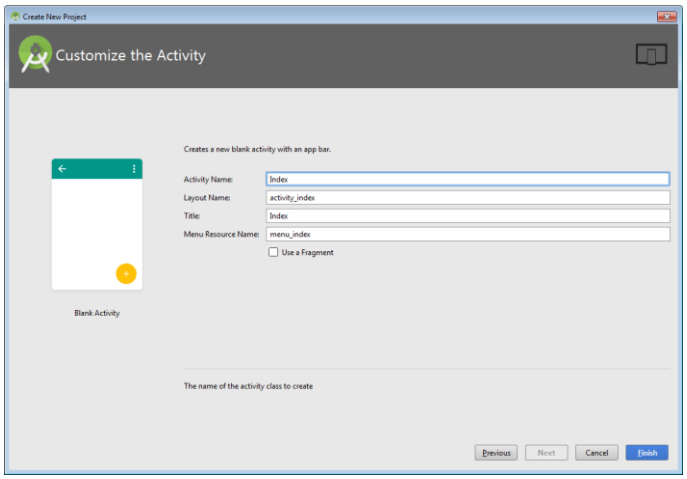
Obrázek 10 - Vytvoření nové aplikace - výběr API



Zdroj: vlastní zpracování.

Dalším krokem je výběr a pojmenování šablony pro aktivitu. K dispozici jsou různé typy šablon aplikací, které umožňují rychlejší nastavení uživatelského rozhraní. Vývojář vyplní název aktivity (Activity Name), název rámce (Layout Name), titulek (Title) a název menu (Menu Resource Name).

Obrázek 11 - Vytvoření nové aplikace - pojmenování aktivit



Zdroj: vlastní zpracování.

4.1.1.3 Vzhled a struktura prostředí

Android Studio se skládá z několika částí, z nichž nejdůležitější je navigace, programovací část, ovládací prvky studia a panel pro ukázku aplikace.

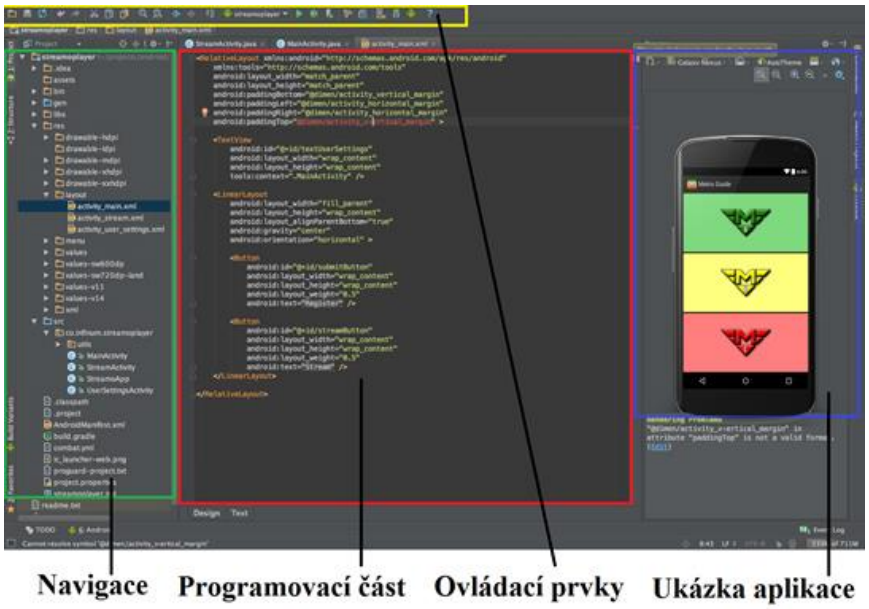
V navigaci je možné najít jednotlivé složky projektu, které je možné promítnout a otevřít je tak do programovací části dvojklikem myší.

V programovací části je možné přepínat mezi otevřenými kartami navigace a upravovat tak jejich podobu. Upravovat podobu karet lze pomocí psaní kódu nebo pomocí metody drag and drop.

V panelu zobrazujícím aktuální aplikaci se promítají změny, které je možné provést psaním kódu v programovací části.

V ovládacích prvcích lze najít tlačítka pro spuštění či ladění aplikace, vytvoření virtuálního stroje, uložení změn a další (Lacko, 2015).

Obrázek 12 - Vývojové prostředí Android Studio

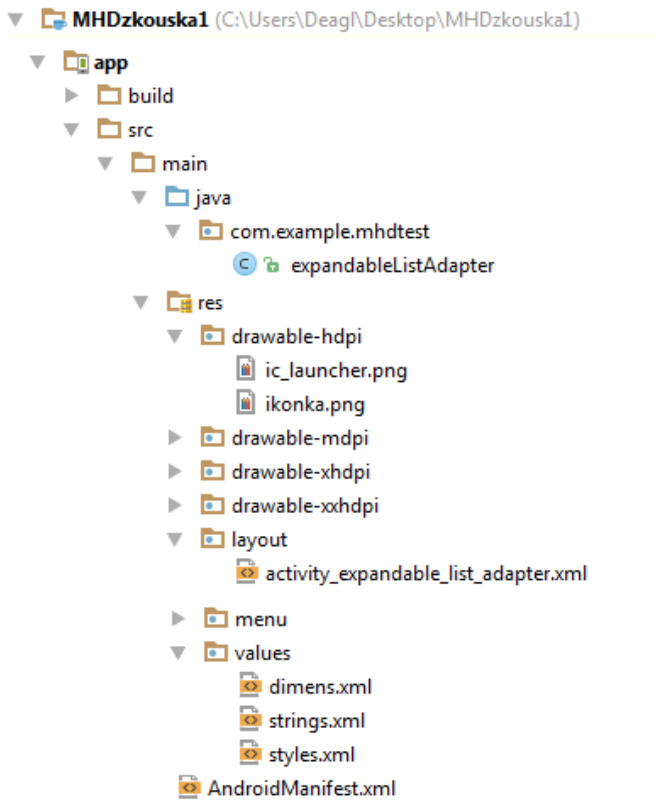


Zdroj: vlastní zpracování.

Ve struktuře projektu, která se nachází v navigaci, je možné najít několik důležitých složek pro projekt. V první řadě je to složka app, která se dělí na:

- build – do této složky se ukládá předkompilovaná část kódu při spuštění aplikace;
- libs – složka libraries, která obsahuje všechny knihovny aplikace;
- src – v této složce je důležitá hlavně složka Main, která se dělí na složky:
    - java – složka obsahující kódy pro aktivity;
    - res – složka obsahující složky jako:
        - drawable – složka pro obrázky;
        - layout – zde se nachází XML soubory, ve kterých se definuje vzhled;
        - values – zde se nachází proměnné aplikace;
- Manifest – nejdůležitější soubor v každé struktuře, obsahuje informace o naší aplikaci, které se předají Androidu předtím, než se spustí (Lacko, 2015).

Obrázek 13 - Ukázka nejdůležitějších složek struktury



Zdroj: vlastní zpracování.

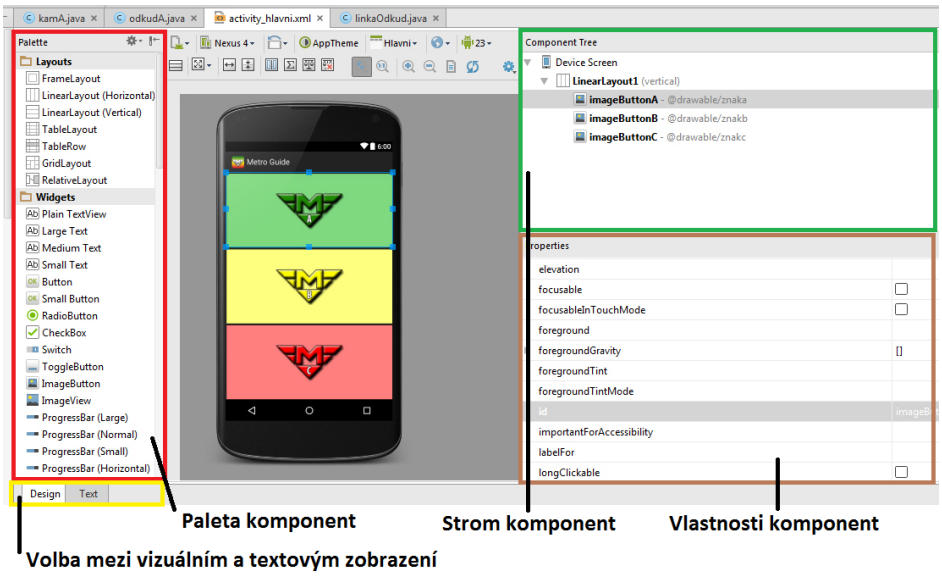
4.1.1.4 Vývoj aplikace

K vytvoření aplikace je nezbytné nejprve definovat funkce, které musí aplikace splňovat, dále navrhnout uživatelské prostředí a na závěr implementovat samotný zdrojový kód. V následujících krocích bude stručně popsána tvorba autorem vytvořené aplikace ve vývojovém prostředí Android Studio.

4.1.1.4.1 Návrh GUI

Pomocí metody drag and drop lze jednoduše vytvořit uživatelské rozhraní aplikace. K vytvoření aplikace byly použity komponenty „layout“, „textView“, „expandableListView“, „button“, „imageButton“, „imageView“ a „label“ nacházející se v paletě komponent. V levém dolním rohu lze přepnout mezi vizuálním a textovým zobrazení projektu. Stromová struktura vybraných komponent se nachází v pravém rohu. V pravém dolním rohu lze nastavit vlastnosti jednotlivých komponent.

Obrázek 14 – Tvorba GUI

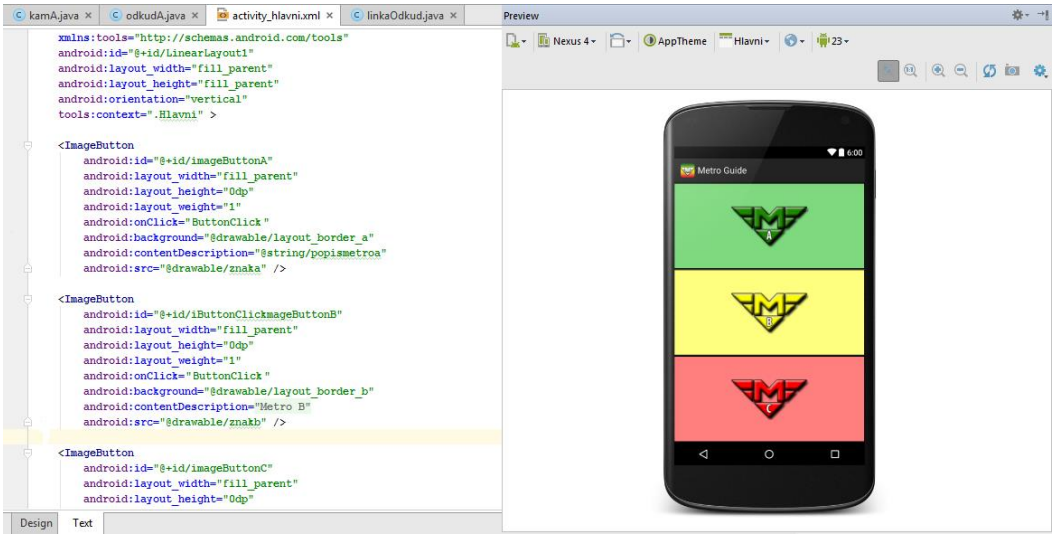


Zdroj: vlastní zpracování.

4.1.1.4.2 Funkce

Nejprve se definují základní funkce, které jsou žádoucí od vytvořené aplikace. Aplikace po spuštění zobrazí úvodní obrazovku s třemi tlačítky, které reprezentují linky metra A, B, C (obr. 15). Každé tlačítko má vlastnost „OnClick“ a také specifické „Id“, díky kterému se ve „switchi“ určí, jaké tlačítko se spustilo.

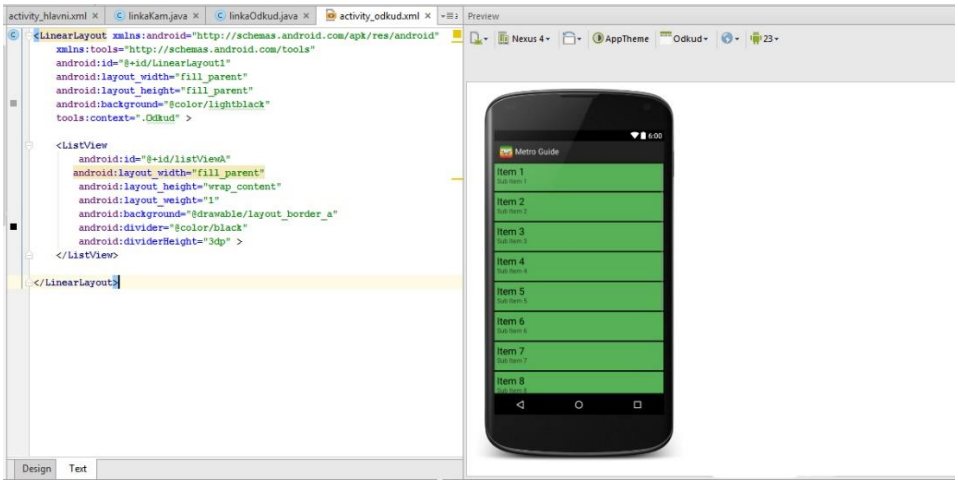
Obrázek 15 - Ukázka aplikace Android – XML hlavní obrazovky



Zdroj: vlastní zpracování.

Po zvolení linky se zobrazí další obrazovka s komponentem „listView“, který obsahuje jednotlivé stanice konkrétní linky metra (obr. 16).

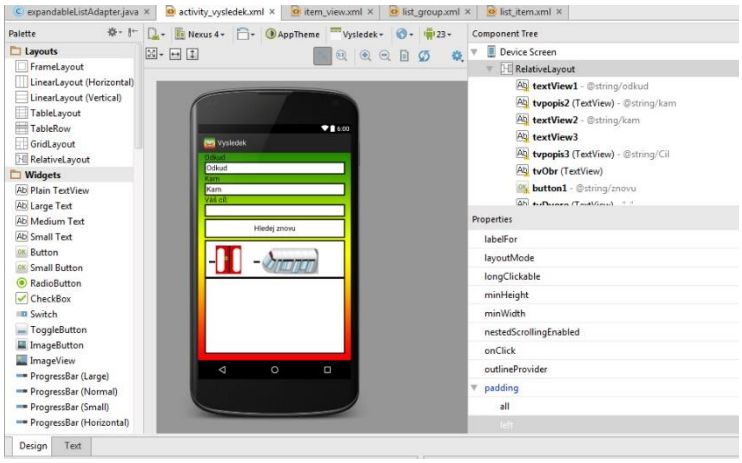
Obrázek 16 – Ukázka aplikace Android – druhá obrazovka s komponentou listView



Zdroj: vlastní zpracování.

Uživatel zvolí stanici, po té je odkázán na třetí obrazovku, která je vzhledově identická s úvodní (obr. 15). Třetí obrazovka uživatele pobídne ke zvolení cílové linky metra a po vybrání linky metra je uživatel odkázán na „listView“, které odpovídá druhé obrazovce (obr. 16). Na této obrazovce uživatel zvolí cílovou stanici metra. Aplikace podle zadaných dat vyhodnotí, kolik má stanice výstupních míst. Pokud má stanice pouze jeden výstup z metra, zobrazí se konečná obrazovka s výsledky (obr. 17).

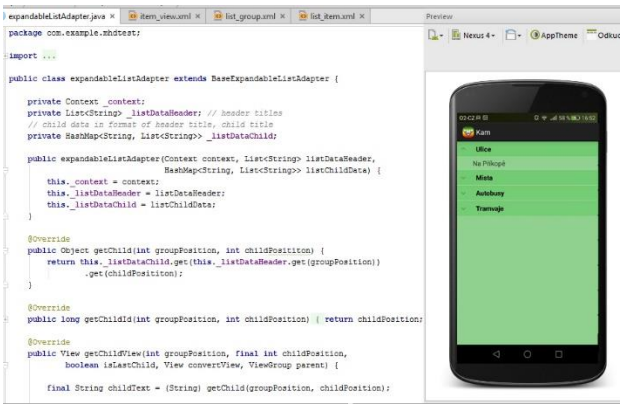
Obrázek 17 - Ukázka aplikace Android – výsledná obrazovka



Zdroj: vlastní zpracování

V případě, že stanice metra má více než jeden výstup, uživatel je odkázán na další obrazovku, která obsahuje komponentu „ExpandableListView“. Tato komponenta uživateli umožní vybrat požadovaný cíl (obr. 18.).

Obrázek 18 - Ukázka aplikace Android – třetí obrazovka



Zdroj: vlastní zpracování.

Podle zvoleného místa se zobrazí konečná obrazovka s výsledky (obr. 17). Výsledná obrazovka obsahuje „textView“ s informacemi o odjezdové stanici, cílové stanici, s číslem vagónu, označení dveří a slovním popisu navigace. Pro lepší a rychlejší ovládání byl na konečnou obrazovku přidán „button“ s názvem „hledej znovu“ odkazující na úvodní obrazovku. Vyhodnocení výsledků je provedeno pomocí „switche“ (Allen, 2013).

4.1.1.4.3 Ukázka implementace kódu

Implementaci zdrojového kódu lze provést v souborech „.java“. Ukázku zdrojového kódu metody „onClick“ lze vidět na obrázku číslo 20.

Obrázek 19 - Ukázka kódu Android - metoda onClick

```
public void buttonClick(View button) {
    Fragment fragment = null;
    switch (button.getId()) {
        case R.id.imageButtonA:
            fragment = new Afragment();
            break;
        case R.id.imageButtonB:
            fragment = new Bfragment();
            break;
        case R.id.imageButtonC:
            fragment = new Cfragment();
            break;
        default:
            break;
```

Zdroj: vlastní zpracování.

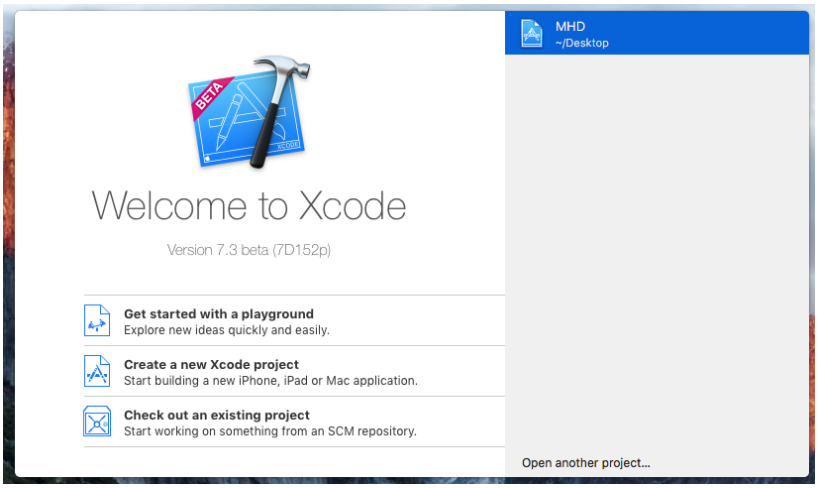
4.1.2 Xcode

Při druhé variantě tvorby mobilní aplikace byla použita verze vývojového prostředí 7.3 a operační systém Mac OS X 10.11 El Capitan. Níže je popsán krok za krokem práce v tomto prostředí.

4.1.2.1 První spuštění

Po prvním spuštění Xcode automaticky otevře okno obsahující seznam novinek a řadu zajímavých odkazů. V prvních krocích bývá vhodné určit některé z předvoleb, které lze najít v „Xcode/Preferences“. V kartě „Building“ lze zvolit složku, do které bude projekt ukládán. V pravém dolním rohu lze najít také užitečnou možnost „Always save“, umožňující uložit projekt před každou kompilací.

Obrázek 20 - První spuštění Xcode

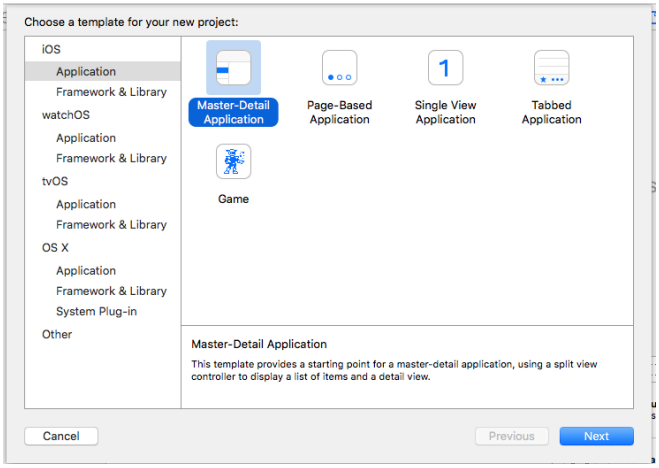


Zdroj: vlastní zpracování.

4.1.2.2 Vytvoření nové aplikace

Xcode nabízí několik šablon pro nový projekt, je možné si vybrat předlohy pro operační systémy iOS, watchOS, tvOS nebo OS X.

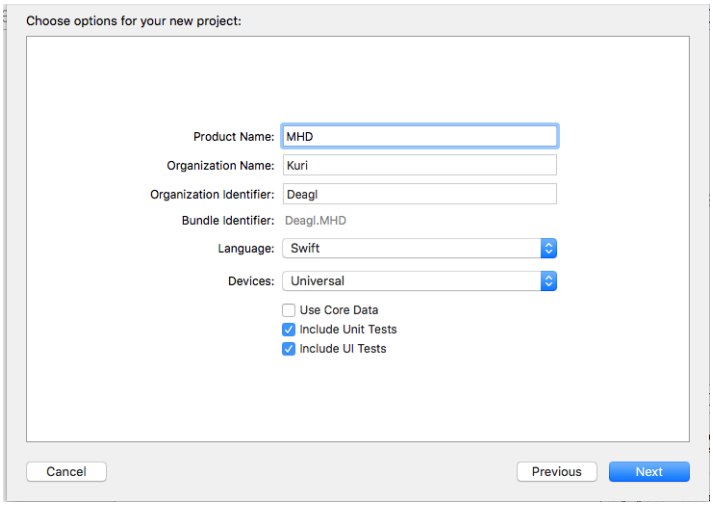
Obrázek 21 - Vytvoření nové aplikace iOS – volba šablony



Zdroj: vlastní zpracování.

Po zvolení programátorem vhodné šablony přichází na řadu zadání jména projektu, organizace, zvolení koncového zařízení aplikace a hlavně programovacího jazyka. Xcode nabízí programování iOS aplikací v jazyku Objective-C a Swift. V posledním kroku vývojář zvolí cílovou složku, kam bude aplikace uložena.

Obrázek 22 - Vytvoření nove aplikace iOS - volba nastavení



Zdroj: vlastní zpracování.

4.1.2.3 Vzhled a struktura prostředí

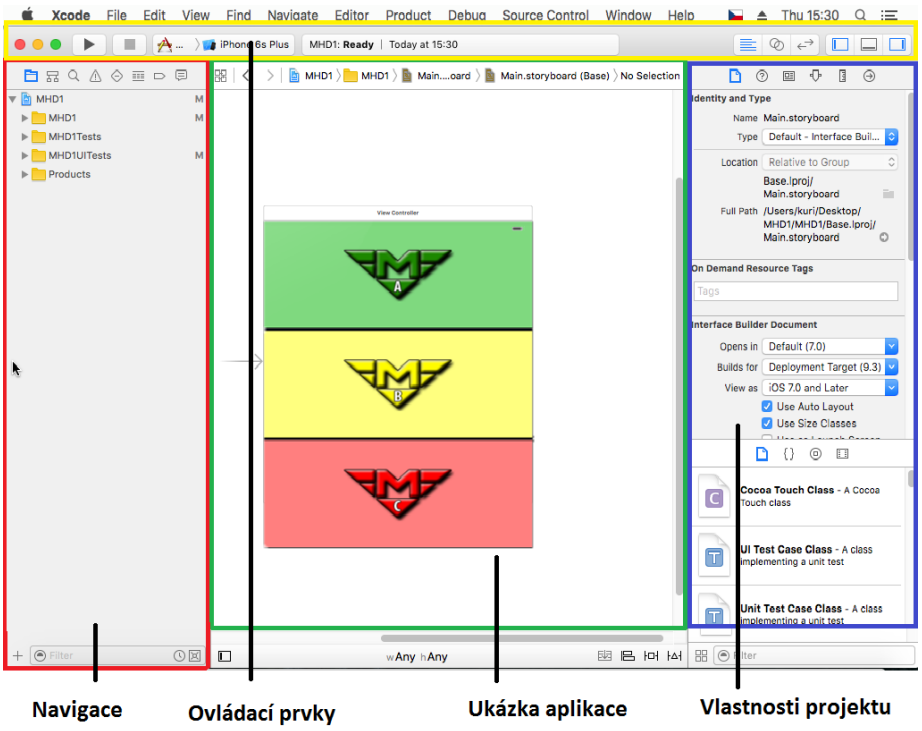
K základní orientaci slouží především levý sloupec v projektovém okně, v němž lze najít celou hierarchickou strukturu projektu. První řádek je nejdůležitější, neboť zastřešuje celou hierarchii, a skupiny složek uvnitř („Model“, „Other Sources“, „Resources“) slouží k lepší orientaci. Lze je jakkoliv změnit, prázdné složky zrušit, či vytvořit složky nové. Soubory lze také libovolně přemísťovat či měnit jejich pořadí pomocí myši, tyto akce nemají na projekt žádný vliv. Avšak tato funkce platí pouze pro skupiny, jež přímo obsahují soubory v projektu. Níže uvedené speciální skupiny jsou spolu s projektem na nejvyšší úrovni a jejich funkce je rozdílná.

Speciální skupiny složek:

- Targets – velmi důležitá složka, každý z prvků reprezentuje jeden cíl projektu a popisuje, jak jej vytvořit.
- Executables – složka reprezentující konkrétní výsledky cílů, používaná zejména při testování a ladění.
- Errors and Warnings – složka obsahující seznam hlášení překladače z posledního pokusu o sestavení projektu.
- Bookmarks – tato složka obsahuje seznam značek pro rychlý a pohodlný přístup na vybrané místo.
- SCM – složka související se správou historie všech změn, které byly provedeny v projektu.
- Project Symbols – obsahuje všechny symboly, které jsou v projektu definovány.
- Implementations Files – v nich je udržován seznam všech souborů z projektu, splňující zadané kritéria (Apple, 2016).

V dolní části vývojového prostředí lze nalézt oblast, kde se vyskytují informace o běhu projektu včetně chyb, kterých se programátor dopustil při vytváření aplikace. Uprostřed se nachází vizuální stránka aplikace, kterou je možné libovolně transformovat pomocí myši (metodou drag and drop). V pravé části se nachází programovací část, ve které je možné kód upravovat. V horní části vývojového prostředí se nachází ovládací prvky projektu.

Obrázek 23 - Popis vývojového prostředí Xcode



Zdroj: vlastní zpracování.

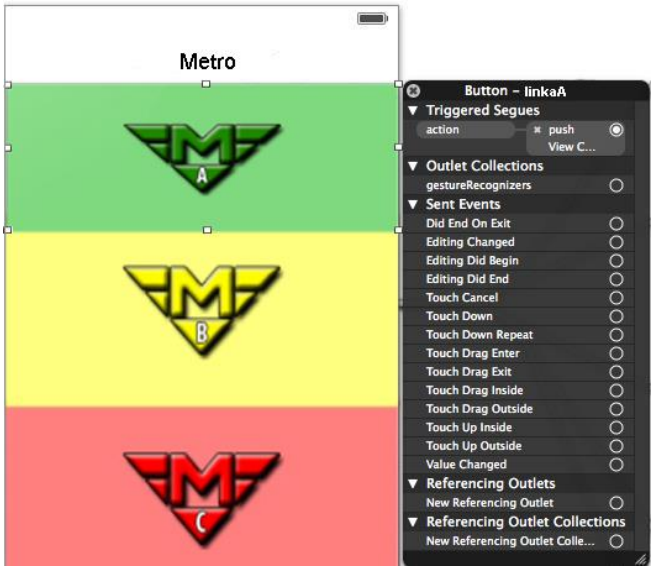
4.1.2.4 Vývoj

V následujících krocích bude nastíněna tvorba aplikace ve vývojovém prostředí Xcode. Funkce aplikace a vzhled obrazovky zůstávají obdobné jako u Android Studia.

4.1.2.4.1 Návrh GUI

Vytvoření stránky aplikace se provádí podobným způsobem jako ve vývojovém prostředí Android Studio, tzn. pomocí metody drag and drop. Struktura menu se vytváří částečně automaticky. Samotné odkazovaní pomocí tlačítka „button“ se provádí pomocí myši. Nejprve je nutné kliknout pravým tlačítkem myši na konkrétní „button“, v tomto případě na „linkaA“ a poté vybrat možnost „action“. Dále stačí klinout na stránku, na kterou chceme odkazovat (podrobněji viz. obrázek 25).

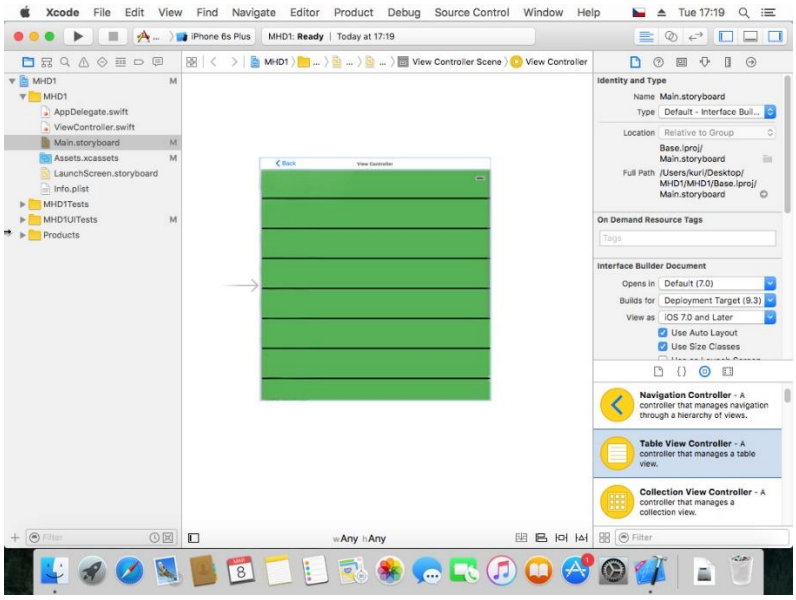
Obrázek 24 – Ukázka aplikace iOS – vytvoření hlavní obrazovky



Zdroj: vlastní zpracování.

K vytvoření druhé obrazovky byla použita komponenta „Table View“, která bude naplněna stanicemi zvolené linky.

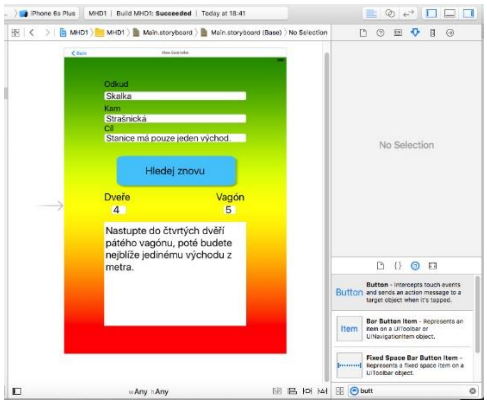
Obrázek 25 – Ukázka aplikace iOS – druhá obrazovka s komponentou Table View



Zdroj: vlastní zpracování.

Výsledná obrazovka aplikace obsahuje „Text Field“ s informacemi o odjezdové stanici, cílové stanici, s číslem vagónu, označení dveří a slovním popisu navigace. Na konečnou obrazovku byl dále přidán „button“ s názvem „hledej znovu“ odkazující na úvodní obrazovku.

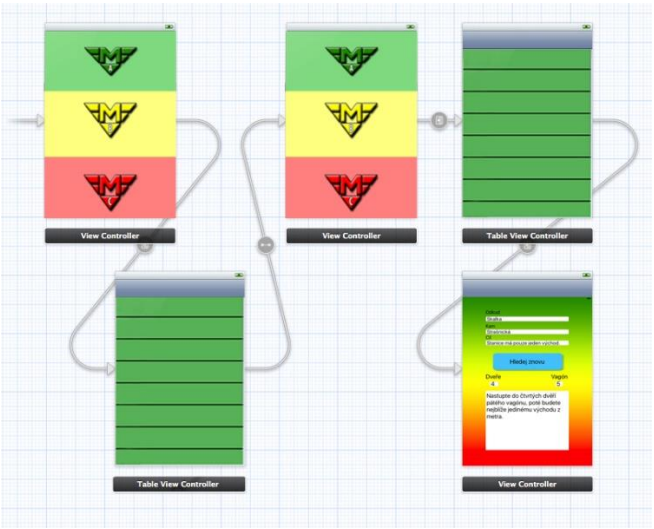
Obrázek 26 - Ukázka aplikace iOS – výsledná obrazovka



Zdroj: vlastní zpracování.

Po tvorbě jednotlivých stránek je zapotřebí stránky provázat. Provázání stránek lze provést pomocí jednoduchého přetahování myši a nastavení odkazu na stránku, která se má zobrazit. Po uskutečnění propojení všech stran, vznikne mapa „views“. Celou strukturu lze vidět na obrázku číslo 28 (Čada, 2009).

Obrázek 27 - Ukázka aplikace iOS – struktura projektu



Zdroj: vlastní zpracování.

4.1.2.4.2 Ukázka implementace kódu

Ukázku zdrojového kódu komponenty “Table View” lze vidět na obrázku 28.

Obrázek 28 – Ukázka aplikace iOS – zdrojový kód Table View

```
class StaniceTableViewController: UITableViewController {

    var data = ["Depo Hostivař", "Skalka", "Strašnická", "Želivského", "Flóra", "Jiřího z poděbrad", "Náměstí míru", "Muzeum", "Můstek",
        "Staroměstská", "Malostranská", "Hradčanská", "Dejvická", "Bořislavka", "Nádraží Veleslavín", "Petřiny", "Nemocnice Motol"]

    override func viewDidLoad() {
        super.viewDidLoad()
        self.tableView.editing = true
    }

    override func tableView(tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
        return data.count
    }

    override func tableView(tableView: UITableView, cellForRowAtIndexPath indexPath: NSIndexPath) -> UITableViewCell {
        let cell = tableView.dequeueReusableCellWithIdentifier("StaniceCell", forIndexPath: indexPath)
        cell.textLabel?.text = data[indexPath.row]
        return cell
    }

    override func tableView(tableView: UITableView, editingStyleForRowAtIndexPath indexPath: NSIndexPath) -> UITableViewCellEditingStyle {
        return .None
    }

    override func tableView(tableView: UITableView, shouldIndentWhileEditingRowAtIndexPath indexPath: NSIndexPath) -> Bool {
        return false
    }

    override func tableView(tableView: UITableView, moveRowAtIndexPath sourceIndexPath: NSIndexPath, toIndexPath destinationIndexPath: NSIndexPath)
        {
        let movedObject = self.data[sourceIndexPath.row]
        data.removeAtIndex(sourceIndexPath.row)
        data.insert(movedObject, atIndex: destinationIndexPath.row)
        NSLog("%@", "\(sourceIndexPath.row) => \(destinationIndexPath.row) \(data)")
    }
}
```

Zdroj: vlastní zpracování.

5 Zhodnocení výsledků

Vývojová prostředí byla hodnocena pomocí Saatyho metody vícekriteriální analýzy variant. K vyjádření preferencí byla použita Saatyho bodovací stupnice: 1 – kritéria jsou stejně významná, 3 – první kritérium je slabě významnější než druhé, 5 – první kritérium je silně významnější než druhé, 7 – první kritérium je velmi silně významnější než druhé, 9 – první kritérium je absolutně významnější než druhé, dále je možné použití mezistupňů (2, 4, 6, 8). Kritéria a váhy Saatyho metody byly zvoleny na základně konzultace s odborníky.

5.1 Zvolená kritéria

Pro výpočet Saatyho metody vícekriteriální analýzy variant byla autorem vybrána tato kritéria:

Vývojové prostředí

Vývojovým prostředím se rozumí přehlednost a dostupnost prostředí. Dále zde byly zohledněny nároky na PC. Android Studio je prostředím pro autora přehlednějším a jeho hardwarové nároky jsou značně nižší než u studia Xcode. Autor ohodnotil hardwarové nároky a přehlednost Android studia „velmi silně významnějšími“ před Xcode.

Podpora platforem OS

Znamená, na jakých operačních systémech je možné vývojové prostředí spustit. Xcode podporuje pouze platformu Mac OS X, zatímco Android Studio podporuje platformy Windows, Linux a dokonce i Mac OS X. Z hlediska autora je podpora platforem operačních systémů Android Studia „silně významnější“ před Xcode.

Náročnost programování

Náročností programování se rozumí, jak dobře musí programátor znát kód a také, co dokáže vytvořit bez jeho znalosti. Z vlastní zkušenosti autor usuzuje náročnost programování v prostředí Xcode jako „téměř silně významnější“ před Android Studiem.

Počet řádků kódu

Počet řádků kódu, který musí programátor napsat k dosažení svého cíle. Zdrojový kód je průměrně o 40% větší v Android Studiu než v prostředí Xcode. Počet řádků kódu v prostředí Xcode je hodnocen jako „silně významnější“ před Android Studiem.

Strávený čas psaním kódu

Strávený čas, který programátor musí vynaložit, aby dosáhl určeného cíle. Podle studie odborníků programátor průměrně stráví o 30% více času psaním kódu v prostředí Android Studia. V této kategorii je Xcode „téměř silně významnější“ před prostředím Android Studio.

Profit aplikací

Profitem aplikací se rozumí výdělek aplikací. I přes větší počet stahovaných aplikací v obchodě Google Play (Android), je výnosnější obchod Apple Store (iOS) (Perez, 2015). Profit aplikací Android je hodnocen jako „téměř absolutně významnější“ před iOS.

Možnost použití nových SDK

Možnosti použití nových SDK lze definovat schopnost programátora použít nové funkce softwarových nástrojů. Android i iOS nabízí dobrou podporu použití nových SDK. V této kategorii autor shledává Android „téměř stejně důležitým“ jako iOS.

Bezpečnost a kvalita aplikací v obchodě

Zde se hodnotí bezpečnost a celková kvalita aplikací v obchodu. Google play provádí rychlé schvalování aplikací, zatímco Apple Store nad každou jednotlivou aplikací stráví průměrně dva týdny. Díky tomu jsou aplikace v Apple Store bezpečnější a kvalitnější. Autor shledává bezpečnost a kvalitu aplikací v Apple Store jako „slabě významnější“ před Google Play.

Tabulka 4 – Výsledky Saatyho metody

Kritérium		**Dílčí ohodnocení variant**	
Název	**Váhy**	**Android**	**iOS**
Vývojové prostředí	0,08	0,88	0,12
Podpora platforem OS	0,15	0,83	0,17
Náročnost programování	0,05	0,25	0,75
Počet řádků kódu	0,15	0,17	0,83
Strávený čas psaním kódu	0,20	0,20	0,80
Profit aplikací	0,10	0,11	0,89
Možnost použití nových funkcí SDK	0,15	0,33	0,67
Bezpečnost a kvalita aplikací v obchodu	0,12	0,25	0,75

Celkové ohodnocení	0,33	0,67
Pořadí	**2**	**1**

Zdroj: vlastní zpracování.

Pomocí Saatyho metody byl zvolen preferovanějším operačním systémem iOS a jeho vývojové prostředí Xcode. Metoda může pomoci začínajícím vývojářům s výběrem platformy. Podrobnější informace o výpočtech Saatyho metody jsou uvedeny v kapitole Přílohy.

Z prováděného projektu jsou zřejmé uvedené výhody a nevýhody vývojových prostředí Android Studio a Xcode.

Tabulka 5 - Android Studio – výhody a nevýhody

Výhody	Nevýhody
✓ Podpora operačních systémů Linux, Windows, Mac OS X; ✓ Více rozšířený programovací jazyk Java; ✓ Možnost aktualizovat aplikaci během několika hodin; ✓ Nízká cena poplatku obchodu; ✓ Obchod přátelský vůči vývojáři; ✓ Menší nároky na PC; ✓ Lehké vytvoření GUI.	✓ Menší počet pluginů; ✓ Delší zdrojový kód; ✓ Více stráveného času psaním kódu; ✓ Nízký profit z prodeje aplikací; ✓ Špatné virtuální emulátory; ✓ Málo zařízení využívající nejnovější API úrovně; ✓ Aplikace v obchodě Google Play jsou potencionálně nebezpečnější.

Zdroj: vlastní zpracování.

Tabulka 6 – Xcode – výhody a nevýhody

Výhody	Nevýhody
✓ Výběr ze dvou programovacích jazyků; ✓ Kvalitnější aplikace v obchodě; ✓ Vysoký profit z prodeje aplikací; ✓ Kratší zdrojový kód; ✓ Méně stráveného času psaním kódu; ✓ Kvalitní virtuální emulátory; ✓ Aplikace nejsou potencionálně nebezpečné; ✓ Lehké vytvoření GUI; ✓ Uživatelé rychle aktualizují OS	✓ Aktualizace aplikace trvá až tři týdny; ✓ Vyšší cena poplatků obchodu; ✓ Větší nároky na PC; ✓ Podpora pouze operačního systému Mac OS X; ✓ Xcode stále nemá optimalizační nástroj kódu pro Swift 2; ✓ Menší komunita programátorů = méně materiálu pro výuku.

Zdroj: vlastní zpracování.

6 Závěr

Vývoj mobilních aplikací vyžaduje zahrnout do plánování samotné aplikace specifické vlastnosti mobilních zařízení. Na rozdíl od stolních počítačů fungují na baterii a mají tak omezenou energii na provoz. Vývojáři mobilních aplikací se proto snaží vytvořit uživatelsky co nejpříjemnější aplikace, co se obsahu i ovládání týče. Je tedy zapotřebí zvážit zvolení vhodné platformy a s ní souvisejícího vývojového prostředí.

Cílem této práce bylo charakterizovat základní pojmy týkající se platforem IOS a Android a jejich vývojových prostředí Xcode a Android Studio. V práci byla podrobně rozebrána jejich stavba, základní struktura, vývoj, historie, zabezpečení a typický programovací jazyk.

Hlavním cílem praktické části bylo porovnat vývoj autorem vytvořené aplikace v prostředí Android Studio a Xcode pomocí vícekriteriální analýzy variant Saatyho metodou. Metoda může pomoci začínajícím vývojářům s výběrem platformy. Pomocí Saatyho metody byl zvolen preferovanějším operačním systémem iOS a jeho vývojové prostředí Xcode (viz. tabulka 4). Při použití jiných metod by byl pravděpodobně dosažen výsledek rozdílný.

Z provedeného projektu jsou zřejmé uvedené výhody a nevýhody vývojových prostředí Android Studio a Xcode (viz. tabulka 5, 6). Vytvoření jednoduché aplikace nemůže promítnout všechny klady a zápory vývojových prostředí. K důkladnému zjištění výhod a nevýhod je zapotřebí naprogramovat aplikaci složitějšího charakteru, což by však trvalo podstatně déle, než je žádoucí v rámci bakalářské práce. Na tuto práci bude v budoucnu možné navázat v diplomové práci.

7 Citovaná literatura

Allen, Grant. 2013. *Android 4: průvodce programováním mobilních aplikací.* Brno : Computer Press, 2013. ISBN 9788025137826.

Android. 2016. <uses-sdk>. *Developer-Android.* [Online] 5. Leden 2016. [Citace: 16. Leden 2016.] http://developer.android.com/guide/topics/manifest/uses-sdk-element.html.

—. 2016. Android Studio. *Developer-Android.* [Online] 3. Leden 2016. [Citace: 19. Leden 2016.] http://developer.android.com/sdk/index.html.

—. 2016. Dashboard. *Developer-Android.* [Online] 7. Únor 2016. [Citace: 25. Únor 2016.] http://developer.android.com/about/dashboards/index.html.

—. 2015. The Android Story. *Android.* [Online] 14. Listopad 2015. [Citace: 25. Únor 2015.] https://www.android.com/history/.

Apple. 2016. Swift. *Developer-Apple.* [Online] 2016. [Citace: 14. Leden 2016.] https://developer.apple.com/swift/.

—. 2015. The App Life Cycle. *Developer-Apple.* [Online] 16. Září 2015. [Citace: 5. Únor 2016.] https://developer.apple.com/library/ios/documentation/iPhone/Conceptual/iPhoneOSProgrammingGuide/TheAppLifeCycle/TheAppLifeCycle.html.

—. 2016. Xcode. *Developer-Apple.* [Online] 2016. [Citace: 16. Leden 2015.] https://developer.apple.com/xcode/.

Bamburic, Mihăiță. 2015. First look at iOS 9 and OS X 10.11 El Capitan public betas. *Betanews.* [Online] 24. Červen 2015. [Citace: 8. Leden 2016.] http://betanews.com/2015/07/10/first-look-at-ios-9-and-os-x-10-11-el-capitan-public-betas/.

Car, Tomislav. 2015. Android development is 30% more expensive than iOS. And we have the numbers to prove it! *Infinum.* [Online] 27. Listopad 2015. [Citace: 15. Únor 2016.] https://infinum.co/the-capsized-eight/articles/android-development-is-30-percent-more-expensive-than-ios.

Costello, Sam. 2015. The History of the iOS. *About.* [Online] 20. Prosinec 2015. [Citace: 4. Leden 2016.] http://ipod.about.com/od/iphonesoftwareterms/a/firmw_history.htm.

Čada, Ondřej. 2009. *Cocoa: úvod do programování počítačů Apple.* Praha : Grada, 2009. ISBN 9788024727783.

Eckel, Bruce. 2000. *Myslíme v jazyku Java.* Praha : Grada, 2000. ISBN 8024790106.

Elitecsoftware. 2012. Historie verzí platformy Android. *Elitecsoftware.* [Online] 11. Duben 2012. [Citace: 15. Prosinec 2015.] http://www.elitecsoftware.cz/historie-verzi-platformy-android/.

Hlavík, Jiří. 2015. 3. díl - Android programování - Životní cyklus a nový projekt. *ITnetwork.* [Online] 17. Únor 2015. [Citace: 14. Únor 2016.] http://www.itnetwork.cz/java/android/tutorial-programovani-pro-android-v-jave-zivotni-cyklus-a-novy-projekt/.

IDC. 2015. Smartphone OS Market Share, 2015 Q2. *IDC.* [Online] 14. Srpen 2015. [Citace: 4. Prosinec 2015.] http://www.idc.com/prodserv/smartphone-os-market-share.jsp.

Isaacson, Walter. 2014. *Steve Jobs.* Praha : Práh, 2014. ISBN 9788072525881.

Kilián, Karel. 2015. Historie Androidu v kostce aneb Od verze 1.0 až po Android M. *Světandroidia.* [Online] 24. Červen 2015. [Citace: 10. Prosinec 2015.] http://www.svetandroida.cz/historie-androidu-201506.

Kovařík, David. 2012. Operační systém v telefonu aneb nahlédněte do světa smartphonů (vědecké okénko). *Mobilizujeme.* [Online] 11. Březen 2012. [Citace: 10. Prosinec 2015.] http://mobilizujeme.cz/clanky/operacni-system-v-telefonu-aneb-nahlednete-do-sveta-smartphonu-vedecke-okenko/.

Lacko, Ľuboslav. 2015. *Vývoj aplikací pro Android.* Brno : Computer Press, 2015. ISBN 9788025143476.

Oracle. 2016. Java. *Oracle.* [Online] 26. Únor 2016. [Citace: 2016. Únor 2016.] http://www.oracle.com/technetwork/java/index.html.

Perez, Sarah. 2015. Why developers STILL prefer iOS over Android. *BGR.* [Online] 14. Duben 2015. [Citace: 13. Únor 2016.] http://bgr.com/2015/04/15/ios-vs-android-developers-revenue-apps/.

Phonescoop. 2012. Smartphone. *Phonescoop.* [Online] 4. Duben 2012. [Citace: 4. Prosinec 2015.] http://www.phonescoop.com/glossary/term.php?gid=131.

Smith, David. 2016. iOS Version Stats. *David Smith.* [Online] 25. Únor 2016. [Citace: 25. Únor 2016.] https://david-smith.org/iosversionstats/.

Smrček, Jakub. 2011. Google Android – velký výlet do historie. *Cnews.* [Online] 3. Březen 2011. [Citace: 5. Leden 2016.] http://www.cnews.cz/google-android-velky-vylet-do-historie.

Strauss, Morgan. 2010. Touch Screens That Changed the World. *GUIFIX.* [Online] 27. Leden 2010. [Citace: 4. Prosinec 2015.] http://blog.guifx.com/2010/01/27/touchscreens-that-changed-the-world/.

Tanenbaum, Andrew S. 2008. *Modern operating systems.* Upper Saddle River : Prentice Hall, 2008. ISBN 0130313580.

Theverge. 2013. iOS: A visual history. *Theverge.* [Online] 16. Červenec 2013. [Citace: 15. Leden 2016.] http://www.theverge.com/2011/12/13/2612736/ios-history-iphone-ipad.

Ujbányai, Miroslav. 2012. *Programujeme pro android.* Praha : Grada, 2012. ISBN 9788024739953.

Wikipedia. 2015. Android Marshmallow. *Wikipedia.* [Online] 12. Prosinec 2015. [Citace: 4. Leden 2016.] https://cs.wikipedia.org/wiki/Android_Marshmallow.

—. 2016. Swift (programovací jazyk). *Wikipedia.* [Online] 2016. [Citace: 8. Leden 2015.] https://cs.wikipedia.org/wiki/Swift_(programovac%C3%AD_jazyk).

Williams, Rhiannon. 2015. Apple iOS: a brief history. *Telegraph.* [Online] 17. Září 2015. [Citace: 5. Leden 2016.] http://www.telegraph.co.uk/technology/apple/11068420/Apple-iOS-a-brief-history.html.

Zandl, Patrick. 1997. Nokia 9000 Communicator (recenze). *Mobil.idnes.* [Online] 25. Srpen 1997. [Citace: 4. Prosinec 2015.] http://mobil.idnes.cz/nokia-9000-communicator-recenze-dkp-/mob_nokia.aspx?c=970825_0003488_telefony.

8 Přílohy

Příloha 1 - Saatyho metoda

Vývojové prostředí	Android Studio	Xcode	b_i	Ohodnocení
Android Studio	1	7	2,65	0,88
Xcode	0,14	1	0,37	0,12

Podpora platforem OS	Android Studio	Xcode	b_i	Ohodnocení
Android Studio	1	5	2,24	0,83
Xcode	0,2	1	0,45	0,17

Náročnost programování	Android Studio	Xcode	b_i	Ohodnocení
Android Studio	1	0,33	0,57	0,25
Xcode	3	1	1,73	0,75

Počet řádků kódu	Android Studio	Xcode	b_i	Ohodnocení
Android Studio	1	0,2	0,45	0,17
Xcode	5	1	2,24	0,83

Strávený čas psaním kódu	Android Studio	Xcode	b_i	Ohodnocení
Android Studio	1	0,25	0,5	0,20
Xcode	4	1	2	0,80

Profit aplikací	Android Studio	Xcode	b_i	Ohodnocení
Android Studio	1	0,125	0,35	0,11
Xcode	8	1	2,83	0,89

Možnost použití nových funkcí SDK	Android Studio	Xcode	b_i	Ohodnocení
Android Studio	1	0,5	0,71	0,33
Xcode	2	1	1,41	0,67

Bezpečnost a kvalita aplikací obchodu	Android	iOS	b_i	Ohodnocení
Android	1	0,33	0,57	0,25
iOS	3	1	1,73	0,75

Příloha 2 – Ukázka aplikace na fyzickém zařízení android