Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačních technologií

Bakalářská práce

Aplikace android – teorie a praxe

Kalamár Dominik

© 2016 ČZU v Praze

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Dominik Kalamár

Informatika

Název práce

Aplikace android – teorie a praxe

Název anglicky

Aplikace android – theory and practice

Cíle práce

Cílem bakalářské práce je charakterizovat problematiku vývoje aplikací pro operační systém Android. Ten se skládá z následujících dílčích cílů: popis architektury platformy Android a vývojových nástrojů, tvorba aplikace v programovacím jazyce Java, distribuce výsledné aplikace.

Metodika

Metodika je založena na studiu a analýze odborných informačních zdrojů. Na základě těchto zdrojů bude vytvořena jednoduchá aplikace. Na základě syntézy teoretických a praktických poznatků bude formulován závěr bakalářské práce.

Doporučený rozsah práce

30 -40 stran

Klíčová slova

Aplikace Android, architektura platformy Android, programovací jazyk Java

---

Doporučené zdroje informací

DIMARZIO, J. F. Programujeme hry pro Android 4. Brno:Computer press a.s. . 2012, 310str. . ISBN: 978-80-251-3754-3.

GRANT, A. Android 4. Brno: Computer Press a.s. 2013. 656 str. .ISBN: 978-80-251-3782-6.

HERODEK, M. Android jednoduše. Brno: Computer press a.s. 2013. 128str. ISBN: 978-80-251-4118-2.

MEIER, R. Professional Android Application Development. 2. vydání. Indianapolis: Wrox. 2010. 576 str. ISBN 0470565527.

MURPHY, M. L. Android 2, Průvodce programováním mobilních aplikací. Brno: Computer Press, a.s.. 2011. 369 str. ISBN 978-80-251-3194-7.

ROGERS, R., LOMBARDO, J. Android Application Development. 1st Edition. Cambridge: O'Reilly Media, Inc. 2009. 336str. ISBN 978-0-596-52147-9.

VÁVRŮ J., UJBÁNYANI M. Programujeme pro Android. Praha: Grada a.s. 2013. 256 str. ISBN 978-80-247-4863-4.

Předběžný termín obhajoby

2015/16 LS – PEF

Vedoucí práce

Ing. Čestmír Halbich, CSc.

Garantující pracoviště

Katedra informačních technologií

Elektronicky schváleno dne 28. 10. 2015

Ing. Jiří Vaněk, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 10. 11. 2015

Ing. Martin Pelikán, Ph.D.

Děkan

V Praze dne 06. 03. 2016

Čestné prohlášení

Prohlašuji, že svou bakalářskou práci "Aplikace android – teorie a praxe" 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 použitých zdrojů 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 14. 03. 2016

_____

Kalamár Dominik

Poděkování

Rád bych touto cestou poděkoval Ing. Čestmíru Halbichovi, CSc. za odborné vedení, za pomoc a rady při zpracování této práce.

Aplikace android – teorie a praxe

Souhrn

Teoretická část práce je zaměřena na seznámení čtenáře s operačním systémem Android a vývojem aplikací na něj. Dále čtenáře obecně seznamuje s historií a následným průběhem vývoje této platformy ve formě aktualizací. Krátce také pojednává o hlavních konkurentech Androidu, kterými jsou iOS a Windows Phone. V závěru teoretické části, seznamuje práce s oficiálním vývojovým prostředím Android Studio a jednou z jeho alternativ App Inventor 2.

V praktické části je zachycen návrh a následný vývoj dvou ukázkových aplikací, jedna ve vývojovém prostředí Android Studio a druhá ve webovém vývojovém prostředí App Inventor 2. V závěru práce je vyhodnocena obtížnost vytváření aplikace v jednotlivých prostředích a zároveň obsáhlost vývojových možností, které v jednotlivých prostředích máme.

Klíčová slova: Android, Java, Android Studio, Google, programování, iOS, Windows Phone, operační systém, use case, drátěný model, App Inventor, Apple

Android Application – Theory and Practice

Summary

The theoretical part is aimed at acquainting the reader with the Android operating system and application development for it. Furthermore, the readers are generally acquainted with the history and the subsequent course of development of this platform in form of updates. It also shortly discuss the major competitors of Android, which are iOS and Windows Phone. The end of the theoretical part introduces readers to the official Android Studio development environment and one of its alternatives the App Inventor 2.

In the practical part, there is captured design and the subsequent development of two samples of similar applications, on developed in Android Studio and the second one developed in App Inventor 2. In the end of this thesis is evaluated difficulty of creating an application in different development environments, and also the robustness of options, that each of this environments provides.

Keywords: Android, Java, Android Studio, Google, programming, iOS, Windows Phone, operating systém, use case, wireframe, Apple

Obsah

Seznam obrázků

Seznam tabulek

1 Úvod

Android jako operační systém je tu s námi již od konce roku 2008. Vyvíjen společností Google, prošel Android až do současnosti mnoha aktualizacemi, které ho ženou technologicky kupředu mílovými kroky. Android nám slouží jako prostředník mezi uživatelem a hardwarem telefonu, aby uživatel mohl s telefonem jednoduše pracovat. Android byl na jeho počátku vydán pod taktovkou Open Handset Alliance složené z 34 společností z oblasti vývoje mobilního hardware a software. Toto společenství se snažilo vytvořit nový otevřený standard pro mobilní zařízení. Krátce na to, byl oznámen nový operační systém Android, který je následně vydán jako open-source. Právě tento fakt napomohl tomu, že se nastartovalo jeho plošné využití u celé řady výrobců smartphone.

Díky rychle získané oblíbenosti, je o této platformě napsáno nespočet knižních publikací, bohužel z důvodu rychlého vývoje Androidu, ale i samotných informačních technologií se tyto publikace stávají velice brzy zastaralé. Někteří profesionální programátoři dokonce tvrdí, že kniha o Androidu je zastaralá už v den vydání. Proto se autor práce rozhodl čerpat nejenom z knižních publikací, ale také z informací poskytnutých online na internetu, které například ze strany samotného Googlu, jsou velice dobře zpracované a hlavně aktuální.

Autor práce si zvolil téma Android z velké části kvůli osobnímu zájmu o tuto problematiku, hlavně v praktické části, kde si ukážeme jak naprogramovat jednoduchou, ale funkční aplikaci. Dalším důvodem je aktuálnost tohoto tématu, která plyne hlavně z toho, že Android je momentálně nejvíce zastoupený mobilní operační systém na světě a díky neustále rostoucímu zájmu, který je každý rok viditelný, nenasvědčuje nic tomu, že by se to v blízké budoucnosti mělo změnit.

2 Cíl práce a metodika

Cíl práce

Cílem bakalářské práce je charakterizovat problematiku vývoje aplikací pro operační systém Android. Ten se skládá z následujících dílčích cílů: popis architektury platformy Android a vývojových nástrojů pro něj a stručné seznámení s platformou Android a jejími konkurenty. Stěžejním cílem je poté vytvoření aplikace v plnohodnotném vývojovém nástroji Android Studio a vytvoření obdobné aplikace v odlehčeném vývojovém prostředí App Inventor 2, nadále z těchto zkušeností zachytit výhody a nevýhody obou vývojových prostředí.

Metodika

Metodika je založena na studiu a analýze odborných informačních zdrojů. Na základě těchto zdrojů bude sepsána teoretická část práce a vytvořena jednoduchá aplikace. Autorem zvolené zdroje jsou převážně internetové a to hlavně z důvodu rychle se vyvíjejícího prostředí, které se těžce zachytává v tištěné podobě. Tyto zdroje jsou v převážné většině dostupné v anglickém jazyce.

V teoretické části práce je stručně vysvětlena historie platformy a zároveň s tím zachycen průběh vývoje po jednotlivých verzích. Dále je zde představen Google Play, jeho funkce a důvod vzniku. V další části práce je vyobrazeno porovnání Androidu s hlavními konkurenty, kterými jsou iOS a Windows Phone. U konkurenčních systému je zmíněna krátká historie, informace o aktuálním systému a výhody, které nabízí oproti Androidu. Poslední teoretická část této práce zkoumá vývojová prostředí, která byla použita v praktické části pro vývoj aplikace. První, autorem zvolené prostředí je oficiální nástroj Android Studio a jako druhé je zvolen webový nástroj App Inventor 2. U těchto prostředí jsou zmíněny základní stavební prvky, nezbytné pro vývoj aplikace.

Praktická část se zaobírá vytvořením dvou autorem navržených aplikací. Aplikace jsou tvořeny ve dvou rozdílných vývojových prostředích, která byla načrtnuta v teoretické části. První aplikace je tedy vytvořena v prostředí Android Studia za využití programovacího jazyka Java a značkovacího jazyka XML. Druhá aplikace je tvořena v prostředí App In-

ventor 2, za využití logického tvoření funkce bez psaní kódu. Vytváření aplikace je zachyceno od samotného návrhu, přes přípravu vývojového prostředí, až po samotné programování.

3 Android teorie

Když se mezi lidmi řekne slovo android, tak si většina z nás už nepředstaví robota podobného člověku, nýbrž mobilní operační systém vyvíjený společností Google. Tento operační systém běžně nalezneme na většině chytrých zařízení, jako jsou smartphone a tablety, od celé řady výrobců. Za úspěchem Androidu stojí v prvé řadě to, že se jedná o open source software, jinými slovy, že je volně šiřitelný zdarma. Právě díky jeho snadné dostupnosti po něm řada výrobců mobilů sáhla. Dalším neméně důležitým důvodem je, že se jedná o produkt společnosti Google, čili již v základu podporuje všechny jeho služby, jako jsou Gmail, Google vyhledávač, Google mapy, YouTube a další. A jelikož tyto aplikace, celosvětově využívají miliony lidí, tak těm samozřejmě vyhovuje, že po přihlášení přes svůj jeden účet, budou mít k dispozici vše, co používají běžně na stolním počítači i na svém chytrém telefonu.

Díky všem výše zmíněným důvodům, jsme v případě Androidu mohli být svědky opravdu raketového startu, kdy se za pouhé dva roky vyhoupl před všechny ostatní mobilní operační systémy a stal se tak na konci roku 2010 nejprodávanějším systémem na chytrých telefonech a toto prvenství drží až do dnes. To samozřejmě Google motivovalo k neustálé podpoře této platformy, díky čemuž se s každou další aktualizací posouvá Android dopředu velice rychle.

Historie

Ještě před tím než vznikl Android, existovala společnost založená bývalým inženýrem ze společnosti Apple, Andy Rubinem. Tato společnost se jmenovala Danger a jejím hlavním výrobkem byl Hiptop, chytrý telefon, který měl pevnou klávesnici a umožňoval prohlížení webu, emailu a posílání instantních zpráv. Tento telefon byl později (po uzavření partnerství s T-Mobile) přejmenován na Sidekick. Nového telefonu si všimli i lidé z Googlu, kterým se zalíbila myšlenka, že by lidé mohli používat Google vyhledávač kdekoli by se právě nacházeli, pouze za pomoci telefonu. V té době se ale ředitelé společnosti Danger rozhodli, že Andyho Rubina nahradí někým jiným, Rubin ale využil toho, že už jeho jméno bylo známé a založil vlastní společnost, kterou pojmenoval Android, Inc.

Hlavní myšlenkou Android, Inc. bylo vytvořit open-source software pro novou generaci chytrých telefonů. Hlavní cíl, kterého se přitom snažili dosáhnout, byla co nejlepší konektivita k internetu pro bezproblémové využívání všech online funkcí telefonu. Na druhém místě bylo vytvoření takové platformy, která bude přívětivá pro programátory, kteří na něj budou vyvíjet aplikace. Vzniku Android, Inc. a jeho cílů si opět všiml i Google, který v roce 2005 nutně potřeboval více chytrých telefonů, na kterých poběží Google vyhledávač v základu a zároveň vyjáři Google chtěli mít možnost konkurovat Blackberry a Microsoftu. Práce, kterou odváděli v Android, Inc. se jim natolik zalíbila, že Google celou společnost odkoupil a přesunul do Kalifornie, kde pokračovali na vývoji. [1]

Google se mimo vyvíjení operačního systému věnoval také vývoji vlastního chytrého telefonu. První prototyp byl znám pod názvem Sooner. Sooner byl podobný dnešním zařízením Blackberry s pevnou QWERTY klávesnicí. Ovšem tento prototyp nešel nikdy do prodeje, protože těsně před jeho vydáním přišel na trh také první iPhone a vývojářům bylo jasné, že Sooner jako přímá konkurence nikdy neobstojí. Proto ve spolupráci s HTC začali pracovat na novém telefonu, který je znám jako G1 popřípadě mimo Spojené Státy jako HTC Dream. Na rozdíl od společnosti Apple, která měla na prvním iPhone pouze dotykovou obrazovku, sázeli vývojáři s jejich G1 na jistotu a přidali pod vysouvací displej ještě pevnou klávesnici. Při vydání se G1 stal výborným základem pro začínající uživatele Androidu. Zajímavostí je, že G1 byl první chytrý telefon, který přišel na trh s podporou multitaskingu a kopírování.

Hlavním důvodem úspěchu G1 nebyl samotný hardware, který nebyl nikterak ohromující, jak by se dalo očekávat, ani nenabízel v podstatě nic navíc oproti konkurenci. Jeho hlavním tahákem se totiž stal software uvnitř telefonu. G1 byl úplně první telefon s Androidem co šel do prodeje, tehdy se jednalo o verzi 1.0 a právě díky tomu, mohli být cítit některé nedodělky, ale i přes to zde byly věci, které překonávaly konkurenci, a bylo vidět, že Google s Androidem směřuje správným směrem. Jednou z takových věcí jsou například widgety, neboli malé aplikace, které si můžeme připnout na hlavní obrazovku, kde v reálném čase fungují (hodiny, novinky, hudební přehrávač atd.).

Po úspěchu G1 se na open source Android zaměřily i větší společnosti, jako je Samsung nebo Motorola a pomocí Android Open Source Project (dále AOSP) začaly upravovat Android pro své zařízení. Funguje to tak, že jakmile Google dodělá aktualizaci, tak ji přes AOSP veřejně vydá a v tu chvíli si jí může každý stáhnout a upravovat si kód podle sebe pro svá zařízení. Jeden z takto upravených Androidu je například TouchWiz, který je od společnosti Samsung. Pro využití všech funkcí od Googlu ovšem nestačí pouze mít funkční upravený kód, ale je potřeba ještě splnit test kompatibility na základě kterého dostaneme od Googlu certifikát a povolení využívat Google služby jako třeba Google play. Pro funkční prostředí ovšem není tento certifikát nutností, jen nebudeme mít k dispozici výše zmíněné služby. [2]

Vzrůstajícího zájmu o Android si všimla i firma Apple, z které se s úspěchem Androidu stal úhlavní nepřítel Googlu. Apple totiž považoval Android od začátku za pouhou kopii iOS na iPhone. Sám Steve Jobs prohlásil, že udělá všechno pro to, aby zničil Android, protože podle něj, se jedná o kradený produkt. Na to Google reagoval výčtem vlastností, které iPhone neposkytoval oproti Androidu, jako je například výkonnější prohlížeč, Adobe Flash player, multitasking nebo cloudová instalace aplikací. Ve výsledku se ovšem Apple do žádné přímé konfrontace s Googlem nepustil, pouze si začal nárokovat patenty proti výrobcům, kteří Android využívají ve svých zařízeních jako je Samsung a HTC. Bohužel pro Apple už bylo pozdě snažit se zastavit rozjetý vlak, kterým se Android stal. [3]

V roce 2014, Google již absolutně dominoval trhu chytrých telefonů a tabletů a začal uvažovat o pokusu prosadit se s Androidem i v jiných odvětvích. Samozřejmě vidina společnosti Google, že Android bude během pár let jako první i v jiných zařízeních, než jen chytré telefony je hnala kupředu, a tak během dvanácti měsíců představili nové produkty, jako jsou hodinky, brýle Google Glass, televize, vlastní notebooky zvané Chromebook, dokonce i auta a to vše fungující na platformě Android. [4]

Verze Android

V této kapitole krátce představím jednotlivé verze Androidu, tak jak vycházely. Zabývat se začneme až verzí 1.5, protože do této verze se nic moc nedělo, nýbrž se pracovalo pouze

na optimalizaci samotného operačního systému. Můžeme si povšimnout, že jednotlivé verze Androidu jsou pojmenovány podle nějaké sladkosti a tato sladkost je následně použita i jako logo dané verze (Pro více informací o verzích viz https://www.android.com/history).

3.1.1 Verze 1.5, Cupcake

Nejdůležitější funkcí přidanou v této verzi je textová predikce u softwarové klávesnice zároveň s uživatelským slovníkem pro vlastní slova. Dle názoru autora se jedná o velice důležitou funkci i dnes, protože nám šetří čas při psaní jakéhokoliv textu. Další výraznou věcí je možnost nahrávat a zároveň i přehrávat soubory ve formátu MPEG-4 a 3GP. [6]

3.1.2 Verze 1.6, Donut

V době této aktualizace bylo potřeba reagovat na velké množství různě velkých zařízení, proto byla přidána podpora různých obrazovek, kdy se nám obraz automaticky přizpůsobí velikosti zařízení. Další významnou funkcí je rychlý vyhledávač, díky kterému nemusíme při hledání konkrétních věcí prohlížet celý seznam, ale můžeme pouze napsat klíčové slovo do vyhledávače a hledaný objekt se nám zobrazí. [6]

3.1.3 Verze 2.1, Eclair

V této verzi byla přidána důležitá aplikace kterou je Google Maps Navigace. Navigace od začátku obsahovala 3D zobrazení trasy, ovládání hlasem a informace o dopravě. Zmíněné ovládání hlasem bylo přidáno i pro ostatní aplikace jako je psaní sms, emailů, vyhledávání atd. V neposlední řade je přidána možnost rozestavení si ikony aplikací na domovské stránce podle sebe popřípadě je rovnou umístit do složek. [6]

3.1.4 Verze 2.2, Froyo

Zde se vývojářům povedlo ještě o něco zlepšit hlasové ovládání. Nyní se nejedná jen o přepisování hlasu na text, ale můžeme si pomocí hlasu nastavit budík, zapsat poznámky, najít si trasu nebo vyhledávat. Další zajímavou novinkou je možnost udělat si z telefonu hotspot na Wi-Fi, to znamená, že můžeme náš telefon využít jako router a připojit si například notebook na internet. [6]

3.1.5 Verze 2.3, Gingerbread

Verze 2.3 do Androidu přidává funkci managementu baterie, díky kterému vidíme, co nám nejvíc spotřebovává baterii, graf spotřeby baterie od posledního nabití a odhad výdrže baterie. Další novinkou je podpora NFC, jedná se o druh bezdrátového přenosu dat, který funguje jen na velmi krátkou vzdálenost, proto je potřeba zařízení mezi kterými probíhá přenos k sobě přiložit. Ovšem výhodou NFC je vysoká rychlost přenosu, možnost využít i pro bezkontaktní čipy jako jsou například u platebních karet (zatím otázka blízké budoucnosti jestli se prosadí). [6]

3.1.6 Verze 3.0, Honeycomb

Zvláštností verze 3.0 je, že není určena na mobily, ale pouze na tablety. Od této logiky ovšem hned vzápětí upustili a začali vydávat aktualizace pro obě zařízení zároveň. Nicméně vztyčnými prvky této verze je úprava vzhledu na tablety, s lepším rozložením prvků pro velké obrazovky. Další důležitou novinkou je přidání systémové lišty, která se nachází ve spodní části a umožňuje nám základní ovládání bez potřeby fyzických tlačítek. [6]

3.1.7 Verze 4.0, Ice Cream Sandwich

Změny v této verzi jsou spíše kosmetické, jedná se především o předělaný vzhled složek na naší domácí obrazovce, dále možnost zobrazit si v grafu využití mobilních dat za určité období, možnost nastavit si limity, popřípadě upozornění a možnost úplně vypnout data při překročení limitu. Poslední novinkou, kterou zmíníme je Android Beam, který nám díky NFC umožňuje sdílet pomocí jednoho kliknutí obrázky, kontakty, videa a aplikace mezi zařízeními. [6]

3.1.8 Verze 5.0, Lollipop

Hlavním tahákem této verze bylo propojení a synchronizování všech našich zařízení. Důvod toho, že takováto synchronizace nebyla potřeba dříve, je že Google se se svým Androidem začal prosazovat i v zařízeních jako jsou chytré hodinky, televize nebo počítače v automobilech. S touto synchronizací můžeme v tom, co děláme plynule pokračovat na jiném zařízení, například při pouštění muziky na mobilu můžeme přejít do auta a tam nám začne písnička pokračovat, kde jsme skončili (samozřejmě musí obě zařízení tuto synchronizaci podporovat, respektive běžet na operačním systému Android).

Dalším důležitý bodem jsou Googlem vytvořené tzv. Material Designs, což jsou v podstatě návody jak navrhovat aplikace, aby byly pohodlně ovladatelné pro většinu uživatelů. Tyto návody se zakládají na dlouholetých zkušenostech a zpětné vazbě prováděné s uživateli. [6]

3.1.9 Verze 6.0, Marshmallow

V době psaní této práce se jedná o nejnovější aktualizaci, která je zatím pouze na 0,5 procentech zařízení. Je zde přepracován model povolených akcí pro aplikace jako je využití mikrofonu, kamery, internetu atd., kdy tato povolení můžeme spravovat i po nainstalování aplikace namísto pouze jednorázového povolení při instalaci a tím zvýšit bezpečnost pro uživatele. Další znatelnou úpravou je lepší přístup k baterii telefonu pomocí vlastnosti Doze, která nám nevyužívaný telefon uspí a pouze jednou za určitou periodu zkontroluje žádosti aplikací, ty splní a znovu se uspí, tím se šetří na životnosti baterie. [6]

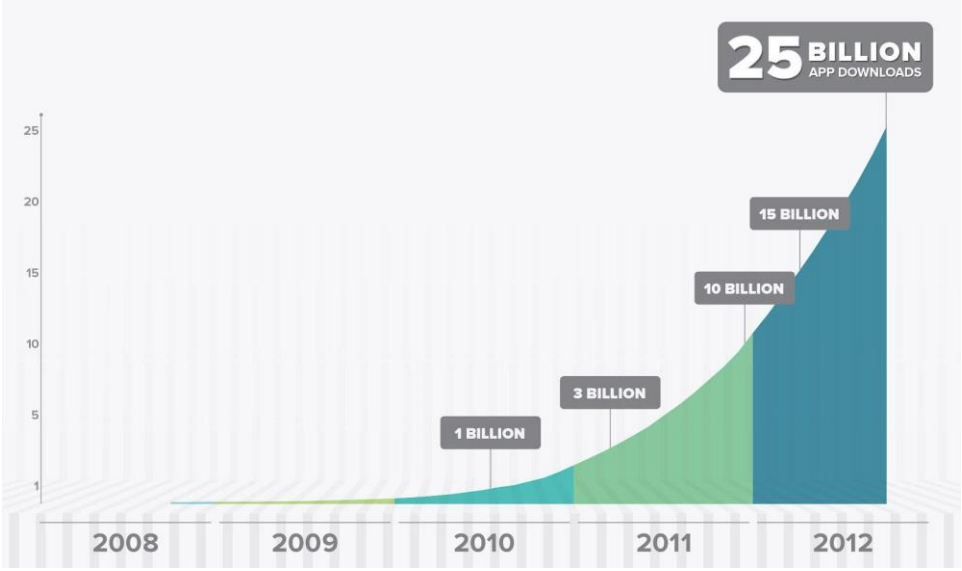
Google play

Jedním ze základních stavebních kamenů Androidu je Google play. Jedná se o aplikaci, která je součástí všech zařízení s operačním systémem Android. Do této aplikace se přes Google účet přihlásíme a tím získáme přístup k obrovskému množství aplikací, muziky, filmů a e-knih, kterými si můžeme obohatit náš telefon. Aplikace máme přehledně rozdělené do kategorií, podle kterých můžeme vyhledávat od nejrůznějších her, přes desktopové aplikace typu Skype až po jen malé widgety které nám poskytují lepší virtuální klávesnici, hodiny, animované pozadí a mnoho dalšího.

Je také potřeba zmínit, že ne všechny aplikace dostupné na trhu Google play jsou zdarma. Tyto částky jsou velmi často téměř symbolické a navíc máme možnost do určité doby aplikaci vrátit za plnou cenu, čili se nemusíme bát, že pokud nebude aplikace pracovat správně na našem zařízení tak jsme vyhodili peníze (to zároveň nutí vývojáře věnovat svým aplikacím maximální péči, co se týče optimalizace). Nově získaná aplikace ať už zdarma nebo placená se automaticky stáhne a nainstaluje rovnou do našeho zařízení.

Část úspěchu Androidu také stojí na vzniku Google play, jelikož možnost stáhnout si pouze aplikace, které opravdu potřebujeme je uživatelsky přívětivá. To ale nenahrává

pouze Googlu, ale také vývojářům kteří se mohou v tomto odvětví prosadit a vydělat nemalé peníze programováním aplikací na Google play, ať už za nějakou cenu nebo za počet stažení a zobrazování reklam v aplikaci. Zvýšený zájem o programování pro Android je znatelný na počtu aplikací k dispozici, kdy podle serveru androidtip.cz jich bylo při vzniku v roce 2009 pouhých 2300, roku 2010 již 100 000 a na konci roku 2013 již rovný milion aplikací. Na následujícím obrázku je zobrazeno, kolik aplikací bylo za období mezi rokem 2008 a rokem 2012 staženo. Podle statistik, růst počtu stažených aplikací stále roste, v roce 2013 se zvedl ještě o dvojnásobek na rovných 50 bilionů. [5]



Obrázek 1 - Počet stažených aplikací za roky 2008-2012 (Zdroj: http://www.androidtip.cz/android-marketu-az-google-play-pohled-historie-nejvetsiho-online-obchodu-aplikacemi-android/#prettyPhoto/2/)

3.1.10 Vznik Google play

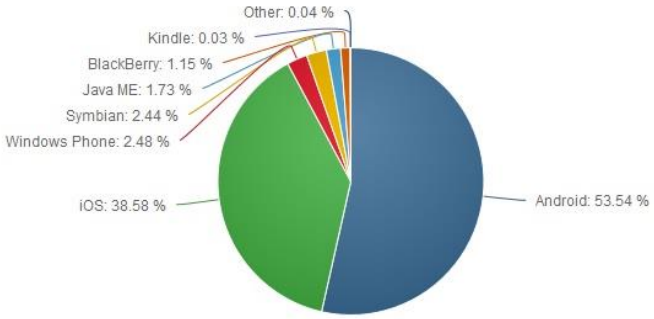
Při uvedení prvního telefonu s Androidem v roce 2008 ještě neexistoval Google play, ale pouze jeho předchůdce s názvem Google market. Ten nám sice umožňoval stáhnout si dostupné aplikace, jejichž počet s každým rokem přibývá po statisících, ale nebyl moc uživatelsky přívětivý. Chyběla zde i možnost instalovat aplikace přes internetový prohlížeč.

Dalším výrazným problémem bylo, že Android market nepokrýval všechna odvětví, ale byly zde navíc ještě aplikace na muziku, Google Music a na e-knihy, Google eBookstore. To mělo špatný vliv na zákazníky, kteří se mohli cítit zmatení, že není vše na jednom mís-

tě. Proto se Google v roce 2012 rozhodl všechny tyto platformy spojit v jednu a takto sjednocená platforma je pojmenovaná právě Google Play. [5]

Porovnání s konkurencí

V následujících kapitolách si představíme dva hlavní konkurenty Androidu. Jedná se o iOS od Apple a Windows Phone od Microsoftu. Pokud jednotlivé operační systémy porovnáme podle podílu na trhu, tak je zřejmé, jak si kdo v prodejích stojí. Podle webu svetandroida.cz byl k 4. 10. 2015 podíl na trhu pro Windows Phone pouhých 2,48 %, z toho lze vyvodit, že jak Apple tak ani Google nikterak moc neohrožuje. Zato podíl mezi Androidem a iOS už není takto markantní a jedná se o 38,58 % na straně iOS a 53,54 % na straně Androidu.



Obrázek 2 - Podíl mobilních OS na trhu (zdroj: http://www.svetandroida.cz/android-vs-apple-obsazeni-201510)

I zde by se tedy mohlo zdát, že si Android oproti konkurenci stojí výborně, nicméně je potřeba vzít v potaz rozdílnou cenovou politiku obou výrobců. Apple totiž na rozdíl od Androidu nenabízí obří množství zařízení pro každou cenovou kategorii, nýbrž vždy pouze jeden model telefonu v různých lehce upravených variantách s cenou rovnající se high-end Android telefonům. Z toho již můžeme vyvodit, že ačkoliv má Apple celkový prodej zařízení nižší, tak jeho reálné výnosy jsou naopak o dost vyšší než výnosy z prodeje Android zařízení. [7]

Pro lepší představu si uvedeme konkrétnější čísla. Podle serveru mobil.idnes.cz byl za čtvrtletí roku 2014 celosvětový zisk z prodeje chytrých telefonů 518 miliard korun a z toho si jen Apple uzmul 88,7 % neboli kolem 460 miliard korun. A Android se s ostatními výrobci dělí o zbývajících 11,3 % neboli něco kolem 58 miliard korun. Zde je vidět, že z hlediska zisku se jedná o propastný rozdíl. Viz Tabulka 1 (z tabulky byly odstraněny ostatní operační systémy z důvodu zanedbatelnosti hodnot). [8]

Operační systém	Provozní zisk Q4 2013 (miliardy dolarů)	Provozní zisk Q4 2014 (miliardy dolarů)	Podíl na provozním zisku Q4 2013	Podíl na provozním zisku Q4 2014
iOS	11,4	18,8	70,5 %	88,7 %
Android	4,8	2,4	29,5 %	11,3 %

Tabulka 1 - provozní zisk z prodejů smartphonů (Zdroj: http://mobil.idnes.cz/apple-ma-z-ios-vice-nez-googlu-z-androidu-fdq-/mob_tech.aspx?c=A150226_155702_mob_tech_LHR)

3.1.11 Apple iOS

Apple Inc., (dříve Apple Computer, Inc.) je obchodní korporace, kterou založil Steve Jobs a Steve Wozniak roku 1976. Začínali výrobou PC, které se jmenovali Macintosh, později se ale uchytil zkrácený název Mac, který se drží až do dnes. Změna přišla až v roce 2001, kdy vydali úspěšný hudební přehrávač iPod a o tři roky později iTunes Music Store. Od této chvíle se z Applu stal hlavní výrobce spotřební elektroniky, která si našla mnoho příznivců po celém světě, to vedlo také k tomu, že odstranili slovo „Computer“ z názvu společnosti. Aktuálně jsou hlavními produkty Applu chytré telefony iPhone, tablety iPad, hudební přehrávače iPod a počítače Mac. [9][10]

3.1.11.1 iOS

Povědomí o nějakém novém systému iOS (tehdy znám pod názvem iPhone OS) začíná prezentací Steva Jobse, kde se představí nový chytrý telefon s názvem iPhone. Na rozdíl od Androidu, který při vydání předčil svojí funkčností konkurenci, tak iPhone potažmo iOS byl za konkurencí, co se týče funkcí operačního systému. iPhone nepodporoval multitasking, 3G, aplikace třetích stran, nebylo zde možné kopírovat a vkládat, nepodporoval posílání MMS ani zde nebylo možné upravit si domácí obrazovku podle sebe. Všechny tyto nedostatky byly veřejně známé a i přesto se iPhone stal velice úspěšným a modlou

velké skupiny lidí po celém světě, stalo se tomu z důvodu, že Apple se nesnažil předčít konkurenci lepšími funkcemi, ale kladl důraz na co nejlepší možný uživatelský zážitek z používání telefonu. Zaměřil se hlavně na rychlost systému, skvěle odladěné aplikace a bezpečnost. [3][11]

3.1.11.2 Výhody iOS

Apple těží hlavně z toho, že jeho iOS není otevřená platforma, to znamená že iOS běží pouze na Apple zařízeních, u kterých vědí, jaké bude mít specifikace. Díky tomu jsou schopni maximálně iOS optimalizovat a poskytnout uživateli nejlepší možný zážitek z používání telefonu. Tato výhoda platí i pro programátory, kteří tvoří aplikace na App Store, protože vědí, na jaké konkrétní zařízení aplikaci tvoří.

Další výhodou je bezpečnost, která je ohrožena nejčastěji přes aplikace třetích stran. Do iOS je možné tyto aplikace oficiálně pořídit pouze přes App Store, na který Apple důkladně dohlíží a každou aplikaci než jí umožní zveřejnění, překontrolují, zdali neobsahuje škodlivý kód, nevhodný obsah a zda aplikace běží plynule. Díky tomu jsou aplikace v App Store mnohem bezpečnější a více uživatelsky přívětivé než aplikace v Google Play.

K bezpečnosti také přispívá to, že iOS neumožňuje uživateli tak vysokou možnost nastavení a přizpůsobení a tím eliminuje riziko, že uživatel něco nastaví špatně a tím usnadní přístup útočníkovy. [11]

3.1.12 Microsoft Windows Phone

Microsoft je americká společnost, která se věnuje vývoji počítačového software, spotřební elektroniky a počítačů. Microsoft měl dva zakladatele, kterými jsou Bill Gates a Paul Allen, kteří díky operačnímu systému MS-DOS, začínali protlačovat svojí společnost do povědomí lidí. Po úspěchu MS-DOS vzrostly akcie Microsoftu raketově a dala za vznik té dnes známé obří společnosti. Nejznámějším produktem se stal Microsoft Windows, z kterého je až do dnes nejpoužívanější operační systém v počítačích na světě.

Operační systém Windows Mobile patří mezi jedny z nejstarších, kdy první verze byla představena již v roce 2000 a nesla název Pocket PC 2000. Vzhledem velice připomínal

desktopové Windows 98 a 2000. Výhodou tohoto systému bylo, že již od začátku obsahoval Microsoft Office aplikace (Excel, Word). Největší podíl se smartphony držel Microsoft v roce 2004 s 23 %, od této doby ale s příchodem iOS a Androidu šel podíl rapidně dolů. Od této doby sice vycházely stále nové revize tohoto systému, nicméně jejich podíl na trhu stále klesal. Microsoft až v roce 2010 uvědomil, že pokud se nepokusí o razantní změnu tak hrozí mobilní divizi krach. [12]

3.1.12.1 Windows Phone

V roce 2010 Microsoft představil na Mobile World Conference 2010 nově vytvořený mobilní operační systém nazvaný Windows Phone. Tento operační systém byl jedinečný jeho designem, který byl inspirován desktopovou verzí Windows 8 respektive designem Metro. Operační systém je tedy složen z jednotlivých dlaždic, které si sami libovolně přestavujeme. Odlišnost od předchozích systému je možné vidět i na tom, že Marketplace (aplikace pro nakupování aplikací) musel být rozdělen na dva, jeden pro starší OS a jeden pro nový Windows Phone. Výhodou nového Marketplace je možnost u všech aplikací stáhnout si nejdříve zkušební demo a až poté si můžeme aplikaci koupit, pokud se nám líbí.

Windows Phone se dočkal příjemného přijetí uživateli a očekávalo se, že dožene konkurenci, nicméně tomu bránily dvě věci a těmi je samotný Microsoft a vývojáři. Jedná se o to, že tato nová platforma byla uzamknutá sama do sebe podobně jako iPhone. Bohužel na rozdíl od iPhone v době Windows Phone existuje již obrovská konkurence a platforma bez velkého výběru aplikací nemůže mít úspěch a na druhou stranu na platformu, která nemá úspěch, nebude nikdo tvořit aplikace.[13][14]

3.1.12.2 Výhody Windows Phone

Výhodou Windows Phone je podobně jako u Apple, že každá aplikace než je vpuštěna na trh, je zkontrolována techniky Microsoftu, jestli není škodlivá, popřípadě nezpůsobuje některé chyby. Další výhodou může být uživatelské rozhraní Metro, na které někteří uživatele mohou být zvyklí z desktopu, a proto ho upřednostní před rozhraním jiným. Poslední výhodou, kterou mohu zmínit je bezproblémový běh systému, který je rychlý stejně jako spouštění systémových programů. [15][16]

Android Studio

Android Studio je oficiální vývojové studio pro vytváření aplikací na Android, které je vyvíjeno firmou Google. Jedná se o jakousi nadstavbu vývojového prostředí IntelliJ IDEA, z kterého si bere kladné vlastnosti, co se týče práce s kódem jako je refactoring nebo našeptávání.

Pomocí tohoto vývojového studia a znalosti jazyka Java a částečně i xml, můžeme tvořit aplikace pro Android. Tyto aplikace následně můžeme debugovat s možností ukázky grafů využití paměti a procesoru, což nám umožňuje aplikace optimalizovat tak, aby využívala co nejmíň dostupných prostředků mobilního zařízení. Další funkcí, kterou nám Android Studio poskytuje, je možnost logování, kdy si zapíšeme do našeho kódu log, a v okamžiku, kdy se k logu kód dostane, vypíše se nám jeho obsah v informačním okně. Tím můžeme ověřovat, zda se program do určité části kódu dostane, a když ano, tak jaké jsou v tu chvíli hodnoty proměnných.

Dále je zajímavý například editor překladů, pokud v naší aplikaci podporujeme více jazyků, tak máme možnost na jedné obrazovce spravovat všechny texty a vidět jak budou přeložené v různých jazycích.

Pro začátečníky, kteří začínají s programování, nabízí Google možnost stažení ukázkových kódů z cloudového úložiště GitHub. Díky tomu si můžeme nahrát do našeho studia některý z dostupný ukázkových kódu a podívat se, jak by se některé funkce měli správně programovat. [17][18]

3.1.13 String Resources

Jedná se o soubor xml, který nám uchovává informace o všech datech typu string v našem projektu. Na tento soubor se poté pouze odkazuje místo toho abychom „natvrdo“ zapisovali text do našeho programu. Na zdrojový string se poté odkazujeme pomocí klíčového slova @string/ doplněný o název stringu. Hlavním prvkem v tomto souboru musí být vždy element <resources> do kterého pak vkládáme text pomocí elementu <string>. Soubor, který pro uchovávání zdrojů se nazývá strings.xml a je uložen ve složce values.

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string
        name="string_name"
        >text_string</string>
</resources>
```

Obrázek 3 - Ukázka zápisu zdroje stringů (zdroj: http://developer.android.com/guide/topics/resources/string-resource.html)

Otázka je, proč vlastně takovéto zdroje stringů využívat, když můžeme všechny texty psát rovnou bez zbytečného odkazování se jinam. Prvním důvodem je to, že pokud nějaký String využíváme v našem programu vícekrát, tak ho za prvé nemusíme stále přepisovat a za druhé pokud v něm nalezneme chybu, tak nám jí stačí opravit pouze jednou ve zdrojovém souboru, namísto hledání, kde všude se daný String nachází. Dalším a podle mě ještě důležitějším důvodem je jednoduchá možnost lokalizace našeho programu a to z důvodu toho, že souborů strings.xml můžeme mít více a proto pro více jazykových verzí pouze přeložíme celý předchozí strings.xml do jiného jazyka. Tyto soubory se pak odlišují složkami, ve kterých jsou uložené, a to pomocí koncovky, která značí pro jaký jazyk je překlad určen. Například pro angličtinu by byl soubor strings.xml uložen ve složce values-en a pro češtinu values-cz. U takto vytvořené lokalizace je také výhoda v tom, že sám Android si podle nastaveného jazyku v daném zařízení vybere i jazykový balík, pokud je daný jazyk dostupný. [19]

3.1.14 Layout

Layoutem rozumíme návrh ovládacích prvků na naší obrazovce. Při vytváření je vhodné držet se návrhu, pokud je nějaký k dispozici. Máme zde dvě možnosti, jak docílit požadovaného rozvržení. Jednodušší možností je přetahování prvků z levé části obrazovky na plochu telefonu pouze použitím myši, s tím že pro každý prvek můžeme nastavovat dodatečné vlastnosti v pravé části obrazovky. Výhodou této metody je hlavně její jednoduchost, díky které si ji během chvíle osvojí téměř každý. Pro zkušenější programátory je zde druhá možnost, kterou je připravení návrhu pomocí xml, zde je ovšem potřeba mít základní

znalost tohoto jazyka. Mezi těmito dvěma metodami se přepínáme pomocí záložek ve spodní části obrazovky s nápisy design a text. [20]

3.1.15 XML návrh layoutu

Extensible markup language zkráceně jen XML, je značkovací jazyk používaný v Androidu pro tvorbu layoutů. Tento jazyk je založen na tvoření značek neboli elementů, kterým pomocí atributů přiřazujeme nějaké vlastnosti. Elementy jsou uvozeny ve špičatých závorkách, a každý element musí být také uzavřen a to lomítkem na konci a uzavřením špičaté závorky, nebo znovu zapsáním elementu s lomítkem před názvem. Atributy se potom nastavují v rámci jednoho elementu a to vždy ve tvaru android:název_atributu.

Pro efektivní využití tohoto jazyka je potřeba osvojit si základní klíčová slova, které nám umožní tvořit návrhy. Základním elementem celého xml je lineární nebo relativní layout, který nám rozlišuje, jak bude naše rozvržení vypadat. V případě lineárního se budou všechny prvky automaticky skládat pod sebe nebo vedle sebe, což můžeme určit pomocí atributu android:orientation. Lineárního layoutu využije zejména u formulářů, kde není potřeba prvky skládat na obrazovce, nýbrž je vhodné je mít všechny zarovnané pod sebou. Naopak relativní layout funguje na principu relativních pozic jednotlivých elementů. To znamená, že pozice každého elementu je specifikována relativně ke sdruženým elementům, například jako nalevo, pod, nad, napravo od tlačítka, nebo relativně k rodiči (relativní nebo lineární layout) to znamená například, levá část obrazovky, spodní část obrazovky atd. [21][22]

Podobně jako u String resources máme možnost vytvořit více jazykových verzích i zde máme možnost vytvořit si více layoutů pro různě velké displeje, což přijde vhod při optimalizaci pro obrovské množství telefonů a tabletů používajících Android jako operační systém. Kromě rozlišení můžeme připravit i více layoutů také pro orientaci zařízení, tedy jestli je vodorovně či svisle (v anglické terminologii land a port).

Hlavní složka pro naše xml je nazvána layout, v té se také nachází výchozí layout. Pokud chceme přidat nový layout pro jiné rozlišení nebo orientaci, musíme si vytvořit znovu složku layout a za pomlčku dopsat klíčové slovo, které bude značit pro jakou velikost daný layout je. Díky tomu je android sám schopen automaticky vybrat pro konkrétní rozlišení používaného zařízení nejlepší dostupný layout (více informací o klíčových slovech na http://developer.android.com/guide/practices/screens_support.html#overview). [20]

3.1.16 Android Manifest

Android manifest je xml soubor, který vyžaduje každá android aplikace, protože nese důležité informace o naší aplikaci, které systém potřebuje pro to, aby mohl kód správně spustit. Součástí manifestu jsou elementy, které v sobě mohou obsahovat další informace v podobě atributů. Elementy a atributy k nim, které můžeme do manifestu přidat, jsou pevně dané a nelze přidávat žádné vlastní (více informací o povolených elementech na http://developer.android.com/guide/topics/manifest/manifest-intro.html). Naopak jsou zde i povinné elementy, které nikdy nesmí chybět, mezi ně patří element <manifest> a ten musí obsahovat ještě element <application>.

Nejčastější setkání s manifestem budeme mít hlavně kvůli tzv. permissions neboli povolením. Ty slouží k ochraně uživatele tím, že nedovolují použít rizikový kód bez toho, abychom si vyžádali povolení. Jedná se o to, že pokud chceme využívat některé funkce telefonu, které by mohly poškodit uživatele, jako je např. přístup ke kameře telefonu, nebo připojení k internetu tak to musíme v manifestu specifikovat, abychom daný kód mohli využít. To se dělá právě pomocí elementu <permission>, do kterého zaznamenáme, jaké povolení požadujeme. Tato požadovaná povolení potom uživatel vidí při instalaci aplikace z Google play a sám se může rozhodnout, jestli je pro danou aplikaci povolí. [23]

3.1.17 Java

Java je objektově orientovaný programovací jazyk od společnosti Sun Microsystems. Tento programovací jazyk je nedílnou součástí tvoření funkční části aplikace v Android studiu, proto se bez javy neobejdeme. Výhodou programování v Javě jako takové je, že aplikace je přenosná mezi operačními systémy a to z důvodu, že se nejprve vytvoří bytecode a ten se

pak pomocí Java virual machine teprve překládá na daném operačním systému (tato výhoda není až tak využitelná při programování pro Android).

Soubory, do kterých budeme psát kód javy, jsou uložené ve složce java a název balíku naší aplikace, zde všechny soubory ve kterých budeme programovat, mají koncovku .java. S prvním vytvořením aplikace se nám také vytvoří MainActivity.java, jedná se o soubor, který slouží jako první stránka aplikace, to znamená, že při spuštění se nám jako první načte právě MainActivity.java. [24]

App Inventor 2

App Inventor je webová aplikace, původně vytvořená Googlem, která nám umožňuje vytvářet aplikace pro Android. Hlavní výhodou App Inventoru je to, že abychom byli schopni vytvořit nějakou funkční aplikaci, tak v podstatě nepotřebujeme umět programovat, respektive znát nějaký programovací jazyk, nýbrž postačuje základní povědomí o tom, jak programování funguje a jak fungují některé základní funkce typu if a for. Na druhou stranu neposkytuje takové možnosti jako klasická vývojová prostředí. Díky tomu je vhodný pro nováčky v programování, kteří chtějí začít vytvářet aplikace.

K provozu App Inventoru nám postačuje libovolný internetový prohlížeč a Google účet. Pokud tyto požadavky splňujeme, jsme připravení vytvářet aplikaci a to pomocí dvou pracovních ploch, mezi kterými se můžeme přepínat pomocí dvou tlačítka vpravo nahoře pojmenovaných designer a blocks. Na tyto dvě obrazovky se nyní blíže podíváme.

3.1.18 App Inventor Designer

Na této obrazovce nám App Inventor umožňuje vytvořit design, jaký bude mít naše aplikace. Podobně jako při návrhu layoutu v Android Studiu i zde prvky přesouváme pomocí metody drag-and-drop na naší obrazovku, ale možnost vytvořit si design pomocí xml zde bohužel chybí. Dále máme možnost nastavit každému z vytvořených prvků některé základní atributy, jako jsou horizontální a vertikální zarovnání, barva, velikost, popisek atd. Pro různé prvky mohou být některé atributy jedinečné. Další důležitou částí v oblasti přidávání prvků je správné pojmenovávání, primárně kvůli tomu že s nimi v kódu dále pracu-

jeme a je potřeba jednotlivé prvky od sebe odlišit, o to víc pokud se jedná o nějaký rozsáhlejší projekt.

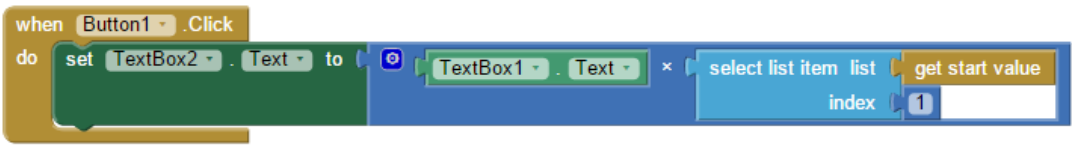
Mimo prvků, které ve finální podobě uvidíme na obrazovce, se v části designer dají přidat ještě tzv. neviditelné komponenty, které nám umožňují později využít jejich funkčnost, nicméně pro uživatele nejsou vidět.

3.1.19 App Inventor Blocks Editor

Tato obrazovka nám umožňuje již samotné naprogramování aplikace. Naprogramováním ovšem nemáme na mysli psaní kódu, nýbrž skládání dohromady takzvaných bloků, které mají sami o sobě nějakou funkci, a pomocí řazení za sebe víc takových bloků tvoříme program.

Základní vestavěné bloky vybíráme z dostupného listu, kde jsou seřazeny podle funkce: Ovládací, logické, matematické, textové, listy, barvy, proměnné a procedury. Dále jsou zde bloky aktivních prvků, které nám slouží k tomu, abychom s těmito prvky mohli dále pracovat.

Bloky nám pomocí volných pozic ukazují, z jaké strany, popřípadě zda jde do nich vložit nějaký prvek, tento princip si můžeme představit podobně jako skládání puzzle, kdy dílky, které nemají otvor na správné straně, k sobě nepůjdou dát. App Inventor nám toto ještě ulehčuje, protože nám umožňuje spojit do sebe pouze takové prvky, které jsou potenciálně funkční, a tím ještě snižuje riziko chyby.



Obrázek 4 - Ukázka bloků (zdroj: vlastní)

4 Vlastní práce

Na základě poznatků získaných z teoretické části této práce, bude nyní vytvořena testovací aplikace. Tato aplikace bude vytvořena ve dvou verzích, první verze bude tvořena ve vývojovém prostředí Android Studio a verze druhá bude obdobná, ale vytvořena ve webovém vývojovém prostředí App Inventor 2. Hlavními faktory, které budu sledovat při tvorbě obou verzí aplikace je náročnost, doba a nabízené možnosti tvorby aplikace.

Aplikace Kurzovní lístek

Název aplikace autor stanovil jako Kurzovní lístek, přičemž hlavní myšlenkou je, aby uživatel mohl zvolit libovolný dostupný měnový kurz, který se následně na pozadí přes internet stáhne a zobrazí uživateli. Autorem zvolený typ souboru pro stažení je JSON a to z důvodu, že nám všechny potřebné informace poskytuje v malém množství dat, což je důležité hlavně v případě, že uživatel není připojen na Wi-Fi, ale na mobilních datech. Autorem zvolený server pro stažení JSON dat je dostupný na stránce fixer.io. Po zobrazení stažených dat je uživateli umožněno vybrat si z tohoto listu jeden další kurz a následně využít přepočtové kalkulačky, kde si daný kurz může přepočíst pro libovolnou částku.

Pro vytvoření této aplikace budeme potřebovat tři obrazovky, mezi kterými bude po jednotlivých krocích uživatel postupovat. Jedna obrazovka bude sloužit pro zobrazení dostupných měn, druhá obrazovka nám stáhne a zobrazí aktuální data a třetí obrazovka poslouží jako kalkulátor jiných částek.

4.1.1 Návrh aplikace

Návrhu aplikace respektive její specifikaci jsem věnoval podstatnou část mého projektu a to hlavně z důvodu abych předešel pozdějším potřebným úpravám a měl konkrétní vizi při vlastním programování. Specifikace, kterou jsem vytvořil, se skládá ze čtyř částí, kterými jsou: cíle, případy užití (tzv. use case), scénář a drátěný nebo také logický model.

4.1.1.1 Cíle

V této části je důležité si stanovit, co vlastně chceme, aby naše aplikace dělala. To znamená, stanovíme si hlavní cíl, v našem případě: Vytvořit aplikaci, ve které uživatel pomocí jednoduchého vyhledání měny bude schopen zobrazit kurzovní lístek dané měny a následně využít přepočtovou kalkulačku.

4.1.1.2 Use case

Use case, česky také používán název případ užití, je druhou částí našeho návrhu aplikace. Jedná se o list kroků, které se provádí při interakci mezi uživatelem a a systémem. Tyto kroky tvoříme opět pro každou obrazovku naší aplikace, aby bylo jasné, jak bude probíhat interakce se systémem. Nepoužíváme zde žádné konkrétní prvky, nýbrž pouze co uživatel očekává od systému. Autorem vytvořené use case jsou následující:

Hlavní strana

- Uživatel očekává možnost vybrání požadované měny
- Uživatel očekává možnost vyhledat požadovanou měnu
- Uživatel očekává, že po vybrání se mu zobrazí kurzovní lístek dané měny

Kurzovní lístek

- Uživatel očekává, že se mu zobrazí kurzy k vybrané měně
- Uživatel očekává možnost přepočítat si kurz pro vybranou měnu

Porovnání měn

- Uživatel očekává, že se mu zobrazí detail dvou měn
- Uživatel očekává možnost přepočítat první měnu na druhou
- Uživatel očekává možnost zobrazení historie přepočtů
- Uživatel očekává možnost vrátit se zpět na kurzovní lístek

Očekává se, že čím rozsáhlejší projekt budeme tvořit, tím obsáhlejší use case budou. Je vhodné si před začátkem tvoření use case a vůbec celého návrhu vytvořit takzvané persony, jedná se o námi připravené fiktivní profily osob, které budou naší aplikaci používat. Dá se očekávat, že 17letý středoškolský student bude využívat naší aplikaci pro jiné účely a jinak než 30letá maminka dvou dětí. Díky tomu můžeme naši aplikaci navrhnout vhodně

s ohledem na potřeby konkrétních lidí. Autor nicméně kvůli jedoduchosti aplikace persony nevytvářel.

4.1.1.3 Scénář

Scénář se při použití v návrhu případů užití využívá zejména k popsání funkční logiky jednotlivých obrazovek aplikace. Jinými slovy, jedná se o instance konkrétních akcí, které mohou v daném případě nastat. Záměr scénáře je stručné a hlavně srozumitelné popsání problému a jeho řešení, nejčastěji pro programátora. Scénáře vytvořené v mém návrhu jsou následující:

Hlavní strana

- Uživatel vybere požadovanou měnu
  - Vyhledání měny podle názvu
  - Vybrání měny ze seznamu měn
- Po vybrání měny se otevře obrazovka „kurzovní lístek“

Kurzovní lístek

- TextView zobrazí, jaká byla vybrána měna
- Podle vybrané měny se stáhne příslušný kurzovní lístek
- V ListView se zobrazí kompletní kurzovní lístek
- Po vybrání druhé měny ze seznamu se otevře obrazovka „porovnání měn“

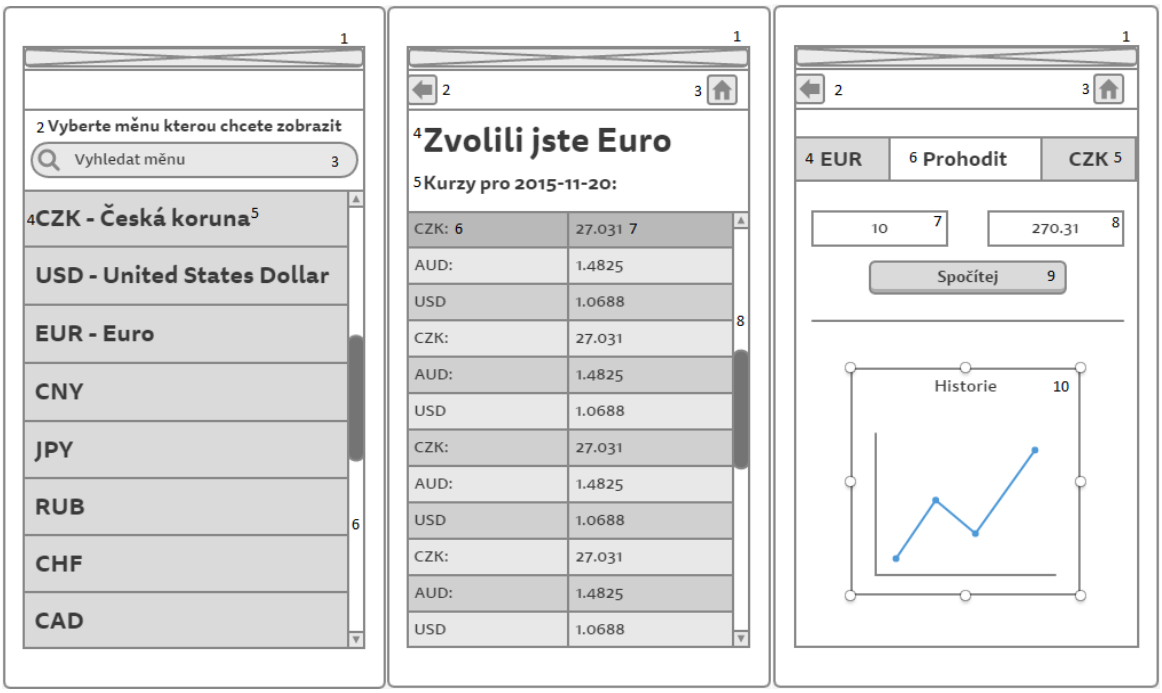
Porovnání měn

- TextView zobrazí zkratku první a druhé vybrané měny
- Po editaci prvního pole a zmáčknutí tlačítka se přepočítá měna pro zadanou hodnotu
- Po přepočtu se daný přepočet započte do historie přepočtů

4.1.1.4 Drátěný model

Poslední, ale neméně důležitou částí mého návrhu je drátěný model. Tento model se tvoří proto, aby bylo jasné, jak bude aplikace přibližně vypadat a fungovat. Jedná se o černobílý návrh bez grafických prvků, který nám v podstatě ukazuje rozmístění logický prvků a provázanost mezi nimi. Grafické prvky zařazuje až grafik právě na základě hotové-

ho drátěného modelu. Tato fáze pak umožňuje programátorovi lépe se orientovat v tom, co je vlastně potřeba naprogramovat. Drátěný model v návrhu autora vypadá následovně:



Obrázek 5 - Drátěný model aplikace (zdroj: vlastní)

Aplikace v Android Studiu

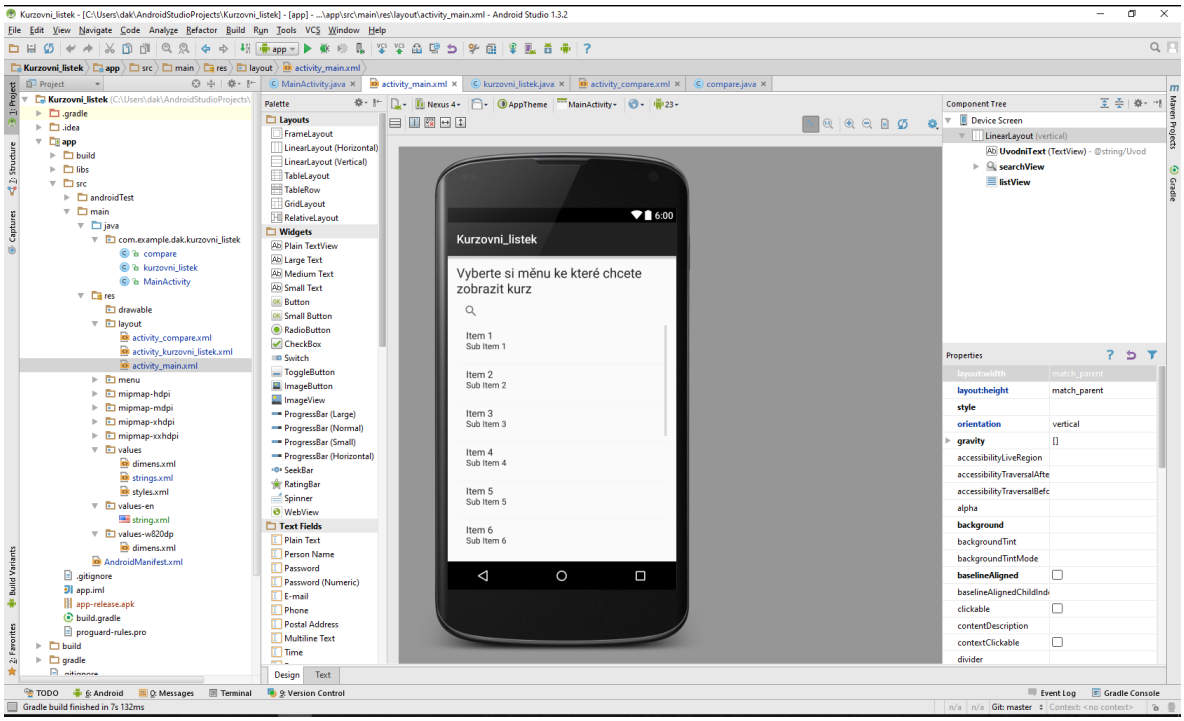
Vývojové prostředí pro programování první verze aplikace kurzovní lístek jsem vybral Android Studio a to hlavně ze dvou důvodů, za prvé, protože se jedná o oficiální vývojářský nástroj pro Android a druhým důvodem je, že se jedná o nově vytvořené studio firmou Google, která si ho může dovolit rozvíjet velice rychle a tím pádem už dle mého názoru předčilo vývojové prostředí, které se používali před tím jako je například Eclipse. Názor autora je, že vzhledem k dnešnímu stavu Android Studia (nejedná se již o rozpracovaný projekt, ale o plnohodnotné vývojové studio) je programování v Eclipse krokem zpět.

4.1.2 Android Studio

Před tím než budeme ovšem moci Android Studio začít používat, musíme ho nejdříve nainstalovat. Nejprve před samotnou instalací musíme stáhnout instalační soubor ze stránek Android Studia (zde si pouze ohlídáme, abychom stahovali verzi pro náš operační systém

tedy Windows, Linux nebo Mac). Tímto je samotné studio připravené, ale abychom mohli programovat, tak potřebujeme ještě vývojové nástroje samotné Javy neboli takzvané JDK, které se stáhne ze stránek Oracle. Po tomto kroku máme již připraveno vše, co potřebujeme pro začátek programování.

Android Studio nás při prvním zapnutí provede pomocí dialogových oken vytvořením projektu, kde s námi nastaví vše potřebné jako je název projektu, kam se nám bude projekt ukládat, na jaké zařízení a jakou verzi Androidu budeme vyvíjet, jestli chceme prázdný projekt nebo připravit nějaké základy typu aktivity. Jakmile se proklikáme vytvořením projektu, dostaneme se na hlavní obrazovku vývojového prostředí. Uprostřed obrazovky si můžeme všimnout obrázku telefonu, ve kterém probíhá návrh každé obrazovky. Pokud si nahoře přepneme záložku na MainActivity.java tak místo obrazovky pro návrh uvidíme obrazovku, kde bude probíhat naše programování. Po levé straně vidíme stromovou strukturu všech souborů a adresářů, které náš projekt využívá, najdeme zde základní xml soubory s layouty obrazovek nebo stringů, dále obrázky, java soubory a android manifest.



Obrázek 6 - Vývojové prostředí Android Studio (zdroj: vlastní)

4.1.3 Layout

Ačkoliv nám v naší aplikaci layout nezajišťuje žádnou funkci aplikace, tak je tím nejvyšším stupněm, s kterým uživatel přijde do styku, proto je potřeba návrh layoutů nezanedbat a věnovat důkladnou péči. Za dobu, co nás provází mobilní aplikace, si uživatelé navykli na některé standardy v rozložení prvků, které je dobré dodržovat, protože při nepřeberném množství aplikací, které jsou nyní na trhu dostupné, může jen špatné rozmístění prvků znamenat, že uživatel sáhne po jiné aplikaci, i když nutně nemusí být funkčně lepší. Ohledně tohoto tématu nám mohou být velice nápomocny, tzv. Google material designs, které nám v podobě obrázkových návodu specifikují, jak by měl návrh vypadat, aby byl co nejvhodnější pro běžného uživatele (více informací o material design na https://www.google.com/design/spec/layout/metrics-keylines.html#).

Nyní už k samotnému tvoření layoutu. Můžeme se rozhodnout, zda využijeme metodu tvoření layoutu pomocí xml nebo pomocí android designeru. Na základě zhodnocení teoretické části této problematiky a z vlastní zkušenosti autora práce si zvolíme metodu psaní vlastního xml. Hlavním důvodem se pro nás tedy stává větší přehlednost a při osvojení i rychlost, protože při praktickém testování se v režimu designu někdy stává, že některé prvky se nechtějí přesunout tam, kde byste je potřebovali a v tu chvíli je stejně potřeba přejít do režimu psaní xml a upravit to zde.

4.1.3.1 Úvodní obrazovka

Díky návrhu máme již ideu, jak bude úvodní obrazovka vypadat, máme zde textview, který uživateli pouze oznámí, co se od něj očekává, dále zde je pole pro vyhledávání a nakonec listview které nám zobrazí dostupné měny.

Otevřeme si tedy ve složce layout soubor aktivity_main.xml, který je automaticky vytvořený s projektem. Jakmile máme soubor otevřen tak se přepneme do módu editování xml pomocí záložky ve spodní části obrazovky. Zde vidíme, že už je zde předpřipravený kód, který nám zajišťuje některé základní funkčnosti. Hlavní tag s názvem LinearLayout, nám oznamuje, že se bude jednat o rozhraní, kde se každý nový prvek zařadí pod prvek, který byl předním. To znamená, že dva prvky nemůžou být vedle sebe. V našem případě nám LinearLayout vyhovuje a proto ho ponecháme tak jak je. Dále zde máme layout_width

a layout_height nastavené na match_parent, to znamená, že náš LinearLayout zabere všechno dostupné místo uvnitř elementu, ve kterém byl vytvořen v našem případě tedy celou obrazovku. Opakem match_parent je wrap_content který využíváme, pokud chceme, aby element byl velký pouze tak, aby se do něj vešly všechny hodnoty, které obsahuje. Poslední důležité atributy jsou padding, ty nám určují odsazení od okraje obrazovky, padding dále rozlišujeme na levý, pravý, horní nebo dolní. Ještě jsme zapomněli zmínit velice důležitou část a tou je, že všechny tagy včetně tagu LinearLayout musí být ukončené, zpravidla stejným tagem jako začínají, akorát se navíc přidá před název lomítko.

První co nyní potřebujeme přidat je nadpis, který uživateli napoví, co má udělat. Docílíme toho pomocí tagu TextView, kterému nastavíme výšku a šířku na wrap_content. Dalším atributem je zde textAppearance, který nám určuje, jaký styl písma bude použit. Atributem text přidáváme asi nejpodstatnější část a to je vlastní text, který se zobrazí uživateli. Text můžeme napsat „natvrdo“, ale my zvolíme variantu odkazu na seznam stringů (viz kapitola 3.1.13 String Resources). Poslední atribut je id, kterým přiřadíme danému prvku jedinečný identifikátor, pomocí kterého se na něj budeme následně odvolávat. Id textView jsme nastavili jako UvodniText.

Dále přidáváme vyhledávací pole a to pomocí tagu SearchView, to uživateli poslouží možností vyhledat si požadovanou měnu. Tento prvek má pouze základní atributy, kterými jsou šířka a výška nastavené na wrap_content a id nazvané SearchView.

Poslední prvkem této obrazovky je ListView, do kterého načteme předem vytvořené pole dostupných měn. Tento prvek obsahuje stejné atributy jak vyhledávací pole a to je šířka a výška jako wrap_content a id nastavené na listView.

Tímto máme první obrazovku připravenou, pokud se chceme podívat, jak náš návrh vypadá, můžeme se opět přepnout zpět do zobrazení designu, kde se nám automaticky vykreslí námi připravené xml. Níže se můžeme podívat na úryvek xml úvodní strany, pro správnou funkci je potřeba využít celý kód. Pro celý kód viz Přílohy.

```
<TextView
    android:layout_width="wrap_content"
```

```
    android:layout_height="wrap_content"
    android:textAppearance="?android:attr/textAppearanceLarge"
    android:text="@string/Uvod"
    android:id="@+id/UvodniText" />

<SearchView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/searchView" />

<ListView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/listView" />
```

4.1.3.2 Kurzovní lístek

Tato obrazovka nám slouží k zobrazení aktuálních kurzů, které se stahují na pozadí. Potřebujeme tedy pouze vytvořit list, ve kterém kurzy zobrazíme a k tomu si připravíme jednoduchý nadpis, aby bylo jasné, na co se díváme.

Jelikož zde už nemáme soubor layoutu vytvořený automaticky, musíme si naše xml nejprve vytvořit. Klikneme pravým tlačítkem na složku layout, zde vybereme new, Layout resource file a ten pojmenujeme jako aktivity_kurzovni_listek, nyní si nově vytvořený soubor otevřeme a jsme připraveni pokračovat dále. Nyní si podle návrhu připravíme textView, které vytvoříme stejně jako v předchozím případě. Oproti předchozímu případu ovšem nepřiřazujeme žádný nápis, protože se o to budeme starat programově, kdy přiřadíme do nadpisu správnou měnu, pro kterou je kurz stažený. Další novinkou zde je layout_alignParent, který nám zajišťuje zarovnání podle rodičovského prvku. V našem případě zarovnáváme na začátek prvku, doleva nahoru. A jako poslední atribut přiřadíme id s názvem textKurzovniListek. ListView připravíme stejně jako na úvodní obrazovce, navíc s využitím znalostí ohledně layout_alignParent.

Na následujících řádcích se můžeme podívat na úryvek prvků, a jak by měly správně vypadat, celý kód je dostupný k nahlédnutí viz Přílohy.

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textAppearance="?android:attr/textAppearanceLarge"
    android:text="Large Text"
    android:layout_alignParentStart="true"
    android:layout_alignParentTop="true"
```

```
    android:layout_alignParentLeft="true"
    android:id="@+id/textKurzovniListek"/>


<ListView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/listView2"
    android:layout_below="@+id/textKurzovniListek"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true" />
```

4.1.3.3 Porovnání

Na této obrazovce budeme využívat dvou oken, jedno pro zadání hodnoty první měny, která se po stisknutí tlačítka přepočítá na měnu druhou. Tyto hodnoty se nám budou po každém výpočtu zapisovat do listu jako historie.

Stejně jako v předchozím případě si nejprve vytvoříme nový soubor xml a ten pojmenujeme aktivity_compare. Zde si vytvoříme dvakrát nadpis, který nám programově zobrazí název první a druhé měny, aby si uživatel mohl ověřit, že má vybráno správně. Vytvoříme si dvakrát textView úplně stejně jako v předchozím v příkladu, jedno nazveme prvniMena a druhé druhaMena. Při tvorbě těchto textových polí jsem navíc oproti předchozím využil atributu layout_align, který má podobnou funkci jako layout_alignParent, jen s tím rozdílem, že pro zarovnání můžeme použít jakýkoliv jiný prvek v našem návrhu, v našem případě tedy k textu druhaMena. Ve výsledku to znamená, že oba texty budou ve stejné výšce.

Dále budeme potřebovat tlačítko, které budeme využívat pro provedení výpočtu. Tlačítko vytvoříme tagem Button. Pro tlačítko využíváme stejné atributy jako u ostatních prvků. Zde bych vypíchl pouze atribut layout_centerHorizontal, který nám tlačítko vycentruje na střed v horizontálním směru. Název id tlačítka jsme nastavili na vypocitej.

Dalšími prvky, které potřebujeme přidat, jsou dvě pole, do kterých budeme zadávat číselné hodnoty pro přepočet. Tyto pole vytvoříme tagem EditText, zde si jich můžeme povšimnout některých zajímavých atributů, jedním z nich je inputType, díky kterému můžeme omezit, co bude možné zadat, my jsme vybrali number, což znamená, že uživateli není

umožněno vepsat nic jiného než čísla. Jeden z atributů, který jsme si ještě nepředstavili je ems, jedná se o jednotku velikosti a nastavujeme přes ní jak bude v daném poli EditText velké písmo.

Nyní už jen potřebujeme vytvořit nadpis Historie, toho docílíme pomocí TextView a pod toto TextView přidáme ListView, do kterého budeme zapisovat historii našich výpočtů. Na ukázkovém kódu můžete vidět, že se jedná znovu o recyklování námi dříve vytvořených prvků. Na tom bych rád demonstroval, že osvojení základů xml není tak těžké jak se na první pohled může zdát a jde jen o to naučit se základní klíčová slova. Celý kód obrazovky porovnání dostupný viz Přílohy.

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textAppearance="?android:attr/textAppearanceLarge"
    android:text="Large Text"
    android:layout_alignBottom="@+id/druhaMena"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true"
    android:id="@+id/prvniMena"/>

<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/Count"
    android:id="@+id/vypocitej"
    android:layout_below="@+id/druhaMena"
    android:layout_centerHorizontal="true" />

<EditText
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:inputType="number"
    android:ems="10"
    android:id="@+id/editText"
    android:layout_below="@+id/prvniMena"
    android:layout_toLeftOf="@+id/vypocitej"
    android:layout_toStartOf="@+id/vypocitej"
    android:layout_marginTop="29dp" />

<ListView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/listView3"
    android:layout_below="@+id/editText"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true"
    android:layout_marginTop="47dp" />

<TextView
```

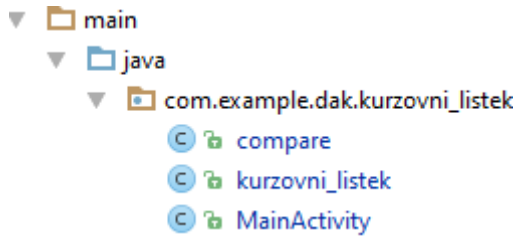
```
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:textAppearance="?android:attr/textAppearanceLarge"
android:text="@string/History"
android:id="@+id/textView5"
android:layout_below="@+id/editText"
android:layout_alignParentLeft="true"
android:layout_alignParentStart="true" />
```

4.1.4 Java

Nyní když máme připravené uživatelské rozhraní pomocí xml, můžeme se vrhnout na programování v Javě, kde těmto elementům přidáme nějakou funkci. Kódy použité v této práci jsou z důvodu zkrácení vytrženy z kontextu a slouží pouze pro ilustrování dané problematiky a nemusí být jako celek funkční. Pro kompletní a funkční kód viz Příloh.

4.1.4.1 Main Activity

Otevřeme si soubor MainActivity.java, který se nachází ve složce main, java, com.example.uzivatelskejmeno.kurzovni_listek (je potřeba doplnit správné uživatelské jméno, které uživatel má v rámci projektu, viz Obrázek 7 - Umístění java souborů). Tato aktivita je přidružena k layoutu activity_main.



Obrázek 7 - Umístění java souborů (zdroj: vlastní)

Ihned po otevření souboru MainActivity.java se nám automaticky vytvoří metoda onCreate. V této metodě se bude řešit vše co se má stát při vzniku aktivity, v případě main activity to znamená, co se stane při startu aplikace. První řádek, který se nám v metodě onCreate vytvořil, se využívá k uložení stavu aplikace a poté možného obnovení toho stavu, což zatím v naší aplikaci potřebovat nebudeme. Druhý řádek se nám stará o to, že k aktuální aktivitě se zobrazí správný soubor xml.

Když už jsme seznámení s vytvořeným java souborem, můžeme se podívat, co potřebujeme naprogramovat na úvodní stránce. Nejprve potřebujeme pole prvků, které bude reprezentovat všechny měny, které si uživatel může zvolit, to jsme si vytvořili ve strings.xml, pomocí tagu <string-array>. V této části ovšem uživatel nemá možnost pole vidět, ale musíme nejdřív naplnit tímto polem listView, které jsme si vytvořili v návrhu. Vytvoříme si tedy proměnnou lv, do které naše listview načteme, a do další proměnné currency_wn[] načteme vytvořené pole ze strings.xml. Nyní už jen s využitím třídy ArrayAdapter, naplníme simple_list_item_1 našimi hodnotami a tento list potom už jen předáme do listview k zobrazení. Níže se můžete podívat na úryvek kódu, který nám naplní listview hodnotami z připraveného pole.

```
//vytvoření proměnné lv
private ListView lv;
//přiřazení proměnné lv, konkrétní list z našeho xml
lv = (ListView)findViewById(R.id.listView);

final String currency_wn[] = getRe-
sources().getStringArray(R.array.currency_whole_name);

//Naplnění ListView názvy měn
ArrayAdapter<String> arrayAdapter = new ArrayAdapter<>(
        this,
        android.R.layout.simple_list_item_1,currency_wn);
lv.setAdapter(arrayAdapter);
```

Aplikace nyní při spuštění uživateli zobrazí seznam názvů měn, který si může prohlížet, ovšem při kliknutí na jakékoliv pole listview se nic nestane a to nyní potřebujeme ošetřit. Docílíme toho pomocí rozhraní setOnItemClickListener, který odposlouchává listView, které specifikujeme a jakmile zjistí, že bylo kliknuto na nějaký prvek tak se provede následující kód. V našem případě, nám spustí novou aktivitu a předá informace potřebné ke stažení správného kurzu v nově vytvořené aktivitě.

Informaci, kterou budeme potřebovat pro stažení správného kurzu je pozice, na kterou bylo kliknuto, to znamená jaké pořadí má prvek, který uživatel vybral. Toho jsme docílili pomocí cyklu for, kdy do objektu MyObject jsme vložili vybranou měnu, tento objekt porovnáme s polem currency_wn[i] a s každým krokem nám i naroste o jedna a tím se přesuneme na následující prvek v poli. Jakmile nám podmínka if oznámí, že se objekt rovná

s položkou v poli, uložíme aktuální hodnotu v i do MainActivity.position. Tuto pozici později využijeme v dalším java souboru.

Když máme tuto informaci tak se už jen potřebujeme dostat na další aktivitu, pro uživatele to znamená, že jakmile vybere hodnotu z listView otevře se mu další obrazovka aplikace. Toho docílíme pomocí Intentu, ve kterém nám stačí specifikovat, v jaké jsme aktivitě a do jaké aktivity se chceme přepnout. Potom už jen tento Intent zavoláme a provede se otevření nové aktivity.

```
lv.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> parent, View view,
                            int position, long id) {


        //Zjištění na co bylo vlastně kliknuto, aby se mohl pro danou
měnu stáhnout správný kurz
        Object MyObject = parent.getAdapter().getItem(position);
        for(int i = 0; i < 31;i++){
            if (currency_wn[i]== MyObject){
                MainActivity.position = i;
            }
        }
Intent a = new Intent(MainActivity.this, kurzovni_listek.class);
startActivity(a);
    }
});
```

Uživatel nyní při spuštění aplikace uvidí list dostupných měn a zároveň pokud si nějakou měnu vybere a klikne na ní, otevře se mu nová, nicméně zatím prázdná aktivita. Tuto aktivitu budeme v následující kapitole plnit daty staženými z internetu, kvůli tomu by bylo vhodné, respektive nezbytné, aby naše aplikace kontrolovala přístup k internetu. To nám zajistí takzvaný ConnectivityManager, který si vytvoříme a následně pomocí něj a funkce getNetworkInfo zjišťujeme, zdali je telefon připojen k síti Wi-Fi nebo k mobilnímu internetu. Následně využijeme podmínky if, kterou obalíme Intent vytvořený předchozí části to nám zajistí, že se další aktivita neotevře, dokud se neověří připojení. Nově vytvořená podmínka nám pomocí klíčového slova isConnected ověří, zdali nám ConnectivityManager vrací alespoň na jedno připojení true a pokud ano tak jsme připojeni k internetu a můžeme otevřít novou aktivitu, pokud ne tak zobrazíme uživateli informaci o tom, že je

pro další postup potřeba připojit se k internetu. Na následujících řádkách demonstruji využití ConnectivityManageru s podmínkou if.

```
//Kontrola připojení k internetu a následné přepnutí aktivity pokud je
připojení OK
ConnectivityManager connManager = (ConnectivityManager) getSystemServi-
ce(Context.CONNECTIVITY_SERVICE);
NetworkInfo mWifi = connMa-
nager.getNetworkInfo(ConnectivityManager.TYPE_WIFI);
NetworkInfo mData = connMa-
nager.getNetworkInfo(ConnectivityManager.TYPE_MOBILE);
if(mWifi.isConnected() || mData.isConnected()){
    //pokud je připojení k internetu spustí se aktivita
    Intent a = new Intent(MainActivity.this, kurzovni_listek.class);
    startActivity(a);
}
else {
    //pokud není připojení k internetu vyskočí informace
    Toast.makeText(MainActivity.this, "Nebylo detekováno internetové při-
pojení", Toast.LENGTH_SHORT).show();
}
```

Teď, když máme vše ošetřeno, můžeme přejít na programování aktivity kurzovní lístek.

4.1.4.2 Kurzovní lístek

V této aktivitě nebude ani tak důležité to co samotný uživatel uvidí, nýbrž co se bude dít na pozadí. Hned při otevření nové aktivity je potřeba stáhnout příslušný JSON z webového serveru fixer.io. Tento JSON následně musíme zpracovat a zobrazit uživateli do listview jen potřebné informace, kterými jsou název měny a hodnota kurzu.

Stejně jako v předchozím případě máme v našem java souboru automaticky vytvořenu metodu onCreate, ve které použijeme znalosti, které jsme využili na předchozí obrazovce, to znamená vytvoření textu, vytvoření listview, plnění listview hodnotami a ošetřit kliknutí na listview. Samozřejmě žádné hodnoty, kterými bychom listview plnili, zatím nemáme, protože je nejprve potřebujeme stáhnout.

O stažení potřebných dat se nám stará třída AsyncTaskParseJson, která nám data stáhne, naparsuje, setřídí pro naše potřeby a poté pošle do listview, aby se mohli zobrazit uživateli. Uvnitř této třídy máme metodu doInBackground, která nám obstarává funkci na pozadí zapnuté aplikace, místo této metody máme k dispozici ještě metodu onPreExecute() a onPostExecute, které se nám starají o to, co se stane před tím, než se provede

doInBackground a co se provede po tom, ovšem pro potřeby naší aplikace je nebudeme potřebovat.

V metodě doInBackground nejprve řešíme správně nastavený link pro stažení souboru, který se liší v koncovce a podle toho stáhne libovolnou měnu. Koncovka je ve tvaru zkratky požadované měny. Tuto koncovku získáme z námi vytvořeného pole zkratek, z kterého vybereme správnou zkratku pomocí indexu, který jsme získali na předchozí obrazovce. To znamená, že pokud uživatel vybral na první obrazovce měnu česká koruna, která měla index 3, tak program se nyní podívá do pole se zkratkami a vybere pole s indexem 3, kde se nachází zkratka CZK. Takto získanou zkratku připojíme jako koncovku naší url adresy.

```
protected String doInBackground(String... params) {

    selectedCurrency = currency[MainActivity.position];
    url = ("http://api.fixer.io/latest?base=")+selectedCurrency;
```

Dalším krokem je otevřít s tímto serverem připojení a stáhnout soubor. Vytvoříme si tedy http klienta a getRequest, do kterého načteme naší url. Poté přes httpResponse spustíme našeho klienta, do kterého vložíme getRequest který nese naší url adresu. Zároveň nám httpResponse vrací status o stažení souboru. Tento status si vytáhneme pomocí funkce statusLine. Do nově vytvořené proměnné typu int nazvané statusCode si načteme ze statusLine pouze kód oznamující zdali se stažení podařilo, tento kód následně porovnáváme s hodnotou 200, která oznamuje, že stažení proběhlo v pořádku. Pokud se tedy kód rovná jinému číslu než 200 navrátíme hodnotu null, protože se stažení nepodařilo. Pokud se stažení podařilo, máme nyní čitelný zdroj bajtů a tento zdroj nyní převedeme do čitelné podoby stringů v podobě JSONu. Využijeme InputStream, který nám získá obsah, ten načteme pomocí bufferedReaderu a stringBuilder nám ho přetvoří na upravitelnou sekvenci znaků. Nyní pomocí cyklu while vytváříme po řádkách čitelný JSON.

```
HttpClient client = new DefaultHttpClient();
HttpGet getRequest = new HttpGet(url);
try{
    HttpResponse response = client.execute(getRequest);
    StatusLine statusLine = response.getStatusLine();
    int statusCode = statusLine.getStatusCode();
    Log.e("TAG", "1");
    if(statusCode != 200){
        return null;
```

```
    }

    InputStream jsonStream = response.getEntity().getContent();
    BufferedReader reader = new BufferedReader(new InputStreamRea-
der(jsonStream));
    StringBuilder builder = new StringBuilder();
    String line;
    while((line = reader.readLine())!=null){
        builder.append(line);
    }
```

Takto zpracovaný JSON nám nyní reprezentuje string s názvem jsonData (příklad struktury JSONu dostupné na adrese http://api.fixer.io/latest). Ovšem jsonData nám nyní reprezentují celý JSON, z kterého potřebujeme získat pouze určitá data konkrétně měnu a kurz, proto si vytvoříme dva JSON objekty, do jednoho z nich převedeme celá jsonData. Druhý objekt poté získá z prvního objektu pouze část pojmenovanou „rates“.

```
String jsonData = builder.toString();
JSONObject jsonObj = new JSONObject(jsonData);
JSONObject ratesObject = jsonObj.getJSONObject("rates");
```

Pokud se podíváme na strukturu zvoleného JSONu, tak část kterou nyní máme je zobrazena jako název měny (klíč) a hodnota kurzu (atribut), my nyní tyto hodnoty od sebe oddělíme a vytvoříme dvě pole, jedno bude reprezentovat názvy (klíče) a druhé kurzy (atributy). Docílíme toho využitím iterátoru, který nám celý JSON projde po jednotlivých klíčích (keys). Takto projdeme celý JSON a vždy přiřadíme klíč do jednoho pole nazvaného nameRate a atribut do druhého pole nazvaného valueRate.

```
Object value = ratesObject.get(key);
String rate = value.toString();
double doubleRate = Double.parseDouble(rate);
valueRate[counter] = doubleRate;
nameRate[counter] = key;
Log.i("TAG", ""+ key + ":" +value);
counter++;
```

Ještě před tím, než tyto hodnoty zobrazíme uživateli, tak je seřadíme, je to z důvodu, že JSON nemá žádný seznam řazení tak dostáváme hodnoty neseřazené, proto je vhodné je před zobrazením seřadit. Řazení provádíme pomocí for cyklu, kdy projdeme celé pole a porovnáváme hodnotu v poli s hodnotou o jedna za ní, pokud je první hodnota větší, než

ta druhá přesune se za ní, pokud ne zůstane pole stejné a pokračuje se na další prvek. Takto se nám seřadí celé pole od A až po Z.

```
for (int n = 0; n < 31; n++) {
    for (int m = 0; m < 30 - n; m++) {
        if ((nameRate[m].compareTo(nameRate[m + 1])) > 0) {
            String swapString = nameRate[m];
            nameRate[m] = nameRate[m + 1];
            nameRate[m + 1] = swapString;
            double swapDouble = valueRate[m];
            valueRate[m] = valueRate[m + 1];
            valueRate[m + 1] = swapDouble;
        }
    }
}
```

Nyní již jen v novém poli spojím zpět dohromady dvě seřazené pole, a toto nové pole se využije pro naplnění listview hodnotami, které reprezentují název měny a k tomu daný kurz. ListView naplníme stejně jako na minulé obrazovce a stejně tak i ošetříme, pokud uživatel klikne na některé z políček v seznamu kurzů. Uživatel po spuštění aplikace uvidí celý kurzovní lístek pro měnu vybranou na první obrazovce.

```
for(int o=0; o<31;o++){
    rate[o] = (nameRate[o]+" " +valueRate[o].toString());
}
```

4.1.4.3 Kalkulátor

Tato obrazovka uživateli umožní přepočíst si měnu. Docílíme toho pomocí dvou editTextů, přičemž do prvního uživatel zadá hodnotu, pro kterou chce měnu přepočíst a po stisknutí tlačítka se mu v druhém zobrazí již přepočtená měna. Dále se ještě uživateli budou pro danou měnu uchovávat provedené přepočty coby historie ve spodní části obrazovky.

Po tom co si najdeme všechny komponenty z našeho xml pomocí findViewById se můžeme rovnou pustit do programování tlačítka. Jakmile uživatel klikne na tlačítko tak je potřeba hned ošetřit že vyplnil číslo do prvního editText (ošetření, že může vyplnit pouze číslo, proběhlo již při připravování layout v xml), toho docílíme pomocí podmínky if, kdy porovnáme hodnotu v editTextu s prázdnou hodnotou, tedy prázdné uvozovky. Pokud ta-

to podmínka bude pravdivá, tak to znamená, že uživatel nezadal nic do prvního pole a je potřeba ho upozornit, že má vyplnit nějaké číslo.

```
final String test = editText.getText().toString();
if(test.matches("")){
    Toast.makeText(compare.this, "Musíte zadat číslo do prvního editboxu
pro přepočet měny", Toast.LENGTH_LONG).show();
    return;
}
```

Pokud ovšem podmínka není pravdivá, můžeme přejít k výpočtu. Nejprve si tedy tuto hodnotu převedeme na string a následně hned na datový typ double. Nyní už jen vynásobíme uživatelem zadanou hodnotu převedenou na double s příslušnou hodnotou z pole valueRate, které jsme si vytvořili ze staženého JSONu. Výslednou hodnotu převedeme zpět do stringu a zobrazíme v druhém editTextu.

```
String prvniHodnotaString = editText.getText().toString();
Double prvniHodnota = Double.parseDouble(prvniHodnotaString);
Double vysledek = (prvniHodnota * kurzov-
ni_listek.valueRate[kurzovni_listek.position2]);
String vysledekS = String.valueOf(vysledek);
editText2.setText(vysledekS);
```

Při vytváření historie nám postačí poskládat vytvořené proměnné k sobě a zobrazit je v listu. Vytvoříme si tedy proměnnou history, do které za sebe umístíme proměnnou row, která se po každém kliknutí zvětší o jedna, čili nám bude počítat, o kolikátý řádek se jedná. V další části přiřadíme za číslo řádku název první měny, pomocí námi vytvořeného pole a indexu, který jsme získali na první obrazovce. Nyní zobrazíme hodnotu, kterou zadal uživatel a za mezeru i výslednou hodnotu, jako poslední nám zbylo zobrazit název druhé zvolené měny tentokrát z pole nameRate, které jsme vytvořili z JSONu. Tuto proměnnou nyní jen vkládáme do listu a tím pádem uživatel při každém kliknutí na tlačítko vidí daný výpočet i v historii.

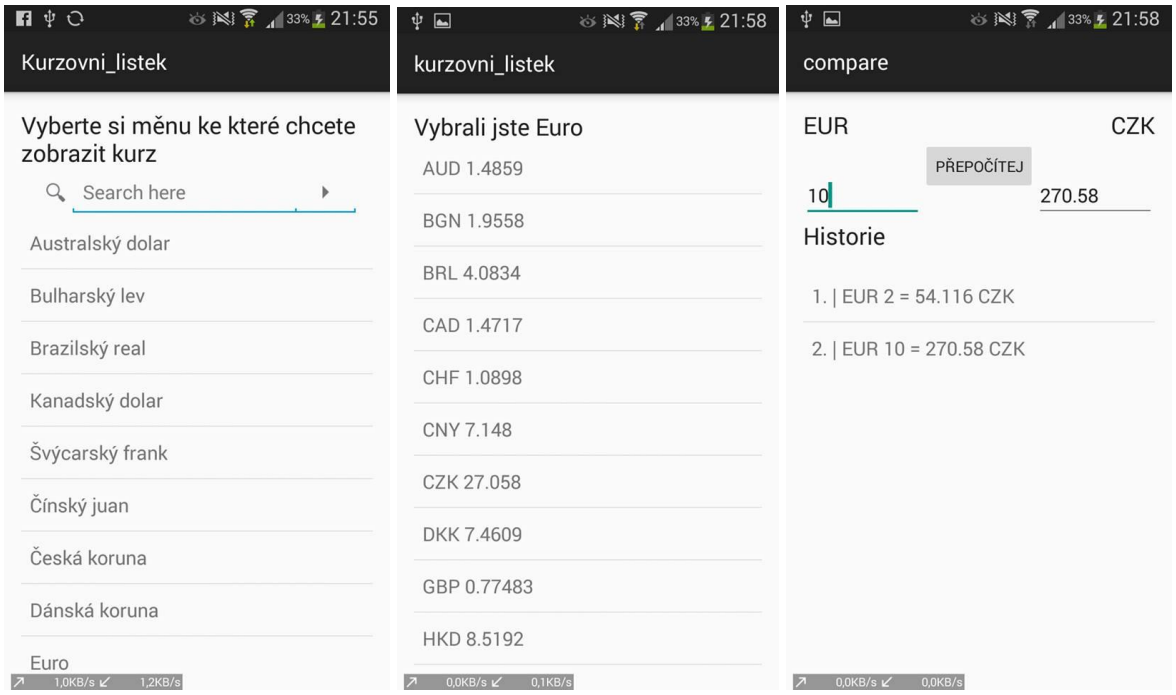
```
history = (row+". | "+currency[MainActivity.position]+"
"+prvniHodnotaString+" = "+vysledekS+" "+ kurzov-
ni_listek.nameRate[kurzovni_listek.position2]);
historyList.add(history);
row++;
```

```
final ListView lv3 = (ListView)findViewById(R.id.listView3);
final ArrayAdapter<String> arrayAdapter = new ArrayAdapter<String>(
```

```
        this,
        android.R.layout.simple_list_item_1, historyList);
lv3.setAdapter(arrayAdapter);
arrayAdapter.notifyDataSetChanged();
```

4.1.5 Testování

Hotová aplikace byla testována na mobilní zařízení Samsung Galaxy S3 pomocí připojení přes USB kabel. Aplikace nebyla nahrána na Google play vzhledem k jednoduchosti a vytváření pouze pro edukativní účely. Následující obrázky zobrazují výslednou aplikace na zmíněném zařízení.



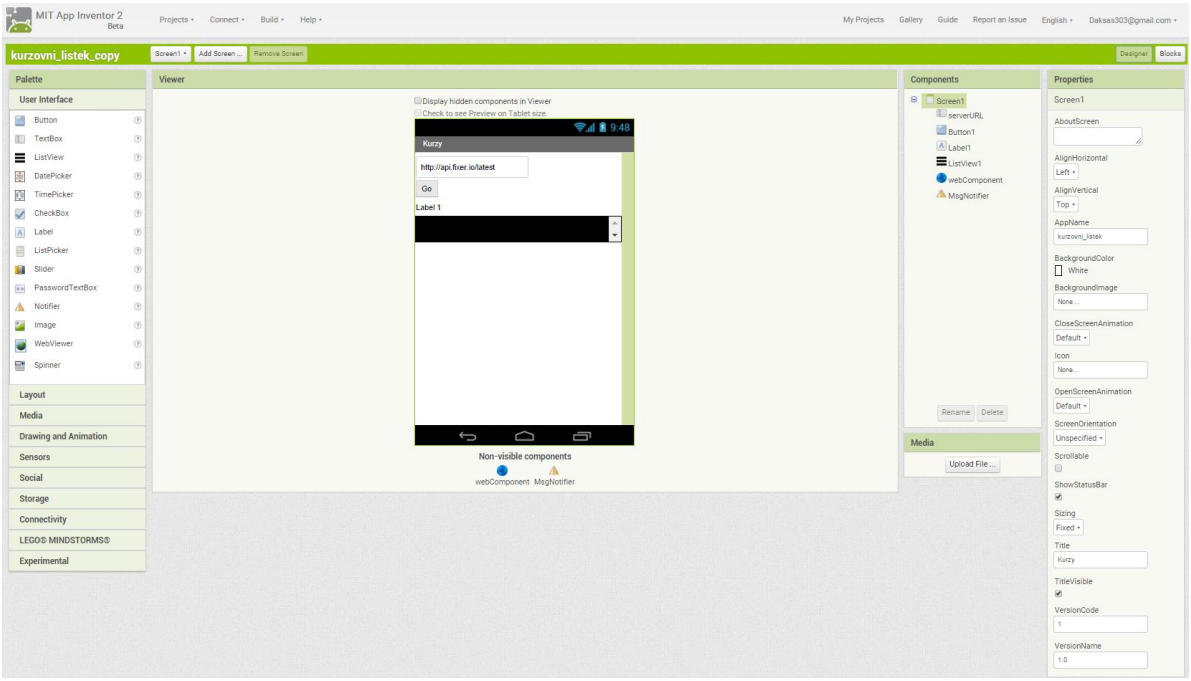
Obrázek 8 - Finální aplikace vytvořená v Android Studiu (zdroj: vlastní)

Aplikace v App Inventoru

Druhou verzi aplikace jsem tvořil podle stejného návrhu jako v první verzi jen s tím rozdílem, že místo Android Studia bylo použito vývojové prostředí MIT App Inventor 2. Výhodou tohoto vývojového prostředí je absence jakéhokoli psaní kódu, místo toho se pro vytvoření funkční logiky aplikace, používá metoda drag and drop, kdy v podstatě myší přesouváme jednotlivé logické bloky a postupně je do sebe kombinujeme do stále větších funkčních částí.

4.1.6 App Inventor 2

App Inventor se na rozdíl od Android Studia z předchozí části nemusí instalovat, nýbrž potřebujeme pouze účet Google a internetový prohlížeč. Pomocí našeho Google účtu se přihlásíme na server http://ai2.appinventor.mit.edu/ a to je vše co potřebujeme udělat, nyní se můžeme pustit do vývoje aplikace.

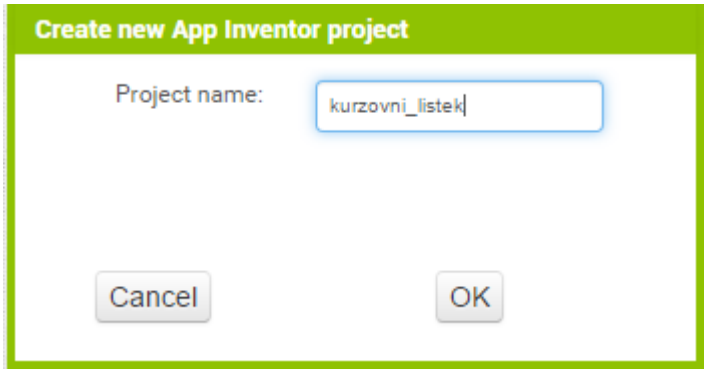


Obrázek 9 - Vývojové prostředí App Inventor 2 (zdroj: vlastní)

Ještě bych rád zmínil, že využití Google účtu nám umožňuje pracovat na aplikaci na různých zařízeních s téměř okamžitou synchronizací bez nutnosti přenášet ručně jakákoliv data.

4.1.7 Programování App Inventor 2

Jelikož po prvním přihlášení na stránky App Inventoru ještě nemáme vytvořený žádný projekt, je naší jedinou možností kliknout na tlačítko „Start new project“, které nám zobrazí dialogové okno, kde do dialogového okna zadáme název aplikace, v mém případě tedy „Kurzovni_lístek“ (Názvy je možné zadávat pouze bez diakritiky a bez mezer).



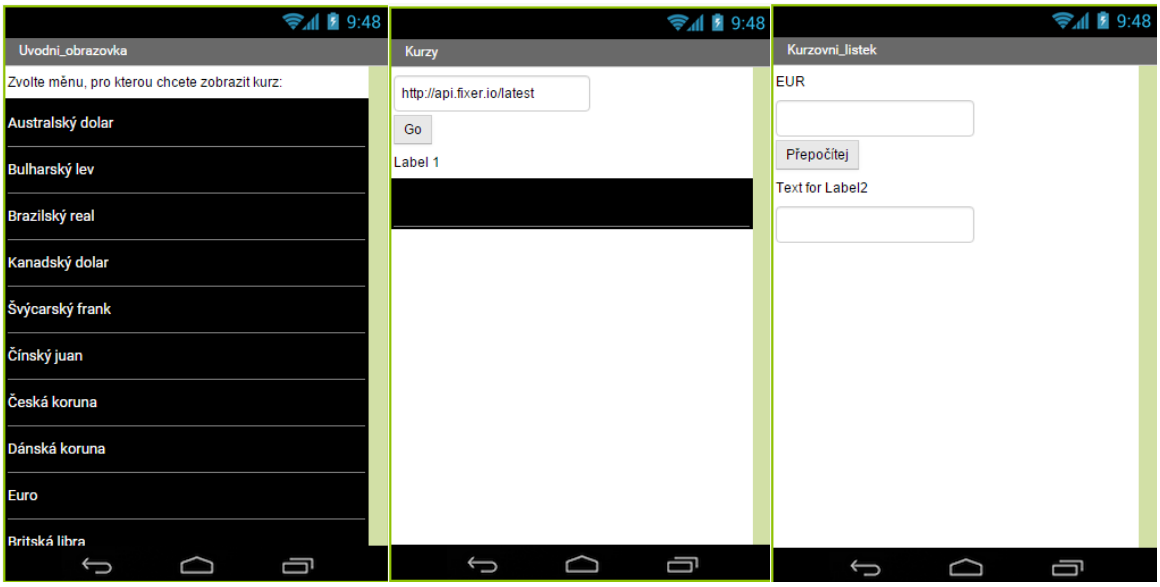
Obrázek 10 - Pojmenování projektu App Inventor (zdroj: vlastní)

Po úspěšném vytvoření projektu se nám zobrazí podobně jako v Android Studiu úvodní obrazovka, která slouží hlavně pro vytvoření vzhledu jedné strany aplikace, tato obrazovka se nazývá „Designer“.

4.1.7.1 Designer

S využitím metod popsaných v teoretické části jsme si vytvořili tři obrazovky, které bude uživatel využívat. Při pojmenovávání obrazovek si můžeme povšimnout, že první vytvořená obrazovka Screen1 nejde žádným způsobem za pomoci App Inventoru přejmenovat a to z důvodu, že samotný zdrojový kód App Inventoru pracuje s tím, že se tato obrazovka takhle bude jmenovat. Bohužel jsme tedy nuceni mít obrazovky: Uvodni_obrazovka, Screen1 (tato obrazovka by za normálních okolností byla pojmenována jako Kurzy) a Kurzovni_listek.

Posledním krokem je přidání neviditelných komponent, mezi které v našem případě patří webComponent a MsgNotifier, první zmíněný nám umožňuje využít funkci pro stažení JSON souboru z internetu a druhý nám umožňuje vytvářet dialogová okna, pokud se něco stane, v našem případě se jedná o nějakou chybu. Na následujícm Tím máme hotovo a můžeme přejít na druhou část, která je schovaná pod tlačítkem „blocks“.



Obrázek 11 - Finální design aplikace v App Inventor 2 (zdroj: vlastní)

4.1.7.2 Bloky

Po kliknutí na tlačítko blocks se nám otevře prázdná pracovní plocha, na které budeme tvořit celé naše programování. Jak už název napovídá, budeme zde využívat tzv. bloky, které reprezentují určité části kódu, které ale nutně nepotřebujeme znát. Seznam použitelných bloků nalezneme v levé části obrazovky, podobně jako tomu bylo v případě aktivních prvků v designeru. Nyní se již podíváme, jak vytvoříme konkrétní bloky v našem případě, nejprve se podíváme na stránku Uvodni_obrazovka. Přepneme se pomocí rozbalovacího okna a vpravo se pomocí tlačítka dostaneme na vývojovou stránku.

4.1.7.2.1 Uvodni_obrazovka

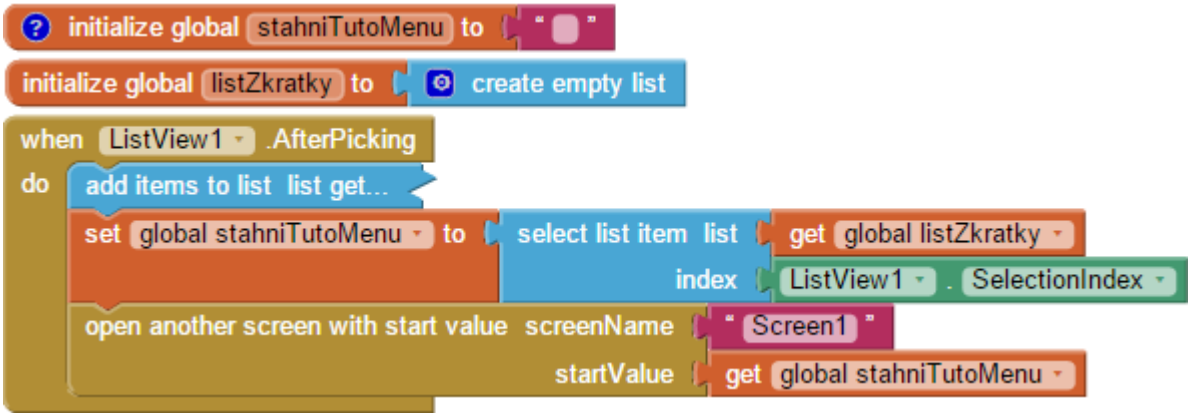
První řádek nám vytváří proměnnou stahniTutoMenu, do které zatím vkládáme pomocí růžového bloky prázdný text. Tato proměnná bude sloužit k předání zkratky kurzu následující stránce, která ho bude stahovat z internetu.

Druhý řádek nám vytváří prázdný list pojmenován listZkratky, do toho listu jsem vložil všechny kurzy, které je možné stáhnout a z kterých bude uživatel vybírat.

U třetího řádku si můžeme povšimnout, že je vloženo víc bloků do jednoho bloku hlavního. Hlavní blok nám říká, co se stane v případě, že uživatel vybere položku z ListView1.

ListView1 nám reprezentuje vytvořený list na úvodní obrazovce z předchozí části, který vidí uživatel na úvodní stránce a vybere si z něj požadovaný kurz.

V případě, že je tedy vybrána položka z daného listu tak se provedou následující kroky: do námi předpřipraveného listu se načtou zkratky měn (Přidávání položek je úmyslně zmenšeno, protože se jedná o repetitivní činnost). Další krok nám z tohoto listu vybere tu zkratku, která odpovídá měně (indexu měny) vybrané uživatelem. A poslední krok nám otevře obrazovku Screen1 a pošle tam proměnou stahniTutoMenu, ve které je uložena zkratka vybrané měny.



Obrázek 12 - Úvodní obrazovka ukázka bloků (zdroj: vlastní)

4.1.7.2.2 Screen1

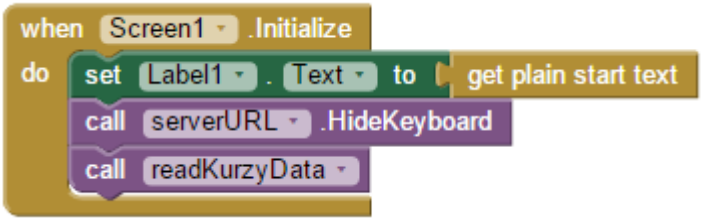
Stejně jako na předchozí obrazovce se nám i nyní při vytváření tvoří i nové proměnné potřebné pro správnou funkci. Tyto proměnné vám stručně v následujícím odstavci představím.

První proměnná je vybranyKurz, jedná se o proměnnou, která na konci cyklu bude uchovávat informace o tom, jaký kurz uživatel vybral pro porovnání s první měnou. Následně vytváříme čtyři listy, které mají podobnou funkci, jedná se o listy s názvy kurzyData, kurzyCele, kurzyNazvy a kurzyCisla. Každý z těchto listů uchovává určitou část staženého souboru JSON. KurzyData uchovávají celý JSON, kurzyCele uchovávají název kurzu a danou hodnotu, kurzyNazvy uchovávají pouze názvy kurzů a kurzyCisla naopak pouze hodnoty kurzů.

Poslední proměnná, kterou zde tvoříme, se jmenuje kurzyURL a jedná se o adresu, z které se bude stahovat samotný soubor JSON. To znamená, že ji tvoří pevný základ „http://api.fixer.io/latest?base=“, ke kterému se následně připojí zkratka vybraného kurzu z předchozí stránky, např. při vybrání České koruny bude adresa vypadat následovně „http://api.fixer.io/latest?base=CZK“.

Obrázek 13 - Proměnné vytvořené na začátku obrazovky Screen1 (zdroj: vlastní)

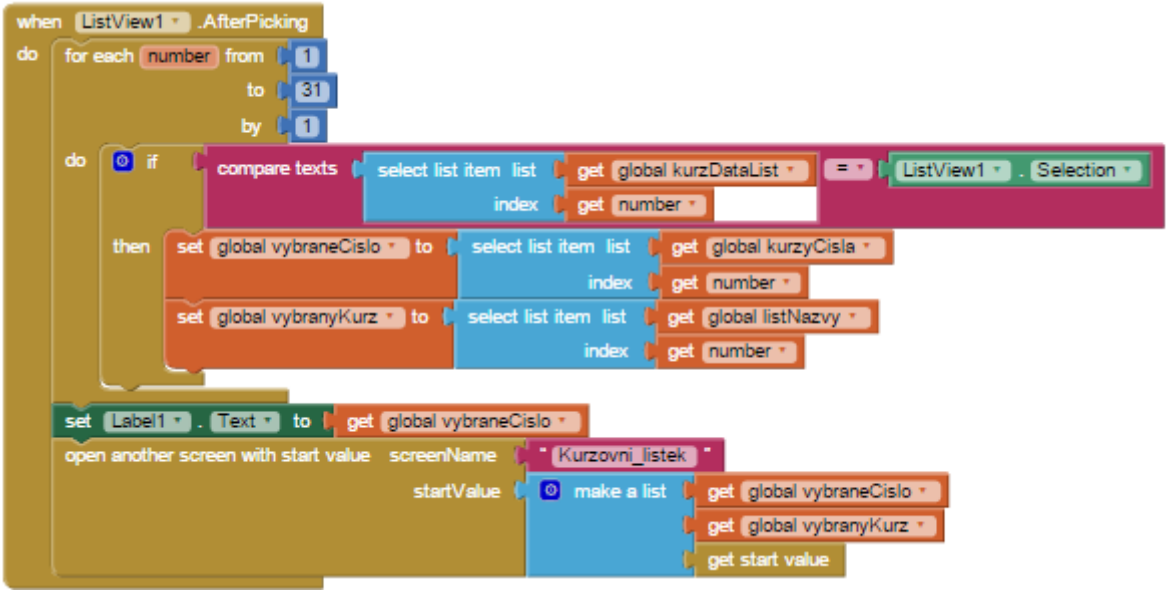
Další částí je řídící blok Initialize, který se provede v momentě kdy je vybraná obrazovka načtená. V našem případě se první provede vypsání adresy uložené v kurzyURL do textBoxu. Další krok nám zobrazí do labelu, jaká měna se bude stahovat. Třetí krok schová klávesnici, která automaticky vyskakuje, pokud je v návrhu přítomen TextBox. A posledním krokem je zavolání procedury readKurzyData.



Obrázek 14 - Ukázka řídícího bloku initialize (zdroj: vlastní)

Procedura readKurzyData nám s pomocí webComponent vezme adresu JSONu a stáhne ho. Další částí této procedury je blok, který čeká, dokud mu webComponent neoznámí, že má stažený text. Po přijetí textu se provádí další příkazy, první je kontrola tzv. responseCode, který porovnáváme s hodnotou 200 a pokud se rovná tak víme, že se stažení povedlo a můžeme pracovat dále. V tomto kroku již nastavíme do proměnné kurzyData dekódovaný JSON a pomocí cyklu for each se naplní ostatní dříve vytvořené listy a ListView1 se z příslušného listu naplní (pro ukázku kódu viz Příloh, z důvodu rozsáhlosti skupiny bloků)

Poslední z hlavních bloků, které zde máme, nám zajišťuje funkci, pokud uživatel klikne na libovolnou měnu v ListView1. Nejprve potřebujeme zjistit, jakou měnu uživatel vybral, to vyřešíme opět pomocí for each cyklu který se nám zastaví na čísle kdy se rovná vybraná měna s měnou v listu kurzyCele a toto číslo využijeme jako index pro zjištění pozice měny. Jakmile máme tuto znalost tak z listu kurzyCisla a kurzyNazvy vybereme nazev a hodnotu správné měny a následně otevřeme další obrazovku Kurzovni_listek a předáme dvě proměnné, které jsme získali výše z listů.

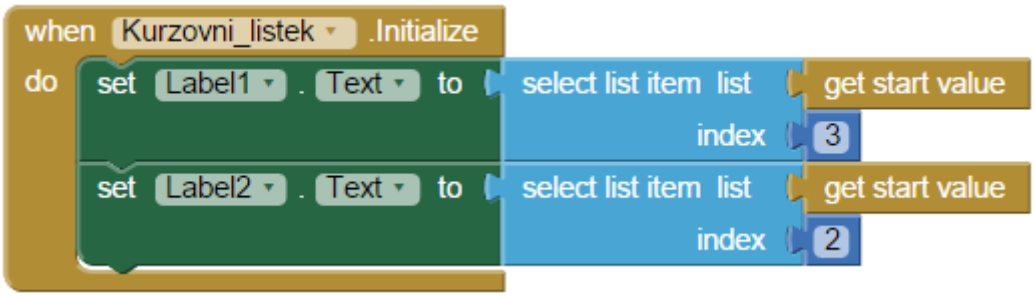


Obrázek 15 - Ukázka ošetření kliknutí uživatelem do listView (zdroj: vlastní)

4.1.7.2.3 Kurzovní lístek

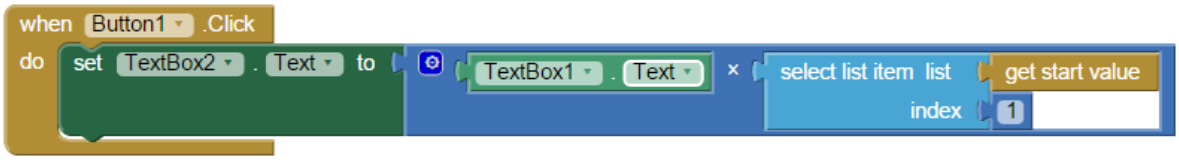
Poslední obrazovka, kterou si zde ukážeme, nese název kurzovni_listek. Jedná se o obrazovku, na které bude probíhat přepočet prvního vybraného kurzu na druhý vybraný kurz. To bude provedeno pomocí dvou textBoxů, přičemž do prvního zadáme množství měny pro přepočet a po stisknutí tlačítka přepočítej se nám v druhém textBoxu zobrazí vypočítaná hodnota.

Docílili jsme toho následujícím způsobem: stejně jako v předchozím případě zde máme blok, který nám říká, co nastane, když se obrazovka načte. Nastane to, že první label nastavíme na hodnotu názvu vybrané první měny a druhému labelu nastavíme název druhé měny.



Obrázek 16 - Nastavení hodnot po načtení obrazovky (zdroj: vlastní)

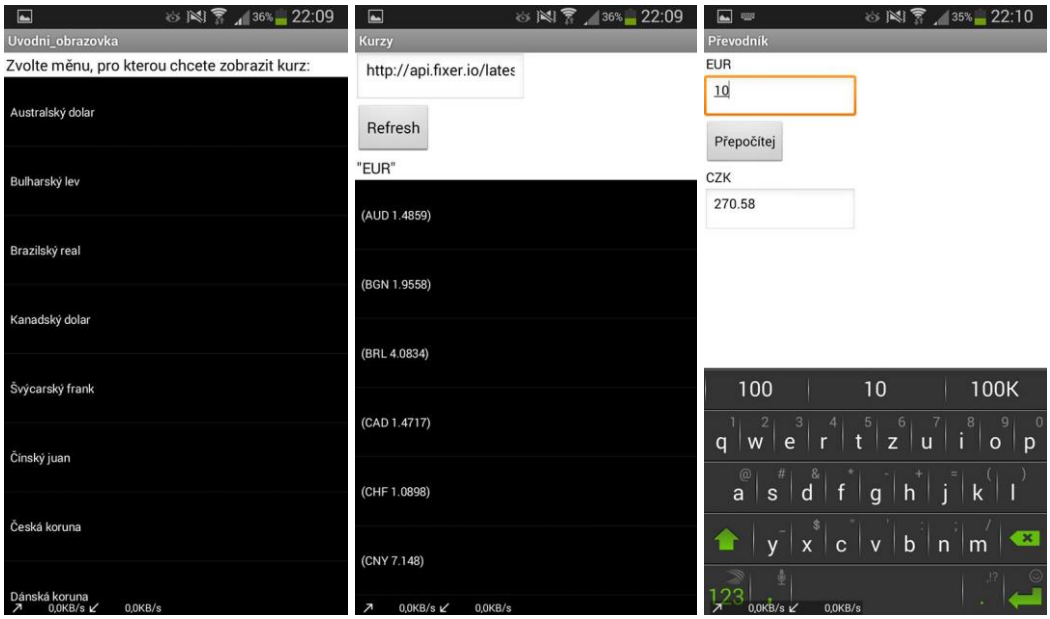
Nyní už jen potřebujeme ošetřit, co se stane v případě, že uživatel klikne na tlačítko a chce přepočíst měnu. Vyřešili jsme to jednoduchým způsobem a to takovým, že v případě, že je zmáčknuto tlačítko tak se vezme hodnota, která byla zadaná do pole TextBox1 a vynásobí se hodnotou, kterou nám posílá předchozí obrazovka s indexem jedna, jinými slovy vezme ze startovních hodnot tu, která v sobě nese číselnou hodnotu vybraného kurzu. Posledním krokem je, že se do textBoxu2 nastaví nově vypočítaná hodnota.



Obrázek 17 - Ošetřené kliknutí na tlačítko (zdroj: vlastní)

4.1.8 Testování

Hotová aplikace byla testována na mobilní zařízení Samsung Galaxy S3 pomocí funkce App Inventoru spojit se se zařízením přes Wi-Fi. K testování byla využita oficiální aplikace MIT AI2 Companion. Aplikace nebyla nahrána na Google play vzhledem k jednoduchosti a vytváření pouze pro edukativní účely. Následující obrázky zobrazují výslednou aplikace na zmíněném zařízení.



Obrázek 18 - Finální aplikace vytvořená v App Inventoru 2 (zdroj: vlastní)

5 Zhodnocení výsledků

Praktická část této práce byla zaměřena na vytvoření dvou podobných aplikací v různých vývojových prostředích. První verze aplikace s názvem Kurzovní lístek byla vytvořena ve vývojovém prostředí Android Studio, za pomoci programovacího jazyka a značkovacího jazyka XML. Na základě toho, že pro vývoj aplikace v tomto prostředí je potřeba alespoň základní znalost jazyka Java a dále z důvodu dostupných materiálu převážně v anglickém jazyce byla aplikace ohodnocena jako středně obtížná. Vývoj takovéto aplikace je vhodný pro programátora, který buďto má již nějaké znalost s programováním, nebo je ochoten se jim naučit, proto pro běžného uživatele tuto metodu nedoporučuji.

Na rozdíl od toho aplikace vytvořená ve vývojovém studiu App Inventor 2 je podstatně jednodušší, než aplikace v Android Studiu. K vytvoření této aplikace totiž nebylo za potřebí žádných znalostí z oblasti programování. Vzhledem k tomu i časová dotace potřebná k vytvoření je podstatně kratší než u Android Studia. Při vytváření aplikace se pohybujeme v rámci hodin pro App Inventor a v rámci dnů, až týdnů pro Android Studio, pro středně náročný projekt. Tyto časy také odráží fakt, že v App Inventoru nejsou takové možnosti a proto aplikace musí být zjednodušovány.

Vytvoření druhé aplikace v App Inventoru dokazuje, že toto prostředí je vhodné pro každého, kdo chce vytvořit rychle jednoduchou aplikaci. Oproti Android Studiu je ovšem notně omezeno ve vývojových prostředcích, které máme dostupné. Naprostému začátečníkovi může například oproti Android Studiu chybět možnost přidat si vlastní logy do kódu a tak zjistit jestli se daný kód provede správně, nebo krokování kódu, kdy program prochází krok po kroku a sleduje průběh. Takové funkce App Inventor nenabízí a začátečníkovi se může stát, že pokud udělá někde chybu tak jí nebude moct jednoduchým způsobem odhalit.

Vývojové prostředí App Inventor je dostatečné pro vývoj aplikace běžného uživatele, který nemá zájem na učení se programovacího jazyka a velkým vývojovým prostředím, ovšem očekává alespoň potřebu základního logického myšlení. Vhodné je i pro začínající programátory, kterým bych ale doporučoval po pár testovacích aplikacích přejít na plnohodnotná

vývojová studia. Naopak Android Studio je vhodné pro zkušené programátory v jazyce Java, ale i pro začátečníky, kteří jsou ochotni se v tomto jazyce zlepšovat. Nicméně Android Studio se stává nezbytností v případě tvoření komplexnějšího většího projektu.

6 Závěr

Teoretická část této práce byla zaměřena na popsání platformy Android, obecné seznámení s historií tohoto operačního systému a rozepsání jednotlivých verzí se zachycením důležitých funkcí. Podrobněji byla rozebrána hlavní komponenta pro distribuci aplikací, kterou je Google play. Dále teoretická část pojednává o konkurenčních operačních systémech iOS a Windows Phone a seznamuje uživatele s výhodami těchto systému, které by měl uživatel zvážit před rozhodnutím pro Android. V závěru teoretické části jsou popsány dvě autorem vybrané vývojové prostředí Android Studio a App Inventor. Tato vývojová prostředí jsou podrobně popsána, aby s nimi byl čtenář seznámen před praktickou částí.

Na základě znalostí získaných v teoretické části, byla vypracována část praktická, která se zaobírá vytvořením dvou obdobných aplikací pro operační systém Android, pro dvě rozdílná vývojová prostředí. Hardware využitý pro vytváření aplikací byl osobní počítač s operačním systémem Windows 10 a chytrý mobilní telefon Samsung Galaxy S3 s operačním systémem Android 4.2.2 Jelly Bean. První ze dvou aplikací byla vytvořena ve vývojovém prostředí Android Studio za využití značkovacího jazyka XML a programovacího jazyka Java. Vývoj v tomto prostředí byl doporučen vývojářům s alespoň základní znalostí jazyka Java.

Druhá aplikace byla vytvořena na webovém vývojovém prostředí App Inventor 2. Vytvořením této aplikace bylo dokázáno, že App Inventor díky absenci programování je vhodný pro kohokoliv, kdo si chce vytvořit jednoduchou aplikaci v časovém rámci několika hodin. Zároveň bylo zhodnoceno, že tento vývojový software není vhodný pro náročnější projekty.

Android se svým postavením na trhu dokazuje, že vývoj aplikací je jedním z hlavním stavebních kamenů pro oblíbenost operačního systému. Android se stal dominantním na trhu právě z důvodu vytváření příjemného prostředí pro vývojáře a otevřeností systému. S přílivem vývojářů také exponenciálně přibývaly více či méně kvalitní aplikace a vytvořily tak dnes obří Google Play. S opačným problémem se potýká Windows Phone, který s nedostatečným počtem aplikací odrazuje zákazníky. Ačkoliv by se podle podílů mobilních zařízení se systémem Android na trhu mohlo zdát, že brzy Android celou kon-

kurenci vytlačí, není tomu tak. Ačkoliv společnost Apple se svým operačním systémem iOS nedominuje v počtu prodaných zařízení, tak Android překonává v tržních číslech a to hlavně z důvodu držení si standardu a nevpouštění na trh levná zařízení. Z toho důvodu není pravděpodobné, že v blízké budoucnosti jedna z těchto společností absolutně převládne nad trhem mobilních operačních systémů.

Hlavní cíl této bakalářské práce, seznámit čtenáře s operačním systémem Android a vyhodnocení rozdílů dvou různých vývojových prostředí na tento systém, byl splněn.

7 Seznam použitých zdrojů

1. ANDROID CENTRAL. *Android Pre-history* [online][cit. 2. 12. 2015]. Dostupné z: <http://www.androidcentral.com/android-pre-history>
2. ANDROID CENTRAL. *Android early days* [online][cit. 4. 12. 2015]. Dostupné z: <http://www.androidcentral.com/androids-early-days>
3. ANDROID CENTRAL. *Android makes it big* [online][cit. 5. 12. 2015]. Dostupné z: <http://www.androidcentral.com/android-makes-it-big>
4. ANDROID CENTRAL. *Android Everywhere* [online][cit. 7. 12. 2015]. Dostupné z: <http://www.androidcentral.com/android-everywhere>
5. ANDROID TIP. *Od Android marketu až po Google play*[online][cit. 10. 12. 2015]. Dostupné z: <http://www.androidtip.cz/android-marketu-az-google-play-pohled-historie-nejvetsiho-online-obchodu-aplikacemi-android>
6. ANDROID. *The story of android* [online][cit. 15. 12. 2015]. Dostupné z: <https://www.android.com/history>
7. SVĚT ANDROIDA. *Android ovládl trh a předhnal sytém iOS* [online][cit. 20. 12. 2015]. Dostupné z: <http://www.svetandroida.cz/android-vs-apple-obsazeni-201510>
8. MOBIL IDNES. *Vyrábět Androidy je charita* [online][cit. 2. 1. 2016]. Dostupné z: <http://mobil.idnes.cz/apple-ma-z-ios-vice-nez-googlu-z-androidu-fdq-/mob_tech.aspx?c=A150226_155702_mob_tech_LHR>
9. IPHONE MANIA. *Jablečná historie: od Apple I. po iPhone (1. kapitola)* [online][5. 1. 2016]. Dostupné z: < http://iphonemania.mobilmania.cz/?q=node/96>
10. IPHONE MANIA. *Jablečná historie: od Apple I. po iPhone (3. kapitola)* [online][5. 1. 2016]. Dostupné z: < http://iphonemania.mobilmania.cz/?q=node/95>
11. APP MAGAZIN. *V čem je iOS lepší než Android* [online][7. 1. 2016]. Dostupné z: <http://www.app-magazin.cz/slider/nazor-servismana-ios-a-android-zarizeni-v-cem-je-ios-lepsi-nez-android/>
12. CNEWS. *OS Windows Mobile/Phone: strmá cesta historií* [online][9. 1. 2016]. Dostupné z: <http://www.cnews.cz/os-windows-mobilephone-strma-cesta-historii>

13. CNEW. *OS Windows Mobile/Phone: strmá cesta historií* [online][9. 1. 2016]. Dostupné z: < http://www.cnews.cz/os-windows-mobilephone-strma-cesta-historii/strana/0/2>
14. MS POWER USER. *A history of Windows Phone* [online][10. 1. 2016]. Dostupné z: <http://mspoweruser.com/a-history-of-windows-phone-the-road-to-threshold/>
15. VŠE O WINDOWS PHONE. Windows Phone nebo Android [online][12. 1. 2016] Dostupné z: <http://www.windowsphone9.cz/windows-phone-nebo-android/>
16. MOBIL IDNES. Windows Phone 8 test [online][12. 1. 2016]. Dostupné z: <http://mobil.idnes.cz/windows-phone-8-test-0qr-/telefony.aspx?c=A121102_163805_telefony_vok>
17. DEVELOPER ANDROID. *Android Studio Overview* [online] [15. 1. 2016]. Dostupné z: <http://developer.android.com/tools/studio/index.html>
18. DEVELOPER ANDROID. *Features* [online][15. 1. 2016]. Dostupné z: <http://developer.android.com/tools/studio/studio-features.html>
19. DEVELOPER ANDROID. *String Resources* [online][17. 1. 2016]. Dostupné z: <http://developer.android.com/guide/topics/resources/string-resource.html>
20. DEVELOPER ANDROID. *Layouts* [online][20. 1. 2016]. Dostupné z: <http://developer.android.com/guide/topics/ui/declaring-layout.html>
21. DEVELOPER ANDROID. *Relative Layout* [online][20. 1. 2016]. Dostupné z: <http://developer.android.com/guide/topics/ui/layout/relative.html>
22. DEVELOPER ANDROID. *Linear Layout* [online][20. 1. 2016]. Dostupné z: <http://developer.android.com/guide/topics/ui/layout/linear.html>
23. DEVELOPER ANDROID. *App Manifest* [online][23. 1. 2016]. Dostupné z: <http://developer.android.com/guide/topics/manifest/manifest-intro.html>
24. TECH TARGET. *Java* [online][24. 1. 2016]. Dostupné z: <http://searchsoa.techtarget.com/definition/Java>
25. SCHILDT, Herbert. *Java 7: výukový kurz*. 1. vyd. Brno: Computer Press, 2012. ISBN 978-80-251-3748-2.
26. DIMARZIO, J. *Programujeme hry pro Android 4*. 1. vyd. Brno: Computer Press, 2012. ISBN 978-80-251-3754-3.

8 Přílohy

Přílohy vzhledem k obsáhlosti zdrojových kódů jsou dostupné v přiloženém zabaleném zip souboru nebo na optickém disku.