Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačního inženýrství

Bakalářská práce

Vývoj aplikací pro platformu Android

Michael Škrlant

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Michael Škrlant

Informatika

Název práce

Vývoj aplikací pro platformu Android

Název anglicky

Application development for Android OS

Cíle práce

Cílem práce je porovnat možnosti vývoje aplikací pro mobilní OS Android s využitím rozdílných vývojových nástrojů a programovacích jazyků. Dílčím cílem je porovnat implementaci konkrétní ukázkové aplikace ve vybraných jazycích.

Metodika

Metodika práce je založena na studiu a analýze odborných informačních zdrojů. Na základě syntézy zjištěných poznatků budou popsány jednotlivé alternativní vývojové prostředky a jazyky pro tvorbu aplikací pro Android OS. Dále bude v několika vybraných prostředcích implementována totožná ukázková aplikace. Na základě poznatků získanách při jednotlivých implementacích budou shrnuty rozdíly mezi jednotlivými prostředími a způsobem implementace.

Doporučený rozsah práce

35-40 stran

Klíčová slova

programování, android, java, c#, xamarin, mobilní aplikace

Doporučené zdroje informací

Android 4. Computer Press, 2013. ISBN 978-80-251-3782-6.
C# 2005 Programujeme profesionálně. Computer Press, 2006. ISBN 80-251-1181-4.
UJBÁNYAI, Miroslav. Programujeme pro Android. Grada, 2013. ISBN 978-80-247-4863-4.

Předběžný termín obhajoby

2015/16 ZS – PEF

Vedoucí práce

Ing. Jiří Brožek, Ph.D.

Garantující pracoviště

Katedra informačního inženýrství

Elektronicky schváleno dne 10. 11. 2014

Ing. Martin Pelikán, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 10. 11. 2014

Ing. Martin Pelikán, Ph.D.

Děkan

V Praze dne 20. 11. 2015

Čestné prohlášení

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

V Praze dne 25.11.2015 _____

Poděkování

Rád bych touto cestou poděkoval panu Ing. Jiřímu Brožkovi, Ph.D. za vedení a cenné rady poskytované při zpracování této bakalářské práce. Dále děkuji rodičům za podporu, kterou mi během studia a psaní této práce poskytli.

Vývoj aplikací pro platformu Android

Application development for Android OS

Souhrn

Tato bakalářská práce představuje vývoj aplikace na mobilní zařízení s operačním systémem Android včetně následné tvorby aplikací. Teoretická část se zabývá historií a architekturou operačního systému Android, možnostmi různých vývojových prostředí, programovacími jazyky, které využívají, a možnostmi publikování aplikace. Praktická část využívá těchto znalostí k tvorbě dvou totožných aplikací ve vývojových prostředích Android Studio a Visual Studio s pluginem Xamarin. Jsou zde popsány jednotlivé komponenty aplikací s příklady implementace kódu. Výsledkem jsou dvě totožné aplikace a porovnání vývoje aplikace pomocí různých vývojových prostředí.

Klíčová slova

Android, Programování, Mobilní aplikace, Java, C#, Xamarin

Summary

This bachelor thesis presents the development of applications on mobile devices running Android OS, including creation of applications. The theoretical part deals with the history and architecture of the operating system Android, the possibilities of different development environments, programming languages they use, and the possibilities of publishing applications. The practical part uses this knowledge to create two identical applications in development environments Android Studio and Visual Studio with plugin Xamarin. All components of the applications are described, including the code implementation examples. The result is two identical applications and comparing the development of applications using different development environments.

Keywords

Android, Programming, Mobile application, Java, C#, Xamarin

Obsah

1. Úvod

Mobilní telefony jsou jednou z nejrychleji se vyvíjejících technologií. Před patnácti lety uměly telefony jenom volat a posílat SMS. Dnes jsou telefony multifunkční zařízení, které dokáží nahradit osobní počítače a mají neustálý přístup k internetu. Jejich funkčnost a možnosti rozšiřují aplikace. Trh s mobilními aplikacemi roste den ode dne a ukazuje, že má velkou budoucnost. Tato práce se bude zabívat vývojem aplikací pro aktuálně nejrozšířenější operační systém na mobilních telefonech.

V této práci bude probrána problematika vývoje operačního systému mobilních zařízení Android. Také budou probrány možnosti vytváření aplikací v různých vývojových prostředích a jejich šíření. Závěrem bude vše demonstrováno na aplikaci zobrazující články ze servru Agris.cz.

2. Cíl a metodika

Cílem práce je probrat historii a architekturu operačního systému android. Dále představit různé možnosti vývoje pro tento operační systém. Tyto nasbírané znalosti využít v praktické části a s jejich pomocí vytvořit mobilní aplikaci v různých vývojových prostředích. Na základě získaných poznatků při tvorbě jednotlivých aplikací, shrnout rozdíly mezi jednotlivými vývojovými prostředími.

V teoretické části je představen operační systém Android. Pro tuto část jsou informace čerpány z internetových zdrojů a z literatury uvedené ve zdrojích. Vzhledem k rychlému zastarávání tištěných zdrojů byl jako hlavní zdroj informací zvolen internet. V praktické části je demonstrována totožná aplikace ve dvou různých vývojových prostředích. K programování byla především využita dokumentace z oficiálních stránek, sekundárně pak informace z programátorských fór.

3. Teoretická východiska

3.1 OS Android

Android je operační systém pro mobilní telefony, tablety, GPS navigace a další zařízení. Je to open source systém založený na jádře Linuxu. Protože se jedná o otevřený systém, mají všichni uživatelé a výrobci přístup ke zdrojovému kódu, který si mohou upravit dle vlastních potřeb. Je to univerzální systém, který se lze přizpůsobit hardwarové výbavě různých zařízení. Dokáže pracovat na různých procesorech a rozlišeních displeje. Díky této vlastnosti se tento operační systém rozšířil nejen na mobilní telefony a tablety, ale také na GPS navigace, televize, fotoaparáty, multimediální centra a netbooky.

Obrázek 1 - Logo Android

3.1.1 Historie Androida

Operační systém android začala vyvíjet firma Android Inc. Založená v roce 2003. V srpnu 2005 tuto firmu odkoupil Google Inc. A udělal z ní svojí dceřinou firmu. Google později 5. listopadu 2007 vytváří konsorcium „Open Handset Alliance“. Je to seskupení výrobců mobilních telefonů za cílem vyvinout standard pro mobilní zařízení. V tomto seskupení se nachází například Samsung, HTC, LG, Sony, NVIDIA, Intel, Dell, T-Mobile, Vodafone. Ten samý den Google oznámil první verzi mobilního operačního systému Android.

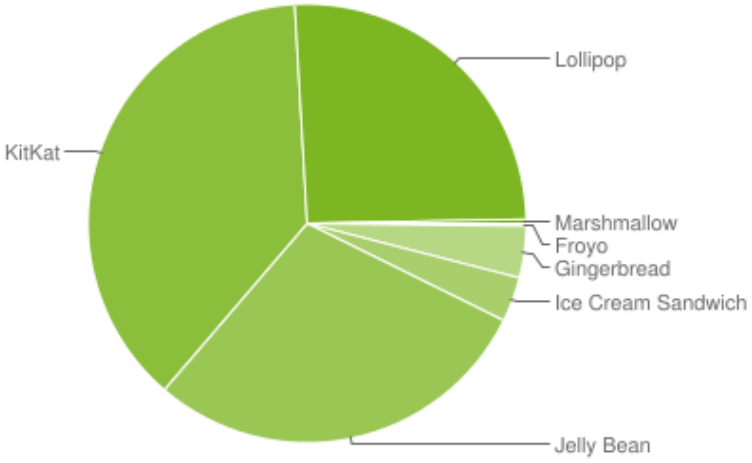
Prvním mobilním telefon s Androidem se stal HTC Dream, v některých státech známý pod názvem T-Mobile G1. Byl představen v říjnu 2008 a na český trh se dostal počátkem roku 2009. Byl to dotykový telefon s hardwarovou qwerty klávesnicí, displejem o rozlišení 320x480 pixelů, procesorem Qualcomm a s fotoaparátem o rozlišení 3,2 Megapixelů.[1]

3.1.2 Fragmentace verzí

Jednou z mála nevýhod systému android je fragmentování na různé verze systému na zařízeních, které vlastníci aktuálně používají. Důvodem této fragmentace jsou různé hardwarové zdroje jednotlivých zařízení. Od oznámení nové verze Androidu Googlem trvá půl roku až rok, než ji výrobci telefonů použijí v nových zařízeních. Výrobci telefonů používají pro každý model různé procesory, paměti, kamery a další hardwarové části. Pro každý odlišný model telefonu tedy musí výrobci na míru upravit originální verzi Androidu od Googlu. Protože je tento proces celkem náročný, tak dostane většina telefonů během své životnosti pouze jeden upgrade na novou verzi systému Android. Vývojáři aplikací pro tento systém se díky této fragmentaci verzí, musí rozhodnout, jestli budou využívat nejnovější prostředky z poslední verze, nebo jestli udělají aplikaci kompatibilnější se staršími verzemi systému na úkor těchto nových prostředků.[2]

Verze	Název	API	Distribuce
2.2	Froyo	8	0.2%
2.3.3-2.3.7	Gingerbread	10	3.8%
4.0.3-4.0.4	Ice Cream Sandwich	15	3.3%
4.1.x	Jelly Bean	16	11.0%
4.2.x		17	13.9%
4.3		18	4.1%
4.4	KitKat	19	37.8%
5.0	Lollipop	21	15.5%
5.1		22	10.1%
6.0	Marshmallow	23	0.3%

Tabulka 1 - Distribuce verzí Androidu [2]



Obrázek 2 - Graf distribuce verzí Androidu [2]

3.1.3 Historie verzí

První verze Androidu 1.0 byla vydána na telefonu HTC Dream v září 2008. Obsahovala jen základní výbavu, jako je webový prohlížeč, fotoaparát, widgety na domovské stránce, notifikace, synchronizace kalendáře a kontaktů s Googlem, multimediální přehrávač, mapy Google. Tato verze Androida byla závisla na hardwarových tlačítkách svého zařízení. Tato verze byla v únoru 2009 aktualizována na verzi 1.1, která především řešila nalezené chyby.

Koncem dubna 2009 vyšla verze 1.5 Cupcake. Byla to první verze, u které Google započal tradici nazívání verzí podle cukrovinek. V této verzi přibyla klávesnice na obrazovce, čímž zmizela potřeba hardwarové klávesnice u nových zařízení. Další velká novinka byla schopnost nahrávat a přehrávat video, která u předchozích verzí chyběla. Tato verze také už uměla automaticky otáčet obsah obrazovky podle polohy telefonu. Za zmínku ještě stojí podpora widgetů, které lze měnit a podpora kopírování, vkládání ve webovém prohlížeči.

Obrázek 3 - Obrazovka s widgety na Androidu 1.5 Cupcake
Zdroj:http://androidzone.org/2013/05/historia-de-android-la-evolucion-a-lo-largo-de-sus-versiones/

Následující verze Android 1.6 Donut uvedena v září 2009, přinesla podporu pro různá rozlišení a poměry stran obrazovky, díky čemuž se mohla rozšířit do dalčích zařízení. Tato verze zavedla nový Android Market, nové prostředí fotoaparátu a galerie. Přibyla podora gest a syntéza řeči.

Měsíc poté v říjnu 2009 přišla velká aktualizace Android 2.0/2.1 Eclair tato verze přinesla celkové zrychlení systému a podporu větších rozlišení displeje. Webový prohlížeč získal nové prostředí a podporu HTML5. Fotoaparát získal digitální zoom a podporu přisvětlovací diody. Byl aktualizován seznam kontaktů a Mapy Google. Přibily animované tapety a Bluetooth 2.1. S tímto Androidem přicházely na trh už i high-endové modely telefonů jako HTC DEsire, Motorola Droid, Samsung Galaxy S a také vůbec první telefon od Googlu, Nexus One.

Další nová verze Androidu byla vydána až v květnu 2010 a to ve verzi 2.2 Froyo. Díky kompilátoru JIT (just-in-time) Dalvik se podařilo zvýšit rychlost spouštění aplikací dvakrát až přetkrát. Dále byla vylepšena správa paměti RAM. Přibyla možnost instalovat aplikace na paměťovou kartu a možnost vytvořit z telefonu WiFi hotspot, nebo sdílet připojení k internetu přez kabel USB. Také přibyla podpora Adobe Flash pomocí distribuce přes Android Market.

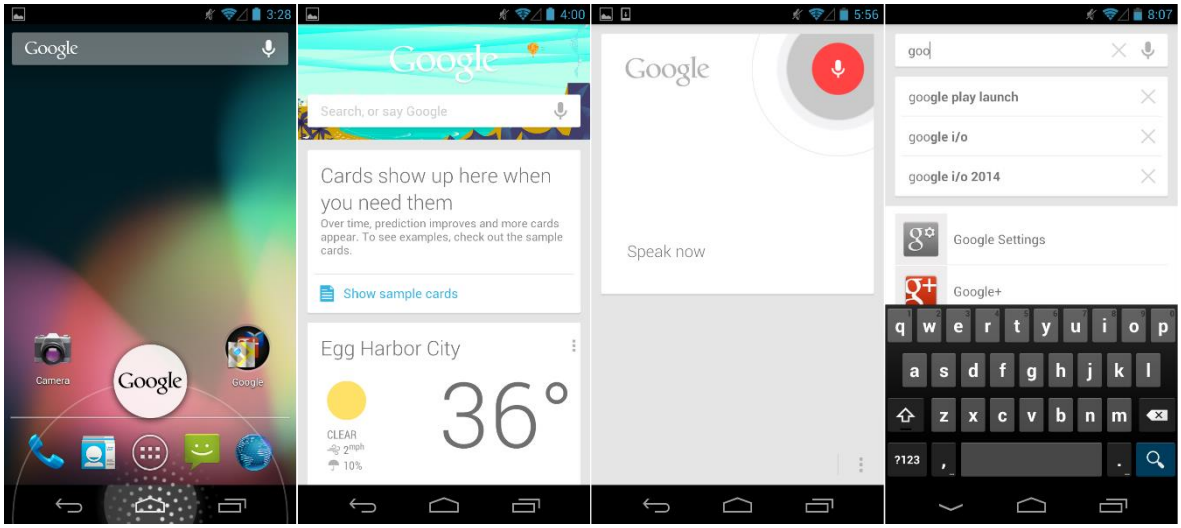
6. prosince 2010 následovala verze 2.3/2.4 Gingerbread. Tato verze zavedla podporu NFC, podporu obrazovek s vysokým rozlišením, nativní podporu senzorů, jako jsou barometr a gyroskop. Byla vylepšená správa napájení, virtuální klávesnice a funkce kopírovat a vložit. Přibyla podpora kamery na čelní straně telefonu a podpora přehrávání videí ve formátech WebM/VP8. Vylepšen byl Garbage colector, který se stará o uvolňování prostředků systému tím, že ukončuje nepoužívané programy a jejcih části. Byla zavedena také řada kosmetických změn pro zjednodušení a urychlení prostředí. S touto verzí Androida byl vydán druhý telefon od Googlu řady Nexus a to telefon Nexus S vyráběný Samsungem.

Během roku 2010 značně vzrostla popularita tabletů, díky tabletu iPad spolčnosti Apple. Výrobci tedy na trh chtěli uvést své vlastní tablety, ale Android 2.3 gingerbread pro ně nebyl vhodný. Podporoval sice displeje s vysokým rozlišením, ale počítal s malými rozměry. Proto vznikla verze Androidu 3.0 Honeycomb, která se kromě drobných změn moc od předchozí verze nelišila. Jediných velkých změn se dočkalo uživatelksé rozhraní, které bylo upraveno pro displeje velkých rozměrů. Tato verze Androida nebyla moc populární a výrobci tak raději nabízeli tablety se starší verzí systému.

Po neúspěšném předešlém pokusu vydat Android přímo pro tablety, se Google vydal cestou vytvořit systém Android, který bude vhodný jak pro mobilní telefony, tak i pro tablety a další zařízení. Výsledek byl Android 4.0 Ice Cream Sandwich, který byl vydán 19. října 2011. Tato verze byla vylepšenou verzí Androidu 3.0 Honeycomb, která byla zmenšená pro mobilní telefony. Zavedla mnoho kosmetických úprav, včetně softwarových tlačítek navigace. Uveden byl ukazatel přenesených dat a funkce Android Beam pro přenos souborů pomocí NFC. S touto verzí ukončil Google podporu Adobe Flash v nativním prohlížeči.

Android 4.1/4.2/4.3 Jelly Bean byla série aktualizací zaměřená na vylepšení designu a plynulosti uživatelského rozhraní, kde se Google snažil eliminovat prodlevy systému a zrychlit jeho odezvu. Novinkou byly chytré karty Google Now, zobrazující uživateli informace, o kterých se předpokládá, že je uživatel bude chtít na základě jeho návyků. Například zobrazení stavu dopravy před cestou do práce. Dále přibyla možnost mít více uživatelských účtů a rozpoznávání hlasu offline.

Koncem října 2013 byl Představen Android 4.4 KitKat. Tato verze systému se zaměřila hlavně na vzhled, kde došlo patrně k největším změnám. Přinesla hlavně optimalizaci pro zařízení s menší pamětí RAM. Přibyla možnost použít nového virtuálního stroje Android Runetime – ART místo virtuálního stroje Dalvik, díky lepšímu překladu programů by mělo dojít ke snížení spotřeby energie a zrychlení aplikací. Přibyla možnost zvolit si aplikaci pro správu SMS a domácí obrazovky.



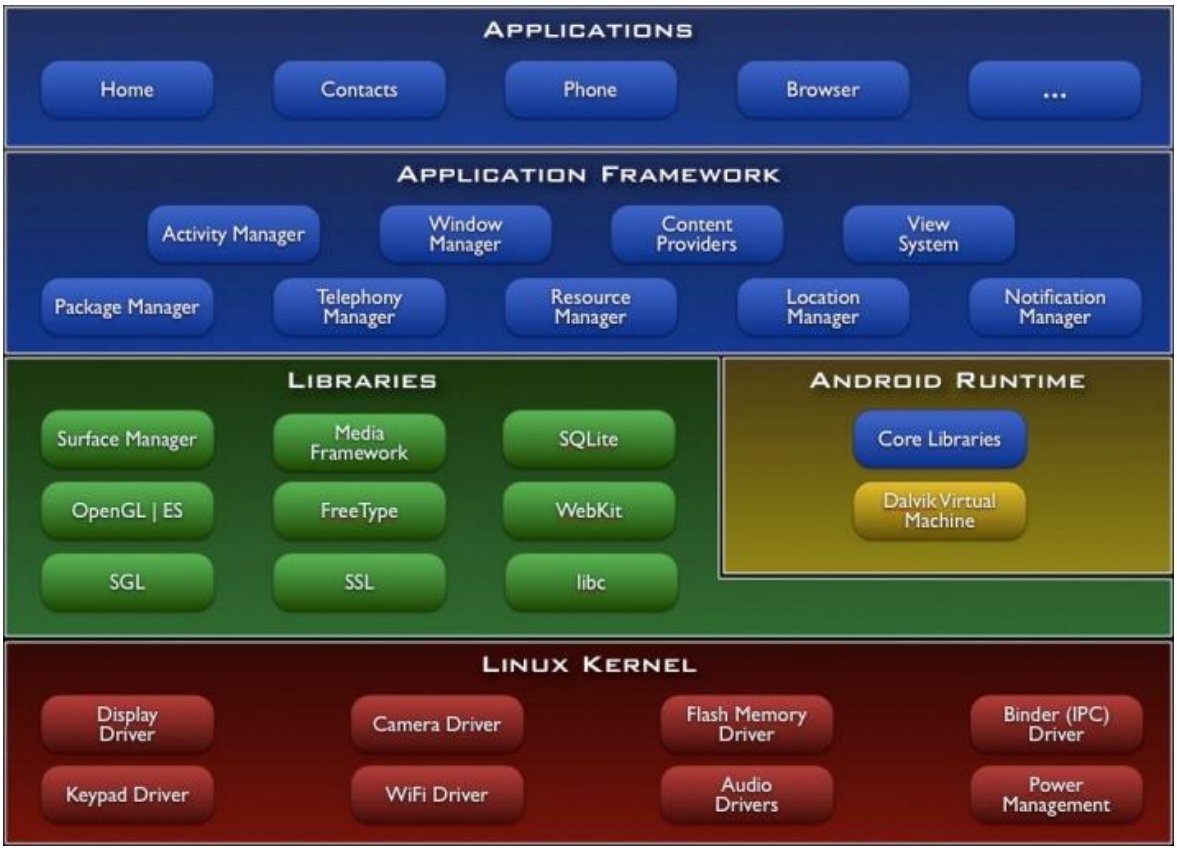
Obrázek 4 - Android 4.3 Jelly Bean s kartami Google Now [3]

Android 5.0/5.1 Lollipop vyšel 12. listopadu 2014. Přinesl velké změny ve vzhledu a to hlavně díky použití nového designu "Material design" a inovované oznamovací oblasti. Vylepšená byla spotřeba energie díky projektu Volta. Virtuální stroj ART plně nahradil virtuální stroj Dalvik. Příbyla podpora 64bitových procesorů, uživatelsky nastavitelné priority pro oznámení aplikací, podpora náhledu při tisku a funkce chytrého zamykání telefonu s vylepšenou uzamčenou obrazovkou.[3]

Poslední a nejnovější verzí je Android 6.0 Marshmallow dostupný od 5. října 2015. Marshmallow se hlavně zaměřuje na vylepšování uživatelského rozhraní a komfort uživatele. Největší změnou je nové nastavení oprávnění aplikacím, kde lze aplikacím povolovat jednotlivá oprávnění. Funkce Google Now se dočkala vylepšení a lepší integrace do systému. Přibyl nový režim úspory energie Doze, který dokáže přivést zařízení do módu spánku. Nová je taky podpora senzoru otisku prstů a seznam aplikací. Výrazně lepší je také podpora a integrace paměťových karet.[4]

3.1.4 Architektura

Architektura operačního systému Android je rozdělena do pěti vrstev. Každá z těchto vrstev má svůj účel a provádí odlišné operace. Ve skutečnosti ale dochází k prolínání mezi vrstvami a k jejich spolupráci. Vrstvy tak nejsou mezi sebou striktně odděleny.



Obrázek 5 - Architektura operačního systému Android

Zdroj: http://androidmarket.cz/android/jak-vypada-android-uvnitr-aneb-co-je-rom-kernel-bootloader-a-dalsi/

1. Jádro operačního systému (Linux Kernel)

Linux Kernel neboli jádro je nejnžší vrstva operačního systému. Kernel je základem operačního systému. Jeho hlavní funkce je zajistit komunikaci mezi hardwarem a softwarem, k tomu mu slouží Drivery (ovladače). Mimo to také zajišťuje kontrolu nad systémem a koordinaci činnosti všech běžících procesů, podporu správy paměti, správu napájení, zajišťuje síťové spojení atd. Při startu zařízení je jádro zavedeno do operační paměti a je mu předáno řízení. Operační systémy mají jádro postavené na Linuxu.

2. Knihovny (Libraries)

Druhou vrstvou architektury Android jsou knihovny (Libreries). Jedná se o řadu rozhraní API pro vývoj aplikací, například: android.util, android.webkit, android.database, android.widget, android.app. Dále se tu nachází nativní knihovny Androidu. Ty jsou napsány v C/C++ a jsou využívány různými komponenty systému. Tyto knihovny jsou vývojářům aplikací poskytnuty prostřednictvím Android Application Framework. Mezi tyto knihovny patří například: OpenGL, SQLite, SSL, SGL, FreeType, libc.

3. Virtuální stroj (Android Runtime)

Tato vrstva obsahuje virtuální stroj, který překládá kód programovacího jazyka Java do spustitelného kódu. Tato vrstva také obsahuje základní knihovny programovacího jazyka Java.

Do verze Androidu 4.3 byl virtuálním strojem Dalvik. Překlad aplikace pro Android, probíhá zkompilováním Java kódu do Java byte kódu pomocí Java kompilátoru. Poté se překompiluje Java byte kód do Dalvik byte kódu pomocí kompilátoru Dalvik. Tento výsledný Dalvik byte kód je pak spuštěn na virtuálním stroji Dalvik.

Od verze Androidu 4.4 je virtuálním strojem ART – Android RuneTime. Tento nový virtuální stroj přinesl zrychlení aplikací a úspory energie pomocí dopředné kompilace. Tato dopředná kompilace fungeje tak, že se aplikace napsaná ve zdrojovém kódu Java přeloží pomocí kompilátoru do Java byte kódu a následně se přeloží do byte kódu Dalvik. Při instalaci do Androidu se jednou provždy zkompiluje do nativního kódu procesoru zařízení. Pro zachování zpětné kompability se staršími aplikacemi je stále pčítomen původní kompilátor Dalvik.

4. Aplikační rámec (Application Framework)

Čtvrtá vrstva Application Framework je pro vývojáře aplikací nejdůležitější. Aplikační rámec obsahuje knihovny Java, které tvoří vlastní systémové API. Aplikační rámec umožňuje vývojářům přistoupit k nejrůznějším službám, které mohou vývojáři využívat přímo ve svých aplikacích. Ty jim následně dovolí například přistoupit na prvky, používat hardware zařízení (tlačítka, senzory), spouštět jiné aplikace na pozad nebo přistoupit k jejich

obsahu (kontakty), nastavovat alarmy, nebo přistoupit na prvky graficko-uživatelského rozhraní. Mezi základní služby aplikačního rámce patří:

- View Systém – Obsahuje prvky používané k sestavení graficko-uživatelského rozhraní, jako jsou textová pole, seznamy, tlačítka, přepínače atd.
- Content Providers – Umožnuje přistoupit a pracovat s obsahem jiných aplikací, jako například Kalendář, Kontakty atd.
- Resource Manager – Poskytuje přístup k „nekódovým“ zdrojům, jako jsou grafika, řetězce, přidané soubory.
- Notification Manager – Umožňuje všem aplikacím zobrazit vlastní upozornění (notifikaci) ve stavovém řádku.
- Activity Manager – Řídí životní cyklus aplikací, jejich start, průběh a ukončení. Poskytuje orientaci v zásobníku s aplikacemi.
- Package Manager – Poskytuje informace o aplikacích nahraných do operačního systému.

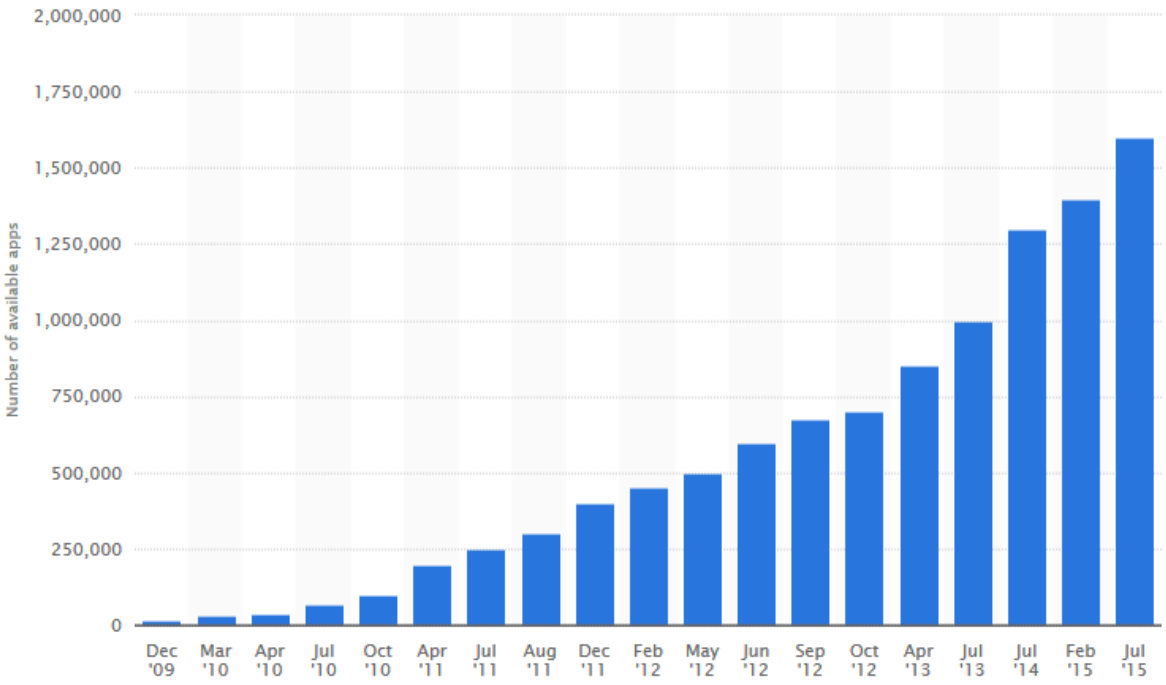
5. Aplikace (Applications)

Pátá tedy poslední nejvyšší vrstva představuje již samotné aplikace, které jsou využívány uživateli zařízení. Patří sem již předinstalované aplikace výrobcemi, tak i aplikace doinstalované uživateli.[5][6]

3.2 Vývoj aplikací

Aplikace pro operační systém Android jsou obvykle vyvíjeny v programovacím jazyce Java za pomocí oficiální sady Android SDK (software development kit), to však neznamená, že je nelze programovat v něčem jiném. Programátoři mají pro vytváření aplikací navýběr od jednoduchých vývojových prostředí, u kterých ani nemusejí mít znalost programovacích jazyků, až po prostředí složité a komplexní, které jim naopak dá větší kontrolu nad prostředky zařízení pro které je aplikace vyvíjena.

Vyvinuté aplikace jsou běžně dostupné ke stažení na službě Google Play (dříve Android Market), buď zcela zdarma, nebo za poplatek. V květnu 2012 byl počet aplikací dostuplných na Google Play 500 tisíc, v červenci 2013 počet takto dostupných aplikací překročil jeden milion. V červenci 2015 bylo na Google Play dostupných 1.6 milionu aplikací ke stažení.[7]

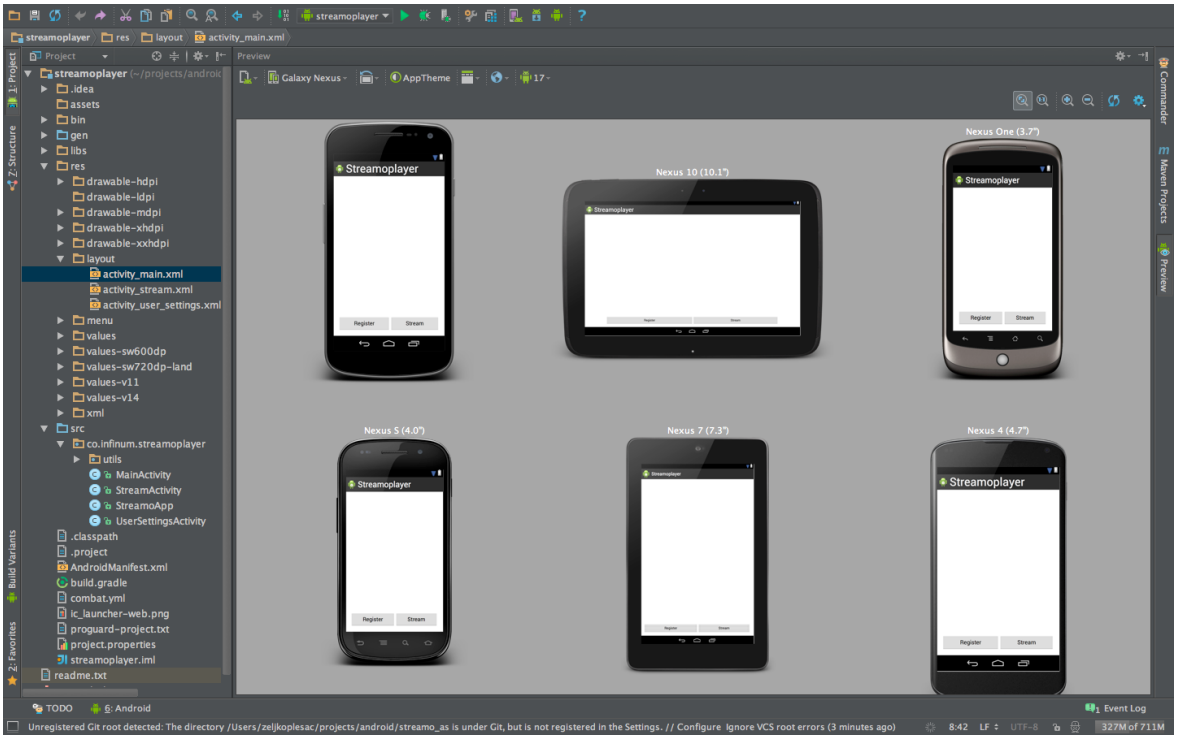


Obrázek 6 - Graf počtu dostupných aplikací na Google Play [7]

3.2.1 Android studio

Android Studio je vývojové prostředí od firmy Google založené na IntelliJ IDEA. Je to prostředí speciálně vytvořené pro vývoj aplikací na Android. Poprvé bylo představeno 16. května 2013, od kdy prošlo fází vývoje, až do prosince 2014 kdy byla představena první stabilní verze. Android Studio nahradilo Eclipse, které pro vývoj na android používalo plugin ADT (Android Development Tools), jako Googlem oficiálně podporované vývojové prostředí. S přechodem k Android Studiu byl ukončen vývoj pluginu ADT.[8] Vývojové prostředí Android Studio nabízí například:

- Šablony pro vytváření běžných aplikací.
- Bohatý editor s možností editovat uživatelské rozhraní metodou „drag and drop“ a možnost náhledu na vícero konfigurací obrazovek zařízení.
- Nástroje Lint pro kontrolu a testování kompability a odchytávání problémů.
- Zabudovaná podpora Google Cloud Platform
- Refactoring a rychlé opravy kódu specifického pro Android.
- Podepisování aplikací.



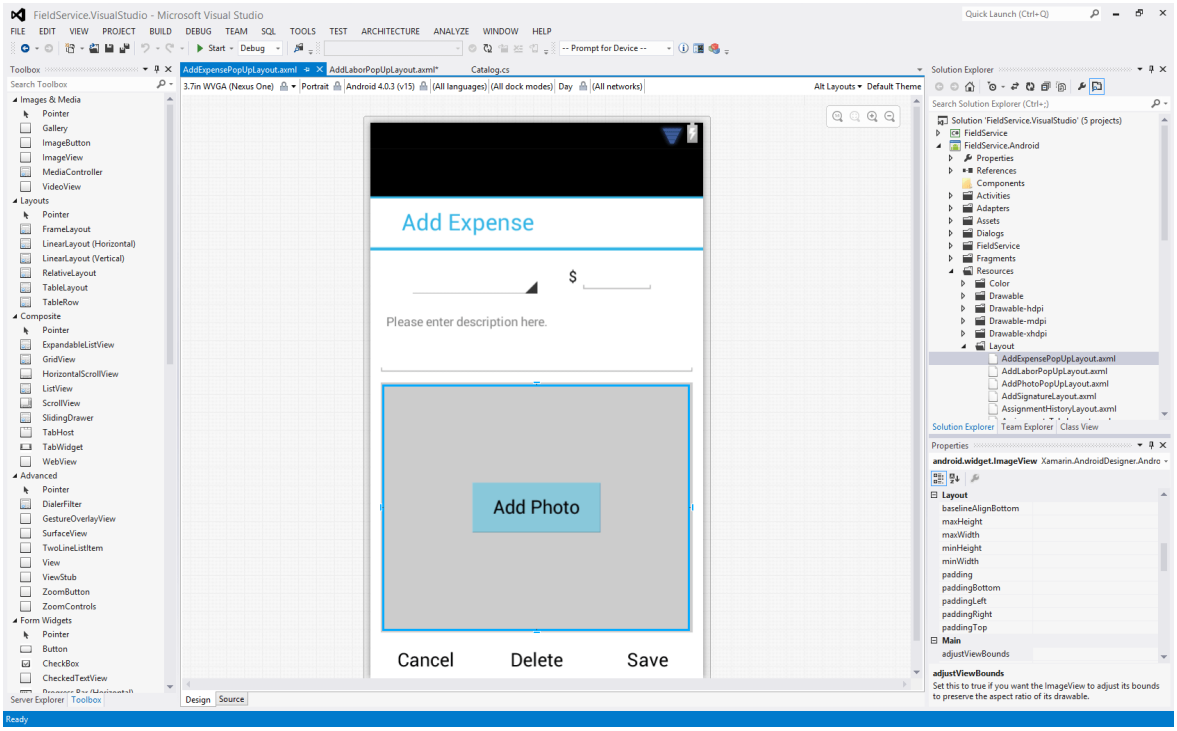
Obrázek 7 - náhled na vícero konfigurací obrazovek v Android Studiu

Zdroj: https://infinum.co/the-capsized-eight/articles/android-studio-vs-eclipse-1-0

3.2.2 Xamarin

Nástroje Xamarin jsou multiplatformní, tedy umožňují vytvářet aplikace pro telefony a tablety Android, iOS a Windows. Aplikace jsou v Xamarinu vytvářeny v programovacím jazyce C#. Nástroje Xamarin jsou dostupné jako vývojové prostředí Xamarin studio, nebo jako plugin ve vývojovém prostředí Visual Studio společnosti Microsoft.

Xamarin podporuje automatické dokončování kódu a nápovědu při jeho psaní, ve Visual Studiu k tomu využívá technolgii IntelliSense. Také nabízí možnost buildování a debuggování aplikací a jejich spuštění v simulátoru nebo na připojeném zařízení Android. Xamarin také jako Android Studio nabízí k tvorbě aplikací využití grafický editor uživatelského rozhraní s technologií „drag and drop“, která umožňuje umisťovat prvky grafického rozhraní jednoduchým kliknutím a přetažením myši.[9]



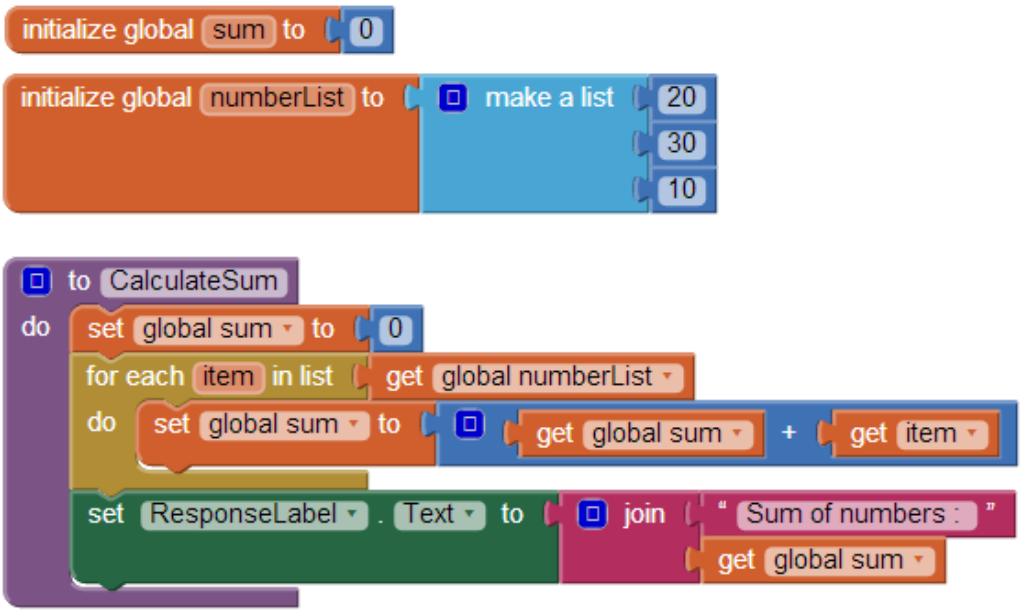
Obrázek 8 - Editor uživatelského rozhraní v Xamarin pro Visual Studio

Zdroj: http://www.xlsoft.com/jp/products/xamarin/images/visual-studio/

3.2.3 App Inventor

App Inventor je open-source webový nástroj na vytváření jednoduchých aplikací pro Android. Původně byl poskytován společností Google, jeho vývoj však převzal Massachusettský technologický institut (MIT). App Inventor je zaměřený především na nováčky v programování, protože k vytváření aplikací v něm není potřeba znát programovací jazyk jako je Java nebo C#. Toto ale nese sebou jistou nevýhodu v tom, že se v App Inventoru nedají vytvářet složité algoritmy a aplikace. Zajímavé v tomto nástroji je spouštění a debuggování aplikace, které probíhá v reálném čase a mohou se tak v reálném čase testovat prováděné změny.

App Inventor se skládá ze dvou částí. První část je App Inventor Designer. Je to editor uživatelského rozhraní s funkcí drag and drop podobný těm jako ve vývojových prostředí Xamarin a Android Studio. Druhá část je App Inventor Blocks editor sloužící k vytváření jednoduchých algoritmů. Ty se vytvářejí pomocí bloků představující jednotlivé prvky uživatelského rozhraní (tlačítka, textová pole atd.), texty, číselné promněné, seznamy, matematické a logické funkce (+, -, =, <, AND, OR, NOT, TRUE, atd.), řídící funkce (while, if, else, otevření a zavření nové obrazovky, atd.) a metody. Tyto bloky lze v grafickém editoru poskládat k sobě a vytvořit tak metodu nebo rovnou celou aplikaci.[10]



Obrázek 9 - Algoritmus na sečtení hodnot v seznamu, vytvořený pomocí bloků [10]

3.2.4 Programovací jazyk

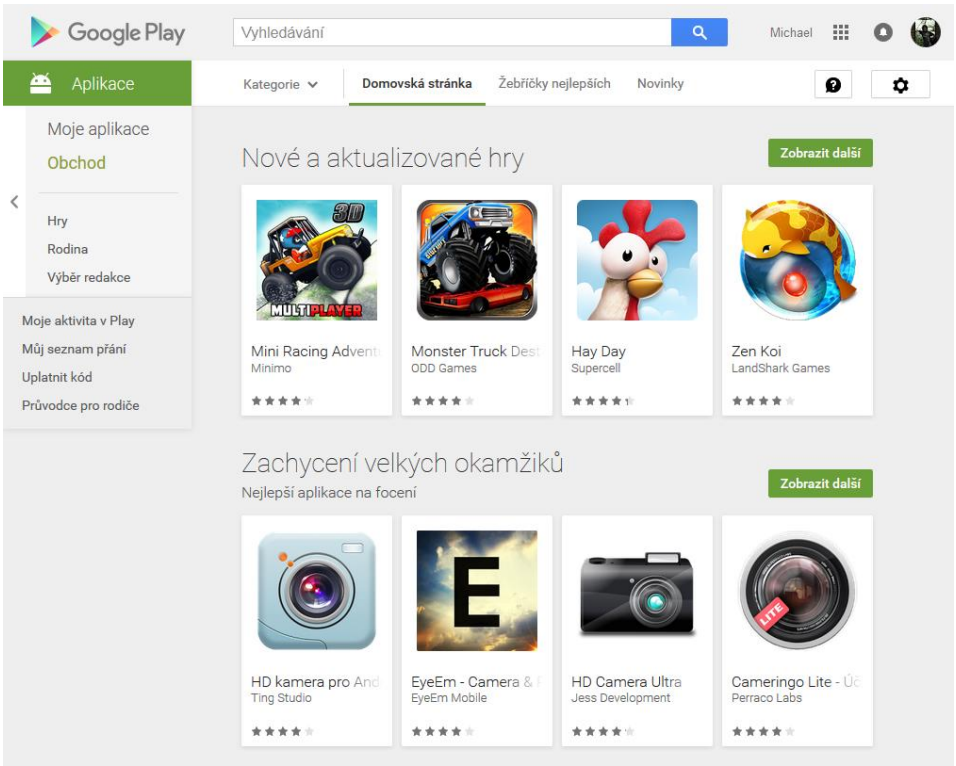
K vytváření aplikací pro Android ve vývojových prostředích Eclipse a Android Studio zvolil Google jazyk Java. Jedná se objektově orientovaný programovací jazyk, vytvořen jako derivace programovacích jazyků C a C++. Vytvořila ho v roce 1995 firma SUN, kterou později odkoupila firma Oracle. V květnu 2007 byly uveřejněny zdrojové kódy jazyka a tak od té doby je Java vyvýjena jako open source. Jednou z předností tohoto jazyka je jeho multiplatformnost. Java se tak vyskytuje na mobilních telefonech (Java ME), osobních počítačích (Java SE), tak i na firemních počítačích a sítích (Java EE). Multiplatformnost je dána tím, že při překladu se místo strojového kódu vytvoří byte kód (mezikód). Tento byte kód je nezávyslí na architektuře procesoru. Program pak lze spustit na libovolném počítačí nebo zařízení, který má vyrtuální stroj Javy (Java Virtual Machine - JVM) k interpretaci byte kódu. Na Androidu tuto interpretaci provádí virtuální stroj Dalvik. Tyto virtuální stroje mohou pracovat v režimu „just-in-time“, kdy se překládá do strojového kódu jen ten kód, který je zapotřebí. [11]

Při programování ve vývojovém prostředí Xamarin je použit objektově orientovaný programovací jazyk C#. Byl vyvinut v roce 2000 zároveň s platformou .NET Framework společností Microsoft. Jedná se o jazyk založen na programovacích jazycích C a C++. C# je považovaný za konkurenční jazyk programovacího jazyka Java, se kterým má mnoho společného.[12]

Veškeré knihovny v operačním systému Android jsou psány v C/C++. Pokud je zapotřebí, lze napsat vlastní knihovnu, která se zkompiluje a nainstaluje. Avšak toto je pouze doporučeno v případech, ve kterých by díky nové knihovně došlo k výrazné optimalizaci aplikace (práce s velkým objemem dat, složité výpočty, práce s grafikou).

3.2.5 Publikace aplikace

Vytvořené aplikace je zapotřebí nějakým způsobem distribuovat koncovým uživatelům. K tomu slouží obchod s Android aplikacemi Google Play. Ten nabízí vývojářům možnost nejen aplikace prodávat ale i bezplatně šířit. K publikaci aplikací na službě Google Play je zapotřebí vlastnit vývojářský účet, ten se získá jednorázovou platbou o výši $25. Náhrát lze pouze podepsané aplikace certifikátem identity autora aplikace. Google Play lze využít k distribuci alpha a beta verzí aplikací, které se automaticky aktualizují na zařízeních testerů. Po nahrání aplikace na Google Play je zapotřebí ke zveřejnění vyplnit údaje o aplikaci jako je název, krátký a úplný popis aplikace, zvolit typ a kategorii aplikace a přiřadit obrázky z aplikace. Následuje schvalování aplikace, které bylo nově zavedeno v březnu 2015. Nejdříve musí vývojář vyplnit dotazník o obsahu aplikace (zobrazení násilí, drog, sexu nebo použití vulgárních výrazů). Tento dotazník slouží k určení přístupnosti pro mladistvé a děti podle norem daných zemí, například ESRB (Entertainment Software Rating Board) pro americký trh a PEGI (Pan-European Game Information) pro Evropu. Následně aplikace prochází ruční kontrolou obsahu Googlem. Poté už jen vývojář zvolí, jestli bude aplikace zdarma nebo placená a nastavit ve kterých zemích bude dostupná.[13][14]



Obrázek 10 - Obchod Google Play

Druhou možností je publikovat v alternativních službách jako je například Amazon Appstore společnosti Amazon. Další takovou celkem populární službou je F-Droid, který je katalogem FOSS (Free and Open Source Software) aplikací. Mezi další takovéto služby patří GetJar, AppsLib, Slide ME, Mobogenie, Aptoide a jiné. Většina těchto služeb je dostupná ke stažení jako aplikace pro Android.

Třetí možností je šířit aplikace pomocí sdílení na webových stránkách nebo FTP servrech, odkud si je zájemci mohou stáhnout. Pro ulehčení přístupu k těmto souborům, lze třeba využít QR kódy s přímými odkazy ke stažení, které umí přečíst většina telefonů s OS Android.

Při instalaci aplikací z alternativních služeb nebo přímo ze staženého souboru .apk, je zapotřebí povolit v nastavení zařízení instalování z neznámých zdrojů.

Obrázek 11 - Hodnocení aplikace dle standardů daných zemí [14]

4. Vlastní práce

K vytvoření aplikace byly zvoleny vývojová prostředí Android Studio a Visual Studio s pluginem Xamarin. Dále se vlastní práce věnuje porovnání a shrnutí rozdílů mezi jednotlivými prostředími a způsobu implementace. Celé kódy aplikace jsou v příloze.

4.1 Popis Aplikace

Jako ukázková aplikace byla vybrána aplikace pro stažení a zobrazení článků ze serveru Agris.cz. Po otevření aplikace se zobrazí seznam článků. Uživatel si může stisknutím zobrazit celý článek. U položek u kterých je na servru Agris.cz pouze odkaz na původní článek, který je hostován na jiných servrech, má uživatel možnost stiskem tlačítka zobrazit původní článek ve webovém prohlížeči zařízení. Veškerá data jsou do aplikace stahována z internetu, díky tomu má uživatel přístup vždy k nejnovějším článkům.

4.2 Návrh uživatelského rozhraní

Uživatelské rozhraní aplikace je velmi jednoduché a přehledné. Skládá se ze dvou částí. První je seznam článků, které má uživatel na výběr k přečtení. Seznam zobrazí názvy článků, které po kliknutí zobrazí celý článek. Na konci zobrazeného seznamu se nacházejí dvě tlačítka pro načtení další nebo předchozí stránky seznamu. Druhou částí uživatelského rozhraní je zobrazení zvoleného článku. Při zobrazení článku uživatel uvidí název článku a datum zveřejnění, pod kterými se zobrazí text článku. Pokud je článek hostován na jiných servrech než je Agris.cz, tak se místo textu článku zobrazí jen perex a tlačítko pro zobrazení článku pomocí webového prohlížeče zařízení.

4.3 Vytvoření uživatelského rozhraní

Uživatelské rozhraní a jeho tvorba je v obou vývojových prostředích stejná. Celá aplikace je tvořena pomocí fragmentů. Hlavním layoutem activity_main.xml je pouze prázdný <FrameLayout>, do kterého se načítají jednotlivé fragmenty. K vytvoření seznamu článků je použit ListFragment, který má svůj vlastní defaultní layout seznamu. K tomuto layoutu je připojena patička footer.xml, která obsahuje <LinearLayout> s horizontální orientací. Ten obsahuje tlačítka umístěná vedle sebe pro načítání dalších stránek seznamu.

Kód patičky footer.xml:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:id="@+id/footer">
    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:orientation="horizontal">
        <Button
            android:layout_width="0dp"
            android:layout_weight="2"
            android:layout_height="fill_parent"
            android:id="@+id/butPrev"
            android:text="@string/previous_button"/>
        <Button
            android:layout_width="0dp"
            android:layout_weight="2"
            android:layout_height="fill_parent"
            android:id="@+id/butNext"
            android:text="@string/next_button"/>
    </LinearLayout>
</LinearLayout>
```

K zobrazení článku slouží layout fragment_article.xml. V něm se nachází <ScrollView> umožňující uživateli vertikálně posouvat celý jeho obsah. Ten tvoří název článku, datum, <WebView> zobrazující text článku včetně tabulek a odkazů, tlačítko a upozornění pro otevření článku ve webovém prohlížeči zařízení. Část kódu fragment_article.xml :

```
<ScrollView
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:orientation="vertical">
        <TextView
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:id="@+id/aTitle"
            android:textSize="20sp"
            android:textStyle="bold"/>
        <TextView
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:id="@+id/aDate"
            android:textSize="16sp"/>
        <WebView
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:id="@+id/aText"/>
```

4.4 Třídy, objekty a metody

Aplikace využívá metody z knihoven Android. V obou vývojových prostředích Android Studio a Xamarin se tyto metody volají stejně, až na to, že názvy těchto metod v Xamarinu začínají velkým písmenem.

4.4.1 Třída MainActivity

Třída, která se stará o zobrazování a výměnu fragmentů za pomocí transakcí. Tato třída je aktivita. K transakcím fragmentů v Android Studiu využívá FragmentManager z knihovny android.support.v7.app.AppCompatActivity. V Xamarinu k transakcím fragmentů využívá FragmentTransaction z knihovny android.App. Tato třída obsahuje tři metody, první metoda onCreate se volá automaticky po zapnutí aplikace a slouží k prvnímu načtení seznamu článků. Tato metoda vytvoří nový fragment seznamu článků a připojího do fragment kontajneru hlavního layoutu.

Metoda onCreate v Android Studiu:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    if (savedInstanceState == null){
        ArtListFragment artListFragment =
          ArtListFragment.newInstance(1,true);
        getFragmentManager().beginTransaction()
          .add(R.id.fragment_container,artListFragment).commit();
        this.getFragmentManager().executePendingTransactions();
    }
}
```

Metoda onCreate v Xamarinu:

```
protected override void OnCreate(Bundle savedInstanceState){
    base.OnCreate(savedInstanceState);
    SetContentView(Resource.Layout.Main);

    if (savedInstanceState == null) {
        ArtListFragment artListFragment = ArtListFragment.newInstance(1, true);
        FragmentTransaction fragmentTx = this.FragmentManager.BeginTransaction();
        fragmentTx.Add(Resource.Id.fragment_container, artListFragment);
        fragmentTx.Commit();
    }
}
```

Zbylé dvě metody tato třída implementuje ze třídy seznamu článků, která je volá. Tyto metody slouží ke změně zobrazovaného fragmentu, za pomocí transakce fragmentů podobné té jako v první metodě. Metoda reloadArtListFragment vymění stávající fragment za fragment seznamu článků, je využívána ke změně stránek. Metoda loadArticleFragment slouží k načtení a fragmentu článku, který zobrazí uživateli jím vybraný článek.

Metody v Android Studiu:

```
@Override
public void loadArticleFragment(String id, ArticleList.ArticleItem
articleItem) {
    ArticleFragment articleFragment =
      ArticleFragment.newInstance(id, articleItem);
    getFragmentManager().beginTransaction()
    .replace(R.id.fragment_container,
    articleFragment).addToBackStack(null).commit();
    this.getFragmentManager().executePendingTransactions();
}

@Override
public void reloadArtListFragment(int page,boolean first) {
    ArtListFragment artListFragment =
      ArtListFragment.newInstance(page,first);
    getFragmentManager().beginTransaction()
    .replace(R.id.fragment_container, artListFragment).commit();
    this.getFragmentManager().executePendingTransactions();
}
```

Metody v Xamarinu:

```
public void loadArticleFragment(string id, ArticleList.ArticleItem article){
    ArticleFragment articleFragment = ArticleFragment.newInstance(id, article);
    FragmentTransaction fragmentTx = this.FragmentManager.BeginTransaction();
    fragmentTx.Replace(Resource.Id.fragment_container, articleFragment);
    fragmentTx.Commit();
}

public void reloadArtListFragment(int page, bool first) {
    ArtListFragment artListFragment = ArtListFragment.newInstance(page, first);
    FragmentTransaction fragmentTx = this.FragmentManager.BeginTransaction();
    fragmentTx.Replace(Resource.Id.fragment_container, artListFragment);
    fragmentTx.Commit();
}
```

4.4.2 Objekty

Aplikace pracuje se dvěma objekty. První je object Article představující celý jeden článek. Obsahuje jeho ID, název, datum vydání, perex, text článku a url odkaz pokud článek není hostován na Agris.cz. S tímto objektem pracuje fragment ArticleFragment. Druhým objektem je ArticleList který obsahuje seznam objektů ArticleItem, představující seznam článků k zobrazení na výběr uživateli. Tyto články mají atributy ID, headline obsahující titulek článku, perex, a url adresu článku.

4.4.3 Třída ArtListFragment

Tato třída je fragment zobrazující seznam článků. Dědí ze třídy ListFragment obsažené v knihovnách Androidu. To umožňuje využít ListView, které vytvoří seznam, a adaptérk jeho naplnění. Po vytvoření fragmentu se zavolá metoda triggerDownload,která stáhne seznam článků a uložího do objektu ArticleList. Metoda triggerDownload nejdříve zkontroluje stav připojení k internetu a poté spustí stahování dat.

V Android Studiu probíhá stahování dat pomocí asynchroní třídy DownloadList která dědí ze třídy AsyncTask. Tato třída pracuje ve třech krocích/metodách. První krok je metoda onPreExecute která proběhne před stahováním dat a vyvolá dialog který je zobrazen během stahování. Druhý krok je metoda doInBackground která běží napozadí a zařizuje stahování dat pomocí metody downloadUrl, které je předá třetímu kroku. Třetím krokem je metoda onPostExecute, která zpracovává stažená data. Stažená data jsou ve formátu Json, jsou parsována a uložena do objektu ArticleList.

Stažení dat v Android Studiu:

```
@Override
protected Void doInBackground(String... urls) {
    try{
        jsonString = downloadUrl(urls[0]);
    }catch (IOException e){
        Log.e("Service Handler", "problem");
        return null;
    }
    return null;
}
```

```
private String downloadUrl(String myurl)throws IOException{
    InputStream inputStream = null;
    URL url = new URL(myurl);
    HttpURLConnection connection =
      (HttpURLConnection) url.openConnection();
    connection.setReadTimeout(10000);
    connection.setConnectTimeout(15000);
    connection.setRequestMethod("GET");
    connection.setDoInput(true);
    try {connection.connect();
        inputStream = connection.getInputStream();
        java.util.Scanner scanner =
         new java.util.Scanner(inputStream).useDelimiter("\\A");
        return scanner.hasNext() ? scanner.next():"";
    }finally {
        if (inputStream!=null)
            inputStream.close();
        connection.disconnect();
    }
}
```

Parsování a uložení Jsonu do seznamu v Android Studiu:

```
if (jsonString!=null){
    try {
        ArticleList.clearArticlesList();
        JSONObject jsonObject = new JSONObject(jsonString);
        JSONArray articlesArr;
        articlesArr = jsonObject.getJSONArray(TAG_ARTICLES);
        for (int i=0;i<articlesArr.length();i++){
            JSONObject articlesObj = articlesArr.getJSONObject(i);
            String id = articlesObj.getString(TAG_ID);
            String headline = articlesObj.getString(TAG_HEADLINE);
            String perex = articlesObj.getString(TAG_PEREX);
            String url =articlesObj.getString(TAG_URL);
            ArticleList.ArticleItem articleItem=
            new ArticleList.ArticleItem(id, headline, perex, url);
            ArticleList.addArticleItem(articleItem);
        }
    }catch (JSONException e){
        e.printStackTrace();
    }
}else{
    Log.e("ServiceHandler", "Couldn't get any data from the url");
}
```

V Xamarinu probíhá stahování a parsování dat v úloze async Task. Nejdříve se stáhnou data ve formátu Json, který je uložen do Stringu a předán metodě ParseAndDisplay k parsování. Tato metoda využívá k parsování knihovnu json.NET. Po sparsování dat, jsou data uložena do objektu ArticleList.

Stažení dat v Xamarinu:

```
private async Task DownloadList(string url){
    HttpWebRequest request = (HttpWebRequest)HttpWebRequest
      .Create(new System.Uri(url));
    request.ContentType = "application/json";
    request.Method = "GET";
    using (WebResponse response = await request.GetResponseAsync()){
         using (StreamReader reader =
          new StreamReader(response.GetResponseStream())){
                    String content = reader.ReadToEnd();
                    ParseAndDisplay(content);
          }
    }
}
```

Parsování a uložení Jsonu do seznamu v Xamarinu:

```
private void ParseAndDisplay(String json) {
    String editedJson = json.Remove(0, 10);
    editedJson=editedJson.Remove(editedJson.Length - 1);
    ArticleList.clearArticlesList();
    dynamic data = Newtonsoft.Json.JsonConvert.DeserializeObject(editedJson);
    for (var i = 0; i < data.Count; i++){
        dynamic item = data[i];
        item = item.Substring(1, item.Length - 2);
        String id = (string)data.id_text;
        String headline = (string)data.nazev;
        String perex = (string)data.perex;
        String url = (string)data.url;
        ArticleList.ArticleItem articleItem =
          new ArticleList.ArticleItem(id,headline, perex, url);
        ArticleList.addArticleItem(articleItem);
    }
```

Porovnáním těchto kódů zjistíme, že stahování dat je v C# v Xamarinu mnohem jednodušší než za pomocí Javy v Android Studiu. Naopak si Java oproti C# poradila s parsováním Jsonu lépe.

Poté tato třída pomocí metody onViewCreated vykreslí ListView obsahující data z ArticleList a ještě k němu připojí patičku footer.xml se dvěma tlačítky. Obsluha těchto tlačítek se v Android Studiu a v Xamarinu velice liší. V Android Studiu je řešena pomocí rozšíření o View.OnClickListener, který se přiřadí tlačítkům. Po stisku tlačítka volá metodu onClick, která dává funkčnost těmto tlačítkům. V Xamarinu je toto řešeno elegantněji bez potřeby implementace View.OnClickListener. Funkčnost tlačítek je udělána pomocí události tlačítka Click

Tlačítka v Android Studiu:

```
        view.findViewById(R.id.butNext).setOnClickListener(this);
    }

    @Override
    public void onClick(View v) {
        switch (v.getId()) {
            case R.id.butNext:
                page++;
                page_changed=true;
                triggerDownload("http://www.agris.cz/dalsi-novinky
                      ?vratmi=json&page=" + page + "&pageSize=20");
                break;
        }
    }
```

Tlačítka v Xamarinu:

```
Button nextButton = view.FindViewById<Button>(Resource.Id.butNext);
nextButton.Click += delegate{
    page++;
    page_changed = true;
    triggerDownload("http://www.agris.cz/dalsi-novinky?vratmi=json&page="
                    + page + "&pageSize=20");
};
```

Třída ještě obsahuje metodu onListItemClick, která se volá, když uživatel klikne na položku v seznamu. Tato metoda zavolá metodu loadArticleFragment ze třídy Main. Toto je řešeno pomocí rozhraní OnFragmentInteractionListener této třídy, které třída Main implementuje. Tato metoda se v Android Studiu a v Xamarinu neliší.

Metoda onListItemClick a rozhraní OnFragmentInteractionListener v Android Studiu:

```
@Override
public void onListItemClick(ListView l, View v, int position, long id) {
    super.onListItemClick(l, v, position, id);
    if (mListener != null) {
        mListener.loadArticleFragment(ArticleList.ITEMS.get(position).id,
                                      ArticleList.ITEMS.get(position));
    }
}


public interface OnFragmentInteractionListener {
    void loadArticleFragment(String id,
                             ArticleList.ArticleItem articleItem);
    void reloadArtListFragment(int page,boolean first);
}
```

4.4.4 Třída ArticleFragment

Třída ArticleFragment zobrazuje uživateli jím zvolený článek. Tato třída dědí ze třídy Fragment zknihoven Androidu. Třída funguje dvěma různými způsoby, lišícími se tím, jestli se článek nachází na servrech Agris.cz, nebo je hostován jinde. Pokud je článek na servrech Agris.cz, tak ho fragment po vytvoření stáhne a sparsuje. Stahování probíhá totožně jako ve třídě ArtListFragment. Po stažení dat se data parsují do objektu Article a přiřadí jednotlivím prvkům uživatelského rozhraní. Zde se ukázalo parsování dat, být jednodušší v Xamarinu.

Parsování dat a naplnění prvků UI v Android Studiu:

```
JSONObject aJsonObj = new JSONObject(jsonString);
 title=aJsonObj.getString(TAG_HEADLINE);
 perex=aJsonObj.getString(TAG_PEREX);
 text=aJsonObj.getString(TAG_TEXT);
 date=aJsonObj.getString(TAG_DATE);
article= new Article(title, idArticle,text,perex,date,"");

tvTitle=(TextView)getActivity().findViewById(R.id.aTitle);
tvDate=(TextView)getActivity().findViewById(R.id.aDate);
webview=(WebView)getActivity().findViewById(R.id.aText);

tvTitle.setText(article.title);
tvDate.setText(article.date);
webview.loadData(article.text, "text/html; charset=UTF-8", "UTF-8");
```

Parsování dat a naplnění prvků UI v Xamarinu:

```
Article jArticle = JsonConvert.DeserializeObject<Article>(json);

tvTitle = Activity.FindViewById<TextView>(Resource.Id.aTitle);
tvDate = Activity.FindViewById<TextView>(Resource.Id.aDate);
webview = Activity.FindViewById<WebView>(Resource.Id.aText);

tvTitle.Text = article.title;
tvDate.Text = article.date;
webview.LoadData(article.text, "text/html; charset=UTF-8", "UTF-8");
```

Druhou možností je, že se článek nenachází na servru Agris.cz. V tom případě se data článku (titulek, datum, perex a odkaz na článek) převezmou z objektu ArticleList.ArticleItem a uloží do objektu Article. Dále pro oba způsoby následuje metoda onCreateView, která tomuto fragmentu přiřadí layout fragment_article, podle kterého má vykreslit uživatelské rozhraní. Poté proběhne metoda onViewCreated, která do layoutu přiřazuje jednotlivé prvky uživatelského rozhraní. A pokud se jedná o článek hostovaný na jiných servrech než na Agris.cz, tak těmto prvků přiřadí hodnoty objektu Article a vykreslí tlačítko pro otevření

článku ve webovém prohlížeči zažízení. Obsluha tohoto tlačítka probíhá stejně jako ve třídě ArtListFragment.

Otevření článku ve webovém prohlížeči zařízení v Android Studiu:

```
Intent browserIntent = new Intent(Intent.ACTION_VIEW,
                                  Uri.parse(artItem.url));
startActivity(browserIntent);
```

Otevření článku ve webovém prohlížeči zařízení v Xamarinu:

```
var uri = Android.Net.Uri.Parse(articleOriginal.url);
var intent = new Intent(Intent.ActionView, uri);
StartActivity(intent);
```

4.5 Android manifest

Android manifest je důležitým základním souborem, který má každá aplikace. Manifest obsahuje důležité informace o aplikaci pro systém Android, které jsou potřebné ještě před spuštěním aplikace. Mezi takovéto informace patří:

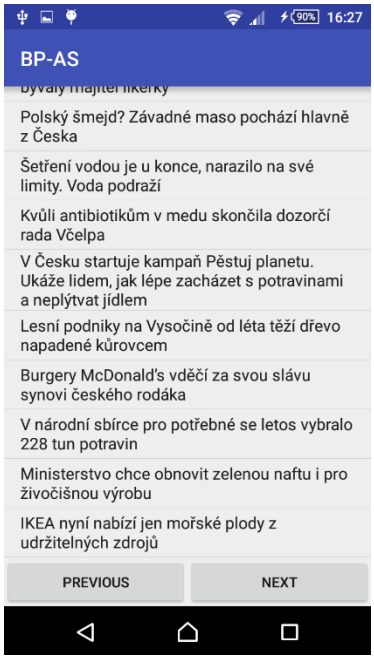
- Název balíčku aplikace sloužící jako unikátní identifikátor aplikace
- Popis komponentů aplikace (aktivity, služby, atd)
- Oprávnění aplikace
- Minimální úroveň Android API, které aplikace potřebuje
- Seznam přidaných knihoven

V manifestu této aplikace jsou oprávnění pro zjištění stavu stavu připojení k síti a oprávnění pro připojení k internetu.

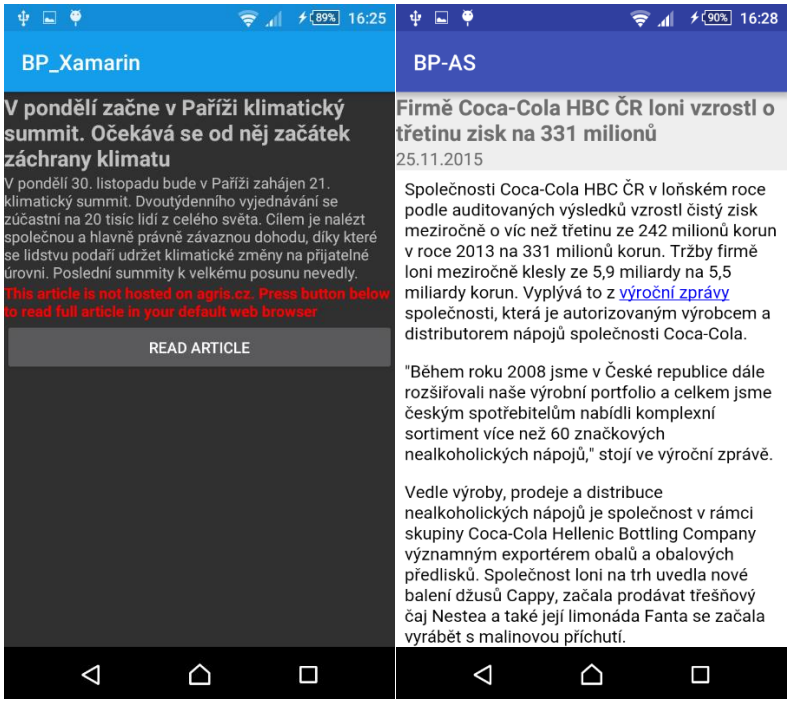
```
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permission android:name="android.permission.INTERNET" />
```

4.6 Vzhled aplikace

Aplikace se skládá ze dvou obrazovek. První je seznam článků a druhá je obsah článku. Vzhled se liší pouze použitým stylem, který je zvolen defaultní v každém vývojovém prostředí. Xamarin používá tmavě šedý styl, zatímco android používá světle šedý.



Obrázek 12 - Seznam článků v aplikaci vytvořené v Android Studiu



Obrázek 13 - Články v aplikacích vytvořených v Xamarinu a Android Studiu

5. Výsledky a diskuze

V teoretické části byla probrána historie operačního systému Android a fragmentace jeho verzí. Dále byla popsána jeho architektura, která se skládá z pěti vrstev. Bylo probráno několik vývojových prostředí. Pro vývoj úplně základních aplikací by měl postačit App Inventor, který může posloužit zájemcům o programování a začínajícím programátorům jako výuková pomůcka, ve které se mohou naučit používat základní příkazy, jako jsou podmínky a cykly. K vývoji pokročilejších aplikací poslouží Android Studio nebo plugin Xamarin ve Visual Studiu. Xamarin také nabízí možnost vytváření multiplatformních aplikací. V poslední řadě bylo představeno několik služeb k publikaci aplikací s tím, že byl popsán postup publikace v obchodu Google Play.

Praktická část byla zaměřena na vytvoření dvou totožných aplikací k zobrazování článků ze servru Agris.cz. Tyto aplikace byly vytvořeny ve vývojovém prostředí Android Studio a ve vývojovém prostředí Visual Studio Xamarin. Aplikace vytvořené v obou prostředích splnily svůj účel, ale nabízí se řada možností jak tyto aplikace vylepšit. Například možnost zobrazit více článků v seznamu, nebo udělat seznam nekonečně scrollovatelný. Další možností by mohlo být zobrazování náhledů obrázků k článkům v seznamu. Také by se mohla někomu hodit možnost si články uložit v telefonu, a přečíst si je v režimu offline.

Přestože se povedlo vytvořit dvě totožné aplikace v obou prostředích. Tak ze získaných zkušeností lze říci, že pro vytvoření aplikace pro Android, je lepší Android Studio než Xamarin plugin ve Visual Studiu. Android Studio nabízí více nápověd, lepší automatické dokončování textu a možnost rovnou aplikaci publikovat. Xamarin plugin oproti tomu trpí neúplnou integrací do Visual Studia. Chybělo například automatické dokončování textu při tvorbě XML souborů. Také se vyskytlo pár problémů jako je třeba špatné nebo žádné updatování souboru Resource.Designer.cs, který má nastarost využívání všech použitých zdrojů. Nelze tak toto vývojové prostředí doporučit, vývojáři by ho měli volit jen v případech, kdy chtějí programovat multiplatformní aplikaci, nebo chtějí využít knihoven C#. Android Studio se tedy stává lepší volbou pro vytváření aplikací.

6. Závěr

V první kapitole se tato práce zaměřuje na operační systém Android, jeho historii, aktuálním verzím na trhu a jeho architektuře. Kapitola druhá se zaobírá vývojem aplikací pro Android. Jsou v ní probírány vývojová prostředí Android Studio, Xamarin plugin pro Visual Studio a App Inventor. Dále se kapitola věnuje možnostem publikování a šíření hotové aplikace.

Získané poznatky při zpracování teoretické části byly spolu s osobními zkušenostmi využity při vytváření aplikací v praktické části práce. K vývoji aplikací byl použit počítač se systémem Windows 7 a telefon Sony. Aplikace byly vytvořeny pomocí jazyku Java v Android Studiu a pomocí jazyku C# ve Visual Studiu s pluginem Xamarin. Zdrojové kódy aplikací se nachází v příloze práce.

Cílem této práce bylo shrnutí všech potřebných znalostí pro započetí vývoje aplikace pro operační systém Android. A následné vytvoření dvou totožných aplikací v různých vývojových prostředích a jejich porovnání. Tento cíl byl splněn.

7. Seznam použitých zdrojů

1. Svět Androida. Krátké ohlédnutí za historií Androidu [online]. [cit. 3. 11. 2015]. http://www.svetandroida.cz/kratke-ohlednuti-za-historii-androidu-201305
2. Google Android developer. Dashboard [online] [cit. 20. 11. 2015]. http://developer.android.com/about/dashboards/index.html
3. Svět Androida. Historie Androidu v kostce aneb Od verze 1.0 až po Android M [online]. [cit. 3. 11. 2015]. http://www.svetandroida.cz/historie-androidu-201506
4. AndroidPIT. Android 6.0 Marshmallow: all the key features explained [online]. [cit. 3. 11. 2015]. https://www.androidpit.com/android-m-release-date-news-features-name
5. VÁVRŮ, Jiří a Miroslav UJBÁNYAI. *Programujeme pro Android*. 2. rozšířené vydání. Praha: Grada Publishing a.s., 2013. ISBN 978-80-247-4863-4.
6. Android. ART and Dalvik [online]. [cit. 10. 11. 2015]. https://source.android.com/devices/tech/dalvik/index.html
7. Statista. Number of available applications in the Google Play Store from December 2009 to July 2015 [online]. [cit. 11. 11. 2015]. http://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/
8. Google Android developer. Android Studio Overview [online]. [cit. 11. 11. 2015]. https://developer.android.com/tools/studio/index.html
9. Xamarin. Developer Center[online]. [cit. 11. 11. 2015]. http://developer.xamarin.com/
10. MIT App Inventor [online]. [cit. 11. 11. 2015]. http://appinventor.mit.edu/explore/
11. Oracle. Java Language and Virtual Machine Specifications[online]. [cit. 12. 11. 2015]. http://docs.oracle.com/javase/specs/
12. Microsoft MSDN. C# and Java: Comparing Programming Languages [online]. [cit. 12. 11. 2015]. https://msdn.microsoft.com/en-us/library/ms836794.aspx
13. Google Play. Nahrávání a distribuce aplikací [online]. [cit. 13. 11. 2015]. https://support.google.com/googleplay/android-developer/answer/113469?hl=cs
14. Svět Androida. Google po vzoru Applu schvaluje aplikace do Obchodu Play ručně [online]. [cit. 13. 11. 2015]. http://www.svetandroida.cz/google-play-schvalovani-201503

8. Přílohy

8.1 Seznam obrázků

8.2 Seznam tabulek

8.3 Seznam příloh

Příloha 1 - Zdrojový kód souboru aktivity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/fragment_container"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
```

Příloha 2 - Zdrojový kód souboru footer.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:id="@+id/footer">
    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:orientation="horizontal">
        <Button
            android:layout_width="0dp"
            android:layout_weight="2"
            android:layout_height="fill_parent"
            android:id="@+id/butPrev"
            android:text="@string/previous_button"/>
        <Button
            android:layout_width="0dp"
            android:layout_weight="2"
            android:layout_height="fill_parent"
            android:id="@+id/butNext"
            android:text="@string/next_button"/>
    </LinearLayout>
</LinearLayout>
```

Příloha 3 - Zdrojový kód souboru fragment_article.xml

```
<FrameLayout   xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.michael.BP_AS.ArticleFragment">
    <ScrollView
        android:layout_width="fill_parent"
        android:layout_height="fill_parent">
        <LinearLayout
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:orientation="vertical">
            <TextView
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:id="@+id/aTitle"
                android:textSize="20sp"
```

```
            android:textStyle="bold"/>
        <TextView
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:id="@+id/aDate"
            android:textSize="16sp"/>
        <WebView
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:id="@+id/aText"/>
        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="fill_parent"
            android:orientation="vertical"
            android:visibility="gone"
            android:id="@+id/urlLayout">
            <TextView
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:id="@+id/aPerex"/>
            <TextView
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:textStyle="bold"
                android:textColor="#ff0000"
                android:text="@string/url_article"/>
            <Button
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:text="@string/url_button"
                android:id="@+id/openUrl"/>
        </LinearLayout>
    </LinearLayout>
</ScrollView>
</FrameLayout>
```

Příloha 4 - Zdrojový kód souboru MainActivity.java

```
package com.example.michael.bp_as;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;


public class MainActivity extends AppCompatActivity implements
ArtListFragment.OnFragmentInteractionListener{


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        if (savedInstanceState == null){
            ArtListFragment artListFragment =
ArtListFragment.newInstance(1,true);
```

```
getFragmentManager().beginTransaction().add(R.id.fragment_container,artLi
stFragment).commit();
            this.getFragmentManager().executePendingTransactions();
        }
    }

    @Override
    public void loadArticleFragment(String id, ArticleList.ArticleItem
articleItem) {
        ArticleFragment articleFragment = ArticleFragment.newInstance(id,
articleItem);

getFragmentManager().beginTransaction().replace(R.id.fragment_container,
articleFragment).addToBackStack(null).commit();
        this.getFragmentManager().executePendingTransactions();
    }

    @Override
    public void reloadArtListFragment(int page,boolean first) {
        ArtListFragment artListFragment =
ArtListFragment.newInstance(page,first);

getFragmentManager().beginTransaction().replace(R.id.fragment_container,
artListFragment).commit();
        this.getFragmentManager().executePendingTransactions();
    }
}
```

Příloha 5 - Zdrojový kód souboru ArtListFragment.java

```
package com.example.michael.bp_as;

import android.app.Activity;
import android.app.AlertDialog;
import android.app.ProgressDialog;
import android.content.Context;
import android.content.SharedPreferences;
import android.net.ConnectivityManager;
import android.net.NetworkInfo;
import android.os.AsyncTask;
import android.os.Bundle;
import android.app.ListFragment;
import android.preference.PreferenceManager;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.LinearLayout;
import android.widget.ListAdapter;
import android.widget.ListView;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;
```

```
import java.io.IOException;
import java.io.InputStream;
import java.net.HttpURLConnection;
import java.net.URL;


public class ArtListFragment extends ListFragment implements
View.OnClickListener{

    private static final String TAG_ARTICLES = "clanky";
    private static final String TAG_ID = "id_text";
    private static final String TAG_HEADLINE = "nazev";
    private static final String TAG_PEREX = "perex";
    private static final String TAG_URL = "url";
    private static int page;
    private static boolean page_changed=false;
    private static boolean first;

    private OnFragmentInteractionListener mListener;

    public ArtListFragment() {
    }

    public static ArtListFragment newInstance(int p,boolean f) {
        ArtListFragment fragment=new ArtListFragment();
        Bundle args = new Bundle();
        fragment.setArguments(args);
        page=p;
        first=f;
        return fragment;
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        if (first)
            triggerDownload("http://www.agris.cz/dalsi-
novinky?vratmi=json&page=" + page + "&pageSize=20");
        else{
            ListAdapter mAdapter = new
ArrayAdapter<>(getActivity(),android.R.layout.simple_list_item_1,android.
R.id.text1,ArticleList.ITEMS);
            setListAdapter(mAdapter);
        }
    }


    @Override
    public void onViewCreated(View view, Bundle savedInstanceState) {
        super.onViewCreated(view, savedInstanceState);
        LayoutInflater layoutInflater =
getActivity().getLayoutInflater();
        LinearLayout listFooterView =
(LinearLayout)layoutInflater.inflate(R.layout.footer,null);
        getListView().addFooterView(listFooterView);
        view.findViewById(R.id.butNext).setOnClickListener(this);
        view.findViewById(R.id.butPrev).setOnClickListener(this);
        if (page==1)
            view.findViewById(R.id.butPrev).setEnabled(false);
```

```
    }

    @Override
    public void onClick(View v) {
        switch (v.getId()) {
            case R.id.butNext:
                page++;
                page_changed=true;
                triggerDownload("http://www.agris.cz/dalsi-
novinky?vratmi=json&page=" + page + "&pageSize=20");
                break;
            case R.id.butPrev:
                page--;
                page_changed=true;
                triggerDownload("http://www.agris.cz/dalsi-
novinky?vratmi=json&page=" + page + "&pageSize=20");
                break;
        }
    }

    @Override
    public void onListItemClick(ListView l, View v, int position, long
id) {
        super.onListItemClick(l, v, position, id);
        if (mListener != null) {

mListener.loadArticleFragment(ArticleList.ITEMS.get(position).id,
ArticleList.ITEMS.get(position));
        }
    }

    @Override
    public void onAttach(Activity activity) {
        super.onAttach(activity);
        try {
            mListener = (OnFragmentInteractionListener) getActivity();
        } catch (ClassCastException e) {
            throw new ClassCastException(activity.toString()
                    + " must implement OnFragmentInteractionListener");
        }
    }

    @Override
    public void onDetach() {
        super.onDetach();
        mListener = null;
    }

    public interface OnFragmentInteractionListener {
        void loadArticleFragment(String id, ArticleList.ArticleItem
articleItem);
        void reloadArtListFragment(int page,boolean first);
    }


    private void triggerDownload(String stringUrl) {
        ConnectivityManager connMan =
(ConnectivityManager)getActivity().getSystemService(Context.CONNECTIVITY_
SERVICE);
```

```
        NetworkInfo netInfo = connMan.getActiveNetworkInfo();
        if (netInfo!= null && netInfo.isConnected())
            new DownloadList().execute(stringUrl);
        else{
            AlertDialog.Builder alertBuilder= new
AlertDialog.Builder(getActivity());
            alertBuilder.setMessage(R.string.no_connection);
            AlertDialog alertDialog = alertBuilder.create();
            alertDialog.show();
        }
    }


    private class DownloadList extends AsyncTask<String, Void, Void> {
        String jsonString;
        private ProgressDialog pDialog;

        @Override
        protected void onPreExecute() {
            super.onPreExecute();
            pDialog = new ProgressDialog(getActivity());

pDialog.setMessage(getResources().getString(R.string.loading_data));
            pDialog.setCancelable(false);
            pDialog.show();
        }

        @Override
        protected Void doInBackground(String... urls) {
            try{
                jsonString = downloadUrl(urls[0]);
            }catch (IOException e){
                Log.e("Service Handler", "problem");
                return null;
            }
            return null;
        }

        private String downloadUrl(String myurl)throws IOException{
            InputStream inputStream = null;
            URL url = new URL(myurl);
            HttpURLConnection connection =  (HttpURLConnection)
url.openConnection();
            connection.setReadTimeout(10000);
            connection.setConnectTimeout(15000);
            connection.setRequestMethod("GET");
            connection.setDoInput(true);
            try {connection.connect();
                inputStream = connection.getInputStream();
                java.util.Scanner scanner = new
java.util.Scanner(inputStream).useDelimiter("\\A");
                return scanner.hasNext() ? scanner.next():"";
            }finally {
                if (inputStream!=null)
                    inputStream.close();
                connection.disconnect();
            }
        }
```

```
        @Override
        protected void onPostExecute(Void result) {
            super.onPostExecute(result);

            if (pDialog.isShowing())
                pDialog.dismiss();

            if (jsonString!=null){
                try {
                    ArticleList.clearArticlesList();
                    JSONObject jsonObject = new JSONObject(jsonString);
                    JSONArray articlesArr;
                    articlesArr = jsonObject.getJSONArray(TAG_ARTICLES);
                    for (int i=0;i<articlesArr.length();i++){
                        JSONObject articlesObj =
articlesArr.getJSONObject(i);
                        String id = articlesObj.getString(TAG_ID);
                        String headline =
articlesObj.getString(TAG_HEADLINE);
                        String perex = articlesObj.getString(TAG_PEREX);
                        String url =articlesObj.getString(TAG_URL);
                        ArticleList.ArticleItem articleItem=new
ArticleList.ArticleItem(id, headline, perex, url);
                        ArticleList.addArticleItem(articleItem);
                    }
                }catch (JSONException e){
                    e.printStackTrace();
                }
            }else{
                Log.e("ServiceHandler", "Couldn't get any data from the
url");
            }
            if (first) {
                ListAdapter mAdapter = new ArrayAdapter<>(getActivity(),
android.R.layout.simple_list_item_1, android.R.id.text1,
ArticleList.ITEMS);
                setListAdapter(mAdapter);
            }
            if (page_changed)
                mListener.reloadArtListFragment(page,false);
        }
    }
}
```

Příloha 6 - Zdrojový kód souboru ArticleFragment.java

```
package com.example.michael.bp_as;


import android.app.AlertDialog;
import android.app.ProgressDialog;
import android.content.Context;
import android.content.Intent;
import android.net.ConnectivityManager;
import android.net.NetworkInfo;
import android.net.Uri;
```

```
import android.os.AsyncTask;
import android.os.Bundle;
import android.app.Fragment;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.webkit.WebView;
import android.widget.LinearLayout;
import android.widget.TextView;

import org.json.JSONException;
import org.json.JSONObject;

import java.io.IOException;
import java.io.InputStream;
import java.net.HttpURLConnection;
import java.net.URL;


public class ArticleFragment extends Fragment implements
View.OnClickListener{

    private static final String TAG_HEADLINE = "title";
    private static final String TAG_PEREX = "perex";
    private static final String TAG_DATE = "date";
    private static final String TAG_TEXT = "text";

    public static String idArticle;
    public static ArticleList.ArticleItem artItem;
    static Article article;

    static String title;
    static String date;
    static String perex;
    static String text;

    TextView tvTitle;
    TextView tvDate;
    TextView tvPerex;
    WebView webview;
    LinearLayout urlLayout;

    public ArticleFragment() {
    }

    public static ArticleFragment newInstance(String id,
ArticleList.ArticleItem articleItem) {
        ArticleFragment fragment = new ArticleFragment();
        Bundle args = new Bundle();
        fragment.setArguments(args);
        idArticle=id;
        artItem=articleItem;
        return fragment;
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```

```
        if (getArguments()!=null) {
            if (artItem.url.equals(""))
                triggerDownload("http://www.agris.cz/clanek/" +
(idArticle) + "?vratmi=json");
            else
                article = new Article(artItem.headline, idArticle, null,
artItem.perex, null, artItem.url);
        }
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup
container,
                             Bundle savedInstanceState) {
        return inflater.inflate(R.layout.fragment_article, container,
false);
    }


    @Override
    public void onViewCreated(View view, Bundle savedInstanceState) {
        super.onViewCreated(view, savedInstanceState);
        view.findViewById(R.id.openUrl).setOnClickListener(this);

        tvTitle=(TextView)getActivity().findViewById(R.id.aTitle);
        tvDate=(TextView)getActivity().findViewById(R.id.aDate);
        tvPerex=(TextView)getActivity().findViewById(R.id.aPerex);
        webview=(WebView)getActivity().findViewById(R.id.aText);

urlLayout=(LinearLayout)getActivity().findViewById(R.id.urlLayout);

        if (getArguments()!=null) {
            if (artItem.url.equals("")) {
            } else {
                urlLayout.setVisibility(View.VISIBLE);
                tvDate.setVisibility(View.GONE);
                webview.setVisibility(View.GONE);
                tvTitle.setText(article.title);
                tvPerex.setText(article.perex);
            }
        }
    }

    @Override
    public void onClick(View v) {
        switch (v.getId()) {
            case R.id.openUrl:
                Intent browserIntent = new Intent(Intent.ACTION_VIEW,
Uri.parse(artItem.url));
                startActivity(browserIntent);
                break;
        }
    }

    private void triggerDownload(String stringUrl) {
        ConnectivityManager connMan =
(ConnectivityManager)getActivity().getSystemService(Context.CONNECTIVITY_
SERVICE);
        NetworkInfo netInfo = connMan.getActiveNetworkInfo();
```

```
        if (netInfo!= null && netInfo.isConnected())
            new DownloadArticle().execute(stringUrl);
        else{
            AlertDialog.Builder alertBuilder= new
AlertDialog.Builder(getActivity());
            alertBuilder.setMessage(R.string.no_connection);
            AlertDialog alertDialog = alertBuilder.create();
            alertDialog.show();
        }
    }

    private class DownloadArticle extends AsyncTask<String, Void, Void> {
        String jsonString;
        private ProgressDialog pDialog;

        @Override
        protected void onPreExecute() {
            super.onPreExecute();
            pDialog = new ProgressDialog(getActivity());

pDialog.setMessage(getResources().getString(R.string.loading_data));
            pDialog.setCancelable(false);
            pDialog.show();
        }

        @Override
        protected Void doInBackground(String... urls) {
            try{
                jsonString = downloadUrl(urls[0]);
            }catch (IOException e){
                Log.e("Service Handler", "problem");
                return null;
            }
            return null;
        }

        private String downloadUrl(String myurl)throws IOException{
            InputStream inputStream = null;
            URL url = new URL(myurl);
            HttpURLConnection connection =  (HttpURLConnection)
url.openConnection();
            connection.setReadTimeout(10000);
            connection.setConnectTimeout(15000);
            connection.setRequestMethod("GET");
            connection.setDoInput(true);
            try {
                connection.connect();
                inputStream = connection.getInputStream();

                java.util.Scanner scanner = new
java.util.Scanner(inputStream).useDelimiter("\\A");
                return scanner.hasNext() ? scanner.next():"";
            }finally {
                if (inputStream!=null)
                    inputStream.close();
                connection.disconnect();
            }
        }
```

```
        @Override
        protected void onPostExecute(Void aVoid) {
            super.onPostExecute(aVoid);

            if (pDialog.isShowing())
                pDialog.dismiss();

            if (jsonString != null) {
                try {
                    JSONObject aJsonObj = new JSONObject(jsonString);
                     title=aJsonObj.getString(TAG_HEADLINE);
                     perex=aJsonObj.getString(TAG_PEREX);
                     text=aJsonObj.getString(TAG_TEXT);
                     date=aJsonObj.getString(TAG_DATE);
                    article= new Article(title,
idArticle,text,perex,date,"");


tvTitle=(TextView)getActivity().findViewById(R.id.aTitle);

tvDate=(TextView)getActivity().findViewById(R.id.aDate);

webview=(WebView)getActivity().findViewById(R.id.aText);
                    tvTitle.setText(article.title);
                    tvDate.setText(article.date);
                    webview.loadData(article.text, "text/html;
charset=UTF-8", "UTF-8");

                } catch (JSONException e) {
                    e.printStackTrace();
                }
            } else {
                Log.e("ServiceHandler", "Couldn't get any data from the
url");
            }
        }
    }
}
```

Příloha 7 - Zdrojový kód souboru Article.java

```
package com.example.michael.bp_as;

public class Article {
    public String title;
    public String id;
    public String text;
    public String perex;
    public String date;
    public String url;

    public Article(String title, String id, String text, String perex,
String date, String url) {
        this.title = title;
        this.id = id;
        this.text = text;
```

```
        this.perex = perex;
        this.date = date;
        this.url = url;
    }
}
```

Příloha 8 - Zdrojový kód souboru ArticleList.java

```
package com.example.michael.bp_as;

import java.util.ArrayList;
import java.util.List;

public class ArticleList {
    public static List<ArticleItem> ITEMS = new ArrayList<>();

    public static void addArticleItem(ArticleItem item) {
        ITEMS.add(item);
    }
    public static void clearArticlesList(){
        ITEMS.clear();
    }
    public static class ArticleItem {
        public String id;
        public String headline;
        public String perex;
        public String url;

        public ArticleItem(String id, String headline, String perex,
String url) {
            this.id = id;
            this.headline = headline;
            this.perex = perex;
            this.url = url;
        }

        @Override
        public String toString() {
            return headline;
        }

        public String getId() {
            return id;
        }
    }
}
```

Příloha 9 - Zdrojový kód souboru MainActivity.cs

```
using Android.App;
using Android.OS;

namespace BP_Xamarin
{
```

```
    [Activity(Label = "BP_Xamarin", MainLauncher = true, Icon = "@drawable/icon")]
    public class MainActivity : Activity,
ArtListFragment.OnFragmentInteractionListener
    {
        protected override void OnCreate(Bundle savedInstanceState)
        {
            base.OnCreate(savedInstanceState);
            SetContentView(Resource.Layout.Main);

            if (savedInstanceState == null)
            {
                ArtListFragment artListFragment = ArtListFragment.newInstance(1,
true);
                FragmentTransaction fragmentTx =
this.FragmentManager.BeginTransaction();
                fragmentTx.Add(Resource.Id.fragment_container, artListFragment);
                fragmentTx.Commit();
            }
        }


        public void loadArticleFragment(string id, ArticleList.ArticleItem article)
        {
            ArticleFragment articleFragment = ArticleFragment.newInstance(id,
article);
            FragmentTransaction fragmentTx =
this.FragmentManager.BeginTransaction();
            fragmentTx.Replace(Resource.Id.fragment_container, articleFragment);
            fragmentTx.Commit();
        }

        public void reloadArtListFragment(int page, bool first)
        {
            ArtListFragment artListFragment = ArtListFragment.newInstance(page,
first);
            FragmentTransaction fragmentTx =
this.FragmentManager.BeginTransaction();
            fragmentTx.Replace(Resource.Id.fragment_container, artListFragment);
            fragmentTx.Commit();
        }
    }
}
```

Příloha 10 - Zdrojový kód souboru ArtListFragment.cs

```
using System;
using System.Linq;
using System.Threading.Tasks;
using System.Net;
using System.IO;

using Android.App;
using Android.Content;
using Android.OS;
using Android.Views;
using Android.Widget;
```

```
using Android.Net;


namespace BP_Xamarin
{
    public class ArtListFragment : ListFragment
    {
        private static int page;
        private static Boolean page_changed = false;
        private static Boolean first;
        private OnFragmentInteractionListener mListener;

        public ArtListFragment()
        {
        }

        public static ArtListFragment newInstance(int p, Boolean f)
        {
            ArtListFragment fragment = new ArtListFragment();
            Bundle args = new Bundle();
            fragment.Arguments = args;
            page = p;
            first = f;
            return fragment;
        }

        public override void OnCreate(Bundle savedInstanceState)
        {
            base.OnCreate(savedInstanceState);
            if (first)
                triggerDownload("http://www.agris.cz/dalsi-
novinky?vratmi=json&page=" + page + "&pageSize=20");
            else
            {
                String[] headlines = new String[ArticleList.ITEMS.Count];
                for(int i = 0; i < ArticleList.ITEMS.Count; i++)
                {
                    headlines[i] = ArticleList.ITEMS[i].ToString();
                }
                this.ListAdapter = new ArrayAdapter<string>(Activity,
Android.Resource.Layout.SimpleExpandableListItem1,
                    Android.Resource.Id.Text1, headlines);
            }
        }

        public override void OnViewCreated(View view, Bundle savedInstanceState)
        {
            base.OnViewCreated(view, savedInstanceState);
            LayoutInflater layoutInflater = Activity.LayoutInflater;
            LinearLayout listFooterView =
(LinearLayout)layoutInflater.Inflate(Resource.Layout.footer, null);
            ListView.AddFooterView(listFooterView);
            Button nextButton = view.FindViewById<Button>(Resource.Id.butNext);
            Button prevButton = view.FindViewById<Button>(Resource.Id.butPrev);
            nextButton.Click += delegate
             {
                page++;
                page_changed = true;
                triggerDownload("http://www.agris.cz/dalsi-
novinky?vratmi=json&page=" + page + "&pageSize=20");
```

```
             };
            prevButton.Click += delegate
             {
                 page--;
                 page_changed = true;
                 triggerDownload("http://www.agris.cz/dalsi-
novinky?vratmi=json&page=" + page + "&pageSize=20");
             };
            if (page == 1)
                view.FindViewById<Button>(Resource.Id.butPrev).Enabled = false;
        }


        public override void OnAttach(Activity activity)
        {
            base.OnAttach(activity);
            try
            {
                mListener = (OnFragmentInteractionListener)Activity;
            }
            catch (Exception e)
            {
                throw new Exception(activity.ToString()
                        + " must implement OnFragmentInteractionListener"+e);
            }
        }


        public override void OnListItemClick(ListView l, View v, int position, long
id)
        {
            base.OnListItemClick(l, v, position, id);
            if (mListener != null)

mListener.loadArticleFragment(ArticleList.ITEMS.ElementAt(position).id,ArticleList.I
TEMS.ElementAt(position));
        }

        public interface OnFragmentInteractionListener
        {
            void loadArticleFragment(String id, ArticleList.ArticleItem article);
            void reloadArtListFragment(int page, Boolean first);
        }

        private async void triggerDownload(String stringUrl)
        {
            ConnectivityManager connMan =
(ConnectivityManager)Activity.GetSystemService(Context.ConnectivityService);
            NetworkInfo netInfo = connMan.ActiveNetworkInfo;
            if (netInfo != null && netInfo.IsConnected)
            {
                await DownloadList(stringUrl);
            }
            else
            {
                AlertDialog.Builder alertBuilder = new
AlertDialog.Builder(Activity);
                alertBuilder.SetMessage(Resource.String.no_connection);
                AlertDialog alertDialog = alertBuilder.Create();
                alertDialog.Show();
```

```
            }
        }

        private async Task DownloadList(string url)
        {
            HttpWebRequest request = (HttpWebRequest)HttpWebRequest.Create(new
System.Uri(url));
            request.ContentType = "application/json";
            request.Method = "GET";
            using (WebResponse response = await request.GetResponseAsync())
            {
                using (StreamReader reader = new
StreamReader(response.GetResponseStream()))
                {
                    String content = reader.ReadToEnd();
                    ParseAndDisplay(content);
                }
            }
        }

        private void ParseAndDisplay(String json)
        {
            String editedJson = json.Remove(0, 10);
            editedJson=editedJson.Remove(editedJson.Length - 1);
            ArticleList.clearArticlesList();
            dynamic data =
Newtonsoft.Json.JsonConvert.DeserializeObject(editedJson);
            for (var i = 0; i < data.Count; i++)
            {
                dynamic item = data[i];
                item = item.Substring(1, item.Length - 2);
                String id = (string)data.id_text;
                String headline = (string)data.nazev;
                String perex = (string)data.perex;
                String url = (string)data.url;
                ArticleList.ArticleItem articleItem = new
ArticleList.ArticleItem(id, headline, perex, url);
                ArticleList.addArticleItem(articleItem);
            }

            if (first)
            {
                 String[] headlines = new String[ArticleList.ITEMS.Count];
                 for (int i = 0; i < ArticleList.ITEMS.Count; i++)
                 {
                     headlines[i] = ArticleList.ITEMS[i].ToString();
                 }
                 this.ListAdapter = new ArrayAdapter<string>(Activity,
Android.Resource.Layout.SimpleExpandableListItem1,
                     Android.Resource.Id.Text1, headlines);
            }
            if (page_changed)
               mListener.reloadArtListFragment(page, false);

        }
    }
}
```

Příloha 11 - Zdrojový kód souboru ArticleFragment.cs

```
using System;

using Android.App;
using Android.Content;
using Android.OS;
using Android.Views;
using Android.Widget;
using Android.Webkit;
using Android.Net;
using System.Threading.Tasks;
using System.IO;
using Newtonsoft.Json;
using System.Net;

namespace BP_Xamarin
{
    public class ArticleFragment : Fragment
    {
        public static String idArticle;
        public static ArticleList.ArticleItem articleOriginal;
        static Article article;

        TextView tvTitle;
        TextView tvDate;
        TextView tvPerex;
        WebView webview;
        LinearLayout urlLayout;

        public ArticleFragment()
        {
        }

        public static ArticleFragment newInstance(String id, ArticleList.ArticleItem
articleItem)
        {
            ArticleFragment fragment = new ArticleFragment();
            Bundle args = new Bundle();
            fragment.Arguments = args;
            idArticle = id;
            articleOriginal = articleItem;
            return fragment;
        }

        public override void OnCreate(Bundle savedInstanceState)
        {
            base.OnCreate(savedInstanceState);
            if (Arguments != null)
            {
                if (articleOriginal.url.Equals(""))
                    triggerDownload("http://www.agris.cz/clanek/" + (idArticle) +
"?vratmi=json");
                else
                    article = new Article(articleOriginal.headline, idArticle, null,
articleOriginal.perex, null, articleOriginal.url);
            }
        }
```

```
        public override View OnCreateView(LayoutInflater inflater, ViewGroup
container, Bundle savedInstanceState)
        {
            return inflater.Inflate(Resource.Layout.fragment_article, container,
false);
        }

        public override void OnViewCreated(View view, Bundle savedInstanceState)
        {
            base.OnViewCreated(view, savedInstanceState);
            Button openUrlButton = view.FindViewById<Button>(Resource.Id.openUrl);
            openUrlButton.Click += delegate
            {
                var uri = Android.Net.Uri.Parse(articleOriginal.url);
                var intent = new Intent(Intent.ActionView, uri);
                StartActivity(intent);
            };
            tvTitle = view.FindViewById<TextView>(Resource.Id.aTitle);
            tvDate = view.FindViewById<TextView>(Resource.Id.aDate);
            tvPerex = view.FindViewById<TextView>(Resource.Id.aPerex);
            webview = view.FindViewById<WebView>(Resource.Id.aText);
            urlLayout = view.FindViewById<LinearLayout>(Resource.Id.urlLayout);

            if (Arguments != null)
            {
                if (articleOriginal.url.Equals(""))
                {
                }
                else
                {
                    urlLayout.Visibility = ViewStates.Visible;
                    tvDate.Visibility = ViewStates.Gone;
                    webview.Visibility = ViewStates.Gone;
                    tvTitle.Text = article.title;
                    tvPerex.Text = article.perex;
                }
            }
        }


        private async void triggerDownload(String stringUrl)
        {
            ConnectivityManager connMan =
(ConnectivityManager)Activity.GetSystemService(Context.ConnectivityService);
            NetworkInfo netInfo = connMan.ActiveNetworkInfo;
            if (netInfo != null && netInfo.IsConnected)
            {
                await DownloadArticle(stringUrl);
            }
            else
            {
                AlertDialog.Builder alertBuilder = new
AlertDialog.Builder(Activity);
                alertBuilder.SetMessage(Resource.String.no_connection);
                AlertDialog alertDialog = alertBuilder.Create();
                alertDialog.Show();
            }
        }
```

```
        private async Task DownloadArticle(string url)
        {
            HttpWebRequest request = (HttpWebRequest)HttpWebRequest.Create(new
System.Uri(url));
            request.ContentType = "application/json";
            request.Method = "GET";
            using (HttpWebResponse response =await request.GetResponseAsync() as
HttpWebResponse)
            {
                if (response.StatusCode != HttpStatusCode.OK)
                    Console.WriteLine("Error fetching data. Server returned status
code: {0}", response.StatusCode);
                using (StreamReader reader = new
StreamReader(response.GetResponseStream()))
                {
                    String content = reader.ReadToEnd();
                    ParseAndDisplay(content);
                }
            }
        }

        private void ParseAndDisplay(String json)
        {
            tvTitle = Activity.FindViewById<TextView>(Resource.Id.aTitle);
            tvDate = Activity.FindViewById<TextView>(Resource.Id.aDate);
            webview = Activity.FindViewById<WebView>(Resource.Id.aText);

            Article jArticle = JsonConvert.DeserializeObject<Article>(json);
            tvTitle.Text = article.title;
            tvDate.Text = article.date;
            webview.LoadData(article.text, "text/html; charset=UTF-8", "UTF-8");
        }
    }
}
```

Příloha 12 - Zdrojový kód souboru Article.cs

```
using System;

namespace BP_Xamarin
{
    public class Article
    {
        public String title;
        public String id;
        public String text;
        public String perex;
        public String date;
        public String url;

        public Article(String title, String id, String text, String perex, String
date, String url)
        {
            this.title = title;
            this.id = id;
            this.text = text;
            this.perex = perex;
            this.date = date;
            this.url = url;
```

```
        }
    }
}
```

Příloha 13 - Zdrojový kód souboru ArticleList.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Newtonsoft.Json;

namespace BP_Xamarin
{
    public class ArticleList
    {
        public static List<ArticleItem> ITEMS = new List<ArticleItem>();

        public static void addArticleItem(ArticleItem item)
        {
            ITEMS.Add(item);
        }
        public static void clearArticlesList()
        {
            ITEMS.Clear();
        }


        public class ArticleItem
        {
            [JsonProperty("id_text")]
            public String id;
            [JsonProperty("nazev")]
            public String headline;
            public String perex;
            public String url;

            public ArticleItem(String id, String headline, String perex, String url)
            {
                this.id = id;
                this.headline = headline;
                this.perex = perex;
                this.url = url;
            }

            public String toString()
            {
                return headline;
            }

            public String getId()
            {
                return id;
            }
        }
    }
}
```