ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

PROVOZNĚ EKONOMICKÁ FAKULTA

KATEDRA INFORMAČNÍCH TECHNOLOGIÍ

Bakalářská práce

Vývoj mobilních aplikací pro os Android

Tomáš Jeřábek

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Katedra informačních technologií

Provozně ekonomická fakulta

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Tomáš Jeřábek

Informatika

Název práce

Vývoj mobilních aplikací pro OS Android

Název anglicky

Development of mobile applications for the Android OS

Cíle práce

Cílem práce je detailní představení vývoje aplikací na mobilní zařízení s operačním systémem Android. Dílčí cíle jsou:
- charakterizovat architekturu OS Android
- uvést a zhodnotit speciökace user interface pro OS Android a dostupných programovacích jazyků
- návrh a tvorba aplikace pro OS Android včetně uvedení možnosti jejího šíření
- hodnocení a závěry.

Metodika

Teoretická část práce představuje prostředí operačního systému Android a jednotlivé části vývoje mobilních aplikací. Zde používané informace jsou čerpány z internetových zdrojů a literatury uvedené ve zdrojích. Praktická část práce je zaměřena na demonstraci možností při vývoji mobilní aplikace. Na základě teoretických poznatků a výsledného řešení mobilní aplikace budou zformulovány závěry a doporučení bakalářské práce.

Doporučený rozsah práce

30 – 40 stran

Doporučené zdroje informací

ALLEN, Grant. Android 4: průvodce programováním mobilních aplikací. 1. vyd. Překlad Jakub Mužík.
Android Develpoers. Introduction to Android [online]. 2014. Dostupné z:
http://developer.android.com/guide/index.html
Brno: Computer Press, 2013, 656 s. ISBN 978-80-251-3782-6.
(Grada). ISBN 978-80-247-3995-3.
UJBÁNYAI, Miroslav. Programujeme pro Android. Vyd. 1. Praha: Grada, 2012, 187 s. Průvodce

Předběžný termín obhajoby

2015/06 (červen)

Vedoucí práce

Ing. Jiří Vaněk, Ph.D.

Elektronicky schváleno dne 31. 10. 2014

Ing. Jiří Vaněk, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 11. 11. 2014

Ing. Martin Pelikán, Ph.D.

Děkan

V Praze dne 10. 03. 2015

Čestné prohlášení

Prohlašuji, že svou bakalářskou práci "Vývoj mobilních aplikací pro os 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 10. 3. 2015 _____

Poděkování

Děkuji panu Ing. Jiřímu Vaňkovi, 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 mobilních aplikací pro OS Android

Development of mobile applications for the Android OS

Shrnutí

Tato bakalářská práce představuje vývoj aplikace na mobilní zařízení s operačním systémem Android včetně následného vytvoření aplikace pro demonstrování zde uvedených informací. Teoretická část práce se zabývá vlastnostmi operačního systému Android, možnostmi vývojových prostředí a dostupnými programovacími jazyky. Dále je zde probrána možnost šíření aplikace jak oficiální, tak neoficiální cestou. Praktická část demonstruje využití těchto znalostí při tvorbě aplikace. Je zde uveden postup vývoje aplikace včetně návrhu UI i implementace kódu. Výsledkem je funkční aplikace pro vyhledávání polohy a její zobrazení na mapě.

Klíčová slova

Android, Google, Smartphone, Java, Android studio, App Inventor, Eclipse.

Summary

This bachelor thesis presents the development of applications for mobile devices running Android OS, including creation of applications for demonstrating the information contained herein. The theoretical part deals with the properties of the operating system Android, the possibilities of development environments and programming languages available. Further, it discusses the possibility of the spread of applications both official and unofficial way. The practical part demonstrates the use of this knowledge to create an application. There is described the process of application development, including UI design and code implementation. The result is working application to search and show position on the map.

Key Words

Android, Google, Smartphone, Java, Android studio, App Inventor. Eclipse.

Obsah

1. Úvod

Mobilní telefony prošly za poslední desetiletí velkým vývojem. Na počátku tisíciletí uměly telefony přehrávat polyfonní melodie popřípadě tvořit vlastní vyzvánění, přistupovat na internet přes WAP a odesílat SMS. Dnes je však vše jiné, telefony často nahrazují osobní počítače a jsou vybaveny nejrůznějšími aplikacemi, které nám to umožňují. Oblast mobilních aplikací má jistě budoucnost a tak se tato práce bude věnovat možnostem vytvoření právě aplikace na operační systém Android, který je aktuálně nejpoužívanějším systémem na telefonech.

V předpokládané práci bude probrána problematika vývoje operačního systému Android, možnosti vývojových prostředí pro vytvoření aplikace i její šíření. Na závěr bude vše demonstrováno na aplikaci pro určení polohy.

2. Cíl a metodika

Cílem práce je představené vývoje aplikací na mobilní zařízení s operačním systémem Android. Dále probrat historii, vlastnosti a možnosti při programování pro tento operační systém. Tyto znalosti budou využity v praktické části a s jejich pomocí bude vytvořena mobilní aplikace.

V teoretické části je představen OS Android. Zde používané informace jsou čerpány z internetových zdrojů a literatury uvedené ve zdrojích. Vzhledem k rychlému zastarávání tištěných zdrojů je zvolen hlavní zdroj informací internet. Praktická část práce je zaměřena na demonstraci možností při vývoji mobilní aplikace. Pro programování je využito především dokumentace na oficiálních stránkách ale i programátorská fóra.

3. Teoretická východiska

3.1 OS Android

Android je open source systém pro mobilní telefony a další zařízení, jako tablety nebo GPS navigace. Protože je Android šířen jako open source, má každý uživatel nebo výrobce přístup ke zdrojovým kódům, jako u systémů založených na Linuxu.

3.1.1 Historie

S vývojem systému Android začala firma Android Inc. založená v roce 2003 v Palo Alto v Californii. V roce 2005 tuto společnost koupila firma Google. Google rovněž založil skupinu tzv. OHA[1]. Do tohoto uskupení Google pozval 84 firem, jako HTC nebo Samsung, které vyrábějí zařízení a telefony, a výrobce čipů, jako Qualcomm nebo Nvidia. Z operátorů, kteří nabízejí telekomunikační služby a distribuci telefonů, je zde T-Mobile. V roce 2005 Google představil svůj systém Android, čímž potvrdil spekulace, že chce vstoupit na trh s mobilními telefony. V roce 2008 ještě rozšířil OHA o dalších 14 členů, mezi nimi například Asus, Sony (Ericsson), Vodafone [1].

Ačkoliv byl první Apoužit na mobilním telefonu s 3,2“ velkým displejem a rozlišením 480 x 320 pixelu, je vyroben jako univerzální systém, který se přizpůsobí velikosti a rozlišení displeje. Dále jde použít na různé procesory, může pracovat se sim kartou nebo bez, paměťovou kartou, fotoaparátem a dalšími zařízeními. Díky tomu se tento systém dokázal rozšířit na většinu smartphone[2] a tabletů, ale také na multimediální centra, DVD přehrávače, do televizí, mikropočítačů velkých jako flash disk, netbook, a dokonce i do fotoaparátů.

Obrázek 1 - Logo Androidu

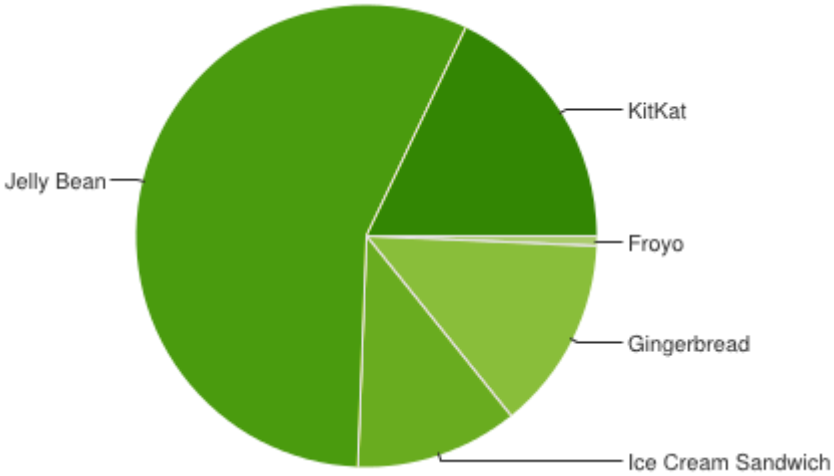
[1] OHA – Open Handset Alliance – uskupení firem pro vývoj Androidu a zařízení pro něj
[2] Telefon s operačním systémem

3.1.2 Fragmentace

Přes veškeré pozitivní vlastnosti to s kompatibilitou není až tak dokonalé, jak by se mohlo zdát. I když v současnosti existují 3 hlavní výrobci procesorů a jednou tolik výrobců telefonů, neznamená to, že by byl vývoj aktualizací nikterak rychlý. Poté co Google oznámí novou verzi Androidu, trvá cca půl roku, než ostatní výrobci tento systém použijí pro nové zařízení. Loňské modely se dočkají většinou pouze jedné aktualizace. Pokud by byl Android tak kompatibilní, nemusel by výrobce upravovat originální verzi od Google pro každý model zvlášť. Celý tento proces tedy zpomaluje vývoj systému, kdy se vývojáři musí rozhodnout, zda budou využívat nejnovější možnosti z poslední verze, nebo udělají aplikaci kompatibilní s předchozí verzí. Vývoj rovněž komplikuje velké množství různých zařízení. Na rozdíl od firmy Apple, kde ročně vydají cca 5-7 nových zařízení, Google nemůže ovlivnit, jaká zařízení budou na trhu, a proto každý výrobce uvádí i 2x za rok novou řadu všech telefonů.[2]

Verze	Jméno verze	API	Rozšíření
2.2	Froyo	8	0.7%
2.3	Gingerbread	10	13.5%
4.0	Ice Cream Sandwich	15	11.4%
4.1	Jelly Bean	16	27.8%
4.2		17	19.7%
4.3		18	9.0%
4.4	KitKat	19	17.9%

Tabulka 1 - Rozdělení verzí Androidu [2]



Obrázek 2 - graf rozdělení verzí Androidu [2]

3.1.3 Historie verzí

První verze Androidu 1.0 byla poprvé použita s telefonem HTC Dream. Ten byl vydán v září 2008 a měl pouze základní výbavu, jako prohlížeč webových stránek, fotoaparát bez natáčení nebo nastavení rozlišení atd., kalendář, kontakty a email, který se synchronizoval s Google účtem nebo aplikaci pro Youtube. Tento systém se dočkal aktualizace v únoru 2009, která převážně opravovala chyby.

V květnu 2009 následovala verze 1.5. Ta přidala tzv. widgety[3] (Obrázek 3), tedy miniaplikace, které jsou na ploše a automaticky se aktualizují. Mezi oblíbené patří například widget počasí, emailu nebo třeba televizního programu. Rovněž přináší podporu videí ve formátu MPEG a 3GP. Také přibyla možnost přidání vlastních knihoven, to umožní rychlejší aplikace.

Obrázek 3 – Příklad widgetu v Android 1.5

Zdroj: http://androidat.weebly.com/15-cupcake.html

[3] Miniaplikace na domovské obrazovce.

Následovala verze 1.6 uveřejněná v září 2009. Přidala podporu hlasového vyhledávání a vylepšenou aplikaci fotoaparátu s možností natáčet videa a prohlížet je v galerii. Také přibyla podpora gest – dotyků a pohybů prstů po displeji. V neposlední řadě také podpora dalších bezdrátových technologií jako CDMA nebo převod textu na zvuk.

V říjnu 2009 vyšla nová verze 2.0, která přinesla další možnosti pro fotoaparát, jako podpora blesku, vylepšený prohlížeč s podporou HTML5, dále využívá možností multitouch[4] a dostal přepracované UI[5].

Velké změny přišly ve verzi 2.2, která byla vydána v květnu 2010. Ta se stala velmi oblíbenou a na dlouhou dobu (spolu s 2.3) ovládla smartphone s Androidem. To především díky zvýšení výkonu, možnosti využít telefon jako wifi hotspot nebo podpoře Adobe flash. Nově je zde i podpora HD displejů a využití JIT kompilátoru. JIT kompilátor zrychlil spouštění aplikací 2x - 5x

Verze 2.3 přináší spíše kosmetické změny, větší rozlišení displejů, podporu NFC a podporu pro další senzory. Také byl vylepšen Garbage colector, který ukončuje nepoužívané části programu, a tak uvolňuje prostředky systému.

V této době nastal boom tabletů, ale nebyl pro ně vhodný systém. Android 2.3 počítal s velkým rozlišením, ale s malým displejem, a tak nebyl příliš přívětivý. Proto byl uveden Android verze 3.0 (), který byl především pro tablety, tedy zařízení s obrazovkou od 7“. Nicméně kromě upraveného UI pro velké displeje nebyl příliš odlišný a ani úspěšný, na rozdíl od iPadu, který ovládl trh tabletů.

[4] Multitouch – využití více dotyků současně, vyžaduje kapacitní displej
[5] UI – uživatelské rozhraní

Obrázek 4 - Android 3.0 úvodní obrazovka

Zdroj: http://androidat.weebly.com/30-honeycomb.html

Od verze 4.0 se toho příliš nemění. Jde spíše o kosmetické úpravy. Rovněž skončila éra Adobe Flash a Google proti němu začal bojovat, vypnul ho ve výchozím prohlížeči atd.

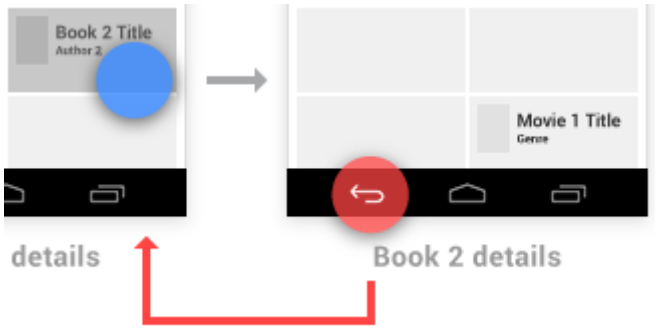
Verze 4.1 přináší 60fps animace s vsync[6], díky čemu je obraz plynulý a bez záseků. 4.2 přináší možnost připojit přes Bluetooth nebo USB od počítače k telefonu a využít ho pro hraní. 4.3 přináší podporu 4K rozlišení. Dála tato edice Jelly Bean využívá již linuxové jádro 3.4.0.

Zřejmě největší změna, které je ve verzi 4.4 z října 2013, je využití ART místo JIT. Bez aktivování je stále využívána JIT kompilace, ale testování ART a nového virtuálního stroje dává naději na zrychlení aplikací až o 50% a snížení spotřeby energie díky lepšímu překladu programu pro daný procesor. Také vyžaduje minimálně 340MB RAM, ale doporučeno je alespoň 512MB [3].

[6] Vertikální synchronizace obrazu

3.1.4 UI Androidu

UI neboli uživatelské rozhraní je zřejmě nejdůležitější prvek operačního systému pro telefon. Vzhledem k relativně malému displeji a omezenému prostoru pro tlačítka je potřeba mít dobře promyšlené ovládání. Jde ale také o prvky, jako virtuální tlačítka, vysouvací menu nebo různá zaškrtávací políčka, která musí mít dostatečnou velikost pro zmáčknutí prstem a to na různých velikostech obrazovek [4].



Obrázek 5 - Tlačítko zpět

Zdroj: https://developer.android.com/design/get-started/principles.html

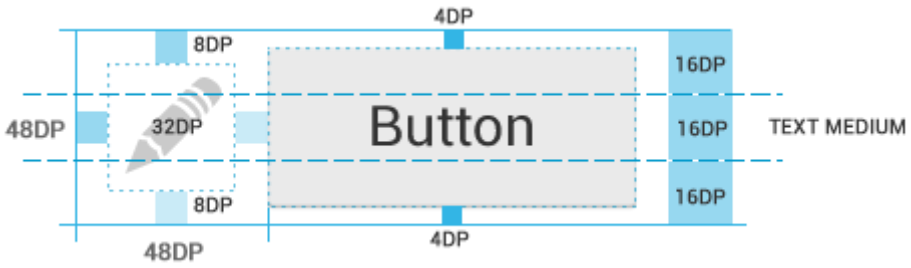
Na obrázku č. 5 je názorně předvedeno, jak je provázáno prohlížení aplikací. Díky tomu, že je vždy tato lišta s třemi tlačítky vidět, není problém se kdykoliv vrátit zpět nebo aplikaci minimalizovat a dostat se na výchozí plochu. Přestože se může zdát tato funkce nebo vlastnost jako samozřejmá, dříve byla řešena pomocí HW tlačítek na těle přístroje. Zde byl problém s tím, že každý výrobce mohl nějaké tlačítko přidat nebo ubrat podle své nástavby na Android, a tak se zde objevovalo tlačítko pro menu atd.

Obrázek 6 - Velikost prvků

Zdroj: https://developer.android.com/design/style/devices-displays.html

Vzhledem k tomu, že se Android používá na rozlišeních od 480*320 pixelů po 1920*1080 a je jisté, že se bude dále zvětšovat, musíme k tomu aplikaci a její prvky přizpůsobit. Také je rozdíl, pokud dané rozlišení je na 3“ obrazovce nebo 7“ tabletu. Z toho vyplývá, že je nutné znát i velikost úhlopříčky pro kvalitní zobrazení. Z toho důvodu je doporučeno začít vytvářet prvky, například od největšího (podle největšího plánovaného DPI), a poté zmenšovat a vytvářet menší prvky, které načteme při malém rozlišení.

Pro zajištění dostatečné velikosti prvků na obrazovce použijeme tzv. „48dp“. V podstatě má každý prvek alespoň 48 pixelů, a tak i na nejmenších obrazovkách nebo vysokých rozlišení má mezi doporučenými 7-10 mm. Tak lze docílit toho, že půjde zmáčknout prstem. Na obrázku č. 7 máme ikonu pera pouze 32 pixelu velkou, ale má 16 pixelů okraj, který pomůže zabránit přehmatům.



Obrázek 7 - Velikost prvků UI

Zdroj: http://developer.android.com/design/style/index.html

3.2. Vývojové prostředí

Pro vytvoření aplikace na OS Android je dostatek aplikací od jednodušších po složité a komplexní prostředí. Aplikaci může vytvořit každý (od začátečníka po zkušeného programátora) ovládající různé programovací jazyky.

Mezi pokročilé vývojové prostředí patří bezesporu Eclipse a nový program od Google Android studio. Podobné je i prostředí Xamarin, které spolu s Eclipse nabízí multiplatformní vývoj, můžeme tedy aplikaci uložit pro různé platformy mobilních telefonů bez dodatečných úprav.

Významný program, z grafického pohledu a vývoje her, je bezesporu Unity. Jeho výhoda je vedle podpory práce s grafikou také možnost vytvoření aplikace i pro PC nebo konzole. Na závěr je zde uveden menší program pro začínající programátory nebo ty, co si chtějí vyzkoušet vlastní aplikaci a to App Inventor. Pro začátečníky se hodí mimo jiné i proto, že běží ve webovém prohlížeči.

```
public class Main {
    public static void main(String[] args) {
        System.out.println("Goodbye, World!");
    }
}
```

Obrázek 8 - Příklad java kódu "Hello world"

3.2.1 Eclipse

Vývoj Eclipse byl započat na přelomu tisíciletí v IBM. Je to multiplatformní prostředí pro programování a je univerzální, dá se tak využít pro nejrůznější jazyky, jako C, C++, PHP, Phyton, Javascript, Java a mnoho dalších. To vše dokáže díky možnosti instalace pluginů[7].

Eclipse však není navržen pro tvorbu Android programů, a tak je potřeba stáhnout plugin Android Development Tools. ADT umožňuje rychle vytvořit nový Android program, vytvořit UI nebo vybrat správné API. ADT už není ve vývoji a nebude aktualizováno [8].

Vzhledem k ukončenému vývoji ADT stojí za zvážení začít s Android studiem, které v prosinci 2014 přešlo do stabilní verze, a tak je perspektivnější pro programovaní.

Další součást funkčního prostředí je JDK (Java Development Kit). Obsahuje JRE, tedy část pro spouštění Java programů, ale především kompilátor javac pro vytváření programů využívajících jazyk a knihovny Java. V případě, že bude vývojové prostředí oznamovat, že není JDK nainstalováno, je třeba nastavit proměnné prostředí JAVA_HOME na adresu, kde je JDK nainstalováno.

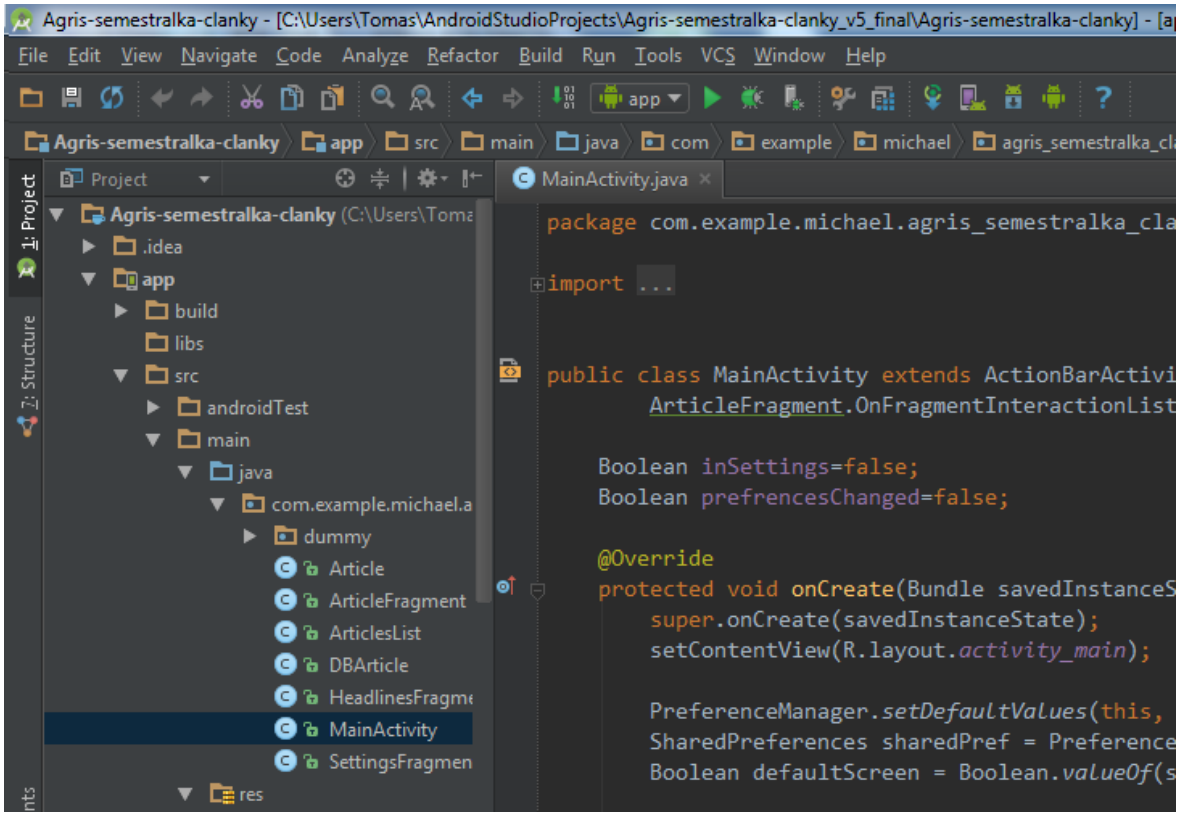
Poslední část je Android SDK. SDK funguje na bázi balíčků, a tak je potřeba stáhnout další části. To, co je potřeba, se liší od cílové verze OS, ale obecně je dobré stáhnout celý balík pro jednu verzi Androidu. Ten obsahuje například grafická téma pro náhled, SDK platform nebo soubor pro rozběhnutí virtuálního Androidu pro spouštění aplikací, pokud není k dispozici reálné zařízení. Navíc je potřeba stáhnout balíček Build-tools, Platform-tools a Tools. V případě využívání dalších funkcí při programování je třeba stáhnout také Android Support Repository a Library, které se využijí například při práci s fragmenty.

[7] Software, který rozšiřuje funkčnost původní aplikace

3.2.2 Android studio

Je vývojové prostředí pro platformu Android založené na IntelliJ IDEA. Vyvíjí ho firma Google a první verze byla představena 16. května 2013. Od té doby prošel beta verzí a 8. prosince 2014 byla vydána stabilní verze 1.0. Součástí instalace je Android Studio IDE, SDK Tools a kompilátor.

Oproti instalaci Eclipse je instalace Android studia snadnější. Není třeba ručně stahovat SDK ani ADT. Oboje je již zabudováno do studia, a tak zbývá jen JDK. Po spuštění studia a spuštění SDK Manager stáhneme stejně jako u Eclipse balíčky pro cílovou verzi Androidu a můžeme programovat.



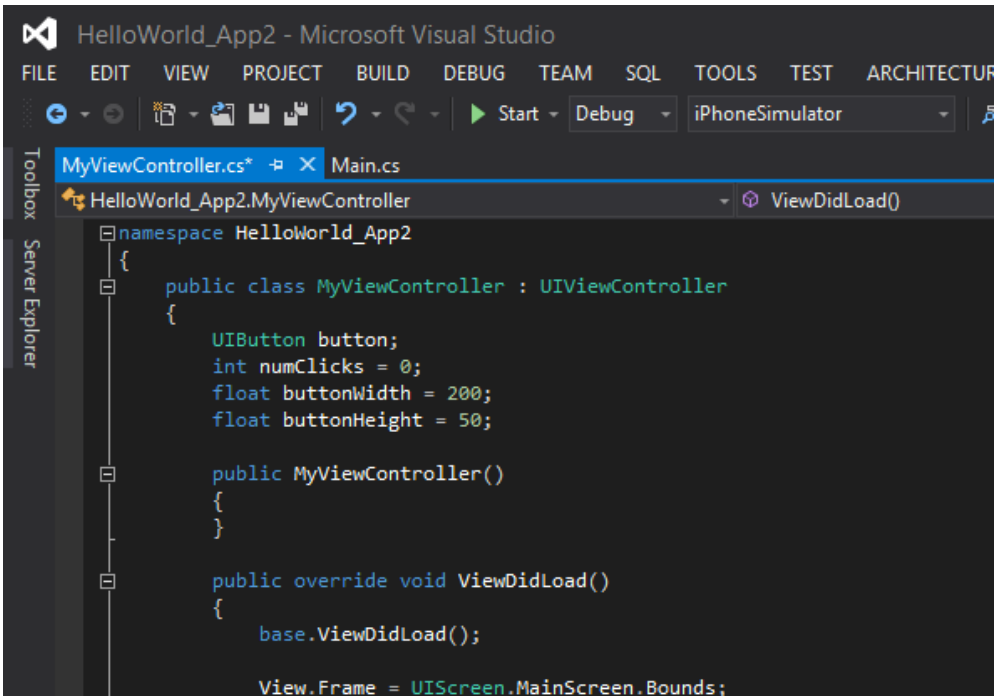
Obrázek 9 - Snímek z prostředí Android Studio

3.2.3 Xamarin

Xamarin studio je vývojové prostředí pro vytváření aplikací pro telefony s iOS, Android a také Windows. Aktuální verze 2.0 spojila iOS a Android vývojářské nástroje do jednoho balíku. Xamarin dovoluje vytvořit nativní aplikace pomocí jazyku C# a to buď právě v Xamarin studiu anebo ve Visual studiu, s pluginem Xamarin.

Xamarin pro Visual Studio podporuje funkci IntelliSense, automatické dokončování kódu a nápovědu při jeho psaní. Součástí je rozšíření pro debuggování a buildování aplikace a její simulace v PC nebo spouštění na zařízení.

Tvorba aplikace probíhá podobně jako u Android studia, díky grafickému návrháři není třeba vypisovat xml kód tlačítek a dalších prvků, ale je možné je přetáhnout a umístit na virtuální obrazovku zařízení.



Obrázek 10 - Xamarin pro Visual Studio

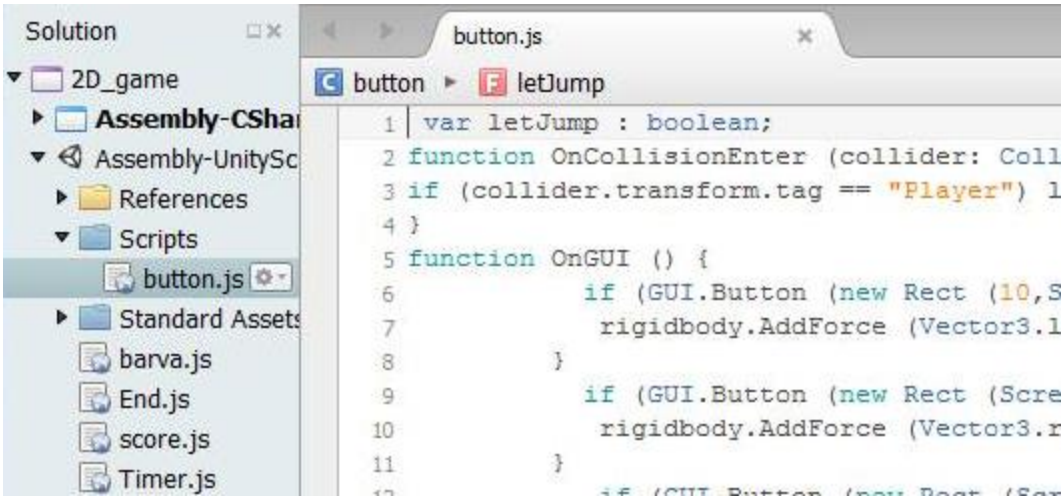
Zdroj: http://blog.xamarin.com/introduction-to-building-xamarin.ios-

3.2.4 Unity

Tento multiplatformní herní engine vyvinutý firmou Unity Technologies v roce 2005 byl původně pouze pro Mac OS. Aktuálně podporuje 15 platforem. Mezi ně patří mobilní OS pro smartphone, jako Android, iOS, Blackberry nebo Windows Phone. Kromě mobilních OS lze aplikaci vytvořenou v Unity spustit také na PC s operačním systémem Windows, Linux nebo Mac OS X. Další možnost rozšíření aplikace je možná s herními konzolemi od Playstation 3 po Xbox One. Na závěr stojí za zmínku možnost spustit aplikace také ve webovém prohlížeči [6].

Unity je profesionální nástroj a pro firmy, které vydělají více než 100 000 $, je zpoplatněn (verze Unity pro). Pro ostatní firmy, programátory studenty a všechny ostatní je zdarma a to ve verzi free, která je o některé funkce plné verze ochuzena.

Pro programování je použit jazyk C# a JavaScript[7]. Engine podporuje mimo jiné také technologii PhysX, která slouží k simulování chování materiálů, jako voda nebo zdí při pohybu, výbuchu atd., a to vše v reálném čase. Hlavní funkcí tohoto frameworku je vytváření her a tím pádem práce s grafikou. Je zde možnost využívat vestavěné nástroje na vytváření animací a jejich úpravu nebo částicových efektů. Také podporuje umělou inteligenci, grafické efekty, jako stíny, odlesky nebo level of detail, což je změna náročnosti vykreslování objektů a také jejich kvality [7].



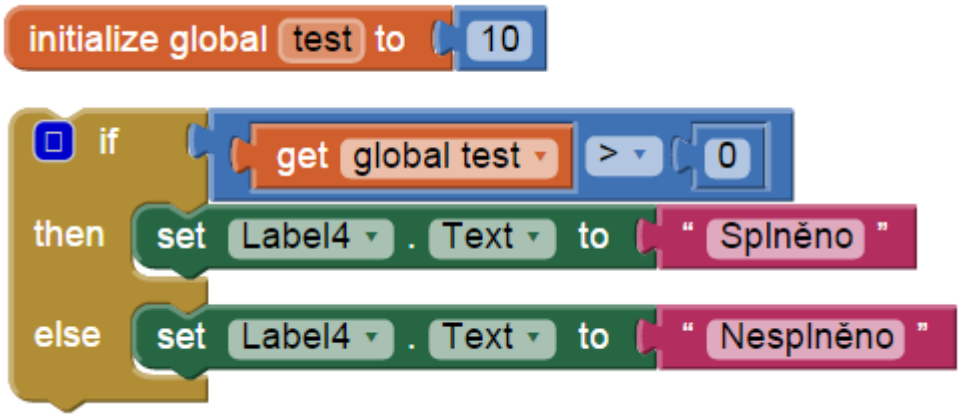
Obrázek 11 - Zdrojový kód z unity

3.2.5 MIT App Inventor

App Inventor je ideální pro vytvoření malé nepříliš náročné aplikace. Tento opensource[8] program, původně vyvinutý firmou Google, aktuálně vyvíjejí na Massachusetts Institute of Technology. První veřejná verze byla vydána 15. prosince 2010. V této verzi bylo pro programování – sestavování bloků kódu využita Open Block javová knihovna [5].

V roce 2011 Google ukončil vývoj této aplikace a předal ji MIT, kde ji dále vylepšují. Výsledkem jejich práce je App Inventor 2, uveřejněný v roce 2012. Hlavní změna je v nahrazení Open Block javascriptovou knihovnou Blockly. Díky tomuto kroku běží celá aplikace ve webovém prohlížeči a není potřeba nic instalovat.

App Inventor využívá stejně jako Android studio a ostatní programy návrhář, ve kterém se jednotlivé prvky, jako tlačítka, obrázky nebo textové pole, umístí na plochu znázorňující displej telefonu. Díky tomu je vidět, jak bude aplikace vypadat, a dál je možné prvky zarovnat nebo roztáhnout přes šířku displeje. Dále je zde v podobném stylu laděný blokový editor (Blockly), ve kterém se funkce, podmínky, proměnné nebo čtení dat ze senzoru přetahují ze seznamu těchto prvků a umísťují na „plátno“. Pro kompilaci programu využívá Framework Kawa napsaný v jazyku Java.



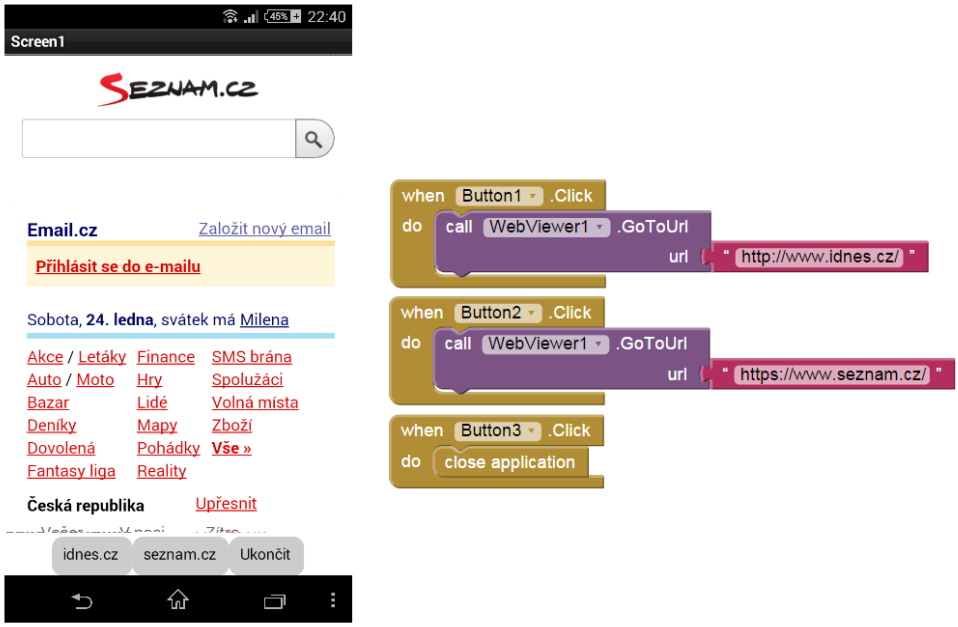
Obrázek 12 - Ukázka definice proměnné a bloku s funkcí If a nastavení textu do výstupu

[8] S volně šiřitelným zdrojovým kódem

Asi nejzajímavější věc vedle přehlednosti práce v tomto programu je jeho debuggování[9] a spouštění. Pokud využijeme aplikace do telefonu MIT AI2 Companion, získáme možnost debuggování v reálném čase (odezva je asi 2-3 sekundy) a na rozdíl od jiných vývojových programů není třeba připojovat telefon přes kabel k PC, instalovat ovladače a další nepříjemnosti. Při změně ovládacích prvků se aplikace restartuje a změny kódu vykoná, až když je tento kód znovu spuštěn.

Tato aplikace je tedy vhodná především pro nováčky v programování. Přestože se dají v této aplikaci vytvořit malé aplikace s grafickým rozhraním, jako kompas nebo ovladač pro robota s Bluetooth, jsou možnosti této aplikace omezené.

Pokud ale vytvoříme web optimalizovaný pro telefony, je otázkou několika minut vložit do aplikace modul pro webový prohlížeč a nastavit mu pevnou adresu, kterou otevře při spuštění aplikace. Dále je možné přidat tlačítka pro navigaci.



Obrázek 13 - Náhled aplikace a nastavení tlačítek.

[9] Odstraňování chyb z programu.

3.2.6 Programovací jazyk

Pro programování v programu od Google – Android studio nebo Eclipse je potřeba ovládat jazyk Java. Je to objektově orientovaný programovací jazyk, který byl vytvořen v roce 1995 firmou SUN. Později byla firma SUN koupena firmou Oracle. V roce 2007 byly uvolněny zdrojové kódy a od té doby je vyvíjen jako open source. Byl vyroben jako multiplatformní jazyk a díky tomu mohly aplikace běžet na libovolném operačním systému a zařízení. Pro obyčejné telefony je platforma Java ME (J2ME), pro stolní počítače Java SE a pro distribuované systémy Java EE. Aby se Google vyhnul placení licenčních poplatků za využívání Javy, používá virtuální stroj Dalvik. Přesto se některé třídy v knihovně Dalviku podobají těm od Javy, a proto firma Oracle vede soudní spor s Googlem o legálnosti použití Javy v Androidu.

V případě použití výše zmíněných vývojových prostředí lze také použít jazyk C/C++. V tomto jazyce jsou psány veškeré knihovny v OS Android. Dále si můžeme napsat vlastní knihovnu, kterou následně kompilujeme a nainstalujeme. Nicméně vytvářet knihovny se doporučuje, pouze pokud to výrazně zrychlí běh programu, jako například pro engine náročné hry nebo pro složité výpočty, nikoliv proto, že v jazyce C umíme programovat lépe.

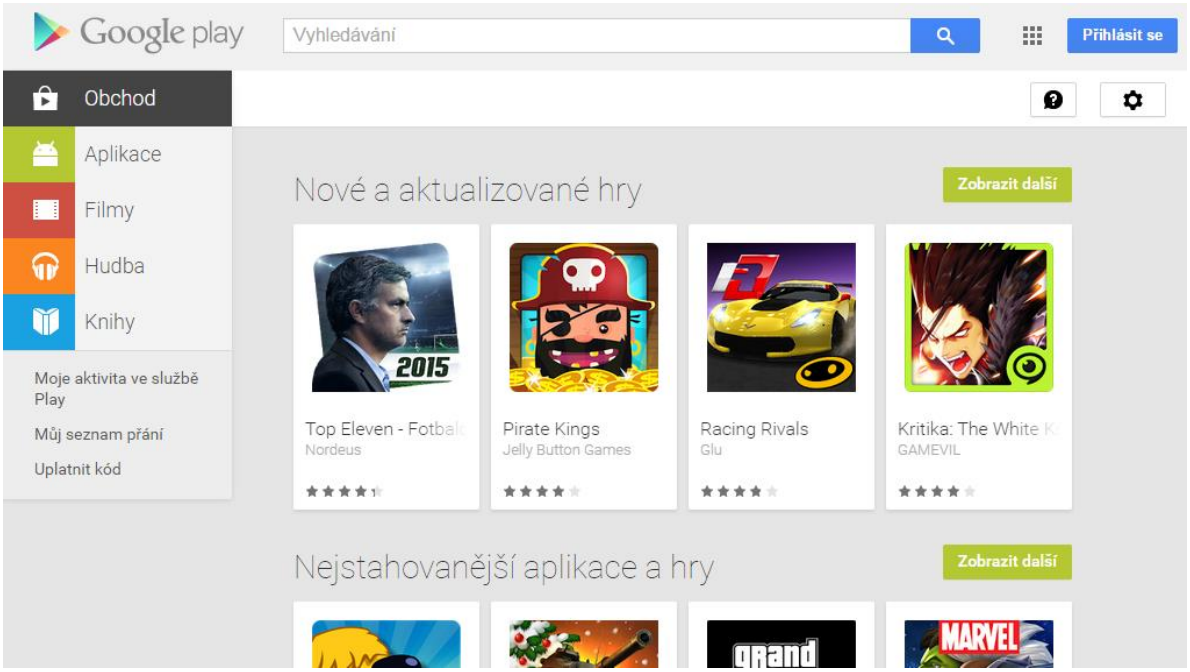
Při využití prostředí Xamarin a Unity lze použít C#. Ten je použit jako „univerzální jazyk“, který následně lze zkompilovat a vytvořit aplikace na různé platformy, jako iOS, které nativně používá Objectiv C nebo Android s Javou.

3.3 Vydání aplikace

Google play – obchod s Android aplikacemi je největší ze všech co do počtu aplikací. To je způsobeno především tím, že neprobíhá žádná kontrola aplikací, jako třeba u Apple appstore. Kvůli větší uživatelské základně má také o polovinu více stažených aplikací, ale za zakoupené aplikace inkasoval polovinu toho, co Apple [9].

Pro publikování na Google play je potřeba mít zaplacený vývojářský účet. K tomu stačí zaplatit jednorázově registrační poplatek 25$. Je možné nahrávat alpha a beta verze, které se automaticky aktualizují u testerů/zákazníků, kteří mají aplikaci staženou.

Nahrát lze pouze „podepsaná“ aplikace, k tomu je potřeba vygenerovat kód, který se musí uschovat a při jehož ztrátě nelze aplikaci updatovat ani znovu nahrát. Po průchodu alpha a beta testem a vyplnění informací a obrázků lze aplikaci publikovat.



Obrázek 14 - Vzhled Google play na PC

Další možností je využít například Amazone store, kde je potřeba splnit stejné věci jako u Android obchodu. Aplikace Amazone store je ke stáhnutí v Google play a pro instalaci aplikací z Amazone je nutné mít povolené instalování aplikací z cizích zdrojů. A pokud má uživatel tuto možnost povolenou a aplikaci chceme distribuovat zdarma, může být ke stažení i samotný .apk soubor (kterým se aplikace instaluje) i přes webové stránky jako odkaz nebo qrkód.

Pokud je dostupný FTP server nebo jiné úložiště, na kterém je jisté, že soubor vydrží, není nutné využít oficiální „store“ – obchody s aplikacemi. Samozřejmě je zde nutné povolit možnost instalací aplikací z neznámých zdrojů. Další krok po uložení je vytvořit QR kód. To se dá udělat přes webové stránky, např.: http://goqr.me. Po získání QR kódu ho lze umístit na web, vytisknout na plakáty, vizitky, trička atd. Druhá možnost je odeslat ho emailem nebo v chatu jako hypertextový odkaz - http://1drv.ms/1G63CqQ

Obrázek 15 - Odkaz pro stáhnutí aplikace z praktické části práce

4. Vlastní práce

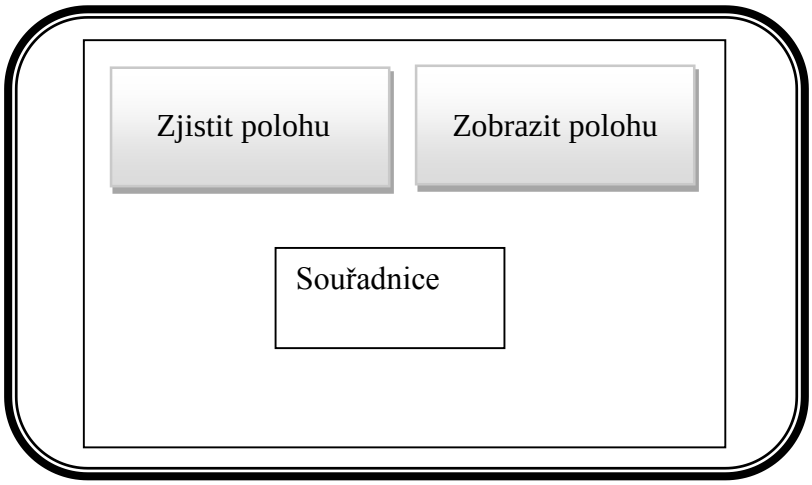
Pro vytvoření aplikace, ve které je využito zde uvedených informací a poznatků z teoretické části, byl zvolen program Android Studio. Dále se vlastní práce věnuje návrhu uživatelského prostředí, vysvětlení kódu programu a následně publikování aplikace.

4.1 Aplikace GPS Locator

Tato aplikace má za cíl nalezení vaší polohy a její zobrazení na slepé mapě ČR. Z názvu vyplývá, že jsou k nalezení umístění přístroje využity souřadnice GPS. Pro zakreslení do mapy je vložen obrázek mapy, do které je zakreslen křížek znázorňující polohu. Aplikace samozřejmě není nejpřesnější navigací ani mapou na trhu, ale má pouze velikost 2,5 MB a také nevyžaduje internetové připojení. Lze ji tedy využít kdekoliv na území ČR, kde nejsou vysílače mobilních operátorů.

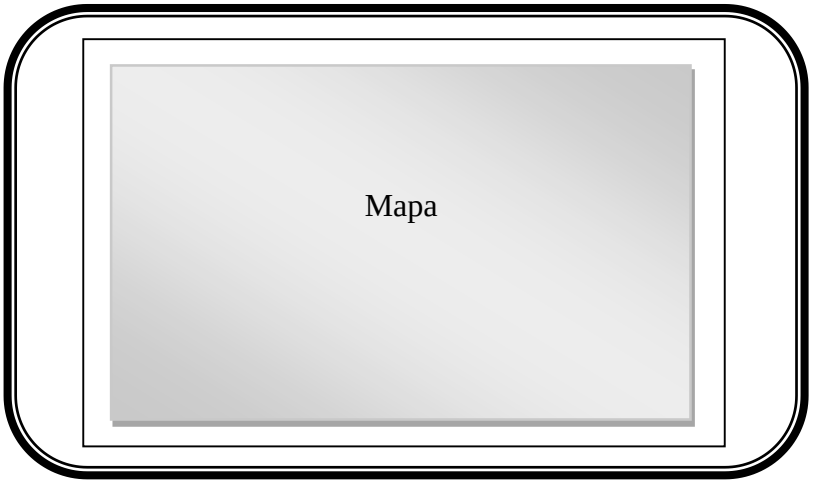
4.2 Návrh UI

Aplikace má velmi jednoduché a přehledné rozhraní. Na následujícím obrázku (Obrázek 16 - UI aplikace a výchozí obrazovka je vidět plánované rozvržení prvků na obrazovce. 2 tlačítka pro načítání dat z GPS senzoru a druhé pro zobrazení mapy. Pod tlačítky je textové pole pro výpis souřadnic. Na druhém obrázku (Obrázek 17 - UI aplikace a druhá obrazovka s mapou) je znázorněna mapa, která bude přes celou obrazovku. Aplikace je dělaná pouze v režimu landscape[10] vzhledem k rozměrům mapy České republiky.



Obrázek 16 - UI aplikace a výchozí obrazovka

[10] Režim „naležato“



Obrázek 17 - UI aplikace a druhá obrazovka s mapou

4.3 Vytvoření UI v Android studiu

Pro rozvržení byl použit Relative Layout a ten je odsazen od okrajů obrazovky. <**RelativeLayout**> umožňuje nastavení polohy ovládacích prvků, ale pro zarovnání do řádek je zbytečně komplikovaný, a tak je využit jen pro základ rozvržení. Dále je zde několik vnořených Linear Layout pro tlačítka a textové pole. První **<LinearLayout>** s orientací **"vertical"** zajišťuje řazení prvků pod sebe, zatímco každý další vnořený Linear Layout má nastavenou orientaci na **"horizontal"**, a tak se prvky řadí zleva doprava. Následující kód je část souboru activity_main.xml a vkládá tlačítka pro aplikaci. Tlačítka jsou pojmenované anglicky Button a pomocí **android:id=** se tlačítko identifikuje v případě interakce uživatele. Zpracování probíhá pomocí OnClickListener v souboru MainActivity.java.

```
<RelativeLayout>

  <LinearLayout
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:id="@+id/linearLayout" >

    <LinearLayout
      android:orientation="horizontal"
      android:layout_width="fill_parent"
      android:layout_height="120dp"
      android:layout_above="@+id/linearLayout"
      android:layout_centerHorizontal="true" >
```

```
      <Button
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:text="Zjistit polohu"
        android:id="@+id/poloha"
        android:textSize="22dp" />

      <Button
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:text="Zobraz na mapě"
        android:id="@+id/mapa"
        android:textSize="22dp" />
    </LinearLayout>
  </LinearLayout>
</RelativeLayout>
```

4.4 Java třídy

4.4.1 MainActivity.java

Tato třída nebo také aktivita vzhledem k rozšíření o android.activity se stará o využití ostatních tříd. Využívá metody z třídy poloha, které po kliknutí na tlačítko s id=poloha zjišťují polohu. Ta je poté uložena do proměnných lon a lat. Po kliknutí na tlačítko s id=mapa se vytvoří instance třídy DrawView se souřadnicemi. Tlačítka jsou kontrolovány pomocí OnClickListener, jinak řečeno při stisknutí obdrží kód informaci o stisknutí daného tlačítka. Pomocí funkce `switch(v.getId())` se aplikace rozhoduje ze dvou případů, kdy porovná ID tlačítek a vybere danou možnost. Následuje ukázka podstatných částí kódu.

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

    Button mapa = (Button) findViewById(R.id.mapa);
    Button poloha = (Button) findViewById(R.id.poloha);

    mapa.setOnClickListener(this);
    poloha.setOnClickListener(this);
  }

  @Override
  public void onClick(View v) {
    Poloha gps = new Poloha(MainActivity.this);

    //vyhledání textových polí pro zápis souřadnic
    TextView textlongtitude = (TextView)
findViewById(R.id.textlongtitude);
    TextView textlatitude = (TextView) findViewById(R.id.textLatitude);

    switch (v.getId()) {
      case R.id.mapa: //zobrazení mapy
        drawView = new DrawView(this, getIntLat(), getIntLon());
```

```
        gps.stopUsingGPS(); //vypnuti aktualizace gps
        setContentView(drawView);
        break;

      case R.id.poloha: //vlozeni dat z gps do textview
        if (gps.canGetLocation()) { // kontrola zapnutí gps
          textlongtitude.setText(String.valueOf(gps.getLongitude()));
          lon = gps.getLongitude(); //vypsání zeměpisné délky
          textlatitude.setText(String.valueOf(gps.getLatitude()));
          lat = gps.getLatitude(); //vypsání zeměpisné šířky
        } else {
          // Dotázat na zapnutí gps
          gps.showSettingsAlert();
        }
        break;
    }
  }
}
```

4.4.2 DrawView.java

Třída poloha má na starost vytvoření canvas, do kterého se vloží mapa a následně podle souřadnic zakreslí poloha. Po vytvoření instance třídy se uloží souřadnice do proměnných lon a lat. Pro výpočet bylo potřeba několik pomocných čísel. Například šířka republiky je cca 6,639° a výška 2,42°. Dále je třeba zjistit šířku displeje a podle ní vložit velkou mapu. Po získání souřadnic je od aktuální polohy odečtena hodnota nejzápadnější poledník a nejsevernější rovnoběžka protínající území ČR. Vzniklá hodnota se vydělí šířkou či výškou ČR a vznikne hodnota ukazující, v kolika % mapy bude křížek. Tuto hodnotu je nutno vynásobit šířkou vykreslené mapy. Celý kód je v příloze.

```
public class DrawView extends View {
  Paint paint = new Paint();
  double lat;   double lon;

  @Override
  public void onDraw(Canvas canvas) {
    double sirka = getWidth() * 0.9;
    double vyska = sirka*0.5833;
    double lat_range = 2.42;
double lon_range = 6.639;
    lon = lon - 12.169;
    lat = 51 - lat;
// získání hodnoty 0 – 1 v závislosti na poloze a vynásobení šířkou mapy
    double zakresLon = (lon/lon_range) * sirka;
// získání hodnoty 0 – 1 v závislosti na poloze a vynásobení výškou mapy
    double zakresLat = (lat/lat_range) * vyska;
    int zakresLon2 = (int) zakresLon; // převedení double na int
    int zakresLat2 = (int) zakresLat;

        //načtení mapy
```

```
    Bitmap pozadi= BitmapFactory.decodeResource(getResources(),
R.drawable.mapa_sidla);

        //umístění mapy do canvas
    canvas.drawBitmap(pozadi, new Rect(0,0,5000,5000),
                      new Rect(0,0,(int)sirka,(int)vyska),paint);

    paint.setStrokeWidth(4);
    paint.setColor(Color.RED);
        //zakreslení polohy do mapy
    canvas.drawLine(zakresLon2-20, zakresLat2-20,
                    zakresLon2+20, zakresLat2+20, paint);
    canvas.drawLine(zakresLon2+20, zakresLat2-20,
                    zakresLon2-20, zakresLat2+20, paint);
  }
}
```

4.4.3 Poloha.java

Tato třída pomocí metody `isGPSEnabled` zjišťuje, zda je zapnuté GPS, a pokud ne, řekne uživateli, ať služby polohy zapne. Poté se postará o aktualizování souřadnic v časovém intervalu pro úsporu baterie. Souřadnice lze získat pomocí zavolání metody `getLongitude()` a `getLatitude()`. Celý kód je umístěn v příloze.

```
  // výpis latitude(šířky)
  public double getLatitude(){
    if(location != null){
      latitude = location.getLatitude();
    }
    return latitude;
  }

  // výpis longtitude(délky)
  public double getLongitude(){
    if(location != null){
      longitude = location.getLongitude();
    }
    return longitude;
  }
```

4.5 Android manifest

Základní soubor, který musí mít každá aplikace a to v kořenovém umístění. Manifest uchovává důležité informace o aplikaci, která využívá operační systém. Mezi tyto informace patří pojmenování balíčku, pod kterým se aplikace hlásí v systému a jmenuje se tak složka ve které je nainstalována. Také popisuje aktivity a služby, oprávnění, verzi sdk atd.

V tomto soboru je pro běh aplikace využívající GPS nezbytné přidat řádky obsahující

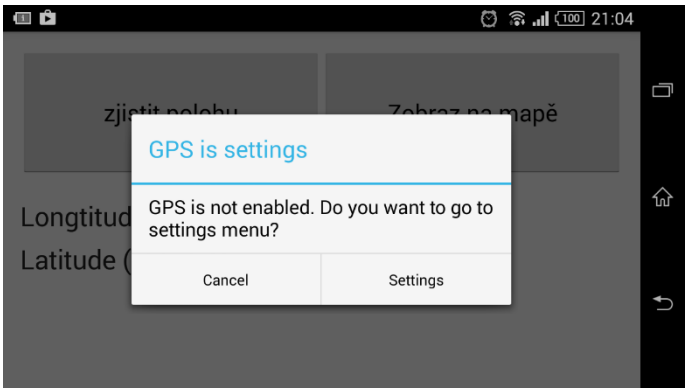
```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
```

Tento kód zajistí, aby aplikace měla přístup k souřadnicím, v opačném případě by se nepodařilo souřadnice načíst. Také se žádost o oprávnění polohy zobrazuje při instalaci aplikace.

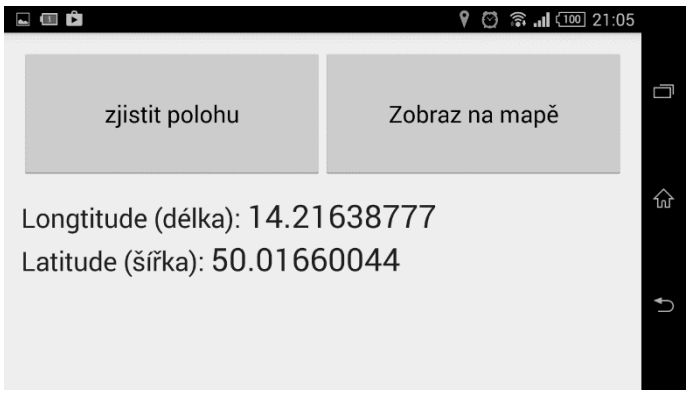
Další důležitá část manifestu je minSDKVersion. `<uses-sdk android:minSdkVersion="16" />` znamená, že aplikace půjde spustit pouze na zařízení s API level 16. To odpovídá verzi Android 4.1. Cílem je vyvíjet aplikaci pro nejnižší API aby aplikace pokryla největší možnou část trhu, ale zároveň s využitím nových knihoven a funkcí z novějších verzi Androidu se musí verze API zvyšovat. Této problematice se práce věnuje na straně 11 (3.1.2 Fragmentace). Překvapit může také to, že tato informace občas nebývá v manifestu, ale v Gradle.build souboru. Pokud je verze sdk vyplněna v obou souborech, Gradle je upřednostněn.

4.6 Vzhled finální aplikace

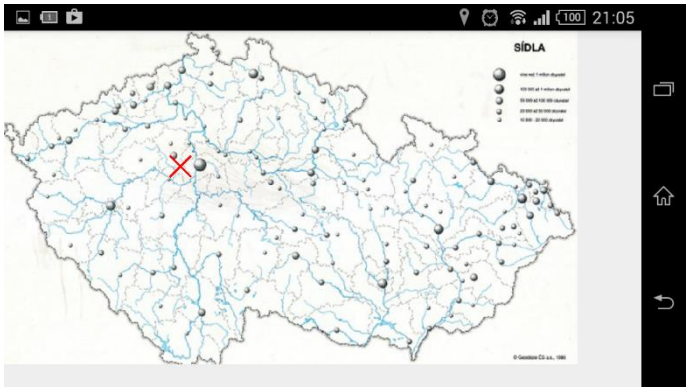
Aplikace se skládá z 2 obrazovek. První, na které je ovládání (Obrázek 18, Obrázek 19) a zobrazování souřadnic. Druhá s mapou a aktuální polohou (Obrázek 20)



Obrázek 18 - Vyhledávání s vypnutou GPS - oznámení



Obrázek 19 - Po nalezení polohy



Obrázek 20 - Zobrazení mapy s polohou

Zdroj mapy: http://ostrava-educanet.cz/svoboda/vyuka/septima/testy7.htm

5. Výsledky a diskuze

V teoretické části je probráno několik vývojových prostředí. Mezi nimi je uživatelsky přívětivý App Inventor, který je doporučen pro začínající programátory, popřípadě zájemce o programování. V něm se naučí práci se základními příkazy typu if, for a nebo while. Pro vývoj kvalitnější aplikace na OS Android, je zřejmě nejlepší volbou Android studio. Výsledná aplikace bude menší vzhledem k importování pouze nutných knihoven a také lze psát samotný kód na rozdíl od využívání bloků.

Praktická část byla zaměřena na vytvoření aplikace využívající GPS data k určení polohy. Tato aplikace plní svůj účel hledání polohy dokonale, přesto je řada možností, které by ji vylepšily. Například možnost odeslat SMS zprávu s aktuálními souřadnicemi nebo hypertextovým odkazem na Google mapy. Další možnost je přidání online funkce, kde by bylo možné nainstalovat aplikaci na více telefonech, které by si přes datový soubor na FTP, popř. jinde vyměňovaly polohu a zakreslovali ji do map ostatních zařízení. Také by šlo přidat mapu světa a další funkce.

6. Závěr

V první kapitole se tato práce věnuje OS Android, jeho historii, aktuálním verzím a problémům vývoje aplikace. Druhá kapitola uvažuje možnosti vývojových prostředí od odlehčených typu App Inventor po plnohodnotné jako Eclipse nebo Android studio. Dále práce seznamuje s možnostmi distribuce a šíření aplikace.

Poznatky získané při zpracování teoretické části byly spolu s osobními zkušenostmi využity při vytváření aplikace a praktické části práce. Pro vývoj aplikace byl použit vlastní mobilní telefon Sony a počítač s OS Windows, který sloužil k testování aplikace a jejímu vývoji. Aplikace GPS Locator byla vytvořena pomocí Android studia a s využitím jazyku Java. Její zdrojový kód včetně komentářů se nachází v příloze práce.

Cílem této práce bylo shrnutí všech potřebných znalostí pro započetí vývoje aplikace pro mobilní telefon včetně seznamu vývojových prostředí, použitelných programovacích jazyků i možností distribuce aplikace. Mnoho zdrojů je v angličtině, a tak touto prací usnadním českým zájemcům o programování pro Android hledání informací.

Hlavním cílem bylo představení vývoje aplikací na mobilní zařízení s OS Android. Tento cíl byl splněn.

7. Seznam použitých zdrojů:

7.1 Literatura a internetové zdroje

1. PC Life. Android, historie a funkce aneb chytrý mobilní operační systém [online] [cit. 15. 7. 2014]. http://www.pclife.cz/256-android-historie-a-funkce-aneb-chytry-mobilni-operacni-system/
2. Google Android developer. Dashboard [online] [cit. 16. 7. 2014]. http://developer.android.com/about/dashboards/index.html
3. Mobilenet.cz. Co umí který android. Jakub Vitásek [online] [cit. 17. 7. 2014]. http://mobilenet.cz/clanky/co-umi-ktery-android-prinasime-vam-prehled-jeho-verzi-5778
4. Google Android developer. Design [online] [cit. 19. 7. 2014]. https://developer.android.com/design/index.html
5. O MIT App Inventor [online] [cit. 22. 1. 2015]. http://appinventor.mit.edu/explore/about-us.html
6. Unity – podporované platformy [online] [cit. 26. 1. 2015]. http://unity3d.com/unity/multiplatform
7. Unity programovací jazyk [online] [cit. 26. 1. 2015]. http://forum.unity3d.com/threads/best-programming-language-in-unity.43150/
8. Vývoj aplikací pro android [online] [cit. 27. 1. 2015]. http://www.zdrojak.cz/clanky/vyvijime-pro-android-zaciname/
9. Apple App Store vs Google Play [online] [cit. 27. 1. 2015]. http://www.ibtimes.co.uk/apple-app-store-vs-google-play-revenue-downloads-1470328
10. Google Inc. Android API Levels. [Online] [cit: 28. 1. 2015]. http://developer.android.com/guide/appendix/api-levels.html
11. Android location and gps [Online] [cit: 29. 1. 2015]. http://www.androidhive.info/2012/07/android-gps-location-manager-tutorial/

8. Přílohy

8.1 Seznam obrázků

8.2 Seznam tabulek

8.3 Seznam příloh:

Příloha č. 1 – Zdrojový kód souboru activity_main.xml

```
<RelativeLayout
  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"
  android:paddingLeft="@dimen/activity_horizontal_margin"
  android:paddingRight="@dimen/activity_horizontal_margin"
  android:paddingTop="@dimen/activity_vertical_margin"
  android:paddingBottom="@dimen/activity_vertical_margin"
  tools:context=".MainActivity" >

  <LinearLayout
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:id="@+id/linearLayout" >

    <LinearLayout
      android:orientation="horizontal"
      android:layout_width="fill_parent"
      android:layout_height="120dp"
      android:layout_above="@+id/linearLayout"
      android:layout_centerHorizontal="true" >

      <Button
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:text="zjistit polohu"
        android:id="@+id/poloha"
        android:textSize="22dp" />

      <Button
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:text="Zobraz na mapě"
        android:id="@+id/mapa"
        android:textSize="22dp" />
    </LinearLayout>

    <LinearLayout
      android:layout_marginTop="15dp"
      android:orientation="horizontal"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:layout_gravity="center_horizontal">

      <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
         android:textAppearance="?android:attr/textAppearanceLarge"
        android:text="Longtitude (délka): "
        android:textSize="25sp"
        android:id="@+id/textView" />

      <TextView
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="30sp"
         android:textAppearance="?android:attr/textAppearanceLarge"
        android:text="0"
        android:id="@+id/textlongtitude" />
    </LinearLayout>

    <LinearLayout
      android:orientation="horizontal"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:layout_gravity="center_horizontal">

      <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
         android:textAppearance="?android:attr/textAppearanceLarge"
        android:text="Latitude (šířka): "
        android:textSize="25sp"
        android:id="@+id/textView3" />

      <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="30sp"
         android:textAppearance="?android:attr/textAppearanceLarge"
        android:text="0"
        android:id="@+id/textLatitude" />
    </LinearLayout>
  </LinearLayout>
</RelativeLayout>
```

Příloha č. 2 – Zdrojový kód souboru MainActivity.java

```
package com.example.tomas.gpslocator;

//načtení Java knihoven
import android.os.Bundle;
import android.support.v4.app.FragmentActivity;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

public class MainActivity extends FragmentActivity implements
View.OnClickListener {

  double lon;
  double lat;

  double intlon;
  public double getIntLon() {
    return intlon;
  }
  public void setIntLon(double lon) {
    this.intlon = lon;
```

```
  }

  double intlat;
  public double getIntLat() {
    return intlat;
  }
  public void setIntLat(double lat) {
    this.intlat = lat;
  }

  DrawView drawView;

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

    Button mapa = (Button) findViewById(R.id.mapa);
    Button poloha = (Button) findViewById(R.id.poloha);

    mapa.setOnClickListener(this);
    poloha.setOnClickListener(this);
  }

  @Override
  public void onClick(View v) {
    Poloha gps = new Poloha(MainActivity.this);

    //vyhledání textových polí pro zápis souřadnic
    TextView textlongtitude = (TextView)
findViewById(R.id.textlongtitude);
    TextView textlatitude = (TextView) findViewById(R.id.textLatitude);

    switch (v.getId()) {
      case R.id.mapa: //zobrazení mapy

drawView = new DrawView(this, getIntLat(), getIntLon());
        gps.stopUsingGPS(); //vypnuti aktualizace gps
        setContentView(drawView);
        break;

      case R.id.poloha: // vlozeni dat z gps(Poloha) do textview v
aktivite...
        if (gps.canGetLocation()) { // kontrola zapnutí gps
          textlongtitude.setText(String.valueOf(gps.getLongitude()));
          lon = gps.getLongitude(); //vypsání zeměpisné délky
          textlatitude.setText(String.valueOf(gps.getLatitude()));
          lat = gps.getLatitude(); //vypsání zeměpisné šířky

          setIntLon( gps.getLongitude()); //uložení souřadnic do proměnné
          setIntLat( gps.getLatitude());
        } else {
          // Dotázat na zapnutí gps
          gps.showSettingsAlert();
        }
        break;
    }
  }
}
```

Příloha č. 3 – Zdrojový kód souboru DrawView.java

```
package com.example.tomas.gpslocator;

import android.content.Context;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.graphics.Rect;
import android.view.View;

public class DrawView extends View {
  Paint paint = new Paint();

  double lat;
  double lon;

  public DrawView(Context context, double lat, double lon) {
    super(context);
    this.lat = lat; //uložení souřadnic při vytvoření instance
    this.lon = lon;
  }

  @Override
  public void onDraw(Canvas canvas) {
    double sirka = getWidth() * 0.9;
    double vyska = sirka*0.5833;
    double lat_range = 2.42;
double lon_range = 6.639;
lon = lon - 12.169;
    lat = 51 - lat;
// získání hodnoty 0 - 1 v závislosti na poloze a vynásobení šířkou mapy
    double zakresLon = (lon/lon_range) * sirka;
// získání hodnoty 0 - 1 v závislosti na poloze a vynásobení výškou mapy
    double zakresLat = (lat/lat_range) * vyska;
    int zakresLon2 = (int) zakresLon; // převedení double na int
    int zakresLat2 = (int) zakresLat;
    //načtení mapy
    Bitmap pozadi= BitmapFactory.decodeResource(getResources(),
R.drawable.mapa_sidla);

    //umístění mapy do canvas
    canvas.drawBitmap(pozadi, new Rect(0,0,5000,5000),
                      new Rect(0,0,(int)sirka,(int)vyska),paint);

    paint.setStrokeWidth(4);
    paint.setColor(Color.RED);
    //zakreslení polohy do mapy
    canvas.drawLine(zakresLon2-20, zakresLat2-20, zakresLon2+20,
zakresLat2+20, paint);
    canvas.drawLine(zakresLon2+20, zakresLat2-20, zakresLon2-20,
zakresLat2+20, paint);
  } }
```

Příloha č. 4 – Zdrojový kód souboru Poloha.java

```
package com.example.tomas.gpslocator;

import android.app.Service;
import android.content.Context;
import android.content.Intent;
import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;
import android.os.Bundle;
import android.os.IBinder;
import android.app.AlertDialog;
import android.content.DialogInterface;
import android.provider.Settings;
import android.util.Log;

public class Poloha extends Service implements LocationListener {

  private final Context mContext;

  boolean isGPSEnabled = false;
  boolean isNetworkEnabled = false;
  boolean canGetLocation = false;
  Location location; // location
  double latitude; // latitude
  double longitude; // longitude

  // minimální vzdálenost pro aktualizaci
  private static final long MIN_DISTANCE_CHANGE_FOR_UPDATES = 10;

  // minimální čas pro aktualizaci
  private static final long MIN_TIME_BW_UPDATES = 1000 * 60 * 1;

  // Deklarování Location Manager
  protected LocationManager locationManager;

  public Poloha(Context context) {
    this.mContext = context;
    getLocation();
  }

  public Location getLocation() {
    try {
      locationManager = (LocationManager) mContext
          .getSystemService(LOCATION_SERVICE);

      // Test zapnutí gps
      isGPSEnabled = locationManager
          .isProviderEnabled(LocationManager.GPS_PROVIDER);

      // test sítě
      isNetworkEnabled = locationManager
          .isProviderEnabled(LocationManager.NETWORK_PROVIDER);

      if (!isGPSEnabled && !isNetworkEnabled) {
        // pokud není signál
      } else {
```

```
        this.canGetLocation = true;
        // získání polohy od operátora
        if (isNetworkEnabled) {
          locationManager.requestLocationUpdates(
              LocationManager.NETWORK_PROVIDER,
              MIN_TIME_BW_UPDATES,
              MIN_DISTANCE_CHANGE_FOR_UPDATES, this);
          Log.d("Network", "Network");
          if (locationManager != null) {
            location = locationManager
                .getLastKnownLocation(LocationManager.NETWORK_PROVIDER);
            if (location != null) {
              latitude = location.getLatitude();
              longitude = location.getLongitude();
            }
          }
        }
        // pokud je zapnutá gps
        if (isGPSEnabled) {
          if (location == null) {
            locationManager.requestLocationUpdates(
                LocationManager.GPS_PROVIDER,
                MIN_TIME_BW_UPDATES,
                MIN_DISTANCE_CHANGE_FOR_UPDATES, this);
            Log.d("GPS Enabled", "GPS Enabled");
            if (locationManager != null) {
              location = locationManager
          .getLastKnownLocation(LocationManager.GPS_PROVIDER);
              if (location != null) {
                latitude = location.getLatitude();
                longitude = location.getLongitude();
              }
            }
          }
        }
      } catch (Exception e) { e.printStackTrace(); }
      return location;
    }

    //ukoncení načítání dat z gps
    public void stopUsingGPS(){
      if(locationManager != null){
        locationManager.removeUpdates(Poloha.this);
      }
    }

    // výpis latitude(šířky)
    public double getLatitude(){
      if(location != null){
        latitude = location.getLatitude();
      }
      return latitude;
    }

    // výpis longtitude(délky)
    public double getLongitude(){
      if(location != null){
        longitude = location.getLongitude();
```

```
    }
    return longitude;
  }
  //test zapnuté gps
  public boolean canGetLocation() {
    return this.canGetLocation;
  }


  @Override
  public void onLocationChanged(Location location) {
  }

  @Override
  public void onProviderDisabled(String provider) {
  }

  @Override
  public void onProviderEnabled(String provider) {
  }

  @Override
  public void onStatusChanged(String provider, int status, Bundle extras)
{
  }

  @Override
  public IBinder onBind(Intent arg0) {
    return null;
  }
}
```