Katedra informatiky
Přírodovědecká fakulta
Univerzita Palackého v Olomouci

BAKALÁŘSKÁ PRÁCE

Aplikace Spotting pro OS Android

2015

Milan Jiříček

Vedoucí práce: Mgr. Jiří Zacpal, Ph.D.

Studijní obor: Aplikovaná informatika, prezenční forma

Bibliografické údaje

Autor:	Milan Jiříček
Název práce:	Aplikace Spotting pro OS Android
Typ práce:	bakalářská práce
Pracoviště:	Katedra informatiky, Přírodovědecká fakulta, Univerzita Palackého v Olomouci
Rok obhajoby:	2015
Studijní obor:	Aplikovaná informatika, prezenční forma
Vedoucí práce:	Mgr. Jiří Zacpal, Ph.D.
Počet stran:	42
Přílohy:	1 CD/DVD
Jazyk práce:	český

Bibliograhic info

Author:	Milan Jiříček
Title:	Application Spotting for Android
Thesis type:	bachelor thesis
Department:	Department of Computer Science, Faculty of Science, Palacký University Olomouc
Year of defense:	2015
Study field:	Applied Computer Science, full-time form
Supervisor:	Mgr. Jiří Zacpal, Ph.D.
Page count:	42
Supplements:	1 CD/DVD
Thesis language:	Czech

Anotace

Práce se zabývá vývojem aplikace pro systém Android, která umožní uživatelům vyhledávání míst pro skateboardery, bikery a bruslaře. Pomocí ní lze vkládat fotky tzv. spotů a skate parků a tato místa vyhledávat a prohlížet podle různých filtrů, jako je vyhledávání podle aktuální polohy, druhu sportu a dalších aspektů. Aplikace dále umožňuje zobrazení vybraných míst na mapě a je zveřejněna pomocí služby Google Play.

Synopsis

The topic of this thesis is an application development for Android operating system. The application allows the users searching for places for skateboarders, bikers and roller-skaters. It offers the possibilities to insert photos of spots and skate parks as well as to search for these places using different filters; current position, sport type and other aspects. The application also allows displaying chosen places on a map and is published via Google Play service.

Klíčová slova: Android; mobilní aplikace

Keywords: Android; mobile application

Děkuji vedoucímu bakalářské práce panu Mgr. Jiřímu Zacpalovi, Ph.D. za konzultace a odborné vedení. Dále bych chtěl poděkovat rodině za jejich podporu při studiích.

Místopřísežně prohlašuji, že jsem celou práci včetně příloh vypracoval/a samostatně a za použití pouze zdrojů citovaných v textu práce a uvedených v seznamu literatury.

datum odevzdání práce podpis autora

Obsah

Seznam obrázků

Seznam tabulek

Seznam vět

Seznam zdrojových kódů

1 Úvod

Cílem bakalářské práce je tvorba mobilní aplikace pro platformu Android, která umožní uživatelům vyhledání míst pro skateboardery, bikery a bruslaře. S rostoucím počtem chytrých telefonů, jsou dnes již tyto přístroje dostupné prakticky komukoliv. Díky tomu mohou aplikace vytvořené pro tyto telefony ulehčit každodenní činnosti a život bez nich si už neumíme představit.

V dnešní době patří mezi nejrozšířenější operační systémy pro chytré telefony Android, iOS a Windows Phone.

Jak je již psáno, pro vývoj aplikace byla zvolena platforma operačního systému Android od společnosti Google. Jedním z kritérií pro volbu OS Android je jeho rozšířenost na trhu, kdy za celý rok 2014 se prodala více než miliarda zařízení s tímto operačním systémem. Dalším z důvodů je osobní vlastnictví mobilního zařízení s tímto systémem a několikaletá zkušenost s každodenním užíváním. V neposlední řadě jsou aplikace pro Android vyvíjeny v programovacím jazyce Java, který je mi velice blízký.

Aplikace, která je pojmenována Spotting, bude sloužit pro vyhledání, přidání a prohlížení míst (tzv.spotů). Spoty bude možno zobrazovat jako místa určená zeměpisnými souřadnicemi na mapě. Ke spotům bude možno vkládat jejich fotografie, které budou moci být otagovány. Aplikace bude obsahovat funkce pro vyhledávání a prohlížení spotů podle různých filtrů, jako je vyhledávání podle aktuální polohy, druhu sportu a dalších aspektů. Podrobnější funkce aplikace jsou popsány v praktické části bakalářské práce.

2 Přehled již existujících řešení

Před výběrem bakalářské práce bylo nutné zjistit, jestli již neexistují podobné aplikace. Vybrané aplikace byly staženy z Google Play a poté nainstalovány a zkoušeny na mobilním telefonu HTC Evo 3D s verzí OS Android 4.0.3.

- Instagram - jako první zmiňuji právě Instagram a to z důvodu, že klientská aplikace využije filosofie designu v současné době této populární aplikace. Ta umožňuje svým uživatelům sdílení fotografií. Formát fotografie je odlišný a pořízené snímky jsou ve čtvercovém formátu. Budu využívat i podobné menu, které se skládá z pěti tlačítek v dolní části obrazovky (viz. obr. 1). Dané aplikace se budou od sebe odlišovat převážně ve funkčnosti.

- SkateSpots - je tvořena dvěma částmi menu a pracovní plochou. Hlavní menu je koncipováno formou pěti ikonek ve spodní části displeje. První zleva je hlavní kanál, kde vidíme fotografie posledně nahraných spotů. Hned vedle vpravo se nachází tlačítko, které zobrazí mapu s vyznačenými spoty. Dále pro přidání spotu, kde po spuštění mi aplikace přestala pracovat na mobilním telefonu HTC Evo 3D, ale na jiným zařízení fungovalo bez problému. Poté se nachází seznam oblíbených a poslední zobrazí námi nahrané spoty. Z GUI[1] nemám téměř co vytknout. V dané aplikaci mi převážně chybí rozsáhlejší vyhledávání míst podle důležitých aspektů např. do určité vzdálenosti, druhu místa a sportu, daného povrchu, obtížnosti atd. a poté zobrazení vyhledaných spotů, které si bude uživatel moct projít a vybrat si dle sebe.

- Tambalea Skate Spots - po prvním spuštění se zobrazí mapa s vyznačenými spoty, nad ní je menu tzv. Tab bar skládající se ze 4 záložek. V dolní části se nachází tři tlačítka, které jsou nevkusně zvolené obrázky ložiska a skateboardu s popiskem. Za další záložkou se mapa nahradila seznamem spotů, které jsou řazeny pod sebou. Jeden řádek je rozvržen tak, že na levé straně je ikonka označující typ a vedle ní doplňující informace. Vyhledávání je zde velice obtížné. Chybí zde obrázek spotu, který u vyhledávání hraje jednu z nejvýznamnějších rolí. V dolní části obrazovky se nachází tlačítka pro filtrování seznamu spotů, která jsou opět nevkusně zvolena jako obrázky skateboardů. V další záložce pojmenované „MY SPOTS“, by uživatel očekával pouze sebou nahrané, ale nachází se zde i tlačítko pro přidání, které by se hodilo spíše do horního menu. Při přidávání se musíme proklikat mapou pro získání polohy a až poté můžeme pokračovat. Zde mi chybí využití GPS modulu, který je integrovaný ve většině mobilních zařízení. V poslední záložce nastavení se můžeme odhlásit. Po znovu spuštění aplikace se musíme znovu přihlásit, což je velmi otravné.

Analýza aplikací na trhu poukázala na některé nedostatky. K hlavním nedostatkům patří vzhled uživatelského rozhraní, který má rozhodující význam při výběru

[1] Grafické uživatelské rozhraní.

Obrázek 1: Úvodní obrazovka aplikace Instagram.

aplikace uživatele a následným používáním. Dále byly vynechány některé funkce, které by aplikace tohoto typu měla mít. Testování přineslo mnohá ponaučení při vývoji aplikace.

3 Platforma Android

Platforma Android je operační systém založený na Linuxovém jádru, která je v první řadě určená pro mobilní zařízení:

- chytré telefony
- navigace
- PDA

Ale můžeme se s ním setkat i např. v MP3 přehrávačích a náramkových hodinkách. Od jádra přejímá funkce zajišťující bezpečnost systému, správu procesů, paměti a sítí a ovladače všech vnitřních senzorů a komponent. Systém je dostupný jako open source a tím se liší od konkurenčních firem.

Při vývoji pro mobilní zařízení, se programátor potýká např. s menší výkonností a velikostí paměti v porovnání se stolními počítači. V tomto se snaží Android o kompromis a umožňuje psát aplikace v programovacím jazyce Java a používat většinu knihoven, které se používají pro stolní počítače.

3.1 Krátká historie

Kalifornská společnost Android Inc. byla založena v roce 2003. Lidé, kteří stáli u jejího zrodu jsou Andy Rubin, Nick Sears, Richard Miner a Chris White.

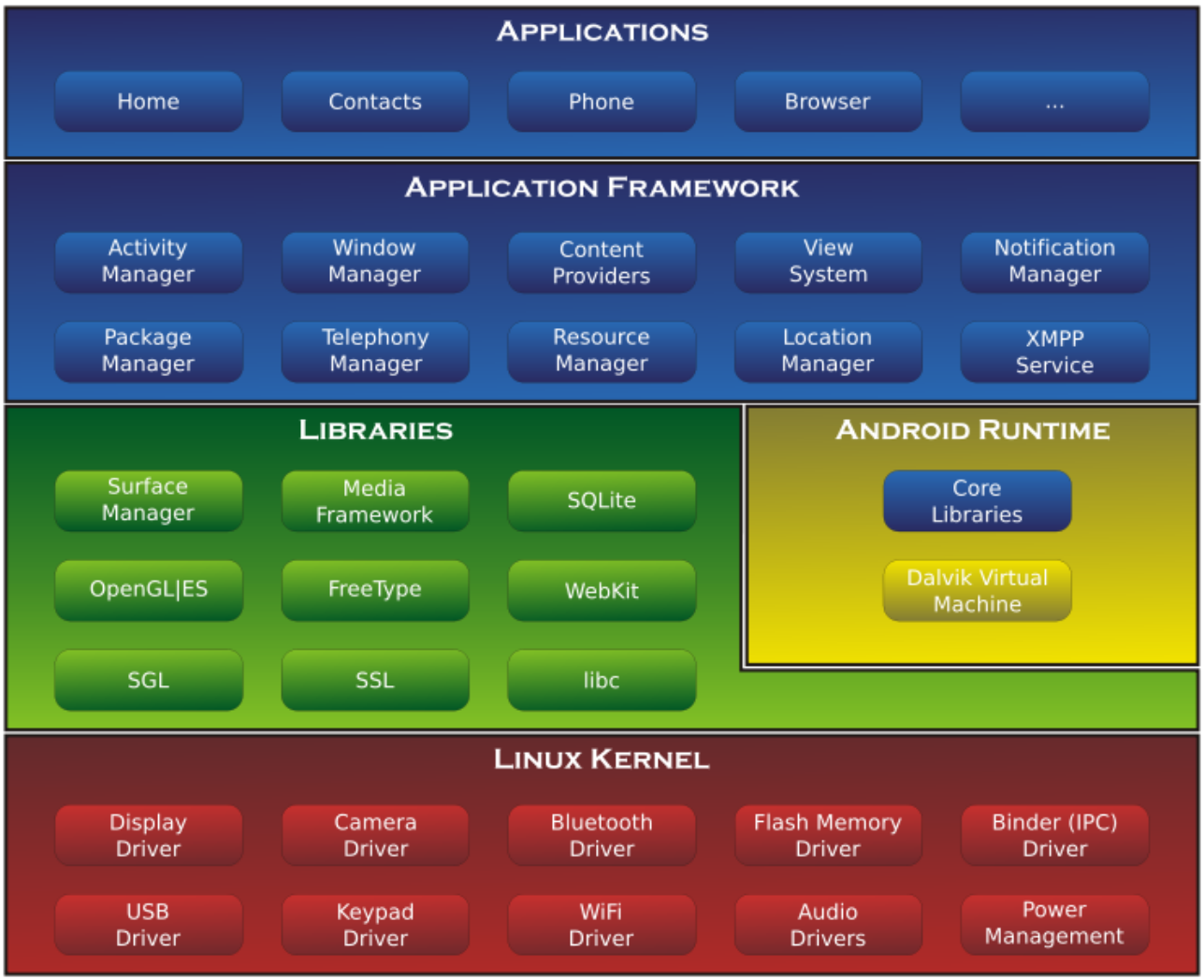
V roce 2005 byla prodána společnosti Google Inc. a stala se tak jednou z jejich dceřiných společností.[1]

V roce 2007 byla vytvořena skupina Open Handset Alliance. Ta byla složena ze společností zabývající se výrobou mobilních zařízení, čipů nebo aplikací. V jejichž čele stojí společnosti jako je např. Google, HTC, Samsung, Motorola a mnoho dalších. V rámci Open Handset Alliance vydala produkt Android, jako open source, který je nezávislý na použitém hardwaru. Díky tomu, že je open source, výrobci telefonů nemusejí platit za licenci pro jeho použití. To se odráží na cenách telefonů a může být díky tomu i v low-end zařízeních[2]. Tím získalo větší počet uživatelů a dominanci na trhu.

[2] zařízení v nízké cenové kategorii

3.2 Architektura platformy Android

Architektura platformy Android se skládá z pěti vrstev a je znázorněna na obr. 2. Níže popíši jednotlivé vrstvy.



Obrázek 2: Architektura Androidu[2]

- Linuxové jádro - nejnižší vrstvou architektury je jádro operačního systému, která zajišťuje komunikaci mezi hardwarem a softwarem. Základ tvoří ovladače, které zajišťují již zmiňovanou komunikaci. Dále poskytuje další základní služby jako jsou např.:
  - správa paměti
  - správa procesů
  - správa napájení

- Knihovny - nativní knihovny androidu jsou základními funkcemi. Obsahují např.
  - Open GL knihovnu pro práci s 3D grafikou.
  - Open SGL pro práci s 2D grafikou.
  - SQLite slouží jako relační databáze pro ukládání a práci s daty.
  - Knihovna médií pro práci s mediálními soubory. Obsahuje kodeky pro různé formáty audia a videa. Například formáty AAC, MPEG4, JPG, PNG.
  - FreeType vektorové a bitmapové vykreslování písma.

  Tyto funkce jsou poskytnuty vývojářům za pomocí Android Application Framework.

- Běhové prostředí Android - vrstva slouží pro běh aplikací, které jsou psány v programovacím jazyce Java. Ty jsou poté přeloženy do Java byte kódu a následně do mezikódu za pomocí Dalvik kompilátoru. Výsledný byte kód je spuštěný na virtuálním stroji Dalvik. Je navržený tak, že každá aplikace je samotný proces s vlastní instancí virtuálního stroje. Jedním z důvodů, proč nebyl zvolený virtuální stroj (JVM), jsou licenční podmínky. Dalším z důvodů pro zavedení Dalvik byla optimalizace virtuálního stroje pro potřeby mobilních zařízení. Je zřejmé, že hlavní roli hrála úspora energie a především výkon. Obsahem této vrstvy jsou i knihovny programovacího jazyka Java, ale také knihovny pro zařízení se systémem Android.

- Aplikační rámec - umožňuje přístup k různým službám, které vývojáři mohou používat ve svých aplikacích např.
  - tvorba uživatelského rozhraní.
  - používat hardware zařízení.
  - spouštět jiné aplikace na pozadí.

- Aplikační vrstva - je nejvyšší vrstva, ve které jsou spuštěny samotné aplikace.

3.3 Volba úrovně API

Za dobu co je systém Android na trhu, vzniklo několik verzí právě tohoto OS. Zavedl se standart pro psaní programů pro konkrétní verzi systému tzv. úroveň API[3]. Proto je před vývojem aplikace velmi důležitá volba cílové verze systému android resp. volba úrovně API. Může zde dojít k problému, kdy zvolíme příliš

[3]Reprezentuje celočíselnou hodnotu, která jednoznačně identifikuje verzi revize programového vybavení, který je poskytován prostřednictvím dané verze OS. Její hlavní roli je zajištění kompatibility mezi aplikacemi a přístroji, do kterých mají být nainstalovány.

novou verzi. Tím můžeme přijít o potenciální zákazníky, kteří mohou naši aplikaci využívat.

Google zveřejňuje statistiky zastoupení jednotlivých verzí OS Android, které přistupují do služby Google Play[4] (viz. tabulka 1). Na základě těchto údajů se vývojář může rozhodnout, kterou úroveň API zvolí. Pro svoji aplikaci jsem zvolil úroveň API 15 a vyšší. Jedním z důvodů je, že funkce obsažené v knihovnách jsou dostatečné pro můj program a také bude fungovat na více než 90% zařízeních.

Verze	**Kód. označení**	**API**	**Podíl na trhu**
2.2	Froyo	8	0.4%
2.3.3 – 2.3.7	Gingerbread	10	6.9%
4.0.3 – 4.0.4	Ice Cream Sandwich	15	5.9%
4.1.x	Jelly Bean	16	17.3%
4.2.x	Jelly Bean	17	19.4%
4.3	Jelly Bean	18	5.9%
4.4	KitKat	19	40.9%
5.0	Lollipop	21	3.3%

Tabulka 1: Údaje verzí Android přistupující do služby Google Play během sedmi dní k 2. 3. 2015[3]

[4]Služba pro stahování hudby, filmů, knih, aplikací a her pro platformu Android, kterou oficiálně provozuje společnost Google.

4 Potřebné nástroje při vývoji aplikace

Vývoj pro platformu Android může být různý. Nejčastěji se používá jazyk Java společně s XML a balíčkem vývojových nástrojů tzv. SDK. Lze využít i další možnost jako je Android NDK, což je balík pro vývoj v jazyce C/C++. V této práci se zabývám vývojem právě v jazyce Java, protože použití NDK, se má pouze v případě, pokud to není nezbytně nutné pro fungování aplikace.

4.1 Java Development Kit

Java Development Kit (dále JDK) od firmy Oracle, je nezbytný pro vývoj v jazyce Java. Tvoří soubor základních knihoven a nástrojů. Základní součástí je Java Runtime Environment (dále JRE), který slouží pro spouštění aplikací i vývojových nástrojů a obsahuje virtuální stroj a sadu základních knihoven pro vývoj. Nástroje jsou k dispozici pro operační systémy Linux, Mac OS X, Solaris a Windows.

Některé nástroje obsažené v JDK:

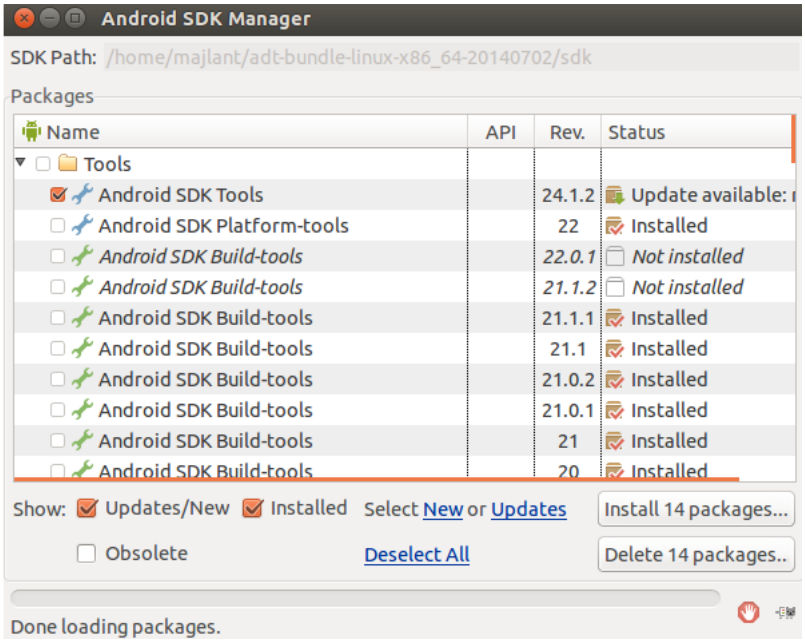
- java - spouštěč přeloženého Java byte kódu.
- javac - překladač, umožňuje převod java souboru do Java byte kódu.
- jdb - debugger pro ladění programů.
- javadoc - generátor dokumentace ze zdrojových kódů.
- jar - správce archivů JAR.

4.2 Software Development Kit

Software Development Kit (dále SDK), je balíček nástrojů pro vývoj aplikací pro platformu Android. Nástroje jsou k dispozici pro operační systémy Linux, Mac OS X a Windows. Dále si ukážeme nejpoužívanější nástroje.

4.2.1 SDK manager

Nástroj, který slouží ke správě vašeho SDK balíčku. Poskytuje informace o již nainstalovaných nástrojích a zároveň nám hlídá, aby byly aktuální.



Obrázek 3: SDK manager ukazující informace o nástrojích.

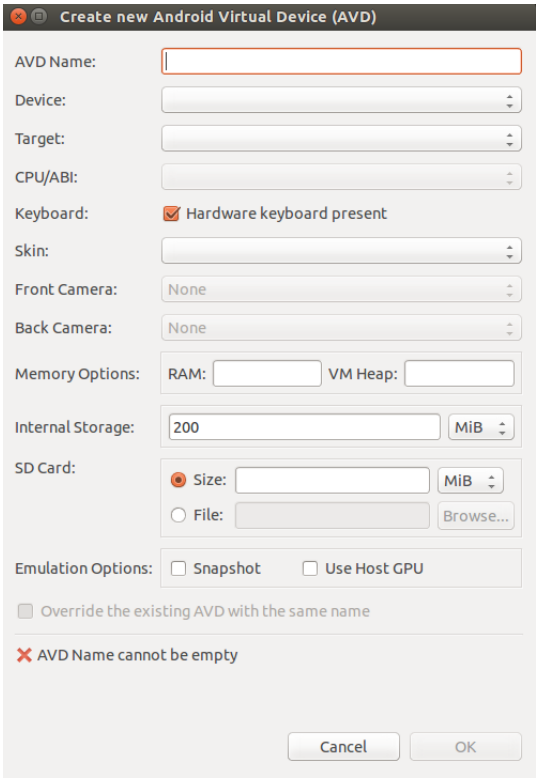
4.2.2 AVD

Nedílnou součástí při implementování je ladění, ke kterému je potřeba, aby aplikace byla spuštěna na nějakém zařízení. K tomu nám může sloužit zařízení, které je připojené přes USB[5] a je potřeba přepnout do vývojářského módu, nebo můžeme využít emulátoru, který se nazývá Android Virtual Device (dále AVD). Je tedy možné spustit aplikaci a nasimulovat chování téměř všech zařízení s OS Android.

K jeho vytvoření a nastavení konfigurace hardwaru a softwaru slouží AVD manager. Hardwarovou konfiguraci lze měnit dle požadavků pro náš vývoj. Můžeme si nastavit různé parametry. Mezi nejdůležitější patří rozlišení obrazovky, velikost paměti, zda bude mít fotoaparát a další. Z hlediska softwaru se nastavuje úroveň API.

Při vývoji pro Android se bez emulátorů neobejdeme a to z důvodu, že je mnoho zařízení s různými typy obrazovek právě pro tuto platformu. Na obrázku 4, můžeme vidět, jak se vytváří nové AVD.

[5]Univerzální sériová sběrnice. Port pro připojení periferií, přenosných datových úložišť a dalších zařízení.



Obrázek 4: Tvorba AVD.

4.2.3 Dalvik Debug Monitor Server

Dalvik Debug Monitor Server (dále DDMS) je nástroj, který slouží k ladění spuštěné aplikace, buď na emulátoru nebo fyzickém zařízení. DDMS obsahuje např. modul LogCat, který nám umožňuje sledovat veškeré procesy, které probíhají na zařízení v reálném čase. Také můžeme za pomocí DDMS nasimulovat některé situace, které by při běhu aplikace mohli nastat:

- příchozí textovou zprávu.
- příchozí hovor.
- odeslání informací na zařízení o poloze.
- a další.

4.2.4 Draw 9-patch

Draw 9-patch je editor, pomocí kterého je možné vytvářet bitmapové obrázky. Na nich lze nastavit body, které se budou při změně velikosti roztahovat. Využívá se např. při vytváření tlačítek, kdy lze jedno tlačítko použít s různými rozměry bez deformací.

4.3 Vývojové prostředí

Dalším krokem před vývojem je výběr vývojového prostředí (dále IDE). Tento krok je velice důležitý, protože kvalitní IDE nám může usnadnit práci při vývoji a zvýšit naši produktivitu.

4.3.1 Eclipse

Jako první zmiňuji Eclipse IDE, které se používá zejména pro programovací jazyk Java. Zmiňuji ho právě proto, že bylo po dlouhou dobu jako jediné podporováno společností Google pro vývoj aplikací pro platformu Android.

Pro vývoj pro Android v Eclipse se musí doinstalovat plugin Android Development Tools (dále ADT). Je to plugin, který poskytuje sadu nástrojů, které jsou integrovány s IDE. To nabízí přístup k mnoha funkcím. Dále poskytuje GUI přístup k mnoha z SDK nástrojů příkazového řádku, stejně jako nástroj pro návrh uživatelského rozhraní pro rapid prototyping, projektování a budování uživatelského rozhraní vaší aplikace. Toto IDE má jedinou výhodu vzhledem k Android Studiu a tou je podpora NDK. [4]

4.3.2 NetBeans

NetBeans IDE se stejně jako Eclipse používá zejména pro programovací jazyk Java, ve kterém je i naprogramován. Umožňuje programování i v jiných jazycích jako PHP, C++ a dalších.

Při použití NetBeans pro vývoj Android aplikace se také musí doinstalovat plugin NBAndroid[6]. Tento plugin je neoficiální a není podporován společností Google. Na obrázku 5, můžete vidět Netbeans IDE se spuštěným AVD a nalevo od něj můžete vidět LogCat.

4.3.3 Android Studio

Jako poslední zmiňuji Android Studio IDE, které je od prosince roku 2014 oficiálně podporováno společností Google. Jeho předchůdcem byl již zmiňovaný Eclipse v kombinaci s pluginem ADT , jehož vývoj byl ukončen. Toto prostředí je založené na IntelliJ IDEA[7].

Instalace probíhá na rozdíl od Eclipse a NetBeans velice jednoduše. Stačí si stáhnout a nainstalovat instalační balík. K instalaci je potřeba Java od společnosti Oracle. Jak jste si určitě všimli, nejsou potřeba žádné pluginy. [5]

4.3.4 Výběr IDE

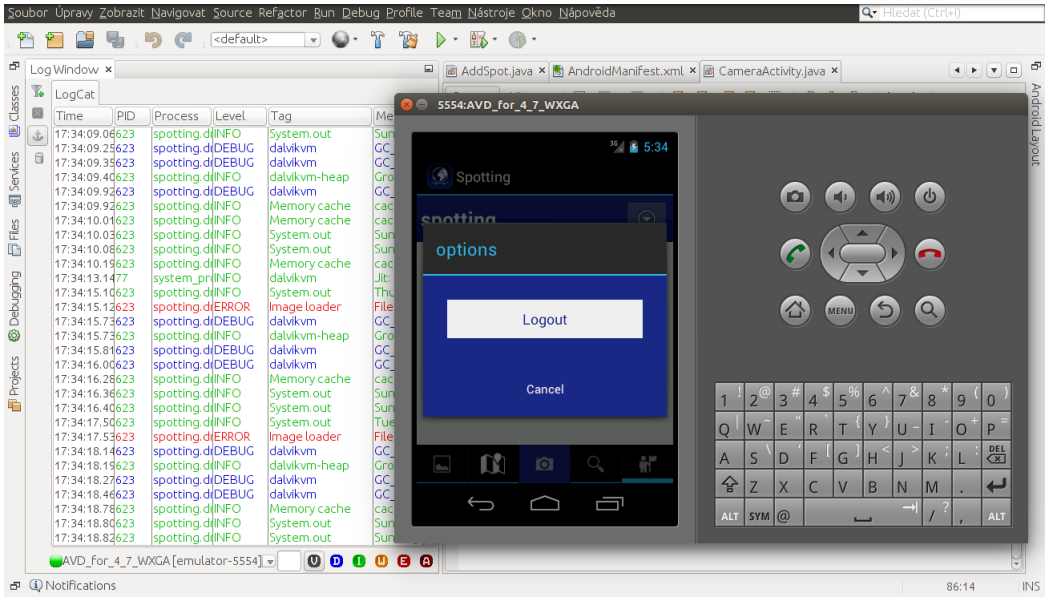
Výběr IDE byl velice složitý. Hlavním z důvodů mého výběru bylo, že v dané době Android Studio bylo ještě ve vývoji a oficiálně byl podporován pouze Eclipse. Takže mně zůstal výběr mezi NetBeans a Eclipse. Po dlouhém rozmýšlení jsem

[6]Open source software distribuovaný pod Apache License 2.0.
[7]IDE pro programování v jazycích Java, Groovy a dalších.

si vybral NetBeans a to z důvodu, že s ním mám dobré a velké zkušenosti na rozdíl od Eclipse. Po dobu vývoje jsem se nesetkal s žádným větším problémem s IDE.

Nyní kdybych si měl zvolit, bylo by to jednoznačně Android Studio a to z důvodu, že se obvykle vyžadují zkušenosti pro přijetí do práce s tímto IDE a práce s ním má jisté výhody:

- oficiální podpora.
- rychlost.
- menší spotřeba paměti.



Obrázek 5: NetBeans IDE.

5 Základy vývoje Android aplikací

Aplikace pro Android sestávají z několika komponent activities (aktivity), services (služby), content providers (poskytovatelé obsahu) a broadcast recievers (přijímače). Komponenty jsou deklarovány v souboru AndroidManifest.xml, který specifikuje celou aplikaci. Na základě tohoto souboru systém ví, co lze jak volat a jak může komponenta spolupracovat s okolím. V této sekci si popíšeme k čemu slouží a jejich životní cyklus, který určuje jak komponenta vzniká a zaniká. Informace jsou převzaty ze zdroje [6].

5.1 Aktivity

V aplikaci reprezentují prezentační vrstvu. Každá aktivita obvykle odpovídá jedné obrazovce aplikace. Slouží tedy jako hlavní prostředek pro interakci s uživatelem, i když je možné, aby neměly uživatelské rozhraní, ale v této práci se s tím nesetkáme. UI lze definovat dvěma způsoby:

1. v XML souboru.
2. za běhu programově.

Tyto dva způsoby lze i kombinovat. Obvykle se definuje výchozí vzhled pomocí XML souboru a potom při uživatelově interakci se může tento vzhled za běhu měnit.

Aplikace většinou obsahují více aktivit, které jsou mezi sebou vázány a jedna z nich je určená jako hlavní. Ta se zobrazí ihned po spuštění aplikace. Aktivita může za pomocí záměrů mezi jednotlivými aktivitami přepínat a předávat data. Vždy, když začne nová aktivita, předchozí aktivita se zastaví a uloží do zásobníku, ale je možné se k ní vrátit za pomocí tlačítka zpět.

5.1.1 Životní cyklus aktivit

OS Android se nachází převážně na mobilních zařízeních, které trpí nedostatkem operační paměti. Je zřejmé, že některé aktivity jsou důležité a některé méně. Proto v důsledku nedostatku operační paměti je možné násilné ukončení aktivity. Hlavním důvodem je obvykle potřeba systému uvolnit paměť jiné aktivitě. V této sekci si popíšeme životní cyklus aktivit tzn. popis stavů, ve kterých se může nacházet nebo událostí od jejího vzniku až po její ukončení.

Každá aktivita prochází vlastním životním cyklem a nachází se v jednom znásledujících čtyř stavů:

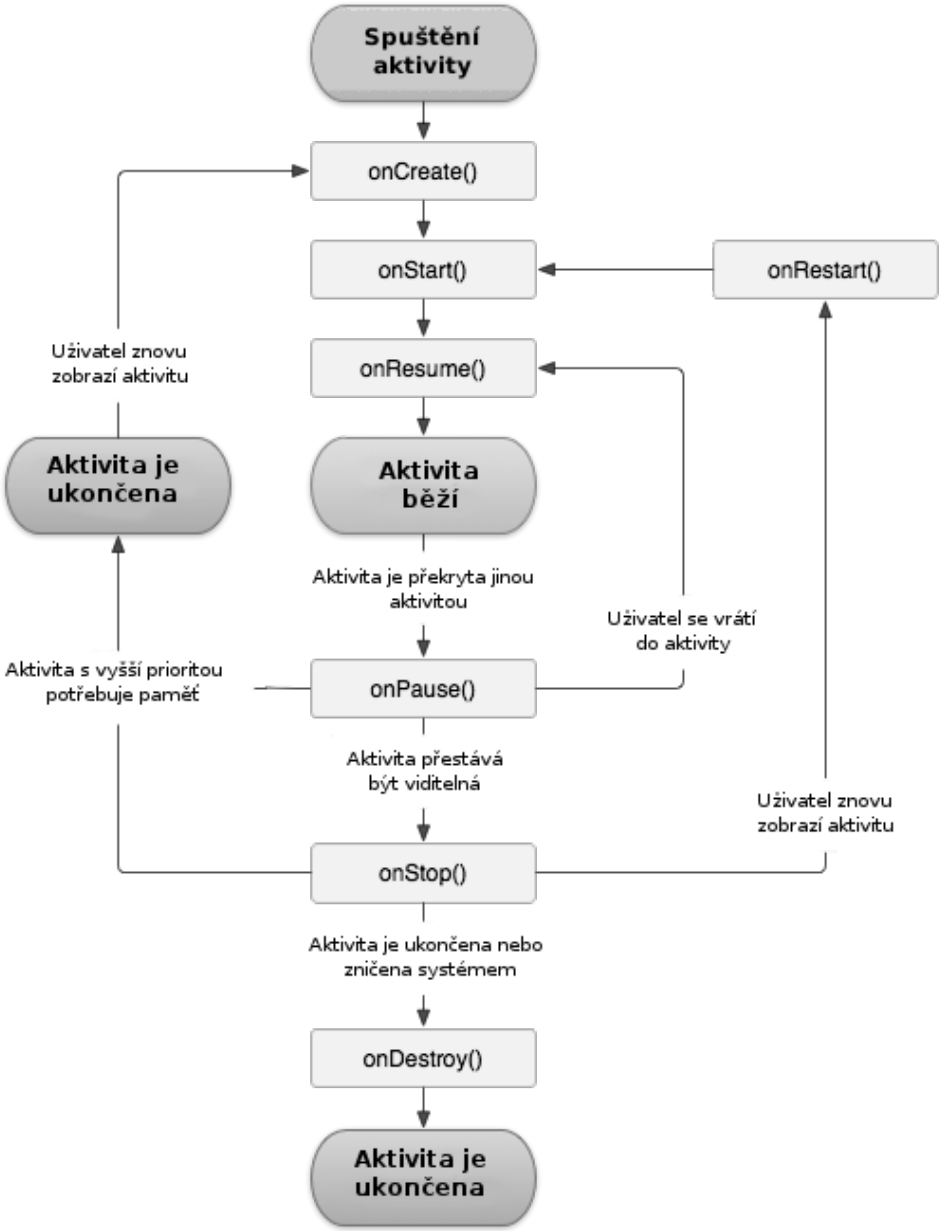
- aktivní - aktivita je spuštěna a běží v popředí.
- pozastavená - aktivita je spuštěna, běží a je viditelná, ale uživatel s ní nemůže interagovat. To bývá způsobeno částečným překrytím aktivity jinou, která nezabírá celou obrazovku nebo je z části průhledná. Tato situace většinou nastane při zobrazení dialogového okna.

- zastavená - aktivita je spuštěna, běží, ale není viditelná, je překryta jinými aktivitami. Nyní může být aktivita ukončena, pokud je potřeba uvolnit paměť.
- mrtvá - aktivita nebyla spuštěna, nebo byla ukončena násilně z důvodu nedostatku dostupné paměti, nebo jí ukončil uživatel. Jestliže má být aktivita znovu spuštěna do popředí, je potřeba jí znovu vytvořit.

5.1.2 Callback metody

Systém při přecházení mezi jednotlivými stavy volá callback metody. Životní cyklus aktivity a přecházení mezi stavy je znázorněno na obr. 6.

- metoda onCreate() se volá při prvním spuštění aktivity. V této metodě se inicializuje UI a provádí se zde veškeré operace, které je potřeba provést pouze jednou a nejsou závislé na dalším způsobu použití aktivity. Metoda se také volá v případě, pokud běžela a změnily se jí prostředky, které aktivita využívá ke své činnosti a je potřeba ji znovu vytvořit. K tomuto dochází např. při změně orientace obrazovky, kdy je nutné znovu vytvořit UI.
- metoda onDestroy() je volána při ukončování aktivity. Tato metoda se hodí pro uvolnění prostředků, které jsme získali při vytváření aktivity. Je volána vždy, když je aktivita ukončena funkcí finish(), ale pokud je ukončena násilně z důvodu nedostatku dostupné paměti nemusí k volání metody dojít.
- metoda onRestart() se volá tehdy, když byla aktivita zastavena a má se znovu spustit.
- metoda onStart() se volá před tím, než se stane aktivita viditelnou.
- metoda onResume() se volá těsně před přesunem aktivity do popředí. Zde je možné si změnit UI na základě nějaké události, která nastala.
- naopak metoda onPause() je volána, pokud se do popředí dostane jiná aktivita. Zde je výhodné zrušit vše, co jste provedli v metodě onResume() např. ukončení všech procesů mimo hlavní vlákno, uvolnění všech prostředků(např. fotoaparát), které si aktivita vyžádala atd.
- metoda onStop() se volá tehdy, když aktivita není viditelná.



Obrázek 6: Životní cyklus aktivit.

5.2 Intenty

Intenty jsou základními komunikačními prvky mezi jednotlivými komponentami aplikace. Je to tedy objekt třídy Intent, který slouží pro spuštění aktivity nebo služby a dokáže přenášet jednoduchá data mezi těmito komponentami. Rozdělujeme je na explicitní a implicitní:

- explicitní - používají se ke spuštění jedné konkrétní aktivity. Systém přesně ví, jak reagovat.

```
1 Intent i = new Intent(context, CameraActivity.class);
```

Zdrojový kód 1: Ukázka explicitního intentu.

- implicitní - používají se v případě, kdy je jasné co má být provedeno, ale není definované jak danou činnost provést. To necháme na systému.

```
1 Uri uri = Uri.parse("http://www.spotting.com");
2 Intent i = new Intent(Intent.ACTION_VIEW, uri);
3 startActivity(i);
```

Zdrojový kód 2: Ukázka implicitního intentu.

Na příkladu můžete vidět otevření webové stránky z aktivity pomocí nějakého internetového prohlížeče.

5.3 Poskytovatelé obsahu

Jak jsem zmiňoval výše, lze přes intenty posílat a přijímat menší množství dat, které se ale nehodí pro větší objemy dat. Z tohoto důvodu jsou v systému poskytovatelé obsahu, kteří jsou určení ke sdílení dat mezi jednotlivými aplikacemi v celém systému. Systém je dodáván s poskytovateli obsahu pro některé základní typy dat a řadu z nich najdeme v balíčku Android.provider. Jestliže chceme sdílet některá data, je to možné dvojím způsobem:

1. a to vytvořením vlastního poskytovatele obsahu za pomocí třídy contentProvider[8].
2. nebo využitím stávajícího poskytovatele obsahu, kdy musí poskytovat stejný typ dat a také musíte mít povolený přístup pro zápis.

5.4 Služby

Aktivity a poskytovatelé obsahu jsou entity s krátkou životností, které lze kdykoliv vypnout. Proto existují služby, které jsou navrženy tak, aby mohli provádět dlouhotrvající operace na pozadí. Služby neposkytují GUI a nejsou proto vázány na grafické rozhraní aplikace. Např. lze tedy přehrávat hudbu, i když aktivita, při které bylo spuštěno přehrávání, už byla ukončena.

[8]Komponenta spravující sdílená aplikační data.

5.5 Přijímače

Komponenta odvozená od třídy android.app.BroadcastReceiver slouží k naslouchání oznámení. Poté umožňuje reagovat aplikaci na nějakou událost, která nemusela vzniknout v rámci aplikace, ale mohla vzniknout systémem nebo jinou aplikací. Příkladem může být oznámení o příchodu SMS zprávy.

6 Návrh aplikace

V této části se budu zabývat návrhem aplikace, která mi poskytne následné informace k samotné implementaci. Zaměřuji se na popis diagramu případu užití, návrhu architektury a popisu použitých knihoven.

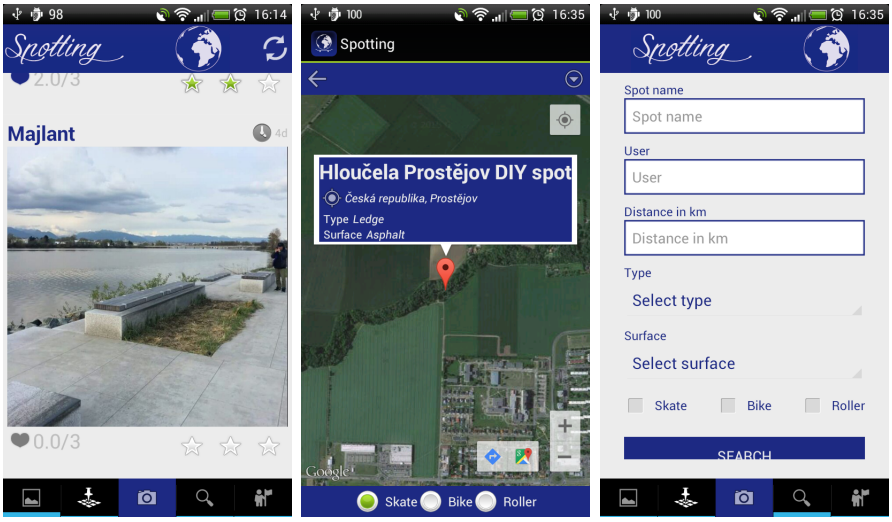
6.1 Diagram případů užití

Diagram případu užití znázorněn na obr. 7 popisuje funkční požadavky na systém. Zobrazuje jednotlivé případy užití, které nám znázorňují interakce mezi

Obrázek 7: Diagram případů užití aplikace.

uživatelem aplikace a aplikací samotnou.

- Přihlásit se - funkce na přihlášení do systému, která se spustí při prvním spuštění, pokud uživatel jíž není přihlášen.
- Zaregistrovat se - funkce pro vytvoření účtu do systému.
- Zobrazit nahrané spoty uživatelů - zobrazení nahraných spotů probíhá automaticky po zapnutí aplikace, jestliže se uživatel již někdy přihlásil do systému.
- Aktualizovat spoty - po stisknutí aktualizujícího tlačítka se obnoví seznam nahraných spotů.
- Zobrazit spot na mapě - vyvolá novou aktivitu pro zobrazení spotu na mapě, kdy se přímo přiblíží pohled na vybraný spot.
- Zobrazit detail spotu - umožňuje zobrazit detailní popis po kliknutí na spot ze seznamu zobrazených.
- Spustit navigaci ke spotu - spuštění této funkce se vyvolá pomocí implicitního intentu, který převezme některá z aplikací pro navigaci.
- Zobrazit spoty na mapě - vyvolá novou aktivitu se zobrazenými skate spoty na mapě. Lze dále přepínat mezi spoty pro cyklisty a bruslaře.
- Přidat spot - tato funkce vyvolá novou aktivitu, kdy se spustí fotoaparát. Zde lze pořídit snímek dvojím způsobem a to vyfotografováním, nebo výběrem z galerie. Po pořízení snímku ve čtvercovém formátu se spustí nová aktivita se zobrazeným snímkem. Jestliže se nám pořízený snímek nepovedl můžeme se vrátit zpět pro znovu vytvoření, nebo pokračovat a spustit novou aktivitu pro otagování a nahrání vytvořeného spotu na server.
- Vyhledat spoty - tato funkce slouží pro vyhledávání, kdy zadáním určitých parametrů ji upřesníme např. zvolíme vzdálenost do 5km a zaškrtneme políčko skate, tím se nám vyvolá nová aktivita se seznamem skate spotů do dané vzdálenosti.
- Zobrazit mnou nahrané spoty - zobrazím v seznamu všechny uživatelem nahrané spoty.
- Odebrat spot - umožňuje po kliknutí na spot ze seznamu uživatelem nahraných jeho odstranění.
- Upravit spot - umožňuje po kliknutí na spot ze seznamu uživatelem nahraných změnit některé údaje.
- Odhlásit se - funkce na odhlášení ze systému.



Obrázek 8: Obrazovka se seznamem spotů (vlevo), mapy se spoty (uprostřed) a pro vyhledávání (vpravo).

6.2 Architektura aplikace

Součástí systému je serverová část a klientská android aplikace komunikující s tímto serverem. Komunikace mezi android klientem a serverem je skrz REST[9] služby. Jako datové úložiště je zvoleno MySQL.

Využívám tedy třívrstvé architektury, kdy je aplikace rozdělena na to co uživatel vidí a na to, co se odehrává na pozadí na straně serveru.

Vrstvy architektury:

- Prezentační vrstva - klientská mobilní aplikace pro platformu Android.
- Aplikační vrstva - prostředník mezi prezentační a datovou vrstvou. Je tvořena PHP skripty společně s SQL příkazy, které běží na webovém serveru. Zvolil jsem free hostingový server Endora.cz, který splňoval nejlépe mé požadavky např. bezplatnou registraci domény III. řádu, bezplatný hosting a podporu MySQL databází.
- Datová vrstva - její úlohou je ukládání dat do perzistentního úložiště. V mém systému, jak jsem již zmiňoval byla zvolena relační databáze MySQL.

Výhodou zvolené architektury je rozdělení výkonu mezi zařízení uživatele a serveru. Díky tomuto může aplikace běžet i v low-end zařízeních. Dále umožňuje nezávislé správy jednotlivých vrstev.

[9]Representational State Transfer - architektura webových aplikací, která nám umožňuje přistupovat k datům.

6.3 Použité knihovny

Při vývoji byla použita i řada externích knihoven, které si v této kapitole rozebereme.

6.3.1 Volley

Je knihovna vyvíjená společností Google, která slouží pro snadnější a hlavně rychlejší používání síťových požadavků. Mezi její výhody patří automatické plánování síťových požadavků, umožnění více souběžných připojení k síti, poskytnutí paměťového cachování a také umožňuje zrušit pouze jednu nebo všechny žádosti.

Její nevýhodou je, že nedokáže stahovat větší soubory a to z důvodu, že udržuje všechny odpovědi v paměti.[7]

6.3.2 Google Play Services

Google Play Services je soubor služeb, jejichž hlavním účelem je ulehčení práce vývojářům. V této aplikaci využívám služby pro získání polohy od nejlepšího poskytovatele a přístup k Google Maps.

Jestliže knihovna Google Play Services není nainstalována na zařízení, tak služby na daném zařízení nebudou fungovat.[8]

Další použité knihovny (httpclient-4.3.6, httpcore-4.3.3, httpmime-4.3.6) využívané při odesílání dat s ukazatelem průběhu na server viz. kapitola 7.1.8.

7 Implementace

V této části se budu zabývat programátorskými aspekty práce. Je zde popsána většina vytvořených tříd. Dále budu dělit třídy podle jejich typu.

7.1 Activity

O aktivitách jsem se již zmiňoval v kapitole 5.1, nyní si popíšeme jednotlivé aktivity použité v aplikaci.

7.1.1 Login

Přihlašovací obrazovka, pomocí které se lze přihlásit do systému. Aktivita `Login` obsahuje dvě tlačítka (pro přihlášení a pro přepnutí do aktivity `Registration` pro založení účtu) a formulář (pro vyplnění údajů uživatele). K přihlášení je potřeba internetového připojení, z důvodu uložení dat o uživateli v MySQL databázi na straně serveru. O výměnu dat mezi klientskou aplikací a databází se stará PHP skript `logAndReg.php`. Jejím úkolem je dostat požadavek od klienta, provést dotazy na databázi a odeslat odpověď zpět klientovi.

K síťovým požadavkům využívám knihovnu volley viz. 6.3.1, která mi stahování dat ze serveru velice usnadňuje. Jinak bych musel např. stahovat data ve vlastním vlákně přes `AsyncTask`. Po přijetí požadavku se vytvořené API připojí na databázi přes `newConnection.php` skript a zkontroluje, zda se předané údaje shodují s uživatelem v databázi a odešle za pomocí funkce `json_encode` klientovi odpověď zpět ve formátu JSON.

Ukázka odpovědi s chybovou zprávou

```
{
    "error":true,
    "error_msg":"Incorrect nick or password!"
}
```

Ukázka odpovědi s úspěšným přihlášením

```
{
    "error":false,
    "users_id":"157",
    "user":{
        "nick":"Majlant",
        "email":"jir.mil@centrum.cz"
    }
}
```

Po úspěšném přihlášení se nastaví za pomocí `Shared preferences`[10], že je uživatel již přihlášen. Práci s ní má na starosti třída `SPManager`. Dále se uloží data o uživateli do lokální databáze SQLite a spustí se aktivita `Spotting`.

[10] Úložiště, kam lze ukládat nějaká data.

Po znovu spuštění aplikace se již nemusíme přihlašovat. Funkce `checkLogged` provede kontrolu stavu přihlášení i v tzv. offline režimu.

7.1.2 Registration

Registrační obrazovka, pomocí které lze založit účet do systému. Obsahuje dvě tlačítka (pro zaregistrování a přepnutí do aktivity `Login`) a formulář (pro vyplnění údajů uživatele) skládající se ze čtyř `EditTextů`.

Registrace probíhá obdobně jako již bylo zmíněno u aktivity `Login`. Před odesláním požadavku na server se ale provede na straně klienta základní validace formuláře. Provede se kontrola vyplnění všech polí ve formuláři, validace mailu, kterou provádí funkce `isValidEmail`, shody zadaných hesel a minimální délky hesla, která musí přesahovat délku pěti znaků.

7.1.3 Spotting

`Spotting` je hlavní aktivita obsahující pět záložek. Je to takový rozcestník funkcí aplikace.

Po prvním spuštění je volána metoda `getSpots`, která požádá server o šest posledně nahraných spotů uživatelů. Odpověď je opět ve formátu JSON viz. ukázka na konci kapitoly 7.1.3. Tu má na starosti `fileDownload.php` skript. Metoda `doResponseSpots` zpracuje odpověď, vytvoří instanci třídy `SpotLoadAdapter` viz. třída 7.2.1 a ta se nastaví jako adaptér pro ListView[11] `spotsInListView`. Poté je zobrazen seznam spotů, které můžeme procházet rolováním.

Další funkcí implementovanou v této třídě je vyhledávání. Komunikace se serverem je prováděna v metodě `searchSpots`. Ta probíhá obdobně jako v předchozím odstavci. Jen se provádí se `searchSpots.php` skriptem. Pro prohledávání textů v MySQL databázi využívám jednoduchého operátoru LIKE. Umožňuje vyhledávat řetězce jen dle části textu. Lze i vyhledávat do určité vzdálenosti. K tomu využívám Haversinův vzorec. Ten dokáže mezi dvěma body na zeměkouli dané zeměpisnou délkou a šířkou vypočítat jejich vzdálenost.[9] Výše popsaná problematika je implementovaná ve funkci `getSpotsByUser`. Ta najde záznamy v tabulce podle vytvořeného dotazu a údajů uživatele.

[11] Rozložení zobrazující rolující seznam položek.

Ukázka odpovědi PHP skriptu fileDownload.php


```
[
    {
        "tag":"download",
        "error":false,
        "spot":{
            "like":"false",
            "my_rate":-1,
            "img_name":"IMG_20150716_073244.jpg",
            "user_name":"Jirka",
            "spot_name":"test",
            "type":"Park",
            "surface":"Asphalt",
            "bike":"true",
            "roller":"false",
            "skate":"false",
            "description":"",
            "latitude":"49.5902812",
            "longitude":"17.2494468",
            "number_rate":"0",
            "date":"2015-07-16 07:33:20",
            "rate":"0"
        },
        "spot_id":"241"
    }
]
```


7.1.4 Map

Aktivita `Map` znázorňující obrazovku s mapou. Pro přidání map do aplikace jsem využil `Google Maps Android API`. Toto API automaticky obsluhuje přístup ke Google Maps serverům, stahování dat, reakce na gesta uživatele a zobrazení samotné mapy.[10] Mapa je zobrazena ve fragmentu třídy `com.google.android.gms.maps.MapFragment`. Další funkcí mimo prosté zobrazení je možnost změny typu mapy na satelitní, normální či s terénem, kterou provádí metoda `myClickHandler`. Dalším stěžejním bodem bylo přidání bodů (spotů) tzv. `markery` do mapy na zvolených místech a vytvoření informačního okna, které se zobrazí po kliknutí na kterýkoli z nich. Přidání se implementovalo jednoduchým voláním `GooleMap.addMarker`. Pro vlastní informační okno byla potřeba vytvořit `InfoWindowAdapter` třídu, přepsat `getInfoContents`, který poskytuje vlastní obsah a zavolat `GooleMap.setInfoWindowAdapter` pro nastavení vlastního `InfoWindowAdapter`.

Zobrazené spoty lze i filtrovat dle druhu sportu. Ke stahování informací o spo-

tech je opět využita knihovna volley komunikující s `downloadAllSpots.php` skriptem.

7.1.5 MapSpot

Aktivita `MapSpot` znázorňující obrazovku s mapou a vyznačeným spotem na ní. Dále se s ní nebudu zabývat. Informace o `MapSpot` jsou velice obdobné jako v kapitole 7.1.4.

7.1.6 CameraActivity

Aktivita `CameraActivity` znázorňující obrazovku vlastního fotoaparátu. Aktivita slouží k získání obrázku (spotu). K tomu slouží dva způsoby:

- Z galerie - po výběru této možnosti se uživateli naskytne možnost vybrat si obrázek již uložený v jeho zařízení. Při implementaci využívám implicitního `Intentu` viz. následující ukázka.

```
1 Intent picPathIntent = new Intent(Intent.ACTION_PICK);
2 picPathIntent.setType(getDataImageType());
3 startActivityForResult(picPathIntent, SELECT_PHOTO);
```

Zdrojový kód 3: Následujícím kódem použitý v aplikaci se vyvolá nabídka pro výběr snímku z galerie.

  Konstanta `Intent.ACTION_PICK` říká, že se jedná o požadavek pro získání adresy dat. Ty jsou reprezentovány pomocí `Uri` třídy `android.net.Uri`. Dále za pomocí `setType` bylo specifikováno, o jaký typ dat se jedná. Ke spuštění byla použita metoda `startActivityForResult` a je tedy zřejmé, že poté co nově spuštěná aktivita dokončí zadaný úkol, musí předat výsledek zpět (`Uri`) rodičovské aktivitě. K tomu bylo ještě zapotřebí do implementovat metodu `onActivityResult`, pro zpracování výsledku. Z důvodu, že v aplikaci jsou využívány obrázky ve čtvercovém formátu, je po získání výsledku volána metoda `performCrop`. V ní se nastaví parametr akce s názvem `"com.android.camera.action.CROP"` a další požadované parametry. Opět v metodě `onActivityResult` se získá výsledek, nyní akcí pro ořezání obrázku.

- Z kamery - tato možnost umožní vyfotografování obrázku. Pro vytvoření vlastního fotoaparátu používám balíček `android.hardware.Camera`, který obsahuje rozhraní pro jednotlivé kamery zařízení se systémem Android. Nyní již lze využívat novější `android.hardware.camera2` API, ale pouze pro úroveň API 21 a výše. Následně popíši některé základní kroky, které byly použity k vytvoření vlastního fo-

toaparátu. Nejdříve je získána instance objektu `Camera` metodou `getCameraInstance()`. Poté byla vytvořena instance vlastní třídy `CameraPreview`, která rozšiřuje `SurfaceView` a implementuje `SurfaceHolder.Callback` rozhraní. Tato třída umožňuje zobrazit náhled obrazu z fotoaparátu a za pomocí zpětných volání zachytit vytvoření a zrušení `View`. Nyní připojíme oblast pro vykreslování k fotoaparátu metodou `Camera.setPreviewDisplay()` s parametrem instance třídy `SurfaceHolder`. Nakonec byla potřeba zahájit zobrazování snímků z fotoaparátu metodou `Camera.startPreview()`. Před pořizování snímků, bylo nutností nastavení jejich kvality. To se provádí přes objekt `Camera` v metodě `setBestQualityCamera()`. Teď nám již nic nebrání v jejich pořízení. Načtení obrázků provádím pomocí metody `Camera.takePicture()`. V aplikaci využívám snímky ve formátu JPEG, z toho důvodu implementuji rozhraní `Camera.PictureCallback` pro příjem obrazových dat. Posledním stěžejním problémem je ořezání obrázku do čtvercového formátu a uvolnění kamery. Při vývoji první problematiky jsem se potýkal s problémem, kdy byla vyvolávána výjimka `java.lang.OutOfMemoryError`. Důvodem bylo, že zařízení se systémem Android mají omezené množství paměti pro každou aplikaci. Řešením problému bylo přidat do souboru manifest `android:largeHeap="true"` a tím byla navýšena přidělená paměť aplikace, která je pro práci s obrázky téměř nutností. Nesmíme opomenout na uvolnění kamery, z důvodu, že je to zdroj, který je sdílený mezi aplikacemi. Uvolnění provádím metodou `stopPreviewAndFreeCamera()` implementovanou v třídě `CameraPreview`.[11]

7.1.7 ShowPhoto

`ShowPhoto` je aktivita znázorňující obrazovku s obrázkem pořízený v `CameraActivity`. Je to takový prostředník mezi dvě mi aktivitami. Samotný obrázek je uložen na zařízení v externí paměti. Pro jeho získání je mezi aktivitami předávána cesta k souboru, která je specifikována absolutní cestou.

7.1.8 AddSpot

`AddSpot` slouží pro otagování a odesílání spotů na server. V této sekci se zabývám implementací získání geografických souřadnic (zeměpisná šířka/zeměpisná délka) a odesílání dat na server.

- Určení polohy - pro určení polohy lze využít dvě funkce, kdy souřadnice získáme pomocí:
  - Zadáním adresy: pro získání zeměpisné šířky a délky z adresy využívám třídu `Geocoder` implementovanou v třídě `LocationFromAddressDialog` rozšiřující `AlertDialog.Builder`. Tento proces je nazýván geokódování.[12]

  - Poskytovatele polohy GPS a síťového poskytovatele: Android doporučuje využívat jejich nejnovější API zvané `Google Play services location API`, které jsem se rozhodl i využívat. To nabízí snadnější a přesnější určování polohy. Říkají, že by měla být i snížená spotřeba baterie. Tyto služby jsou poskytované prostřednictvím Google Play služeb. API vyžaduje nejdříve integrovat Google Play služby pomocí `GoogleApiClient`. To provádím metodou `buildGoogleApiClient()` volanou v `onCreate()`. Poté je připojena metodou `connect()` v `onStart()`. Jenž je `GoogleApiClient` úspěšně připojen, tak v `onConnected` zaregistrujeme `LocationListener` pro aktualizaci polohy metodou `startLocationUpdates()`. Odpojení je provedeno metodou `disconnect()` v `onStop()`.[13]

- Odesílání dat na server - v této části se zabývám odesíláním binárních dat s ukazatelem průběhu na server. Ke komunikaci se server nevyužívám knihovnu volley jako doposud. Nehodí se pro odesílání větších souborů a nepodporuje ani ukazatel průběhu (progress bar). Pro odesílání využívám metody POST protokolu HTTP[12] z balíčku `org.apache.http`. HTTP komunikace nelze provádět v UI vlákně aplikace. Pro tyto účely jsem rozšířil třídu `Upload`, ve které je implementována veškerá logika komunikace o třídu `AsyncTask`. Nyní již je veškerá komunikace prováděna na pozadí a nezasahuje do běhu UI vlákna. Samotná třída `AsyncTask` implementuje několik metod `onPreExecute`, `doInBackground`, `onPostExecute` a `onProgressUpdate`. Odesílání se spustí v hlavním vlákně pomocí metody `execute` a zavolá se metoda `onPreExecute`, ve které se zobrazí ukazatel průběhu. Po zobrazení ukazatele průběhu se vykoná komunikace se serverem implementována v metodě `uploadFile`, která je spuštěná v `doInBackground`. Po dokončení této metody se zavolá `onPostExecute`. V ní se zruší ukazatel a spustí se aktivita se seznamem spotů.

7.1.9 Search

Aktivita `Search` znázorňující obrazovku se seznamem spotů (`ListView`) s nastaveným adaptérem `SpotLoadAdapter` viz. kapitola 7.2.1.

7.2 Adaptéry

Třída adaptér se využívá jako most mezi `AdapterView` (v této práci využívám `ListView`) a daty, která dané `View` zobrazují. Stará se o vytváření jednotlivých položek `View` v seznamu, dle vlastního rozložení prvků. [14]

[12]Protokol určený pro přenos HTML dokumentů mezi klientem a serverem.

7.2.1 SpotLoadAdapter

Pro vytvoření `ListView` se seznamem spotů, je vždy potřeba vytvořit instanci třídy `SpotLoadAdapter` s parametry `Activity` a seznamem spotů (`List<SpotModel>`). Ta se stará o předání dat a jejich zobrazení v `ListView`. Jedna položka v seznamu se skládá ze tří `TextView` a jednoho `ImageView`, `ImageButtonu` a `RatingBaru`. Fotky zobrazované v `ImageView` se stahují ze serveru. Po jejich přidání výkon prohlížení výrazně poklesl, aplikace byla téměř nepoužitelná. Řešení jsem našel v tomto tutoriálu [15]. Problém byl v tom, že se v paměti udržoval pouze zobrazovaný a při srolování na jiný se musel nejdříve stáhnout. To je velice neefektivní a proto byla vytvořena vyrovnávací paměť[13] na SD kartě v adresáři `/mnt/sdcard/spottingCache`. Vše je implementované v následujících třídách:

- `SpotImageLoader` - slouží pro stahování obrázků ve vlastním vlákně z URL[14] adresy, nebo mohou být načteny z vyrovnávací paměti, pokud již byly staženy dříve.
- `FileCache` - slouží k vytvoření adresáře na SD kartě, získání souboru z adresáře pomocí URL obrázku a také můžeme použít metodu `clear` pro odstranění všech souborů z adresáře.
- `MemoryCache` - slouží k nastavení limitu velikosti složky vyrovnávací paměti a také k jejímu vymazání.
- `Utils` - slouží k překopírování staženého obrázku do paměti metodou `copyStream`.

7.2.2 DetailsListViewAdapter

Jednoduchý adaptér, které slouží k vytvoření a nastavení dvou `TextView`. Adaptér se nastavuje pro `ListView` implementovaný v třídě `SpotDetailsDialog` viz. kapitola 7.3.2.

7.3 Dialogy

Dialog je malé vyskakovací okno, které se nejčastěji využívá ke sdělení nějaké informace uživateli, k nějakému rozhodnutí (souhlas/odmítnutí), či aby vyčkal na dokončení prováděné úlohy. Android nabízí celou řadu předdefinovaných jednoduchých dialogů. V aplikaci využívám např.

- `ProgressDialog`, který dává uživateli najevo, že aplikace pracuje.
- `AlertDialog`, který je jeden z nejpoužívanějších dialogů. Umožňuje zobrazit titulek, až tři tlačítka a nějaký obsah, který může mít vlastní rozvržení.

[13] Paměť sloužící k dočasnému uschování dat.
[14] Adresa určující umístění dokumentu na internetu.

V aplikaci byly potřeba i složitější dialogy. K tomu jsem si vytvořil vlastní dialogy rozšiřující třídu `Dialog` s definováním vlastního rozvržení. Jedná se o dialog pro změnu hesla, úpravu údajů o spotu, zobrazení informací a další. Následuje popis některých dialogů.

7.3.1 ChangePassDialog

`ChangePassDialog` je vlastní dialog využívající se ke změně přihlášeného uživatele do aplikace. Zobrazuje tři `EditTexty` (pro zadání starého a nového hesla) a dvě tlačítka (pro potvrzení požadavku nebo zavření dialogu). V samotné třídě je implementována komunikace se serverem knihovnou volley v metodě `uploadChangePass`.

7.3.2 SpotDetailsDialog

`SpotDetailsDialog` je dialog zobrazující seznam (`ListView`) nastavený na instanci `DetailsListViewAdapter` adaptéru. Ten je vytvořen s parametry `Context` a seznamem `ArrayList`, ve kterém jsou obsažená data pro vypsání. Ty jsou vytvářeny v metodě `createModels`. Samotný dialog tedy zobrazuje veškeré informace o spotu.

7.3.3 DialogSpot

`DialogSpot` se zobrazí po kliknutí na kterýkoli nahraný spot ze seznamu spotů uživatelů. Obsahuje čtyři tlačítka:

- pro zobrazení spotu na mapě - tato funkce je implementována v metodě `setMapEvent`.
- pro zobrazení informací o spotu - tato funkce je implementována v metodě `setDetailsEvent`.
- pro spuštění navigace od uživatelovi polohy ke spotu - tato funkce je implementována v metodě `setNavigationEvent`.
- pro zavření dialogu - tato funkce je implementována v metodě `setCancelEvent`.

7.3.4 DialogEdit

`DialogEdit` se zobrazí po kliknutí na některý spot ze seznamu uživatele nahraných. Slouží k úpravě či změně údajů k danému spotu. Všechna políčka pro úpravu jsou nastavená dle předchozího zadání údajů v konstruktoru voláním metod, jako jsou např. `setNickView` pro jméno, `setTypeView` pro typ, `setSurfaceView()` pro povrch a další. Komunikace se serverem je zprostředkovávaná přes `editSpot.php` skript a je implementovaná v metodě `editUserSpot` knihovnou volley.

7.4 Ostatní třídy

V této kapitole se budu zabývat třídami, které nelze dělit dle typu.

7.4.1 SpotModel

`SpotModel` je třída implementující rozhraní `Parcelable`. Byla vytvořena jako datový nosič. Obsahuje proměnné pro uchování informací o spotu. Využívá se tedy pro sdílení dat mezi aktivitami, kdy nám nestačí primitivní datové typy. Data je možné poslat spouštěné aktivitě pomocí záměru.

```
1 Intent i = new Intent(context, Search.class);
2 i.putParcelableArrayListExtra(Search.SPOTS_LIST, searchSpots);
3 startActivity(i);
```

Zdrojový kód 4: Následujícím kódem použitý v aplikaci se spustí aktivita `Search` se seznamem objektů `SpotModel`.

```
1 Intent i = this.getIntent();
2 spots = i.getParcelableArrayListExtra(SPOTS_LIST);
```

Zdrojový kód 5: Tímto kódem naopak předaný seznam získáme zpět.

7.4.2 Config

Třída obsahující řetězcové konstanty URL adres.

7.4.3 DetailsModel

Třída, která byla vytvořena pro zapouzdření proměnných a metod do jednoho objektu.

7.4.4 AppController

`AppController` rozšiřující třídu `Application`. Je to třída, která potřebuje udržet globální stav. Využívá se při práci s knihovnou volley, kdy udržuje základní objekty a frontu požadavků. [16]

Závěr

Cílem práce bylo vytvořit aplikaci pro mobilní platformu Android, která umožní uživatelům vyhledávání míst pro skateboardery, bikery a bruslaře a následné zveřejnění pomocí služby Google Play.

Jelikož s programováním pro tuto platformu jsem neměl žádné zkušenosti, tak to bylo pro mě velkým přínosem. Za hlavní přínos při tvorbě této práce pokládám nastudování a vyzkoušení nových technologií. Vyzkoušel jsem si některé nástroje z balíčku SDK, seznámil jsem se s vývojem Android aplikací, z jakých sestávají komponent či jejich životního cyklu. V neposlední řadě práci s některými poskytovanými funkcemi jako je např. Google Maps Android API pro práci s mapami, Google Play services location API pro určení polohy a další.

Mezi další náměty pro možné vylepšení aplikace je využití Facebook SDK. To umožní snadnější přihlášení do aplikace přes uživatelův Facebook účet a využití funkce sdílení na sociální síť Facebook. Dále by bylo vhodné obohatit aplikaci o možnost okomentování spotů, přidání dalších prvků pro umožnění snadnějšího vyhledávání na základě pozdějších požadavků uživatelů.

Conclusions

The aim of the work conducted was to create an application for Android mobile platform, which allows users to search places for skateboarders, bikers and roller-skaters and its subsequent publication on Google Play.

I found this work very beneficial, as I had previously no experience in programming for this platform. The main benefit I acquired while working on this creation was studying and testing new technologies. I experienced work with some of the SDK package tools, I acquainted myself with Android application development process, I learnt what components they are made of and what their life cycle is. Least but not last I worked with some of its functions, such as Google Maps Android API for work with maps, Google Play services location API for determining location, and others.

One of the suggestions to improve the application is to utilize Facebook SDK. This will allow easier access via user's Facebook account and the utilization of sharing function on the Facebook social network. It would also seems suitable to upgrade the application to allow adding comments, and also to put in place additional components to facilitate simpler searching on the basis of users' further requirements.

Bibliografie

[1] Google Buys Android for Its Mobile Arsenal. In: *Businessweek - Business News, Stock market & Financial Advice* [Online]. 16. 8. 2005 [Citace 2. 4. 2015]. Dostupné z: ⟨*http : //www.bloomberg.com/bw/stories/2005 − 08 − 16/google − buys − android − for − its − mobile − arsenal*⟩

[2] Android (operating system). In: *Wikipedia, the free encyclopedia* [Online]. 29. 6. 2012 [Citace 2. 4. 2015]. Dostupné z: ⟨*http : //en.wikipedia.org/wiki/File : Android − System − Architecture.svg*⟩

[3] Google Inc. Android Platform Version. In: *Android Developers* [Online]. 2. 3. 2015 [Citace 3. 4. 2015]. Dostupné z: ⟨*http : //developer.android.com/about/dashboards/index.html*⟩

[4] Google Inc. Eclipse with ADT In: *Android Developers* [Online]. 2015 [Citace: 13. 5. 2015]. Dostupné z: ⟨*http : //developer.android.com/tools/help/adt.html*⟩

[5] Google Inc. Android Studio In: *Android Developers* [Online]. 2015 [Citace: 13. 5. 2015]. Dostupné z: ⟨*http : //developer.android.com/tools/studio/index.html*⟩

[6] GRANT Allen *Android 4* Brno: Computer Press, 2013. [Citace: 13. 5. 2015]

[7] Google Inc. Volley library In: *Android Developers* [Online]. 2015 [Citace: 14. 7. 2015]. Dostupné z: ⟨*https : //developer.android.com/training/volley/index.html*⟩

[8] Google Inc. Google Play Services In: *Android Developers* [Online]. 2015 [Citace: 14. 7. 2015]. Dostupné z: ⟨*https : //developers.google.com/android/guides/overview*⟩

[9] Haversine formula In: *Wikipedia* [Online]. 1. 7. 2015 [Citace: 24. 7. 2015]. Dostupné z: ⟨*https : //en.wikipedia.org/wiki/*⟩

[10] Google Inc. Google Maps Android API v2 In: *Android Developers* [Online]. 14. 7. 2015 [Citace: 19. 7. 2015]. Dostupné z: ⟨*https : //developers.google.com/maps/documentation/android/intro*⟩

[11] Google Inc. Camera In: *Android Developers* [Online]. 2015 [Citace: 20. 7. 2015]. Dostupné z: ⟨*http : //developer.android.com/guide/topics/media/camera.html*⟩

[12] Google Inc. Geocoder In: *Android Developers* [Online]. 17. 7. 2015 [Citace: 21. 7. 2015]. Dostupné z: ⟨*http : //developer.android.com/reference/android/location/Geocoder.html*⟩

[13] Google Inc. Google Play services location API In: *Android Developers* [Online]. 2015 [Citace: 21. 7. 2015].
Dostupné z: ⟨*http : //developer.android.com/training/location/index.html*⟩

[14] Google Inc. Adapter In: *Android Developers* [Online]. 2015 [Citace: 17. 5. 2015].
Dostupné z: ⟨*http : //developer.android.com/reference/android/widget/Adapter.html*⟩

[15] Cache In: *ANDROIDHIVE* [Online]. 10. 7. 2012 [Citace: 23. 7. 2015].
Dostupné z: ⟨*http : //www.androidhive.info/2012/07/android − loading − image − from − url − http/*⟩

[16] Google Inc. Volley-Singleton In: *Android Developers* [Online]. 2015 [Citace: 24. 7. 2015]. Dostupné z: ⟨*https : //developer.android.com/intl/zh − cn/training/volley/requestqueue.html*⟩

A Obsah přiloženého CD/DVD

doc/
: V tomto adresáři se nachází text bakalářské práce ve formátu PDF nesoucí název jiricek.pdf. Dále všechny soubory potřebné k jejímu vygenerování.

dist/
: Obsahuje instalační balíček Spotting.apk pro platformu Android. K instalaci je potřeba umístit tento soubor do paměti zařízení a pomocí nějakého správce souborů jej spustit. Tím se zahájí proces instalace Android aplikace.

src/
: Tento adresář je rozdělen na dva podadresáře a to client, kde jsou umístěny zdrojové kódy klientské aplikace a další soubory pro vytvoření spustitelné verze programu. Dále je zde podadresář server, kde jsou zdrojové kódy php skriptů.

readme.txt
: Obsahuje informace pro spuštění aplikace.

U veškerých cizích převzatých materiálů obsažených na CD/DVD jejich zahrnutí dovolují podmínky pro jejich šíření nebo přiložený souhlas držitele copyrightu. Pro všechny použité (a citované) materiály, u kterých toto není splněno a nejsou tak obsaženy na CD/DVD, je uveden jejich zdroj (např. webová adresa) v bibliografii nebo textu práce nebo v souboru `readme.txt`.