VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

SBĚR STATISTIK O POHYBU UŽIVATELŮ

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE JAN HANÁK
AUTHOR

BRNO 2015

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

SBĚR STATISTIK O POHYBU UŽIVATELŮ

GATHERING STATISTICAL DATA ABOUT USER LOCATIONS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE JAN HANÁK
AUTHOR

VEDOUCÍ PRÁCE Ing. RADEK BURGET, Ph.D.
SUPERVISOR

BRNO 2015

Abstrakt

Cílem této bakalářské práce je popsat tvorbu mobilní aplikace pro platformu Android pracující s geolokací a vytvořit funkční webovou serverovou aplikaci umožnující pomocí webového rozhraní vytvářet data pro klientskou aplikaci a k tomu sbírat a následně zobrazovat statistická data o pohybu uživatelů klientské aplikace. Serverová aplikace je založená na aplikačním rámci Nette (PHP), a klientská aplikace na Android development kit (Java).

Abstract

Goal of this bachelors thesis is to describe development of mobile application for the Android platform working with geolocating and to create functional web server application, which allows using web interface to create data for client application and also collect and display statistical data about movement of client application users. Server application is based on framework Nette (PHP) and client application on Android development kit (Java).

Klíčová slova

Android, aplikace, geolokace, zpracování polohy, Java, PHP, Nette

Keywords

Android, aplication, geolocation, location gathering, Java, PHP, Nette

Citace

Jan Hanák: Sběr statistik o pohybu uživatelů, bakalářská práce, Brno, FIT VUT v Brně, 2015

Sběr statistik o pohybu uživatelů

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Radka Burgeta, Ph.D.

. . . . . .
Jan Hanák
19. května 2015

Poděkování

Tímto bych rád poděkoval vedoucímu bakalářské práce Ing. Radkovi Burgetovi, Ph.D. za cenné rady, vedení práce a poskynutí možnosti vypracovat si bakalářskou práci na toto zadání. Dále bych rád poděkoval testerům této aplikace - Pavel Mícek, Jakub Rambousek, Zdeňka Štumfolová. Nakonec děkuji mé rodině, která mi byla v průběhu mého studia oporou.

Obsah

Seznam obrázků

Kapitola 1

Úvod

Téměř každý nový chytrý telefon běží na operačním systému iOS, Windows Phone nebo Android. Tyto systémy proměňují přístroj původně určený k jediné činnosti – k telefonování, na multifunkční zařízení, jehož potenciál ještě nebyl plně využit. Jednou z předností těchto zařízení s operačním systémem Android o proti stolním počítačům je, že mohou aktivně pracovat s měnící se lokací, na což bude zaměření této bakalářské práce.

Využití konstantně kontrolované lokace v rámci aplikací má obrovský potenciál při tvorbě nových aplikací, ale i při cílení reklamy podle lokace. Aplikace na platformě Android umožnující zaslat uživatelovi zprávu v případě, že projde kolem konkrétní lokace s telefonem u sebe nabízí do budoucna zajímavé obchodní využití. Aplikace, která takto pracuje s lokací, Trip Tip a pro ni vytvořený webový server s administračním rozhraním, je náplní této bakalářské práce.

V psané části si prvně rozebereme použité technologie jako je operační systém Android a příprava prostředí pro tvorbu aplikací pro Android, dále Nette framework použitý pro tvorbu webového serveru. V následující kapitole je rozebrán návrh systému – webového serveru a klientské mobilní aplikace. Kapitola 4 obsahuje informace o implementaci webového serveru, o databázi a uspořádání dat v databázi a hlavně o komunikačním rozhraní poskytující klientské aplikaci. Kapitola 5, stěžejní kapitola bakalářské práce, obsahující informace o implementaci klientské aplikace na operačním systému Android, popisuje části jednotlivé části aplikace jako je ukládání a skladování dat, komunikace se serverem, snímání lokací a uživatelské rozhraní aplikace tvořené aktivitami. V předposlední kapitole – testování systému z pohledu serveru, klienta a jejich komunikace. Na závěr je shrnuta funkčnost systému a možnost rozšíření tohoto systému do budoucna.

Kapitola 2

Rozbor použitých technologií

2.1 Android OS

Android je operační systém založený na jádru Linuxu. Je určený původně pro zařízení s dotykovými displeji jako jsou „chytré“ telefony a tablety. S jeho rozšířením a penetrací trhu byly vytvořeny pomocí upraveného rozhraní i verze pro Televize(Android TV), Hodinky(Android Wear) a Auta(Android Auto).

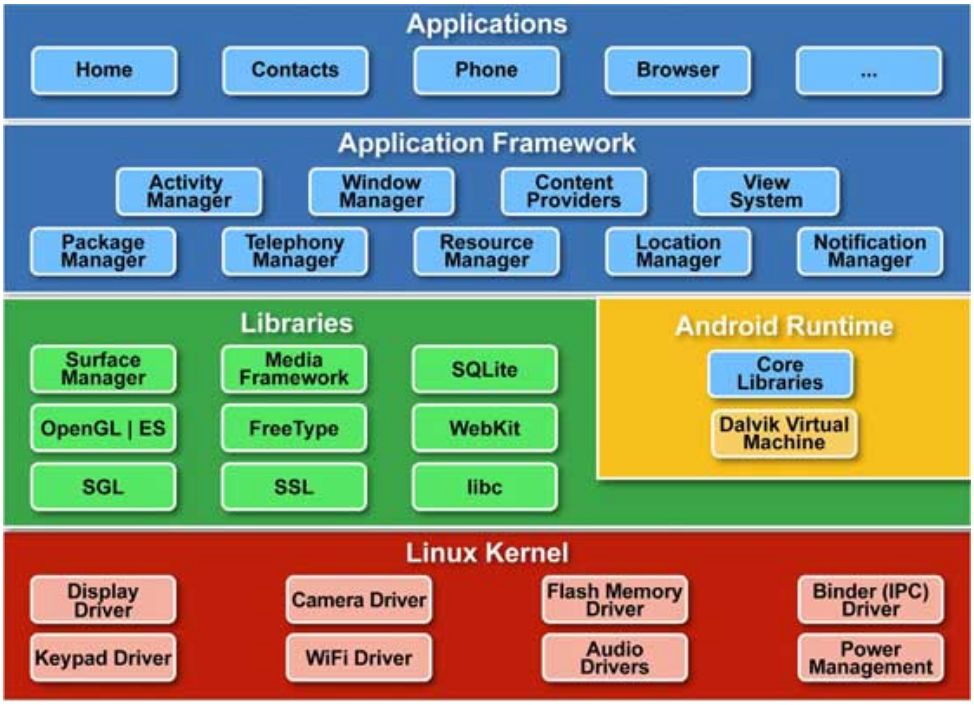
Android je vyvíjen skupinou Open Handset Alliance, vedenou společností Google.[4] Zdrojový kód Androidu je dostupný zdarma a pod licencí otevřeného softwaru. Google zveřejnil většinu kódu pod Apache licencí verze 2.0 a změněné jádro Linuxu je pod licencí GNU verze 2[1].

Nejnovější verze Android 5.0 s kódovým označením Lollipop je dostupný ke stažení od 12. listopadu 2014. Tento update vylepšuje uživatelské rozhraní, stabilitu systému, zvyšuje výdrž baterie a přináší nové funkce.

2.1.1 Architektura

Operační systém android je skupina programových komponent, které jsou rozděleny do pěti sekcí a čtyř vrstev viz obrázek 2.1. Linuxové jádro přímo pracuje s hardwarem, a je nejnižší vrstvou. S aplikacemi naopak přímo pracuje uživatel, který k žádné jiné vrstvě nemá přímo přístup.

[1] General Public License – všeobecná veřejná licence GNU[5]



Obrázek 2.1: Android architektura před verzí 5.0[3]

Linuxové jádro – Linux kernel

První a nejnižší vrstvou je Linux ve verzi 2.6, poskytující základní systémové funkce jako manažer procesů, paměti, zařízení(kamera, klávesnice, displej a jiné). Jádro také zajištuje funkce, ve kterých je Linux velmi kvalitní, jako je práce se sítí a s ovladači zařízení.

Knihovny – Libraries

V druhé vrstvě jsou kolekce knihoven včetně jádra webového prohlížeče WebKit, knihovna libc, knihovny pro práci se SQLite databází, která se používá ke skladování a sdílení dat v rámci aplikací. Dále SSL knihovny pro bezpečnost na internetu a knihovny na přehrávání audia a videa.

Android Runtime

Android Runtime je prostředí používané pro běh aplikací v operačním systému. Android Runtime nahradil Dalvik Virtual Machine, která byla v systému Android až do verze 5.0 (od verze Android 4.4 byl Android Runtime zahrnut jako technologické preview), kde ji Android Runtime zcela nahradil. Android Runtime i Dalvik Virtual Machine jsou určeny k převodu aplikačního byte-kódu do nativních instrukcí, které jsou později provedeny zařízením.

Android Runtime je třetí sekcí architektury a dostupná na druhé vrstvě. Spolupracuje s Linuxovým jádrem na managementu paměti a použití více vláken, jenž je přirozená vlastnost jazyku Java. Android Runtime umožňuje každé Android aplikaci běžet ve svém vlastním procesu a se svojí vlastní instancí Android Runtime.

Android Runtime také poskytuje sbírku nezbytných knihoven, které umožňují vývojářům aplikací psát pomocí standardní Javy.

Aplikační Framework – Application Framework

Vrstva aplikačního frameworku poskytuje služby aplikacím formou Java tříd. Vývojáři aplikací mohou využít těchto služeb v jejich aplikacích. Například manažer poloh pro získávání geolokace, notifikační manažer, pro zobrazování notifikací nebo i manažer zdrojů, který aplikaci předává přiložené multimediální a xml soubory.

Aplikace – Applications

Všechny aplikace pro Android naleznete v této vrchní vrstvě. Nově nainstalované i systémové aplikace se nachází pouze zde. Jsou to například aplikace Kontakty, Telefon, Webový prohlížeč, Hry a další.[3]

2.1.2 Aplikace

Při programování aplikací pro Android se setkáme s následujícími strukturami:

- Activity – Aktivita: Stavebními bloky uživatelského rozhraní jsou aktivity. Aktivitu si můžeme představit jako entitu systému Android analogickou k oknu nebo dialogu klasické aplikace pro počítače nebo jako stránku klasické webové aplikace. Systém Android je navržen tak, aby podporoval množství nenáročných aktivit, takže může uživatelům umožnit otevírání nových aktivit dotykem v aplikaci a návrat do dříve otevřených aktivit pomocí tlačítka Zpět podobně jako ve webovém prohlížeči.

- Service – Služba: Služby jsou procesy v pozadí aplikace, které jsou uživateli skryty. Jsou přizpůsobené k tomu, aby běžely po celý životní cyklus aplikace,to je opak vůči aktivitám, které jsou entity s krátkým životním cyklem a lze je kdykoliv ukončit. Jsou nezávislé na aktivitách a podobají se službám nebo démonům jiných operačních systémů. Službu lze používat například ke kontrole dostupných aktualizací v informačním kanálu RSS nebo k přehrávání hudby na pozadí, dokonce i když její řídící aktivita již neběží, čehož se dosáhne pomocí přichycení služby na notifikaci. V kapitole o implementaci klientské aplikace se o službách dozvíte více 5.5.

- Content provider – Poskytovatel obsahu: Poskytovatelé obsahu poskytují úroveň abstrakce pro jakákoliv data uložená v zařízení, ke kterým lze přistupovat z více aplikací. Vývojový model aplikací pro Android podporuje zpřístupnění dat aplikace i ostatním aplikacím. Dosahuje se tím právě přes poskytovatele obsahu, který umožňuje zachovat si plnou kontrolu nad přístupem k vašim datům. Poskytovatelem obsahu je například místní SQLite databáze.

- Intent – Záměr: Záměry jsou systémové zprávy, které kolují v zařízení a upozorňují aplikace na různé události, počínaje změnami stavu hardwaru přes příchozí data až po události aplikací. Záměry jsou velmi podobné zprávám a událostem jiných operačních systémů. Na záměr může aplikace nejen reagovat ale vytvořit i svůj vlastní a spustit pomocí něj jiné aktivity.[1]

- Broadcast receiver: Broadcast reciever je komponenta sloužící k zachytávání záměrů. V případě zachycení záměru aktivuje svůj kód, kterým může být například spuštění jiné komponenty. Stejně jako služby, tak ani broadcast reciever nemá uživatelské rozhraní. Příklad použití broadcast recieveru může být reakce na oznámení o nízkém stavu baterie, zapnutí telefonu, doručení SMS zprávy, stažení dat.[6]

Každá aplikace navíc musí mít soubor AndroidManifest.xml v jejím kořenovém adresáři. Manifest poskytuje systému základní informace o aplikaci, které systém musí obdržet před zapnutím každé aplikace. Mezi tyto informace patří například:

- Jméno Java balíčku aplikace. Jméno balíčku slouží jako jedinečný identifikátor aplikace mezi aplikacemi v zařízení i v obchodu Google Play.
- Poskytnuté komponenty aplikace – aktivity, služby, broadcast recievery a poskytovatelé obsahu ze kterých se aplikace skládá.
- Deklarace povolení, která aplikace musí mít, aby mohla plnit svou činnost. Například přístup k internetu, přístup k lokacím. Jsou zde i povolení pro interakci s jinými aplikacemi.
- Deklarace minimální verze Android, kterou aplikace vyžaduje.
- Seznam knihoven, které do aplikace musí být připojeny.
- Dále obsahuje název aplikace, název služby, odkaz na ikony a číslo verze aplikace.[11]

2.1.3 Příprava prostředí

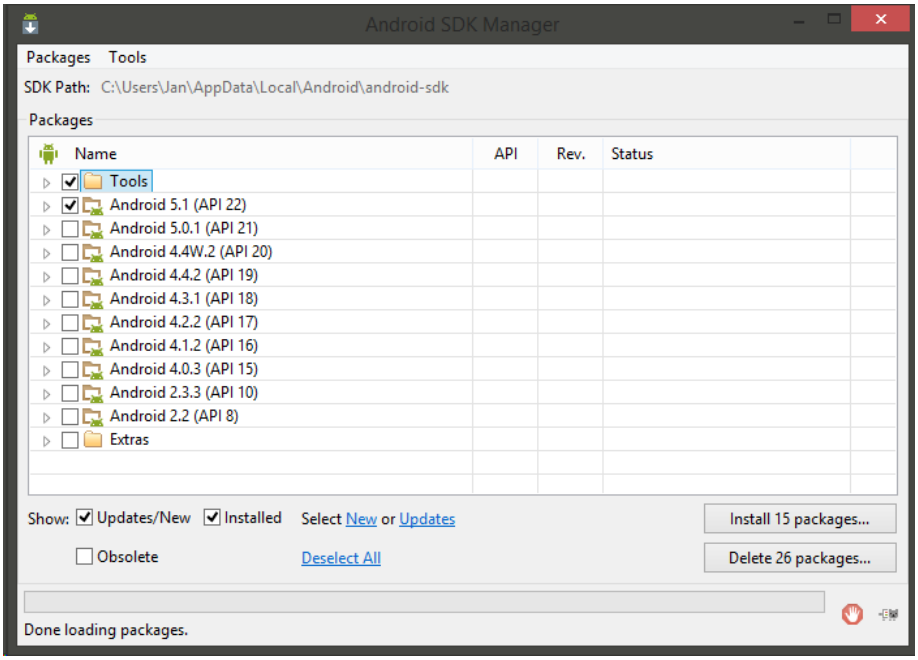
Před začátkem programování aplikací na Android je třeba nachystat veškeré komponenty a nástroje. Postup bude popsán pro operační systém Windows 8.1. Pro tvorbu na Linuxu je postup velmi podobný.

Java Develompent Kit

Aplikace pro Android se obvykle programují v jazyce Java. Zdrojový kód jazyka Java je následně převeden do formátu, se kterým Android skutečně pracuje(byte-kód Dalvik). K vývoji aplikací pro Android je potřeba oficiální sada Oracle Java SE Development kit(JDK). Sadu lze zdarma stáhnout z webu jazyka Java.[16]

Android SDK

Android SDK(Software development kit) poskytuje nástroje potřebné k vytváření a testování aplikací pro Android. Vývojové nástroje jsou zdarma ke stažení na webu Android Developers. Po rozbalení a spuštění SDK je třeba vybrat složku Tools a nejnovější verzi Android knihovny – tedy 5.1 v době psaní této bakalářské práce. Tyto je třeba nainstalovat.[14]

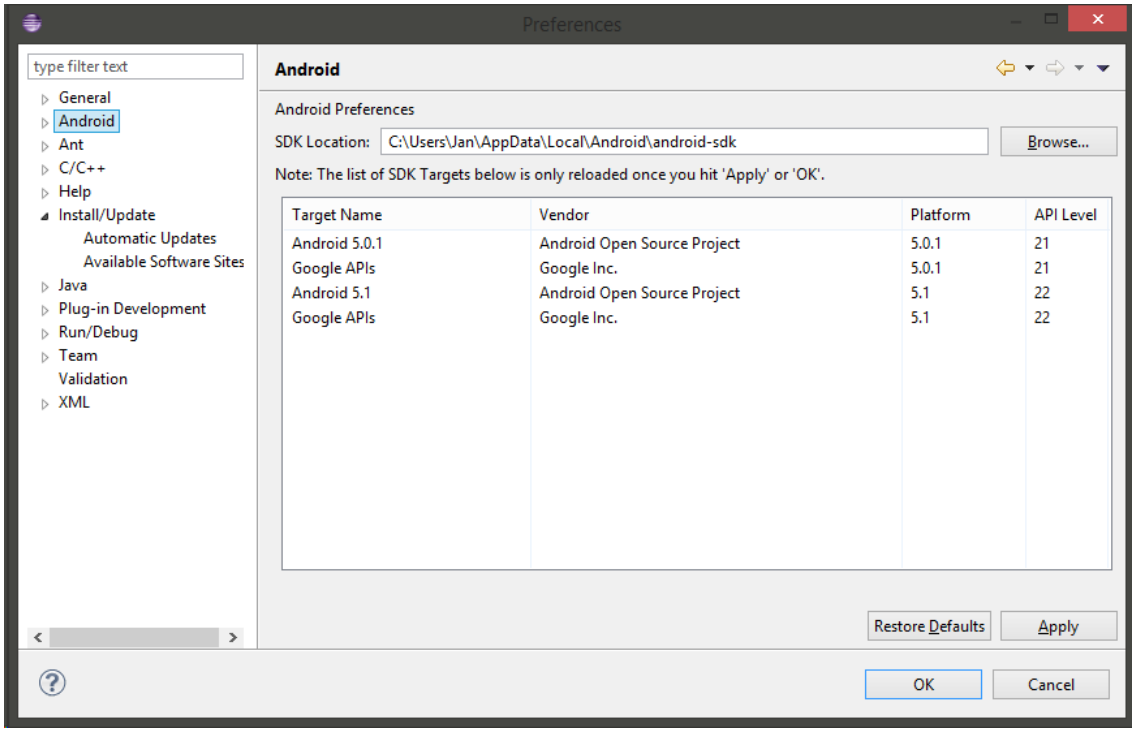


Obrázek 2.2: Android SDK s vybranými potřebnými položkami

Eclipse a ADT

Pro vývoj aplikací pro Android je možné používat prostředí Eclipse, které je možné si stáhnout z webu Eclipse – https://eclipse.org/. Ideální balíček je Eclipse IDE for Java Developers. [15]

Dále je třeba nainstalovat modul Android Developer Tools(ADT). V nainstalovaném prostředí Eclipse v nabídce Help je třeba kliknout na položku Install Software. Zde zadejte URL adresu: https://dl-ssl.google.com/android/eclipse/ a prostředí Eclipse si z této adresy stáhne ADT modul.



Obrázek 2.3: Eclipse instalace ADT

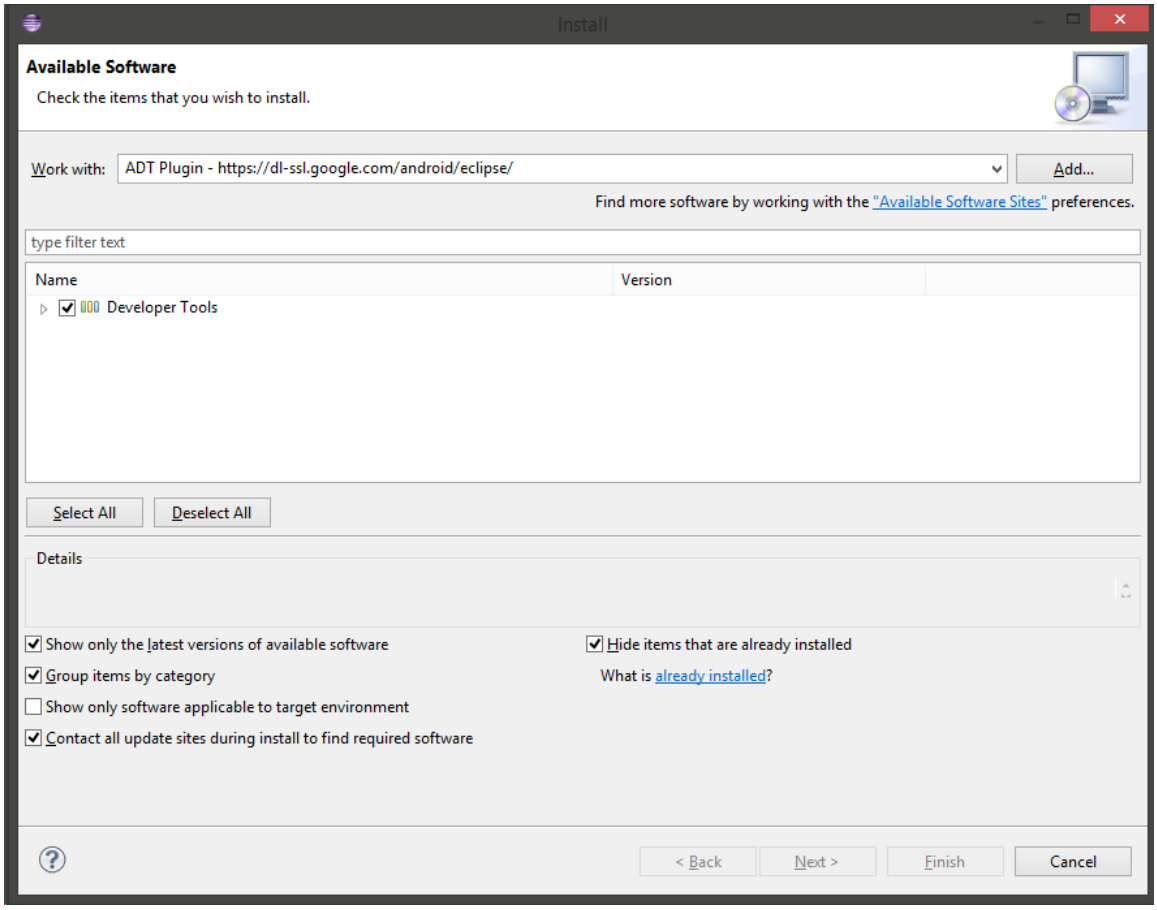
Poslední částí přípravy je přidání Android SDK do prostředí Eclipse přes nabídku Window a položku Preferences. Zde se nachází menu Android kde je třeba zadat cestu k dříve nainstalovanému Android SDK.

2.2 Další mobilní systémy

Pokud by nebyla zvolena jako cílová platforma Android pro klientskou mobilní aplikaci, kvalitní alternativy by byly platformy iOS od Apple, Windows Phone od Microsoftu nebo Symbian od Symbian Foundation. Všechny tři varianty byly však v případě práce zavrhnuty z důvodu nevlastnění telefonů s těmi operačními systémy(V případě iOS by byl potřeba navíc počítač s operačním systémem MAC OS). Proti Symbianu navíc hraje fakt, že jeho poslední oficiální update vyšel v roce 2012.[26] Ve prospěch Android naopak hrálo jeho zastoupení na trhu a autorova zkušenost s programováním v jazyku Java.

2.3 Nette

Webový server je navržený a naprogramovaný pomocí aplikačního rámce Nette, což je open source framework pro tvorbu webových aplikací v PHP 5. Zaměřuje se na eliminaci bezpečnostních rizik a rychlost. Využívá událostmi řízené programování a z velké části je založen na použití komponent. Původním autorem Nette Frameworku je David Grudl, o jeho další rozvoj se stará organizace Nette Foundation. [19][24]



Obrázek 2.4: Připojení Android SDK do Eclipse

2.4 Databáze

Jako databázový systém byl zvolen MySQL vytvořený švédskou firmou MySQL AB, současně vlastněný společností Sun Microsystems, která je dceřinou společností Oracle Corporation. Hlavními autory MySQL jsou Michael Widenius a David Axmark.

MySQL je multiplatformní databáze. Komunikace s ní probíhá pomocí dialektu jazyka SQL s některými rozšířeními. Pro svou snadnou implementovatelnost (lze jej instalovat na Linux, MS Windows a další operační systémy), výkon a především díky tomu, že se jedná o volně šiřitelný software, má vysoký podíl na v současné době používaných databázích. MySQL je od počátku optimalizování pro rychlost, za cenu určitých zjednodušení. Při práci s MySQL lze použít pouze jednodušší způsoby zálohování a pohledy, triggery se například v prvních verzích vůbec nevyskytovaly. [18][23]

Kapitola 3

Návrh systému

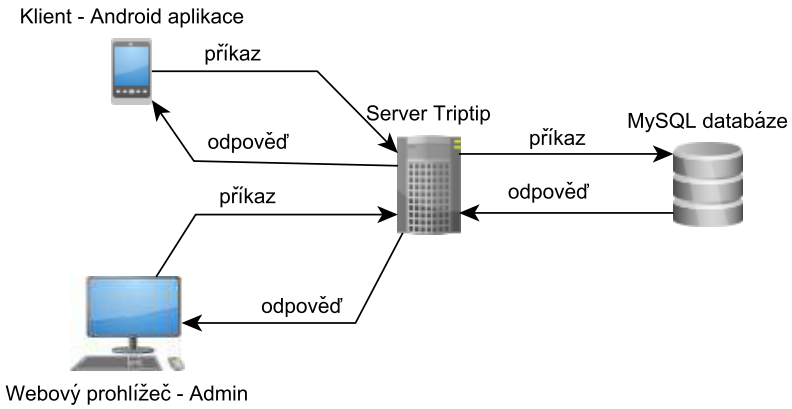
Architektura toho systému je založena na architektuře klient-server mezi klientem – aplikací pro Android a webovým serverem, který představuje kolekci programů dostupnou přes internet. K webovému serveru je dostupné administrační prostředí dostupné přes internetový prohlížeč. Viz Obrázek 3.1.

Charakteristika klienta – aplikace:

- Aktivní
- Posílá žádosti serveru
- Čeká a dostává odpovědi

Charakteristika serveru:

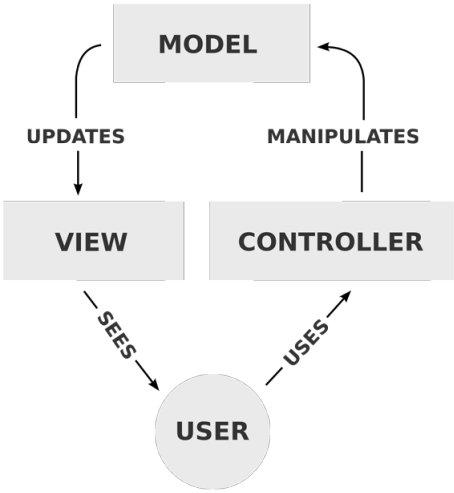
- Pasivní
- Naslouchá na síti a reaguje na žádosti klientů
- Při přijetí požadavku jej obslouží



Obrázek 3.1: Architektura systému

3.1 Návrh webového serveru

Při návrhu webového serveru byla snaha dodržet architekturu Model-View-Controller Viz obrázek 3.2. To je softwarová architektura, uživatelské rozhraní, rozdělující datový model aplikace a řídicí logiku do tří komponent tak, že modifikace některé z nich má jen minimální vliv na ostatní.



Obrázek 3.2: Architektura Model-View-Controller [22]

- Model – objekt reprezentující data, v tomto případě sada příkazů pro externí databázi.
- View – převádí stav modelu do podoby vhodné k interaktivní prezentaci
- Controller – poskytuje logiku systému. Stará se o změny dat a reaguje na události.[17][21]

3.2 Návrh klientské mobilní aplikace

Návrh klientské mobilní aplikace počítá s aplikací založenou na aktivitách a službě poskytující periodické snímání lokací. Služba navíc bude zařizovat motivační hodnotu ve formě zpráv závislých na lokaci, tato část bude hlavní motivační aspekt pro uživatele.

Kapitola 4

Implementace webového serveru

Webový server je implementovaný v architektuře MVC, kde Model zastupují soubory ve složce app/model/ starající se o zprostředkování obsahu a změny v něm. Druhou část architektury View zastupují soubory latte ve složce app/templates/. Řídící část architektury Controller reprezentují soubory ve složce app/presenters/.

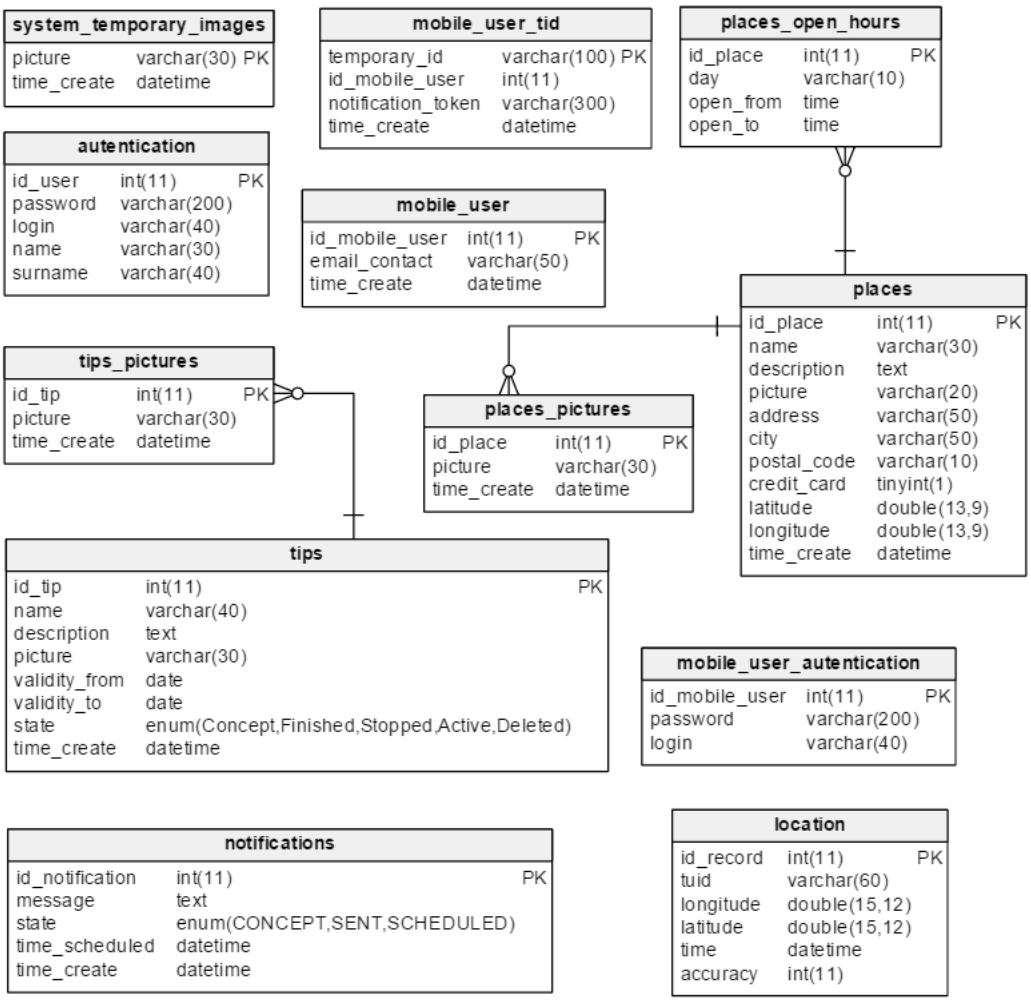
4.1 Struktura

V této sekci najdete hlavní prvky implementace webového serveru.

- Model
  - WebISModel – metody pro webové administrační rozhraní.
  - MobilAPIModel – metody pro komunikaci mezi klientskou aplikací a serverem.
- View
  - Bussiness/Deal create – zobrazení tvorby místa/tipu.
  - Bussiness/Deal default – zobrazení listu míst/tipů.
  - LocationData default – zobrazení mapy se záznamy.
- Controller – Presenter
  - MobileProtocolV1Presenter – řídící funkce pro komunikaci mezi klientskou. aplikací a serverem
  - LocationDataPresenter – řídící funkce pro práci s lokačními záznamy.
  - DealPresenter/BussinessPresenter – řídící funkce pro tvorbu místa/tipu.

4.2 Databáze

Při komunikaci mezi databází a webovým serverem se opět vyskytuje architektura klient-server. Webový server, v tomto vztahu klient, pošle dotaz do databáze a ta odpoví jako server. Pokud dotaz pochází z aplikace působí webový server jako zprostředkovatel. Přímé přístup do databáze, jinak než přes webový server napsaný v Nette by představoval bezpečnostní riziko. Obrázek 4.1 popisuje tabulky databáze použité na webovém serveru.



Obrázek 4.1: Schéma databáze webu

4.3 Komunikace a příkazy

Komunikace probíhá tak, že klient zašle příkaz HTTP POST na server, server příkaz zpracuje a odešle klientu odpověď formou JSONObjektu, ve které se může nacházet i JSONArray. Odpověď si klient rozparsuje a uloží do databáze. Lokace odesílaná z telefonu na web se posílá také ve formě JSONArray. Pouze obrázky se z webu stahují metodou HTTP GET.

Příkazy odesílané na web, jsou rozdělené na dvě skupiny podle tabulek s příkazy 4.1 a 4.2. V první tabulce jsou příkazy, které žádají o data zobrazovaná a posílají se vždy při zapnutí aplikace Viz Tabulka 1 4.1. Tyto příkazy nezaznamenávají žádná data na server.

Druhá skupina příkazů v Tabulce 2 4.2 je vyvolaná interakcí uživatele s aplikací, kromě příkazu na odeslání lokací, který se provede také vždy při zapnutí aplikace. Odeslaná data se zapisují do databáze na serveru.

Příkaz – PostType	Parametry	Odpověď
listPlaces	• appKey	JSONArray s místy
listNotifications	• appKey	JSONArray s notifikacemi
listTips	• appKey	JSONArray s tipy

Tabulka 4.1: Tabulka příkazů 1 – stažení dat

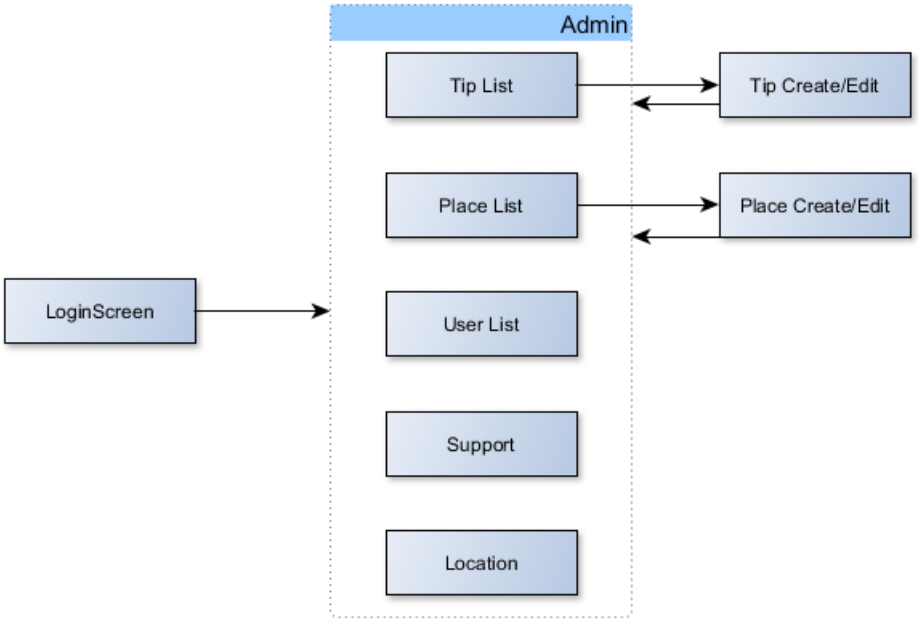
Příkaz – PostType	Parametry	Odpověď
login	• appKey • email • protection • hash	Výsledek přihlášení
register	• appKey • email • protection • hash	Výsledek registrace
setAccessStatistics	• appKey • locations • protection • device	Výsledek odeslání lokací
askForSupport	• appKey • protection • message	Výsledek odeslání zprávy

Tabulka 4.2: Tabulka příkazů 2 – komunikace

4.4 Uživatelské rozhraní

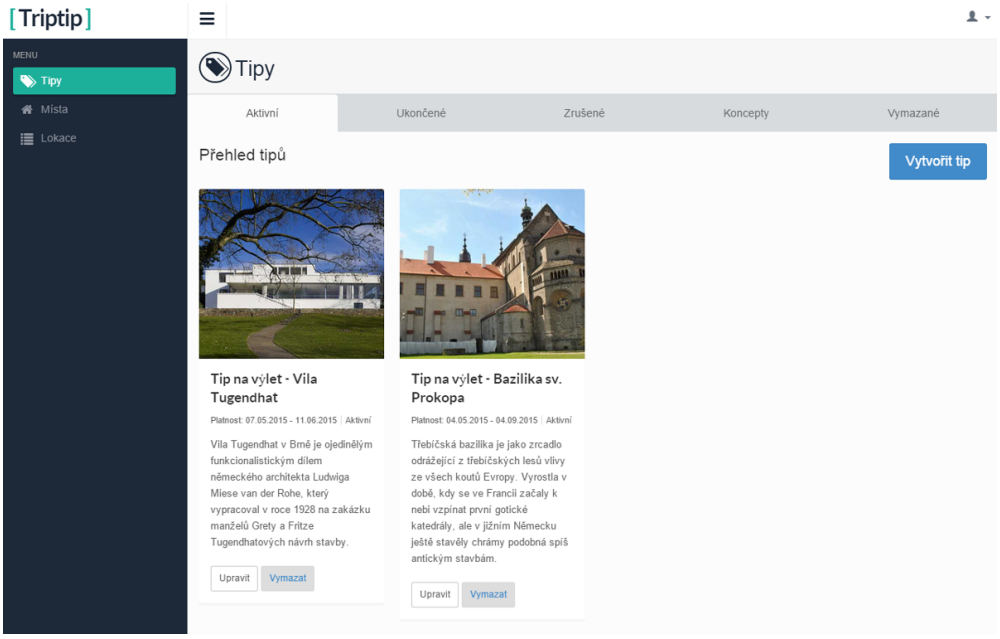
Pro implementaci uživalského rozhraní na webovém serveru jsem použil knihovnu Bootstrap 3, kombinující HTML, CSS3 a Javascript, a na nich postavený Bracket Responsive Bootstrap 3 Admin Template s ukázkou na obrázku 4.3.[20] Uživatel (v případě serveru se uživatelem myslí správce webu) může přes uživatelské rozhraní například jednoduše vytvářet, editovat nebo mazat nové tipy a místa nebo v obrazovce s mapou zobrazovat pohyb uživatelů podle vybraných kriterií.

Podle návrhu uživatelského rozhraní je přístup do systému otevřen až po přihlášení. Poté už je možný volný pohyb v rámci skupiny Admin, viz Obrázek 4.2.



Obrázek 4.2: Návrh uživatelského rozhraní webového serveru – označení obrazovek a přechody mezi nimi

Při ukládání obrázků na webový server, byl použit drag and drop plugin, umožnující obrázky přetáhnout z plochy nebo ze složky přímo do prostoru tohoto plugin. Možnost vybrat obrázek kliknutím a výběrem přes vyskakující okno zůstala zachována.



Obrázek 4.3: Snímek obrazovky webového serveru

Kapitola 5

Implementace klientské aplikace

Android aplikace Triptip naprogramovaná v rámci této bakalářské práce slouží k získávání polohy uživatelů a k zasílání notifikací v případě, že se uživatel přiblíží k některému z uložených lokačních bodů. Aplikace je složena ze čtyř prvků:

- Aktivity – soubor aktivit starající se o zobrazení obsahu a vykreslující uživatelské rozhraní.
- Data Manager – Data Manager se stará o aktuálnost dat, při zapnutí aplikace jej aktivuje hlavní aktivita. Data Manager pošle žádost na server o data a ty uloží přes SqLite Helper do databáze.
- SqLite Helper – Stará se o přístup do databáze. Data Manager přes SqLite Helper do databáze data ukládá, aktivity si přes SqLite Helper data načtou a zobrazí.
- Service Manager – Service Manager se stará o ukládání uživatelovy polohy. Je navržen tak, aby měl co nejmenší zátěž na baterii, uživatele nijak neobtěžoval a zobrazoval notifikace v případě, že se přibližná lokace shoduje s lokací uloženého místa v databázi.

5.1 Knihovny

Při programování této Android aplikace Triptip byly použity následující volně dostupné knihovny, které je nutno zahrnout do aplikace:

- Google Play Services – Zajišťují přístup k lokačním službám.[10]
- Universal Image Loader – Zajišťuje jednoduché asynchronní stažení a zobrazení obrázku.[9]
- Android Support v7 Appcompat – Pro podporu starších verzí systému Android. Díky této knihovně je minimální Android verze pro tuto aplikaci – Android 2.3.3 Gingerbread.[8]

5.2 Databáze

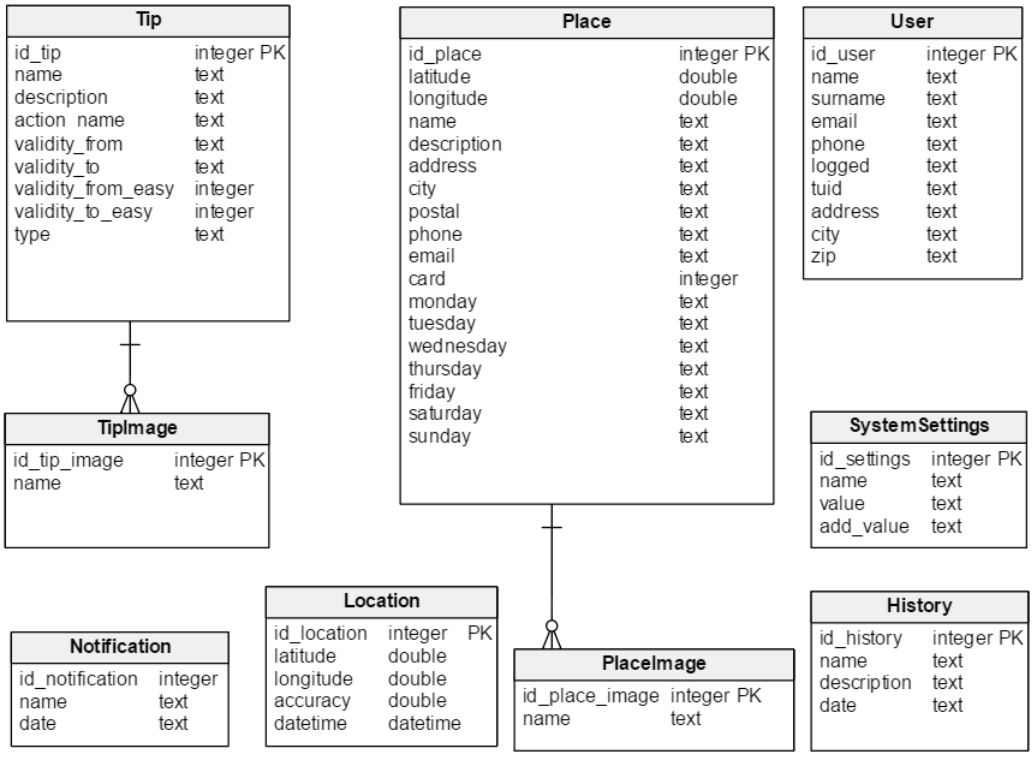
Android používá pro aplikace databázový systém SQLite, který obsluhuje třída SQLiteOpenHelper[7], ze kterého jsem vytvořil potomka (cz.triptip.sql.SQLiteHelper) s

funkcemi pro inicializaci databáze, smazání, nahrávání, úpravu a načtení záznamů z databáze. Pole záznamů je vždy vkládáno do databáze jako celek. Všechny pole se načítají a ukládají v asynchronních vláknech. Všechny přístupy do databáze jsou ve funkcích ošetřeny smyčkou proti mnohonásobného přístupu do databáze, viz Algoritmus viz ukázka ukládání do databáze 5.1.

Algoritmus 5.1 Ukázka funkce na zápis do databáze telefonu

```
    public long insertLocation(LocationDB locationEntry) {
    SQLiteDatabase db = getWritableDatabase();
    while (db.isDbLockedByCurrentThread()) {
    }
    // Create a new map of values, where column names are the keys
    ContentValues values = new ContentValues();
    values.put(LocationEntry.COLUMN_LATITUDE, locationEntry.getLatitude());
    values.put(LocationEntry.COLUMN_LONGITUDE, locationEntry.getLongitude());
    values.put(LocationEntry.COLUMN_ACCURACY, locationEntry.getAccuracy());
    values.put(LocationEntry.COLUMN_DATETIME, locationEntry.getDateTime());

    // Insert the new row, returning the primary key value of the new row
    long id_location;
    while (db.isDbLockedByCurrentThread()) {
    }
    id_location = db.insert(LocationEntry.TABLE_NAME, "null", values);
    return id_location;
}
```

Databáze využívá minimálního propojení mezi entitami, jelikož slouží hlavně jako úložiště dat, viz schéma 5.1.

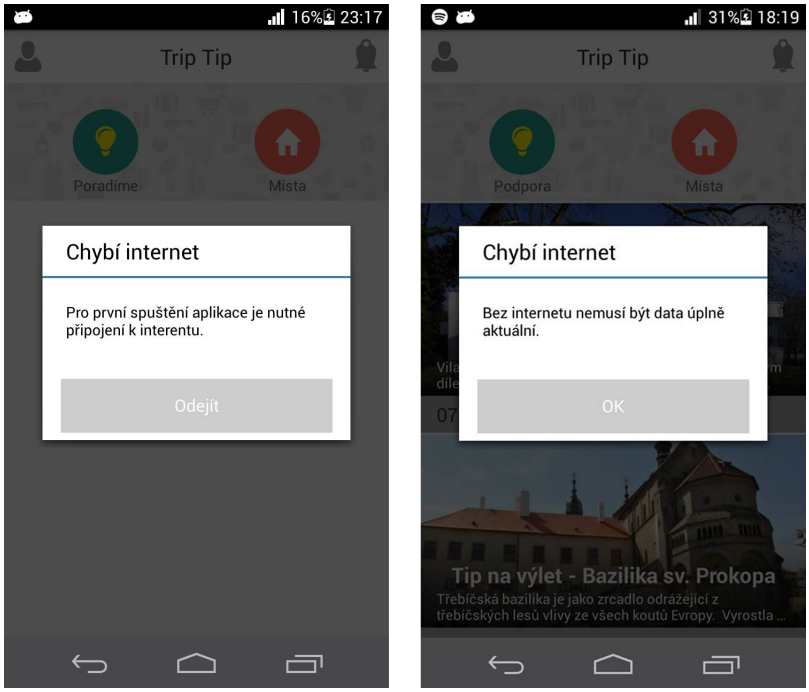


Obrázek 5.1: Schéma databáze aplikace

5.3 Data Manager

Data Manager je potomkem třídy Application a slouží k posílání dotazů na web. Při zapnutí aplikace pošle dotazy o pole míst, tipů, notifikací. Po obdržení odpovědi se přijatá data asynchronně uloží do databáze. Dále při startu aplikace odešle data o lokacích uložená v databázi, která pravidelně zaznamenává služba. Data o lokaci se posílají dávkově při zapnutí, aby se dosáhlo šetření ve spotřebě baterie a datovém provozu.

Jelikož data potřebná v aplikaci jsou stahovaná ze serveru, probíhá při startu kontrola na dostupnost internetu. Pokud je aplikace zapnuta poprvé a bez dostupného připojení, Data Manager zobrazí dialog a poté se ukončí aplikaci. Pokud aplikace již má nějaká data načtená a není internet dostupný, pouze se zobrazí dialog o tom, že data v aplikaci nemusí být aktuální.

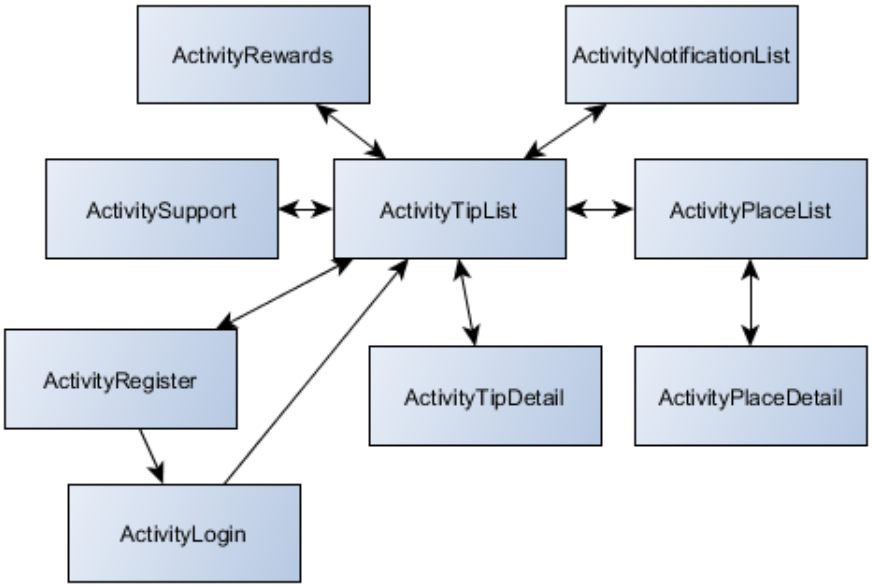


Obrázek 5.2: Pro první zapnutí je potřebné připojení k internetu, pokud jsou nějaká data načtena je to jen doporučení

5.4 Aktivity a uživatelské rozhraní

Aktivity jsou prvkem, který tvoří základ aplikace.

Hlavní aktivitou je ActivityTipList, proto ji systém zapne při startu aplikace. Jak je vidět z návrhu uživatelského rozhraní 5.3, tak většina Aktivit je dostupná z AktivityTipList. Pouze aktivity ActivityPlaceDetail a ActivityLogin nejsou přímo dostupné z ActivityTopList a přístup k nim je přes jejich rodičovskou aktivitu.



Obrázek 5.3: Návrh uživatelského rozhraní aplikace – označení(aktivit) obrazovek a přechody mezi nimi

Stisk tlačítka zpět mimo hlavní aktivitu provede standardní akci, což je ukončení současné aktivity a spuštění rodičovské aktivity. Vyjímku tvoří pouze ActivityLogin, která spustí při tlačítku Zpět ActivityTipList. Nedošlo-li k vypnutí aplikace omylem, tak v hlavní aktivitě je pro tlačítko zpět nastavena funkce, která vyžaduje zmáčknutí tlačítka zpět dvakrát v rozmezí dvou sekund viz Algoritmus 5.2.[25]

Obrázky v aktivitách se načítají asynchronně pomocí Universal Image Loaderu. Image Loader má vlastní cache paměť, proto obrázky z internetu načte pouze jednou. Poté již načítá ze svého úložiště.

Algoritmus 5.2 Double press for exit

```
private boolean doubleBackToExitPressedOnce = false;

@Override
public void onBackPressed() {
   if (doubleBackToExitPressedOnce) {
      super.onBackPressed();
      return;
   }
   this.doubleBackToExitPressedOnce = true;
   Toast.makeText(this, getString(R.string.doubleExit),
       Toast.LENGTH_SHORT).show();
   new Handler().postDelayed(new Runnable() {
      @Override
      public void run() {
         doubleBackToExitPressedOnce = false;
      }
   }, 2000);
```

```
}
```

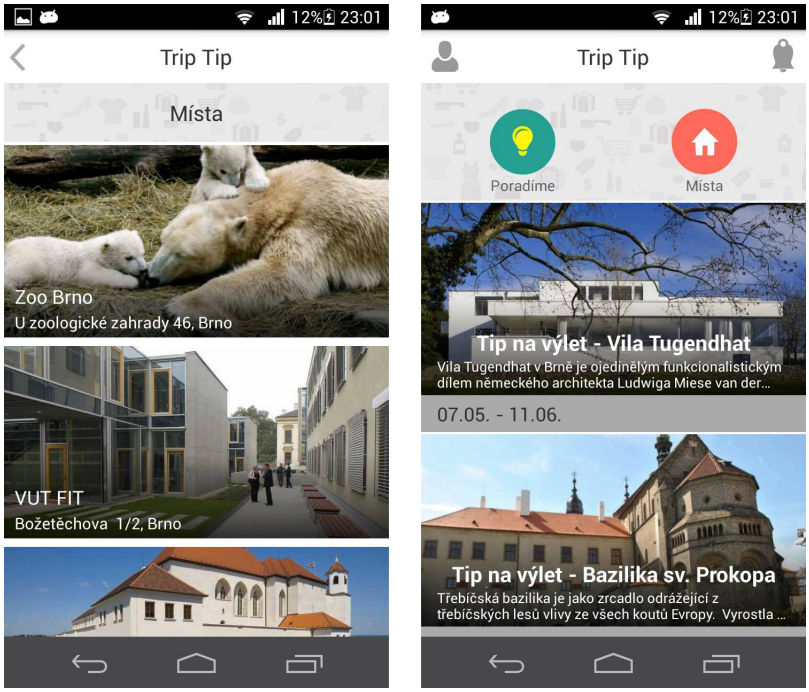
Aktivity v této aplikaci jsou rozděleny do čtyř následujících tříd:

5.4.1 Aktivity s listem

Aktivity na zobrazování listů záznamů. Každá z těchto aktivit má adaptér. Aktivita při startu nastaví adaptér na ListView z načteného layoutu. Poté si z databáze uloží do paměti ArrayList dat, který pošle adaptéru a ten se postará o zobrazení dat. Mezi tento typ aktivit patří ActivityTipList, ActivityPlaceList a ActivityNotificationList. Adaptéry PlaceAdapter a TipAdaptér mají navázané OnClickListenery na každou položku v listu, kterým vyvolají zobrazení Aktivity s detailem. Id položky která se bude zobrazovat, se do aktivity pošlou přes Intent Viz algoritmus 5.3.

Algoritmus 5.3 Odeslání intentu s identifikátorem

```
@Override
public void onClick(View v) {
   TipDB tip = list.get(position);

   Intent main = new Intent(cntxt, ActivityTipDetail.class);
   main.putExtra(ConstantsTripTip.ID_TIP, tip.getId());
   main.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK |
       Intent.FLAG_ACTIVITY_MULTIPLE_TASK);
   cntxt.startActivity(main);

}
```



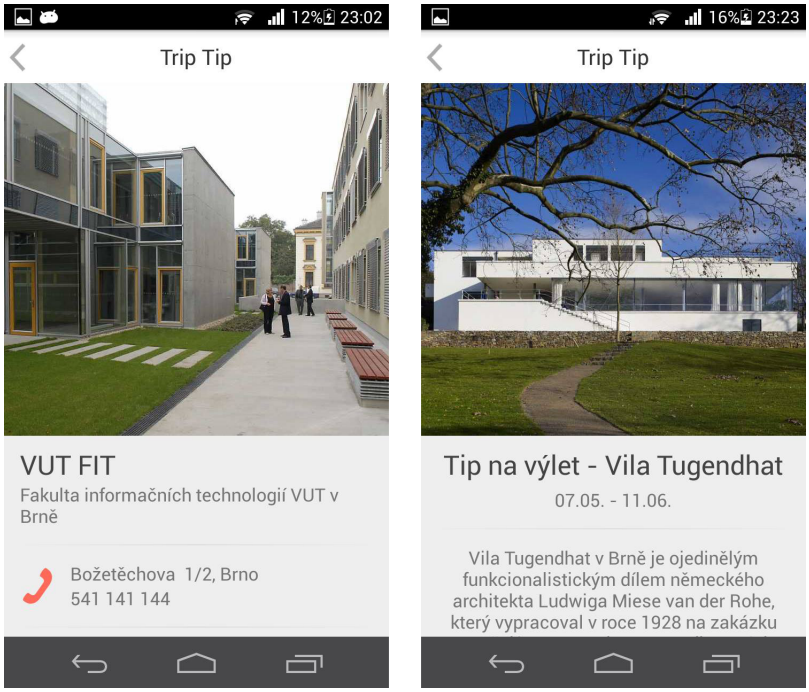
Obrázek 5.4: Ukázky aktivity – ActivityPlaceList(vpravo) a ActivityTipList

5.4.2 Aktivity Detail

Mezi tyto aktivity patří ActivityTipDetail a ActivityPlaceDetail. Ty si ve funkci `onCreate()` načtou id položky do globální proměnné pomocí intentu Viz algoritmus 5.4. Poté si načtou instanci SQLiteHelper a stáhnou si data o konkrétní položce, kterou potom vykreslí do aktivity.

Algoritmus 5.4 Příjem intentu s identifikátorem

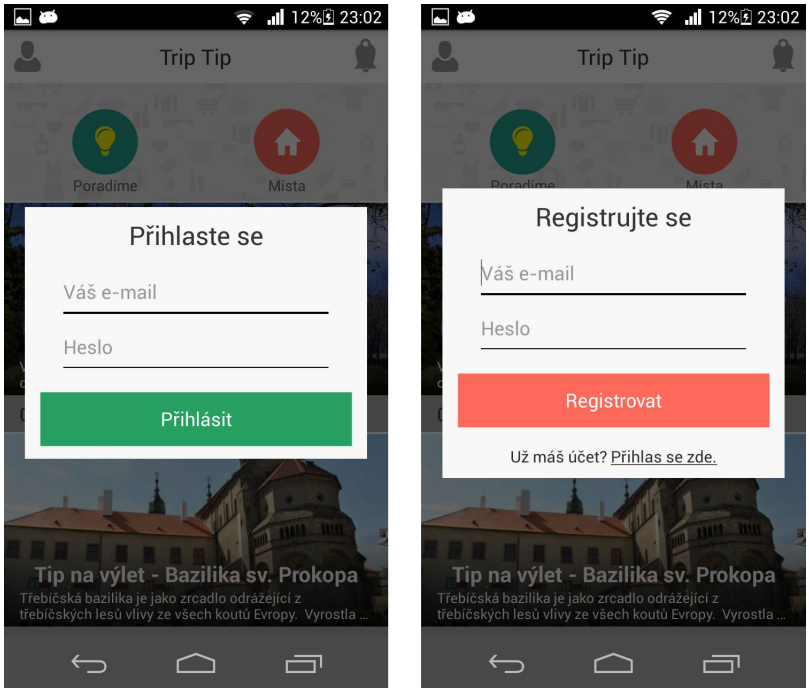
```
Integer id_tip = 0;

 @Override
 protected void onCreate(Bundle savedInstanceState) {
   ...

   Intent mIntent = getIntent();
   id_tip = mIntent.getIntExtra(ConstantsTripTip.ID_TIP, 0);
 }
```



Obrázek 5.5: Ukázky aktivity – ActivityPlaceDetail(vpravo) a ActivityTipDetail

5.4.3 Přihlašovací aktivity

Aktivity pro přihlášení nebo registraci – ActivityLogin, ActivityRegister mají nastavený vzhled, aby vypadaly jako dialog. Toho je docíleno tak, že v AndroidManifestu mají místo klasického stylu nastaveno `android:theme=@android:style/Theme.Dialog`, čímž se vizuálně chovají jako dialog. Aktivita provede kontrolu vstupu a v případě, že je správný, odešle HTTP POST dotaz na webový server, který přihlášení zpracuje a odešle stav přihlášení. Záporné odpovědi zavolají vypsání chyby. Kladná uloží uživatele do vnitřní databáze zařízení a ukončí aktivitu.

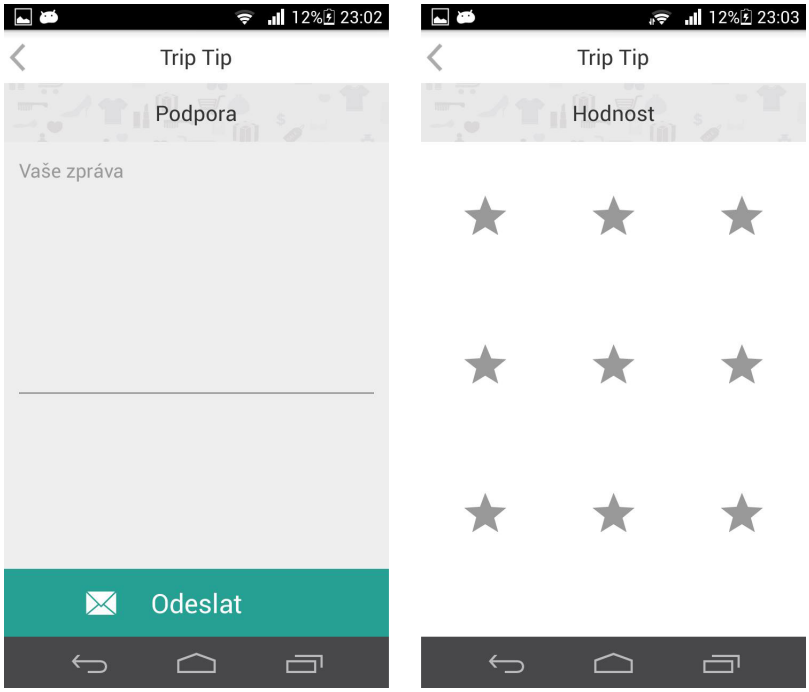


Obrázek 5.6: Ukázky aktivity – ActivityLogin(vpravo) a ActivityRegister

5.4.4 Zbývající aktivity

Pro vstup do aktivity ActivitySupport je třeba být přihlášen do systému. V Aktivitě je možnost odeslat zprávu pro administrátora systému nebo zpětnou vazbu na aplikaci. Možnost poslat zprávu vývojářům přes aplikaci je dobrá praxe, která může předejít v případě nespokojenosti uživatele špatnému hodnocení na Google Play obchodu.[2]

Akvitita ActivityRewarads zobrazuje počet míst ve formě vyplněných hvězd, které uživatel s aplikací navštívil. Tyto záznamy se tvoří v ServiseManager a jsou uloženy v SQLite databázi.



Obrázek 5.7: Ukázky aktivity – ActivitySupport(vpravo) a ActivityRewarads

5.5 Služba

Služba ServiceManager přijímá lokaci z Location Manageru. Třída Location Manager poskytuje přístup do systému lokačních služeb. Tyto služby dovolují aplikaci periodicky obnovovat geografickou lokaci zařízení.[13] Location Manager má možnost využít jako zdroj lokace:

- GPS poskytovatel – načítá lokaci pomocí GPS. Tato varianta je nejpřesnější ale má největší spotřebu baterie. Odchylka je v řádech metrů.
- Síťový poskytovatel – Méně přesná varianta. Odchylka se pohybuje mezi metry až do kilometrů. Síťový poskytovatel je přesnější v případě že telefon zachytí Wi-Fi síť. Google vlastní databázi fyzických lokací Wi-Fi sítí, kterou buduje od roku 2007. Díky ní dokáže telefon mnohem přesněji zjistit lokaci, bez velké spotřeby baterie.[12]
- Pasivní poskytovatel – Nespotřebovává žádné, zdroje jelikož nevytváří dotaz na lokaci pouze zachytí lokace od síťového a GPS poskytovatele a ty pak předá dále. Pokud žádné dotazy na lokaci neprobíhají je pasivní poskytovatel neúčinný.

Abych snížil rozsah telefonů, na kterých by sbírání lokací nefungovalo implementoval jsem Location manager s pasivním providerem i se síťovým poskytovatelem. A to z důvodu, že z neznáme příčiny na modelu HTC One X nefungoval síťový poskytovatel a na modelu Motorola XT615 pasivní poskytovatel.

Obnovení lokace je nastaveno na podmínky jednou za 10 minut, nebo při změně lokace o více jak 500 metrů. Každá přijatá změna se uloží do databáze a při zapnutí aplikace se data odešlou na web a při potvrzeném příjmu jsou vymazány z databáze.

Při tvorbě služby bylo zásadní vyřešit problém služby, která by fungovala neustále. Při tvorbě jsem zkoušel službě přidat funkci startForeground(), která službu přichytí na notifikaci, a služba by poté měla fungovat i po vypnutí aplikace. Služba přestože podle systému standardně fungovala, nevykazovala žádnou činnost. Usuzuji, že při nedostatku paměti jí android odebral zdroje a proto nemohla plně fungovat.

Problém jsem vyřešil tak, že službu zapínám pomocí BroadcastReceiveru WakeUp 5.5 vždy, když se změní záznam telefonního stavu nebo při startu telefonu. Do změny stavu se počítá například odchozí a příchozí hovor nebo SMS. Toto řešení jsem navrhl a experimentálně otestoval a dosahuje dobrých výsledků.

Algoritmus 5.5 BroadcastReceiver WakeUp

```
public class WakeUp extends BroadcastReceiver {
    @Override
    public void onReceive(Context arg0, Intent arg1) {
      Intent service = new Intent(arg0, ServiceManager.class);
      arg0.startService(service);
    }
}
```

Kapitola 6

Testování systému

Testování toho systému bylo rozděleno na podbody dokud nebyl celý systém kompletní a plně funkční. Zde se dozvíte jak probíhaly jednotlivé části testování.Na otestování systému a poskytnutí svých lokačních dat, jsem požádal pár svých přátel zmíněných v poděkování.

6.1 Webový server

Funkčnost webového serveru a jeho administračního rozhraní, jež bylo testováno především skrze jeho administrační rozhraní a debugovací režim aplikačního rámce Nette přes webové prohlížeče Mozilla, Chrome a Internet Explorer.

Komunikační rozhraní mezi klientskou aplikací a webovým serverem bylo použitím aplikace otestováno pomocí doplňku k prohlížeči Chrome – Postman, umožnující posílat příkazy POST na webový server a zobrazit odpověď.

6.2 Aplikace

Vytvořit kvalitní a stabilní aplikaci pro Android není jednoduché a jelikož velkému množství přístrojů a konfigurací to vyžaduje i důkladné testování. Při testování sbírání lokace byly využity telefony s operačním systémem Android Viz tabulka 6.1

Výrobce	Model	Verze OS
HTC	One X	4.0.4
Motorola	XT615	2.3.3
Huawei	P6	4.2.2
Huawei	P6	4.4
Samsung	G800	4.2.2

Tabulka 6.1: Tabulka testovaných telefonů

Stabilita, vyladěný vzhled a správná funkčnost klientské aplikace byly testovány především mnou a od testerů jsem na toto téma dostal minimální zpětnou vazbu.

Kvalita snímání lokace a zasílání notifikací služby byla nejnáročnější částí na implementaci i testování. Lokační data jsou zaznamenávaný pouze pokud je služba aktivní a to se nedá zařídit se stoprocentní jistotou. V tabulce 6.2 je vidět výsledek 24 hodinového sbírání lokací před odesláním na webový server. Je zde vidět, služba zaznamenává data v

dostatečně krátkých interval. Výkyvy v intervalech jsou způsobeny pozdějším startem služby po jejím ukončení. Zvětšení odchylky (accuracy) je způsobeno nedostupností lokačního poskytovatele od síťového poskytovatele.

latitude	longitude	accuracy(m)	datetime
49.2264	15.8699	26	2015-05-16 18:18:52
49.2264	15.8699	26	2015-05-16 18:19:07
49.2197	15.8737	2684	2015-05-16 19:39:15
49.2264	15.8699	26	2015-05-16 21:06:19
49.2197	15.8737	2684	2015-05-16 23:34:31
49.2264	15.87	40.5	2015-05-16 23:38:13
49.2264	15.8699	25	2015-05-17 00:08:25
49.2264	15.8698	21	2015-05-17 00:38:49
49.2265	15.8699	23	2015-05-17 01:08:56
49.2265	15.8699	24	2015-05-17 01:39:04
49.2264	15.8699	30	2015-05-17 02:09:10
49.2264	15.87	36	2015-05-17 04:04:32
49.2264	15.8699	30	2015-05-17 08:13:16
49.2264	15.8699	22	2015-05-17 08:59:06
49.2265	15.8699	20	2015-05-17 09:29:11
49.2264	15.87	30	2015-05-17 09:59:24
49.2264	15.87	20.282	2015-05-17 10:29:28
49.2019	15.9016	3796	2015-05-17 11:07:38
49.2106	15.9105	2742	2015-05-17 11:24:29
49.2156	15.8944	40.5	2015-05-17 11:36:14
49.2264	15.8701	21.332	2015-05-17 11:52:48
49.2197	15.8737	2684	2015-05-17 12:09:16
49.2197	15.8737	2684	2015-05-17 12:56:10
49.2264	15.8699	26	2015-05-17 13:09:10
49.2147	15.8892	2350	2015-05-17 13:34:18
49.2264	15.8701	32.091	2015-05-17 14:04:41
49.2197	15.8737	2684	2015-05-17 14:19:16
49.2264	15.87	20.182	2015-05-17 14:34:12
49.2197	15.8737	2684	2015-05-17 14:47:17
49.2267	15.8711	2526	2015-05-17 15:19:48
49.2197	15.8737	2684	2015-05-17 15:38:11
49.226	15.8683	2867	2015-05-17 16:08:16
49.2197	15.8737	2684	2015-05-17 16:38:33
49.2197	15.8737	2684	2015-05-17 18:16:54

Tabulka 6.2: Výsledek dvaceti čtyř hodinového testování sběru lokací v telefonu

Kapitola 7

Závěr

Cílem této práce bylo vytvořit server-klient systém, jehož první částí je mobilní klient aplikace na platformě Android, která získává lokační data odesílá je na server, poskytuje uživateli obsah a upozorní uživatele v případě, že se vyskytne na specifické lokaci zaznamenané v databázi. Druhou částí je webový server, který přes administrativní rozhraní umožňuje tvorbu dat zasílaných klientským aplikacím, příjímání lokací a jejich vizuální zobrazení.

Nejkritičtější část – pravidelné snímání lokace uživatele a to i v případě, že nemá aplikaci zapnutou, ale i vše bylo splněno a systém je funkční a bylo by zajímavé ho obchodně uplatnit. Není ovšem jasné a ani bych netipoval, zda by měla aplikace založená na zaznamenávání pohybu úspěch v případě placené aplikace nebo aplikace s reklamou.

V rámci rozšíření by bylo zajímavé systém propojit s nějakým obchodním řetězcem, který by mohl zobrazovat reklamu ve formě notifikace pouze v případě, že se uživatel nachází v blízkosti jeho pobočky. Díky čemuž by vývoj a provoz systému financoval potenciální zájemce o rozesílání geograficky závislých upozornění. Uživatele aplikace by mohly motivovat slevy dostupné pouze pro ně.

Jako druhou variantu rozšíření by byla možnost systém přetransformovat, tak aby zákazníci mohli vkládat své podniky/pobočky a speciální nabídky pomocí webového rozhraní. Uživatelé klientské aplikace by pak získaly možnost využít tyto speciální nabídky po obdržení geograficky závislých upozornění.

Při těchto rozšíření by však bylo důležité udržet pokles výdrže baterie telefonu s klientskou aplikací na minimu.

Literatura

[1] Grant, A.: *Android 4 Průvodce programováním mobilních aplikací*. Computer press, 2013, 978-80-251-3782-6.

[2] Grant, A.: *Tapping In ? 21 Strategies for Building Viral Mobile Apps People Will Love*. David Paul Albert, 2014, 9781311946232.

[3] WWW stránky: Tutorials point - Android Architecture. `http://www.tutorialspoint.com/android/android_architecture.htm`, [cit. 2015-04-28].

[4] WWW stránky: Open Handset Alliance. `http://www.openhandsetalliance.com/`, [cit. 2015-05-01].

[5] WWW stránky: General Public Licence. `http://www.gnugpl.cz/`, [cit. 2015-05-04].

[6] WWW stránky: Android developer - Fundamentals. `http://developer.android.com/guide/components/fundamentals.html`, [cit. 2015-05-05].

[7] WWW stránky: Android developer - SQLiteOpenHelper. `http://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html`, [cit. 2015-05-05].

[8] WWW stránky: Android support v7 Appcompat. `https://github.com/koush/android-support-v7-appcompat/`, [cit. 2015-05-05].

[9] WWW stránky: Android Universal Image Loader. `https://github.com/nostra13/Android-Universal-Image-Loader`, [cit. 2015-05-05].

[10] WWW stránky: Google Play Services. `https://github.com/rakyll/google-play-services`, [cit. 2015-05-05].

[11] WWW stránky: Android manifest. `http://developer.android.com/guide/topics/manifest/manifest-intro.html`, [cit. 2015-05-06].

[12] WWW stránky: Router location. `http://www.huffingtonpost.com/2011/04/25/android-map-reveals-router-location_n_853214.html`, [cit. 2015-05-06].

[13] WWW stránky: Android developer - Location. `http://developer.android.com/reference/android/location/LocationManager.html`, [cit. 2015-05-07].

[14] WWW stránky: Android SDK. https://developer.android.com/sdk/index.html, [cit. 2015-05-08].

[15] WWW stránky: Eclipse Tools. https://eclipse.org/, [cit. 2015-05-08].

[16] WWW stránky: Oracle - Java SE. http://www.oracle.com/technetwork/java/javase/overview/, [cit. 2015-05-08].

[17] WWW stránky: 2c - Model View Controller. http://c2.com/cgi/wiki?ModelViewController, [cit. 2015-05-10].

[18] WWW stránky: Domovské stránky MySQL. https://www.mysql.com/, [cit. 2015-05-10].

[19] WWW stránky: Domovské stránky Nette Frameworku. http://nette.org/cs/, [cit. 2015-05-10].

[20] WWW stránky: Themeforest - Bracket Responsive Bootstrap 3 Admin Template. http://themeforest.net/item/bracket-responsive-bootstrap-3-admin-template/6894362, [cit. 2015-05-10].

[21] WWW stránky: Wikipedie - Model View Controller. http://cs.wikipedia.org/wiki/Model-view-controller, [cit. 2015-05-10].

[22] WWW stránky: Wikipedie - Model View Controller Image. http://upload.wikimedia.org/wikipedia/commons/thumb/a/a0/MVC-Process.svg/2000px-MVC-Process.svg.png, [cit. 2015-05-10].

[23] WWW stránky: Wikipedie - MySQL. http://cs.wikipedia.org/wiki/MySQL, [cit. 2015-05-10].

[24] WWW stránky: Wikipedie - Nette Framework. http://cs.wikipedia.org/wiki/Nette_Framework, [cit. 2015-05-10].

[25] WWW stránky: Android Snippets - Double back press to exit. http://www.androidsnippets.com/double-back-press-to-exit, [cit. 2015-05-12].

[26] WWW stránky: Wikipedie - Symbian. http://en.wikipedia.org/wiki/Symbian, [cit. 2015-05-18].

Příloha A

Obsah elektronického nosiče

Struktura elektronického nosiče:

- **/technicka-zpravasrc/** - zdrojové soubory technické zprávy
- **/technicka-zprava.pdf** - technická zpráva ve formátu PDF
- **/Triptip.apk** - instalační soubor pro systém Android vytvořené aplikace Triptip
- **/klient-src/** - zdrojové soubory Android aplikace Triptip
- **/klient-libs/** - knihovny potřebné pro překlad aplikace Triptip
- **/server-src/** - zdrojové soubory webového serveru
- **/database.sql** - kopie databáze webového serveru

Příloha B

Přístupové údaje k administračnímu rozhraní

Webový server je dostupný na doménové adrese http://junglak.cloudapp.net/triptip/www/index.php. Přístupové údaje pro vstup do systému:

- Email – vutbrfit@test.cz
- Heslo – romeo416

Příloha C

Instalace webového serveru

Pro instalaci webového serveru je třeba mít funkční prostředí pro webový server s databází MySQL a PHP. Postup je poté následující:

1. Importuje databázi.
2. Zkopírujte zdrojové soubory webového serveru.
3. Nastavte připojení na databázi v souboru **app/config/config.neon**.
4. Nastavte úvodní stránku serveru na soubor **www/index.php**.