ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

Fakulta elektrotechnická

Katedra počítačů

Bakalářská práce

Měření síly signálu v datových mobilních sítích pomocí zařízení s Androidem

Zdeněk Svatoň

Vedoucí práce: Ing. Jakub Doležal

Studijní program: Softwarové technologie a management, Bakalářský

Obor: Softwarové inženýrství

22. května 2015

Čestné prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principu při přípravě vysokoškolských závěrečných prací.

Datum: 21. 5. 2015

……..……

České vysoké učení technické v Praze
Fakulta elektrotechnická

katedra počítačů

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: **Zdeněk Svatoň**

Studijní program: Softwarové technologie a management
Obor: Softwarové inženýrství

Název tématu: **Měření síly signálu v datových mobilních sítích pomocí zařízení s Androidem**

Pokyny pro vypracování:

Seznamte se s vývojem aplikací pro OS Android. Vytvořte aplikaci pro měření a vizualizaci síly signálu v mobilních sítích s následujícími funkcemi:

1) Detekce základnových stanic sítě, které jsou v dosahu telefonu.
2) Zobrazení síly signálu a dalších informací o základnových stanicích pomocí Android API.
3) Načtení poloh stanic a jejich zobrazení na mapě společně s polohou uživatele. Polohy stanic budou získány z webových zdrojů (upřesní vedoucí), poloha uživatele bude načtena pomocí GPS.
4) Zobrazení síly signálu stanice, ke které je telefon připojen, v reálném čase pomocí grafu. Detekce a vyznačení změny obsluhující stanice.
5) Zobrazení míst předchozích měření na mapě společně s průměrnou silou signálu.
6) Export informací o signálu a GPS polohy telefonu do textového souboru ve formátu CSV.

Vývoj aplikace provádějte v souladu s obecně platnými pravidly pro zajištění kvalitního kódu a dokumentace. Ověřte funkčnost aplikace změřením síly signálu v reálném prostředí.

Seznam odborné literatury:

Grant Allen: Android 4: průvodce programováním mobilních aplikací. Computer Press, 2013
Karim Yaghmour: Embedded Android. O'Reilly, 2013
Martin Sauter. From GSM to LTE: An Introduction to Mobile Networks and Mobile Broadband. Wiley, 2011
Abd-Elhamid M. Taha, Hossam S. Hassanein, Najah Abu Ali. LTE, LTE-Advanced and WiMAX: Towards IMT-Advanced Networks. Wiley, 2011

Vedoucí: Ing. Jakub Doležal

Platnost zadání: do konce letního semestru 2015/2016

L.S.

doc. Ing. Filip Železný, Ph.D.
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 26. 3. 2015

Abstrakt

Cílem této bakalářské práce je návrh, realizace a testování mobilní aplikace pro platformu Android umožňující získávání informací o aktuálním stavu mobilní sítě a zobrazování informací o kvalitě signálu a základnových stanicích pomocí Google Maps.

Abstract

The objective of this thesis is the design, implementation and testing of a mobile application for Android platform, which will be collecting data about actual state of mobile network and display information about signal quality of a base transceiver stations using the Google Maps.

Obsah

Seznam obrázků

Seznam tabulek

1. Úvod

Na trhu je dnes mnoho aplikací pro platformy iOS i Android, které se zabývají problematikou kvality signálu a pokrytí, méně už těch, které by poskytovaly údaje o základnových stanicích a jejich jednotlivých směrových anténách. Žádná z nich ovšem nespojuje naměřené hodnoty signálu se základnovými stanicemi nebo dokonce směrovými anténami. Proto vznikla aplikace Sigmet pro platformu Android, která umožňuje vytvořit statistiku naměřených údajů o pokrytí základnovou stanicí, ale i o větší oblasti, která může být pokryta několika základnovými stanicemi. Dále je možné sledovat, ve kterém bodě a při jakých hodnotách signálu, dochází k přepnutí mezi obsluhujícími stanicemi nebo anténami téže stanice.

Aplikace Sigmet vznikla také jako učební pomůcka, která by studentům umožňovala sledovat charakteristiky mobilních sítí a případně provádět detailní analýzu pokrytí jednotlivých oblastí. Data poté mohou být po exportu analyzována například pomocí nástroje Matlab a využita při vědecké práci v oboru mobilních sítí. Aplikace by mohla mít přínos i pro lidi žijícím v hůře pokrytých oblastech.

Mezi hlavní přínosy této aplikace patří měření síly signálu a agregování těchto hodnot k jednotlivým anténám, vyznačeným na mapě. Vzhledem k tomu, že mobilní operátoři v České republice neposkytují seznam svých vysílačů pomocí žádného veřejně dostupného API, byla tato data získána se stránek GSM web, pro který sbírají data její uživatelé. Ani zde nebylo dostupné žádné veřejné API, aplikace tedy získává data z této webové stránky pomocí http klienta. Nedostatek paměti pro vizualizaci nasbíraných dat na mobilním zařízení byl řešen agregací pomocí techniky geohashing.

2. Rešerše existujících produktů

Před samotnou analýzou byl proveden průzkum existujících produktů, které nabízejí podobné funkce. Patří mezi ně NetMonster, OpenSignal, Network Signal Info, NetMetr,NetworkRadar, iPhone Field Test a další. První tři jmenované jsou blíže popsány níže, ostatní se od nich příliš neliší.

2.1. NetMonster

Aplikace NetMonster je velice kvalitní produkt z hlediska získávání informací o vysílačích, umí je taktéž zobrazovat na mapě a využívá knihovnu pro shlukování značek vysílačů. Jeho velkou výhodu je, že data o vysílačích stahuje v podobě aktualizací. Veškerý další provoz pak může probíhat bez připojení k internetu a tím dochází k významné úspoře energie. Tato aplikace ovšem neumí ukládat, zobrazovat data o signálu a ukládat záznamy o jednotlivých měřeních. Další nevýhodu je, že ačkoli v databázi zařízení jsou údaje o všech základnových stanicích, na mapě jsou zobrazeny jen ty, ke kterým bylo zařízení v minulosti připojeno a nelze zobrazit všechny, které jsou v databázi.

Obrázek 2.1: Zobrazení mapy v aplikaci NetMonster [1]

2.2. OpenSignal

OpenSignal je velice kvalitní, ale komerční aplikace, dostupná pro platformy Android i iOS. Umožňuje kromě měření síly signálu, měřit například i rychlost odezvy připojení k internetu. Aplikace je crowdsource, takže každý uživatel ve chvíli kdy aplikaci používá, zároveň poskytuje naměřená data ostatním uživatelům aplikace. Tím vzniká velice detailní mapa pokrytí nejen signálem telefonním, ale i signálem bezdrátových sítí. Distribuovaná architektura aplikace navíc umožňuje ušetřit výkon mobilního zařízení, který bývá zpravidla poměrně malý, a provádět na výkon náročnější výpočty a databázové operace na výkonném serveru. Chybí ovšem možnost vytvářet a ukládat vlastní měření. Aplikace také neposkytuje informace o tom, kde a při jaké síle signálu došlo k přepojení k jiné základnové stanici.

Obrázek 2.2: Zobrazení mapy v aplikaci Open Signal [2]

2.3. Network Signal Info

Tato aplikace je zaměřená pouze na měření kvality signálu jak telefonního i signálu bezdrátových sítí. Dokáže zobrazit připojenou stanici na mapě, ale nezakresluje na mapu naměřený signál.

Obrázek 2.3: Ukázka grafu aplikace Network Signal Info [3]

3. Analýza

Formalizované a jasně specifikované požadavky jsou nezbytnou součástí vývoje jakéhokoli softwarového systému. Funkční požadavky odrážejí zákazníkovo očekávání toho, co by měl systém umožňovat, zatímco systémové požadavky jsou spíše zachycením různých omezení a podmínek na systém kladených, které přímo nesouvisí s jeho funkcionalitou.

Systém bude průběžně zaznamenávat data o naměřeném signálu a dělat o nich statistiky. Jeho největším přínosem bude možnost různých, pro uživatele přehledných vizualizací, jenž umožní analyzovat kvalitu signálu v oblastech, kde probíhalo měření. Předpokládá se vědecké a technické použití pro diagnostiku mobilních síti.

V případě této práce vystupuje v roli zákazníka vedoucí bakalářské práce, a požadavky jsou definované zadáním. Z těchto požadavků lze poté vytvořit seznam případů užití, který bude použit pro následující vývoj.

3.1. Funkční požadavky

Předem byly stanoveny požadavky na import dat (tab. 3.1), zobrazení dat (tab. 3.2), zobrazení událostí v aplikaci (tab. 3.3), vizualizaci pomocí mapy (tab. 3.4) a ukládání jednotlivých měření (tab. 3.5)

Tabulka 3.1: Požadavky na import dat

REQ 001	Import podrobností o vysílačích	Systém bude využívat import dat z webu gsmweb.cz.

Tabulka 3.2: Požadavky na zobrazení dat

REQ 002	Graf	Systém bude zakreslovat aktuální úroveň signálu do grafu a v grafu budou znázorněna i přepojení mezi anténami.

Tabulka 3.3: Požadavky na zobrazení událostí v aplikaci

REQ 003	Log	Systém bude zobrazovat stručně popsané události a případné chyby, které systému nastanou.

Tabulka 3.4: Požadavky na vizualizaci pomocí mapy

REQ 004	Zobrazení vysílačů	Systém bude zakreslovat jednotlivé vysílače do mapy spolu s podrobnostmi o nich.
REQ 005	Zobrazení signálu	Systém bude zakreslovat aktuální úroveň signálu do mapy.
REQ 006	Zobrazení změny antény	Systém bude zakreslovat místo, kde došlo ke změně připojení mezi anténami.
REQ 007	Mapa pokrytí	Systém bude umožňovat vytvořit mapu síly signálu z naměřených hodnot.
REQ 008	Zobrazení antény	Systém bude umožňovat vyhledání antény přímo na mapě.
REQ 009	Analýza oblasti	Systém bude zobrazovat statistiky o signálu a anténách agregovaných ke geografickým oblastem.

Tabulka 3.5: Požadavky na ukládání jednotlivých měření

REQ 010	Ukládání jednotlivých měření	Systém umožní zaznamenat a pojmenovat jednotlivá měření tak, aby je bylo možné zpětně analyzovat.
REQ 011	Export	Systém bude umožňovat export měření.
REQ 012	Smazání měření	Systém bude umožňovat smazání měření.
REQ 013	Seznam měření	Systém bude umožňovat prohlížet uložená měření.
REQ 014	Seznam antén	Systém bude umožňovat prohlížet uložená data o anténách.

3.2. Systémové požadavky

Systémové požadavky pro vývoj aplikace Sigmet uvedené v tabulce 3.6 specifikují nároky na zařízení, technologie použité při vývoji a prokazatelnost kvality kódu.

Tabulka 3.6: Systémové požadavky

REQ 015	Opensource	Celá aplikace bude vybudována na opensource technologiích.
REQ 016	Perzistence	Aplikace bude data o měření ukládat do databáze SQL Lite a bude využívat opensource ORM Framework.
REQ 017	Kompatibilita	Aplikace bude podporovat specifikované verze operačního sytému android a bude optimalizována pro zařízení s různou velikostí displeje.
REQ 018	Testovaní	Kritické části systému budou mít alespoň 85% pokrytí řádků pomocí jednotkových testů.
REQ 019	Kapacita	Aplikace musí být schopná zobrazit alespoň 1000 unikátních pozic antén na mapě a alespoň 100 000 údajů o měření signálu.

3.3. Možné problémy

V této podkapitole jsou shrnuta rizika spojená s provozem aplikace na mobilním zařízení.

3.3.1. Informace o poloze

Pro uložení údajů o naměřeném signálu je nutné znát přesnou geografickou polohu. Tu lze získávat třemi způsoby - pomocí mobilní sítě, pomocí signálu z dostupných bezdrátových sítí a pomocí GPS. Protože k vytvoření záznamu je potřebná co nejpřesnější poloha, byla pro tuto aplikaci využita poslední z možností. Ta je ovšem velmi náročná na spotřebu energie. V případě že je GPS vypnutá, nebude aplikace správně pracovat.

3.3.2. Připojení k internet

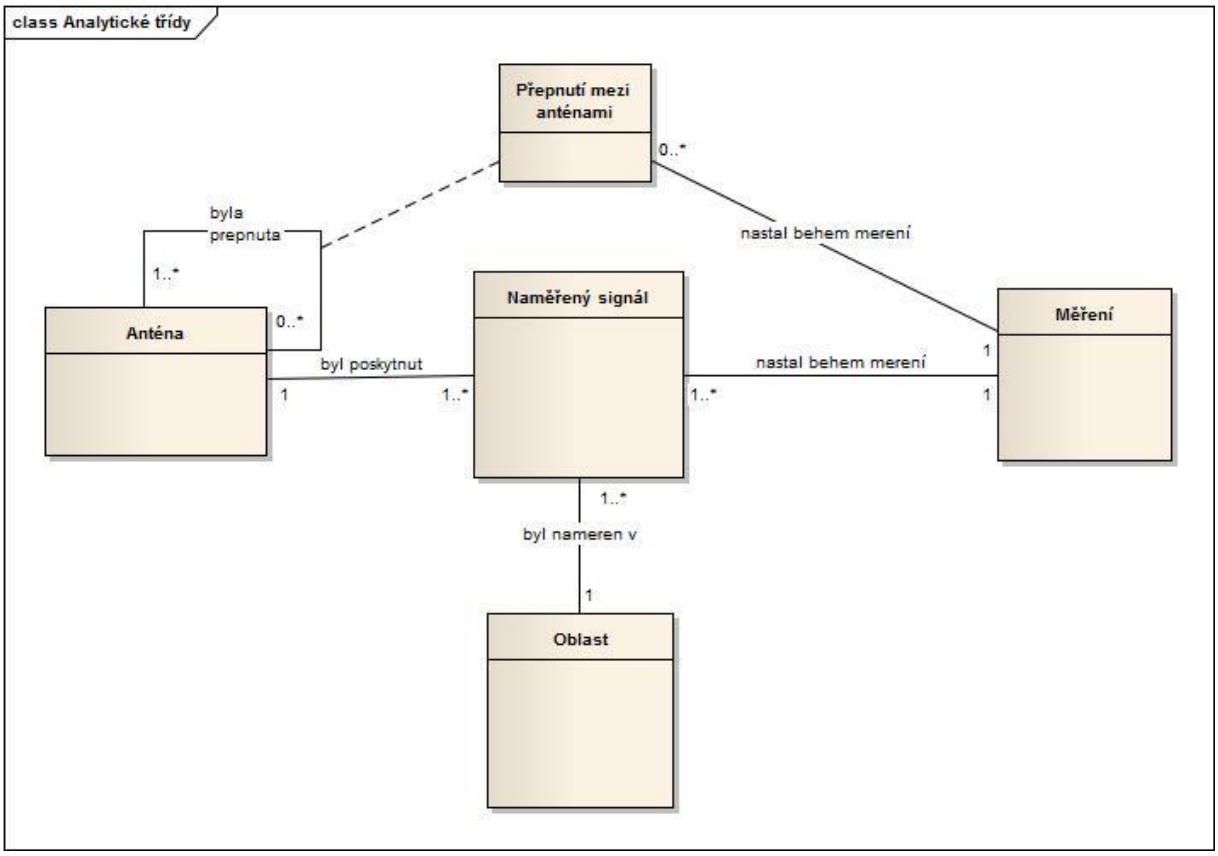
Pro získávání informací z GSMwebu je nutné mít v telefonu zapnuté připojení k internetu, ať už pomocí wifi nebo pomocí mobilních dat. Pokud není připojení k internetu dostupné, nebude možné ukládat data o nových základnových stanicích. Ty co už byly jednou načteny, zůstávají uloženy v databázi zařízení

3.3.3. Objem dat

Objem měřených dat s postupem času značně narůstá. To je problém nejen z pohledu jejich ukládání, ale i zobrazování. Množství dat, se kterými bude aplikace schopná pracovat, by měly ukázat zátěžové testy. Toto chování se může na různých zřízeních lišit v závislosti na velikosti jejich paměti.

3.4. Analytický model tříd

Analytický model tříd popisuje strukturu tříd použitých pro reprezentaci dat, a ukazuje, jak spolu data souvisí. Podrobnější informace o atributech připravované aplikace budou uvedeny dále v textu.



Obrázek 3.1: Doménové třídy

4. Výběr technologií

4.1. Technologie použité pro vývoj

4.1.1. Android

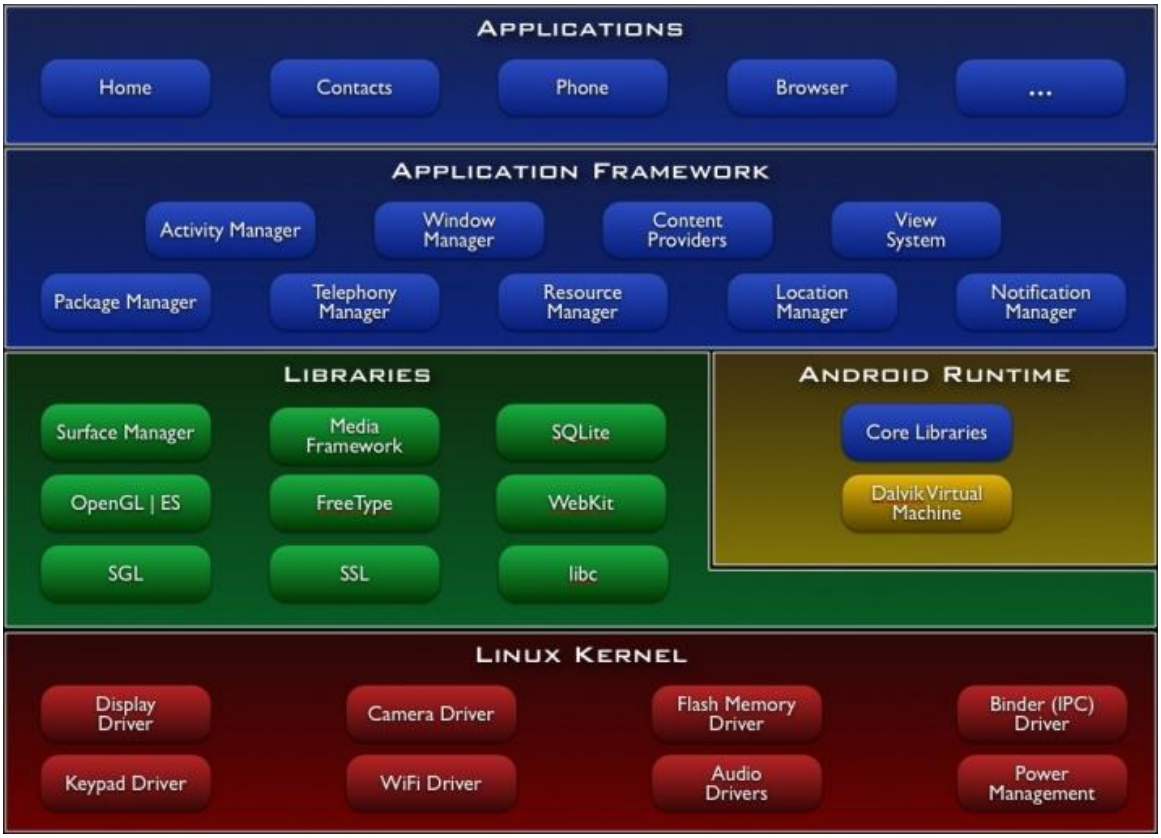
Operační systém Android je založen na jádře operačního systému Linux a v současné době je vyvíjen společností Google. Při vývoji systému byla brána v úvahu omezení, kterými disponují klasická mobilní zařízení, jako výdrž baterie, menší výkonnost a málo dostupné paměti. Zároveň bylo jádro Androidu navrženo pro běh na různém hardwaru. Systém tak může být použit bez ohledu na použitý chipset, velikost či rozlišení obrazovky [4].

4.1.2. Architektura OS Android

Architektura operačního systému Android je rozdělena do 5 vrstev (obr.4.1). Každá vrstva má svůj účel a nemusí být přímo oddělena od ostatních vrstev. Nejnižší vrstva architektury je jádro operačního systému, které tvoří abstraktní vrstvu mezi používaným hardware a zbytkem software ve vyšších vrstvách. Jádro systému Android je postaveno na Linuxu. Je využito jeho mnoha vlastností, jako podpora správy paměti, správa sítí, zabudované ovladače nebo správa procesů. Velmi využívanou vlastností je souběžný běh aplikací, které jsou spuštěny jako samostatné procesy s oprávněním stanoveným systémem. Tato vlastnost přispívá ke stabilitě a také ochraně systému. Naopak systém nepodporuje grafické rozhraní X Window System a ani úplnou sadu GNU knihoven. Důvodem použití jádra Linux bylo také možnost poměrně snadné kompilace pro různá zařízení a tím zaručená přenositelnost.

Android Runtime vrstva obsahuje aplikační virtuální stroj Dalvik, a základní knihovny programovacího jazyka Java. Knihovny se svým obsahem blíží platformě Java Standard Edition. Hlavním rozdílem je nepřítomnost knihoven AWT a Swing pro uživatelské rozhraní, které byly nahrazeny knihovnami uživatelského rozhraní pro Android. Dalším rozdílem je přidání knihoven Apache pro práci se sítí.

Další vrstvou jsou knihovny, které jsou napsány v jazyce C nebo C++ a využívají různé komponenty systému. Tyto knihovny jsou vývojářům poskytnuty prostřednictvím Android Application Framework.



Obrázek 4.1: Architektura OS Android [5]

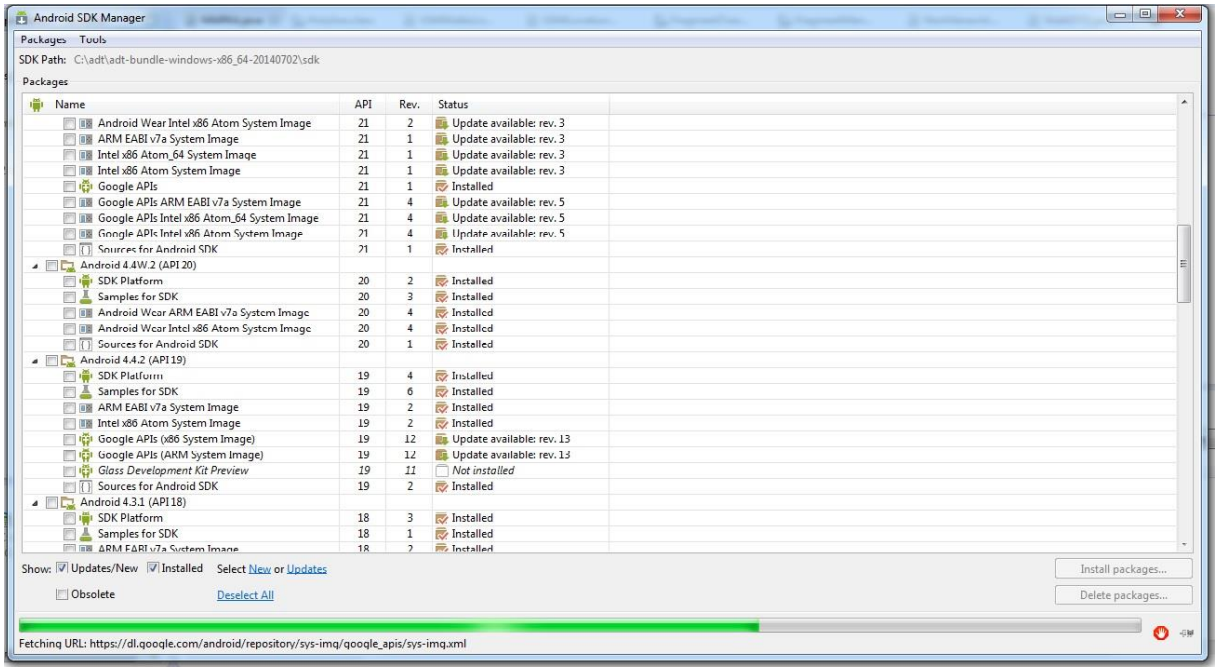
Vrstva Appliaction Framework je pro vývojáře nejdůležitější. Poskytuje přístup k velkému počtu služeb, které mohou být použity přímo v aplikacích. Nejdůležitější součásti jsou uvedeny níže [6].

- Sada prvků View – tyto prvky jsou použity pro sestavení uživatelského rozhraní jako seznamy, textová pole, tlačítka a jiné.
- Content providers – umožňuje přístup k obsahu jiných aplikací, například kontakty.
- Resource Manager – poskytuje přístup ke zdrojů, jako jsou například řetězce, obrázky a textové soubory.
- Notification Manager – umožňuje všem aplikacím zobrazit vlastní upozornění ve stavovém řádku.
- Aktivity Manager – řídí životní cyklus aplikací.

Nejvyšší vrstvu tvoří základní aplikace, která je využívána běžnými uživateli. Může jít o aplikace předinstalované nebo dodatečně stažené. Například e-mailový klient, SMS program, kalendář a další aplikace třetích stran [6].

4.1.3. Eclipse a Android Development Tool

Android Development Tools [7] je zásuvným modulem vývojového prostředí Eclipse [8], který nabízí velkou škálu funkcí pro vývoj softwaru pro platformu Android. Například SDK Manager, který umožňuje spravovat nainstalované balíčky pro jednotlivé verze, správce virtuálních zařízení pro testování a nástroj LogCat [9] jenž usnadňuje práci s logem zařízení. Seznam balíčku pro toto vývojové prostředí je uveden na obrázku 4.2.
V současnosti začíná být více využíváno vývojové prostředí Android Studio [10].



Obrázek 4.2: Adroid SDK Manager

4.1.4. Dalvik

Celý projekt je implementován v jazyce Java, který je pro vývojáře výhodnou kombinací jazyka a platformy. V tomto případě jde o virtuální stroj Dalvik, který byl vyvíjen pro Android teamem společnosti Google. Má registrově orientovanou architekturu, která využívá základních vlastností Linuxového jádra, jako je správa paměti nebo práce s vlákny. Ke vzniku tohoto virtuálního stroje vedly dva důvody. Prvním důvodem byla licenční práva, kdy jazyk Java a jeho knihovny jsou volně šiřitelné, zatímco JVM (Java Virtual Machine) není. Druhým důvodem byla optimalizace virtuálního stroje pro mobilní zařízení a to především v oblasti poměru úspory energie a výkonu.

Dalvik VM je od verze Androidu 5.0 zcela nahrazen Android Runtime [6]. Dalvik VM byl navržen tak, aby mobilní zařízení mohlo efektivně spouštět několik instancí. Každá aplikace v operačním systému Android spouští svůj vlastní proces s vlastní instancí virtuálního stroje. Místo standardních .class souborů jsou zde použity soubory Dalvik Executable (.dex), které jsou optimalizovány tak, aby měly co nejmenší paměťové nároky na zařízení a lépe

vyhovovaly registrově orientované architektuře virtuálního stroje. Tyto soubory jsou vytvořeny ze standardních .class souborů pomocí nástroje dx a poté jsou archivovány do jediného souboru s příponou apk.

4.1.5. Heatmap utility

Pro zobrazování mapy pokrytí je použit nástroj Heatmap utility, který umožňuje vizualizaci rozložení geografických dat na mapě v podobě mapy intenzity (viz obrázek 4.3). Jako datový model využívá datovou strukturu Point Quadtree [11], jenž je modifikací binárního stromu pro reprezentování dvojrozměrných dat. Tvar stromu závisí na pořadí vkládání dat. Tento strom je velice efektivní pro vyhledávání dvojdimenzionálních dat a obvykle funguje se složitostí O(log n). V této aplikaci strom usnadňuje shlukování dat pro generování mapy. Je součástí externí knihovny Utility Library [12] , jež je vyvíjena společností Google. Kód této knihovny je volně přístupný na server Git hub a je možné jej pro potřeby vlastní aplikace upravit.

Obrázek 4.3: Ukázka Heat mapy [12]

4.1.6. Clustering utility

Marker Clustering utility usnadňuje práci s mnoha markery (značkami na mapě) při různých úrovních přiblížení. Když si uživatel prohlíží mapu ve velkém měřítku, jsou markery shluknuty do jediné značky a tím je prohlížení a vyhledávání značek usnadněno [13].

4.1.7. SQLite

SQLite je relační databázový systém obsažený v relativně malé knihovně napsané v jazyku C. Na rozdíl od databázových systémů založených na principu klient-server, kde je databázový server spouštěn jako samostatný proces, je SQLite pouze malou knihovnou, která je propojena k aplikaci pomocí jednoduchého rozhraní. Každá databáze je uložena v samostatném souboru .dbm, kde se data ukládají za použití jednoduchého primárního klíče do stejně velkých bloků. SQLite je součástí jádra prostředí Android. Aplikace implementované v jazyce Java využívají k přístupu k databázi třídy z balíčku `android.database.sqlite`. Tato databáze má několik omezení, jedním z nich je maximum řádků v tabulce (2^{64}), ale tento limit je pro použití v připravované aplikaci nepodstatný, protože takovéto množství dat nelze získat [14].

4.1.8. ORMLite

Objektově relační mapování velice usnadňuje práci s uloženými daty a jejich převádění na instance tříd v programovacích jazycích. Object Relational Mapping Lite (ORMLite) je nenáročný nástroj pro ukládání Java objektů do SQL databáze. Nabízenými funkcemi je ovšem srovnatelný s nástroji Hibernate a iBaits, ale oproti nim nevyžaduje žádnou konfiguraci pomocí souboru xml, nemá žádné závislosti na knihovnách třetích stran (xcerces jaxb etc..) a je velice dobře uzpůsoben pro použití na operačním systému Android [15].

4.1.9. Android plot library

Android plot je API pro tvorbu dynamických i statických grafů vytvořená pro operační systém Android. Umožňuje vytvářet grafy, které zobrazují data dynamicky a snadno se pro ně dá implementovat změna měřítka. Velkou výhodou je výborná integrace s Eclipse, takže téměř každá část grafu může být nakonfigurována pomocí xml [16].

4.2. Geohash

Původně vznikl pro snazší vkládání zeměpisných souřadnic do URL. Lze ho ovšem velmi výhodně využít pro agregování geografický dat v databázi.

Geohash je řetězcová reprezentace pravoúhle ohraničené oblasti vytvořená prokládáním bitů získaných z hodnot zeměpisné šířky a zeměpisné délky. Například souřadnice reprezentované zeměpisnou šířkou N 39.54, W 107.32 budou zařazeny do ohraničené oblasti s Geohash 9x58vy4. Delší řetězce Geohash odpovídají menším souřadnicovým oblastem, této vlastnosti

může být během hashovacího procesu využito k získání oblastí s požadovanou granularitou [17].

Například použití pouze prvních dvou znaků, to znamená 10 bitů, z Geohash řetězce, popisuje oblast o rozměrech přibližně 600x1000 kilometrů. Zvětšením délky řetězce na 4 znaky (20 bitů) se velikost oblasti zmenší na 20x30 kilometrů [17].

Přesnost Geohashe použitá v systému může být uživateli nastavena dle jejich potřeb. Pokud uživatel plánuje uchovávat data spadající do malých oblastí, může využít delších Geohash řetězců. Využití kratších Geohash řetězců je naopak vhodné pro data roztroušená po celém světě. Přesnost lze velmi snadno změnit podle potřeb aplikace.

Velmi výhodné je, že dochází ke značné kompresi dat a velice to usnadňuje výběr dat z databáze a jejich agregování k nějaké oblasti. Místo ověřování souřadnic libovolného bodu proti obdélníku zadanému dvěma klasickými souřadnicemi, stačí ověřit, zda geohash daného bodu začíná stejnou posloupností znaků jako geohash oblasti. Nevýhodou ovšem je, že oblasti se stejnou délkou geohashe nemají stejnou velikost v závislosti na zeměpisné šířce. Toto není tak podstatné u menších oblastí, ovšem při agregování do oblastí v řádech stovek kilometrů by to znamenalo nezanedbatelný problém.

5. Implementace

Vyvíjená mobilní aplikace, slouží převážně k měření síly signálu, proto byla nazvána Sigmet, což je zkratka slov signál a metr. Celá tato aplikace byla vyvíjena v prostředí Android Developer Tool, které nabízí velké množství užitečných funkcí pro vývoj aplikací na platformě Android.

Při implementaci byla využita třívrstvá architektura zahrnující následující vrstvy:

- Aplikační vrstva – představuje vlastní logiku aplikace a umožňuje distribuovat data do ostatních vrstev.
- Prezentační vrstva – obsahuje logiku pro zobrazování dat uživateli.
- Datová vrstva – soustřeďuje logiku získávání a ukládání doménových objektů do databáze.

Všechny tyto vrstvy jsou popsány v následujících kapitolách. Předchází jim popis nastavení aplikace pomocí souboru AndroidManifest.xml.

5.1. Android manifest

5.1.1. Oprávnění

V Android manifestu [18] je nutné definovat omezení přístupu k datům nebo částem kódu na zařízení. Omezení slouží k jasné identifikaci systémových prostředků, které aplikace vyžaduje. Pokud aplikace vyžaduje přístup k systémovým zdrojům chráněným přístupovými právy, je nutné toto deklarovat pomocí elementu `<uses-permission>` v manifestu aplikace. Díky tomu lze během instalace určit, zda bude aplikaci oprávnění přiděleno tím, že se ověří autority, které certifikát podepsaly, v některých případech je dotázán přímo uživatel. Pokud je oprávnění přiděleno, aplikace může libovolně přistupovat k systémovému zdroji. Pokud ovšem přiděleno není, pokus o přístup k systémovému zdroji skončí neúspěšně bez jakéhokoli upozornění.

Seznam všech vyžádaných oprávnění pro aplikaci Sigmet:

```
<permission
      android:name="ca.cvut.sigmet.permission.MAPS_RECEIVE"
      android:protectionLevel="signature" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name=
  "android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name=
  "com.google.android.providers.gsf.permission.READ_GSERVICES" />
<uses-permission android:name=
  "android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name=
  "android.permission.ACCESS_FINE_LOCATION" />
```

```
<uses-permission android:name=
  "android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name=
  "com.android.vending.CHECK_LICENSE" />
```

5.1.2. Klíč pro goole maps

Android Google Maps je balíček, který je součástí knihovny Google Play Services [19] a který může být stažen pomocí nástroje Android SDK Manager.

Pro používání mapy je nejprve nutné se zaregistrovat online pomocí Google Developers Console [20]. Každý projekt, který využívá Google API, musí mít unikátní identifikátor. To umožňuje monitorovat každý požadavek tohoto projektu kvůli sledování vytížení a vyúčtování služeb. Tento klíč musí být uveden v androidManifest.xml (viz níže).

```
<meta-data
    android:name="com.google.android.maps.v2.API_KEY"
    android:value="<vygenerovaný klíč>" />
```

5.1.3. Specifikace verze Android

Každá aplikace musí specifikovat, se kterými verzemi operačního systému Android je kompatibilní. To lze učinit pomocí elemetu `uses-sdk`. Jeho atribut `minSdkVersion` určuje minimální nutnou verzi pro spuštění aplikace a atribut `targetSdkVersion` je verze Android proti které byla aplikace testována.

```
<uses-sdk
    android:minSdkVersion="14"
    android:targetSdkVersion="21" />
```

5.2. Business vrstva

5.2.1. Zpracování událostí v telefonu

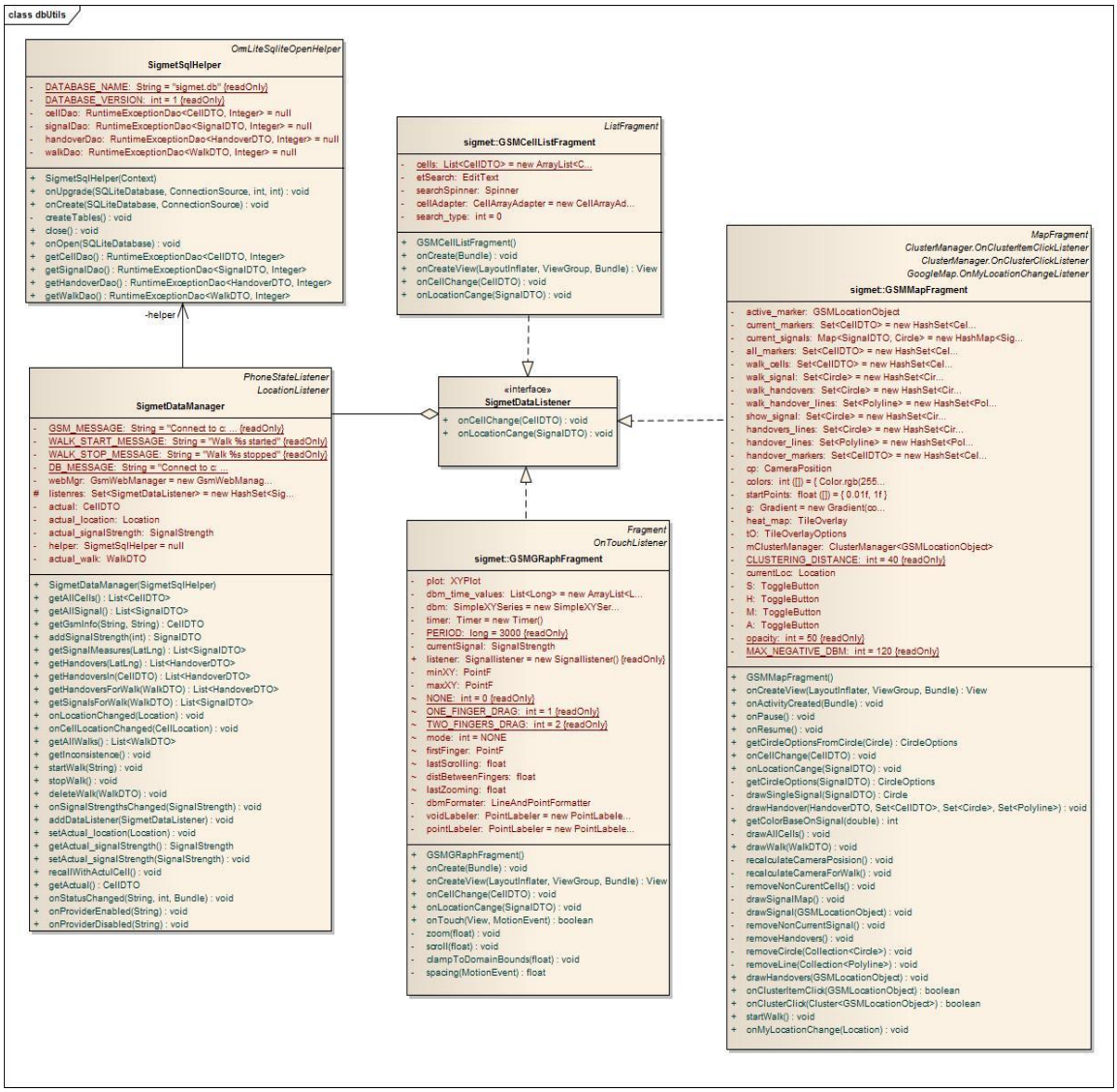
Aplikace Sigmet zpracovává události o změně síly signálu a změně základnové stanice. Centrální komponentou, která tyto události zpracovává je třída `SigmetDataManager`, který také obstarává spouštění všech SQL dotazů. To provádí pomocí Data Access Object (DAO), které jsou mu zprostředkované třídou `SigmetSQLHelper`. K vytváření SQL dotazů je použita třída `QueryBuilder`, která usnadňuje jejich sestavování, a není tak nutné pracovat s dotazy jako s řetězci.

Třída `SigmetDataManager` implementuje rozhraní `LocationListener` a je zaregistrován pomocí metody `requestLocationUpdates` [21] jako observer třídy

`LocationManager` tak, aby zpracovával pouze polohu získanou pomocí GPS. Získávání polohy pomocí signálu Wifi nebo GSM signálu je příliš nepřesné a pro účely aplikace nevyhovující. Dále je nastavena doba ověřování polohy na 2,5 vteřiny a minimální vzdálenost od poslední zaměřené polohy je nastavena na 40 metrů. Tyto hodnoty byly stanoveny jako přiměřené na základě zkušeností získaných během měření, ale v budoucnu by měly být nastaveny uživatelem tak, aby si je mohl sám nastavit podle povahy měření, které provádí.

Třída `SigmetDataManager` je také potomkem třídy `PhoneStateListener` a je zaregistrována do třídy `TelephonyManager` z Android API. Ten umožňuje sledovat změnu úrovně signálu a změnu antény, ke které je telefon připojen.

Aby nedocházelo k závislosti bussines vrstvy na vrstvě prezentační, bylo použito rozhraní `SigmetDataListener` (obr. 5.1). To implementují fragmenty, které zobrazují aktuální údaje.



Obrázek 5.1: Diagram tříd Bussines vrstvy

5.2.2. Parsovaní stránek GSM web

Pokud se údaje o anténě ještě nenachází v databázi zařízení, dojde k jejich stažení ze stránek GSMWeb.cz. K tomu je použit standardní Apache HTTP klient, který je volně k dispozici v systému Android, a není nutné ho do projektu importovat jako externí knihovnu. K získání údajů ze stránky je použita kombinace JDOM z balíčku org.w3c.dom a xPath z balíčku javax.xml.xpath. Vzhledem k tomu, že HTML kód získaný ze stránky GSMWeb.cz není validní xml dokument, bylo nutné provést několik drobných úprav pomocí regulárních výrazů. V níže uvedeném příkladu je ukázána část kódu, který pomocí regulárních výrazů vybere z kódu stránky element tabulky a z jeho potomků poté odstraní nevalidní atributy.

```
Matcher m = Pattern.compile("<table>.*?</table>",
Pattern.DOTALL).matcher(s);
if (!m.find()) {
throw new Exception("cannot find table");
}
s = m.group();
s = s.replaceAll("<tr.*?>", "<tr>");
s = s.replaceAll("<td.*?>", "<td>");
s = s.replaceAll("(<[iI][mM][gG].*?>)", "");
s = s.replaceAll(" ", "");
```

5.3. Datová vrstva

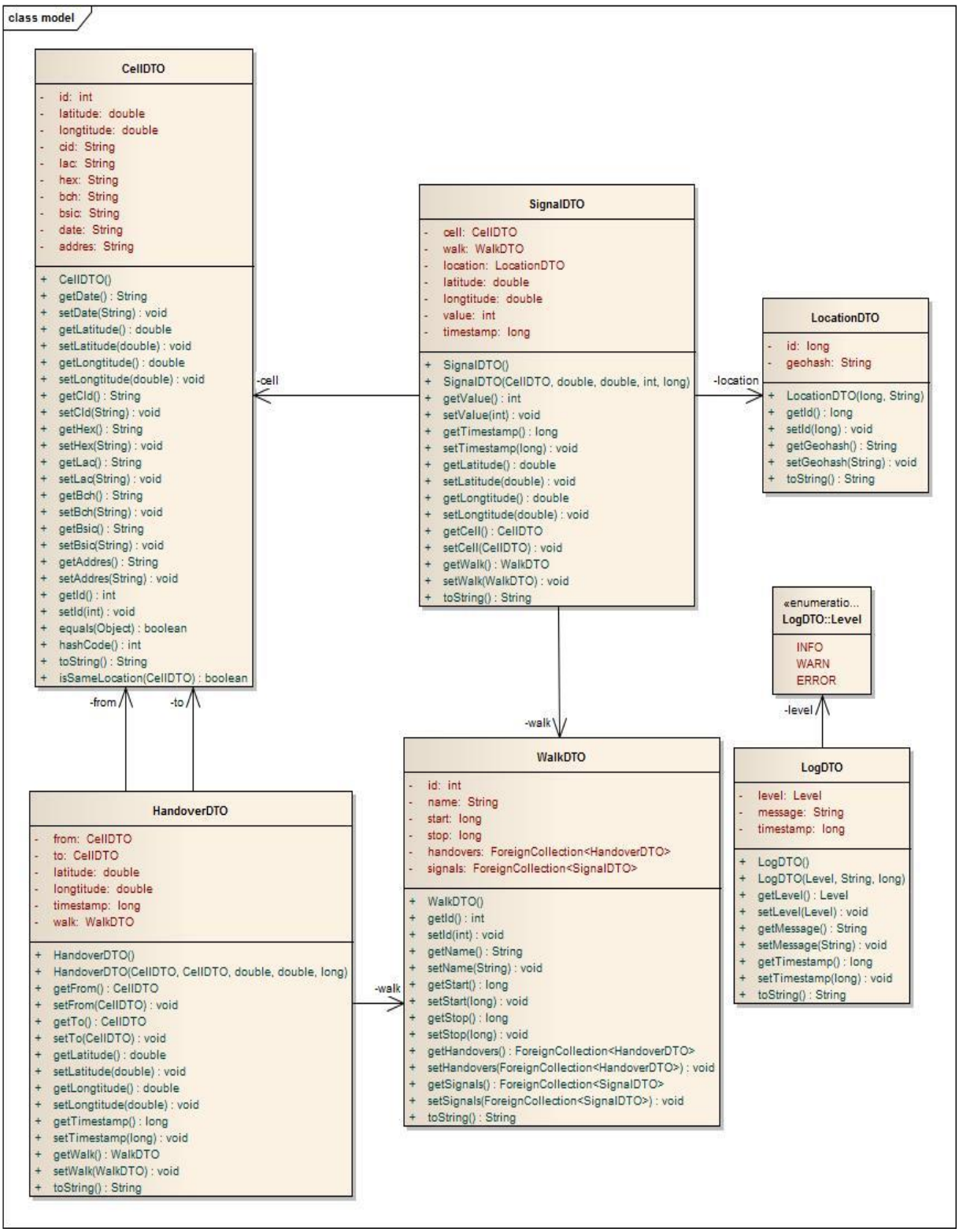
Tato vrstva se stará o propojení aplikace s relační databází SQLite. Toto spojení je realizováno pomocí knihovny ORM Lite (viz kapitola 4.1.6). Anotace ORM lite jsou poté přidány do jednotlivých tříd, které mají být uložené a přístupné pomocí DAO objektů. Tabulky v databázi jsou vytvořeny po spuštění hlavní aktivity (viz kapitola 5.4.1), která je potomkem `OrmLiteBaseActivity`, a kromě životního cyklu Aktivity obstarává i životní cyklus připojení k databázi a aktualizování datového modelu. Vytvoření tabulek je sice iniciované hlavní aktivitou, ale hlavní logika je uložena ve třídě `SigmetSqlHelper`, která je potomkem `OrmLiteSqliteOpenHelper`. Níže je uvedena ukázka kódu, který vytvoří novou tabulku na základě anotací v objektu `CellDTO`. Prvním parametrem je objekt obsahující údaje o databázovém spojení.

```
TableUtils.createTable(connectionSource, CellDTO.class);
```

5.3.1. Doménové třídy

Doménové třídy představují všechna data ukládaná aplikací do databáze. Níže jsou tyto třídy blíže specifikovány.

- CellDTO - představuje anténu, její podrobné údaje, popis a obsahuje její polohu.
- WalkDTO - představuje měření, ke kterému jsou data agregována.
- SignalDTO - je hodnota naměřeného signálu, obsahuje přesný čas a lokaci měření.
- HandoverDTO - zachycuje místo a čas, na kterém došlo k přepojení z jedné antény na jinou.
- LogDTO – Zachycuje popis události, která v aplikaci nastala, například informace o načítání údajů o jednotlivých anténách nebo o chybových stavech, ke kterým v aplikaci došlo.



Obrázek 5.2: Diagram doménových tříd

5.3.2. Použité anotace nástroje ORMLite

Níže je uveden seznam anotací, ve kterém je vysvětleno jejich použití v aplikaci.

Tabulka 5.1 Použité anotace [22]

@DatabaseField	Definuje to, jak bude atribut třídy uložen. Níže jsou některé její parametry: • CanBeNull – určuje, zda anotovaný atribut může být prázdná hodnota (Null). • ColumnName – název sloupce v tabulce, pokud není nastaven, je jako název sloupce použit název atributu. • Foreign – udává, že atribut je netrivialní object, který je taktéž perzistován pomocí ORMLite, v databázi je pak vytvořen sloupec s názvem atributu prodloužený o koncovku _id, která odkazuje na existující řádek jiné tabulky. • ForeignAutoCreate - umožňuje vytvořit vnořený objekt, pokud pro něj v databázi ještě neexistuje příslušný řádek. • ForeignAutoRefresh – objekt je automaticky obnovován. • GeneratedId – vytvoří unikátní hodnotu atributu. Pozor, pokud je nějaká hodnota z tabulky smazána, může být použita znovu. To znamená, že pokud nedojde ke smazání této hodnoty v řádcích, kde je použita jako cizí klíč, vytvoří se chybná vazba na jiný objekt. Proto je lepší pro generování klíčů využívat sekvence. • GeneratedIdSequence – při vytvoření tabulek v databázi vytvoří sekvenci, která je použita pro vyplnění hodnoty tohoto atributu. To je velmi výhodné pro generování unikátních klíčů. • Index – určuje, zda nad tímto atributem bude vytvořen index. • UseGetSet – pro získání tohoto atributu budou použity metody get a set, to je výhodné, pokud je v nich umístěna nějaká další logika. • UniqueIndex – zajišťuje, že atribut v databázi bude mít jedinečnou hodnotu a že pro tento sloupec bude vytvořen index.
@DatabaseTable	Definuje, že třída bude uložena. Níže jsou uvedeny některé její atributy: • DaoClass – Třída data acces objektu, pro tuto třídu. • TableName – jméno tabulky, do které bude třída uložena.

@ForeignCollectionField	Definuje, že atribut je kolekce cizích objektů vytvořených relací 1:N. Níže jsou uvedeny některé její parametry: • ColumnName – jméno sloupce použitého pro vazbu. • Eager – kolekce je zaplněna automaticky ve chvíli, kdy je objekt načten z databáze, pomocí jakéhokoli dotazu. • MaxEagerLevel – určuje, do jaké hloubky bude docházet k automatickému načítání cizích kolekcí.

5.3.3. Životní cyklus DAO objektů

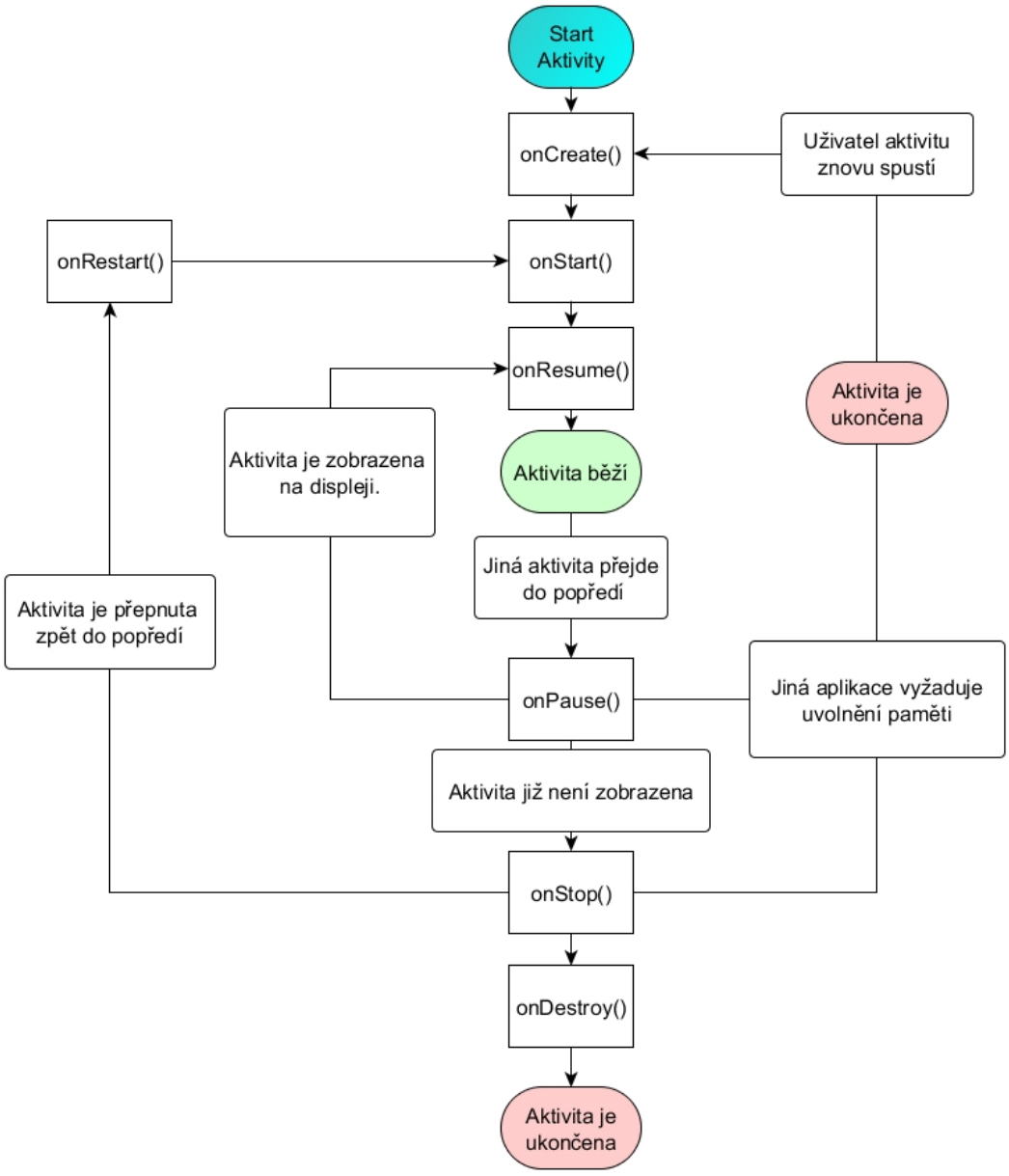
Přístup k DAO pro jednotlivé třídy je spravován třídou `SigmetSqlHelper`. Ta se stará o inicializaci těchto objektů při připojení k databázi a o jejich destrukci v případě odpojení databáze.

5.4. Prezentační vrstva

Do této vrstvy patří všechny třídy, které obstarávají interakci s uživatelem. V případě aplikace Sigmet je to jediná aktivita, která spravuje životní cyklus několika fragmentů (viz kapitola 5.4.2.), mezi nimiž lze přepínat pomocí menu.

5.4.1. Aktivita

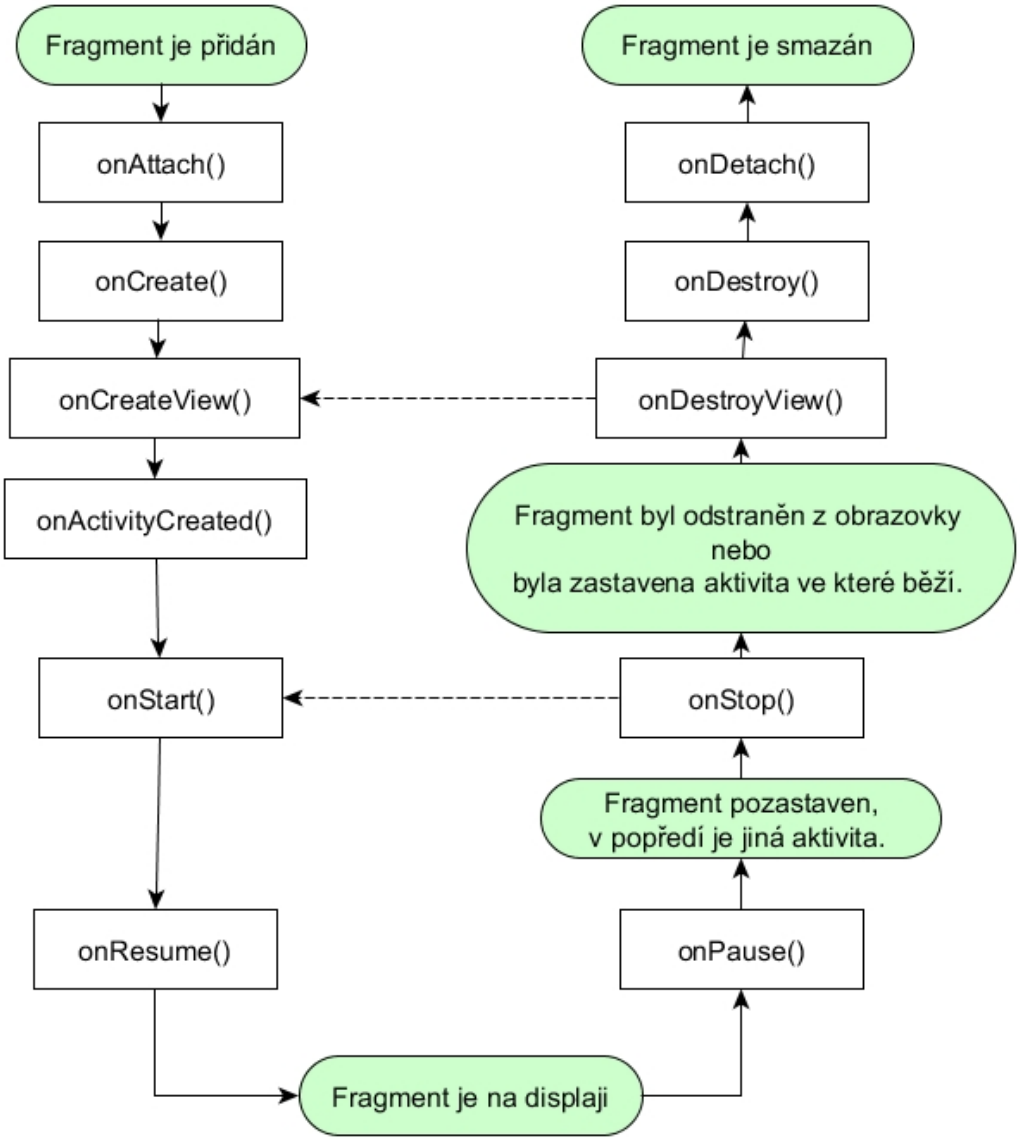
Aktivita se v průběhu svého života může nacházet ve třech základních stavech - může být na popředí a mít uživatelský vstup, dále může být pouze viditelná (třeba jen částečně) nebo může být pozastavená na pozadí. Při přechodu aktivity mezi stavy jsou volány systémová zpětná volání (callbacky). Tato volání vymezují jednotlivé stavy aktivity a definují její životní cyklus. Toho je využito při ukládání a obnovování stavu této aplikace, která je atypická v tom, že měření musí probíhat i když aktivita není právě zobrazena na displeji.
Na níže uvedeném schématu jsou zobrazeny všechny fáze životního cyklu aktivity a metody, které jsou při přechodu mezi jednotlivými stavy volány.



Obrázek 5.3: Životní cyklus aktivity [23]

5.4.2. Fragment

Jak již bylo řečeno výše, fragmenty jsou v aplikaci využity ke správě životního cyklu jednotlivých obrazovek. Jejich stavy jsou poměrně úzce navázány na životní cyklus aktivity, přesto zde jsou jisté odlišnosti. Pokud je zavolána metoda `onPause()` na aktivitě, ve které se fragment nachází, následuje volání metody `onPause()` na fragmentu. Toto platí i pro metody `onDestroy()` a `onCreate()`.



Obrázek 5.4: Životní cyklus fragmentu [24]

5.4.3. Asynchronní zpracování IO operací

Operační systém Android obsluhuje všechny vstupní události vyvolané uživatelem v jediném vlákně, tomuto vláknu se říká hlavní vlákno (main thread). Při tvorbě aplikací mělo být kvůli zachování rychlé odezvy dlouho trvající operace, například vstupně výstupní operace nebo komunikace se vzdáleným serverem, použito asynchronní zpracování. Pokud dojde k zablokování hlavního vlákna na dobu delší než 5 sekund, tak Android vyvolá *Application not responding* dialog. Z tohoto dialogu může uživatel aplikaci ukončit.

Android nabízí několik možností jak tohoto dosáhnout. Lze použít třídy `Loader`, `Handler` nebo `AsyncTask`.

Následující úryvek kódu demonstruje načtení údajů o všech anténách z databáze pomocí asynchronní úlohy reprezentované třídou `AsyncTask`, ta byla vybrána z důvodu nenáročné implementace. Tato třída je parametrizována pomocí generic, kde první je typ vstupních parametrů pro metodu `doInBackround`, druhý představuje typ parametrů pro metodu `onProgress` a třetí je typ parametrů pro metodu `onPostExecute`. Načtení je provedeno v metodě `doInBackground(Void... params)`, která je spuštěna mimo hlavní vlákno, po jejím ukončení je v hlavním vlákně vyvolána metoda `onPostExecute(List<CellDTO> result)`, ve které jsou výsledky zobrazeny na mapě.

```
AsyncTask<Void, Void, List<CellDTO>> getAll =
      new AsyncTask<Void, Void, List<CellDTO>>() {

      @Override
      protected List<CellDTO> doInBackground(Void... params) {
            try {
                  return SigmetActivity.dataManager.getAllCells();
            } catch (SQLException e) {
                  SigmetLogger.error(e.getMessage());
                  cancel(true);
            }
            return Collections.emptyList();
      }

      @Override
      protected void onPostExecute(List<CellDTO> result) {
            result.removeAll(current_markers);
            result.removeAll(walk_cells);
            all_markers.addAll(result);
            Collection<GSMLocationObject> newLycCreated =
      GSMLocationObject.addAll(result);
            mClusterManager.addItems(newLycCreated);
            mClusterManager.cluster();
      }
};
getAll.execute((Void)null);
```

5.4.4. Mapa

Mapa je implementována pomocí GoogleMap fragmentu. Jsou na ní zakresleny údaje o jednotlivých anténách, pro přehlednost shluknuté podle úrovně zvětšení do clusteru, který lze rozkliknout a tím dojde k přiblížení a rozpadu na jednotlivé značky, které reprezentují stožáry, na nichž jsou antény umístěny.

Na obrázku 5.5 je zobrazena mapa pro prohlížení měření. Každý marker představuje jednu základnovou stanici. Jednotlivé funkcionality jsou popsány níže.

1) Po stisku tlačítka START je zobrazen dialog pro zahájení měřené

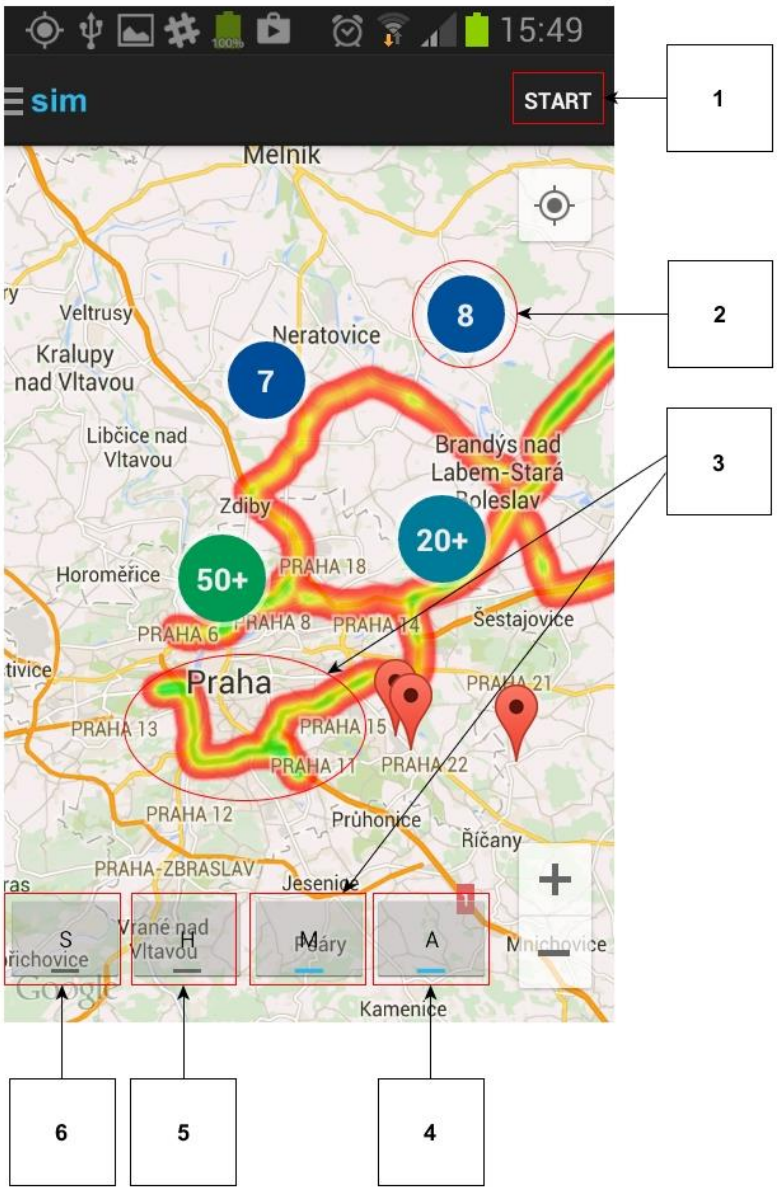
2) Antény jsou pro přehlednost shromážděny do shluků, po kliknutí na tento shluk dojde k upravení pozice kamery a rozpadu shluku na menší shluky nebo jednotlivé značky

3) Po stisknutí tlačítka M je vygenerována mapa kvality signálu ze všech naměřených hodnot

4) Po stisknutí tlačítka A dojde k zobrazení všech antén. Opětovným stiskem jsou antény opět skryty a na mapě zůstanou zobrazeny jen antény z aktuálního měření

5) Po stisknutí tlačítka H jsou po kliknutí na marker zobrazeny všechna přepojení spojená s označeným umístěním

6) Po stisknutí tlačítka S jsou po kliknutí na značku zobrazeny všechna měření signálu spojená s tímto umístěním



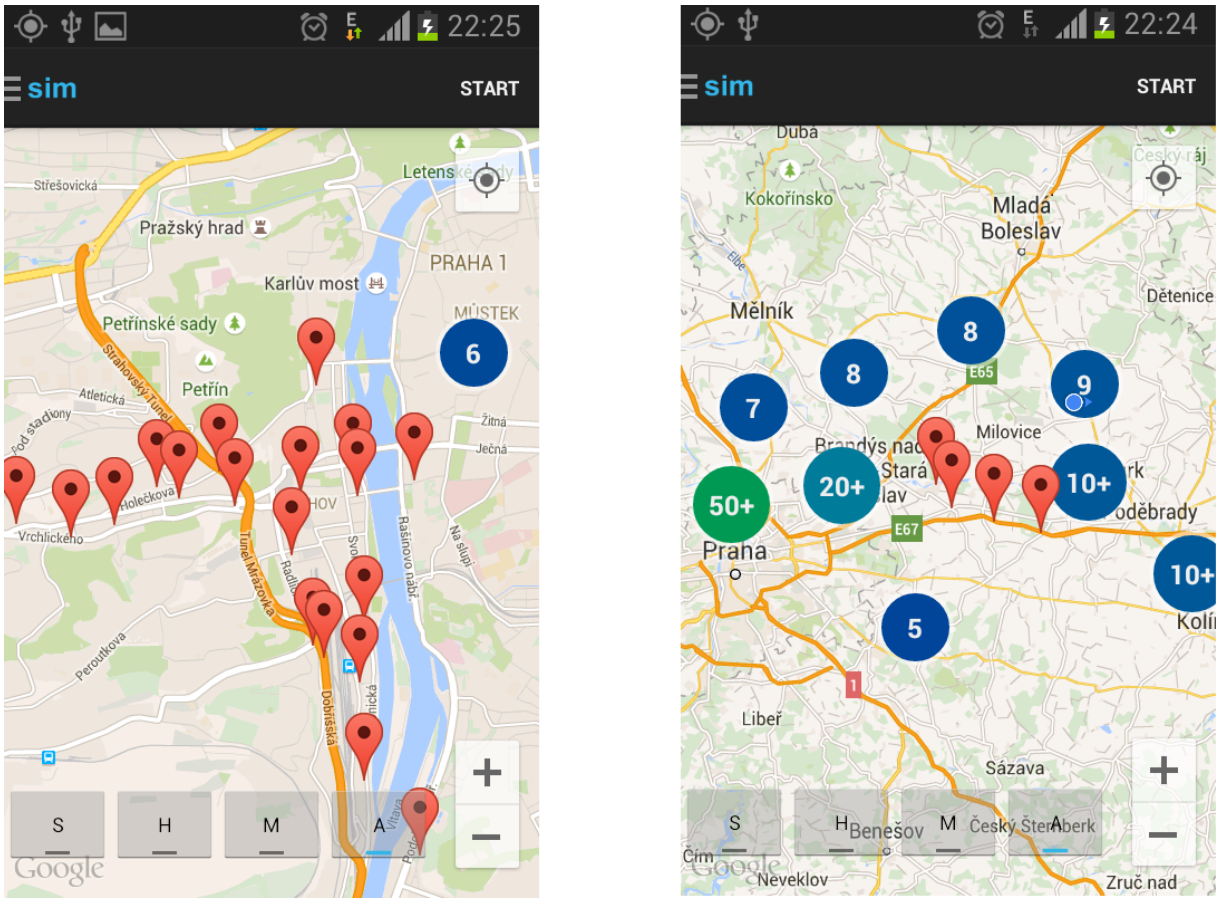
Obrázek 5.5: Mapa pro prohlížení měření

5.4.5. Heat mapa

Pro použití v připravované aplikaci Sigmet byl upraven algoritmus generování bitmapy tak, aby popisoval charakteristiku signálu na mapě. Původní algoritmus zachycoval spíše hustotu rozložení bodů, což bylo pro zobrazování měření signálu nevhodné. Efektu tepelné mapy je knihovnou dosaženo pomocí konvoluce. Protože parametry pro generování konvolučního jádra nebylo možné nastavit veřejnými metodami, byly upraveny přímo v kódu knihovny.

5.4.6. *Shlukování značek na mapě*

Při zobrazování velkého množství značek na mapě, které reprezentují rozmístění vysílačů, dochází k jejich překrývání, obzvláště pokud jsou nahromaděny v malé oblasti. Proto byl v aplikaci Sigmet použit nástroj Google Maps Android Marker Clustering Utility pomáhající zvýšit přehlednost zobrazení značek na mapě při různých úrovní přiblížení (obr. 5.6). Tento nástroj je také součástí knihovny Utility library a byl pro účely aplikace Sigmet mírně upraven.



(a) (b)

Obrázek 5.6: Zobrazení shluků značek při (a) velkém a (b) malém přiblížení.

5.4.7. *Graf*

Graf aktuálně naměřeného signálu je implementován pomocí knihovny Android Plot a umožňuje měnit měřítko osy X, kde jsou uvedeny časové údaje o měření.

5.4.8. Seznam antén

Tato obrazovka zobrazuje prostý seznam všech antén, které jsou v aplikaci již zachyceny. Pomocí jednoduchého input boxu zde lze v tomto seznamu vyhledávat pomocí různých kritérií.

5.4.9. Seznam měření

Seznam všech měření vytvořených uživatelem umožňuje pomocí kontextového menu zobrazit všechny naměřené údaje na mapě, měření smazat nebo exportovat ve formátech xml a csv.

5.4.10. Prohlížeč měření

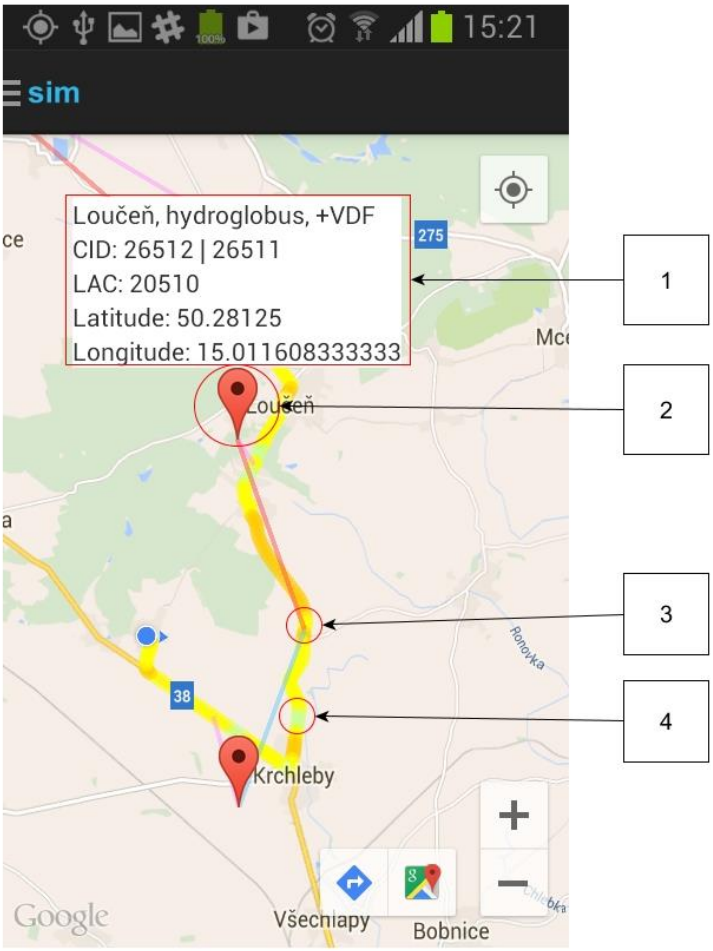
Prohlížeč měření je upravená mapa sloužící pouze k prohlížení uložených měření mající níže uvedené funkcionality, což je vyobrazeno na obrázku 5.7.

1) Tooltip je zobrazen po kliknutí na značku. V řádku CID jsou vidět čísla všech antén umístěných na tomto místě. LAC je Location Area Code přidělený tomuto místu.

2) Značka znázorňující umístění antén

3) Znázornění přepojení mezi anténami. Růžová čára znázorňuje původní anténu, modrá směřuje k nově připojené anténě

4) Značka znázorňující hodnotu naměřeného signálu, červená znamená úroveň kolem -90dbm, světle zelená pak úroveň kolem -10dbm



Obrázek 5.7: Mapa prohlížení měření

6. Testování

Testování aplikace činí podstatnou část jejího životního cyklu. Účely testování jsou v různých etapách vývoje odlišné. Pro každou funkcionalitu vznikají nejprve jednotkové testy, jejichž účelem je odhalit co nejvíce chyb v kódu a měly by vznikat průběžně s kódem samotným, nebo dokonce ještě před ním. Dobrým pokrytím kódu testy se zrychlí vývojový cyklus, protože k odhalení chyb dojde téměř okamžitě. Na druhou stranu je nutné si uvědomit, že sto procentní pokrytí kódu testy nezajistí její bezchybnost a nezbaví nás nutnosti provádět testy integrační a manuální.

V dalších etapách jde hlavně ověření toho, že aplikace dělá to, co bylo se zákazníkem dohodnuto. V tomto případě přicházejí na řadu akceptační testy, tedy testování zákazníkem. Tento druh testování probíhal pro účely aplikace Sigmet při konzultacích s vedoucím práce, na kterých byly demonstrovány nově doplněné funkcionality

6.1. Jednotkové testy

Pro jednotkové testy byl pomocí nástroje Android Development Tools vytvořen samostatný projekt, který umožňuje spouštění testů přímo na zařízení nad nainstalovanou aplikací. Samotná implementace testů je pak realizována pomocí Framework Unit. Bohužel pro framework ORM Lite neexistuje žádný nástroj, který by umožňoval nastavení stornování transakce (rollbacku) pro všechny testy, takže bylo nutné řídit transakce ručně.

6.2. Manuální testy

Manuální testy byly nedílnou součástí při vývoji této aplikace. Měření probíhalo přímo v terénu, a to jak ve městech, tak i v řidčeji osídlených oblastech s horším pokrytím signálu. Testovalo se při chůzi, jízdě autem, tramvají, vlakem, dokonce i ultralehkým letadlem.

6.3. Automatizované testování

K automatizovanému testování byl využit Framework Robotium [25], který je určen pro testování různých případů užití. Je možné pomocí něj simulovat uživatelské chování nadefinováním posloupnosti klikání na daná místa na obrazovce. Testování pomocí frameworku Robotium je velice podobné práci s frameworkem Selenium, který slouží výhradně pro testování webových stránek přímo v prohlížeči. Samotné testy jsou strukturou kódu velmi podobné testům vytvořeným pomocí knihovny jUnit. Hlavními třídami pro vývoj testů je třída `Solo`, pomocí jejíž instance je možné obsluhovat prvky GUI jako kontextová

menu, dialogy a input boxy. O životní cyklus aktivity během testu se stará třída `ActivityInstrumentationTestCase2`, která vytváří testovanou aktivitu za využití systémové infrastruktury.

6.4. Akceptační testy

Pokud všechny předchozí testy proběhly bez nedostatků, je možné spustit akceptační testy. Scénáře těchto testy by měly být sestaveny na základě požadavků zákazníka před zahájením vývoje. Test musí být navržen tak, aby šlo jednoznačně rozhodnou o jeho výsledku a tak, aby množina možných stavů výsledků byla co nejmenší.

6.4.1. Kontrola načtení stanice

Popis: Uživatel chce znát základnové stanice ve svém okolí.

Testovací data: Vzhledem k povaze aplikace není možné vytvořit data uměle. Test by měl probíhat v několika vybraných lokacích.

Očekávané chování: Data nasbíraná po několika hodinách v terénu jsou porovnána se stránkou GSMWeb, včetně správného zakreslení do mapy.

6.4.2. Kontrola funkčnosti vizualizace měřeného signálu

Popis: Uživatel chce pozorovat změny síly signálu vzhledem ke svojí poloze

Testovací data: Test lze provádět pouze v terénu. Pro ověření je potřeba mít nějaký konkurenční produkt se schopností měřit sílu telefonního signálu s velkou přesností.

Očekávané chovaní: Naměřené údaje budou odpovídat skutečným. I při různých nastaveních měřítka grafu

6.4.3. Kontrola zobrazení zaznamenaného měření

Popis: Uživatel chce zobrazit naměřená data

Testovací data: Jakákoliv nasbíraná a ověřená data.

Očekávané chování: Na mapě jsou zobrazeny všechny základnové stanice ke kterým byl telefon během měření připojen, dále jsou zobrazeny všechny údaje o naměřené síle signálu a všechny místa kde došlo k přepojení mezi věžemi.

6.4.4. Kontrola exportu do CSV

Popis: Uživatel chce vyexportovat nasbíraná data do souboru CSV

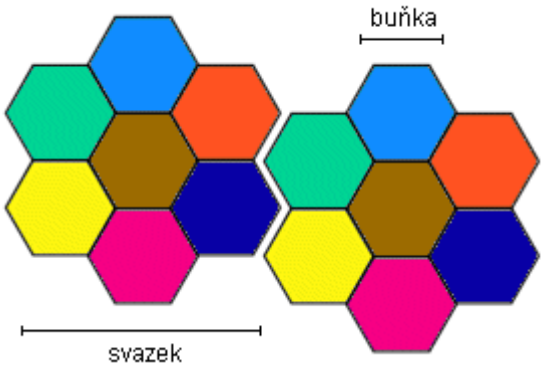
Testovací data: Jakákoliv nasbíraná a ověřená data

Očekávané chování: CSV soubor obsahuje všechna nasbíraná data.

7. Provedená měření

7.1. Sledované údaje v mobilní síti

Celulární systém je struktura založená na určitém obrazci (obr 7.1), ve kterém je definované rozložení dostupného kmitočtového přídělu. Tento obrazec je pak možné zapojit tak, že jeho opakováním lze teoreticky pokrýt nekončící území, neboť buňky do sebe zapadají a vzájemně na sebe navazují. Tyto se skládají z BTS (Base Transceiver Station).



Obrázek 7.1: Buňková struktura [26]

7.2. Jednotky použité pro měření

Síla signálu je v aplikaci měřena pomocí jednotek zvaných Decibel-milliwat, obvyklá zkratka je dBm nebo dBmW. Tato jednotka je hojně využívána pro měření signálů, pro svoji schopnost zachytit, jak vysoké, tak i velmi nízké hodnoty jediným celým číslem. Na rozdíl od jednotky decilbell (dB), která je jednotkou bezrozměrnou, vyjadřující pouze poměr mezi dvěma hodnotami, je dBm jednotka absolutní, určená pro měření hodnot s velkým rozptylem. Jednotka dBm je popsána vzorcem 7.1, kde P je výkon.

$$x = 10log_{10}\ \frac{P}{1mW} \tag{7.1}$$

Síla signálu v běžné GSM síti se pohybuje kolem 30 až -111 dBm.

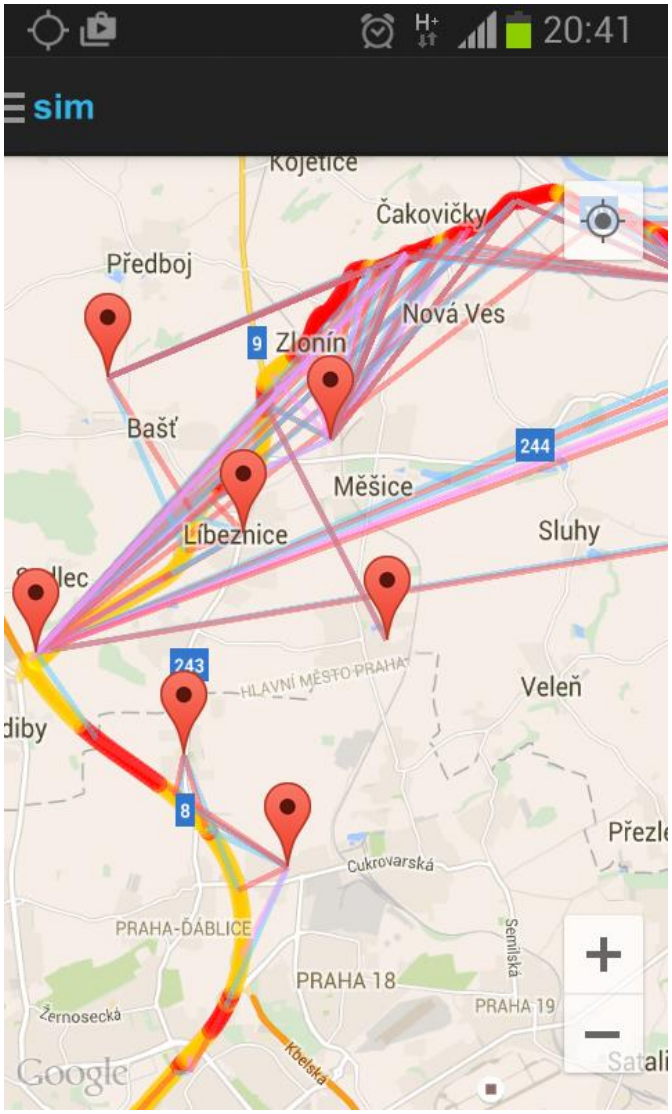
ASU (Arbitrary Strength Unit) je celočíselná hodnota odpovídající síle signálu měřeného mobilním telefonem. ASU lze převést na dBm následujícím vzorcem.

$$dBm = 2 \times ASU - 113 \tag{7.2}$$

7.3. Naměřené hodnoty

V rámci testování bylo provedeno i několik měření, která měla přibližně určit tvar oblasti pokryté jednou směrovou anténou. To se ukázalo jako téměř nesplnitelný úkol. Protože anténa telefonu není příliš přesná a pouhým otočením telefonu může dojít k jeho přepojení k jiné anténě. ro jedno z měření bylo použito dokonce letadlo, které s aplikací kroužilo kolem jedné věže. Ovšem i zde docházelo k poměrně nahodilému přepojování mezi okolními věžemi a tvar pokryté oblasti nebylo možné určit.

Obrázek 7.1 znázorňuje extrémní četnost přepojování mezi několika věžemi z téhož místa. Je zde možné vidět, že jednotlivé spojení se překrývají a takový výsledek měření je velmi nepřehledný



Obrázek 7.2: Ukázka provedeného měření s vysokou četností přepojování mezi jednotlivými věžemi

8. Závěr

V rámci této práce byla navržena a implementována snadno a intuitivně ovladatelná aplikace. K jejím silným stránkám patří zobrazování mapy signálu, které vypadá velice efektně a poskytuje zajímavou vizualizaci naměřených dat. Dále také oproti podobným aplikacím umožňuje ukládání a prohlížení jednotlivých měření. Což je přínosné zejména při jejím vědeckém a technickém použití pro diagnostiku mobilních síti. Aplikace je užitečná pro vizuální analýzu chování sítě v reálných podmínkách a korelaci s polohou.

Aplikace byla důkladně testována na několika úrovních, Nejvíce smysluplné je použití na malé ploše. Nicméně měření je limitováno kvalitou antény mobilního telefonu.

Aplikace využívá agregování dat, vzhledem k tomu, že při generování mapy pokrytí ze všech naměřených údajů docházelo k problémům s nedostatkem paměti, která byla aplikaci přidělena. Toto by bylo možné dále vylepšit ukládáním vypočítaných bitmap pro jednotlivé oblasti v databázi a průběžnou aktualizací na pozadí za běhu aplikace. Při zobrazení by pak nemuselo docházet k jejich výpočtu a tím by se podstatně snížily nároky na operační paměť.

Aplikace nemůže prozatím konkurovat komerčním aplikacím v této oblasti a to zejména kvůli absenci centrálního uložiště, které by umožňovalo získávat data od všech uživatelů a tím vytvořit detailní mapu pokrytí. Její předností je ovšem možnost zobrazování signálu jednotlivých antén a do budoucna se nabízí velké množství možností pro další rozšiřovaní. Jako nejpřínosnější se jeví přidání serverové části, která by mohla být nasazena na některé z nízkonákladových platforem. Možnost naměřená data sdílet by podstatně zvýšila přínos této aplikace. Další velkou výhodou tohoto řešení by bylo snížení nároků na zařízení, kdy výsledné bitmapy teplené mapy by místo toho, aby byly pokaždé vypočítávány na mobilním zařízení, byly vytvořeny podstatně výkonnějším serverem a do mobilního zařízení pouze staženy a poté průběžně aktualizovány. Serverová část by zprostředkovávala i data o základnových stanicích pomocí nějakého API, což by podstatně snížilo objem přenášených dat při získávání těchto informací. Dalším krokem by mělo být i testování na větším počtu zařízení například pomocí aplikace Testdroid [27]. Zajímavé by bylo shromažďovat i jiné údaje než pouze sílu signálu, například dobu odezvy a chybovost přenášených dat.

Literatura

1. **APK, Android.** NetMonster APK. [Online] [Citace: 15. květen 2015.] http://id.apk.downloadatoz.com/NetMonster,24666.html.

2. **OpenSignal, Inc.** *Open Signal.* [Online] [Citace: 10. květen 2015.] http://opensignal.com/blog/2013/01/23/launching-opensignal-2/.

3. Network Signal Info V1.2.3 Screenshots . *Phoneky.* [Online] 20. květen 2015. http://phoneky.com/android/?p=preview&id=d1d3827&st=3.

4. **Meier, Reto.** *Professional Android 4 Application Development.* Indianapolis : John Wiley & Sons, Inc., 2012. 978-1118102275.

5. **Google.** Architektura OS Android. *Android Developers.* [Online] [Citace: 30. duben 2015.] http://developer.android.com/images/system-architecture.jpg.

6. **Burian, Pavel.** *Internet inteligentních aktivit.* Praha : Grada, 2014. 978-80-247-5137-5.

7. **Google.** Android Developers. *Android Developer Tools.* [Online] [Citace: 10. červen 2014.] http://developer.android.com/tools/help/adt.html.

8. **IBM.** *Eclipse.* [Online] [Citace: 2. červen 2014.] https://eclipse.org/.

9. **Google.** logcat. *Android Developers.* [Online] [Citace: 12. červen 2014.] http://developer.android.com/tools/help/logcat.html.

10. —. Android Studio Overview. *Android Dvelopers.* [Online] [Citace: 28. duben 2015.] http://developer.android.com/tools/studio/index.html.

11. **Wikipedie, Přispěvatelé.** *Quadtree.* [Online] [Citace: 5. duben 2015.] http://en.wikipedia.org/wiki/Quadtree.

12. **Google.** Google Maps Android Heatmap Utility. *Google Developers.* [Online] Google. [Citace: 3. Květen 2015.] https://developers.google.com/maps/documentation/android/utility/heatmap.

13. —. Google Maps Android Marker Clustering Utility. *Android Developers.* [Online] [Citace: 5. červenec 2014.] https://developers.google.com/maps/documentation/android/utility/marker-clustering.

14. About SQLite. *SQLite.* [Online] [Citace: 5. březen 2015.] https://www.sqlite.org/about.html.

15. OrmLite - Lightweight Object Relational Mapping (ORM) Java Package . *OrmLite.* [Online] [Citace: 15. březen 2015.] http://ormlite.com/.

16. **AndroidPlot.** *AndroidPlot.* [Online] [Citace: 3. únor 2015.] http://androidplot.com/.

17. *Expressive Query Support for Multidimensional Data in Distributed Hash Tables.* **Malensek, Matthew a Lee Pallickara, Sangmi and Pallickara, Shrideep.** Chicago : Institute of Electrical and Electronics Engineers, 2013. 9781467344326.

18. **Google.** App Manifest. *Android Developers.* [Online] [Citace: 2. červenec 2014.] http://developer.android.com/guide/topics/manifest/manifest-intro.html.

19. —. *Google Play.* [Online] [Citace: 1. březen 2015.] https://play.google.com.

20. —. *Google Developers Console.* [Online] Google. [Citace: 3. červen 2014.] https://console.developers.google.com/project.

21. —. *Android Developers.* [Online] LocationManager. [Citace: 5. březen 2015.] http://developer.android.com/reference/android/location/LocationManager.html.

22. ORMLite Core 4.48 API . *ORMLite.* [Online] [Citace: 20. duben 2015.] http://ormlite.com/javadoc/ormlite-core/.

23. **Google.** Activity. *Android Developers.* [Online] [Citace: 1. květen 2015.] http://developer.android.com/reference/android/app/Activity.html.

24. —. Fragment. *Android Developers.* [Online] [Citace: 1. květen 2015.] http://developer.android.com/reference/android/app/Fragment.html.

25. —. User scenario testing for Android. *Google Code.* [Online] [Citace: 5. duben 2015.] https://code.google.com/p/robotium/.

26. **Richtr, Tomáš.** *Technologie pro mobilní komunikaci.* [Online] [Citace: 15. květen 2015.] http://tomas.richtr.cz/mobil/bunk-princip.htm.

27. **Bitbar.** *Testdroid.* [Online] [Citace: 20. květen 2015.] http://testdroid.com/.

Příloha A

Seznam použitých zkratek

API - Application Programming Interface

ASU - Arbitrary Strength Unit

BTS - Base Transceiver Station

CID – Cell Identity

CSV - Comma Separated Values

DAO - Data Access Object

DTO - Data transfer object

GPS - Global Positioning System

GSM - Global System for Mobiles

GUI - Graphical User Interface

HTML - HyperText Markup Language

HTTP - Hypertext Transfer Protocol

IO - Input/Output

JVM - Java Virtual Machine

LAC - Location Area Code

ORM - Object Relational Mapping

OS - Operating System

SDK - *S*oftware Development Kit

SQL - Structured Query Language

XML - Extensible Markup Language

Příloha B

Obsah přiloženého CD

Složka	**Obsah**
Text	Text samotné práce ve formátech docx a pdf
Source	Zdrojový kód aplikace
	Upravený zdrojový kód knihovny Utility Library
	Projekt s testy
Instal	Instalační balíček aplikace Sigmet pro Android