ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

Fakulta elektrotechnická

Katedra telekomunikační techniky

Parametry okolních buněk

květen 2015

Bakalant: **Matěj Korych**

Vedoucí práce: **Ing. Pavel Bezpalec, Ph.D.**

Čestné prohlášení

Prohlašuji, že jsem zadanou bakalářskou práci zpracoval sám s přispěním vedoucího práce a konzultanta a používal jsem pouze literaturu v práci uvedenou. Dále prohlašuji, že nemám námitek proti půjčování nebo zveřejňování mé bakalářské práce nebo její části se souhlasem katedry.

Datum: 22. 5. 2015

……...……

podpis bakalanta

České vysoké učení technické v Praze
Fakulta elektrotechnická

katedra telekomunikační techniky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: **Korych Matěj**

Studijní program: Komunikace, multimédia a elektronika
Obor: Síťové a informační technologie

Název tématu: **Parametry okolních buněk**

Pokyny pro vypracování:

Navrhněte, vytvořte a ověřte aplikaci pro platformu Android, která zobrazí parametry buněk mobilní sítě dostupných v nejbližším okolí mobilního telefonu.

Seznam odborné literatury:

[1] Eberspächer, J. et. al.: GSM - Architecture, Protocols and Services, 3rd Edition. John Wiley & Sons, 2008. ISBN 978-0-470-03070-7.
[2] Elektronické materiály dostupné na http://developer.android.com/sdk [on-line].

Vedoucí: Ing. Pavel Bezpalec, Ph.D.

Platnost zadání: do konce letního semestru 2015/2016

prof. Ing. Boris Šimák, CSc.
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 12. 12. 2014

Anotace:

Tématem bakalářské práce je návrh, realizace a následné testování aplikace pro mobilní operační systém Android. Tato aplikace zobrazuje plnou škálu parametrů, jež se týkají mobilní pozemní buňkové sítě a to ve všech síťových specifikacích, které se v České republice používají.

Aplikace využívá funkcí veřejně dostupného API společnosti Google a v kombinaci s veřejně dostupnou databází mobilních buněk, představuje přehledný nástroj pro monitorování sítě.

Klíčová slova:

Android, Cell ID, CID, Location Area Code, LAC, Java, Google, GPS, MCC, MNC

Abstract:

The topic of this bachelor thesis is to design, create and test an application for mobile operating system Android. This application provides a wide scale of parameters which belongs to terrestrial mobile cellular network through all main specifications available in Czech Republic.

The application uses public API of the Google company and public database of mobile cells and provides simple and powerful utility for network monitoring.

Index Terms:

Android, Cell ID, CID, Location Area Code, LAC, Java, Google, GPS, MCC, MNC

Obsah

1. Úvod

Tématem této bakalářské práce je navrhnout, zrealizovat a otestovat aplikaci pro mobilní operační systém Android, která umožní uživateli zjistit všechny podstatné informace o buňkové mobilní síti, ve které se právě nachází.

Po domluvě s vedoucím práce byly stanoveny hlavní cíle práce, které se týkají především funkčnosti aplikace. Aplikace musí zejména umožňovat přehledné zobrazení identifikace sítě jako takové a dále také identifikačních údajů jednotlivých buněk, a to jak aktuální buňky se kterou telefon komunikuje, tak veškerých okolních buněk, které je schopen mobilní telefon detekovat.

Dále musí být aplikace schopna pracovat s veřejně dostupnou databází z webu www.gsmweb.cz, která obsahuje zejména umístění jednotlivých buněk a to formou slovního popisu a dále i pomocí přesných GPS souřadnic. Tyto údaje aplikace využije společně s daty získanými z vlastního GPS zařízení a následně vypočítá vzdálenost mobilního telefonu od detekovaných buněk. Pomocí rozhraní aplikace Google Maps dále umožní zobrazení umístění buňky v mapě. Samozřejmostí je možnost automatického stažení potřebné databáze z internetu a její případná aktualizace.

Téma bakalářské práce bylo vybráno především z důvodu snoubení telekomunikačních a programátorských znalostí do jedné komplexní práce, která umožnuje kreativní řešení a jejímž výsledkem je konkrétní produkt, který může dále být používaný širokým spektrem uživatelů. Zároveň toto téma umožňuje nabýt hlubších znalostí v oblasti programování aplikací pro operační systém Android. Tyto znalosti mohou býti v budoucnu velice užitečné.

Velký důraz v průběhu řešení bakalářské práce byl dán na testování vyvinuté aplikace na různých zařízeních. V průběhu testování bylo zjištěno několik odlišných způsobů chování operačního systému Android na zařízeních jednotlivých výrobců. Této oblasti je věnována jedna z kapitol této práce.

2. Teoretická část

V teoretické části práce bude blíže představen mobilní operační systém Android, dále architektura aplikace v tomto operačním systému a také bude vysvětlen princip mobilní buňkové sítě.

2.1. Operační systém Android

Následující kapitoly blíže pojednávají o historii a architektuře systému Android.

2.1.1. Historie

Mobilní operační systém Android byl uveden na trh v roce 2008 společností Google, respektive uskupením Open Handset Alliance, které sdružuje 84 společností podnikajících v různých odvětvích. Členy tohoto uskupení jsou například výrobci mobilních telefonů Samsung, LG, HTC, mobilní operátoři Telefónica, T-Mobile a další společnosti, které se zabývají především výrobou hardwaru.

Vývoj systému začal verzí 1.0 a prozatím poslední verze nese číselné označení 5.1. Za zmínku jistě stojí i systém pojmenování jednotlivých verzí. Od verze 1.5 se mimo číselného pojmenování uvádí i slovní pojmenování, které vždy označuje nějakou cukrovinku. Při vymýšlení pojmenování nové verze, se vždy postupuje na cukrovinku začínající na následující písmenko v abecedě, než začínalo pojmenování předchozí verze. Již jsme se tedy mohli setkat s názvy 1.5 Cupcake (koláček), 1.6 Donut (kobliha), 2.0-2.1 Eclair (kremrole), 2.2 Froyo (mražený jogurt), 2.3 Gingerbread (perníček), 3.0-3.2 Honeycomb (včelí plástev), 4.0. Ice Cream Sandwich (ruská zmrzlina), 4.1-4.3 Jelly Bean (želé bonbony), 4.4 KitKat a konečně v aktuálních verzích 5.0-5.1 lze ochutnat Lollipop (lízátko).

2.1.2. Architektura operačního systému Android

Architekturu operačního systému Android je možné v zásadě rozdělit na pět vrstev. Nejnižší a zároveň jedna z nejdůležitějších vrstev, na které operační systém stojí je jádro neboli kernel.

Jádro se stará o přímou komunikaci s hardwarem telefonu. Pokud chce jakákoliv vyšší vrstva systému komunikovat s hardwarem, musí o toto požádat jádro, které vyšší vrstvě takzvaně poskytne službu. Jádro má také na starosti například správu všech běžících procesů, správu paměti, sítě a podobně. Součástí jádra je soubor ovladačů potřebných pro komunikaci s hardwarem. Tyto ovladače bývají pro každý model telefonu specifické, jelikož zpravidla každý telefon je vybaven různou skladbou hardwaru.

Jádro operačního systému Android je postaveno na jádře systému Linux, zpravidla na verzi 2.6. Jádro Androidu však není úplně shodné s jádrem Linuxu. Neobsahuje totiž některé knihovny, které na mobilních telefonech nejsou potřeba. Se systémem Linux má však Android velmi mnoho společných vlastností. Hlavní předností Androidu je jeho velká otevřenost. Velká část systému je vyvíjena jakožto open source, tudíž si může jakýkoliv výrobce, či koncový uživatel systém přizpůsobit k obrazu svému. [1] [2]

Další vrstvou systému jsou takzvané systémové knihovny. Tyto knihovny jsou napsané v jazyce C/C++ a jsou potřebné pro zajištění základního chodu jádra. Součástí této vrstvy jsou například knihovny Open GL a SGL, které zajištují grafické zobrazení. [1] [2]

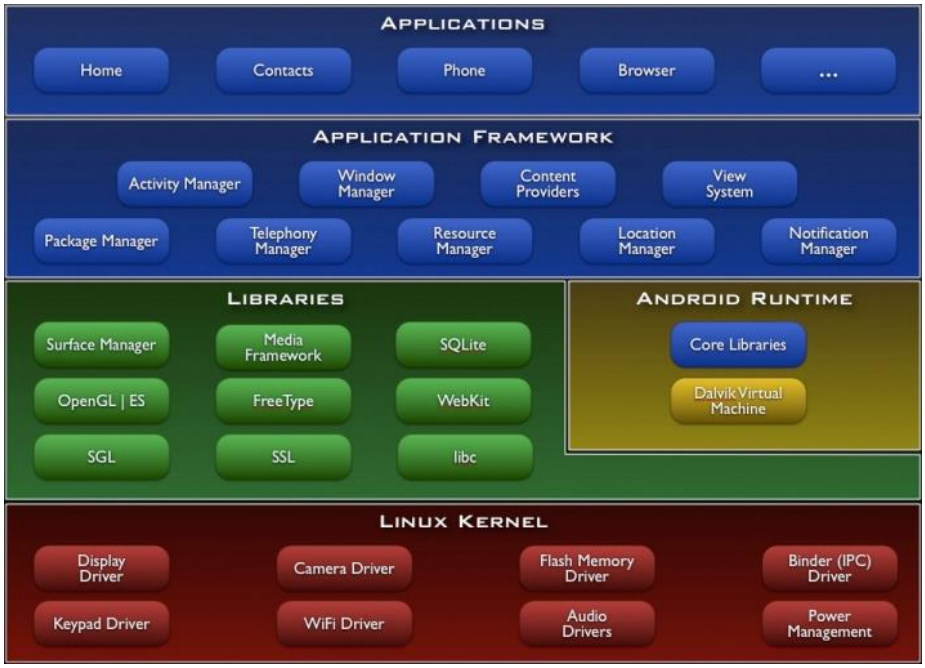
Pro běh aplikací slouží vrstva Android Runtime neboli běhové prostředí systému Android. Tato vrstva mimo základních knihoven jazyka Java obsahuje především virtuální stroj Dalvik Virtual Machine (DVM). Tento virtuální stroj plní naprosto stejnou funkci jako virtuální stroj jazyka Java – Java Virtual Machine, který je známý z prostředí PC. Aplikace jazyka Java se nedají spustit přímo spuštěním jejich zkompilovaného bytecodu, jako v jiných programovacích jazycích. K jejich spuštění je zapotřebí virtuální stroj, na kterém aplikace běží.

Nativní virtuální stroj jazyka Java však nemohl být v systému Android použit, jelikož autorská práva k tomuto stroji drží společnost Oracle (dříve Sun Microsystems), která tento virtuální stroj neposkytuje jako open source. Společnost Google tedy vyvinula vlastní virtuální stroj, který nejdříve převede Java bytecode do svého Dalvik bytecodu a až ten je posléze spuštěn. Ačkoliv je Dalvik Virtual Machine rozdílný oproti Java Virtual Machine, například obsahuje vlastní grafické knihovny a je přizpůsobený mobilním telefonům, společnost Oracle v roce 2010 zažalovala společnost Google za porušení autorských práv [3]. Spor do dnešních dnů nebyl definitivně rozhodnut, avšak donutil společnost Google vyvinout zcela novou běhovou vrstvu nazvanou Android Runtime (ART), která již není založena na žádném virtuální stroji a její princip stojí na převedení Java bytecodu do nativního kódu procesoru. Tento překlad se provede při instalaci a aplikace již zůstane v tomto kódu, který se již přímo vykonává procesorem. Výhodou tohoto řešení je mimo vyřešení právních sporů také zvýšení výkonu aplikace. Nevýhodou může být delší doba instalace a zabrání většího místa v paměti výsledným spustitelným kódem. DVM byl nahrazen prostředním ART počínaje verzí Androidu 5.0. [4]

Předposlední vrstvou systému Android je vrstva Application Framework, ve které se nachází knihovny jazyka Java. Tyto knihovny jsou již přímo využívány uživatelskými aplikacemi pro přístup k různým funkcím systému. Prostřednictvím této vrstvy využívají aplikace například přístup k telefonním funkcím, určení polohy telefonu, práci s notifikacemi, přístupu k síti atd. S touto vrstvou pracuje programátor aplikací, jelikož je to jediná vrstva, se kterou může uživatelská aplikace komunikovat. [1] [2]

Nejvyšší vrstvu systému tvoří vlastní uživatelské aplikace bez ohledu na to, jestli již byly v systému předinstalovány, či byli nainstalovány uživatelem. Jinými slovy lze říci, že tato takzvaná aplikační vrstva je jediná, kterou uživatel fyzicky vidí na svém displeji a se kterou interaguje.

V níže uvedeném schématu je názorně vidět hierarchie vrstev operačního systému Android.



Obrázek 1 – Architektura operačního systému Android, převzato z [5]

2.1.3. Vývojové nástroje

Vývoj aplikací probíhá primárně v programovacím jazyce Java. K psaní aplikací je zapotřebí vývojový kit Android Software Development Kit (Android SDK), který obsahuje veškeré knihovny, debugger, emulátor operačního systému Android a samozřejmě také veškerou potřebnou dokumentaci. Nejpoužívanější vývojové prostředí pro vývoj aplikací je Eclipse IDE, které bylo donedávna i oficiálně doporučováno společností Google. V prosinci 2014 vydala společnost Google stabilní verzi svého vlastního vývojového prostředí s názvem Android Studio, které je od té doby jediné oficiálně doporučované vývojové prostředí. Android SDK včetně Android Studia je dostupné pro operační systémy Windows, Linux i Mac OS X.

Vývoj aplikací je možný také v jiných programovacích jazycích. Oficiální alternativou podporovanou společností Google je použití jazyků C/C++ a to pomocí sady Native Development Kit (NDK). Nicméně tato alternativa není příliš doporučována, jelikož kódy v jazycích C/C++ vykazují v systému Android zpravidla menší výkonnost než kódy v jazyce Java. Použití sady NDK je tedy vhodné pouze pro ty vývojáře, kteří nemají příliš zkušeností s jazykem Java, případně mají již hotové kódy v jazycích C/C++, takže jim tímto odpadá nutnost přepisovat tento kód do jazyka Java. [6]

Další alternativou může být například jazyk HTML5. Jazyk HTML5 slouží pro vývoj webových aplikací, což znamená, že pro běh těchto aplikací je nutné v systému Android využít webový prohlížeč ať už vestavěný nebo jakýkoliv jiný. Nevýhodou těchto aplikací je velice limitovaný přístup k funkcím systému. Aplikace v HTML5 například nemůže používat rozhraní Bluetooth, pořizovat audio a video záznamy, spouštět jinou aplikaci atd. Je nutné také počítat s tím, že HTML5 je podporováno až od verze systému 2.0. [7]

Mezi další alternativní jazyky patří například: JavaScript, Ruby, Adobe AIR, C# a mnoho dalších. Zastoupení těchto alternativních jazyků je však velice minoritní a drtivá většina aplikací je v současné době vyvíjena v jazyce Java.

2.2. Architektura aplikace v operačním systému Android

Nyní se blíže podíváme na nejdůležitější součásti, které jsou potřeba pro běh aplikace. Veškeré zdrojové kódy aplikace jsou uloženy v souborech s příponou java. V nich se pak nachází vlastní zdrojový kód v jazyce Java. V kódu jazyka Java se píše veškerá funkcionalita aplikace. Zpravidla se v jazyce Java nepíše grafické rozhrání, které je praktičtější definovat v souborech Layout, které budou popsány později. Nicméně systém Android podporuje vytvoření grafického rozhrání i v jazyce Java, ačkoliv je mezi vývojáři využíváno minimálně.

V následujících kapitolách budou představeny základní stavební kameny aplikace.

2.2.1. Aktivita

Základním stavebním prvkem všech aplikací pro systém Android je Aktivita (anglicky Activity). Aktivita je v zásadě to, co uživatel vnímá jako jednu konkrétní obrazovku aplikace. Aktivitu lze také přirovnat k oknu aplikace, tak jak je známé například z operačního systému Windows pro PC. [8]

2.2.2. Služba

Služba (anglicky Service) je proces, který běží na pozadí a provádí nějakou činnost nezávisle na běhu aktivity, která služba zavolala. Jedná se o obdobu démonů z desktop operačních systémů. Služba v systému Android může například zajišťovat stahování souborů, přehrávaní hudby a jiné. [9]

2.2.3. Poskytovatel obsahu

Poskytovatelé obsahu (anglicky Content Providers) jsou entita, která má za úkol umožňovat přístup k určitým datům. Pokud je například potřeba zpřístupnit data vygenerovaná aplikací pro jiné aplikace, vytvoří se poskytovatel obsahu. Aplikace, která bude chtít tato data číst, osloví takto vytvořeného poskytovatele obsahu a ten na základě nadefinovaných algoritmů rozhodne, zda data zpřístupní. [10]

2.2.4. Záměry

Záměry (anglicky Intents) jsou v podstatě zprávy, pomocí kterých spolu komunikují jednotlivé součásti aplikace. Pomocí záměru může aktivita spustit jinou aktivitu, případně službu. Záměry se také používají pro informování součástí aplikace o nějaké události. Například pokud telefon ztratí signál, může o tom být aplikace informována systémem pomocí záměru, pokud je v ní vytvořen patřičný filtr záměru, který bude tento záměr detekovat. [11]

2.2.5. Manifest

Součástí každé aplikace musí být též soubor zvaný Manifest psaný v jazyce XML. Manifest je možné považovat za takový soupis vlastností aplikace. Zapisuje se do něj například

minimální požadovaná verze operačního systému a to jako parametr `android:minSdkVersion`.

Tato informace se však nezapisuje ve formátu verze systému, tak jak je všeobecně známa, například tedy 2.3, 4.0, 4.2 a podobně, nýbrž jako takzvaná verze vývojového rozhraní API (Application Programming Interface). V případě systému Android je API číslováno od čísla 1, které odpovídá verzi systému 1.0. až po současné API 22, které odpovídá nejnovější verzi 5.1 [12]. Bližní pohled na číslování API nabídne následující tabulka.

Platform Version	API Level	VERSION_CODE
Android 5.1	22	LOLLIPOP_MR1
Android 5.0	21	LOLLIPOP
Android 4.4W	20	KITKAT_WATCH
Android 4.4	19	KITKAT
Android 4.3	18	JELLY_BEAN_MR2
Android 4.2, 4.2.2	17	JELLY_BEAN_MR1
Android 4.1, 4.1.1	16	JELLY_BEAN
Android 4.0.3, 4.0.4	15	ICE_CREAM_SANDWICH_MR1
Android 4.0, 4.0.1, 4.0.2	14	ICE_CREAM_SANDWICH
Android 3.2	13	HONEYCOMB_MR2
Android 3.1.x	12	HONEYCOMB_MR1
Android 3.0.x	11	HONEYCOMB
Android 2.3.4 Android 2.3.3	10	GINGERBREAD_MR1
Android 2.3.2 Android 2.3.1 Android 2.3	9	GINGERBREAD
Android 2.2.x	8	FROYO
Android 2.1.x	7	ECLAIR_MR1
Android 2.0.1	6	ECLAIR_0_1
Android 2.0	5	ECLAIR
Android 1.6	4	DONUT
Android 1.5	3	CUPCAKE
Android 1.1	2	BASE_1_1
Android 1.0	1	BASE

Obrázek 2 - Tabulka API číslování, převzato z [12]

Tato informace slouží především pro obchod Google Play, který pokud procházíme jeho nabídku ze systému s nižší než minimální verzí, která je uvedena u Manifestů aplikací, tyto aplikace vůbec nenabídne. Stejně tak, pokud se pokoušíme ručně aplikaci v telefonu nainstalovat, správce instalace si přečte manifest a pokud není splněna podmínka minimální verze, instalace selže. Pokud tedy v manifestu určíme jako minimální API číslo 9, aplikaci nebude možné nainstalovat na Android 2.2 a nižší.

Další důležitá věc, která musí být v Manifestu uvedená, jsou oprávnění, o které si aplikace žádá při instalaci. Oprávnění se jednoduše zapisují ve tvaru:

```
<uses-permission android:name="OPRÁVNĚNÍ_O_KTERÉ_ŽÁDÁME"/>
```

V Manifestu je nutné uvést všechna oprávnění, která jsou pro chod aplikace potřeba. V případě, že se tak nestane, bude aplikace při pokusu o přístup k prostředku, ke kterému nemá oprávnění, ukončena. V tabulce uvedené níže jsou příklady některých oprávnění. Plný seznam je pak k nalezení například v [13].

INSTALL_PACKAGES	Allows an application to install packages.
INSTALL_SHORTCUT	Allows an application to install a shortcut in Launcher
INTERNAL_SYSTEM_WINDOW	Allows an application to open windows that are for use by parts of the system user interface.
INTERNET	Allows applications to open network sockets.
KILL_BACKGROUND_PROCESSES	Allows an application to call `killBackgroundProcesses(String)`.
LOCATION_HARDWARE	Allows an application to use location features in hardware, such as the geofencing api.
MANAGE_ACCOUNTS	Allows an application to manage the list of accounts in the AccountManager
MANAGE_APP_TOKENS	Allows an application to manage (create, destroy, Z-order) application tokens in the window manager.
MANAGE_DOCUMENTS	Allows an application to manage access to documents, usually as part of a document picker.
MASTER_CLEAR	Not for use by third-party applications.
MEDIA_CONTENT_CONTROL	Allows an application to know what content is playing and control its playback.
MODIFY_AUDIO_SETTINGS	Allows an application to modify global audio settings
MODIFY_PHONE_STATE	Allows modification of the telephony state - power on, mmi, etc.
MOUNT_FORMAT_FILESYSTEMS	Allows formatting file systems for removable storage.
MOUNT_UNMOUNT_FILESYSTEMS	Allows mounting and unmounting file systems for removable storage.
NFC	Allows applications to perform I/O operations over NFC

Obrázek 3 – Příklady oprávnění, převzato z [13]

Dále musí být v Manifestu zapsané všechny Aktivity, služby, filtry záměrů, používané knihovny atd. Příklad podoby konkrétního Manifestu je uveden v kapitole 3.2.1.

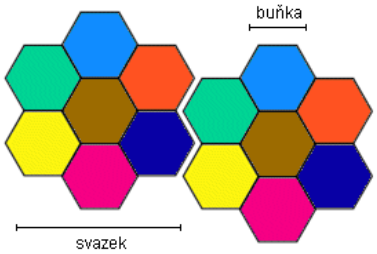
2.2.6. Definice grafického rozvržení

Definice grafického rozvržení (Layout) se definují stejně jako Manifest v souborech s příponou xml. Tyto soubory jsou umístěny v složce /res/layout projektového adresáře. Soubor rozvržení musí být přiřazen pro každou aktivitu, u které je požadováno grafické rozhraní. Jeden soubor definice rozvržení může být společný i pro více aktivit. Definice grafického rozvržení se definuje v jazyce XML. Konkrétní příklady těchto souborů jsou uvedeny v kapitole 3.2.2.

Od verze 3.0 byl do systému Android zakomponován systém fragmentů. Fragment je vrstva, která zjednodušuje implementaci rozvržení pro zobrazení na různých velikostech obrazovky.

2.3. Pozemní mobilní buňková síť

Princip pozemní buňkové mobilní sítě byl definován v roce 1946 v Bellových laboratořích. Tento princip spočívá v rozdělení území na takzvané buňky, které mají hexagonální (šestihranný) tvar. Tyto buňky se spojují do svazku o větším počtu buněk, zpravidla o sedmi buňkách. Grafické znázornění buňkové sítě je možné vidět na následujícím obrázku.

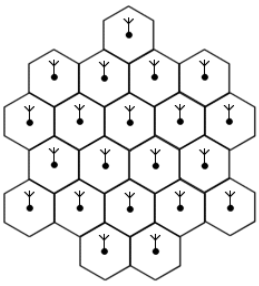


Obrázek 4 – Myšlenka buňkové sítě, převzato z [14]

Spojováním těchto svazků lze pokrýt libovolně velké území. Pro jednoduchost uvažujme, že pro pokrytí jedné buňky bude použit jeden radiový kanál, a že velikost interferenční oblasti, tedy oblasti, ve které se nesmí použít dvakrát stejný kanál, jelikož by došlo k vzájemnému rušení, bude rovna poloměru pěti buněk. Pokud se tedy použijí svazky o sedmi buňkách a v každém svazku bude použito sedm kanálů, pro každou buňku jeden, vzdálenost dvou stejných kanálů bude přibližně rovna poloměru pěti buněk a tudíž nedojde k interferenci.

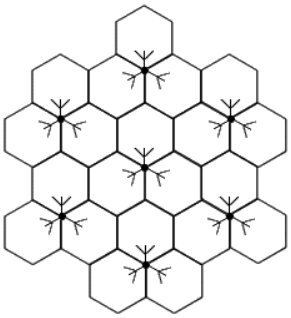
Buňková síť tedy pomáhá šetřit radiové kanály, avšak za cenu vyhrazení jedné základnové stanice pro každou buňku. V praxi je však toto řešení nevhodné, jelikož by jedna základnová stanice obsluhovala příliš velké území. Z důvodu snížení vysílacích výkonů a zároveň navýšení kapacity se využívá takzvané sektorizace. Její princip bude vysvětlen na následujícím příkladu. Svazek z výše uvedeného obrázku rozdělíme na 21 menších buněk, tedy každou buňku ze sedmi buňkového svazku rozdělíme na 3 menší.

Vznikne svazek o 21 základnových stanicích, viz následující obrázek:



Obrázek 5 – Svazek 21 buněk, převzato z [15]

Počet základnových stanic se však dá zredukovat zpět na původních sedm díky jejich umístění do společného bodu třech sousedních buněk.



Obrázek 6 - Sektorizace, převzato z [16]

Původní všesměrovou anténu nahradí v tomto případě tři směrové (sektorové) antény, každá s azimutem vyzařování 120°. Každá tato anténa má svůj vlastní vysílač, který pracuje na vlastním kanálu rozdílném od sousedních kanálů. Buňky se samozřejmě mohou i překrývat. Překrývání se používá zejména v exponovaných oblastech, kde se pohybuje velké množství uživatelů, typicky města. Překrýváním získáme mimo zvýšení kapacity sítě v dané oblasti také zálohu pro případ, kdy by jedna z buněk vypadla. V případě výpadku by danou oblast stále pokrývala druhá buňka a uživatel by tak měl stále přístup ke službám.

Buňky se rozdělují podle velikosti území, které pokrývají, dle následujícího schématu [17], [18]:

Makrobuňka (Macrocell) – poloměr od 1 km do cca 30 km

- Používá se pro pokrytí velkého území, zpravidla málo osídleného. Typické použití je ve venkovských oblastech. Antény jsou umístěné nad úrovní zástavby.

Mikrobuňka (Microcell) – poloměr řádově několik stovek metrů

- Použití v městské zástavbě, kde se využívá šíření v ulicích, které slouží jako vlnovody. Antény se umísťují zpravidla pod úrovní střech.

Pikobuňka (Picocell) – poloměr řádově několik desítek metrů

- Typické použití nalézá ve vnitřním pokrytí budov s velkou koncentrací uživatelů, kde může jednak zvyšovat kapacitu sítě, jednak zkvalitnit pokrytí v případě, že pokrytí z venkovních buněk není dostatečné.

Deštníková buňka (Umbrella Cell)

- Zvláštní typ buňky, který se využívá pro pokrytí malých míst bez signálu. Zpravidla velikostně odpovídají mikrobuňkám a pikobuňkám.

Femtobuňka – poloměr cca do 20 metrů

- Speciální typ buňky určený pro pokrytí domácností. Jedná se o malé zařízení, které se připojí do sítě pomocí internetu a umožní tak pokrytí signálem tam, kde k dispozici není. Veškeré náklady na pořízení a provoz této buňky nese koncový uživatel. V České republice poskytuje femtobuňky mobilní operátor Vodafone. [19]

V následující kapitolách se seznámíme s podobou a odlišnostmi radiové přístupové sítě u jednotlivých typů mobilních sítí.

2.3.1. Síť GSM

V síti GSM (Global System for Mobile Communications), označované často jako takzvaná síť druhé generace (2G), je radiová přístupová síť tvořena dvěma hlavními prvky. Prvním prvkem jsou základnové stanice, které se v síti GSM nazývají Base Transceiver Stations (BTS). Tyto základnové stanice řídí Base Statition Controller (BSC). BSC řídí všechny BTS, které pod něj spadají a stará se především o řízení a přidělování radiových kanálů, které BTS využívají. Dále mají na starosti handover, tedy přechod účastníků mezi buňkami, respektive základnovými stanicemi. BSC se však stará pouze o handover mezi BTS, které spravuje. Pokud uživatel přechází mezi dvěma BTS, kdy je každá pod správou jiného BSC, tak je tento handover řízen jádrem sítě, konkrétně ústřednou MSC. V síti GSM společnosti O2 Czech Republic a.s. je momentálně v provozu 73 BSC a 5324 BTS. Průměrně tedy jeden BSC obsluhuje 73 BTS. [20]

2.3.2. Síť UMTS

V případě sítě UMTS (Universal Mobile Communications System), obecně nazývané také jako síť třetí generace (3G), byla vyvinuta nová podoba radiové přístupové sítě nazývaná jako UTRAN (UMTS Terrestrial Radio Access Network). Mimo provozních odlišností, jako je schopnost přepojování okruhů i paketů se stejnou prioritou (v síti GSM má přepojování okruhů vyšší prioritu) [21] a schopnost takzvaného soft handoveru, kdy nejdříve proběhne přechod na novou buňku a až posléze dojde k opuštění stávající buňky, dochází také ke změně terminologie. Základnové stanice se tu nazývají Node B a řídí je Radio Network Controller (RNC). RNC zastává prakticky stejnou roli jako BSC u sítě GSM, RNC oproti BSC implementuje některé funkce, které v síti GSM poskytují až komponenty jádra sítě (například šifrování). [22]

2.3.3. Síť LTE

Síť LTE (Long Term Evolution), někdy též nesprávně označovaná jako síť čtvrté generace 4G, disponuje odlišným uspořádáním přístupové sítě oproti starším generacím. V síti LTE již nejsou základnové stanice podřízené žádnému kontroléru a každá z nich funguje v rámci funkcionality přístupové sítě naprosto autonomně. Tyto základnové stanice se nazývají eNode B (Evolved Node B) a jejich název má evokovat „pokročilejší“ Node B známý z UMTS sítě. eNode B dokáží mezi sebou přímo komunikovat a tím zajišťovat handover a další funkce. Základnové stanice jsou přímo napojené na jádro sítě a tím je docíleno rychlejší odezvy sítě. Síť LTE také přináší snížení nákladů díky absenci kontrolérů, které u sítí nižších generací tvoří nezanedbatelnou položku při pořízení a především v následné údržbě těchto prvků. Zároveň se tímto výrazně omezuje zranitelnost sítě, kdy výpadek kontroléru může vyřadit z provozu až několik desítek základnových stanic. [23]

3. Praktická část – Vlastní aplikace NetInfo

Zadáním této bakalářské práce bylo vytvořit přehlednou aplikaci pro zobrazení parametrů okolních buněk. Pro tuto aplikaci byl vybrán název NetInfo. V níže uvedených podkapitolách bude podrobně popsána struktura této aplikace.

3.1. Parametry poskytované aplikací

Aplikace NetInfo umožňuje zobrazení několika parametrů, které budou blíže představeny v následujících kapitolách.

3.1.1. Cell ID

Cell ID je identifikační číslo buňky. Může být též uvedeno pod zkratkou CID. Cell ID tvoří společně s LAC, který bude podrobněji představen později, unikátní identifikační číslo každé jednotlivé mobilní buňky. V případě GSM (2G) sítě má toto číslo délku 2 bajty, může tedy nabývat hodnoty od 0 do 65 535.

V případě sítě UMTS (3G) je situace trochu složitější. Buňky těchto sítí se primárně identifikují údajem UTRAN Cell ID (Long Cell ID, LCID), což je číslo o délce 3,5 bajtu a může tedy nabývat hodnot 0 až 268435455.

U sítí 3G je LCID složeno ze dvou údajů, jedním z nich je Cell ID ve stejném významu jako u sítí GSM, tedy číslo buňky. Druhým je pak číslo kontroléru (RNC). Číslo RNC lze získat z LCID jako celočíselný podíl LCID a čísla 65536. Cell ID poté reprezentuje zbytek po celočíselném dělení LCID a čísla 65536. [24]

V sítích LTE probíhá identifikace pomocí Cell Identity (CI), které se skládá z eNode-B ID, tedy z identifikačního čísla základnové stanice a čísla buňky na dané základnové stanici. eNode-B ID získáme jako celočíselný podíl CI a čísla 256 a číslo buňky získáme jako zbytek po dělení CI a čísla 256. [25]

3.1.2. Location Area Code

Další neméně důležitý údaj je Location Area Code (LAC). LAC má délku 2 bajty a může tedy nabývat hodnoty od 0 do 65535. Jedná se o kód určité geografické lokace, v této lokaci může být jedna a více buněk. V České republice zpravidla jedno číslo LAC pokrývá zhruba území jednoho okresu, případně městské čtvrti ve velkých městech.

LAC je definován pro sítě GSM a UMTS, v sítích LTE je definován obdobný údaj se stejným významem s názvem Tracking Area Code (TAC). [26]

3.1.3. Mobile Country Code a Mobile Network Code

S údajem LAC souvisí také následující dva údaje MCC a MNC. MCC je zkratkou pro Mobile Country Code. Jedná se o třímístné číslo, které jednoznačně určuje zemi, respektive území, na kterém se daná síť nachází.

Samotného mobilního operátora pak identifikuje třímístné číslo Mobile Network Code (MNC). Kombinace MCC+MNC tedy jednoznačně identifikuje konkrétního mobilního operátora v konkrétní zemi. Například pro ČR je vyhrazen MCC 230 a jednotliví operátoři pak mají přidělené MNC začínající od čísla 01, které vlastní T-Mobile. MCC samozřejmě může být přiděleno i pro operátora, který působí celosvětově, například satelitní mobilní operátoři. V tomto případě používají MCC 901. [27]

Kombinace kódů MCC, MNC, Cell ID, a LAC tedy jednoznačně určuje konkrétní buňku v konkrétní lokalitě, ve vlastnictví konkrétního operátora, působícího v konkrétní zemi. Tedy jinými slovy tato kombinace jednoznačně určuje buňku v rámci celého světa.

3.1.4. Síla signálu

Poslední informací, týkající se mobilní sítě, kterou aplikace zobrazuje, je síla přijímaného signálu. Tento údaj aplikace zobrazuje ve dvou jednotkách a to primárně v jednotkách ASU (Arbitrary Signal Unit), ve kterých tento údaj poskytuje systém Android. Pro lepší názornost aplikace zobrazuje také hodnotu v jednotkách dBm, kterou získá převodem z jednotky ASU.

Jednotka ASU je definována v rozmezí hodnot 0-31, kde 0 ASU odpovídá síle signálu menší nebo rovno -113 dBm a 31 ASU odpovídá -51 dBm a více. Hodnota 99 ASU značí neznámou sílu signálu. [28]

Přepočet ASU na dBm je tedy možné provést pomocí vzorce:

$$dBm = 2 * ASU - 113$$

Výše uvedený výpočet platí pouze pro síť GSM, v síti UMTS a LTE se výpočet provádí jinak. Jelikož však aplikace kvůli omezení, které bude popsáno dále v textu, zobrazuje sílu signálu pouze v síti GSM, výpočet pro ostatní typy sítí nebude uveden.

3.2. Vývoj aplikace

V následujících kapitolách se blíže podíváme na strukturu aplikace NetInfo. Rozebereme si nejdůležitější soubory, které aplikaci tvoří, přiblížíme si některé důležité pasáže ze zdrojového kódu a vysvětlíme i některá omezení.

3.2.1. Manifest – rodný list aplikace

O významu manifestu pojednává kapitola 2.2. Nyní se podíváme na podobu vybraných pasáží manifestu aplikace NetInfo. Kompletní podobu manifestu je možné shlédnout na přiloženém CD.

Níže uvedená část tvoří takzvanou hlavičku manifestu. Zde je umístěn zejména název balíčku aplikace (package). V tomto případě byl balíček pojmenován dle konvence, kdy první element značí doménu země vývojáře (používají se národní domény, ale například i doména com), druhý element značí vývojářskou instituci, případně jméno či přezdívku autora a posledním elementem bývá název aplikace. V případě aplikace NetInfo, byl vybrán název balíčku *cz.fel.netinfo*. V dalších řádcích je uvedena verze aplikace. Dle konvencí se finální verze označuje jako 1.0.

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="cz.fel.netinfo"
    android:versionCode="1"
    android:versionName="1.0" >
```

Následující řádky značí minimální verzi systému, na které má aplikace autorem dovoleno být nainstalována, v tomto případě byla zvolena verze API 8, která odpovídá verzi sytému 2.2 Froyo viz obrázek č. 2 v kapitole 2.2.5. Dále je zde uvedena cílová verze systému. Zpravidla se zde uvádí nejnovější dostupná verze. V tomto případě je tedy uvedeno API 22 odpovídající nejnovější verzi 5.1.

```
    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="22" />
```

Další blok tvoří výčet oprávnění, o která si aplikace žádá. Konkrétně jsou to oprávnění: *čtení stavu telefonu* (nutné pro zjištění parametrů buněk), *přístup k přesné a přibližné poloze* (nutné k zjištění geografické polohy telefonu), *přístup k internetu a zjištění dostupnosti síťových připojení* (nutné k aktualizaci databáze), dále *oprávnění k zapisování a čtení externího uložiště* (nutné pro práci s databází) a v poslední řadě také *oprávnění pro práci se službami Google* (nutné pro použití Google Maps).

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

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

```
<uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission
android:name="android.permission.READ_EXTERNAL_STORAGE"/>
<uses-permission
android:name="com.google.android.providers.gsf.permission.READ_GSERVICES"
/>
<permission
    android:name="cz.fel.netinfo.permission.MAPS_RECEIVE"
    android:protectionLevel="signature" />
 <uses-permission android:name="cz.fel.netinfo.permission.MAPS_RECEIVE"
/>
```

V manifestu je také uložena hodnota klíče, který je potřeba pro funkčnost Google Maps v aplikaci. Tento klíč je přidělen společností Google po registraci autora na webu https://console.developers.google.com a je platný pouze s unikátním podpisem této aplikace.

```
<meta-data
android:name="com.google.android.maps.v2.API_KEY"
android:value="AIzaSyCCexlDMYu0LRdEKs8u2Vv9XArd2TwQ-ls" />
<meta-data
android:name="com.google.android.gms.version"
android:value="@integer/google_play_services_version" />
    </application>

</manifest>
```

3.2.2. Definice rozvržení

Aplikace NetInfo obsahuje celkem tři aktivity a každá z nich disponuje vlastní definicí rozvržení.

3.2.2.1. Rozvržení hlavní aktivity NetInfo – aktualnibunka.xml

Rozvržení hlavní aktivity obsahuje nejvíc elementů ze všech použitých rozvržení. Jako takzvaný rodičovský element je zde použit kontejner LinearLayout, který zabírá celou velikost obrazovky pomocí příznaku "match_parent" u parametrů definujících šířku a výšku kontejneru.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
```

V tomto kontejneru se pak nachází další 4 bloky vytvořené pomocí dvou kontejnerů TableRow a dále pomocí kontejneru TextView a Button. Kontejner TableRow definuje vždy jeden řádek, u kterého pak může vývojář definovat jeho šířku, výšku, případně počet sloupců v každém řádku. V případě rozvržení hlavní aktivity jsou všechny 4 kontejnery definovány na celou šířku obrazovky.

Kontejner TextView obsahuje již pouze zobrazení konkrétního textového pole a kontejner Button dle svého názvu obsahuje tlačítko, ke kterému je připojena určitá funkce.

V horní části obrazovky je umístěn první kontejner TableRow, který obsahuje dva kontejnery Button. Výška kontejneru TableRow stejně tak jako šířka a výška obou kontejnerů Button se přizpůsobuje dynamicky jejich obsahu pomocí parametru "wrap_content". Jejich velikost tedy záleží na rozlišení obrazovky a velikosti písma v konkrétním telefonu.

```
<TableRow
        android:id="@+id/tableRow1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center" >

        <Button
            android:id="@+id/button1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />

        <Button
            android:id="@+id/button2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="" />
    </TableRow>
```

Pod prvním kontejnerem TableRow je vložen kontejner TextView, který obsahuje nadpis „Aktuální buňka:", který je proveden s příznaky zarovnání na střed, použití velkého písma a černé barvy písma.

```
<TextView
        android:id="@+id/textView2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:text="Aktuální buňka:"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:textColor="#000000" />
```

Třetí element tvoří druhý kontejner TableRow, který obsahuje textové pole TextView. Toto textové pole slouží pro zobrazení informací o aktuální buňce a je tedy provedeno s příznaky použití střední velikosti písma černé barvy s fontem Serif pro lepší přehlednost. Samozřejmostí je také zarovnání na střed.

```
TableRow
        android:id="@+id/tableRow2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_weight="0.33">

    <TextView
        android:id="@+id/textView1"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:gravity="center"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:textColor="#000000"
        android:typeface="serif" />
    </TableRow>
```

Poslední element tvoří tlačítko na spodním okraji obrazovky, které zabírá celou šířku obrazovky a jeho výška se přizpůsobuje jeho obsahu.

```
<Button
    android:id="@+id/button3"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:gravity="center" />
```

Výsledný náhled definice rozvržení v grafickém editoru prostředí Eclipse je vidět na následujícím obrázku.

Obrázek 7 - Náhled definice rozvržení

3.2.2.2. Rozvržení aktivity Okolní buňky – okolnibunky.xml

Aktivita Okolní Buňky má za úkol pouze vypisovat informace o okolních buňkách. Z tohoto důvodu je zde použito jednoduché rozvržení, které obsahuje pouze kontejner TextView, který zabírá celou obrazovku a používá písmo Serif v černé barvě.

```
<TextView xmlns:android="http://schemas.android.com/apk/res/android"
     android:id="@+id/textView1"
     android:layout_width="match_parent"
     android:layout_height="match_parent"
     android:textColor="#000000"
     android:gravity="center"
     android:typeface="serif" />
```

3.2.2.3. Rozvržení aktivity Mapa – mapa.xml

Aktivita Mapa slouží pro zobrazení aktuální polohy buňky v mapě. Této funkci je také přizpůsobeno rozvržení, které obsahuje 3 elementy a to dva kontejnery TableRow a jeden kontejner Button, které jsou zastřešeny rodičovským kontejnerem LinearLayout.

V horní části obrazovky je umístěn první kontejner TableRow, který obsahuje kontejner typu TextView. Toto textové pole slouží pro zobrazení vzdálenosti od aktuální buňky.

V prostřední části obrazovky je umístěn kontejner TableRow, který obsahuje Fragment s mapou.

```
<TableRow
    android:id="@+id/tableRow2"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_weight="0.98" >

    <fragment
        android:id="@+id/mapa"
        android:name="com.google.android.gms.maps.SupportMapFragment"
        android:layout_width="match_parent" />
</TableRow>
```

V dolní části obrazovky je umístěn kontejner Button, který obsahuje tlačítko sloužící k přepínání mapy mezi základním a satelitním zobrazením.

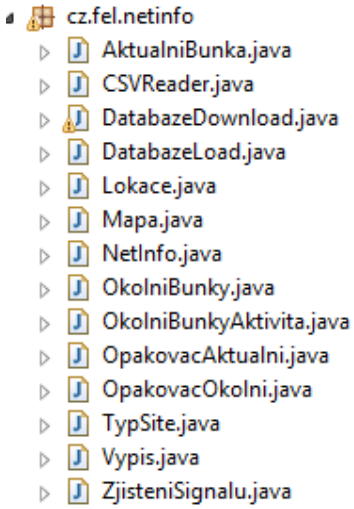
Výslednou podobu rozvržení je možné vidět na obrázku č. 19 v kapitole 3.3.3.

3.2.3. Zdrojový kód v jazyce Java

V následujících kapitolách budou blíže představeny vybrané důležité pasáže ze zdrojového kódu jazyka Java, ve kterém jsou psány veškeré funkcionality aplikace mimo výše zmíněného grafického rozhraní. Pro lepší přehlednost a snadnější implementaci funkcí, bylo zvoleno uspořádání ve stylu zásuvných modulů. Toto uspořádání spočívá ve vytvoření několika souborů se zdrojovým kódem, kde každý tento soubor představuje jednu třídu. Základem aplikace jsou tři třídy, kde každá obsahuje jednu aktivitu. Zdrojové kódy těchto aktivit jsou poměrně strohé a ke většině úkonů používají jiné třídy, ve kterých jsou definovány jednotlivé funkce. V podstatě se dá říci, že zdrojové kódy aktivit mají za úkol pouze inicializovat grafické rozhraní a spojovat jednotlivé zásuvné moduly reprezentované třídami do funkčního celku.

Výhodou tohoto řešení je snadná úprava jednotlivých funkcí, přidávání nových funkcí a také snadnější hledání chyb.

Kompletní zdrojové kódy všech tříd jsou k dispozici na přiloženém CD.



Obrázek 8 - Seznam všech tříd aplikace NetInfo

3.2.3.1. Zobrazení informací o aktuální buňce

V následujících kapitolách budou představeny třídy, které slouží k zobrazení informací o aktuální buňce.

3.2.3.1.1. Třída NetInfo

Třída *NetInfo* je stěžejní třída celé aplikace. Tato třída obsahuje stejnojmennou aktivitu, která je spuštěna jako první při spuštění aplikace. Při spuštění aplikace operační systém najde třídu *NetInfo* a v ní spustí metodu *onCreate()*. V této metodě je nejprve inicializováno grafické prostředí a poté je zavolána metoda *pomocnametoda()*, která vytvoří statické instance některých knihovních tříd.

Tyto instance jsou pak využívány ostatními třídami, které nejsou aktivitami. Důvodem proč jsou tyto instance vytvářeny v třídě *NetInfo*, je omezení systému Android, kdy o přístup k vybraným systémovým službám může žádat pouze aktivita.

V dalším kroku je zavolána třída *OpakovacAktualni*, respektive její metoda *run()*, která má za úkol periodické zjišťování informací o aktuální buňce.

Třída *NetInfo* dále také obsahuje logiku pro ovládání tlačítek na obrazovce. Po stisknutí tlačítka *Okolní buňky* dojde k vytvoření záměru, pomocí kterého je poté spuštěna aktivita *OkolniBunkyAktivita*. Stejně tak po stisknutí tlačítka pro zobrazení v mapě dojde k vytvoření záměru a spuštění aktivity *Mapa*. V případě stisknutí tlačítka *Aktualizace databáze* však nedojde ke spuštění samostatné aktivity, nýbrž pouze k zavolání metody *download()* třídy *DatabazeDownload* a zároveň k deaktivaci tlačítka.

Poslední oddíl, který zdrojový kód třídy *NetInfo* obsahuje, jsou metody *ObnovaUI1()* a *ObnovaUI2()*. První jmenovaná slouží k dynamické změně obsahu textového pole s informacemi o aktuální buňce pomocí volání metody *vypisAktualni()* třídy *Vypis.* Druhá metoda slouží k opětovné aktivaci tlačítka *Aktualizace databáze* po proběhnutí aktualizace.

Vzhledem k relativně velkému rozsahu zdrojového kódu této třídy, není tento kód uveden v textu této práce.

3.2.3.1.2. Třídy AktualniBunka a TypSite

Třída *AktualniBunka* slouží k získávání informací o aktuální buňce a obsahuje jedinou metodu *CellID()*. V této metodě jsou poté získávány všechny informace o aktuální buňce, které je telefon schopný poskytnout.

Pro přístup k informacím o telefonních službách dostupných v telefonu, je v systému Android k dispozici knihovní třída *TelephonyManager* [29]. V této třídě je k dispozici nepřeberné množství metod, pomocí kterých lze tyto informace zjistit. Vlastní metoda *CellID()* používá volání celkem čtyř metod této třídy. Konkrétně se jedná o metody: *getNetworkOperator()* (pro zjištění MCC+MNC, viz kapitola 3.1.3.), *getNetworkOperatorName()* (pro zjištění názvu operátora), *getNetworkType()* (pro zjištění typu sítě) a *getCellLocation()*, která vrací objekt typu *GsmCellLocation* [30]. Z objektu *GsmCellLocation* jsou posléze volány metody *getCid()* a *getLac()* pro získání Cell ID respektive LAC.

Metoda *CellID()* dále ještě obsahuje přepočty mezi Long Cell ID a Cell ID, respektive RNC v případě sítě UMTS a Cell Identity a eNode-B ID, respektive číslem buňky v případě sítě LTE, viz kapitola 3.1.1.

Jelikož metoda *getNetworkType()* vrací číselný identifikátor typu sítě [31], je následně volána vlastní třída TypSite, respektive její metoda *typsite(int typsiteint)*. Tato metoda přebírá jako vstupní parametr proměnou typu int získanou z metody *getNetworkType()*. V metodě *typsite()* je umístěn rozhodovací mechanismus tvořený přepínačem switch. Dle hodnoty proměnné *typsiteint* je proveden příslušný blok kódu, ve kterém je přiřazen název typu sítě a dále ještě vybráno správné názvosloví týkající se daného typu sítě.

Níže je k vidění zdrojový kód metody *CellID()* třídy *AktualniBunka*. Kompletní zdrojový kód této třídy a třídy *TypSite* je k dispozici na přiloženém CD.

```
    public static void cellID() {

        try {
        GsmCellLocation cellLocation = (GsmCellLocation)
NetInfo.telephonyManager.getCellLocation();
networkOperator=(String)NetInfo.telephonyManager.getNetworkOperator();
nazevOperatora=(String)NetInfo.telephonyManager.getNetworkOperatorName();
        typsiteint=NetInfo.telephonyManager.getNetworkType();
        cidshortprevious=cidshort;
        if (cellLocation.getCid()<=65535)
        { cidshort = cellLocation.getCid();}
        else  if(typsiteint!=13){
            cidlong=cellLocation.getCid();
            cidshort=cellLocation.getCid()%65536;
            RNCnumber=cellLocation.getCid()/65536;
        }
        else {
            cidlong=cellLocation.getCid();
            cidshort=cellLocation.getCid()/256;
```

```
            RNCnumber=cellLocation.getCid()%256;
            }
         LAC = cellLocation.getLac();
         TypSite.typsite(typsiteint);
                }
                catch (NullPointerException e)
        {}
    }
}
```

3.2.3.1.3. Třída OpakovacAktualni

Tato třída obsahuje pouze dvě metody a to metodu run() a metodu stop(). Zavoláním metody run() dojde ke spuštění vlákna, které provádí periodické získávání informací o aktuální buňce.

Postupně jsou tímto vláknem volány metody: *CellID()* třídy *AktualniBunka* (pro získání informací o aktuální buňce), *vyberdatabaze()* třídy *DatabazeLoad* (pro výběr databáze dle aktuálního operátora), *databaze()* třídy *DatabazeLoad*, do které jsou předány parametry Cell ID aktuální, LAC, a Cell ID předchozí, na jejichž základě je v této metodě proveden dotaz v databázi, a v neposlední řadě metoda *vzdalenost()* třídy *Lokace* (pro získání vzdálenosti od aktuální buňky).

Nakonec je zavolána metoda *ObnovaUI1()* třídy *NetInfo* pro přepsání textového pole v aktivitě *NetInfo*.

Perioda opakování metody *run()* je nastavena na dobou jedné sekundy. Tímto řešením zachováme zdání běhu aplikace v reálném čase a zároveň ušetříme čas procesoru telefonu.

Zavoláním druhé metody *stop()* je docíleno zastavení běhu vlákna.

Níže je uveden kompletní zdrojový kód této metody:

```
package cz.fel.netinfo;

import android.os.Handler;

public class OpakovacAktualni implements Runnable {
    static Handler casovac=new Handler();
    static NetInfo net=new NetInfo();

public void stop() {
    casovac.removeMessages(0);
}
@Override
public void run() {
    AktualniBunka.cellID();
    DatabazeLoad.vyberdatabaze();

DatabazeLoad.databaze(AktualniBunka.cidshort,AktualniBunka.lac,AktualniBu
nka.cidshortprevious);
    Lokace.vzdalenost();
    net.ObnovaUI1();
    casovac.postDelayed(this,1000);

}
}
```

3.2.3.1.4. Třída ZjisteniSignalu

Pro zjištění signálu byla vytvořena třída *ZjisteniSignalu*, která obsahuje metodu *onSignalStrengthsChanged(SignalStrength signalStrength)*. Zjišťování síly signálu probíhá v aplikaci *NetInfo* pomocí takzvaného naslouchače (listener), který pomocí třídy *PhoneStateListener* naslouchá změně síly signálu. V případě, že je zaznamenána změna síly signálu, je naslouchačem zavolána metoda *onSignalStrengthsChanged(SignalStrength signalStrength)* s objektem typu *SignalStrength* jako parametrem. V této metodě je potom z objektu *SignalStrength* zavolána metoda *getGsmSignalStrength()*. Tato metoda vrací sílu signálu v jednotkách ASU. Dále je zde také proveden přepočet mezi jednotkami ASU a dBm viz kapitola 3.1.4. a korekce získané hodnoty pro telefon Huawei Ascend P2 viz kapitola 4.2. V třídě *SignalStrength* jsou k dispozici metody pro získání síly signálu pouze ze sítí GSM a CDMA. Získat sílu signálu ostatních sítí je možné pomocí metod třídy *CellSignalStrength*. Tato třída je však k dispozici až od verze systému 4.2. Vzhledem k zachování kompatibility se staršími verzemi systému, a také k tomu, že autor aplikace nevlastní telefon s verzí systému 4.2 a novější, není v aplikaci implementováno získání síly signálu pro sítě UMTS a LTE. Níže je k nahlédnutí zdrojový kód této třídy.

```
package cz.fel.netinfo;

import android.telephony.PhoneStateListener;
import android.telephony.SignalStrength;

class ZjisteniSignalu extends PhoneStateListener {
    public static String sila;

     @Override
     public void onSignalStrengthsChanged (SignalStrength signalStrength)
     {

        super.onSignalStrengthsChanged(signalStrength);

       int sila1=signalStrength.getGsmSignalStrength();
       int sila2;

       if (Math.signum(sila1)==-1)
       {
        sila2=(sila1+113)/2;
        sila=String.valueOf(sila1)+" dBm"+", "+String.valueOf(sila2)+"
asu";
        }
     else
         if(sila1==99)
         {sila="Neznáma";}
         else
     {
            sila2=sila1*2-113;

            sila=String.valueOf(sila2)+" dBm"+",
"+String.valueOf(sila1)+" asu";

     }
}
}
```

3.2.3.2. Zobrazení informací o okolních buňkách

Zobrazení informací o okolních buňkách mají na starosti celkem tři třídy, které budou představeny v následujících kapitolách.

3.2.3.2.1. Třída OkolniBunkyAktivita

Třída *OkolniBunkyAktivita* obsahuje stejnojmennou aktivitu, jejímž úkolem je zobrazení okolních buněk. Stejně, jako každá jiná aktivita v systému Android, obsahuje metodu *onCreate()*, jenž má úkol prvotní inicializaci aktivity. V případě této aktivity je nejprve provedena inicializace grafického prostředí, které je zde tvořeno pouze jediným textovým polem. Dále je spuštěna služba pro přístup k GPS a nakonec zavolána metoda *run()* třídy *OpakovacOkolni*. Tento opakovač funguje obdobně, jako v případě opakovače pro aktuální buňku a bude blíže představen v následující kapitole.

Stejně jako v třídě *NetInfo* i zde je obsažena metoda *ObnovaUI1()*, která má za úkol přepis textového pole v průběhu běhu aktivity.

Níže je opět k dispozici kompletní zdrojový kód této třídy:

```
package cz.fel.netinfo;

import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;

public class OkolniBunkyAktivita extends Activity {
    public static TextView a;
    OpakovacOkolni opakovacOkolni=new OpakovacOkolni();


    @Override
    public void onCreate(Bundle icicle) {
        super.onCreate(icicle);
        setContentView(R.layout.okolnibunky);
        a=(TextView)findViewById(R.id.textView1);
        Vypis.inicializace(a);
        Lokace.lokace();
        opakovacOkolni.opakovac();
    }

    @Override
    public void onStop()
    {
        super.onStop();
        opakovacOkolni.stop();

    }

    public void ObnovaUI()
    {
    runOnUiThread(new Runnable() {
        @Override
        public void run() {
            Vypis.vypisOkolni(OkolniBunkyAktivita.a);

    }});
    }}
```

3.2.3.2.2. Třída OkolniBunky

Třída *OkolniBunky* má stejně jako třída *AktualniBunka* za úkol zjištění informací o buňce, nicméně v tomto případě ne pouze o aktuální buňce, ale o všech buňkách v okolí, které je telefon schopen zachytit. Třída obsahuje jedinou metodu *ncellID()*, ve které je nejdříve získána kolekce objektů typu *NeighboringCellInfo* pomocí volání metody *getNeighboringCellInfo()* třídy *TelephonyManager*. Z této kolekce jsou posléze pomocí volání metod *getCid()* a *getLac()* naplněna pole *ncid[]* a *nlac[]*, která obsahují informace o Cell ID respektive LAC všech okolních buněk. S těmito poli je pak dále pracováno při získávání polohy buněk. Metoda *getNeighboringCellInfo()* umožnuje získat informace pouze o okolních buňkách v síti GSM. Okolní buňky ostatních sítí je možné získat z metody *getAllCellInfo()*, která je však opět dostupná až od verze systému 4.2 stejně jako v případě síly signálu. Zdrojový kód třídy *OkolniBunky* je k nahlédnutí níže.

```
package cz.fel.netinfo;

import java.util.List;
import android.telephony.NeighboringCellInfo;

public class OkolniBunky {
    static int[] ncid;
    static int[] nlac;
    static int pocet;
    public static void ncellID(){

AktualniBunka.networkOperator=(String)NetInfo.telephonyManager.getNetwork
Operator();
        List<NeighboringCellInfo> ncellLocation =
NetInfo.telephonyManager.getNeighboringCellInfo();
        pocet=ncellLocation.size();
        ncid=new int[pocet];
        nlac=new int[pocet];
        for(int i=0; i <pocet; i++){
        ncid[i]=
        ncellLocation.get(i).getCid();
        nlac[i]=
        ncellLocation.get(i).getLac();
        }
            }
    }
```

3.2.3.2.3. Třída OpakovacOkolni

Obsah této třídy tvoří opakovač, který byl navržen speciálně pro potřeby zobrazení okolních buněk. Jeho podoba je tedy mírně odlišná od opakovače pro aktuální buňku. Hlavní odlišností je to, že se zde již nepracuje pouze s jednou buňkou, ale s více buňkami. Z tohoto důvodu, jsou veškeré zjišťované parametry uchovávány v polích. V případě zjištění nulového počtu okolních buněk, již nejsou volány metody pro zjištění ostatních údajů a je rovnou vypsána příslušná informace na obrazovku.

Jelikož může být výpis okolních buněk poněkud rozsáhlý a zatížení telefonu je v tomto případě větší, než u jedné buňky, byla zvolena perioda opakování tohoto opakovače na dobu deset sekund.

3.2.3.3. Zobrazení polohy aktuální buňce v mapě

O zobrazení polohy aktuální buňky v mapě se stará jediná třída, která bude podrobněji představena v následující kapitole.

3.2.3.3.1. Třída Mapa

Třetí a zároveň poslední aktivita je tvořena třídou *Mapa*. Zde je opět v pomocí metody *onCreate()* nejprve inicializováno grafické prostředí a poté je provedena inicializace samotné mapy pomocí metody *initialize()* třídy *MapsInitializer*. Dále je provedeno propojení objektu *mapaa* typu *GoogleMap* s *Fragmentem* z definice grafického rozvržení. Pro práci s mapou se standardně používá Fragment typu *MapFragment*, nicméně pro zachování kompatibility se staršími verzemi systémy je v aplikaci *NetInfo* použit *Fragment* typu *SupportMapFragment*. Z tohoto důvodu jsou do aplikace také zakomponovány knihovny zajišťující kompatibilitu Google Maps s těmito verzemi. V dalším kroku je do mapy vloženo několik parametrů: vycentrování mapy na polohu buňky, nastavení určitého přiblížení a povolení ovládacího prvku nastavení přiblížení. Dále je do mapy také přidán terčík znázorňující polohu aktuální buňky a zavolána metoda *run()*. Tato metoda má za úkol periodicky (každou sekundu) ověřovat, zdali se změnily souřadnice umístění buňky. Pokud se změnily, tak je provedeno přemístění terčíku na nové souřadnice. Zároveň je také obnovován výpis vzdálenosti k buňce. Třída také obsahuje logiku pro práci s tlačítkem umístěným pod mapou. Ukázka části zdrojového kódu této třídy je umístěna níže.

```
public void onCreate(Bundle icicle) {
        super.onCreate(icicle);
        setContentView(R.layout.mapa);
        btn=(Button)findViewById(R.id.button1);
        textView1=(TextView)findViewById(R.id.textView1);
         popisek="Satelitní mapa";
        btn.setText(popisek);
        btn.setOnClickListener(this);
        MapsInitializer.initialize(this);

            try{
        mapaa = ((SupportMapFragment)
getSupportFragmentManager().findFragmentById(R.id.mapa)).getMap();
        souradnice=(new LatLng(Double.valueOf(Lokace.souradnice1),
Double.valueOf(Lokace.souradnice2)));
        souradnicepre=souradnice;
        poloha=CameraUpdateFactory.newLatLng(souradnice);
        mapaa.moveCamera(poloha);
        mapaa.animateCamera(CameraUpdateFactory.zoomTo(13));
        mapaa.getUiSettings().setZoomControlsEnabled(true);
        mapaa.setMyLocationEnabled(true);
        onMapReady(mapaa);
        textView1.setText("Vzdálenost: "+Lokace.vzdalenostm);
            run();

            }
            catch (Exception e)
            {
            }
}
```

3.2.3.4. Podpůrné třídy

Aplikace NetInfo obsahuje také třídy, které neslouží bezprostředně k zobrazování informací, nýbrž poskytují služby ostatním třídám.

3.2.3.4.1. Třída DatabazeDownload

Třída *DatabazeDownload* má za úkol aktualizace databází. Obsahuje jedinou metodu *download()*, která se volá při stisknutí tlačítka *Aktualizace databáze* v aktivitě *NetInfo*. Metoda *download()* spustí nové vlákno, které běží na pozadí, a tak neomezuje ostatní aktivity v jejich činnosti. Dále je nejprve ověřeno, zda již existuje cílová složka, kam se databáze ukládají a pokud ne, tak je vytvořena. V dalším kroku je dle aktuálního MCC+MNC rozhodnuto, která databáze se stáhne (v telefonu mohou být uloženy i používány databáze více operátorů, avšak při aktualizaci je stažena pouze databáze, jenž odpovídá aktuální síti) a poté je otevřeno HTTP připojení na konkrétní URL. Databáze jsou stahovány z URL: http://gsmweb.cz/android/netmonster/. Například pro síť O2 je stažen soubor nm23002.csv z URL http://gsmweb.cz/android/netmonster/nm23002.csv.

3.2.3.4.2. Třída DatabazeLoad

Třída *DatabazeLoad* slouží k získávání dat z databáze. Obsahuje dvě metody: *vyberdatabaze()* a *databaze()*. Prvně jmenovaná slouží k výběru správné databáze dle aktuálního MCC+MNC. Druhá jmenovaná metoda přijímá jako parametr tři proměnné: Cell ID, LAC, a předchozí Cell ID.

Metoda *databaze()* používá ke čtení CSV databází metodu *readNext()* třídy *CSVReader*. Tato třída nebyla vytvořena autorem této práce, nýbrž byla přebrána z [32], kde je poskytována pod svobodnou licencí Apache, která umožňuje její použití bez modifikace a se zachováním autorství.

Po zavolání metody *databaze()* je nedříve rozhodnuto, zdali se aktuální a předchozí CellID liší. Pokud ano, tak je prohledána databáze, ve které se v případě sítí GSM a UMTS hledá shoda Cell ID ve čtvrtém sloupci a LAC v pátém sloupci databáze. V případě sítě LTE je hledána shoda eNode-B ID v šestém sloupci a příznaku 4G v prvním sloupci. Důvod, proč není v síti LTE testována shoda eNode-B a TAC, je ten, že v databázi u některých buněk jejich TAC chybí. V případě nalezení shody, jsou z databáze pro konkrétní buňku převzaty pole 6, 7 a 8. Tyto pole reprezentují: zeměpisnou šířku, délku a slovní popis umístění. Pokud není shoda nalezena, je jako slovní popis umístění použita věta: *„Buňka není v databázi“*.

3.2.3.4.3. Třída Lokace

Tato třída obsahuje dvě zásadní funkce. První funkcí je obsluha GPS v telefonu a druhou je výpočet vzdálenosti mezi telefonem a buňkou. Pro obsluhu GPS je zde použit manažer *LocationManager*, což je systémová služba, pomocí které mohou aplikace přistupovat k poloze telefonu. Tento manažer se volá pomocí metody *requestLocationUpdates()*. Tato metoda přijímá celkem čtyři parametry: 1. poskytovatele polohy (v případě této aplikace poskytovatel *GPS_PROVIDER*), 2. minimální časový rozestup mezi dvěma poskytnutími polohy, 3. minimální vzdálenost, které musí být překročena mezi dvěma poskytnutými polohami, 4. objekt s naslouchačem. Naslouchač funguje stejně jako v případě třídy *ZjisteniSignalu*, tedy vždy když zaznamená změnu polohy, je informace o této poloze uložena do objektu *lokace* typu *Location*. Naslouchač umí také naslouchat změnám stavu GPS, konkrétně tedy zdali je GPS povoleno či nikoliv a podle toho vykonat definované příkazy.

Pro práci se souřadnicemi buněk je vytvořen objekt *lokace1* typu *Location*, do kterého jsou pomocí metod *setLatitude()* a *setLongitude()* vloženy souřadnice získané z databáze buněk. Vzdálenost v metrech mezi lokacemi uloženými v objektech *lokace* a *lokace1* je následné vypočítána metodou *distanceTo()* třídy *Location* a uložena do proměnné vzdalenostm.

Metoda *distanceTo()* používá pro určení vzdálenosti referenční elipsoid vojenského systému WGS 84 (World Geodetic System). V systému Android je tedy matematicky namodelován tento elipsoid, na který jsou následně vloženy dva body, odpovídající zadaným souřadnicím a jejich vzdálenost je poté určena, jako vzdálenost mezi těmito body po povrchu elipsoidu. Více informací o systému WGS 84 je možné nalézt například v [33].

Ukázka kódu metody *vzdalenost()* pro zjištění vzdálenosti:

```
public static void vzdalenost(){
            try{

if(!NetInfo.lokace.isProviderEnabled(LocationManager.GPS_PROVIDER))
                {vzdalenostm="GPS není zapnuto.";}
                else {
            lokace1.setLatitude(Double.valueOf(souradnice1));
            lokace1.setLongitude(Double.valueOf(souradnice2));

if(lokace!=null&&NetInfo.lokace.isProviderEnabled(LocationManager.GPS_PRO
VIDER)){
                vzdalenostm="";

            vzdalenostm=
String.valueOf((int)lokace.distanceTo(lokace1))+" m";}

if(!NetInfo.lokace.isProviderEnabled(LocationManager.GPS_PROVIDER))
            {vzdalenostm="GPS není zapnuto.";}
            }}catch (IllegalArgumentException e)
            {
                vzdalenostm="Vzdálenost nelze určit.";
            }
                        catch (NullPointerException e )
            {
                vzdalenostm="Vzdálenost nelze určit.";
            }}
```

3.2.3.4.4. Třída Vypis

V třídě *Vypis* jsou definovány tři metody, které slouží pro sestavení výpisu zjištěných informací a následné zobrazení v textových polích *TextView*. První metoda se jmenuje *vypisAktualni()* a je určená k výpisu informací o aktuální buňce v aktivitě *NetInfo*. Vzhledem k rozdílným parametrům u jednotlivých typů sítí, je zde pro každý typ sítě definována vlastní podoba výpisu. Před sestavením výpisu tedy probíhá rozhodování podle typu sítě pomocí přepínače switch.

Druhá metoda se jmenuje *inicializace()* a je volána při inicializaci výpisu seznamu okolních buněk. Obsahuje pouze prostý nápis *„Načítání“*.

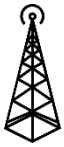
Třetí metoda s názvem *vypisOkolni()* je určena k výpisu informací o okolních buňkách v aktivitě *OkolniBunkyAktivita*. Metoda obsahuje rozhodovací mechanismy, na jejichž základě je možné vypsat upozornění, že nebyly nalezeny žádné okolní buňky, případně, že je zobrazení možné pouze v síti GSM (pokud se telefon nachází v jiné síti).

3.3. Spouštění aplikace

V následujících kapitolách již bude podrobně popsána výsledná podoba běžící aplikace.

3.3.1. Ikona

Aplikace používá grafickou ikonu, která se zobrazuje v Launcheru (grafickém seznamu aplikací po stisknutí tlačítka Home na mobilním telefonu). Pro tuto ikonu byl použit volně šiřitelný Clipart z [34], který byl dále upraven a přizpůsoben pro zobrazení v mobilním telefonu.



Obrázek 9 - Zdrojový Clipart, převzato z [35]

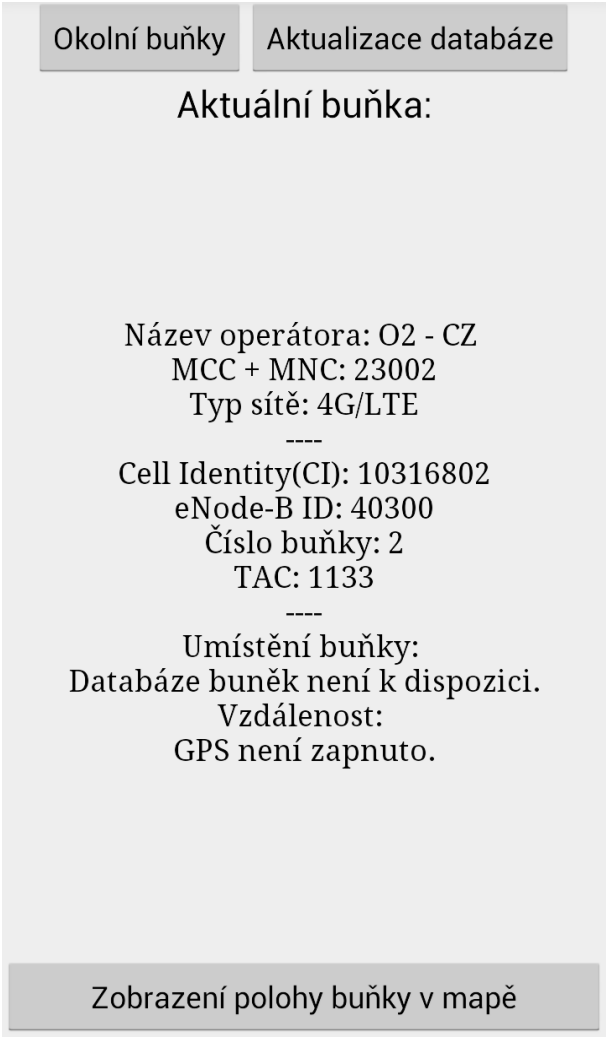
Pro lepší vzhled byl klipart umístěn do kruhu s bílým pozadí a dále byly vytvořeny různě velké verze, pro různé rozlišení obrazovek mobilních telefonů. Všechny tyto verze ikony jsou umístěny v instalačním souboru a operační systém si sám určuje, kterou velikostní verzi použije.

Obrázek 10 - Výsledná podoba ikony

Obrázek 11 - Zobrazení ikony aplikace v Launcheru

3.3.2. Úvodní obrazovka

Po stisknutí ikony, dojde ke spuštění aplikace a zobrazení úvodní obrazovky, kterou představuje aktivita NetInfo.

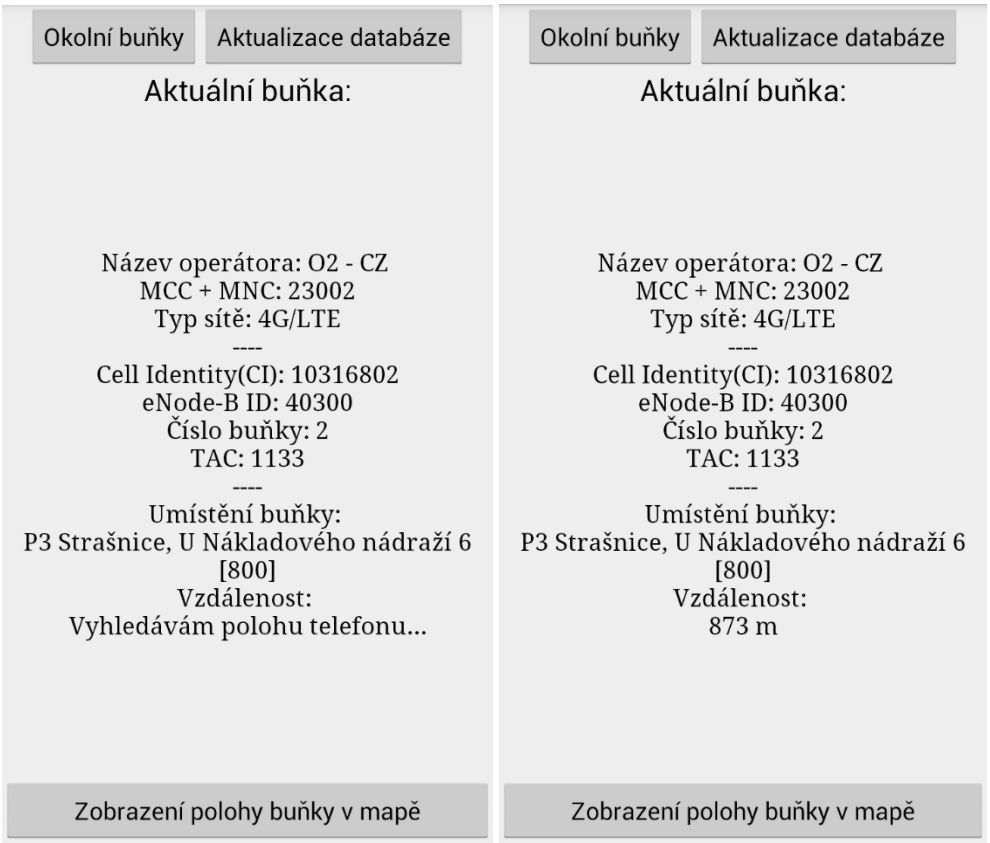


Obrázek 12 - Úvodní strana poprvé spuštěné aplikace

Na úvodní obrazovce aplikace se zobrazují informace o aktuální buňce. První blok tvoří obecné informace o aktuální síti. První položkou je název operátora. Zde je nutné podotknout, že se nejedná o název provozovatele sítě, ve které jsme registrováni, nýbrž se jedná o název mobilního operátora, jehož sim kartu používáme. Pokud tedy využíváme služeb virtuálního operátora, na tomto místě uvidíme jeho název, například Tesco Mobile, ale MCC+MNC (viz kapitola 3.1.3) uvedené níže, bude odpovídat síti operátora O2-CZ. V případě využívání služeb některého ze tří operátorů v ČR, kteří zároveň provozují fyzickou síť, je MCC+MNC kód odpovídající názvu operátora. Poslední položkou tohoto bloku je typ sítě. Aplikace umí rozlišovat nejen mezi základními typy sítí (tedy GSM, UMTS a LTE) ale zároveň dokáže určit, jaká je v daném typu sítě k dispozici technologie pro přenos dat. V GSM síti je tedy rozlišováno, zdali je k dispozici GPRS nebo EDGE, v síti UMTS pak aplikace signalizuje dostupnost technologie HSPA/HSPA+. Pro lepší srozumitelnost laickým uživatelům, aplikace zobrazuje k danému typu sítě i její generaci. Byť u sítě LTE není označení 4G přesné, díky mediálním kampaním, je u široké veřejnosti toto označení zažité, a tak i aplikace tento trend respektuje.

Druhý blok představuje informace o aktuální buňce, ke které je mobilní telefon přihlášen. Podrobné informace o těchto údajích jsou uvedeny v kapitole 3.1 .

Poslední blok je tvořen informacemi o poloze aktuální buňky. Umístění buňky je převzato z databáze GSMwebu. První údaj označuje okres a město, respektive město a městkou část, druhý poté ulici a číslo popisné, případně slovní popis pokud není místo nijak označené (např. příhradový stožár uprostřed pole). V případě sítě LTE je dispozici i frekvence, na které je daná buňka provozována. V posledním poli se zobrazuje vzdálenost mobilního telefonu od buňky v metrech na základě výpočtu ze souřadnic buňky v databázi a souřadnic polohy telefonu ze systému GPS. Bližší popis algoritmu výpočtu je v kapitole 3.2.3.4.3. Pokud není zaměření polohy pomocí systému GPS v mobilním telefonu povoleno, je uživatel o tomto stavu informován. V případě, že je systém GPS povolen, avšak ještě není k dispozici přesná poloha, zobrazí se oznámení o vyhledávání polohy. Po nalezení polohy je zobrazena vzdálenost v metrech.



Obrázek 13 a 14 - Zobrazení vzdálenosti; vyhledávání (vlevo) a nalezení polohy telefonu (vpravo)

V poli Umístění buňky je při prvním spuštění zobrazeno oznámení, že databáze buněk není k dispozici. Chybějící databázi je možné nechat automaticky stáhnout klepnutím na tlačítko *Aktualizace databáze* v horní části obrazovky. Tímto tlačítkem lze taktéž kdykoliv stáhnout nejnovější verzi používané databáze. Při stahování aplikace je však potřeba vzít na vědomí, že velikost databáze pro jednoho operátora je zhruba 3 MB, a proto její stažení pomocí pomalejšího připojení může trvat déle. Po dobu stahování databáze je tlačítko neaktivní a o úspěšném, případně neúspěšném stažení, je uživatel informován pomocí dialogu.

Obrázek 15 a 16 - Stahování databáze buněk; stahování databáze (vlevo) a úspěšně provedená aktualizace (vpravo)

Databáze se ukládá do adresáře NetInfo na externím uložišti. V průběhu vytváření aplikace byla diskutována i možnost ukládání databází pouze do interní paměti telefonu, avšak z důvodu poměrně velké velikosti a také z důvodu možnosti umístění databáze uživatelem ručně, bez nutnosti stahování pomocí aplikace, bylo rozhodnuto o ukládání na externí uložiště. Autor práce si je vědom možné absence paměťové karty a tím pádem nemožnosti využívání databáze. Avšak jelikož moderní telefony již fyzickou paměťovou kartu nevyužívají a místo ní mají vyhrazenou velkou část své interní paměti pro takzvanou emulovanou paměťovou kartu, na kterou samozřejmě aplikace také umí databázi ukládat, není toto autorem považováno za nedostatek. U starých telefonů, kde není k dispozici fyzická ani emulovaná paměťová karta, by bylo ukládání do systémové paměti problematické z důvodu velice omezené kapacity.

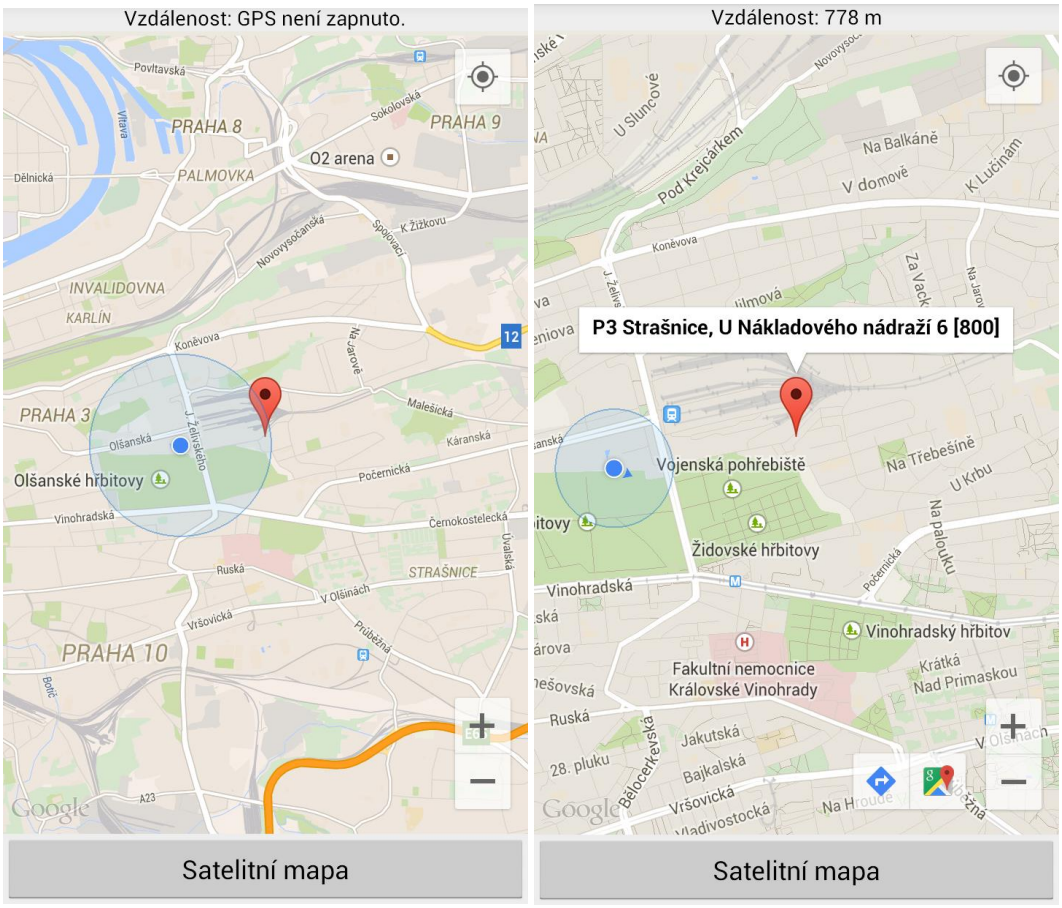
V případě, že telefon není připojen k žádné síti, je zobrazeno oznámení. Stejně tak aplikace počítá i s možností, že aktuální buňka nebude zaznamenaná v databázi. V takovém případě, je uživatel informován, že popis umístění není k dispozici a samozřejmě také, že není možné určit vzdálenost.

Obrázek 17 a 18 - Bez připojení k síti (vlevo) a nenalezení aktuální buňky v databázi (vpravo)

3.3.3. Zobrazení v mapě

Tlačítko umístěné na spodním okraji obrazovky umožňuje zobrazení polohy aktuální buňky v mapových podkladech Google Maps společnosti Google. Zde je nutné podotknout, že pro zobrazení mapy je potřeba připojení k internetu. Po stisknutí tlačítka *Zobrazení polohy buňky v mapě*, dojde k zobrazení a vycentrování, kde poloha aktuální buňky je umístěna ve středu mapy a je znázorněna červeným terčem. Modrým terčem je naopak znázorněna poloha telefonu. Po klepnutí na červený terčík reprezentující polohu buňky je zobrazen popis umístění.

Standardně je použito měřítko mapy 1:20 000, v případě potřeby, může uživatel toto měřítko změnit prostřednictvím ovládacích prvků v pravém dolním rohu nebo pomocí gesta. Standardní zobrazení mapy je také možno změnit na satelitní zobrazení pomocí tlačítka v dolním okraji obrazovky. V horním okraji obrazovky je dispozici informace o vzdálenosti buňky. Pomocí dvou ikon vedle tlačítka ovládání měřítka mapy, je také možné zobrazení v plnohodnotné aplikaci Google Maps, případně naplánování cesty k buňce pomocí aplikace Google Navigace.

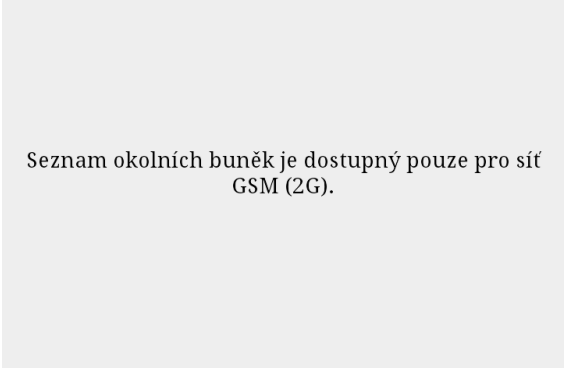


Obrázek 19 a 20 - Zobrazení polohy buňky po přesunu z úvodní obrazovky (vlevo) a zobrazení slovního popisu umístění buňky (vpravo)

Obrázek 21 - Satelitní zobrazení mapy

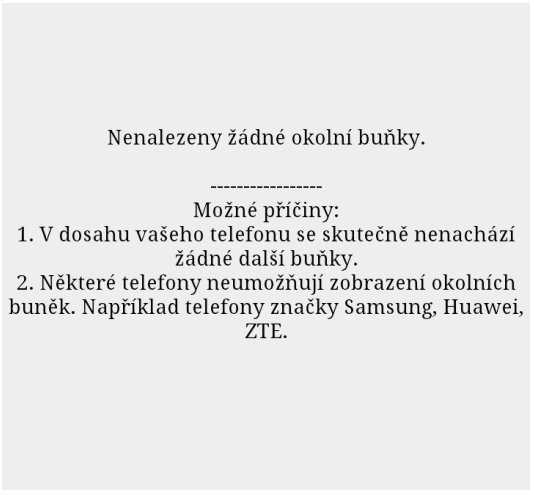
3.3.4. Zobrazení okolních buněk

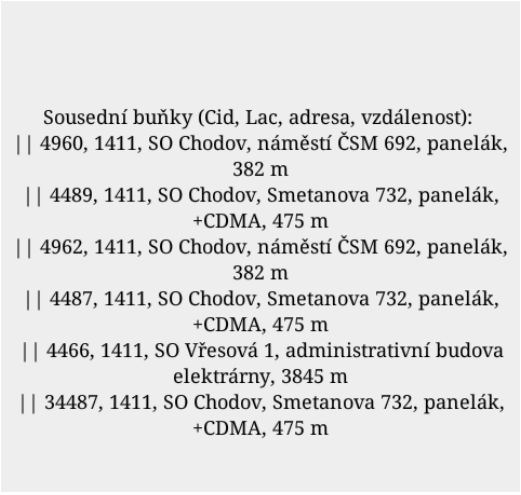
Posledním tlačítkem na úvodní obrazovce je tlačítko *Okolní buňky*. Jak již název napovídá, po stisknutí tohoto tlačítka se lze dostat do výpisu veškerých okolních buněk, které je telefon schopný detekovat. Zobrazení okolních buněk je však limitováno několika omezeními. Prvním omezením je možnost zobrazení okolních buněk pouze v síti GSM. Při pokusu o zobrazení okolních buněk v ostatních typech sítí je uživatel o tomto informován. Příčina tohoto omezení je blíže vysvětlena v kapitole 3.2.3.2.2.



Obrázek 22 - Pokus o zobrazení okolních buněk v síti LTE

Druhé omezení spočívá v tom, že někteří výrobci mobilních telefonů záměrně neumožnují zobrazení okolních buněk, a tudíž aplikace nemá žádnou možnost jejich získání ze systému. V tomto případě je uživateli zobrazena informace o nenalezení žádných okolních buněk a možných příčinách. Více o tomto omezení je možno nalézt v kapitole 4.3. o testování aplikace.





Obrázek 23 a 24 - Nenalezení okolních buněk (vlevo) a výpis okolních buněk (vpravo)

Pokud se telefon nachází v síti GSM a okolní buňky jsou aplikací detekovatelné, je zobrazen seznam všech těchto buněk. U každé z nich je uvedeno CellID, LAC, slovní popis umístění a vzdálenost.

4. Testování aplikace

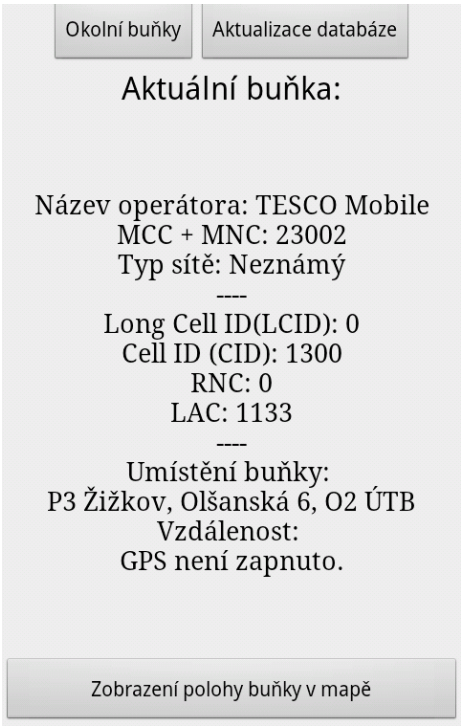
Aplikace byla primárně vyvíjena pro mobilní telefon ***Huawei Ascend P2*** s operačním systémem Android ***4.1 Jelly Bean***, který dokáže pracovat v sítích GSM, UMTS i LTE. V průběhu a po ukončení vývoje byla aplikace testována na následujících dalších mobilních telefonech:

Výrobce	Typ	Verze operačního systému
ZTE	Blade	2.3 Gingerbread (Cyanogenmod 7)
ZTE	Grand X IN	4.0 Ice Cream Sandwich
HTC	One S	4.1 Jelly Bean
LG	L 70	4.4 KitKat
Gigabyte	GSmart Roma R2	4.2 Jelly Bean
LG	G2	4.4 KitKat

Na všech těchto zařízeních jde aplikace spustit, avšak na některých telefonech má odlišné chování způsobené zásahem výrobce do operačního systému, který se poté liší od norem společnosti Google. Tyto odlišnosti si přiblížíme v následujících kapitolách.

4.1. Určení typu sítě

Rozlišování mezi jednotlivými typy sítí funguje bez problémů na všech testovaných telefonech mimo *ZTE Blade*. V telefonu *ZTE Blade* vrací operační systém na dotaz aplikace vždy neznámý typ sítě. Jelikož dle [36] je metoda, která typ sítě obstarává definována od API 1, není zde na vinně stará verze operačního systému. Pravděpodobně je to způsobeno chybou v operačním systému, který v tomto telefonu není originální od výrobce, nýbrž se jedná o alternativní ROM od uskupení CyanogenMod. V tomto případě tedy aplikace vypíše typ sítě jako neznámý a použije rozvržení jako pro síť UMTS. Zobrazí tedy i pole LCID a RNC, které v případě, že nebudou tyto údaje k dispozici, jelikož se bude jednat o síť GSM, zůstanou s hodnotou 0.



Obrázek 25 – Aplikace běžící na telefonu ZTE Blade

4.2. Jednotky síly signálu

Pro zjištění síly signálu využívá aplikace metodu *getGsmSignalStrength()* třídy *SignalStrength*. Dle definice této metody na webu Android Developers [37], vrací tato metoda sílu signálu v jednotkách ASU. Dle této definice se aplikace chová na všech testovaných telefonech mimo Huawei Ascend P2, kde metoda z nepochopitelných důvodů v rozporu s definicí vrací sílu signálu v jednotkách dBm. Po zjištění tohoto problému, bylo v kódu aplikace ošetřeno rozpoznávání jednotek a byl implementován vzájemný přepočet mezi jednotkami. Jelikož aplikace zobrazuje sílu signálu v obou jednotkách, není z uživatelského pohledu možné zjistit, ve kterých jednotkách poskytuje tuto hodnotu operační systém. Tato funkce je tak ve finální verzi funkční na všech testovaných telefonech.

4.3. Zobrazení okolních buněk

Zobrazení okolních buněk je z testovaných telefonů funkční pouze na modelech LG L70, LG G2, HTC One S a Gigabyte GSmart Roma 2. U ostatních testovaných mobilních telefonů se jeví, že telefon nedetekuje žádné okolní buňky. Tento problém je způsoben výrobci telefonů, kteří upravují operační systém Android k obrazu svému a možnost zobrazení okolních buněk záměrně zakáží. Mezi výrobce, kteří ve svých telefonech tuto možnost zakazují, patří například: Samsung (dle [38] údajně veškeré modely), Huawei (minimálně model Ascend P2, pravděpodobně však také všechny modely) a ZTE (nefunkčnost vyzkoušena na modelech Blade a Grand X In). Naopak za bezproblémové lze považovat telefony značky LG.

4.4. Běh aplikace na dual SIM telefonu

Aplikace byla taktéž testována na dual SIM telefonu, kterým byl Gigabyte GSmart Roma 2. V tomto konkrétním telefonu je funkcionalita provozu dvou SIM karet implementována tak, že jeden slot na SIM je považován za primární a druhý za sekundární. Aplikace na tomto telefonu funguje tak, že pokud máme aktivní primární SIM kartu a ta je připojena do sítě, veškeré údaje, které aplikace zobrazuje, jsou relevantní pouze k této primární SIM kartě. Pokud máme aktivní pouze sekundární SIM kartu, není aplikace schopna zobrazit žádné údaje a vypíše, že telefon není připojen k žádné síti. Pokud máme v provozu obě SIM karty, aplikace se chová, jako kdyby telefon obsahoval pouze jednu SIM a to primární SIM a pro tuto zobrazuje informace.

U dual SIM telefonů běžících na systému Android je obecný problém v nepředvídatelnosti jejich chování. Systém Android totiž oficiálně podporuje dual SIM telefony až od nejnovější verze 5.1 [39]. V dřívějších verzích tedy byla implementace této funkcionality čistě na každém výrobci. Výrobci těchto telefonů však často poskytují v lepším případě velice špatnou dokumentaci, v horším případě vůbec žádnou. Pro vývojáře je tak velice těžké vyvinout aplikaci, která by fungovala na dual SIM telefonech různých výrobců.

4.5. Vyhodnocení testování

V níže uvedené tabulce jsou přehledně shrnuty poznatky získané při testování.

Mobilní telefon	Rozpoznání sítě	Zobrazení okolních buněk	Práce s mapou	Informace o aktuální buňce
Huawei Ascend P2	✓	×	✓	✓
ZTE Blade	×	×	✓	✓
ZTE Grand X In	✓	×	✓	✓
LG L70	✓	✓	✓	✓
LG G2	✓	✓	✓	✓
Gigabyte GSmart Roma 2	✓	✓	✓	✓
HTC One S	✓	✓	✓	✓

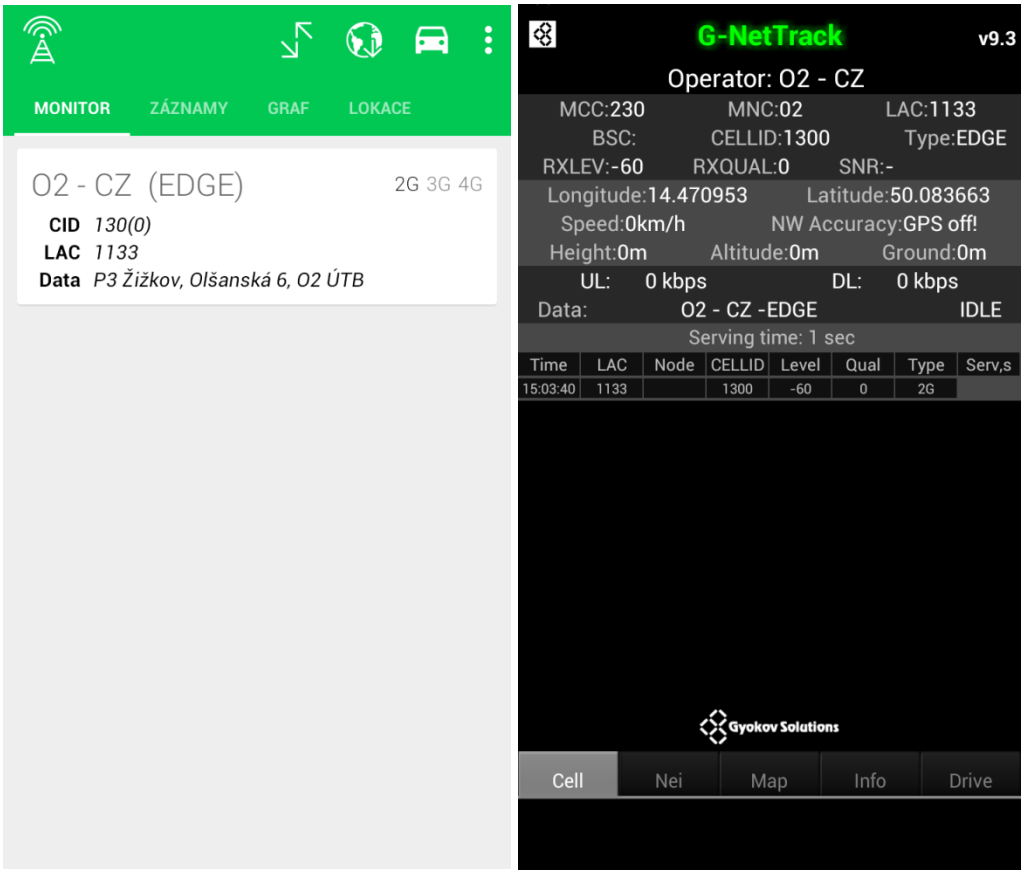
Jako 100% funkční je možné aplikaci označit na telefonech LG L70, LG G2, Gigabyte GSmart Roma 2 a HTC One S. S výjimkou zobrazení okolních buněk je aplikace plně funkční také na modelech Huawei Ascend P2 a ZTE Grand X In. Naopak nejhorší funkčnost ze všech testovaných telefonů má aplikace na telefonu ZTE Blade. V tomto telefonu není možné zobrazit okolní buňky ani typ sítě. Všechny ostatní funkce jsou však i na tomto modelu plně k dispozici.

5. Obdobné aplikace na trhu

V největším obchodu s aplikacemi pro systém Android Google Play, je možné nalézt několik dalších aplikací s podobným zaměřením. Z těch nejvýznamnějších jsou to především NetMonster a G-NetTrack.

NetMonster je česká aplikace pocházející z dílny vývojářského studia Brightify. NetMonster stejně jako NetInfo využívá stejnou databázi buněk z webu Gsmweb, kterou si aplikace dokáže sama stahovat i aktualizovat. Mimo základních věcí, jako jsou informace o aktuální a okolních buňkách a zobrazení buňky v mapě, dokáže tato aplikace též ukládat do záznamu všechny nalezené buňky. Výhodou aplikace NetMonster je především propracovaná grafická podoba a široká paleta nastavení zobrazovaných údajů. Mezi nevýhody této aplikace patří nefunkčnost na starších telefonech (aplikace vyžaduje minimálně verzi 4.0) a problém se zobrazováním síly signálu (testováno na telefonu Huawei Ascend P2, kde v této aplikaci zobrazení síly signálu nefunguje).

G-Net Track je aplikace od vývojářského studia Gyokov Solutions. Tato aplikace je zaměřena především na zaznamenávání pohybu mezi buňkami. Největší výhodou této aplikace je výborné využití Google Maps. V těchto mapách dokáže aplikace zobrazovat velké spektrum informací, například vykreslení trasy pohybu telefonu, kde pomocí barvy zobrazuje i sílu signálu naměřenou v daném místě. Další velkou výhodou je podpora Androidu 2.2 a novějších. Nevýhodou může být poměrně nepřehledné grafické uspořádání a nutnost manuálního stažení databáze a následně její importace do aplikace.



Obrázek 26 a 27 - Aplikace NetMonster (vlevo) a aplikace G-NetTrack (vpravo)

Aplikaci NetMonster je možné stáhnout z [40] a G-NetTrack z [41].

6. Vyhodnocení

Cílem této práce bylo vytvořit funkční aplikaci pro systém Android, která zobrazuje parametry buněk v síti GSM.

V první, teoretické části práce byly představeny základní informace o systému Android, zejména jeho stručná historie, architektura a nástroje pro vývoj aplikací. Dále zde také byla podrobněji představena struktura obecné aplikace pro tento systém. V neposlední řadě byl v této kapitole také představen pojem buňková síť a také podoby radiové přístupové sítě (RAN) pro jednotlivé druhy sítí provozované v ČR.

Navazující praktická část se již zabývala vlastním vývojem aplikace NetInfo. Nejprve byl představen význam jednotlivých parametrů, které aplikace poskytuje. Dále již navazovala kapitola věnující se vlastnímu vývoji, kde byl nejdříve představen vývoj grafického prostředí a dále také podrobný popis zdrojového kódu jazyka Java. Následující kapitola čtenáře podrobně seznámila s reálně běžící aplikací na konkrétním mobilním telefonu včetně bohaté obrázkové dokumentace. Předposlední kapitola byla zaměřena na prezentaci výsledků testování na různých mobilních telefonech a v poslední kapitole byly stručně představeny dvě konkurenční aplikace s podobným zaměřením.

Výsledná aplikace splňuje všechny podmínky zadání a disponuje i částečnou funkcionalitou (vše mimo zobrazení okolních buněk a síly signálu) v sítích UMTS a LTE. Funkcionalita v těchto sítích nebyla v zadání požadována a byla implementována nad jeho rámec v rámci většího rozsahu použitelnosti vyvíjené aplikace.

O umístění aplikace do nabídky obchodu Google Play zatím nebylo rozhodnuto. Vzhledem k částce 25 USD, která se musí uhradit pro vytvoření vývojářského účtu v tomto obchodě, bude zvážena možnost zveřejnění na některém z alternativních obchodů. Aplikace bude v každém případě poskytována zcela zdarma.

V rámci budoucího vývoje bude zvážena možnost přidání dalších funkcí, například možnost logování nalezených buněk, zobrazení více parametrů zejména v síti LTE nebo propracovanější grafické rozhraní.

Tato práce mi přinesla zejména obrovské prohloubení znalosti programování v jazyce Java a získání prvních cenných zkušeností s vývojem softwaru pro mobilní telefony. Výchozí znalosti pro vývoj této aplikace tvořily pouze znalosti z předmětu Programování a následné samostudium.

Doufám, že vytvořená aplikace přinese užitek mnohým uživatelům a to jak těm, kteří se v problematice mobilních sítí pohybují, tak i obyčejným uživatelům, které zajímá, kterou buňku jejich mobilní telefon aktuálně používá.

7. Reference

1. Jak vypadá Android uvnitř aneb co je ROM, kernel, bootloader a další? [online]. [cit. 2015-04-03]. Dostupné z: http://androidmarket.cz/android/jak-vypada-android-uvnitr-aneb-co-je-rom-kernel-bootloader-a-dalsi/
2. *Web o operačním systému Android* [online]. [cit. 2015-04-03]. Dostupné z: http://home.zcu.cz/~hodlova/
3. Update: Oracle sues Google over Java use in Android. [online]. [cit. 2015-04-03]. Dostupné z: http://www.computerworld.com/article/2515001/mobile-wireless/update--oracle-sues-google-over-java-use-in-android.html
4. *Android dostane rýchlejší runtime už v najbližšej verzii* [online]. [cit. 2015-04-06]. Dostupné z: http://dsl.sk/article.php?article=15739
5. Android-system-architecture. [online]. [cit. 2015-04-03]. Dostupné z: http://androidmarket.cz/wp-content/uploads/2011/12/Android-system-architecture.jpg
6. *Google Developers: Android NDK* [online]. [cit. 2015-04-20]. Dostupné z: https://developer.android.com/tools/sdk/ndk/index.html
7. GRANT, A. Android 4. In: *Průvodce programováním mobilních aplikací*. Brno: Computer Press, 2013, s. 585-98. ISBN 978-80-2513-782-6.
8. *Google Developers: Activities* [online]. [cit. 2015-04-05]. Dostupné z: http://developer.android.com/guide/components/activities.html
9. *Google Developers: Services* [online]. [cit. 2015-04-26]. Dostupné z: http://developer.android.com/guide/components/services.html
10. *Google Developers: Content Providers* [online]. [cit. 2015-04-22]. Dostupné z: http://developer.android.com/guide/topics/providers/content-providers.html
11. *Google Developers: Intents and Intent Filters* [online]. [cit. 2015-04-22]. Dostupné z: http://developer.android.com/guide/components/intents-filters.html
12. Google Developers: uses-sdk. [online]. [cit. 2015-04-08]. Dostupné z: http://developer.android.com/guide/topics/manifest/uses-sdk-element.html
13. Google Developers: Manifest.permission. [online]. [cit. 2015-04-10]. Dostupné z: http://developer.android.com/reference/android/Manifest.permission.html
14. Princip buňkového systému. [online]. [cit. 2015-04-09]. Dostupné z: http://tomas.richtr.cz/mobil/bunk-princip.htm
15. *sektor1* [online]. [cit. 2015-04-06]. Dostupné z: http://tomas.richtr.cz/mobil/images/sektor1.gif
16. *sektor2* [online]. [cit. 2015-04-06]. Dostupné z: http://tomas.richtr.cz/mobil/images/sektor2.gif
17. UNZEITIG, L. *MODEL POKRYTÍ ÚZEMÍ BUŇKOVÉ SÍTĚ*. [cit. 2015-04-09]. Dostupné z: https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=7534
18. PRAVDA, I. Základy mobilních sítí. s. 11 [cit. 2015-04-06]. Dostupné z: http://data.cedupoint.cz/oppa_e-learning/2_KME/057.pdf
19. *Femtobuňky Vodafone v ČR!* [online]. [cit. 2015-04-06]. Dostupné z: http://www.femtobunka.cz/2011/06/femtobunky-vodafone-v-cr.html

20. *GSM - The Base Station Subsystem(BSS)* [online]. [cit. 2015-04-10]. Dostupné z: http://www.tutorialspoint.com/gsm/gsm_base_station_subsystem.htm
21. *Radiová přístupová síť - RAN* [online]. [cit. 2015-04-05]. Dostupné z: http://www.umts.wz.cz/Mob_radio_site_3G/RAN.htm
22. *UMTS and LTE Design* [online]. [cit. 2015-04-10]. Dostupné z: https://www.linkedin.com/groups/what-is-difference-between-25G-1722457.S.5932782451788103680
23. *LTE Network Infrastructure and Elements* [online]. [cit. 2015-04-05]. Dostupné z: https://sites.google.com/site/lteencyclopedia/lte-network-infrastructure-and-elements#TOC-2.2-eNode-Bs:-The-Single-E-UTRAN-Node
24. *Long CellID vs. short Cell ID* [online]. [cit. 2015-04-09]. Dostupné z: http://wiki.opencellid.org/wiki/FAQ#Long_CellID_vs._short_Cell_ID
25. *3GPP LTE/LTE-A Standards* [online]. [cit. 2015-04-05]. Dostupné z: https://www.linkedin.com/groups/when-eNB-has-multiple-cells-1180727.S.221912781
26. *Tracking area codes* [online]. [cit. 2015-04-06]. Dostupné z: http://lteuniversity.com/ask_the_expert/f/59/t/2566.aspx
27. *Mobile country code* [online]. [cit. 2015-04-06]. Dostupné z: http://en.wikipedia.org/wiki/Mobile_country_code
28. ETSI TS 127 007. s. 82 [cit. 2015-04-08]. Dostupné z: http://www.etsi.org/deliver/etsi_ts/127000_127099/127007/08.05.00_60/ts_127007v080500p.pdf
29. *Google Developers: TelephonyManager* [online]. [cit. 2015-04-29]. Dostupné z: http://developer.android.com/reference/android/telephony/TelephonyManager.html
30. *Google Developers: GsmCellLocation* [online]. [cit. 2015-04-15]. Dostupné z: http://developer.android.com/reference/android/telephony/gsm/GsmCellLocation.html
31. *Google Developers: TelephonyManager* [online]. [cit. 2015-05-01]. Dostupné z: http://developer.android.com/reference/android/telephony/TelephonyManager.html#NETWORK_TYPE_1xRTT
32. *CSVReader.java* [online]. [cit. 2015-01-20]. Dostupné z: https://code.google.com/p/secrets-for-android/source/browse/trunk/src/au/com/bytecode/opencsv/CSVReader.java
33. JURKINA, M. I. a M. PICK. *Numerické výpočty ve světovém geodetickém referenčním systému 1984 (WGS84)*. [cit. 2015-05-02]. Dostupné z: http://www.army.cz/images/id_7001_8000/7162/VGO_1_2006_priloha_2.pdf
34. *aerial%20clipart* [online]. [cit. 2015-01-15]. Dostupné z: http://www.clipartpanda.com/clipart_images/aerial-20clipart-24965037
35. *Antenna_Tower_clip_art_hight* [online]. [cit. 2015-01-15]. Dostupné z: http://images.clipartpanda.com/transmitter-clipart-aerial-clipart-free-vector-antenna-tower-clip-art_116449_Antenna_Tower_clip_art_hight.png
36. *Google Developers: TelephonyManager* [online]. [cit. 2015-05-01]. Dostupné z: http://developer.android.com/reference/android/telephony/TelephonyManager.html#getNetworkType%28%29
37. *Google Developers: SignalStrength* [online]. [cit. 2015-04-27]. Dostupné z: http://developer.android.com/reference/android/telephony/SignalStrength.html#getGsmSignalStrength%28%29

38. *getting neighboring cells information using android Samsung galaxy smartphones* [online]. [cit. 2015-04-22]. Dostupné z: http://stackoverflow.com/questions/20444703/getting-neighboring-cells-information-using-android-samsung-galaxy-smartphones
39. *Android 5.1 SDK released, new APIs for dual-SIM devices and carrier apps* [online]. [cit. 2015-04-18]. Dostupné z: http://www.androidcentral.com/android-51-sdk-released-new-apis-dual-sim-devices-and-carrier-apps
40. *NetMonster* [online]. [cit. 2015-04-25]. Dostupné z: https://play.google.com/store/apps/details?id=cz.mroczis.netmonster
41. *G-NetTrack* [online]. [cit. 2015-04-25]. Dostupné z: https://play.google.com/store/apps/details?id=com.gyokovsolutions.gnettrack