VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

APLIKACE PRO ZAZNAMENÁNÍ A MEŘENÍ TRASY POMOCÍ GPS SOUŘADNIC

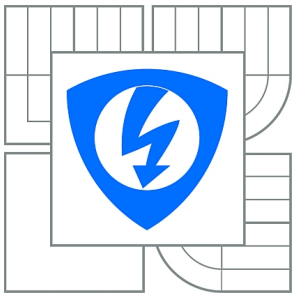
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE MAREK ORGOŇ
AUTHOR

BRNO 2012

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF TELECOMMUNICATIONS

APLIKACE PRO ZAZNAMENÁNÍ A MEŘENÍ TRASY POMOCÍ GPS SOUŘADNIC

APPLICATION FOR RECORDING AND MEASUREMENT OF THE ROUTE USING GPS COORDINATES

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

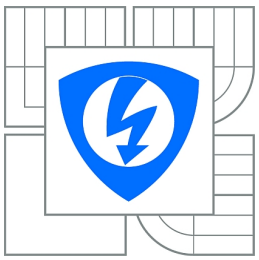
AUTOR PRÁCE — MAREK ORGOŇ

AUTHOR

VEDOUCÍ PRÁCE — Ing. JURAJ SZŐCS

SUPERVISOR

BRNO 2012



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Bakalářská práce

bakalářský studijní obor
Teleinformatika

Student:	Marek Orgoň	***ID:***	125575
Ročník:	3	***Akademický rok:***	2011/2012

NÁZEV TÉMATU:

Aplikace pro zaznamenání a meření trasy pomocí GPS souřadnic

POKYNY PRO VYPRACOVÁNÍ:

Seznamte se s možnostmi vývoje aplikací pro mobilní telefony s OS Android. Navrhněte systém, který bude zaznamenávat a měřit trasu pomocí GPS souradnic. Systém bude sledovat aktuální pohyb a reagovat podle přednastavených hodnot. Aplikace bude graficky zobrazovat ujetou trasu v googlemaps pomocí Google Maps API.no.

DOPORUČENÁ LITERATURA:

[1] MEIER, R. Professional Android Application Development. Wrox, 2008. 432 s. ISBN: 978-0-470-34471-2.
[2] MURPHY, M.L. Beginning Android 2. Apress, 2010. 416 s. ISBN: 978-1-14302-2629-1.

Termín zadání:	6.2.2012	***Termín odevzdání:***	31.5.2012

Vedoucí práce: Ing. Juraj Szőcs
Konzultanti bakalářské práce:

prof. Ing. Kamil Vrba, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

ABSTRAKT

Tato bakalářská práce pojednává o vytváření aplikace pro mobilní systém Android, která bude řešit problematiku zaznamenávání trasy a její následné zobrazení na mapě. První část se zabývá samotným systémem Android a návrhem aplikací pro něj. Ve druhé části je řešen návrh a implementace aplikace pro zaznamenávání trasy a její zobrazení na mapě.

KLÍČOVÁ SLOVA

Android, GPS, Google Map, Google Map API

ABSTRACT

This barchelor thesis discusses the creation of mobile applications for Android, which will address the issue of recording the route and its subsequent display on the map. The first part deals with the actual system Android and design applications for it. The second part dealt with the design and implementation of applications for recording route and subsequent display on the map.)

KEYWORDS

Android, GPS, Google Map, Google Map API

ORGOŇ, Marek *Aplikace pro zaznamenání a měření trasy pomocí GPS souřadnic*: bakalářská práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2012. 49 s. Vedoucí práce byl Ing. Juraj Szőcs

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „Aplikace pro zaznamenání a měření trasy pomocí GPS souřadnic“ jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

Brno ....

(podpis autora)

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu bakalářské práce panu Ing. Juraji Szőcsovi za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Brno ....

(podpis autora)

OBSAH

SEZNAM OBRÁZKŮ

ÚVOD

Účelem této bakalářské práce je seznámení z možnostmi vývoje a vytvoření aplikace pro zaznamenání a měření trasy pomocí GPS. Aplikace bude schopna sledovat aktuální pohyb a podle přednastavených hodnot na něj reagovat. Dále bude schopna tento záznam zobrazit na mapě pomocí Google Maps API a ze zaznamenaných dat spočítá údaje o trase.

Aplikace bude vytvořena pro mobilní operační systém Android, tento systém se používá převážně na mobilních zařízeních a to konkrétně na mobilních telefonech a tabletech, lze jej také nalézt v chytrých televizích. Android je v současné době nejvíce se rozšiřující mobilní operační systém. Aplikace pro něj se programují v jazyce Java s množstvím doplňkových knihoven vytvořených pro tento systém. Většina prodávaných zařízeních s Androidem má systém dodáváný společností Google, která má na jeho vývoji největší podíl.

Vývoj aplikace pro mobilní systém je v určitých ohledech specifický. Tato zařízení mají omezený výpočetní výkon, který je v porovnání s klasickými počítači velmi nízký, proto se při vývoji zohledňuje výpočetní náročnost dané operace například před přesností výsledku. S tím souvisí i omezená kapacita baterie, která napájí dané zařízení. Pokud se jedná o mobilní telefon není žádoucí aby aplikace při korektním použití spotřebovala tolik energie, že dojde k vypnutí telefonu. Posledním z aspektů je malá velikost displeje a dotykové ovládání.

Aplikace tohoto typu se hojně užívají pro zaznamenání sportovních aktivit různých typů. V této práci bude aplikace vytvořena pro rekreační běh. Pro aktuální trasu uživatele bude zobrazovat aktuální pozici na mapě a aktuální informace (rychlost, uběhnutá vzdálenost...). Na základě záznamu vypočítá a zobrazí informace o trase (časy, rychlosti, výškový profil trasy, odhad spotřebované energie...). Na uložených záznamech umožní trénovat a zobrazovat informace o tréninku.

První kapitola této práce pojednává o systému Android, o jeho historii, vnitřní stavbě a programování pro tento systém. Věnuje se také geolokačním možnostem systému a způsobu použití Google Maps API.

Druhá kapitola pojednává o návrhu aplikace, její implementaci a optimalizaci a řešení problémů vyskytujících se při jejím programování.

1 ÚVOD DO SYSTÉMU ANDROID

1.1 Historie

Informace v této části byly převzaty ze zdroje [2]. V roce 2003 byla založena malá společnost Android Inc. Jejími zakladateli byli Andy Rubin, Rich Miner, Nick Sears a Chris White.

V roce 2005 tuto společnost koupil Google, někteří zaměstnanci během akvizice přešli pod Google. Andy Rubin se stal vedoucím týmu, který vyvíjel platformu pro mobilní zařízení postavenou na linuxovém jádře. Očekávalo se, že Google představí svůj vlastní telefon.

V roce 2007 byla založena Open Handset Alliance, nezisková skupina několika firem, jejímž cílem bylo vytvořit otevřené standardy pro mobilní zařízení. Zároveň byl také představen Android, otevřená platforma pro mobilní zařízení, postavená na linuxovém jádře verze 2.6. Google tím dal jasně najevo, že Android není platforma určená pouze pro jeho telefony.

V roce 2008 vyšel Android SDK 1.0, krátce potom začal prodej telefonu T-Mobile G1 vyráběného společností HTC, což byl první telefon běžící pod systémem Android.

1.2 Verze systému

Informace v této části byly převzaty ze zdrojů [2], [3] a [4]. Vývoj Androidu neustále pokračuje a nové verze vycházejí v rychlém sledu. Jsou označovány číslem podle kterého lze určovat hlavní a menší verze. Hlavní verze mají také kódové označení. Nejdůležitější je verze API, která určuje, co nového je v systému implementováno, případně co bylo opraveno. Tab. 1.1 ukazuje přehled všech verzí od počátku do konce roku 2011.

1.2.1 Telefonní a tabletové verze

Android byl od počátku vyvíjen se zaměřením na mobilní telefony. Až do verze 2.2 bylo dodržováno, že následující verze nahrazovala tu předchozí. Po verzi 2.2 byla zároveň vyvíjena verze 2.3 určená pro mobilní telefony a verze 3.0 určená výhradně pro tablety. Aplikace se tedy vyvíjely pro mobil i tablet zvlášť (aplikace pro mobil běžely i na tabletové verzi, byly pouze zvětšeny). Po verzích 2.3.x a 3.x byla vydána verze 4.0, která opět spojuje mobilní telefony s tablety. A stejně tak i aplikace jsou vyvíjeny jednou a na mobilních telefonech a tabletech se zobrazují různě.

Tab. 1.1: Verze systému Android

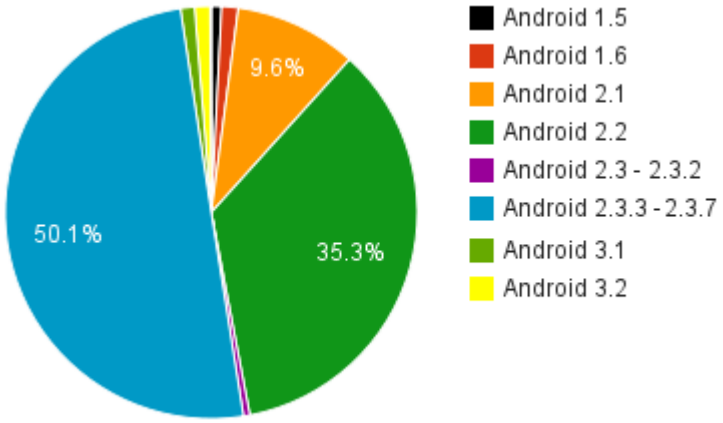
Verze systému	**Verze API**	**Kódové označení**
Android 4.0	14	ICE CREAM SANDWICH
Android 3.2	13	HONEYCOMB
Android 3.1.x	12	HONEYCOMB
Android 3.0.x	11	HONEYCOMB
Android 2.3.3	10	GINGERBREAD
Android 2.3	9	GINGERBREAD
Android 2.2.x	8	FROYO
Android 2.1.	7	ECLAIR
Android 2.0.1	6	ECLAIR
Android 2.0	5	ECLAIR
Android 1.6	4	DONUT
Android 1.5	3	CUPCAKE
Android 1.1	2	BASE
Android 1.0	1	BASE

1.2.2 Segmentovanost a reálné rozdělení verzí

Za největší problém Androidu se v současnosti považuje velká segmentovanost verzí Androidu běžícího na různých zařízeních. Výrobci často vydávají zařízení se starou verzí Androidu a během jeho prodeje nabízejí aktualizace se zpožděním nebo vůbec. Bohužel se stává pravidlem, že výrobce raději s novou verzí systému vydá nové zařízení, než aby aktualizoval současné. Google v současné době vyvíjí tlak na výrobce, aby aktualizovali svá zařízení alespoň 18 měsíců po jejich vydání. I přes to je však Android velmi segmentován.

Na obr. 1.1, který byl převzat ze zdroje [4], je zobrazeno zastoupení jednotlivých verzí Androidu, běžících na zařízeních, které přistupovaly na Android Market během 14ti denního období končícího 3. 11. 2011. Jak lze vidět, je používáno velké množství zařízení se staršími verzemi androidu. Verze 2.3 byla vydána již před rokem a její zastoupení je pouze kolem 50 % užívaných zařízení, což je pouze o 15 % než u verze 2.2, která byla vydána v květnu roku 2010. Je ale vidět, že verze nižší než 2.2 zabírají pouze kolem 10 %.

Verze Androidu a její API je důležitá při vytváření aplikací. Vývojář by měl volit co nejnižší verzi API, aby byla aplikace dostupná pro co největší část užívaných zařízení. Protiklad k tomu je, že s každou novou verzí API se možnosti systému a programování pro něj rozšiřují. Takže je potřeba zvolit kompromis mezi co největším rozšířením zařízení a schopnostmi daného API.



Obr. 1.1: Zastoupení verzí Androidu

1.3 Vnitřní vrstvy systému

Informace v této části byly převzaty ze zdrojů [2], [3]. Android je softwarová platforma pro mobilní zařízení obsahující různé vrstvy, které jsou zobrazeny na obr. 1.2, převzatého ze zdroje [5]. Jednotlivé vrstvy se vzájemně prolínají, ale každá z nich má svůj vlastní účel a je přesně definována.

1.3.1 Linuxové jádro

Android běží na Linuxovém jádře verze 2.6. Linux se stará o systémové služby jako: bezpečnost, správa paměti, správa procesů, síť a ovladače. Dále se jádro chová jako abstraktní vrstva mezi samotným hardwarem a zbytkem systému. Linux jako základ je použit, protože mnoho jeho vlastností vyhovuje celkovému složení systému. Některé z nich jsou:

- **Přenositelnost** – Linux je možno snadno přeložit pro různé procesorové architektury, například x86, ARM. Zároveň je většina jeho částí napsána v jazyku C, což umožňuje výrobcům použití Androidu na mnoha různých zařízeních.
- **Bezpečnost** – Linux byl během svého používání testován na nepřeberné škále zařízení, sloužících mnoha různým účelům. Aplikace pro Android běží každá ve vlastním procesu a je jí dovoleno zasahovat pouze do částí, pro které má oprávnění.

1.3.2 Knihovny

V systému Android je zahrnuto množství nativních knihoven jazyků C a C++. Tyto knihovny jsou používány různými částmi systému a vývojáři k nim mají přístup přes aplikační framework. Mezi některé základní patří:

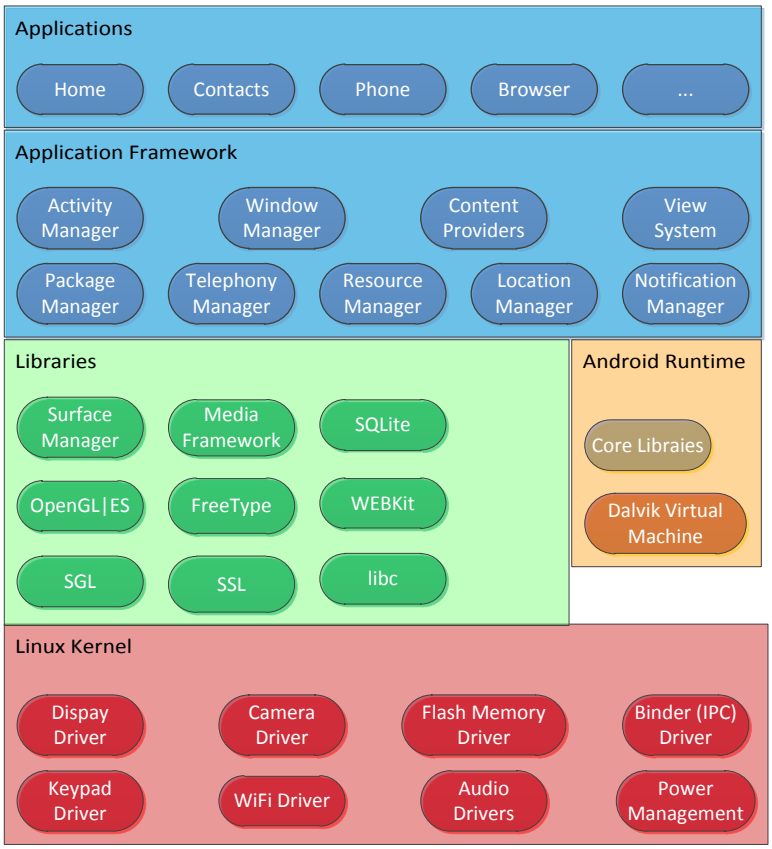
- **Knihovny systému C** – je to odvozená BSD implementace standardních systémových knihoven jazyka C (libc);
- **Knihovny multimédií** – jsou to knihovny pro podporu přehrávání a záznamu mnoha hudebních a video formátů, obsahující podporu MPEG4, H.264, MP3, AAC, AMR, JPG, a PNG;
- **Knihovny displeje** – řídí přístup k podsystému displeje a stará se o 2D a 3D vrstvy grafiky;
- **Knihovny jádra webového prohlížeče** – implementují webový prohlížeč, který se používá k samotnému prohlížení webu a pro aplikace využívající webové zobrazení;
- **SGL knihovny** – základní knihovny 2D grafiky;
- **3D knihovny** – implementace knihoven založená na OpenGL ES 1.0 API. Využívají buď 3D hardwarovou akceleraci, nebo optimalizované 3D softwarové rastrování;
- **knihovny SQLite** – knihovna platformy relační databáze, která je dostupná všem aplikacím.

1.3.3 Běhové prostředí

Každá aplikace pro Android běží ve své instanci Dalvik VM. Dalvik byl od začátku vyvíjen pro Android, aby splňoval specifické požadavky pro mobilní zařízení, jako jsou výdrž na baterii a omezený procesorový výkon. Nebylo zde použito Java VM, který nebyl vyvíjen pro specifické požadavky mobilních zařízení a v době kdy byl vyvíjen Dalvik nebyl Java VM volně dostupný.

Aplikace pro Android psané v jazyku Java jsou standardně zkompilovány do Java byte code pomocí Java překladače. Pří programování pro jazyk Java, by byl tento byte code spuštěn na Java VM. Pro Android je ale Java byte code přeložen překladačem Dalvik do Dalvik byte code, který je pak spuštěn na Dalvik VM. Důvod, proč se překládá zdrojový kód v jazyce Java nejdříve do Java byte code a pak do Dalvik byte code, je ten, že v době kdy byl Dalvik vyvíjen, se jazyk Java často měnil, zatímco Java byte code zůstával více méně stejný.

Z výše uvedeného vyplývá, že pro Android lze programovat v libovolném jazyce, který umožňuje překlad do Java byte code. Teoreticky zde není žádný problém, nicméně Java používaná v Androidu je specifická. Nejblíže má k Java Standard Edition, rozdíl je především u knihoven pro uživatelské prostředí, kde Android používá své vlastní. Takže ve výsledku je dostupná většina standardních knihoven Javy a navíc spousta dalších specifických pro Android. Tyto specifické knihovny jsou oficiálně dostupné pouze pro jazyk Java.



Obr. 1.2: Vrstvy systému Android

1.3.4 Aplikační framework

Vývojová platforma pro Android je prostředí, které poskytuje mnoho specifických knihoven pro Android, přístup k systémovým službám (manažerům), hardwarovým zdrojům (senzory, geo lokace, WiFi, Bluetooth, telefonování).

1.3.5 Aplikace

Nejvyšší vrstva systému je tvořena samotnými aplikacemi. Zařízení s Androidem bývají od začátku prodávána s balíkem licencovaných Google aplikací (e-mailový klient, Google Mapy...) a obvykle s balíkem aplikací přímo od výrobce.

Aplikace jsou distribuovány jako APK soubor, který obsahuje tři hlavní části. První z nich je samotný kód jazyka Java přeložený do Dalvik executable. Druhá je část zdrojů, což jsou všechny části aplikace, které nejsou zdrojový kód (obrázky, videa, XML soubory). Třetí část slouží pří použití nativního programování například v jazyce C nebo C++.

Aplikace pro Android se můžou šířit volně přes web jako APK soubor, nebo přes spoustu různých internetových obchodů. V současné době je z nich největší Android Market, který vlastní Google.

1.4 Hlavní části aplikace

Informace v této části byly převzaty ze zdrojů [1], [2] a [3]. Aplikace pro Android jsou složeny z mnoha různých částí, které komunikují mezi sebou a s ostatními aplikacemi a částmi systému. Tyto části dohromady určují jak se aplikace chová, jak reaguje na uživatelské vstupy a na vstupy ostatních komponent systému. Většinu hlavních částí aplikace si podrobněji probereme v následujících kapitolách.

1.4.1 Aktivity (Activities)

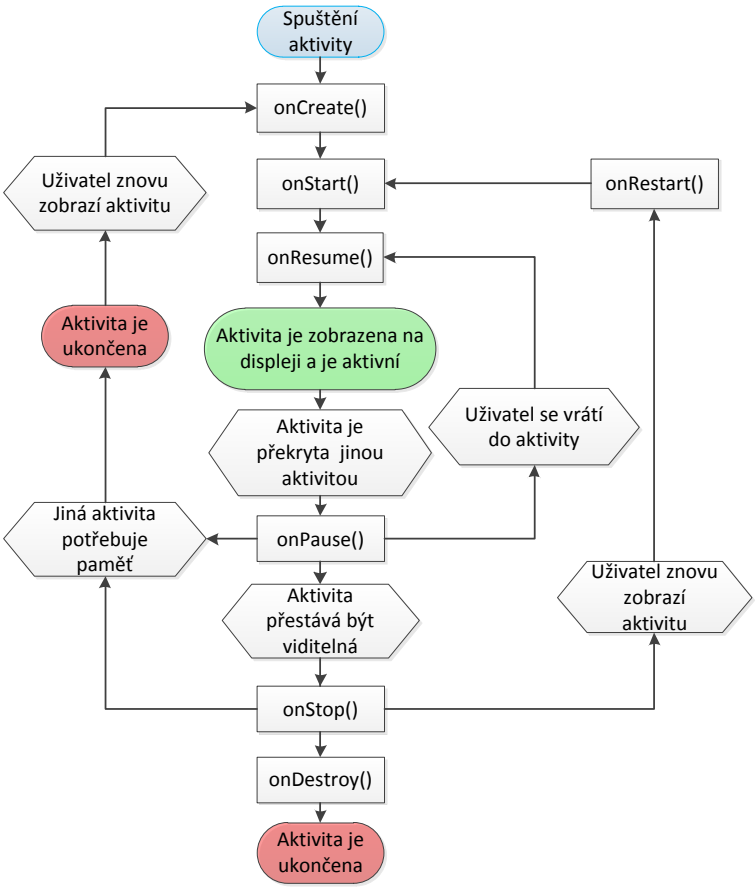
Aktivity jsou uživatelem nejvíce viditelné části programu. Každá aktivita většinou představuje jednu obrazovku aplikace. Slouží jako hlavní prostředek ke komunikaci s uživatelem. Při programování se UI definuje pomocí kódu nebo pomocí XML souboru. Aplikace většinou obsahuje více aktivit, kde jedna z nich je označena jako hlavní. Ta se zobrazí při spuštění aplikace. Uživatel mezi jednotlivými aktivitami přepíná nezávisle na aktuální aplikaci.

Životní cyklus aktivit

První spuštění Aktivity je náročné na systémové prostředky. Pokud aktuální aktivitu nahradí jiná, současná se kompletně neukončí, ale pouze se pozastaví. Je uložena, a pokud by ji uživatel chtěl znovu použít, je pouze vyvolána a přejde do aktivního stavu. Systém automaticky ukončuje dlouho nepoužívané Aktivity z důvodu uvolnění paměti. Změny mezi jednotlivými stavy řídí Activity Manager. Vývojář reaguje na jednotlivé změny mezi stavy implementováním příslušných metod. Jednotlivé stavy životního cyklu aktivity jsou uvedeny na obr. 1.3, který byl převzat ze zdroje [6] a budou dále podrobněji rozebrány.

Spuštění aktivity

Pokud aktivita neběží nebo není uložena v paměti, je aktivita vytvořena a zavolána metoda `onCreate()`, která je velice náročná na systémové prostředky. Poté, co proběhne start aktivity a jsou vykonány příkazy metody `onCreate()`, aktivita přejde ze spouštěcího stavu do aktivního stavu.



Obr. 1.3: Životní cyklus aktivity

Aktivní stav

Do aktivního stavu přejde aktivita buď automaticky přechodem ze spouštěcího stavu, nebo přechodem ze stavu pozastavení případně zastavení. V závislosti na předchozím stavu aktivity jsou postupně zavolány metody `onStart()` a `onResume()`. Názorně to lze vidět na výše zmíněném obr. 1.3. Aktivní stav je stav, při kterém je aktivita zobrazena na displeji a provádí interakci s uživatelem (zobrazování informací, reakce na stisk tlačítek...). V aktivním stavu má aktivita prioritní přístup k systémovým zdrojům za účelem co nejplynulejšího běhu. V aktivním stavu aktivita zůstává dokud není nahrazena jinou aktivitou.

Stav pozastavení

Pokud je aktivita stále viditelná, ale není aktivní, je zavolána metoda `onPause()`. Tento stav není příliš častý, jelikož většina aktivit je zobrazena přes celý displej.

Tato situace většinou nastane při zobrazení dialogového okna. Metoda `onPause()` je zavolána i v případě, že aktivita přechází do stavu zastavení. Aktivita má stále prioritní přistup k systémovým zdrojům, jelikož je zobrazena. Z tohoto stavu Aktivita přechází do stavu zastavení pokud není dále zobrazována, nebo zpět do aktivního stavu pokud se dostane znovu do popředí.

Stav zastavení

V případě, že aktivita přestane být zobrazována na displeji (většinou je nahrazena jinou aktivitou), přejde do stavu zastavení a je zavolána metoda `onStop()`. Aktivita je uložena v paměti pro případ, že by ji uživatel chtěl znovu zobrazit. Pokud se má znovu zobrazit je zavolána metoda `onRestart()` a aktivita přejde do aktivního stavu. Při ukončení aktivita pokračuje přechodem do ukončovacího stavu. Jestliže nastane požadavek na uvolnění paměti, je aktivita smazána z paměti a při jejím dalším spuštění běží od začátku.

Ukončovací stav

Pokud má být aktivita ukončena je zavolána metoda `onDestroy()`. V této metodě může aktivita provést ukončovací akce jako například uložení dat, nicméně není zaručeno že aktivita je řádně ukončena a metoda `onDestoy()` zavolána. Proto se doporučuje provádět tyto akce ve stavu pozastavení.

1.4.2 Intenty (Intents)

Intenty slouží k předávání zpráv a informací (dat) mezi hlavními částmi aplikace. Přímo spouští aktivity, služby a broadcast recivery. Intenty jsou asynchronní, část aplikace, která jej pošle, nemusí čekat zda došlo k jejich provedení. Pomocí intentů lze posílat libovolná data jiné části aplikace.

Intenty se dělí na implicitní a explicitní. Explicitní intent má přesně daného příjemce zprávy, například posílám intent jiné aktivitě vlastní aplikace spolu s daty které má zobrazit. Implicitní intent nemá jasně daného příjemce, například posílám intent s umístěním video souboru v paměti zařízení s požadavkem na přehrávání. Video tedy může přehrát libovolná aplikace k tomu určená. Pokud má uživatel na zařízení nainstalováno více aplikaci na přehrávání videa, uživatel bude vyzván k vybrání aplikace. Pokud vytvářím aplikaci například na funkci souborového manažeru a uživatel označí video soubor a bude použit implicitní intent, video se přehraje v jím preferovaném přehrávači. Ten ani nemusí být v době vývoje aplikace dostupný.

1.4.3 Služby (Services)

Služba je část aplikace, která běží na pozadí a nemá žádné uživatelem viditelné komponenty. O svém spuštění a ukončení by měla uživatele informovat, například notifikací v notifikační liště. Služby se používají, pokud chceme provádět operace na pozadí (přehrávání MP3 skladeb. . .).

Služeb jsou dva typy, spouštěné a vázané (bound). Spouštěné služby spustí jiná komponenta aplikace a pokud dokončí svůj úkol samy se ukončí nebo je ukončí jiná komponenta. Vázané služby jsou navázány na jinou komponentu aplikace. Funguje zde komunikace na bázi klient-server, kde komponenta získává informace od vázamé služby. Ve chvíli, kdy se komponenta odpojí, služba se ukončí. Nicméně lze v jedné službě používat oba způsoby. Například v aplikaci pro přehrávání hudby spustíme službu tím, že chceme přehrávat hudbu. Navážeme se na službu a předáme jí informaci o změně skladby a naopak získáme informace o přehrávané skladbě.

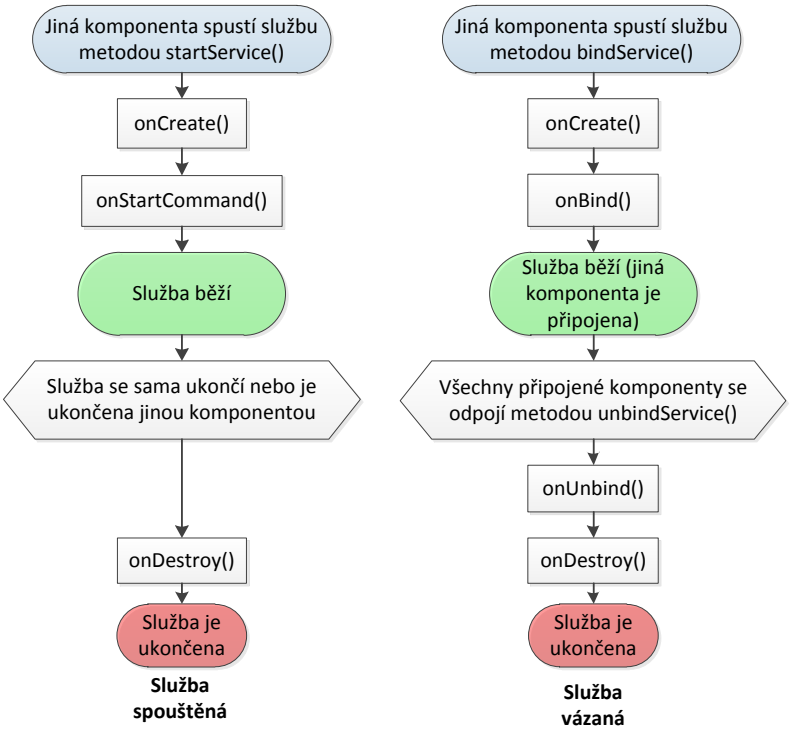
Služba si nevytváří vlastní vlákno, ale běží ve hlavním vlákně aplikace. Takže pro služby, které provádějí složitější operace je doporučeno spustit je ve vlastním vlákně. Jako aktivita má i služba vlastní životní cyklus, který je zobrazen na obr. 1.4,převzaného ze zdroje [7].

1.4.4 Poskytovatelé obsahu (Content Providers)

Každá aplikace v Androidu běží ve vlastní instanci virtuálního stroje Dalvik. K veškerým datům, se kterými pracuje, má přístup pouze tato aplikace. Malé množství dat lze posílat a přijímat skrze intenty, které se ale nehodí pro větší objemy dat. Proto jsou v Androidu poskytovatelé obsahu, je to rozhraní pro sdílení dat mezi aplikacemi. V Androidu se používají ve velké míře například aplikace pro posílaní SMS zpráv. Tato aplikace v sobě nemá žádná data o zprávách, ale spojí se s poskytovatelem obsahu pro SMS zprávy. Ten ale naopak nemá žádné uživatelské rozhraní. Systém výborně slouží v Androidu k uživatelskému přizpůsobení zařízení. Uživatel může mít libovolnou aplikaci pro posílání SMS zpráv. Ta se spojí s poskytovatelem obsahu pro zprávy, což je v Androidu obecný princip.

1.4.5 Stavová hlášení (Broadcast Receivers)

V systému Android je funkce, která hlásí skrz celý systém informace o tom, že nastala určitá událost. Například příchod SMS zprávy, nízký stav baterie, start systému a mnoho dalších. Aplikace se může přihlásit k odběru těchto zpráv. Kromě standardních zpráv lze nadefinovat i vlastní.



Obr. 1.4: Životní cyklus služby: vlevo služby spouštěné, vpravo služby vázané

1.5 Uživatelské rozhraní

Informace v této části byly převzaty ze zdrojů [2] a [3]. Uživatelské rozhraní v systému Android lze tvořit dvěma způsoby, pomocí deklarativního přístupu a programového přístupu. Každý z nich používá vlastní metodu pro tvoření uživatelského rozhraní. V praxi se využívá kombinace obou přístupů. Deklarativním se určuje jak daný objekt vypadá a programovým se určují jeho reakce na uživatelské vstupy.

Deklarativní přístup při tvorbě uživatelského rozhraní

Používá se zde jazyk XML, kterým se pomocí značek určuje jak bude daná obrazovka vypadat. Výhodou tohoto přístupu je, že Andoid Development Tools obsahuje WYSIWYG zobrazovač těchto značek. Nevýhodou je omezení jazyka XML. Slouží skvěle k určení, jak má uživatelské rozhraní vypadat, ale už se s ním hůře řeší reakce na uživatelské vstupy.

Programovací (imperativní) přístup při tvorbě uživatelského rozhraní

Uživatelské rozhraní se programuje v jazyce Java, grafická knihovna je podobná jako Swing. Slouží dobře pro tvorbu reakcí na uživatelské vstupy (kliknutí na tlačítko, označení položky v seznamu). Hůře je použitelný pro samotné programování vzhledu objektů.

1.5.1 Elementy uživatelského rozhraní

Každý prvek zobrazený na obrazovce se nazývá `view`, patří sem například tlačítka, textové popisky, pole pro zaznamenávání textu. Jednotlivé `view` jsou na obrazovce organizovány pomocí prvku `layout`. V systému Android je několik hlavních prvků `layout` mezi ně patří například: `LinearLayout`, `TableLayout`, `FrameLayout`, `RelativeLayout` a `AbsoluteLayout`.

1.6 Geolokace v systému Android

Informace v této části byly převzaty ze zdroje [3]. Android poskytuje možnost získat aktuální polohu. Polohu získává na základě GPS technologie nebo na základě lokalizace pomocí mobilní sítě a Wifi sítí.

1.6.1 Získání aktuální pozice

Zde bude popsán systém, podle kterého aplikace získává aktuální pozici a budou uvedeny prvky, ze kterých se systém skládá. Tyto prvky budou v následující části podrobněji popsány a budou doplněny o prvky, které zde nebyly uvedeny.

Aktuální pozici získáme pomocí systémové služby. Základním prvkem je třída `LocationManager`, jejíž instanci lze získat přímo ze systému. Informaci o aktuální pozici lze získat po zaregistrování systémové služby přes instanci `LocationManager`. Tím se přihlásíme k odběrům informací o poloze. Samotnou informaci zpracovává instance rozhraní `LocationListener`, které jsme předali při registraci služby. Při změně polohy je zavolána příslušná metoda dané instance rozhraní, kde se provede zpracování aktuální polohy. Ve chvíli, kdy přestaneme požadovat informace o poloze, odregistrujeme systémovou službu.

1.6.2 Prvky systému pro získávání polohy

Výše zmíněný systém pro získávání polohy v Androidu obsahuje několik specifických prvků, které budou dále uvedeny.

`LocationManager`

Tato třída poskytuje přístup k systémově službě pro lokaci. Umožňuje registrování informací o změně polohy pomocí metody `requestLocationUpdates()`. Při registraci zvolíme buď poskytovatele polohy nebo instanci třídy `Criteria`, která slouží k rozhodnutí jakého poskytovatele použít. Dále je možnost určit minimální čas nebo vzdálenost, po jejichž uplynutí jsou poskytovány další informace o poloze. Vhodným zvolením těchto dvou parametrů lze nastavit získávání nových informací v optimální četnosti vzhledem k danému použití. Další parametr je cíl poskytovaných informací, nejčastěji instance rozhraní `LocationListener`. Je zde také metoda pro odregistrování získávání informací o poloze a metody pro spravovu, získávání informací o poskytovatelích polohy a metoda pro zjištění poslední polohy.

`LocationListener`

Toto rozhraní slouží k získávání informací o poloze. Instance `LocationManager` přes instanci `LocationListener`, která byla zadána při registraci odběru informací o poloze, zavolá při každém získání nových informací příslušnou metodu:

- `onStatusChanged()` – metoda je zavolána pokud dojde ke změně dostupnosti polohy;
- `onProviderEnabled()` – metoda je zavolána pokud je poskytovatel polohy povolen uživatelem zařízení;
- `onProviderDisabled()` – metoda je zavolána pokud je poskytovatel polohy zakázán uživatelem zařízení;
- `onLocationChanged()` – metoda je zavolána pokud dojde ke změně polohy, metoda získá aktuální objekt `Location`, ve kterém jsou uloženy všechny informace o poloze.

Při použití rozhraní `LocationListener` musí být implementovány všechny výše zmíněné metody.

`Geocoder`

Třída, která provádí geokódování a reverzní geokódování. Geokódování je proces, při kterém je adresa převedena na zeměpisnou šířku a délku. Reverzní geokódování je proces obrácený, při kterém je převedena zeměpisná délka a šířka na adresu.

`LocationProvider`

Abstraktní třída, která slouží k získávání informací o poskytovatelích polohy.
V Androidu jsou dva poskytovatelé polohy. GPS určuje polohu pomocí satelitů. V otevřeném prostoru, je schopen poskytovat informace s přesností na několik metrů.

V uzavřeném prostoru (město) se přesnost liší podle výšky okolních budov a obvykle dosahuje horší přesnosti. Přesnost také záleží na velkosti antény daného zařízení. Při použití tohoto poskytovatele trvá delší dobu než je schopen dodat první informaci o poloze (je to způsobeno tím, že musí najít příslušné satelity). Používání GPS je náročné na baterii.

Druhým poskytovatelem je síťový poskytovatel, ten určuje polohu pomocí GSM a Wifi sítí. Přesnost toho poskytovatele je nižší než u GPS, je ale schopen poskytnout informace o poloze okamžitě po spuštění. Přesnost závisí na kvalitě GSM signálu a množství okolních GSM vysílačů. Je nenáročná na spotřebu baterie, ale vyžaduje připojení k internetu.

`Location`

Třída obsahuje informace o geografické pozici bodu určeného v konkrétním čase. Každá instance obsahuje zeměpisnou šířku a délku a čas kdy byla pořízena. Může obsahovat i další doplňující informace, pokud je poskytovatel polohy poskytuje nebo je schopen je pro dané souřadnice poskytnout.
Metody pro získání informací obsažených ve třídě `Location`:

- `getLatitude()` – vrátí zeměpisnou šířku bodu;
- `getLongitude()` – vrátí zeměpisnou délku bodu;
- `getAccuracy()` – vrátí přesnost bodu v metrech;
- `hasAccuracy()` – vrátí informaci o tom, zda třída obsahuje platnou přesnost;
- `getAltitude()` – vrátí nadmořskou výšku bodu;
- `hasAltitude()` – vrátí informaci, zda třída obsahuje platnou nadm. výšku;
- `getSpeed()` – vrátí rychlost bodu v metrech za sekundu;
- `hasSpeed()` – vrátí informaci o tom, zda třída obsahuje platnou rychlosti;
- `getBearing()` – vrátí azimut směru trasy ve stupních;
- `hasBearing()` – vrátí informaci o tom, zda třída obsahuje platný azimut;
- `getTime()` – vrátí počet milisekund, které uplynuly od 1. ledna 1970 0:0 UTC;
- `getProvider()` – vrátí jméno poskytovatele polohy, který vytvořil tento bod;
- `bearingTo()` – vrátí azimut mezi tímto a zadaným bodem;
- `distanceTo()` – vypočítá vzdálenost mezi tímto a zadaným bodem v metrech.

1.7 Zobrazování Google Map v systému Android

Informace v této části byly převzaty ze zdrojů [3] a [8]. V aplikaci pro Android lze zobrazit mapy z aplikace mapy Google. Google poskytuje pro Android externí knihovnu. Tato knihovna obsahuje balíček tříd, které umožňují stahování, zobrazení, ovládání a práci s Google Map.

Třída, která zobrazuje mapu je `MapView`. Tato třída obaluje Google Maps API a umožňuje pracovat s Google mapami stejným způsobem, jakým pracujeme s ostatními prvky `view`. Mapa, která je vykreslena pomocí `MapView`, umožňuje uživateli, aby ji ovládal stejným způsobem jakým ovládá například standardní aplikaci mapy (posunování mapy, přibližování oddalování mapy). Tento balíček umožňuje vykreslování vrstev na mapu, zpracovávání interakce s uživatelem...

Knihovna není součástí standardní knihovny Android, nicméně většina zařízení obsahuje balíček aplikací od Googlu a také tuto externí knihovnu. Knihovna také není standardně obsažena v Android SDK, a tak se musí do SDK přidat. Pro zobrazení `MapView` je potřeba tomuto prvku zadat Maps Api Key, což je klíč, který se registruje na konkrétního vývojáře prostřednictvím jeho Google účtu.
Další hlavní třídy balíčku, kromě již výše zmíněného `MapView`:

- `MapActivity` – pokud má aktivita zobrazovat `MapView`, nedědíme klasickou třídu `Activity` ale třídu `MapActivity`;
- `MapController` – třída pro nastavování mapy, umožňuje řídit posouvání, přibližování, centrování, animaci na určitý bod mapy;
- `GeoPoint` – třída vytváří bod daný zeměpisnou šířkou a délkou měřenou v mikrostupních;
- `Projection` – rozhraní sloužící k přepočítávání bodu daného zeměpisnou šířkou a délkou na souřadnice x a y v pixelech od horního levého rohu daného `MapView`;
- `Overlay` – základní třída sloužící k přidávání vrstev nad zobrazenou mapu, pro přidání vlastní vrstvy je potřeba přidat do aktuálních vrstev daného `MapView` instanci třídy získané děděním této třídy;
- `ItemizedOverlay` – získán děděním `Overlay` obsahuje položky, které chceme vykreslit na mapu;
- `OverlayItem` – položka `ItemizedOverlay` obsahuje samotný objekt, který chceme vykreslit na mapě;
- `MyLocationOverlay` – vrstva obsahuje aktuální pozici uživatele na mapě včetně zobrazení přesnosti této polohy, může zobrazovat také kompas ukazující směr, kterým je na mapě zařízení natočeno.

1.8 Google Static Maps API

Informace v této části byly převzaty ze zdroje [9]. Google poskytuje API, přes které lze získat obrázek s vybraným mapovým podkladem na kterém jsou zobrazeny předem zadané objekty. Toto API není určeno přímo pro Android, ale hlavně pro webové stránky a aplikace.

Static Maps API vytvoří obrázek ve formátu GIF, PNG nebo JPEG. Tento obrázek je získám pomocí REST API. Pro jeho získání tedy není nutno používat JavaScript. Díky tomuto způsobu získávání dat je možno použít toto API v jakémkoliv systému nebo programovacím jazyce. Stačí když daný jazyk dokáže zaslat HTTP požadavek a zobrazit obrázek. Pro každý obrázek je možno specifikovat mnoho parametrů například pozici na mapě, velikost obrázku, úroveň přiblížení, typ mapy, kvalitu, dále je možno na mapu přidat značky, popisky, zobrazit trasu a mnoho dalšího.

Omezení použití toho API je ze strany Googlu v počtu maximálně 1000 unikátních žádostí na jednoho uživatele za den. Uživatel je zde brán jako koncový uživatel aplikace, která Static Masps API používá. Pokud API použiji v aplikaci, kterou bude používat 1000 lidí, každý z těchto uživatelů může získat 1000 obrázků denně. Po překročení tohoto limitu je vrácen HTTP 403 status, pozastavení používání API je pouze dočasné. Další omezení je technologického rázu a to omezení délky URL adresy na maximálně 2048 znaků. Nicméně vytvořená adresa může být následně prohlížečem nebo jinou službou, která se stará o odeslání HTTP požadavku, kódována a proto může dojít k nárůstu znaků.

Každá z URL použitých v tomto API musí začínat tímto: `http://maps.googleapis.com/maps/api/staticmap?parameters` případně je možnost použití HTTPS. Následují další URL parametry, kde některé jsou vyžadovány a jiné jsou volitelné. Jednotlivé parametry se dle standardu URL oddělují znakem `&`.
Některé z URL parametrů:

- `center` – určuje střed mapy na obrázku, lze jej definovat pomoci zeměpisné šířky a délky nebo adresy (povinný pokud není zobrazena značka);
- `zoom` – určuje úroveň zoomu mapy nastavuje se v rozmezí hodnot od 0 do 21 nejvzdálenější až nejbližší zobrazení; ne pro každou polohu na světě jsou dostupny všechny úrovně zoomu a v případě nedostupnosti mapy je vrácen prázdný obrázek (povinný pokud není zobrazena značka);
- `size` – udává velikost mapy v pixelech (povinný parametr);
- `scale` – udává kvalitu obrázku (volitelný parametr);
- `format` – udává formát obrázku, standardně je nastaven na PNG, jsou ale možné jiné formáty (volitelný parametr);
- `maptype` – udává jaký mapový podklad je použit, k dispozici jsou: obecná, satelitní, hybridní (volitelný parametr);
- `language` – udává jazyk, ve kterém budou zobrazeny popisky na mapě, pokud daný jazyk není podporován je použít výchozí jazyk (volitelný parametr);
- `markers` – umožňuje přidat na mapu jednu nebo více značek, které se mají zobrazit, jednotlivé parametry značky se rozdělují znakem `|` (volitelný parametr);

- `path` – umožňuje na mapu přidat trasu složenou ze dvou nebo více bodů propojených čarou, je možno specifikovat barvu a tloušťku čáry, jednotlivé body se rozdělují znakem `|` (volitelný parametr);
- `sensor` – určuje, zda zařízení používá senzor k určení polohy (povinný parametr).

2 NÁVRH A IMPLEMENTACE APLIKACE

Tato část pojednává o návrhu, tvorbě a optimalizaci aplikace. Informace o programování pro android jsou převzaty ze zdrojů [1], [2] a [3]. Informace o optimalizaci aplikace jsem čerpal ze zdroje [11]. Informace o jazyku Java a jeho funkcích jsem čerpal ze zdroje [10].

2.1 Vývojové prostředí, verze API, testovaní

Aplikaci jsem vyvíjel ve vývojovém prostředí Eclipse, do kterého byl doinstalován a nakonfigurován ADT plugin (plugin pro vyvíjení a testování aplikací pro Android). Následně jsem nainstaloval Android SDK, do kterého byly nahrány příslušné komponenty pro dané API. Podrobný návod, jak nainstalovat a nastavit tyto programy, lze získat ze zdroje [12].

Na základě zastoupení jednotlivých verzí Androidu, jak je uvedeno na obr. 1.1, jsem se rozhodl, že budu aplikaci vyvíjet pro verzi 2.2. Díky tomu bude aplikace spustitelná na přibližně 90 % používaných zařízení. Toto číslo od začátku vývoje vzrostlo přibližně na 93 % a Android verze 2.2 běží zhruba na 23 % zařízení. Během vývoje jsem se nesetkal s problémem, že by pro tuto aplikaci potřebné komponenty nebyly dostupné v této verzi API. Nicméně jsem použil několik metod v současné době označené jako "deprecated", což znamená, že jsou zastaralé a jsou nahrazeny novými metodami. Tyto metody byly ale dostupné pouze ve vyšší (novější) verzi API. Jednalo se většinou o verzi systému 3.x s názvem Honeycomb, což je systém vyvíjený pro tablety. Tyto metody jsou také ve verzi systému 4.0 Ice Cream Sandwich, který je i pro mobilní telefony. Tento systém v současné době používá kolem 3 % zařízení z nichž část jsou tablety. Pro mnou použitou verzi systému 2.2 je příslušná verze API 8. Jelikož aplikace bude obsahovat Google mapy, není použito klasické API verze 8, ale Google APIs verze 8.

Testování aplikace provádím na zařízení ZTE Blade, které má 600 MHz ARMv6 procesor, grafiku Andeno 200, 512 MB paměti RAM a displej s úhlopříčkou 3.5 palce a rozlišením 800x480 bodů. Na zařízení byl nainstalován Android verze 2.3.7. Testovací zařízení patří k levnějším a méně výkoným zařízením. Veškeré poznatky ohledně plynulosti a výkonu aplikace byly učiněny na základě běhu aplikace na tomto zařízení. Používám připojení přes ADB (Android Debug Bridge), což je nástroj, pomocí kterého komunikuji se zařízením přímo z vývojového prostředí a nástroj DDMS (Dalvik Debug Monitor Service), který umožňuje získávat informace zpět do vývojového prostředí pro ladění aplikace.

Pro testování vzhledu uživatelského rozhraní na zařízeních s různou velikostí displeje a rozlišením jsem používal několik AVD (Android Virtual Device), kterým jsem tyto parametry nastavil podle typů zařízení na trhu. Konkrétně jsem používal AVD reprezentující ty nejlevnější zařízení s velikostí displeje kolem 3 palců a rozlišením 240x320 bodů. Dále AVD reprezentující zařízení kolem 4 tisíc korun s úhlopříčkou 3,2 až 3,5 palce a rozlišením 320x480 bodů. A AVD reprezentující high end a to úhlopříčka 4.3 palce s 480x800 bodů a 4.65 palce s 720x1280 bodů.

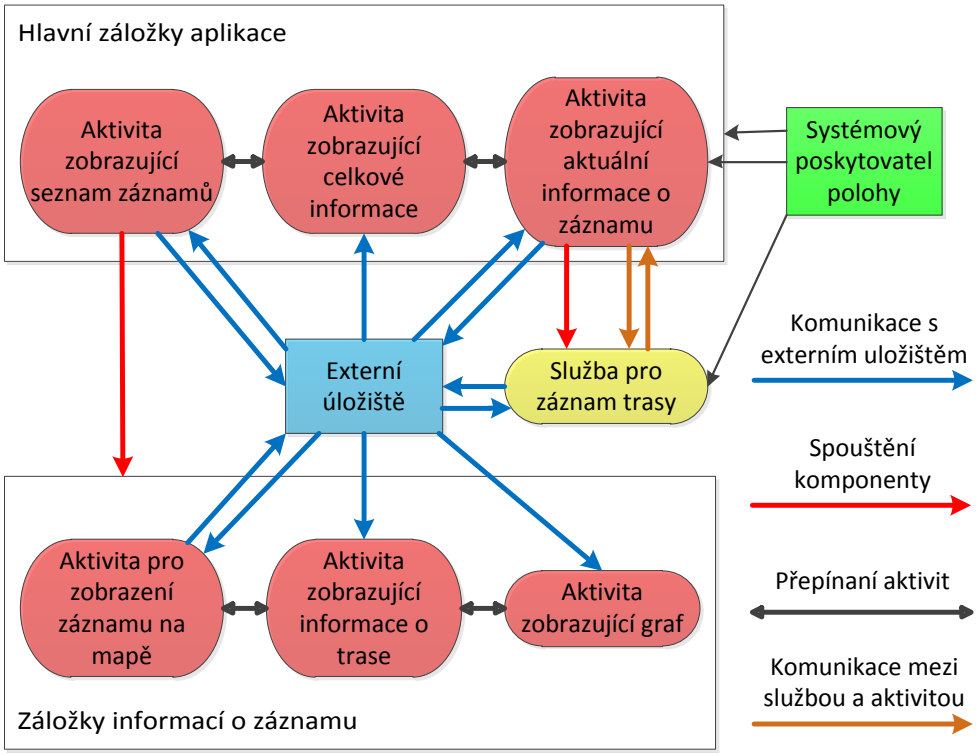
2.2 Návrh aplikace

Jak již bylo výše popsáno, aplikace pro Android se skládá z mnoha různých komponent, které na sobě nemusí být přímo závislé, takže aplikaci lze snadno rozšiřovat a upravovat. Toho využívám při tvorbě, kdy vytvářím základní kostru aplikace, ke které přidávám další rozšíření. Aplikace by měla být po přidání každé nové části funkční.

Tato aplikace byla vytvářena speciálně pro rekreační běh. Aby umožňovala zaznamenávat trasu běhu a následně spočítat informace o daném běhu. Při samotném běhu by měla zobrazovat uživateli důležité informace o aktuálním běhu stejně tak jako aktuální polohu. Následně by měla být schopna tyto data zobrazit jak na mapě, tak v podobě výpisu určitých informací o daném běhu. Pro možnost sledování případného růstu nebo poklesu výkonnosti by měla umožnit uživateli porovnat záznamy běhů. Většinou totiž člověk, který se věnuje tomuto sportu, má několik tras na kterých běhá. Proto by měl mít možnost vytvořit trasu a na ní následně trénovat. Zobrazit si, zdali byl lepší nebo horší než minule případně než v jiný den a to jednak globálně na celé trati tak na dílčích úsecích.

Současná verze aplikace, jejíž blokové schéma je na obr. 2.1, při spuštění zobrazí `TabActivity`. Tato komponenta systému, vytvoří v horní části obrazovky záložky, umožňující uživateli jednoduše se přepínat mezi aktivitami. A to aktivitou zobrazující souhrnné informace o záznamech (celková doba, vzdálenost...), aktivitou, která umožňuje spustit službu pro záznam dat a během záznamu zobrazuje aktuální informace o záznamu a aktuální polohu na mapě, a aktivitou zobrazující seznam záznamů a umožňuje jejich spuštění, smazání, exportování a trénování na nich.

Při vybrání určitého záznamu na mapě se spustí další `TabActivity`, která obsahuje aktivity pro zobrazení trasy záznamu na mapě, zobrazení kompletních informací o záznamu a zobrazení profilu nadmořské výšky v závislosti na čase. V celé aplikaci je dostupné menu, které umožní vyvolat uživatelské nastavení daných hodnot a chování aplikace v určitých situacích. Jednotlivé aktivity a službu proberu v následujících kapitolách podrobněji.



Obr. 2.1: Blokové schéma aplikace

Systém android je provozován na mnoha zařízeních, s různou úhlopříčkou displeje (od 2.8 palce do 4.65 palce) a rozlišením (od 240x320 bodů po 720x1280 bodů) a také různých cenových kategoriích (od 3 tisíc do 15 tisíc). A to se zde pouze mobilní telefony a ne ostatní zařízení s androidem (převážně tablety a stále častěji chytré televize . . .). Při návrhu a vývoji aplikace je vhodné omezit množinu možných zařízení na ty, u kterých má použití aplikace smysl, a tomu přizpůsobit jak optimalizaci vzhledu a uživatelského rozhraní aplikace, tak i optimalizaci samotné aplikace.

2.3 Implementace aplikace

V této části podrobněji rozeberu jak jsem postupoval při vývoji, testování a optimalizaci jednotlivých částí aplikace, jak jsou na sebe tyto části vzájemně napojeny a jak spolu komunikují.

2.3.1 Uživatelské rozhraní na mnoha různých displejích

Jak jsem již zmínil výše, pro testování vzhledu jsem používal několik AVD, které reprezentovaly jednotlivé kategorie zařízení na trhu. Android poměrně jednoduše umožňuje pokrýt toto široké spektrum rozlišení a úhlopříček zařízení. Aplikace a systém neřeší konkrétní rozlišení a úhlopříčku, ale řeší density displeje což je počet bodů na palec. Vývojář tak může vydat různé XML soubory pro různé density, nebo může při definování vzhledu používat jako jednotky tzv. `dp`, což je jednotka velikosti závislá na density daného přístroje. Stejně tak je pro velikost písma jednotka `sp`, což je jednotka podobná `dp` tedy závislá na density, ale taky zohledňuje velikost nastaveného písma na telefonu. Já jsem při vývoji aplikace zvolil druhou cestu, používal jsem jednotky závislé na density. Při vytváření vzhledu se také doporučuje používat relativní rozmístění prvků na obrazovce (v levém horním rohu, vpravo od jiného prvku ...).

2.3.2 Optimalizace

Aplikace se musí být vyvíjena s ohledem na omezené výpočetní prostředky a omezenou kapacitu baterie. Informace ohledně způsobů optimalizace aplikace a jejich implementaci jsem čerpal ze zdroje [11]. Jedním z uvedených poznatků je, že každá uživatelská akce by měla mít odezvu nejpozději do 100 ms, což je mnohdy zdánlivě nemožný cíl. Pokud se mi nepodařilo tento čas splnit, snažil jsem se k němu alespoň co nejvíce přiblížit. Stejně tak jsem se snažil minimalizovat množství a optimalizovat prováděné výpočty.

2.3.3 Aktivita zobrazující celkové informace

Při spuštění aplikace se při výchozím nastavení zobrazí tato aktivita, viz obr. 2.2. Jak již bylo řečeno výše celá aplikace využívá záložkové schéma. Pomocí záložek v horní části aplikace se lze přepínat mezi jednotlivými aktivitami. Při výchozím nastavení se po spuštění zobrazí tato aktivita. Pomocí příslušné volby v nastavení lze jako aktivitu po spuštění zvolit libovolnou aktivitu z tohoto záložkového schématu.

Aktivita zobrazuje celkové informace o záznamech, uběhnutou vzdálenost, čas strávený při běhu, spálené kalorie a maximální rychlost. Dále zobrazuje výpis informací o posledním záznamu a to ve formátu položky seznamu. Při kliknutí na tuto položku je spuštěna aktivita pro zobrazování informací o záznamu. Tyto informace načte z příslušných souborů, které podrobněji rozeberu níže při popisování souborové struktury.

V jakékoliv aktivitě tohoto záložkového schématu je spuštěna systémová služba pro získávání polohy pomocí GPS. I když se případné informace o poloze dále nezpracovávají slouží k tomu aby, v okamžiku, kdy uživatel bude chtít spustit službu pro záznam dat, byla informace o poloze co nejdříve dostupná. Dá se totiž předpokládat, že v tomto záložkovém schématu bude uživatel chtít službu použít.

Obr. 2.2: Aktivita zobrazující celkové informace

2.3.4 Aktivita pro zobrazování aktuálních informací a Služba pro záznam trasy

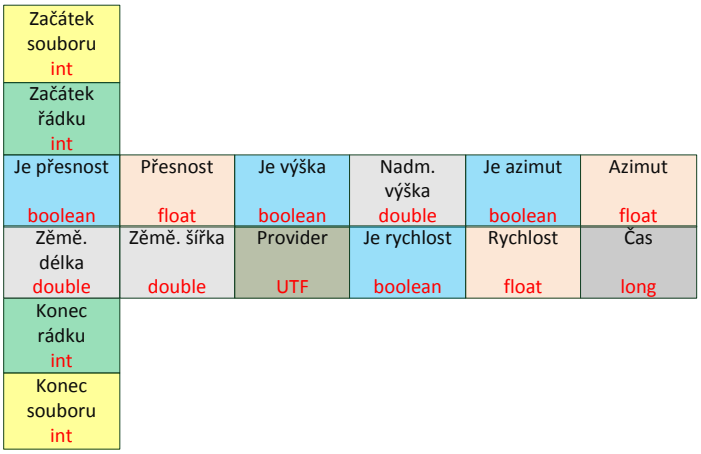
Tato část bude řešit aktivitu a službu a jejich vzájemnou komunikaci. Aktivita je zobrazena na obr. 2.3, jejím účelem je spouštět, pozastavovat a zastavovat službu pro záznam trasy. Dále při probíhajícím záznamu zobrazuje dobu trvání záznamu, aktuální rychlost a uběhnutou vzdálenost. Na `MapView` v dolní části obrazovky zobrazuje aktuální polohu, kompas a již uběhnutou trasu. Služba pro záznam trasy slouží k periodickému zaznamenávání pozice uživatele a jejímu zpracování a uložení.

Obr. 2.3: Aktivita zobrazující aktuální informace o záznamu

Uložení zaznamenaných dat

Každý záznam trasy vygeneruje na základě jeho délky desítky až tisíce objektů typu `Location`. V původním návrhu aplikace jsem tyto položky ukládal do souboru, pojmenovaného podle času a data při začátku zaznamenávání trasy, tak aby je jiná část aplikace mohla plně načíst. Na obr. 2.4 lze vidět datovou strukturu tohoto souboru. Na začátku a konci souboru je vložena unikátní značka. Mezi těmito značkami jsou vloženy imaginární řádky, kde každý řádek obsahuje informace z jednoho objektu typu `Location`. Na začátku a konci řádku jsou značky, které ho ohraničují. Tímto způsobem označování dat v souboru jsem schopen při čtení ověřit zda daný soubor byl v pořádku zapsán.

Všechna zaznamenaná se ukládala protože zpočátku nebylo jasné jak budou dále využita. Největší problém tohoto způsobu bylo načítání dat, kdy půlhodinový záznam obsahoval 1600 položek a doba samotného načtení souboru z paměti byla okolo 600 ms. Pro načtení položek pro zobrazení trasy záznamu na mapě bylo potřeba na každém imaginárním řádku načíst pouze informaci o zeměpisné šířce a výšce. Problémem bylo načítání mnoha dalších, pro toto použití zbytečných položek. Proces



Obr. 2.4: Struktura souboru

načítání znatelně nezrychlilo ani načtení potřebných položek a skok v souboru na další řádek. Dále se nad načtenými daty prováděly při každém použití stejné výpočty (třídění dat, vzdálenost, čas...), i když je stačilo provést pouze jednou a pak je načíst. Proto bylo nutné vytvořit novou strukturu ukládaných dat.

V současné verzi aplikace je již stanoveno, jak se budou zaznamenaná data třídit a co všechno se z nich bude počítat. Proto se ukládají jenom potřebná data. Všechny soubory používají výše zmíněné schéma, na začátku a konci souboru je unikátní značka. Pokud se v souboru opakuje zaznamenávání stejných dat je použito stejného principu imaginárních řádků s unikátní značkou na začátku a konci.

Jak již bylo řečeno, je zde soubor, který obsahuje informace o celkově získaných datech (vzdálenost, kalorie, čas a maximální rychlost). Tento soubor je pojmenován názvem `main`.

Dále je zde soubor, obsahující informace o každém záznamu. Tyto informace slouží k vytvoření seznamu záznamů a to konkrétně: název záznamu, čas a datum pořízení záznamu, místo pořízení, uběhnutou vzdálenost, strávený čas a informace o tréninku. Tento soubor se nazývá `logslist`.

Pro každý ze záznamů existuje několik souborů. Soubor se záznamem trasy, který obsahuje pro každý ze zaznamenaných bodů informace o čase bodu a zeměpisné šířce a délce. Díky tomu byl u stejného záznamu snížen čas načítáni z 600 ms na přibližně 70 ms. Tento soubor je pojmenován podle vytvořeného názvu (uloženého v souboru `logslist`) s přidaným písmenem L.

Další soubor obsahuje informace spočítané ze zaznamenaných dat: čas a datum pořízení záznamu, místo pořízení, uběhnutou vzdálenost, strávený čas, průměrnou a maximální rychlost, spálené kalorie, nastoupanou a sestoupanou nadmořskou výšku, popisky os pro graf a pro každý bod nadmořskou výšku. Toto schéma souboru

je výhodné pro různé použití, např. potřebujeme načíst pouze souhrnné informace o záznamu (nenačítáme dál informace o nadmořské výšce), potřebujeme načíst data pro graf nadmořské výšky (vzhledem k načítání řádu stovek nadmořských výšek je zbytečné čtení souhrnných informací zanedbatelné). Soubor je pojmenován podle vytvořeného názvu s přidaným písmenem I.

Dalším ze souborů je soubor pojmenován podle vytvořeného názvu s přidaným písmenem c. Tento soubor obsahuje informaci o tom, zda je záznam tréninkem, nebo originálním během a případně název originálního záznamu. Dále obsahuje pro každý z kontrolních bodů informaci o jeho zeměpisné šířce a délce, času a informaci zda byl daný úsek lepší než na tréninkovém záznamu. V případě tréninku je ještě vytvořen soubor se seznamem záznamů tohoto tréninku a základních informací o nich (podobně jako v `logslist`).

I když jednotlivé záznamy trasy nejsou datově velké, může jich teoreticky být neomezeně. V systému Android se pro záznam dat aplikací tohoto charakteru používá externí úložiště dat. V zařízeních obsahujících několik GB paměti je tato paměť rozdělena na vnitřní a vnější (externí) paměť, v zařízeních s malou pamětí je všechna paměť použita jako vnitřní, pro vnější paměť slouží SD karta, která je součástí zařízení již při jeho nákupu.

Při ukládání dat na externí úložiště může nastat problém v případě, že v době, kdy je soubor otevřen, dojde k požadavku na odpojení externího úložiště. Android v tomto případě službu, která má soubor otevřený, restartuje. Tato situace může způsobit problémy, proto je soubor otevřen pouze na dobu zápisu.

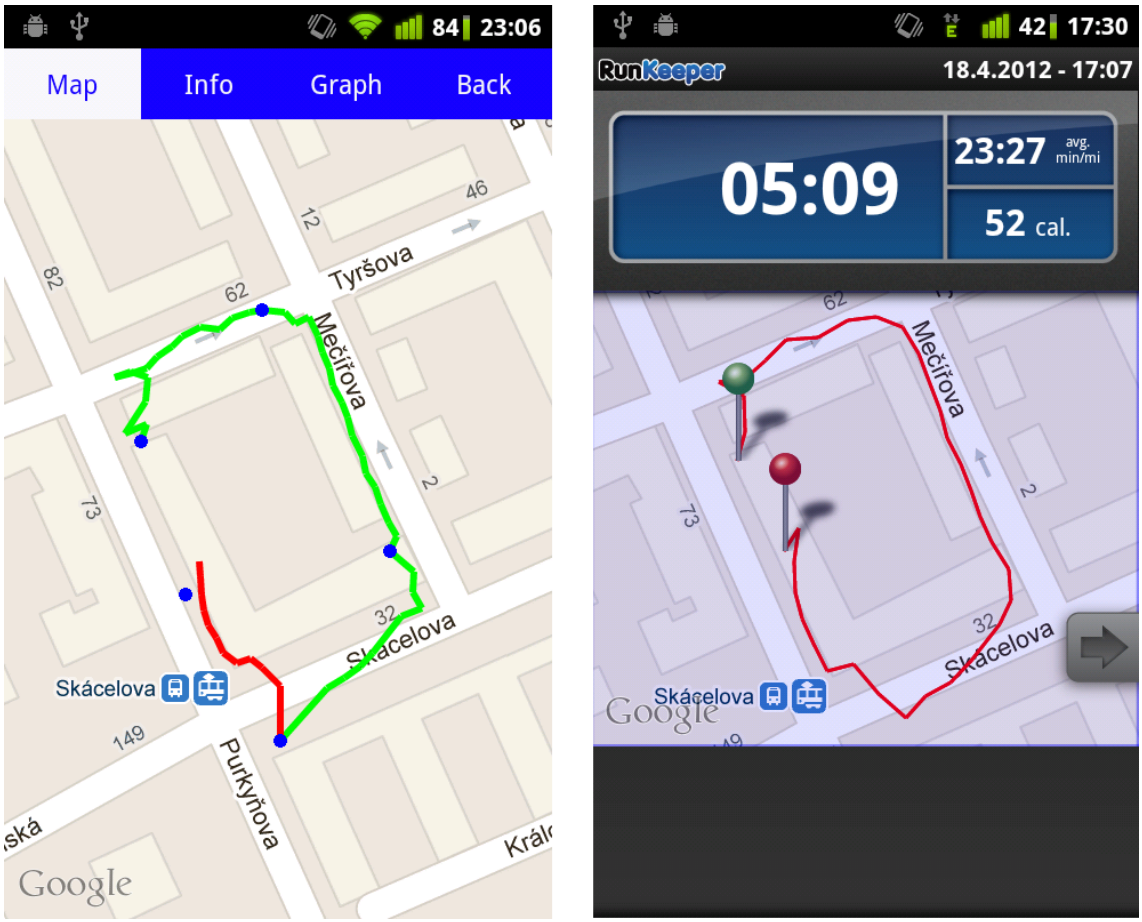
Zjišťování polohy a její třídění

Při startu služby je již informace o poloze s požadovanou přesností dostupná. Jako poskytovatel služby je používána GPS, jelikož poloha podle GSM sítě byla pro záznam cesty příliš nepřesná. Pokud se při běhu služby stane poskytovatel polohy nedostupný, je služba ukončena.

Při zaznamenávání dat nejsou zaznamenána všechna, ale pouze data odpovídající daným podmínkám. V prvním kroku jsou pro záznam vhodná data, jejichž přesnost je vyšší než zadaná. Odfiltrování bodů s nízkou přesností je důležité pro zobrazování trasy. Používám přesnost 30 m a zjistil jsem, že v otevřeném prostranství je nejvyšší přesnost bodů kolem 6 m. Přesnost dat je u této aplikace největší problém, při použití polohy s přesností 30 m jsou poměrně nepřesná a při zobrazení záznamu umisťují uživatele na mapě například do budov. Nelze určit jestli v danou chvíli uživatel skutečně vstoupil do dané budovy, nebo zda jsou špatná data o poloze, přijímaná ze systémového poskytovatele GPS.

Ve druhém kroku jsou zaznamenány jenom ty záznamy, u kterých platí, že vzdálenost k předchozímu záznamu je větší než zadaná vzdálenost. Hodnotu tohoto parametru jsem zvolil 5 m. Tento krok je důležitý pro zredukování počtu bodů, které se mají vykreslit a vyhladí zobrazovanou trasu. Pokud jsou data zaznamenávána z jednoho místa, zamezí, aby byla vykreslena změť čar. I když uživatel stojí na jednom místě, přijímá pozice vzdálené od sebe v jednotkách metrů.

Na obr. 2.5 je vidět záznam pořízený ve stejnou dobu touto aplikací a komerční aplikací z Google Play Store s názvem RunKeeper. Při vytváření záznamu jsem se pohyboval přesně okolo vnitrobloku, běžel jsem po chodníku a například silnici na ulici Skácelova jsem nepřekročil. Obě aplikace zobrazily data tak jak je získaly ze systému GPS a ze zobrazených dat nešlo učit, jestli uživatel opravdu danou silnici překročil či nikoliv.



Obr. 2.5: Porovnání stejného záznamu různými aplikacemi

Notifikace a Text to Speech

Služba by měla uživatele informovat o svém běhu pomocí notifikací. Používám notifikace zobrazované do notifikační lišty a notifikace zvukové.

Při spuštění služby je přidána probíhající notifikace, která zobrazí informaci o probíhajícím zaznamenávání. Po řádném ukončení služby je notifikace odebrána. V případě, že je služba ukončena nekorektně (odpojení poskytovatele GPS) změní notifikace svoji ikonu, statusu notifikace je odebrán probíhající status a její informace se změní na chybové hlášení.

Pomocí nastavení může uživatel zapnout notifikace zvukové a to buď pomocí klasického zvukového upozornění systému nebo pomocí hlasového syntezátoru, který mluví předem nastavené textové řetězce. Kromě oznámení začátku záznamu je možnost periodického upozorňování na základě vzdálenosti (každých 100 m, 500 m, 1 km, 2 km, 5 km), nebo uplynulého času (každou 1 min, 2 min, 5 min, 10 min).

Při zvolení notifikace na základě vzdálenosti je z nastavení načten krok, po kterém je notifikace spouštěna a pokaždé, když je přidán záznam, přidá se jeho vzdálenost k předchozímu záznamu do celkové vzdálenosti. V případě, že je tato vzdálenost větší než požadovaná hodnota, je spuštěna notifikace a hodnota kroku je přidána k pomocné proměnné, aby bylo možné určit, kdy má být notifikace znovu spuštěna.

Při zvolení notifikace na základě času je taktéž načten krok četnosti. A po spuštění služby je zavolána funkce, která provede notifikaci, do pomocné proměnné přidá hodnotu kroku a sama se zavolá znovu za nastavený čas, v tomto případě se čas rovná hodnotě kroku.

Před použitím Text to Speech je potřeba službu hlasového syntezátoru spustit, případně nainstalovat. Poté ji lze kdykoliv zavolat s parametrem ve formě textu, který má být syntezován. Je vhodné po dobu mluvení ztlumit ostatní audio signály (například hrající hudbu), toho je dosaženo tím, že před samotným zavoláním funkce pro mluvení zavoláme funkcí `requestAudioFocus`, která toto ztlumení provede. Po dokončení syntezace je třeba zesílit ostatní audio signály. Jelikož je syntezátor asynchronní je implementována callback funkce, která je zavolána při dokončení syntezace a v ní se provést zesílení audia.

Zobrazení virtuálního partnera na mapě

Pokud uživatel trénuje na některém z předchozím záznamů, je mu na mapě zobrazována pozice z původního záznamu v daném čase běhu. Toho je docíleno podobným principem jako při zvukových notifikací na základě času. Funkce která aktivitě posílá pozici pro vykreslování, se vždy znovu sama zavolá za čas, který je dán rozdílem času bodu následujícího a bodu aktuálního. Tím je docíleno, že se virtuální partner pohybuje po mapě, stejně jako když byl záznam zaznamenán.

Pozastavení běžící služby

Když uživatel během zaznamenávání klikne na tlačítko Pause, je do služby odeslána zpráva, služba má implementovánu callback metodu, která na toto kliknutí reaguje. Při pozastavení je zaznamenán aktuální systémový čas a je zastaveno ukládání dat. Při znovu spuštění je také zaznamenán aktuální čas a délka trvání pauzy je určena jako rozdíl těchto časů.

Toto pozastavení ovlivní ostatní prvky služby. Pro výpočet časů mezi jednotlivými kontrolními body a pro vytvoření virtuálního partnera je potřeba, aby byla informace o času každého bodu uložena i s kompenzací pauzy. Při ukládání informací o bodu je čas ukládán jako rozdíl aktuálního bodu a prvního bodu mínus celková doba pauzy.

Další ovlivněnou částí služby je funkce pro zvukovou notifikaci pomocí času a virtuální partner, které fungují na stejném principu. V okamžiku, kdy je služba informována o pauze, jsou již tyto metody uspány a jsou zavolány za předem nastavený čas. Aby nebyly zavolány během pauzy, jsou zrušeny, nicméně nelze zjistit kolik času zbývá do jejich spuštění. Pokaždé, než se uspí, zaznamenají hodnotu systémového času. Rozdíl času začátku pauzy a tohoto času udává, kolik již uplynulo času od uspání. Při znovu spuštění jsou uspány na zbývající čas. Pokud by proběhlo více pauz během jediného uspání je třeba mít uloženou hodnotu pauz, které proběhly od posledního uspání a tuto hodnotu odečíst od hodnoty, o kterou se bude provádět zbývající uspání.

Výpočet informací ze zaznamenaných dat

Po ukončení zaznamenání dat je spuštěna metoda pro vypočítání a uložení dat pro záznam. Na začátku se stanový čas trvání záznamu, vypočítají kalorie spálené během záznamu a průměrná rychlost. Pomocí metody `Geocoder` je zjištěna adresa začátku záznamu, z této adresy je zjištěno město, kde záznam probíhal. Pokud záznam není tréninkem, určí se po jakých časových úsecích mají být definovány kontrolní body. Pokud záznam trvá víc než 10 min je toto rozmezí dáno jako desetina délky trvání záznamu, jinak se rozdělí po minutě.

Následně se projdou záznamy položka po položce, hledá se maximální rychlost, minimální a maximální nadmořská výška a počítají se nastoupané a sestoupané metry nadmořské výšky. Pro graf nadmořské výšky je nad samotnými daty z důvodu nízké přesnosti aplikován klouzavý průměr. Délka průměru pro jeden bod je pětina všech bodů před i za aktuálním bodem. Toto nastavení vyhladilo optimálně graf, ale výpočet klouzavého průměru trval pro delší záznamy i několik minut. Proto byla délka upravena na 30 bodu před a za, což zmenšilo dobu výpočtu na pár desítek sekund za cenu akceptovatelného zhoršení vyhlazení grafu.

Dále je načten a upraven soubor `main` a do souboru `logslist` jsou na začátek přidány informace o aktuálním záznamu.

Pokud záznam není tréninkem, hledá se u každého bodu zda je jeho čas už je vyšší než čas pro kontrolní body. V kladném případě je kontrolní bod uložen. Ukládá se zeměpisná šířka a délka bodu a čas mezi tímto a předchozím kontrolním bodem. Pokud je záznam tréninkem, načte se soubor s kontrolními body trénovaného záznamu a je postupně hledán nejbližší bod pro daný kontrolní bod. Provádí se měření vzdálenosti mezi aktuálním a kontrolním bodem. V okamžiku, kdy se tato vzdálenost přestane zmenšovat, a je menší než 40 m, a tři po sobě jdoucí vzdálenosti se začnou zvětšovat (uživatel kontrolní bod přeběhl a vzdaluje se od něj), je nejmenší vzdálenost brána jako dosažení kontrolního bodu. Čas na úseku tímto a předchozím kontrolním bodem je odečten. Porovná se z příslušným časem tréninkového záznamu a určí se jestli byl uživatel lepší nebo horší než při tréninku. Po skončení procházení záznamu je zjištěno, jestli je kontrolních bodů stejný jako v tréninkovém záznamu, pokud ne (záznam mohl být ukončen dřív než byly tři vzdalující se body), je mezi posledními body záznamu hledán poslední kontrolní bod. Pak se upraví seznam položek v souboru pro informace o záznamech tréninku a je upraven soubor `logslist`. Problémem této metody je, že pokud jsou zaznamenávaná data příliš nepřesná nebo uživatel běží jinou trasou a v záznamu není bod se vzdáleností do 40 m, nejsou kontrolní body spočítány korektně.

Poté se stanoví hodnoty pro osu y grafu: maximální, minimální a střední hodnota nadmořské výšky a pro osu x časy pro jednotlivé pětiny záznamu. Dále je stažen obrázek reprezentující mapu záznamu pomocí Google Static Maps API, je vytvořen obrázek o velikosti 200x200 pi v úrovni přiblížení takové, aby byl zobrazen celý záznam. Proto stačí poslat pouze každý třicátý bod ze záznamu.

Nakonec jsou na externí úložiště zažízení zapsány všechny vytvořené a upravené soubory. Pro půlhodinový záznam běhu trvá vypočítat a zapsat tato data kolem 30 sekund.

Komunikace mezi aktivitou a službou

Při přepnutí na záložku aktivity se mění obsah textového pole v závislosti na přesnosti získávaného GPS signálu. Pokud je přesnost lepší než nastavená, je pole zelené a služba jde spustit. Pokud je přesnost horší, pole je červené a služba nejde spustit. Pokud uživatel poklepe na tlačítko start spustí se služba, popis toho tlačítka se přepíše na Pause a zároveň se povolí tlačítko pro ukončení služby. Zároveň je spuštěn prvek `Chronometr`, který zobrazuje čas trvání záznamu. Dále se aktivita zaregistruje pro získávání a vykreslování aktuální pozice na mapě.

Po spuštění služba čeká na data z GPS a jsou-li dostatečně přesná spustí se záznam trasy a případně se spustí metody na hlasové notifikace. Jak jsou získávána další data z GPS, vyhovující zadaným podmínkám, jsou ukládána a posílaná do aktivity. Ta zobrazí aktuální rychlost, uběhnutá vzdálenost, souřadnice aktuálního a předchozího bodu pro vykreslení trasy záznamu na mapě a délku trvání záznamu, která je důležitá pro správné nastavení prvku `Chronometr` při přechodu aktivity zpět do aktivního režimu (například pokud uživatel kontroluje informace pouze výjimečně).

V případě, že uživatel v seznamu záznamů vybere některý ze záznamů pro trénink, je jméno tohoto záznamu uloženo do souboru `train`. Pokud aktivita nalezne soubor `train`, kromě výše zmíněného postupu měří vzdálenost mezi aktuální pozicí a začátkem tréninku. Dokud tato vzdálenost není menší než 10 m je zobrazen nápis informující uživatele, aby se přiblížil k tomuto bodu. Zároveň je na mapě vykreslen počáteční bod a kontrolní body. Během záznamu jsou aktivitě též zasílány informace pro vykreslování pozice virtuálního partnera.

Poklepáním uživatele na tlačítko Stop služba zastaví zaznamenávání dat a spustí metodu pro výpočet a uložení informací o záznamu. Po dokončení této metody se služba ukončí.

2.3.5 Aktivita zobrazující seznam záznamů

Třetí a poslední záložkou v hlavním záložkovém schématu je tato aktivita. Slouží k zobrazování, výběru, mazání, exportování a označování záznamů. Je zobrazena na obr. 2.6, v levé části je seznam běžných záznamů, v pravé části seznam záznamů tréninku.

Seznam záznamů je načítán ze souboru `logslist`, v tomto souboru je pro každý ze záznamů uloženo: název, město pořízení, délka, uběhnutá vzdálenost, datum pořízení a informace o tréninku. Záznamy jsou chronologicky řazeny od nejnovějších.

Pokud uživatel poklepe na záznam je spuštěno záložkové schéma zobrazující informace o záznamu. Pokud je poklepání dlouhé, zobrazí se kontextová nabídka, umožnující trénovat na daném záznamu, smazat záznam a exportovat záznam do formátu GPX. V následující části bude popsáno jak fungují jednotlivé komponenty této aktivity.

Zobrazení seznamu záznamů

K zobrazování seznamu záznamů se používá třída `ListView`, v aktivitě je vytvořena instance této třídy, do které je vložen `ArrayAdapter`, který obsahuje data pro zobrazení každého ze záznamů.

Abychom vytvořili vlastní vzhled, je potřeba vytvořit si vlastní `ArrayAdapter`, kde je pomocí XML souboru definován vzhled každé položky seznamu. Pomocí přepsání metody `getView` jsou tomu layoutu přiřazena data. Konstruktor třídy `ArrayAdapter` je upraven, aby tak jako parametr převzal list objektů, které jsem vytvořil pro uložení informací o jednotlivém záznamu.

Pro zrychlení pohybu v seznamu (skrolování) je použit `ViewHolder`, obsahuje informace o položkách layoutu aby nemusely být při každém prvku seznamu vytvářeny znovu.

Obr. 2.6: Seznam běžných záznamů a záznamů tréninku

Organizace tréninkových záznamů

Aplikace umožňuje na jednotlivých záznamech trénovat, tato informace je v seznamu zobrazena slovem training u jednotlivých záznamů. Pokud uživatel na některém ze záznamů trénuje po skončení tréninku, je tento záznam vložen na začátek seznamu (nejnovější záznam). Podle názvu originálního záznamu je vytvořen seznam souborů tréninků (obsahuje originální záznam a všechny další tréninky na stejné trase)

a původní záznam je ze seznamu odebrán. Při poklepání na tréninkový záznam se zobrazí seznam všech tréninků na daném záznamu (pravá část obr. 2.6). Pokud uživatel trénuje na tréninku je odkaz na tento záznam uložen do souboru se seznamem tréninků originálního záznamu této trati. Maximální úroveň vnoření seznamu je 2, byla zvolena pro přehlednost (například pokud uživatel každý den trénuje na včerejším záznamu), aby nevznikal mnohaúrovňový a nepřehledný seznam záznamů.

Smazání záznamu

Pří poklepání uživatele v kontextovém menu na položku smazání záznamu a potvrzení v následujícím dialogu je ve vlastním vlákně (z důvodu nezatěžování vlákna hlavního) provedeno smazání záznamu.

Nejdříve jsou smazány vlastní soubory záznamu, poté je načten upraven a uložen soubor `main`.

V případě, že se jednalo o běžný (ne tréninkový) záznam, je položka reprezentující tento záznam odstraněna z aktuálně zobrazovaného seznamu a upravený seznam je uložen. Pokud je odstraňovaný záznam tréninkový, neupravuje se hlavní seznam záznamů, ale konkrétní seznam tréninkových záznamů.

Exportování záznamu do formátu GPX

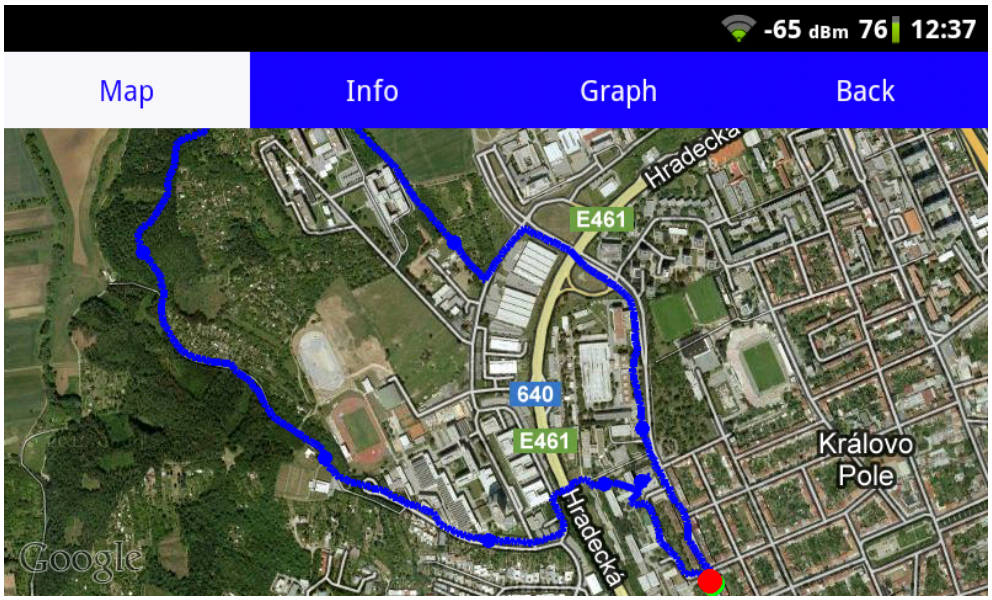
Tato položka kontextového menu umožňuje konvertování záznamu do formátu GPX a jeho následné uložení na externí úložiště zařízení. Formát GPX je standardní formát pro přenos a uchovávání GPS dat, exportování záznamu do tohoto formátu umožní uživateli záznam přenést a například na počítači zobrazit.

GPX je textový XML formát, pro jeho zapisování do souboru používám metodu `FIleWriter`. Na začátek souboru zapíšu standardní hlavičku GPX souboru, za ni zapíšu značky pro záznam trasy a za ně postupně mezi značky pro jednotlivý bod zapíšu zeměpisnou šířku a délku, nadmořskou výšku a čas pořízení bodu. Nakonec zapíšu ukončovací značky pro záznam trasy a koncové pro soubor GPX.

2.3.6 Aktivita pro zobrazení záznamu na mapě

Při poklepání na soubor v aktivitě pro zobrazení seznamu záznamů je spuštěno záložkové schéma pro zobrazení informací o záznamu. Při spuštění je jako parametr uveden název souboru, který se má zobrazit. Toto schéma obsahuje čtyři záložky, záložku pro zobrazení záznamu na mapě, záložku pro zobrazení informací o daném záznamu, záložku pro zobrazení grafu a záložku pro vrácení zpět na seznam záznamů. Toto záložkové schéma umožňuje pomocí uživatelského nastavení vybrat, která z těchto záložek bude zobrazena jako první.

Aktivita pro zobrazení záznamu na mapě je při výchozím nastavení zobrazována jako první a slouží k zobrazení uloženého záznamu na mapě, ke kterému používá třídu `MapView`. Její zobrazení je na obr. 2.7. Při vytváření této aktivity jsem se zabýval několika problémy, jejichž řešení uvedu níže.



Obr. 2.7: Aktivita pro zobrazení záznamu na mapě

Spouštění aktivity

Jak již bylo zmíněno výše, nevhodným návrhem ukládání souboru bylo způsobeno dlouhé čtení souboru. Jelikož se soubor musel načíst na začátku aktivity, prodlužovalo toto načítání čas od poklepání na záznam po zobrazení záznamu. Tento problém byl vyřešen změnou návrhu souborů, v současném návrhu je již tento čas přijatelný.

Čtení souboru ale nebylo jediným místem, kde se při spouštění aktivity čekalo. Vytvoření `MapView`, na které se zobrazovaný záznam vykreslí, trvá kolem 600 ms. Při dalším použití `MapView` dojde k jeho recyklaci a vytvoření trvá kolem 20 ms. Tím, že používáme `MapView` i v aktivitě pro zobrazování aktuálních informací o záznamu, v době zobrazování této aktivity již `MapView` pouze recyklujeme. Aplikací používaná záložková schémata při svém startu předpřipraví zobrazení všech svých záložek. V tomto momentě dojde k vytváření `MapView`, aplikace startuje a nastává vhodná doba, kdy by k čekání na reakci aplikace mělo dojít.

Tato aktivita je jediná, ve které se načítá záznam trasy. Po načtení a zobrazení záznamu ověří, zda je pro daný záznam dostupný obrázek s mapou a místo, kde byl záznam pořízen. Případně tyto informace zjistí a uloží. Pro jejich zjištění je potřebné internetové připojení, které nemuselo být dostupné v době ukončení záznamu.

Vykreslení záznamu trasy na mapě

Pro vykreslování objektů na mapu používám třídu, která vznikla děděním třídy `Overlay`. V této třídě byla přepsána metoda `draw()` sloužící pro vykreslování objektů. Tato třída má na vstupu dva objekty typu `GeoPoint`, které jsou vytvořeny na základě jejich zeměpisné délky a šířky a dva číselné identifikátory, jeden pro barvu, druhý pro určení typu. Metoda `draw()` pomocí rozhraní `Projection` přepočítá vstupní objekty `GeoPoint` na souřadnice bodů x a y na displeji a na základě identifikátoru typu a barvy vykreslí mezi body čáru, nebo v místě prvního bodu vykreslí kruh. Tento objekt typu `Overlay` přidám do vrstvy mapy, která je následně zobrazena.

Pro označení začátku a konce trasy vykreslím zelený a červený kruh, dále v místě každého kontrolního bodu vykreslím modrý kruh. Pokud trasa nebyla tréninkem, je jako barva použitá modrá. Pokud tréninkem byla, v úsecích mezi kontrolními body, kde byl trénink rychlejší než originál, je použitá zelená barva a v místech, kde byl trénink horší, barva červená.

Při vykreslování tedy vykreslím kruhy (začátek, kontrolní body, konec) a pro každé dva následující body trasy provedu jejich konverzi na `GeoPoint`. Vytvořím `Overlay` s čarou mezi nimi a přidám jej na mapu. Samotný proces vykreslení na mapu není časově náročný. Problém nastává při přibližování, oddalování a posouvání po mapě. Tyto procesy jsou při používání subjektivně pomalé a trhané, v závislosti na počtu vykreslených bodů. Pokud je bodů vykresleno pár set (kratší záznam) problém s mapou není, pokud je ale bodů kolem tisíce (delší záznamy) je toto zpomalení znatelné.

Částečné řešení je, při vykreslování nepřipravovat vrstvu pro stíny, jelikož nejsou používány. Tím dojde ke znatelnému zrychlení, ale při větším množství bodů je stále zpomalení znatelné. Při řešení tohoto problému jsem našel a implementoval několik způsobů vykreslování objektu na mapu. V jednom z nich je například při vytváření objektu `Overlay` předáno pole s příslušnými objekty `GeoPoint`, celá trasa je vykreslena v jednom objektu `Overlay` a ten je přidán na mapu. Tato změna ale nepřinesla žádané zlepšení. Všechny ostatní řešení totiž používaly stejný princip jako je používán zde. Jiný způsob vykreslování objektů na mapu jsem nenalezl.

Při porovnání stejně obsáhlého záznamu s již zmíněnou aplikací RunKeeper je zpoždění posunu po mapě znatelné i v této aplikaci. Tento problém by mohl vyřešit systém, který by redukoval počet vykreslovaných bodů s ohledem na úroveň přiblížení a s ohledem na část aktuálně viditelného záznamu.

2.3.7 Aktivita zobrazující informace o trase

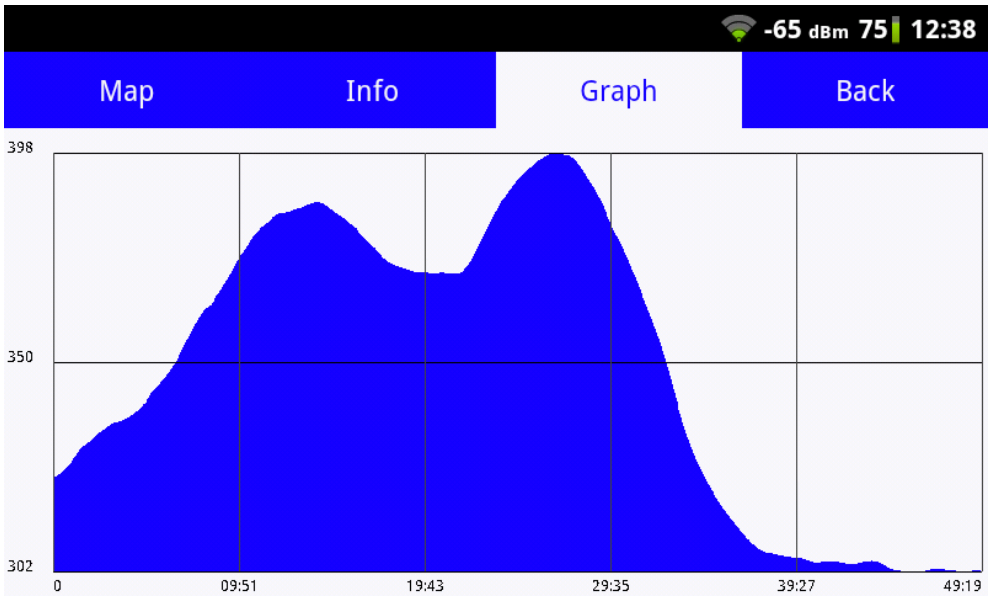
Tato aktivita zobrazující informace o daném záznamu je zobrazena na obr 2.8. Při startu si tyto informace ve vlastním vlákně načte ze souboru a poté je zobrazí. V případě že, se jedná o záznam tréninku, je zobrazen také počet úseků s lepším časem.

Obr. 2.8: Aktivita zobrazující informace o záznamu trasy

2.3.8 Aktivita zobrazující graf

Aktivitu zobrazuje graf závislosti nadmořské výšky na čase, je zobrazena na obr. 2.9. Na začátku ve vlastním vlákně načte soubor a ten poté předá do třídy `GraphView`. Tato třída byla převzata ze zdroje [13].

Třídě je předáno pole prvků `float` reprezentující hodnoty grafu a dvě pole prvků `String` reprezentující popisky obou os. Jelikož k vykreslení grafu používá standardní metody pro vykreslování na displej je snadno pochopitelná a upravitelná. Třídu jsem upravil, aby byla v souladu s barevným schématem aplikace a pro dané určení



Obr. 2.9: Graf závislosti nadmořské výšky na čase

data vykreslovala úhledně a ostře. Aplikace je určena na zaznamenávání běhu, který většinou trvá alespoň 20 minut. Pokud je záznam krátký, například pár minut, je vykreslený graf jednak nepřesný a změna mezi jednotlivými body není hladká.

2.3.9 Uživatelské nastavení Aplikace

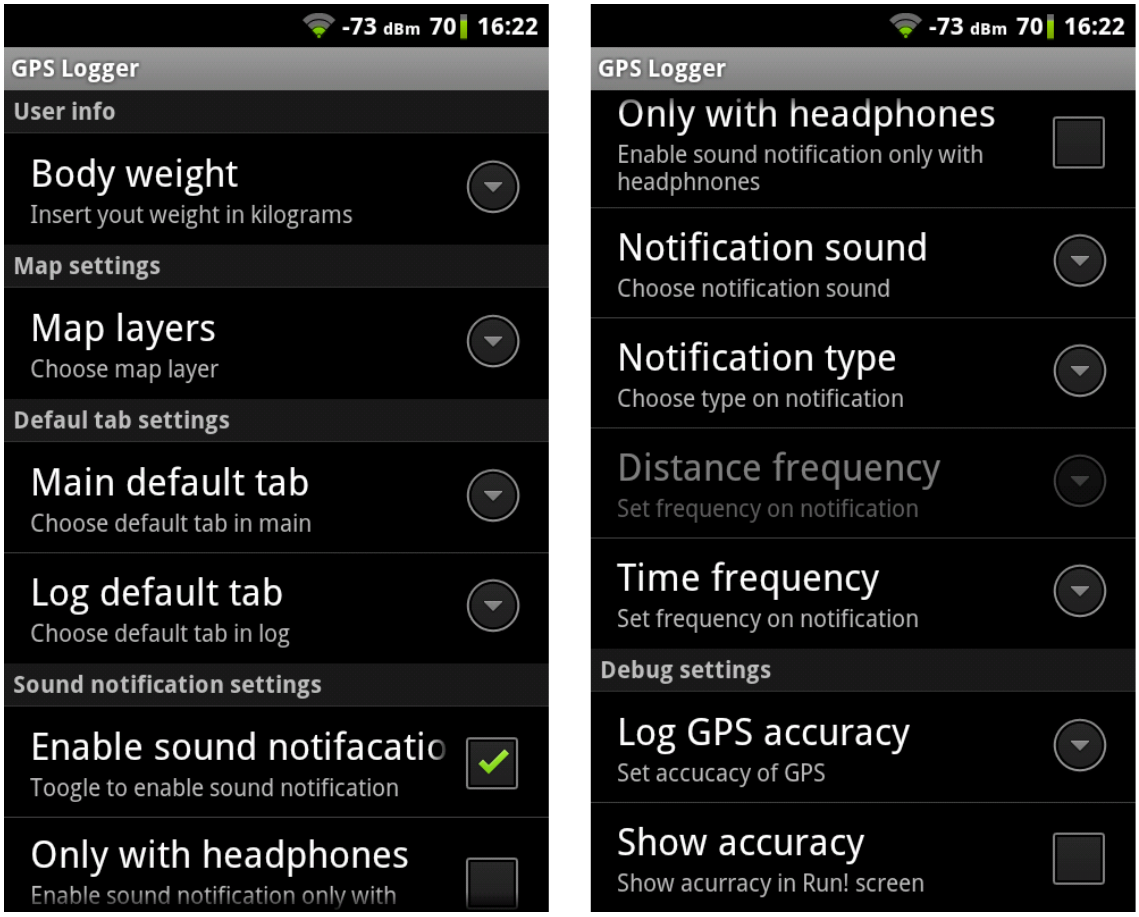
Pokud uživatel v libovolné aktivitě zmáčkne tlačítko menu zobrazí, se standardní menu systému android s možností zobrazit aktivitu pro uživatelské nastavení. Tato aktivita je zobrazena na obr. 2.10 a používá standardní systémový vzhled pro nastavení.

V každé aktivitě, která některou z položek nastavení používá, je implementována callback metoda, která je zavolána vždy, pokud dojde ke změně v některé z položek nastavení.

Nastavení obsahuje:

- položku pro zadání tělesné váhy, používá se pro výpočet spálených kalorií;
- položku pro zvolení mapového podkladu v prvcích `MapView` a to buď klasický nebo satelitní;
- volbu základní aktivity v obou záložkových schématech;
- nastavení zvukových notifikací a to volbu zvukové nebo hlasové notifikace, určení podle čeho se bude notifikace řídit, zda podle vzdálenosti nebo podle času a nastavení jejich četnosti, u zvukové notifikace je i volba zda se má notifikace přehrát pouze když jsou do zařízení zapojeny sluchátka;

- testovací nastavení umožňující v aktivitě, která zobrazuje aktuální informace o záznamu, zobrazovat aktuální přesnost GPS dat;
- nastavení hranice přesnosti, kterou používá služba pro zaznamenávání dat.



Obr. 2.10: Aktivita pro uživatelské nastavení

ZÁVĚR

Aplikace vypracovaná v této práci byla vytvořena se zaměřením na rekreační běh. Současná verze aplikace umožňuje zaznamenat trasu na externí úložiště zařízení a během zaznamenávání zobrazuje na mapě informace o aktuální pozici a již uběhnutou trasu. Umožňuje zaznamenanou trasu zobrazit na mapě a zobrazit informace o ní (délku trvání, uběhnutou vzdálenost, průměrnou a maximální rychlost, časy na kontrolních bodech...). Dále umožňuje zobrazit graf závislosti nadmořské výšky na čase. Obsahuje uživatelsky přívětivý seznam zaznamenaných tras s možností každou z tras exportovat do formátu GPX nebo na záznamu trénovat. Při tréninku je na mapě zobrazován virtuální partner, který reprezentuje původní běh. V celé aplikaci je dostupné menu s uživatelským nastavením.

Aplikace byla vyvíjena a optimalizována s ohledem na specifika mobilního systému. Snažil jsem se co nejvíce minimalizovat počet prováděných výpočtů někdy i za cenu akceptovatelného zhoršení výsledku (klouzavý průměr u grafu). Dále jsem se snažil aby celá aplikace byla svižná, plynulá a uživatel nečekal na interakci s aplikací. Aktivita pro zobrazení záznamu na mapě není při vykreslení mnoha položek na mapu optimálně plynulá. Dále jsem se zaměřil na spotřebu baterie při používání aplikace. Optimalizoval jsem aplikaci s důrazem na zatížení procesoru, protože pro zjišťování polohy je použit integrovaný modul GPS, který je energeticky náročný a jehož spotřebu nelze ovlivnit.

Při srovnání této aplikace s jinými aplikacemi pro zaznamenávání sportovních aktivit například již zmiňovaný RunKeeper je hlavní výhodou ostatních aplikací webová služba, která je spřažená s konkrétní aplikací. Při zaznamenání je záznam nahrán do webové služby, která umožňuje zobrazení a správu záznamů na počítači. Dále obsahují sociální aspekt, ať už napojení na stávající sociální síť (facebook, twitter) nebo použití sítě vlastní (sdílení záznamů s ostatními uživateli). Proto tato aplikace v současné verzi nemá komerční potenciál a případnou konkurenční výhodu, což ale nebylo smyslem této práce.

LITERATURA

[1] MURPHY, M.L. *Beginning Android 2*. Apress, 2010. 416 s. ISBN: 978-1-14302-2629-1.

[2] GARGENTA, M. *Learning Android*. O'Reilly, 2011. 268 s. 978-1-449-39050-1.

[3] *The Developer's Guide* [online]. Poslední aktualizace 17. 11. 2011 [cit. 22. 11. 2011]. Dostupné z URL: <http://developer.android.com/guide/index.html>.

[4] *Platform Versions* [online]. Poslední aktualizace 3. 11. 2011 [cit. 22. 11. 2011]. Dostupné z URL: <http://developer.android.com/resources/dashboard/platform-versions.html>.

[5] *What is Android?* [online]. Poslední aktualizace 1. 12. 2011 [cit. 3. 12. 2011]. Dostupné z URL: <http://developer.android.com/guide/basics/what-is-android.html>.

[6] *Activities* [online]. Poslední aktualizace 1. 12. 2011 [cit. 3. 12. 2011]. Dostupné z URL: <http://developer.android.com/guide/topics/fundamentals/activities.html>.

[7] *Services* [online]. Poslední aktualizace 1. 12. 2011 [cit. 3. 12. 2011]. Dostupné z URL: <http://developer.android.com/guide/topics/fundamentals/services.html>.

[8] *Google Projects for Android: Google APIs* [online]. Poslední aktualizace 2011 [cit. 3. 12. 2011]. Dostupné z URL: <http://code.google.com/intl/cs-CZ/android/add-ons/google-apis/maps-overview.html>.

[9] *Static Maps API V2 Developer Guide* [online]. Poslední aktualizace 2012 [cit. 22. 4. 2012]. Dostupné z URL: <https://developers.google.com/maps/documentation/staticmaps/>.

[10] HEROUT, P. *Učebnice jazyka Java*. Kopp, 2000. 349 s. ISBN: 80-7232-115-3.

[11] GUIHOT, H. *Pro Android Apps Performance Optimization*. Apress, 2012. 279 s. ISBN: 978-1-4302-4000-6.

[12] *Installing the SDK* [online]. Poslední aktualizace 2011 [cit. 4. 12. 2011]. Dostupné z URL: <http://developer.android.com/sdk/installing.html>.

[13] *GraphView* [online]. Poslední aktualizace 2011 [cit. 5. 12. 2011]. Dostupné z URL: <http://android.arnodenhond.com/components/graphview>.

A OBSAH PŘILOŽENÉHO CD

Přiložené CD obsahuje tyto soubory:

- `bakalářská_práce.pdf` – elektronická verze bakalářské práce;
- `GPS_Logger.apk` – vytvořená aplikace, pro nainstalování nakopírovat do telefonu a spustit (nutno mít povoleno: Nastavení - Nastavení aplikací - Neznámé zdroje);
- `ukázková_data.zip` – ukázková data pro aplikaci po nainstalování aplikace extrahovat tento soubor do externího úložiště do adresáře `Android/data`;
- `zdrojový_kód.zip` – archiv s kompletními zdrojovými kódy programu z IDE Eclipse.