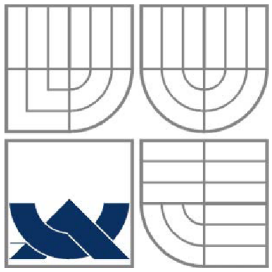
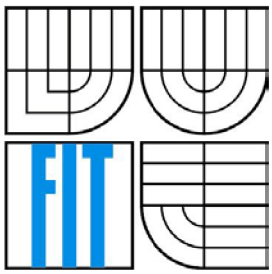


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

KOMUNITNÍ WEB PRO TURISTY

COMMUNITY WEB FOR TURISTS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

ADAM SATRAPA

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. MARTIN HRUBÝ, Ph.D.

BRNO 2010

Abstrakt

Tato bakalářská práce přibližuje návrh, vývoj a testování aplikace, určené především turistům. Aplikace je rozdělena do dvou částí. Serverová část zajišťuje hlavně organizaci uživatelů, nahrávání uživatelských tras, jejich správu a prohlížení, dále pak začleňování uživatelů do skupin a plánování výletů. Druhá část je zastoupena aplikací pro mobilní telefony, která umožňuje uživatelům nahrávat svou aktuální pozici v reálném terénu z GPS zařízení na server a zároveň sledovat pozici ostatních uživatelů.

Abstract

This Bachelor thesis approaches design, development and testing of application designed mainly for tourists. Application is divided into two parts. The server part provides the main organization of users, recording of user tracks, tracks management and viewing, as well as integration into user groups and trip planning. The second part is represented by application for mobile phones, that allows users to upload their actual position in real terrain from GPS device onto a server and also monitor position of other users.

Klíčová slova

mapy, trasy, web, Java ME, PHP, PostgreSQL, PostGIS, GPX, GPS, OpenLayers

Keywords

maps, tracks, web, Java ME, PHP, PostgreSQL, PostGIS, GPX, GPS, OpenLayers

Citace

Satrapa Adam: Komunitní web pro turisty, bakalářská práce, Brno, FIT VUT v Brně, 2010

Komunitní web pro turisty

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Martina Hrubého, Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Adam Satrapa
19. května 2010

Poděkování

Rád bych poděkoval vedoucímu Ing. Martinu Hrubému, PhD. za vedení práce a užitečné rady a nápady, které mi poskytl, a stejně tak za zapůjčení GPS přijímače. Poděkování patří také mému příteli Jiřímu Nováčkovi za jeho připomínky a návrhy při řešení některých problémů a pomoc při testování.

Obsah

1 Úvod

Dnešní dobu bychom mohli, z pohledu uživatelů Internetu, nazvat dobou sdílení. Lidé připojení na Internet dnes sdílí se svými přáteli skrze různé služby téměř vše. Sdílíme osobní údaje, fotografie, videozáznamy a dokonce i aktuální zážitky a pocity. Během pár vteřin můžeme zjistit, co se právě odehrává na opačném konci světa bez jakýchkoli zprostředkovatelů. To, co bylo dříve téměř nemožné, se pomalu stává naprosto běžným.

V posledních letech se také rozmáhá používání zařízení, umožňujících příjem signálu GPS. Velmi často se lze setkat s použitím GPS zařízení například v automobilech, lodích, letadlech a jiných dopravních prostředcích. Dalším neméně podstatným příkladem může být snaha výrobců mobilních telefonů vybavovat nové telefonní přístroje GPS moduly. Proto nástup internetových služeb, využívajících globálního pozičního systému, zaznamenal velký rozmach. Nejlépe je to vidět na rychle se rozšiřující komunitě uživatelů zapojených do hry známé jako Geocaching.

Další relativně novou službou je poskytování mapových podkladů přes web. Velké internetové společnosti jako například Google, Yahoo! či Microsoft umožňují běžným uživatelům využívat služby spojené s mapami. Uživatelé si mohou například vyhledat nejbližší restauraci, zjistit aktuální počasí v daném místě nebo plánovat trasy podle zadaných kritérií. Výhodou těchto služeb je, že valná většina z nich je otevřená vývojářům webových aplikací přes rozhraní API a při splnění určitých podmínek, může tyto služby i mapové podklady kdokoliv začlenit do své webové aplikace.

Tato bakalářská práce spojuje výhody GPS, map a sdílení dat mezi lidmi. Prvotním záměrem tedy bylo vytvořit webovou aplikaci, která by byla schopná zpracovat soubor ve formátu *GPX*. Tento formát a jeho účel je vysvětlen v kapitole 4.9. Dalším požadavkem bylo, aby tato aplikace byla schopna uživateli zobrazit informace z těchto souborů v nějaké jednoduché a přehledné formě. V případě tras jsem se rozhodl zobrazovat celý průběh na pozadí mapového podkladu, čímž by uživatel získal velmi podrobný přehled o svém pohybu v terénu. Zároveň jsem chtěl využít další údaje, které formát GPX nabízí. Tím je například nadmořská výška v každém zaznamenaném bodě. Proto jsem se rozhodl do aplikace začlenit graf, který by uživateli názorně zobrazoval vývoj nadmořské výšky během celé trasy. Další informací, kterou lze ze zaznamenané trasy získat, je čas průchodu každým bodem. To mi umožnilo vypočítat celkovou dobu trasy a po dalších výpočtech také průměrnou rychlost a celkovou vzdálenost.

Ve druhé fázi jsem se chtěl zaměřit na správu takto zaznamenaných tras. Mým požadavkem bylo, aby se každý uživatel mohl zaregistrovat do informačního systému a nahrávat všechny své trasy, mazat je, či upravovat a zároveň je dle svých požadavků sdílet s ostatními uživateli. Ještě před tímto krokem by bylo nutné naimplementovat prostředí, zajišťující přihlašování uživatelů, registraci nových uživatelů, ukládání informací o uživatelích a vztazích mezi nimi do databáze, stejně tak i všech nahraných tras.

Poslední fází, kterou jsem se rozhodl zahrnout do své bakalářské práce, by mělo být spojení celého systému s mobilními telefony. Jako hlavní cíl jsem si stanovil vytvoření aplikace pro mobilní telefony, která by umožňovala registrovaným uživatelům nahrávat svojí pozici či dokonce celou trasu přímo na server, kde by se tyto informace ukládaly a ostatní uživatelé by byly schopni tyto informace vidět téměř v reálném čase. Další, velmi zajímavou službou, by bylo sdílení pozice s ostatními uživateli mobilní aplikace. V konkrétním případě by bylo možné, aby jeden uživatel v terénu mohl získat informaci o tom, kde se právě nacházejí jiní uživatelé a tyto informace zobrazit například na kompasu či mapě.

2 Použité technologie

Tato kapitola obsahuje stručný popis technologií a nástrojů, které jsem využil během implementace této bakalářské práce. Do této kapitoly nebyly zařazeny technologie jako například jazyk HTML, CSS, JavaScript, PHP a GPS a to z důvodu, že jejich princip je všeobecně znám.

2.1 Knihovna OpenLayers

Pro zobrazování map bylo nutné použít některou z dostupných knihoven. V dnešní době není o výběr nouze. Všeobecně známá je například knihovna *Google Maps API*, založená na jazyce JavaScript či technologii *Flash*. Po dřívějších zkušenostech s Google Maps API, kdy jsem při práci s touto knihovnou narazil na určitá omezení a různé chybějící funkce, jsem se rozhodl vyzkoušet knihovnu *OpenLayers*.

Tato knihovna, na rozdíl od Google Maps API, nepodporuje technologii Adobe Flash, ale pouze JavaScript, což ovšem v době vzestupu techniky AJAX a JavaScriptu obecně není nijak na škodu . Další výraznou výhodou oproti knihovně od společnosti Google je nepřeberné množství funkcí, které nabízí. Jako příklad lze uvézt možnost importu mapových podkladů v jakékoliv projekci, velmi rozsáhlé možnosti pro práci s vektory (body, linie, oblasti) a v neposlední řadě také velká míra volnosti, co se vzhledu ovládání týče [1]. Obecně bych si dovolil tvrdit, že Google Maps API je vhodné spíše pro menší mapové aplikace (například zobrazení umístění sídla firmy na mapě na firemním webu s několika málo informacemi, např. otevírací doba, telefon, apod.), kdežto OpenLayers jsou velmi vhodným kandidátem pro rozsáhlé mapové projekty.

OpenLayers je tedy open-source knihovna, určená k prohlížení map a práci s mapovými podklady.

2.2 Databázový systém PostgreSQL

Veškerá uživatelská data jsou uskladněna na serveru v databázi *PostgreSQL*. PostgreSQL lze popsat jako objektově-relační databázový systém. Asi největší předností tohoto systému je bezpečnost, spolehlivost a rozšiřitelnost.

PostgreSQL se už od počátku vývoje řadí mezi open-source software. V současné verzi 8.4.3 podporuje v podstatě všechny SQL standardy a poskytuje mnoho funkcí jako například složené dotazy, triggery, cizí klíče, transakce, pohledy, či uložené procedury a mnoho dalších. Velmi vyzdvihovanou vlastností tohoto databázového systému je rozšiřitelnost, díky níž mohou uživatelé přidávat do databáze vlastní datové typy, operátory, funkce, metody indexování či procedurální jazyky [2].

Ve srovnání s velmi rozšířeným systémem *MySQL*, má PostgreSQL několik odlišností, se kterými se potýká každý, kdo se rozhodne přejít z MySQL na PostgreSQL. Jedná se především o rozdíly v názvech identifikátorů (PostgreSQL je case-sensitive), uvozování řetězců či rozdílný význam některých příkazů (např. příkaz `SHOW TABLES` databáze PostgreSQL nezná) [3].

Asi největším problém pro začínající uživatele PostgreSQL je instalace. Na rozdíl od MySQL, která je většinou součástí větších linuxových distribucí, není instalace úplně triviální a chyby, ke kterým může dojít, se velmi těžko řeší [4].

2.2.1 Administrační nástroj phpPgAdmin

Podobně jako lze databáze MySQL spravovat přes webové rozhraní aplikace *phpMyAdmin*, lze i pro administraci databáze PostgreSQL použít nástroj *phpPgAdmin*. Jak už název napovídá, je phpPgAdmin napsaný v jazyce *PHP* a je tedy možné umístit jeho skripty na server do veřejně přístupného adresáře a spravovat tak celou databázi přes Internet téměř odkudkoli.

PhpPgAdmin byl původně pouze odnoží aplikace phpMyAdmin, dnes už jde o naprosto oddělené a nezávislé projekty.

2.3 Geografická nadstavba PostGIS

PostGIS je nadstavba pro databázový systém PostgreSQL, která umožňuje rozšířit databázi o geografické objekty a funkce potřebné pro práci s nimi. PostGIS implementuje veškeré požadavky kladené specifikací *Simple Feature*, které ustanovilo *Open Geospatial Consortium*. Tento standard specifikuje uložení geografických dat v digitální podobě. K popisu geometrických objektů využívá značkovací jazyk *well-known text* (*WKT*) [5].

PostGIS umožňuje práci s několika geoobjekty zapsaných pomocí WKT:

- *POINT* – bod tvořený souřadnicemi X a Y
  příklad: `POINT(6 10)`
- *LINESTRING* – linie tvořená dvěma a více body
  příklad: `LINESTRING(3 4, 10 50, 20 25)`
- *POLYGON* – uzavřená linie tvořená třemi a více body
  příklad: `POLYGON((1 1,5 1,5 5,1 5,1 1),(2 2,2 3,3 3,3 2,2 2))`

Avšak největším přínosem PostGISu jsou funkce, které nabízí pro práci s těmito geoobjekty. V této bakalářské práci využívám zejména funkci *ST_Distance* pro výpočet délky trasy z uložených bodů a dále funkci *ST_Azimuth*, která vypočítá azimut ze zadaného bodu A do bodu B, což je užitečné ve chvíli, kdy uživatel mobilní aplikace požaduje informaci o směru jiných uživatelů.

2.4 Formát GPX

Většina aplikací pro komunikaci s *GPS* a i některé GPS přijímače umí ukládat informace získané z GPS do datového souboru ve formátu GPX. GPX (GPS eXchange format) je formát odvozený z univerzálního jazyka XML a dědí tak všechny jeho přednosti i nedostatky [6].

Díky tomuto formátu lze srozumitelně zaznamenávat trasy, konkrétní body, poznámky o nadmořské výšce či stavu GPS v konkrétním místě a podobně. Velkou výhodou je přenositelnost tohoto formátu a zároveň jeho otevřenost, takže kdokoliv je schopen zaznamenávat data do GPX souboru. Po ukončení zaznamenávání lze tento výsledný soubor dále zpracovat, což je také částečně cílem této bakalářské práce.

Drobnou nevýhodou tohoto formátu je, že kromě důležitých údajů je nutné ukládat i velké množství značek, které vyžaduje samotná specifikace XML a při ukládání velkého objemu dat (například trasy o délce v řádu stovek kilometrů) se může velikost cílového souboru vyšplhat až na několik desítek megabajtů.

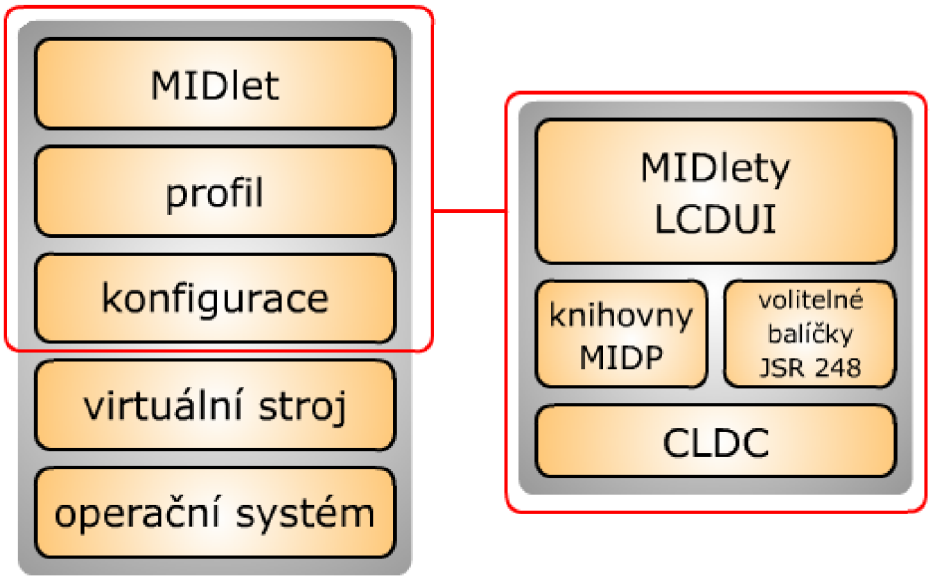
Příklad obsahu souboru GPX je uveden v příloze A.

2.5 Platforma Java ME

Platforma *Java ME* (původním názvem *J2ME*) je součástí rodiny balíků Java, kterou vyvíjí a spravuje společnost Sun Microsystem (dnes je tato společnost již součástí firmy Oracle). Společnost Sun kromě edice Java ME distribuuje také edice pro serverové stanice (Java EE), desktopové stanice (Java SE) a zařízení s velmi omezenými prostředky (Java Card). Platforma Java ME je určena především pro zařízení jako jsou mobilní telefony, set-top boxy, vestavěné systémy a podobně.

Výhodou Javy, ať už je řeč o jakékoliv z výše popsaných edic, je především její přenositelnost. Další velmi podstatnou výhodou Javy ME je flexibilita, kterou vývojářům nabízí. Programátor není například nucen vytvářet si sám uživatelské rozhraní celé aplikace, ale Java mu dovoluje použít grafické prvky, které jsou již zabudovány v samotném operačním systému daného zařízení. Na druhou stranu to přináší problém s tím, že jedna aplikace může na několika zařízeních vypadat velmi odlišně.

Aby bylo možné dosáhnout stavu, kdy je aplikace schopna běžet na každém zařízení s podporou platformy Java ME, bylo nutné definovat tzv. *konfigurace* a k nim přidružené *profily* [7]. Celá organizační struktura je znázorněna na obrázku 2-1.



Obrázek 2-1: Architektura technologie Java ME a konkrétní struktura pro mobilní zařízení

2.5.1 Konfigurace

Jak už bylo řečeno, pro běh aplikace na určitém zařízení, musí toto podporovat konfigurace. Jelikož existuje více druhů zařízení s různým výpočetním výkonem (například výkonnější PDA oproti méně výkonným pagerům), byly vytvořeny dvě základní konfigurace - CLDC a CDC - které určují, které knihovny, funkce a konstrukce jazyka Java a další nezbytnosti bude které zařízení schopno poskytnout. Dále se budu zabývat pouze konfigurací CLDC (Connected limited device configuration), do které spadá většina méně výkonných zařízení, ke kterým se řadí také mobilní telefony.

Tato konfigurace nabízí pouze základní uživatelské rozhraní, omezenou paměť, atd. První verze konfigurace CLDC (označením CLDC 1.0 - specifikace JSR-30) měla kromě jiných omezení i jeden zásadní nedostatek, kterým byla absence aritmetiky reálných čísel, což například u grafických aplikací, vyžadujících práci s desetinnými čísly, bylo velmi limitující [7]. Tento problém řešila většina programátorů po svém a tak vzniklo několik velmi povedených a rozsáhlých knihoven pro práci s reálnými čísly. Příkladem takové knihovny může být Real-Java, jenž definuje nový datový typ Real a operace nad ním (základní početní funkce, goniometrické funkce, odmocniny, apod.).
Následující verze (označení CLDC 1.1 - specifikace JSR 129) už zahrnovala podporu reálných čísel díky datovému typu double, ovšem operace nad tímto datovým typem obsahovaly pouze základní matematické funkce, a proto většina programátorů, vyvíjejících výpočetně náročnější aplikace, zůstala u dříve vytvořených knihoven.

2.5.2 Profily

Profily rozšiřují podřízenou konfiguraci o další třídy podle konkrétního typu zařízení. K těmto třídám lze posléze přistupovat pomocí *API*. Obecně existuje pro každou konfiguraci více profilů, které se vzájemně liší v nabízených knihovnách [8].

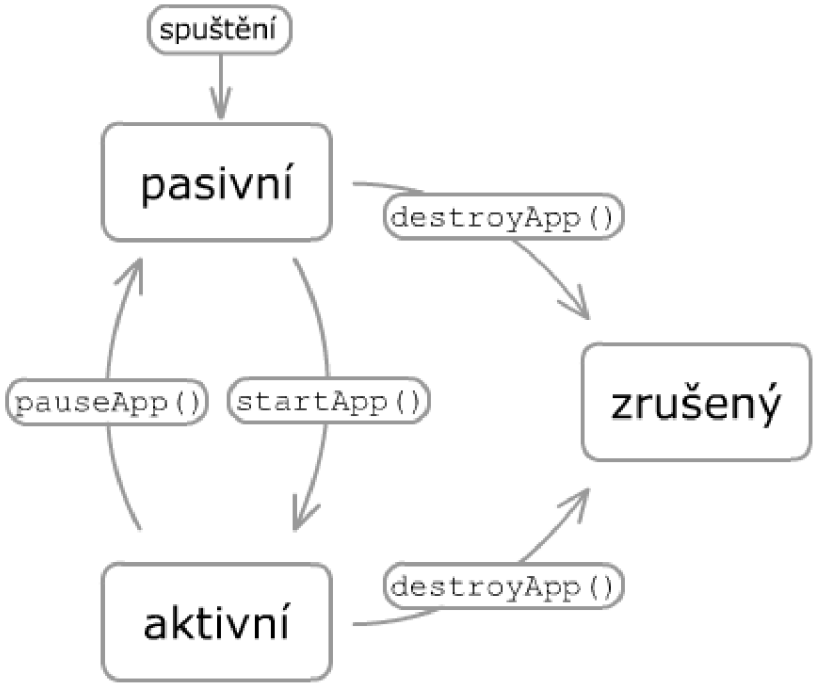
Konfiguraci CLDC lze rozšířit o profil MIDP (*Mobile Information Device Profile)* nebo IMP (*Information Module Profile).* Aplikace pro mobilní telefony zahrnutá v této bakalářské práci je postavena na konfiguraci CLDC s profilem MIDP, proto se ostatními profily nebudu dále zabývat a na dalších řádcích detailněji popíši pouze profil MIDP.

Jak již bylo řečeno, profil MIDP je postaven na konfiguraci CLDC. Jeho hlavním účelem je poskytnout programátorům jednoduché grafické uživatelské rozhraní (*LCDUI*) pro menší displeje mobilních telefonů. Dalším přínosem tohoto profilu je podpora síťových standardů (např. HTTP) a také umožňuje využívat lokální úložiště daného zařízení. Aplikace, které využívají služeb tohoto profilu, jsou obecně nazývány jako *MIDlety*. V současné době Java ME nabízí čtyři verze standardu MIDP. Všechny jsou popsány v následujícím seznamu:

- MIDP 1.0 (říjen 2000)
  - podporuje pouze HTTP protokol
  - žádná podpora pro audiovizuální formáty
  - podpora pouze monochromatických displejů
- MIDP 2.0 (listopad 2002)
  - rozšíření o HTTPS a další síťové protokoly (TCP, UDP, Bluetooth, apod.)
  - podpora větších barevných displejů
  - podpora pro vývoj herních aplikací
  - implementace XML
  - rozšíření o multimediální prvky
- MIDP 2.1 (červen 2006)
  - pouze revize verze 2.0
- MIDP 3.0 (zatím nezveřejněno)
  - pouze ve fázi finálního návrhu
  - spouštění více MIDletů zároveň
  - komunikace mezi MIDlety
  - vylepšení uživatelského rozhraní LCDUI
  - podpora IPv6
  - zabezpečení a distribuce perzistentních dat
  - spousta dalších novinek, vylepšení a změn

2.5.3 MIDlety

Jak už bylo zmíněno výše, MIDlet je aplikace běžící nad profilem MIDP. Chování MIDletů je plně v režii programátora, je ovšem nutné dodržet základní funkční schéma MIDletů. To je uvedeno na obrázku 2-2.



Obrázek 2-2: Stavy MIDletu

Po spuštění aplikace se MIDlet nachází v pasivním stavu. K jeho přepnutí do aktivního stavu dojde zavoláním funkce `startApp()`. V aktivním stavu se spustí tělo MIDletu a ten začne vykonávat svou činnost. Ve chvíli, kdy MIDlet svou činnost dokončí, může se buď ukončit metodou `destroyApp()`, nebo se vrátit do pasivního stavu metodou `pauseApp()`. Všechny tyto tři stavy umožňují programátorovi definovat chování MIDletu při spuštění, zastavení či jeho ukončení [8].

2.5.4 Vývojové prostředí Java ME SDK 3.0

Společnost Sun Microsystems poskytuje programátorům vlastní vývojové prostředí pro vývoj mobilních aplikací v jazyce Java ME pod názvem *Java Platform Micro Edition Software Development Kit 3.0* (zkráceně *Java ME SDK 3.0*). Toto prostředí nahrazuje původní vývojové balíky *Java Wireless Toolkit* a *Java Toolkit for CDC*. Oproti svým předchůdcům je instalace SDK mnohem snazší a většina komponent, jako například sada emulátorů mobilních zařízení, je dostupná ihned po dokončení instalace.

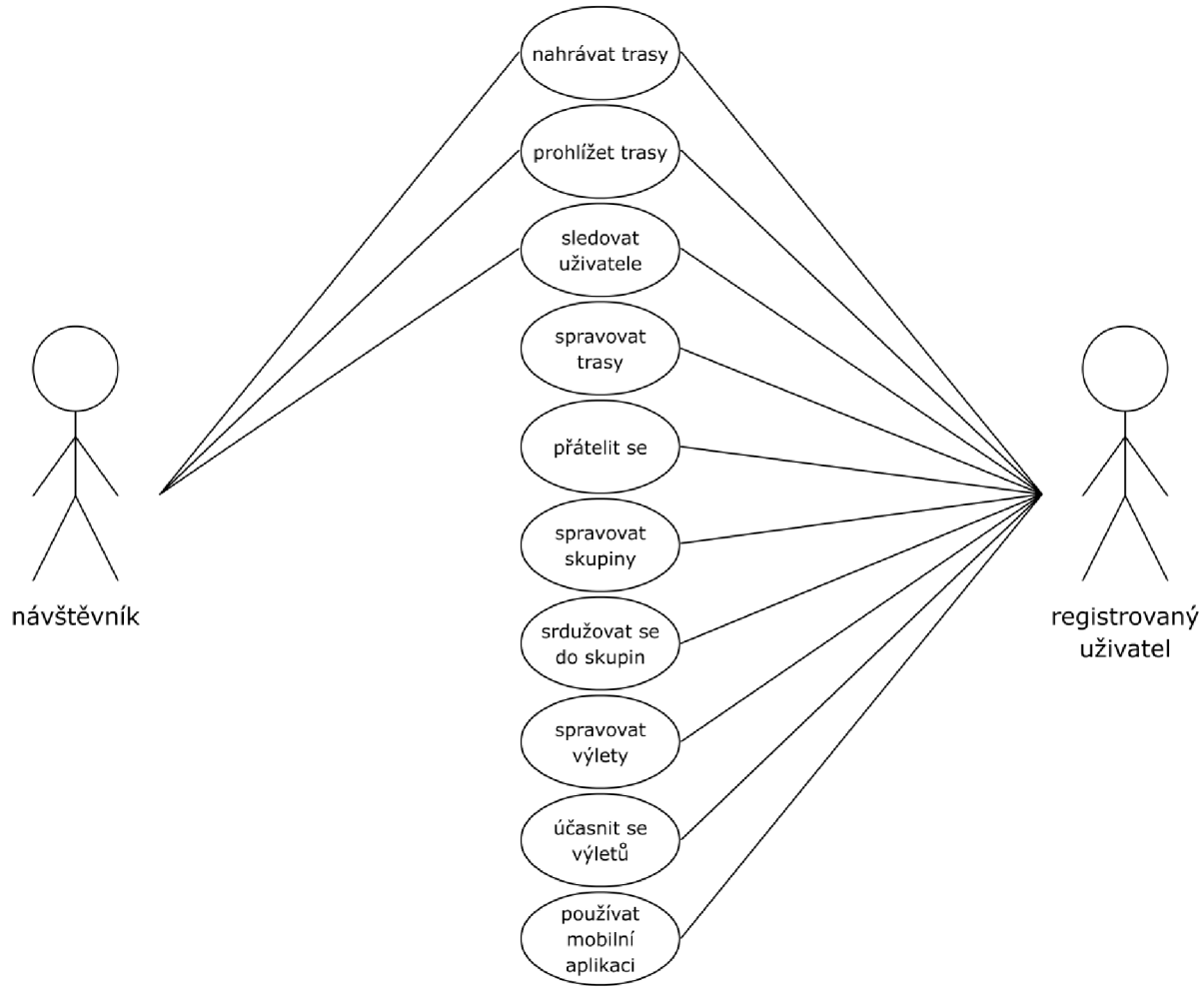
Snad jediným nedostatkem tohoto balíku je mizivá podpora pro testování aplikací využívajících komunikaci přes rozhraní Bluetooth. Existuje sice několik samostatných balíků, které tuto podporu dodatečně přidávají, ale bohužel se mi přes veškerou snahu nepodařilo žádný z těchto balíků během testování alespoň z části zprovoznit. Tento nedostatek jsem vyřešil testováním aplikace na skutečném mobilním zařízení, což ale vzhledem k nutnosti neustálého nahrávání zkompilované aplikace z počítače do paměti tohoto zařízení, bylo časově náročné a nepohodlné.

3 Návrh

Před samotnou implementací systému bylo nutné navrhnout některé důležité části celé aplikace, ať už mobilní či webové. První část návrhu zahrnoval případy použití, databázi, aplikaci pro mobilní telefony a také komunikační protokol mezi touto aplikací a serverem.

3.1 Webová aplikace

Webová aplikace rozlišuje dva typy uživatelů: registrované a neregistrované. Možnosti obou typů uživatelů nejlépe znázorňuje schéma případů užití 3-1. Od tohoto schématu se v podstatě odvíjí celá struktura webové aplikace.



Obrázek 3-1: Schéma případů užití

3.1.1 Navigace

Hlavní menu má pět položek, kde každá položka rozděluje aplikaci na samostatnou sekci. Zároveň je do hlavního menu zabudováno vyhledávací pole.

Pod hlavním menu se nachází podrobnější uživatelský panel, který umožňuje jemnější dělení sekcí na podsekce. Například sekce lidé zahrnuje jak uživatele, tak skupiny. Uživatelský panel má smysl hlavně pro přihlášené uživatele, kteří se přes jeho odkazy mohou rychle přesunout například na seznam všech svých tras či se jedním kliknutím odhlásit. Zároveň uživatelský panel informuje uživatele o nových pozvánkách do skupiny či o požadavcích na spřátelení.

3.1.2 Podpora více jazyků

Už od začátku jsem se snažil navrhnout aplikaci tak, aby bylo možné ve velmi krátkém čase přidat podporu dalších jazyků. Všechny zobrazované texty by měly být pro každý jazyk uloženy v samostatném souboru. Pokud by bylo nutné rozšířit celou aplikaci o další jazyk, stačí pouze vytvořit nový soubor a do něj vložit všechny přeložené texty. Tímto způsobem není nutné výrazně zasahovat do celého systému. Jediným větším problémem bude změna jazyka v obrázcích (například tlačítek hlavního menu), kdy bude nutné vygenerovat pro každý jazyk obrázky nové. Ovšem i tyto změny lze urychlit dobrým návrhem obrázků v grafickém editoru[1].

Navržený systém bude v základním návrhu podporovat český a anglický jazyk.

3.2 Databáze

Pro znázornění vztahů v databázi se nejčastěji využívá tzv. *entity-relationship diagram*. Nejinak tomu bylo při návrhu databáze pro webovou aplikaci. Výsledný diagram je uveden na obrázku 3-2.

Data o uživatelích uchovává tabulka *Uživatel*, přičemž každému uživateli je přidělen jedinečný číselný identifikátor a zároveň si každý uživatel musí zvolit jednoznačné *přihlašovací jméno*, pod kterým bude vystupovat v celém systému. Další atributy jsou buď dobrovolné a uživatel je může zadat během registrace, nebo je systém doplní automaticky (např. *datum registrace*, či implicitní *nastavení*).

Každá trasa, uložená do tabulky *Trasa*, poskytuje informace o obecných údajích, jako je její *název*, *popis* a *datum uložení* do systému. Pro urychlení práce zahrnuje tato tabulka také statistiky, které se vypočítávají po uložení celé trasy do databáze (*celková délka*, *doba*, nejvyšší a nejnižší *nadm. výška*, apod.). Dále je pak každé nové trase přiřazen jedinečný 13 znakový identifikátor, jenž je viditelný pro uživatele. Příznak *způsob záznamu* dělí každou uloženou trasu do dvou skupin – na trasy nahrané staticky (jako GPX soubor přes webové rozhraní) a na trasy nahrané dynamicky přes mobilní aplikaci. S druhou skupinou také souvisí další příznak *stav záznamu*, který poskytuje přehled o tom, zdali bylo nahrávání trasy již ukončeno nebo zdali stále probíhá její ukládání do databáze.

Ke každé trase se váže množina bodů, kterými je konkrétní trasa tvořena. Body jsou ukládány do tabulky *Bod* a jediným povinným parametrem každého bodu je jeho *souřadnice* vyjádřená reprezentací WKT (viz. 4.7). Ostatní atributy jsou defaultně nastaveny jako bez hodnoty z důvodu, že některé GPX soubory neobsahují informaci o jejich hodnotách (např. nadmořská výška, ale někdy i čas), což ovšem také znemožňuje vypočítat k těmto trasám statistiky o průměrné rychlosti, celkové době trasy a podobně.

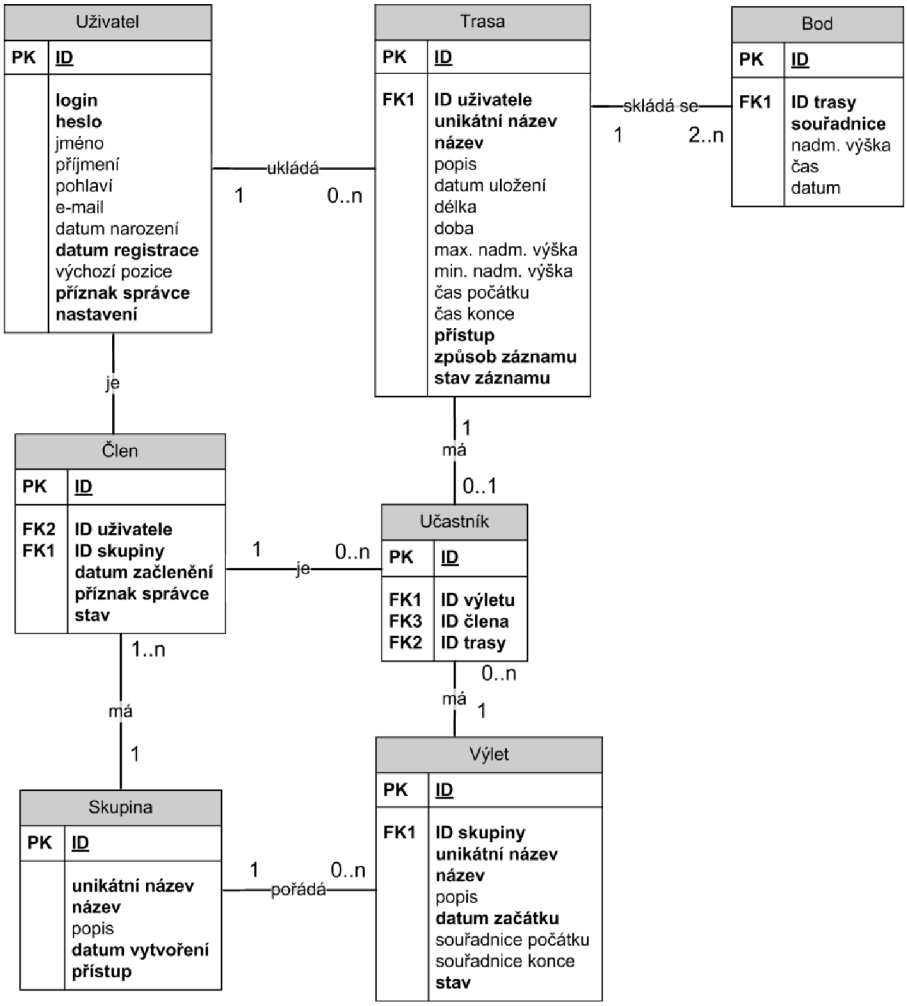
Další důležitou entitou je tabulka *Skupina*. Skupiny umožňují sdružovat uživatele do logických celků a v rámci těchto celků organizovat skupinové výlety. Skupiny může vytvářet libovolný uživatel, který se zároveň stává jejím správcem. Ten musí při jejím založení uvést *název* skupiny a zároveň určit, jakým způsobem se budou moci k této nové skupině připojit jiní uživatelé, což upravuje příznak

[1] Zde je myšleno především použití samostatné vrstvy pro texty při grafickém návrhu.

přístup. Skupiny jsou takto rozděleny na veřejné (připojit se může libovolný přihlášený uživatel) a soukromé (členy vybírá správce skupiny).

Se skupinami dále souvisí tři další entity. Entita *Člen* uchovává seznam uživatelů, kteří mají s danou skupinou nějaký vztah. Návrh počítá se třemi možnými vztahy – uživatel je členem skupiny, uživatel je pozvaný do skupiny a uživatel je vyloučen ze skupiny.

Entity *Výlet* a *Účastník* jsou také součástí skupin. Každý správce může v rámci spravované skupiny naplánovat pro ostatní členy výlety. U každého výletu je nutné vyplnit *název* a *datum začátku* výletu. Výlet se může nacházet v jednom ze tří stavů, který určuje atribut *stav*. Pokud je výlet naplánován, členové skupiny se k výletu mohou připojit, čímž se z nich automaticky stávají účastníci výletu. Ve chvíli, kdy správce přepne výlet do stavu *aktivní*, nemohou se noví účastníci připojit a stávající odpojit. Pokud správce skupiny ukončí aktivní výlet, ten se automaticky nastaví do stavu *ukončený*. Pokud se výlet nachází ve stavu *aktivní*, může každý účastník začít nahrávat do databáze trasu, která bude s tímto výletem propojena.



Obrázek 3-2: Entity-relationship diagram

3.3 Aplikace pro mobilní telefony

Aplikace pro mobilní telefony bude v celém systému představovat klienta. Celá aplikace bude naprogramována v jazyce Java ME. Podmínkou funkčnosti je, aby zařízení, na kterém má být

aplikace nainstalována, podporovalo konfiguraci CLDC, profil MIDP verze 1.0 a zároveň je nutné, aby zařízení disponovalo podporou technologie Bluetooth pro připojení externího GPS přijímače.

Celý návrh mobilní aplikace znázorňuje schéma 3-3. Po spuštění je uživatel požádán o přihlašovací údaje, které zadal při registraci. Ty jsou buď uloženy v perzistentní paměti zařízení z předchozího spuštění aplikace, nebo je musí uživatel zadat ručně. Následně jsou tyto přihlašovací údaje ověřeny na serveru. Po správném přihlášení je uživateli zobrazena hlavní obrazovka.

V tuto chvíli by měl uživatel mít možnost spárovat svůj mobilní telefon s GPS modulem skrze Bluetooth. K tomu je určena obrazovka GPS. Na uživatelův pokyn začne aplikace vyhledávat všechna dostupná Bluetooth zařízení v okolí. Pokud se nějaká zařízení nacházejí v dosahu, jsou uživateli zobrazena v seznamu. V momentě, kdy uživatel vybere správné zařízení a dojde k navázání spojení, vytvoří aplikace nové vlákno, jehož účelem je sbírat veškeré věty, které poskytne GPS zařízení. Toto vlákno po přijetí nové věty zavolá funkci GPS modulu, která tuto větu převezme a rozparsuje na jednotlivé údaje o poloze, směru, nadmořské výšce, atd. Tyto údaje jsou nakonec uloženy uvnitř modulu a mohou být na požádání poskytnuty jiným modulům.

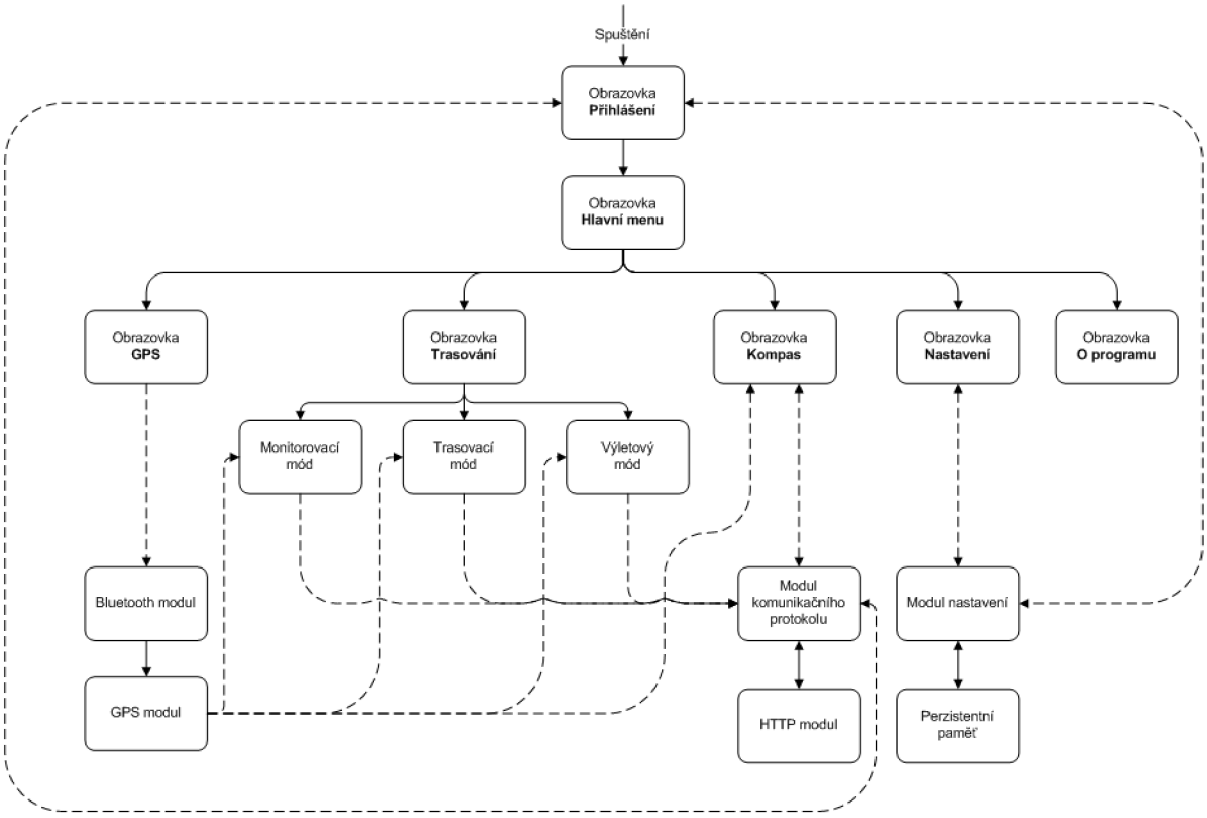
Po připojení GPS zařízení je uživateli umožněno využívat služeb, které ke svojí práci vyžadují údaj o poloze. První z nich je kompas. Kompas by měl uživatele graficky informovat o tom, kterým směrem se právě pohybuje a zároveň na vyžádání zobrazit, v jakém směru se nacházejí jeho přátelé, spolu s jejich přímou vzdáleností od bodu, ve kterém se právě uživatel nachází. Informace o poloze vybraných přátel musí poskytovat server, proto je nutné se v pravidelných intervalech serveru dotazovat.

Druhou funkcí, která vyžaduje spojení s GPS zařízením, je trasování. Během trasování se v pravidelném intervalu, který si může uživatel nastavit, zasílají na server údaje o aktuální poloze, nadmořské výšce a času. Server tyto údaje zpracuje a podle vybraného trasovacího módu zaznamená.

Trasovací módy jsou celkem tři. První mód – *monitorovací* – oznamuje serveru, že má ukládat jen nově přijatý bod a dříve přijaté zahazovat. Pokud by si tedy kdokoliv chtěl zobrazit na mapě webové aplikace polohu uživatele, který právě používá monitorovací mód, viděl by pouze jeho aktuální polohu, nikoliv předchozí body.

Trasovací mód je druhý v pořadí a rozšiřuje monitorovací mód o ukládání všech bodů, které uživatel zašle serveru. Tímto způsobem lze na server nahrávat celou trasu, kterou pak může každý oprávněný uživatel vidět na mapě webové aplikace.

Posledním módem je *výletový mód*. V tomto módu ukládá server přijaté údaje o poloze do databáze stejně jako u trasovacího módu, ovšem s tím rozdílem, že v tomto módu se uživatel účastní výletu naplánovaného v rámci nějaké skupiny. To znamená, že nahrávaná trasa je spjata s nějakým konkrétním výletem.



Obrázek 3-3: Návrh struktury aplikace pro mobilní telefony

U všech módů lze souběžně zobrazit kompas a v něm i směr a vzdálenost k vybraným přátelům. Pouze u výletního módu lze namísto přátel sledovat ostatní účastníky výletu.

3.4 Komunikační protokol

Aby mobilní aplikace mohla komunikovat se serverem, bylo nutné navrhnout a vytvořit komunikační protokol. Vzhledem k tomu, že některé starší typy telefonů podporují pouze profil MIDP verze 1.0 (viz. 4.12.2), který ukládá výrobcům povinnost naimplementovat pouze síťový protokol HTTP, rozhodl jsem se pro využití služeb právě tohoto protokolu. Protokol HTTP je bezestavový a veškerá komunikace probíhá synchronně, což umožňuje použití pouze schématu dotaz - odpověď. Klient tedy zasílá serveru požadavky (např. HTTP GET nebo HTTP POST) na určitý zdroj a server odpoví zasláním příslušného zdroje v těle HTTP odpovědi. Nevýhodou tohoto principu je fakt, že server není schopen za normálních podmínek zaslat klientovi zprávu mimo rámec běžné komunikace dotaz - odpověď.

Druhou nevýhodou tohoto protokolu je, že nevytváří kontext komunikace mezi klientem a serverem. Server neukládá historii předchozích stavů, pouze předá odpověď a nestará se o to, co se událo v předchozím dotazu. Tento problém se řeší použitím tzv. cookies na straně klienta. Cookies jsou v podstatě malé kousky informací, které se uchovávají na klientském zařízení a mohou být při dotazu na server připojeny k hlavičce HTTP protokolu. Aplikace, která běží na serveru a zpracovává HTTP požadavky, může tyto informace následně z HTTP hlavičky vyextrahovat a použít k identifikaci klienta a udržovat tak stav komunikace mezi serverem a klientem.

Při návrhu protokolu jsem se rozhodl využít schopností jazyka PHP k napsání skriptu, který bude volán klientskou aplikací s určitými parametry, které budou obsahovat především číslo zprávy

a další údaje, a jeho výstupem bude konkrétní odpověď na tuto zprávu. Při návrhu jsem použil následující tvar zpráv:

```
cislo;parametr_1;parametr_2;...;parametr_n;
```

kde `cislo` jednoznačně identifikuje zprávu a `parametr_1` až `parametr_n` jsou informace, které vyžaduje konkrétní funkce definovaná zprávou. U odpovědí jsem ještě rozšířil tento tvar o příznak o výsledku vykonané akce (parametr `stav`):

```
cislo;stav;parametr_1;parametr_2;...;parametr_n;
```

Jednotlivé parametry zprávy jsou od sebe odděleny středníkem (;), aby bylo možné tyto zprávy jednoduše zpracovávat.

Pro lepší pochopení uvádím příklad dotazu a odpovědi na zprávu, která zajišťuje ověření identity klienta na serveru a jeho přihlášení. Tvar dotazu je následující:

```
1;login;heslo;
```

kde číslo `1` oznamuje serveru, že uživatel se chce přihlásit. Parametr `login` obsahuje přihlašovací jméno uživatele a parametr `heslo` samozřejmě přístupové heslo uživatele. Odpověď od serveru na tento dotaz je následující:

```
1;stav;
```

kde číslo `1` naznačuje klientovi, že se jedná o výsledek zprávy zajišťující přihlášení a parametr `stav` obsahuje samotný výsledek celé operace. V tomto konkrétním případě může parametr `stav` nabývat pouze dvou hodnot a to buď "`OK`", pokud ověření a přihlášení proběhlo v pořádku, nebo "`ERR`" v případě chyby. Celý proces může tedy vypadat například takto:

```
dotaz od klienta:   1;alice;4l1c3;
odpověď od serveru: 1;OK;
```

V podobném duchu pracují i ostatní navržené zprávy. Jejich kompletní seznam a detailní popis je uveden v příloze B.

4 Implementace

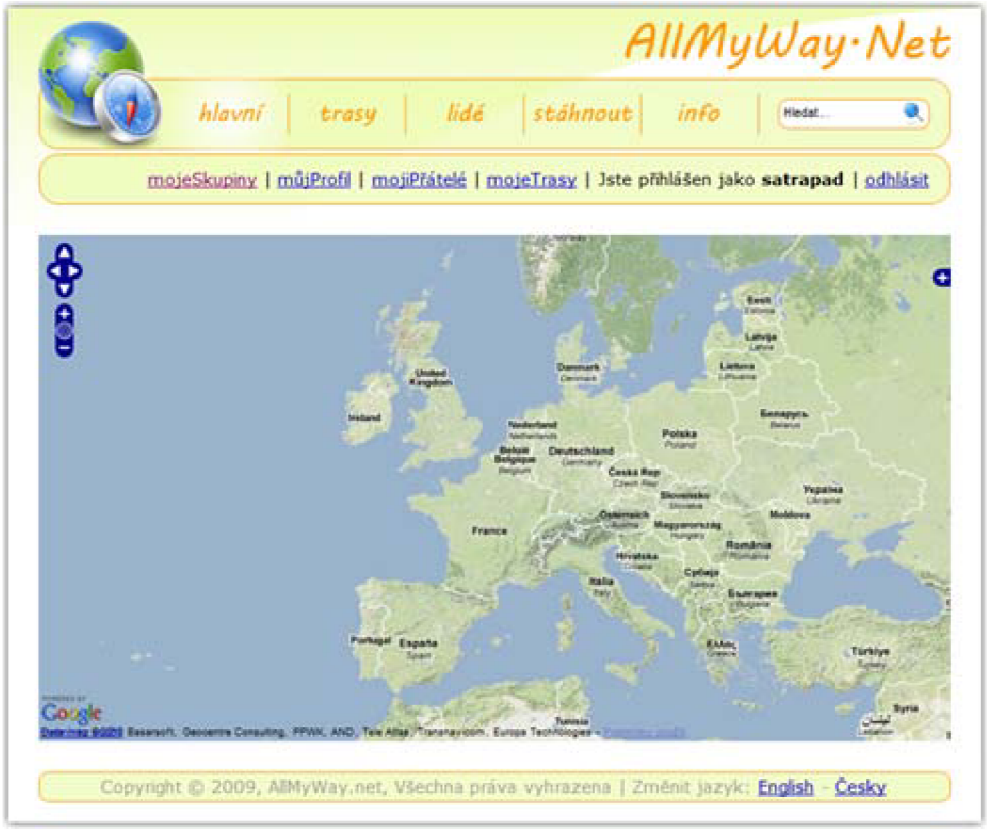
Samotná implementace celého systému byla zřejmě nejrozsáhlejší částí této bakalářské práce. Pokud jde o rozložení, zabrala implementace serverové části spolu s webovou aplikací přibližně 75% veškerého času. Zbylých 25% bylo věnováno implementaci aplikace pro mobilní telefony.

4.1 Grafické rozhraní webové aplikace

Prvním krokem bylo vytvoření návrhu uživatelského rozhraní. Rozhodl jsem se pro použití jednoduchých prvků a vzhledem k faktu, že aplikace měla sloužit především k zobrazování map, rozhodl jsem se umístit hlavní menu spolu s uživatelským panelem do horní - nikoliv levé či pravé - části stránky tak, aby mapové podklady měly co nejvíce prostoru.

V dalším kroku byl celý grafický návrh rozřezán na jednotlivé grafické prvky (tlačítka, lišty, apod.). Nakonec bylo nutné všechny tyto prvky pospojovat zpět do původní podoby za pomoci kombinace kaskádových stylů a HTML.

Výsledný design je vidět na obrázku 4-1.



Obrázek 4-1: Grafické rozhraní webové aplikace s podkladovou mapou

4.2 Volně přístupná uživatelská část

Jak bylo uvedeno v kapitole 5.1, měly být některé části systému přístupné i neregistrovaným uživatelům. Jednalo se především o jednorázové nahrávání tras v podobě GPX souborů, jejich zobrazení na mapě a zpřístupnění pomocí dočasného odkazu.

Během implementace jsem se rozhodl, že trasa, kterou na server nahraje neregistrovaný uživatel, nebude ukládána do databáze, jak jsem to zamýšlel implementovat u registrovaných uživatelů, ale celý GPX soubor se fyzicky uloží na disk serveru, kde by zároveň byl uložen soubor XML, který by uchovával seznam všech takto nahraných tras. V tomto seznamu by se uchovávaly informace o jedinečných názvech nahraných souborů, datum jejich vložení do systému, počet zobrazení a datum posledního zobrazení. Tyto informace by později měly být využity pro automatické mazání těchto dlouho nenavštívených jednorázových tras. Zároveň tento postup znevýhodňuje neregistrované uživatele oproti registrovaným - pokud uživatel zapomene adresu odkazu na zobrazení svojí trasy, musí uživatel nahrát trasu znovu.

Při zobrazování jednorázových tras jsem využil knihovnu OpenLayers spolu s mapovými podklady od společnosti Google. Google poskytuje v podstatě 4 základní mapové vrstvy (terénní, hybridní, satelitní, klasické). OpenLayers bez větších problémů zvládne import všech 4 vrstev. Do poslední vrstvy OpenLayers se vkládá samotný vektor trasy, tvořený jednotlivými liniemi mezi dvojicemi bodů.

Během implementace vektorové vrstvy jsem narazil na problém s rozdílnou projekcí Google map. Po vykreslení celé trasy na mapový podklad byla tato trasa značně odchýlena od své skutečné pozice, ačkoliv tvar trasy odpovídal skutečnosti. Řešením bylo přetransformování souřadnic trasy (EPSG: 4326) na souřadnice systému, který používá Google u svých map (EPSG: 90006).

Druhou zobrazovanou částí měly být statistiky o celé trase a zároveň graf, zobrazující nadmořskou výšku v jednotlivých bodech trasy. Pro zobrazení grafu jsem zpočátku zamýšlel použít knihovnu pro tvorbu grafů také od společnosti Google - *Google Chart API*. Bohužel, po prvních pokusech jsem došel k názoru, že toto rozhranní je vhodné spíše pro zobrazování jednoduchých grafů o malém datovém rozsahu. Pro reálné trasy, které mohou často obsahovat i desítky tisíc údajů o nadmořské výšce, se tato - jinak zajímavá - technologie jevila jako naprosto nepoužitelná. Nakonec jsem se rozhodl použít knihovnu *JSCharts*, která je napsána v jazyce JavaScript a nabízí rozsáhlou základnu funkcí pro vytváření a práci s grafy. Tato knihovna neměla při zobrazování nadmořské výšky u rozsáhlé trasy nejmenší problém.

Jako poslední jsem měl v plánu zobrazit uživateli statistiku o trase. Za nejdůležitější jsem považoval údaj o délce celé trasy, celkovou dobu trvání a průměrnou rychlost. Všechny tyto tři údaje bylo nutné vypočítat. Získat z GPX souboru informaci o celkovém čase nebyl nijak velký problém. Stačilo pouze odečíst čas uložení posledního bodu, ten převézt na vteřiny a od něj odečíst čas zaznamenání prvního bodu také převedený na vteřiny. Mnohem větší problém byl se získáním celkové délky trasy. Standardní GPX soubor totiž sám o sobě neposkytuje žádný údaj o celkové vzdálenosti, či vzdálenosti úseků mezi jednotlivými body a záleží spíše na dobré vůli výrobce zařízení či aplikace, generující GPX soubor, zdali tuto informaci do GPX souboru nějakým způsobem zahrne. Naštěstí existuje několik matematických vzorců, které dokážou s určitou přesností vypočítat přímou vzdálenost mezi dvěma body se zadanými souřadnicemi. Jelikož se ale trasa při nahrávání neukládá do databáze, ale pouze na disk, není možné využít funkci k výpočtu vzdálenosti trasy, kterou poskytuje nadstavba PostGIS. Bylo tedy nutné naimplementovat jeden ze vzorců přímo v jazyce PHP. Rozhodl jsem se použít vzorec obecně nazývaný jako *Haversine formula*. Po implementaci a následném otestování jsem bohužel narazil na nedostatek jazyka PHP, kterým je rychlost při vyhodnocování matematických výrazu. Pro výpočet celkové vzdálenosti trasy složené z řádově

stovek bodů byla doba výpočtu odhadem několik vteřin, ovšem při vyhodnocování tras s řádově tisíci až desetitisíci body se vyšplhala doba výpočtu až na několik minut. Tento problém jsem se rozhodl nijak neřešit, ale spíše tím opět zvýhodnit registrované uživatele před neregistrovanými. Po výpočtu celkové vzdálenosti trasy již nebyl problém ze známého vzorce vypočítat průměrnou rychlost.

Kromě jednorázového nahrávání tras mohou neregistrovaní uživatelé prohlížet trasy registrovaných uživatelů, ovšem pouze za předpokladu, že registrovaní uživatelé svoje trasy těmto uživatelům zpřístupní (trasy označené jako veřejné). Neregistrovaní uživatelé mohou také prohlížet profily registrovaných uživatelů, či procházet veřejné skupiny.

4.3 Uživatelská část podmíněná registrací

Uživatelé, kteří se rozhodnou zaregistrovat, získají tímto krokem značné výhody. Oproti neregistrovaným uživatelům jsou nahrané GPX soubory ukládány do databáze PostgreSQL, která spolu s nadstavbou PostGIS umožňuje provádět nad těmito daty velmi rychlé a rozsáhlé operace. Například výpočet celkové vzdálenosti trasy u registrovaných uživatelů je i u těch nejrozsáhlejších tras otázkou několika málo vteřin.

Další předností registrace je správa všech nahraných tras. Trasy, nahrané registrovaným uživatelem, jsou ukládány do databáze a uživatel s nimi může libovolně manipulovat (mazat, částečně měnit, řídit přístup, apod.)

Registrovaní uživatelé mohou také mezi sebou vytvářet vazby. Především se mohou přátelit. Pokud je mezi dvěma uživateli přátelská vazba, může jeden uživatel prohlížet trasy viditelné pouze pro přátele druhého uživatele a stejně tak to platí i opačně. S tím souvisí určování přístupu k uživatelovým trasám. U každé trasy může její vlastník určit, zdali je tato trasa soukromá (může ji prohlížet pouze vlastník) nebo sdílená (mohou ji prohlížet pouze vlastníkovi přátelé) či veřejná (k trase má přistup každý registrovaný i neregistrovaný uživatel).

Poslední výsadou registrovaných uživatelů je možnost sdružování do skupin.

4.4 Aplikace pro mobilní telefony

Jak už bylo řečeno v kapitole 5.3, měl být celý systém doplněn o aplikaci pro mobilní telefony založenou na platformě Java ME. V této části bych rád ve zkratce popsal řešení implementace této aplikace.

Základem celé aplikace je třída `AMW`, která dědí třídu `MIDlet`, jenž vyžaduje implementaci metod `startApp()`, `destroyApp()` a `pauseApp()`, jak bylo popsáno v části 4.12.3. Metoda `startApp()` nejprve inicializuje modul uživatelského nastavení `Settings`, modul komunikačního protokolu `Comm` a s ním související modul implementující protokol HTTP. Poté je na displej, pomocí metody `display()`, zobrazena přihlašovací obrazovka, což je třída rozšiřující grafickou třídu `Form`.

Třída `Form` vyžaduje implementaci rozhranní `CommandListener`, jejímž úkolem je detekovat stisknutá tlačítka. Při stisku klávesy `CommandListener` automaticky volá metodu `commandAction()`, uvnitř které může programátor každému tlačítku definovat konkrétní akci.

Pokud uživatel spustil mobilní aplikaci poprvé, je vyzván k zadání přihlašovacích údajů. V případě, že uživatel aplikaci již použil, doplní aplikace přihlašovací údaje automaticky a to díky modulu `Settings` a jeho metodám `getUserLogin()` a `getUserPassword()`, které tyto informace získají z perzistentní paměti zařízení.

Do perzistentní paměti si aplikace ukládá nejen přihlašovací údaje uživatele, ale zároveň také některé informace ohledně konfigurace GPS či nastavení aplikace, které si definoval uživatel (např. časový interval odesílání dat na server při trasování). Jak bylo zmíněno o pár řádků výše, toto všechno zajišťuje modul `Settings`.

Po zadání jména a hesla jsou tyto údaje ověřeny na serveru. O komunikaci se serverem se stará modul `Comm`. Tento modul obecně zastřešuje všechny zprávy definované v komunikačním protokolu a stará se především o sestavování dotazů a následné zpracování přijaté odpovědi. Každá zpráva je definována samostatnou funkcí, kterou mohou volat ostatní moduly. Modulu `Comm` je podřízen modul `Http`, který přebírá vygenerované dotazy, vytváří HTTP požadavky typu GET a přijímá odpovědi od serveru a ty následně předává zpět modulu `Comm`.

V případě úspěšného přihlášení je uživateli zobrazena obrazovka definovaná třídou `MainScreen`, jenž by se dala zjednodušeně popsat jako rozcestník, který dává uživateli na výběr několik akcí.

V první řadě je to především připojení k GPS zařízení pomocí Bluetooth. Bez tohoto kroku dojde při spuštění jiných služeb k zobrazení chybové hlášky, která informuje uživatele o nutnosti připojení GPS. Připojení k GPS začíná spuštěním modulu `Bluetooth` a vyhledáním všech dostupných zařízení v okolí metodou `startInquiry()`. Po úspěšném prohledání je zavolána metoda `inquiryComplete()`. Tato metoda vytvoří seznam jmen a adres všech dostupných zařízení a ten následně zobrazí uživateli, aby si mohl vybrat, se kterým zařízením chce komunikovat. Pro vybrané zařízení je spuštěna metoda `searchServices()`. Tím aplikace získá přehled o tom, jaké služby vybrané zařízení nabízí. Poté je zavolána metoda `servicesDiscovered()`, jejímž jedním parametrem je seznam dostupných služeb. Tato metoda postupně projde všechny nabízené služby, a pokud najde požadovanou službu (v případě GPS zařízení jde o službu sériového portu), dojde k navázání spojení s daným zařízením. Po ukončení hledání služeb se volá funkce `serviceSearchCompleted()`. Jejím účelem je informovat o výsledku navázání spojení.

Pokud dojde ke korektnímu spárování telefonu s Bluetooth zařízením, dojde v aplikaci k vytvoření samostatného vlákna, definovaného v modulu `COMMReader`. `COMMReader` v nekonečném cyklu přijímá data, které vysílá GPS zařízení. Ve chvíli, kdy dojde k přijmutí znaku konce řádky `\n` a je tak přijata jedna věta protokolu NMEA, zavolá vlákno metodu `parseNMEA()` modulu `GPS` a jako parametr předá takto získanou větu. Modul `GPS` tuto větu rozparsuje na jednotlivé parametry a podle typu věty uloží potřebná data (např. zeměpisnou délku a šířku, nadm. výšku, počet dostupných satelitů, apod.) do své lokální paměti, odkud mohou být poskytnuta dalším modulům pomocí funkcí, které tento modul také definuje. Tím také dojde k odemčení služeb, vyžadujících ke své práci data z GPS.

K těmto službám patří všechny tři trasovací módy uvedené v návrhu 5.3. Před spuštěním trasování, nabídne uživateli třída implementující formulář `TrackingScreen` výběr jednoho z těchto tří módů.

Nejjednodušším z těchto tří je monitorovací mód. V tomto módu dojde k vytvoření časovače, který - v intervalu nastaveném uživatelem - pouze zasílá serveru aktuální polohu. Aktuální polohu poskytují funkce `getLatitude()` a `getLongitude()` implementované v modulu `GPS` a o zaslání zprávy s aktuální polohou na server se stará funkce `updatePosition()` v modulu `Comm`.

Trasovací mód pracuje na stejném principu jako mód předchozí, ovšem ještě před zahájením trasování musí uživatel zadat název, popis a přístupová práva k této budoucí trase. Poté modul `Comm` přes metodu `startNewTrack()` zašle serveru zprávu, že uživatel se chystá začít zaznamenávat novou trasu a předá mu údaje o této nové trase. Pokud server tento požadavek potvrdí, nastaví

aplikace časovač, který v pravidelném intervalu bude předávat serveru nové body. To umožňuje metoda `newPoint()` modulu `Comm`. Tato metoda předává serveru krom polohy i údaj o nadmořské výšce získané z modulu `GPS` funkcí `getAltitude()`. V momentě, kdy se uživatel rozhodne zaznamenávání trasy ukončit, je vhodné tuto skutečnost oznámit serveru, což umožňuje metoda `finishNewTrack()` v modulu `Comm`.

Princip výletového módu je úplně stejný jako u trasovacího módu, pouze uživatel namísto počátečního zadávání údajů o trase musí vybrat skupinu a výlet, ke kterému chce tuto novou trasu asociovat. Nejprve je tedy nutné požádat server o seznam skupin, ve kterých je uživatel členem (metoda `Comm.getGroupList()`). Následně musí server poskytnout k vybrané skupině seznam aktivních výletů (metoda `Comm.getTripList()`). Nakonec aplikace vyzve server, aby následující trasu, která se bude zaznamenávat, přiřadil k vybranému výletu. Od této chvíle je postup nahrávání trasy totožný s trasovacím módem.

Posledním důležitým prvkem mobilní aplikace je modul `Compass`. Primárním účelem této služby je poskytnout uživateli informaci o směru jeho pohybu. Modul `Compass` dědí z třídy `Canvas`, která umožňuje vykreslovat na displej zařízení různé geometrické objekty. O to se stará metoda `paint()` třídy `Canvas`. Uvnitř této metody definuje programátor veškeré objekty, které mají být zobrazeny, a stejně tak i jejich chování. Grafická podoba kompasu je složena ze šesti soustředných kružnic, kde průměr největší kružnice je odvozen od šířky displeje daného mobilního telefonu. Pro snazší orientaci je kompas rozdělen do 12 segmentů po 30° a vrcholy kompasové růžice jsou označeny počátečními písmeny anglických názvů pro světové strany (N = north - sever, E = east - východ, S = south - jih, W = west - západ). Celý kompas se natáčí podle směru, který udává GPS zařízení. Aktuální kurs poskytuje metoda `GPS.getCourse()`.

Druhým účelem kompasu je zobrazení směru a vzdálenosti k jiným uživatelům (resp. přátelům), kteří právě používají mobilní aplikaci. Pokud uživatel chce tuto funkci použít, musí nejprve serveru oznámit, které přátele si přeje takto sledovat. Je tedy nutné požádat server o seznam všech přátel, což umožňuje metoda `Comm.getFriendList()`. Seznam je uživateli předložen, ten si z něj může vybrat konkrétní přátele a serveru je tento nový seznam sledovaných přátel předán přes metodu `Comm.setSelectedFriends()`. Server si tento seznam uloží a při přijetí požadavku na směr a vzdálenost sledovaných přátel posílá údaje pouze těch přátel, kteří jsou v tomto seznamu uvedeni. To je výhodné zejména z toho důvodu, že klient nemusí s každým požadavkem neustále serveru předávat seznam přátel a lze tím značně snížit velikost přenášených dat. Získání informací o směru a vzdálenosti přátel zprostředkovává metoda `Comm.getFriendsAzimDist()`. Tyto informace jsou poté zobrazeny na kompasu.

4.5 Komunikační protokol

Samotná implementace celého protokolu navrženého v kapitole 5.4 probíhala v podstatě souběžně na straně serveru a na straně klientské aplikace.

Jak už bylo řečeno v kapitole 5.4, jako serverová aplikace byl použit skript napsaný v jazyce PHP, který se spouštěl při každém zavolání ze strany klienta. Volání tohoto skriptu vypadá následovně:

```
mobile.php?msg=message
```

Je tedy nutné volat tento skript metodou GET a samotnou zprávu protokolu `message` předat jako parametr `msg`. Konkrétní příklad volání skriptu:

```
mobile.php?msg=1;alice;4l1c3;
```

Ještě před vyhodnocením se celá zpráva rozloží na jednotlivé parametry a následně se podle identifikačního čísla zprávy zavolá příslušná funkce i s případnými parametry. Po vykonání příslušné akce se na výstup odešle odpověď s výsledkem.

Na straně klienta byla implementace protokolu o něco složitější a to z toho důvodu, že platforma Java ME neumí zpracovávat cookies obsažené v HTTP hlavičce. Zároveň bylo nutné přihlédnout k faktu, že každý dotaz se bude serveru předávat v rámci URI, a proto je nutné všechny neobvyklé znaky (jako např. mezery či diakritiku) zakódovat.

4.6 Popis vybraných problémů

V této části je představena implementace některých problémů, které bylo nutné během implementace vyřešit.

4.6.1 Automatické obnovování polohy uživatele na mapě

Při sledování polohy uživatele na mapě vznikl problém, kdy uživatel sledující tuto polohu musel pro načtení nových informací obnovit v prohlížeči celou stránku. Pro větší komfort jsem do systému zařadil metodu, která se automaticky po určité době dotáže serveru, zda nedošlo k nějaké změně. Jelikož zobrazení polohy na mapě zajišťuje JavaScriptová knihovna OpenLayers, použil jsem automatické načítání technikou AJAX.

Důležitou částí této metody je časovač. V jazyce JavaScript se jako časovač používá funkce `setTimeout(funkce, cas)`, která po uplynutí intervalu `cas` zavolá funkci `funkce`. Tím byl vyřešen problém s pravidelným voláním funkce na obnovení dat v mapě. Dále bylo nutné rozlišit, zda uživatel nahrává přes mobilní aplikaci celou trasu nebo pouze aktuální polohu. V případě trasy bylo nutné získat ze serveru všechny nové – ještě nezobrazené – body, kdežto v případě polohy stačila pouze jednoduchá aktualizace bodu. Proto jsem na straně serveru vytvořil dva nezávislé PHP skripty, kde jeden vrací pouze aktuální polohu vybraného uživatele a druhý vrátí seznam všech nových bodů od bodu zadaného. Aby bylo možné tyto skripty volat, musel být na straně prohlížeče vytvořen objekt *HttpXMLRequest*, který by volal jeden z těchto skriptů i s příslušnými parametry. Po přijetí odpovědi od serveru s novými údaji se v případě pouhé aktualizace polohy pouze přesunul *marker* ukazující polohu na nové místo na mapě. U aktualizace trasy bylo nutné vytvořit z nově přijatých bodů vektor a ten následně „dolepit“ k již existující linii a zároveň přesunout marker, značící poslední bod na trase.

4.6.2 Výpočet délky trasy uložené v databázi

V průběhu implementace jsem narazil na problém, kdy bylo nutné vypočítat celkovou délku trasy uležené v databázi. Nejrychlejší se zdálo být využití funkce *ST_Lenght*, kterou poskytuje PostGIS. Ovšem výsledný dotaz je vcelku netriviální, proto ho zde uvádím jako příklad práce s databází PostgreSQL a rozšířením PostGIS:

```
SELECT ST_Length(ST_Transform(line, 26986)) FROM (SELECT
ST_MakeLine(pts.point_loc) AS line FROM (SELECT point_loc FROM
```

```
amw_points WHERE point_track_id = :track_id ORDER BY point_id) AS
pts) AS distance
```

Celý dotaz začíná ziskem množiny bodů, která tvoří celou trasu, uspořádaných podle pořadí uložení do databáze. Takto získaná množina bodů se následně pomocí funkce *ST_MakeLine* převede na linii. Poté je nutné přetransformovat souřadnice této linie do nějakého souřadného systému, který jako jednotky používá metry. Tento krok je nutný z důvodu, že funkce ST_Length vrací délku objektu v jednotce souřadného systému, což by v případě systému WGS-84 byly stupně. Výsledkem je tedy délka celé trasy v metrech.

4.6.3 Výpočet směru a vzdálenosti sledovaných uživatelů

Během vývoje aplikace pro mobilní telefony jsem narazil na komplikaci při výpočtu směru a vzdálenosti sledovaných uživatelů od aktuální pozice uživatele. Už při návrhu jsem počítal s tím, že server klientovi předá pouze pozice sledovaných uživatelů a informaci o směru a vzdálenosti od nich si klient dopočítá sám. Bohužel, po bližším seznámení s platformou Java ME, která nabízí pouze minimální podporu pro práci s čísly s desetinnou čárkou, jsem usoudil, že tento postup zřejmě nebude možné realizovat. Nabízelo se sice použití externích knihoven pro práci s desetinnými čísly, které si pro svou práci vytvořili sami vývojáři aplikací, ale po vyzkoušení dvou nejrozšířenějších jsem od tohoto nápadu upustil úplně.[2]

Nezbývalo tedy nic jiného, než výpočtem těchto údajů pověřit server a klientovi předat kromě polohy i tyto údaje. Ačkoliv toto řešení nelze označit za špatné, odporuje to původní myšlence, že server pouze poskytuje nezbytné údaje a další potřebné informace budou dopočítány na straně klienta, kde to nebude mít žádný vliv na celý systém.

[2] První z těchto knihoven (Float library) nenabízela potřebné geometrické funkce. Druhá knihovna (Real-Java) se jevila jako velmi dobrá, bohužel výsledky některých goniometrických funkcí pro velmi malá čísla neodpovídaly skutečným hodnotám. Po následné konzultaci výsledků s autorem této knihovny, kdy na straně autora byly výsledky pro konkrétní čísla v pořádku, jsme se shodli, že chyba je zřejmě někde na straně kompilátoru Javy, ovšem nepodařilo se nám zjistit přesnou příčinu.

5 Testování

Už během implementace bylo nutné systém pravidelně testovat. V první části jsem se zaměřil především na testování databáze. Účelem bylo zjistit, jak bude reagovat databáze v případě nahrávání dat z rozsáhlých GPX souborů (v rozsahu 20 000 - 30 000 zaznamenaných bodů). Souběžně s tím jsem testoval, jak rychle je databáze schopna obsloužit více požadavků na uložení trasy. Výsledky byly velmi potěšující a jediným omezením se zdála být rychlost připojení serveru k síti a tedy přenos souboru z klientského počítače.

Ve druhé fázi jsem se zaměřil na zobrazení uložené trasy na mapě a to především komunikaci mezi PHP skriptem a knihovnou OpenLayers. U tohoto testu záleželo především na výpočetním výkonu klientské stanice a použitém prohlížeči. Většina prohlížečů neměla problém zvládnout správně zobrazit celou trasu v rozmezí několika málo sekund. Se zobrazením trasy souvisel také výpočet statistik a znázornění výškového profilu trasy v grafu. Zde nastal již výše zmiňovaný problém při výpočtu délky celé trasy za použití PHP. Zpočátku vracel PHP interpret na serveru informaci o tom, že skriptu, který se staral o výpočet délky, vypršel maximální čas pro běh. Bylo tedy nutné na straně serveru nastavit větší maximální čas běhu PHP skriptů. Po několika pokusech jsem se rozhodl nastavit tuto hodnotu na neomezeně dlouhou dobu, což ovšem není úplně šťastné řešení a v budoucnu bude nutné tento problém vyřešit jiným způsobem. Pokud byl však výpočet délky trasy ponechán na možnostech PostGISu, byl výsledek předán PHP skriptu během velmi krátké doby.

5.1 Testování mobilní aplikace

Prvním krokem při testování mobilní aplikace bylo ověření správné komunikace se serverem. K tomuto účelu jsem vytvořil jednoduchou funkci, která na server zaslala zprávu, obsahující náhodně vygenerovaný řetězec. Pokud server tento náhodný řetězec přijal, bylo jeho úkolem otočit pořadí znaků v řetězci a odeslat ho zpět klientovi. Tím jsem ověřil, že server i klient spolu správně komunikují.

Následně bylo nutné ověřit správné fungování relací mezi klientem a serverem. Ve chvíli, kdy klient zaslal na server první zprávu, musel server vytvořit pro celou komunikaci unikátní relaci a identifikátor relace zaslat i klientovi v hlavičce HTTP protokolu formou cookies. Po přijetí identifikátoru z cookies klientem, bylo nutné tento identifikátor uložit a následně připojit k HTTP hlavičce každého dalšího dotazu. Tím se dosáhlo stavu, kdy server po přijetí dotazu mohl rozlišit, od kterého klienta tento dotaz přišel a díky ukládání informací o relaci mohl na tento dotaz adekvátně reagovat. Správná funkčnost těchto relací byla naprosto nezbytná pro další služby, které měla mobilní aplikace nabízet. Po několika pokusech s přihlášením uživatele a následných úpravách v implementaci, se nakonec podařilo zajistit, aby server i klient znali identifikátor nově vytvořené relace. Tím bylo vyřešeno rozpoznávání klientů na straně serveru a zároveň přemostěna bezstavová komunikace HTTP protokolu.

5.2 Testování v terénu

Po konečném doladění webové i mobilní aplikace bylo nutné otestovat celý systém v praxi.

Pro uskutečnění prvního testu byly zapotřebí dva uživatelé. První měl za úkol projít předem definovanou trasu a za použití mobilní aplikace v monitorovacím módu každých deset vteřin zaslat

serveru zprávu o své aktuální poloze. Úkolem druhého uživatele bylo, sledovat pohyb prvního na mapě s využitím webové aplikace. Oba uživatelé si před začátkem úkolu dohodli referenční body, kterými musí první uživatel projít, aby bylo možné ověřit správnost zaznamenané polohy, její doručení na server a zobrazení na mapě. Pro zajímavost měl první uživatel zaznamenat přesný čas průchodu všemi referenčními body a podobně druhý uživatel měl určit čas, kdy se podle mapy nacházel první uživatel na těchto bodech. Test proběhl přesně podle očekávání, kdy z pohledu druhého uživatele prošel první uživatel všemi referenčními body. Po vyhodnocení času průchodu jednotlivými body z pohledu obou uživatelů bylo zjištěno, že odchylka mezi skutečným časem průchodu a zobrazením průchodu na mapě je v rozmezí 11 – 15 sekund.

Ve druhém testu byl scénář podobný jako při testu prvním, ovšem s tím rozdílem, že v terénu byly uživatelé dva a jejich úkolem nebylo zaznamenávat pouze aktuální polohu, ale celou trasu s tím, že každý uživatel vycházel z jiného místa a cíl měli oba uživatelé společný. Cílem tohoto testu bylo vyzkoušet, jak se celý systém vyrovná s více uživateli v terénu.

Následující test měl vyzkoušet funkčnost mobilní aplikace v místech velmi slabého příjmu GPS signálu a stejně tak signálu GSM. Během cesty vlakem z Brna do Pardubic jsem mobilní aplikací zaznamenával celou trasu a v místech, kde telefon nebyl schopen se připojit k mobilní síti, jsem si udělal poznámku. Podobně jsem postupoval i v případě, kdy ztrátu signálu hlásilo zařízení GPS. Po dokončení záznamu jsem si celou trasu zobrazil na mapě, postupně ji procházel a díky poznámkám jsem vyvozoval, co se stalo v případě výpadku GPS a GSM signálu. V případě výpadku GSM signálu se aplikace nemohla spojit se serverem a nebylo tedy možné předat serveru nově zaznamenané body, což se na vzhledu trasy projevilo nepřirozeným skokem mezi posledním uloženým bodem před výpadkem a prvním příchozím po výpadku. Tento problém by se dal vyřešit implementací jakéhosi vyrovnávacího bufferu, který by během výpadku GSM signálu ukládal všechny zaznamenané souřadnice a ve chvíli, kdy by se mobilní telefon opět připojil do sítě, by aplikace všechny uložené body najednou zaslala na server. V případě výpadku GPS došlo k situaci, že GPS zařízení sice stále vypočítávalo pozici, ovšem se značnou odchylkou (řádově stovky až tisíce metrů od skutečného místa). Tento nedostatek vyřešila implementace mechanismu, který povolil odeslání nové souřadnice na server pouze tehdy, pokud GPS předalo tuto souřadnici s příznakem 3D fix[3].

Dalším v řadě byl test sdílení polohy mezi uživateli mobilní aplikace. K tomuto testu bylo zapotřebí tří uživatelů, kde každý měl svůj mobilní telefon s GPS přijímačem a na něm nainstalovanou aplikaci. Účelem tohoto testu bylo, aby se tito tři uživatelé v terénu - za pomoci vestavěného kompasu - vzájemně našli. Každý uživatel tedy musel na server pravidelně zasílat svou polohu a zároveň žádat server o směr a vzdálenost ke zbylým dvou uživatelům. Poprvé jsme tento test zrealizovali v zalesněném terénu, kde nebyly téměř žádné orientační body. V tomto terénu se kompas velice osvědčil. Při druhém pokusu, který jsme provedli v zastavěné oblasti, se ovšem ukázalo, že kompas by bylo v další fázi vývoje velmi vhodné doplnit mapou, na které by byly vidět i překážky mezi uživateli. Zároveň se nám při tomto testu podařilo odhalit vcelku nepříjemnou chybu, která způsobovala, že v určitém momentě došlo ke zmizení všech údajů o směru a vzdálenosti sledovaných uživatelů na kompasu všech zúčastněných. Po důkladném přezkoumání všech možností se nám podařilo zjistit příčinu. Šlo o to, že pokud se nějaký uživatel rozhodl na kompasu sledovat směr a vzdálenost k jinému - v té chvíli ještě nepřipojenému - uživateli, označil si server tohoto uživatele jako sledovaného. V momentě, kdy se tento sledovaný uživatel přihlásil do systému, server začal automaticky poskytovat informaci o jeho směru a vzdálenosti uživateli, který tyto informace požadoval. Toto chování by bylo v pořádku, ovšem při implementaci jsem přehlédl fakt, že právě

[3] 3D fix označuje stav, kdy GPS zařízení přijímá signál alespoň ze 4 satelitů a je tak schopno určit přesnou polohu i nadmořskou výšku.

připojený uživatel nemá ještě v databázi žádný záznam o poloze a server tedy vypočítává azimut a vzdálenost z hodnoty, která ještě nebyla uživatelem nastavena. To vyústilo v chybu při výpočtu a informace o této chybě se přenesla na výstup, kde ji převzal klient v domnění, že se jedná o standardní odpověď serveru. Následně při pokusu o rozparsování takto přijaté odpovědi došlo na straně klienta k chybě, což funkce, zabezpečující přijetí seznamu informací o směru a vzdálenosti všech sledovaných uživatelů, vyhodnotila jako nesprávnou odpověď a vrátila kompasu prázdný seznam. Kompas při dalším překreslení informací o uživatelích tedy nevykreslil nic. Jakkoliv to může znít složitě, odstranění této chyby byla otázka několika málo minut a bylo nutné změnit chování serveru tak, aby při poskytování informace o tomto uživateli počkal, dokud nebude znát jeho aktuální polohu.

Smyslem posledního testu bylo vyzkoušení výletů v rámci skupiny. K tomuto úkolu byli opět zapotřebí tři uživatelé. Jeden z nich založil novou skupinu, zaslal zbylým dvěma pozvánku do této skupiny a poté naplánoval výlet, u kterého na mapě vyznačil začátek a konec výletu. Členové skupiny se následně k tomuto výletu mohli připojit. Později v terénu tito uživatelé v mobilní aplikaci nastavili výletový mód, přičemž jim byl nabídnut právě tento nově vytvořený výlet. Během trasování mohli tito účastníci sledovat vzájemné pozice na svých kompasech a zároveň i vzdálenost a směr od počátku i konce výletu.

Paradoxně největší problém při testování bylo sehnat lidi s potřebným vybavením (mobilní telefon s podporou technologie Java ME a externí GPS modul).

Aplikace byla testována na mobilních telefonech Sony Ericsson W705, W800i a LG GU230 a jako GPS zařízení byly použity Evolve iTraxx a Navilock BT-399.

6 Instalace serveru

Pro vývoj a testování bylo nutné zajistit server, který by byl schopen poskytnout veškeré služby a aplikace, které jsem chtěl při vývoji bakalářské práce použít. Bohužel, valná většina webhostingových služeb v dnešní době nabízí k ukládání dat pouze databázi MySQL, což byl, vzhledem k mému záměru použít databázi PostgreSQL s geografickou nadstavbou PostGIS, zásadní problém. Možným východiskem by bylo využít služeb některé ze společností, které poskytují tzv. serverhosting či serverhousing, ale po přihlédnutí k poplatkům jsem se rozhodl pro třetí variantu.

Tou byla konfigurace vlastního testovacího serveru, přesně podle mých požadavků. Nároky na hardware nebyly nijak vysoké, jedinou podmínkou byl provoz diskového pole RAID1, který by snížil riziko ztráty důležitých dat na minimum. Jako operační systém jsem zvolil CentOS ve verzi 5.4. CentOS je volně dostupná linuxová distribuce odvozená od komerční distribuce Red Hat Enterprise Linux od společnosti Red Hat. Z vlastní zkušenosti mám ověřeno, že tento systém je velmi stabilní, snadno spravovatelný a vyžaduje minimální údržbu.

Prvním krokem bylo nastavení služby SSH pro možnost vzdálené správy přes terminál a zároveň využití SFTP pro zabezpečený přenos souborů. Následně bylo třeba nastavit a zprovoznit HTTP server, který poskytuje služba Apache. S tím souvisela i instalace služby pro zprovoznění PHP skriptů. Třetím a zřejmě neobtížnějším krokem byla instalace a spuštění databáze PostgreSQL spolu s geografickou nadstavbou PostGIS, která vyžaduje ke svému běhu instalaci dalších dvou knihoven (Proj4, GEOS). Naštěstí na Internetu existuje několik článků detailně popisujících celý proces instalace.

Po dokončení všech kroků jsem měl k dispozici vlastní server připojený do sítě Internet, který mi umožňoval připojení odkudkoliv přes SSH a který sloužil zároveň jako webový, databázový i datový server. Během práce na této bakalářské práci jsem nezaznamenal jediný problém se serverem a všechny nakonfigurované služby plnily svůj účel podle očekávání.

7 Závěr

Cílem této bakalářské práce, bylo navrhnutí a následná implementace systému pro ukládání a správu tras. Systém byl zároveň rozšířen o mobilní aplikaci, která umožňuje uživatelům nahrávání tras přes mobilní telefon a současně umožnit ostatním uživatelům sledovat na mapě pozici přátel. Stanovené cíle byly – až na několik drobností – splněny.

Celý systém je možné otestovat na adrese http://ptakopysk.dlinkddns.com/, kde je také uveden odkaz na stáhnutí mobilní aplikace. Pro přístup lze použít login `test` a heslo `test`.

7.1 Význam práce

Práce na tomto projektu mi přinesla velmi mnoho nových zkušeností. Za tu nejvýznamnější považuji seznámení s jazykem Java ME, který pro mne byl úplnou novinkou. Vyvíjet aplikace v tomto jazyce je až překvapivě jednoduché i pro člověka, který nemá žádné zkušenosti s Javou obecně. Samotné vývojové prostředí Java ME SDK 3.0 velmi usnadňuje programování a v podstatě nutí programátora nedělat příliš mnoho chyb. Vývoj mobilních aplikací mě obecně velmi zaujal a v budoucnu bych se rád zaměřil především na tuto oblast.

Za další velké obohacení považuji knihovnu OpenLayers. Po předchozích zkušenostech s rozhraním Google Maps API jsem byl velmi mile překvapen rozsáhlostí a možnostmi, které OpenLayers nabízí. V kombinaci s technikou AJAX může mít tato technologie do budoucna velký potenciál.

Poprvé jsem se také seznámil s databází PostgreSQL a geografickou nadstavbou PostGIS. Po počátečních problémech s instalací a použitím obou systémů jsem pochopil, proč tuto technologii používají různé velké geografické systémy (např. projekt OpenStreetMaps). Na základě spojení funkcí, které nabízí PostGIS spolu se spolehlivostí a bezpečností, které poskytuje databázový systém PostgreSQL, lze opravdu vystavět velmi robustní geografický informační systém.

7.2 Budoucnost projektu

V budoucnu bych určitě rád pokračoval v dalším vývoji celého projektu. Z těch nejdůležitějších by to byly asi následující body:

- Rozšíření interakce mezi uživateli, především možnost přidávat komentáře k uživatelům, zasílání zpráv mezi uživateli, apod.
- Vytvoření funkcí pro kategorizaci, třídění a hodnocení tras
- Možnost nahrávání fotografií z tras
- Vylepšení mobilní aplikace, především implementace map, zlepšení grafického rozhranní a možnost využití zabudovaného GPS modulu
- Vytvoření aplikace pro platformu Windows Mobile, případně Android či Symbian

Literatura

[1] ŠMEJKALOVÁ, M. *Srovnání aplikačních prostředí pro vlastní webovou mapovou aplikaci* [online]. c2008, [cit. 2010-04-24]. Dostupné na URL: <http://www.gis.zcu.cz/studium/agi/referaty/2008/Smejkalova_SrovnaniAPI/>.

[2] PGSQL.CZ. *PostgreSQL* [online]. c2010, [rev. 2010-04-11] [cit. 2010-04-24]. Dostupné na URL: <http://www.pgsql.cz/index.php/PostgreSQL>.

[3] PGSQL.CZ. *Přechod z MySQL* [online]. c2010, [rev. 2010-01-31] [cit. 2010-04-24]. Dostupné na URL: < http://www.pgsql.cz/index.php/Přechod_z_MySQL>.

[4] PGSQL.CZ. *Instalace PostgreSQL* [online]. c2010, [rev. 2009-11-12] [cit. 2010-04-24]. Dostupné na URL: <http://www.pgsql.cz/index.php/Instalace_PostgreSQL>.

[5] PGSQL.CZ. *PostGIS pro vývojáře* [online]. c2010, [rev. 2008-02-01] [cit. 2010-04-24]. Dostupné na URL: <http://www.pgsql.cz/index.php/PostGIS_pro_vývojáře>.

[6] FOSTER, D. *GPX: the GPS Exchange Format* [online]. c2004, [cit. 2010-04-25]. Dostupné na URL: <http://www.topografix.com/gpx_for_developers.asp>.

[7] WIKIPEDIA FI MUNI. *Java ME* [online]. c2006, [rev. 2009-05-11] [cit. 2010-04-27]. Dostupné na URL: <http://kore.fi.muni.cz:5080/wiki/index.php/Java_ME>.

[8] TOPLEY, K. *J2ME in a Nutshell: A Desktop Quick Reference*. 1. vyd. Sebastopol: O'Reilly & Associates, Inc., 2002. 478 s. ISBN 0-596-00253-X.

Příloha A

Ukázka souboru GPX

```
<?xml version="1.0" standalone="yes"?>
<gpx>
	<trk>
		<trkseg>
			<trkpt lat="50.0405617" lon="15.8049250">
				<ele>214</ele>
				<time>2009-06-05T15:10:00Z</time>
			</trkpt>
			<trkpt lat="50.0405650" lon="15.8049233">
				<ele>214</ele>
				<time>2009-06-05T15:10:01Z</time>
			</trkpt>
			<trkpt lat="50.0405667" lon="15.8049200">
				<ele>214</ele>
				<time>2009-06-05T15:10:02Z</time>
			</trkpt>
			<trkpt lat="50.0405683" lon="15.8049183">
				<ele>214</ele>
				<time>2009-06-05T15:10:03Z</time>
			</trkpt>
			<trkpt lat="50.0405700" lon="15.8049167">
				<ele>214</ele>
				<time>2009-06-05T15:10:04Z</time>
			</trkpt>
			.
			.
			.
			<trkpt lat="50.0216567" lon="15.7555383">
				<ele>222</ele>
				<time>2009-06-05T15:29:56Z</time>
			</trkpt>
			<trkpt lat="50.0216700" lon="15.7555500">
				<ele>222</ele>
				<time>2009-06-05T15:29:57Z</time>
			</trkpt>
		</trkseg>
	</trk>
</gpx>
```

Příloha B

Zprávy komunikačního protokolu

Číslo zprávy:	**1**	
Popis:	**Přihlášení uživatele**	
Dotaz:	`1;login;passwd;`	
Popis parametrů:	`login`	přihlašovací jméno uživatele
	`passwd`	heslo uživatele
Odpověď:	`1;status;`	
Popis parametrů:	`status`	informace o výsledku operace OK = operace proběhla úspěšně ERR = operace se nezdařila

Číslo zprávy:	**2**	
Popis:	**Vytvoření nové trasy**	
Dotaz:	**`2;name;desc;privacy;`**	
Popis parametrů:	**`name`**	název nové trasy
	`desc`	popis nové trasy
	`privacy`	nastavení soukromí 1 = private 2 = shared 3 = public
Odpověď:	**`3;status;`**	
Popis parametrů:	**`status`**	informace o výsledku operace OK = operace proběhla úspěšně ERR = operace se nezdařila

Číslo zprávy:	**3**	
Popis:	**Přidání nového bodu k vytvořené trase**	
Dotaz:	**`3;latitude;longitude;altitude;date;time;`**	
Popis parametrů:	**`latitude`**	šírka ve tvaru SS.MMMM…
	`longitude`	délka ve tvaru SS.MMMM…
	`altitude`	nadmořská výška v metrech
	`date`	datum ve tvaru YYYY-MM-DD
	`time`	čas ve tvaru HH:MM:SS
Odpověď:	**`3;status;`**	
Popis parametrů:	**`status`**	informace o výsledku operace OK = operace proběhla úspěšně ERR = operace se nezdařila

Číslo zprávy:	**4**	
Popis:	**Ukončení trasy**	
Dotaz:	**`4;`**	
Odpověď:	**`4;status;`**	
Popis parametrů:	**`status`**	informace o výsledku operace OK = operace proběhla úspěšně ERR = operace se nezdařila

Číslo zprávy:	**5**	
Popis:	**Zahájení monitorování pozice**	
Dotaz:	**`5;`**	
Odpověď:	**`5;status;`**	
Popis parametrů:	**`status`**	informace o výsledku operace OK = operace proběhla úspěšně ERR = operace se nezdařila

Číslo zprávy:	**6**	
Popis:	**Aktualizace pozice**	
Dotaz:	**`6;latitude;longitude;`**	
Popis parametrů:	**`latitude`**	šírka ve tvaru SS.MMMM…
	`longitude`	délka ve tvaru SS.MMMM…
Odpověď:	**`6;status;`**	
Popis parametrů:	**`status`**	informace o výsledku operace OK = operace proběhla úspěšně ERR = operace se nezdařila

Číslo zprávy:	**7**	
Popis:	**Ukončení monitorování pozice**	
Dotaz:	**`7;`**	
Odpověď:	**`7;status;`**	
Popis parametrů:	**`status`**	informace o výsledku operace OK = operace proběhla úspěšně ERR = operace se nezdařila

Číslo zprávy:	**8**	
Popis:	**Ping**	
Dotaz:	**`8;`**	
Odpověď:	**`8;status;count;`**	
Popis parametrů:	**`status`**	informace o výsledku operace OK = operace proběhla úspěšně ERR = operace se nezdařila
	`count`	počet úspěšných pingů od přihlášení

Číslo zprávy:	**9**	
Popis:	**Odhlášení uživatele**	
Dotaz:	`9;`	
Odpověď:	`9;status;`	
Popis parametrů:	`status`	informace o výsledku operace OK = operace proběhla úspěšně ERR = operace se nezdařila

Číslo zprávy:	**10**	
Popis:	**Získání seznamu uživatelových skupin**	
Dotaz:	`10;`	
Odpověď:	`10;status;group_count;gid_1;gname_1;…;gid_n;gname_n;`	
Popis parametrů:	`status`	informace o výsledku operace OK = operace proběhla úspěšně ERR = operace se nezdařila
	`group_count`	počet uživatelových skupin
	`gid_n`	identifikační číslo skupiny n
	`gname_n`	název skupiny n

Číslo zprávy:	**11**	
Popis:	**Získání seznamu aktivních výletů vybrané skupiny**	
Dotaz:	`11;group_id;`	
Popis parametrů:	`group_id`	identifikační číslo vybrané skupiny
Odpověď:	`11;status;trip_count;trip_id_1;trip_name_1;…;trip_id_n;trip_name_n`	
Popis parametrů:	`status`	informace o výsledku operace OK = operace proběhla úspěšně ERR = operace se nezdařila
	`trip_count`	počet aktivních výletů ve skupině
	`trip_id_n`	identifikační číslo výletu n
	`trip_name_n`	název výletu n

Číslo zprávy:	12	
Popis:	**Změna údajů o aktuálně nahrávané trase**	
Dotaz:	`12;name;desc;privacy;`	
Popis parametrů:	`name`	nový název trasy
	`desc`	nový popis trasy
	`privacy`	nové nastavení soukromí 1 = private 2 = shared 3 = public
Odpověď:	`12;status;`	
Popis parametrů:	`status`	informace o výsledku operace OK = operace proběhla úspěšně ERR = operace se nezdařila

Číslo zprávy:	15	
Popis:	**Získání seznamu uživatelových přátel**	
Dotaz:	`15;`	
Odpověď:	`15;status;friends_count;friend_login_1;friend_status_1;…;friend_login_n;friend_status_n;`	
Popis parametrů:	`status`	informace o výsledku operace OK = operace proběhla úspěšně ERR = operace se nezdařila
	`friends_count`	počet uživatelových přátel
	`friend_login_n`	login přítele n
	`friend_status_n`	stav přítele n 0 = přítel je offline 1 = přítel je online

Číslo zprávy:	16	
Popis:	**Nastavení seznamu sledovaných přátel**	
Dotaz:	`16;count;login_1;login_2;…;login_n;`	
Popis parametrů:	`count`	počet sledovaných přátel
	`login_n`	login vybraného přítele
Odpověď:	`16;status;`	
Popis parametrů:	`status`	informace o výsledku operace OK = operace proběhla úspěšně ERR = operace se nezdařila

Číslo zprávy:	17	
Popis:	**Získání polohy vybraných přátel, kteří jsou online**	
Dotaz:	`17;`	
Odpověď:	`17;status;count;friend_login_1;friend_lat_1;friend_lon_1;…;friend_login_n;friend_lat_n;friend_lon_n;`	
Popis parametrů:	`status`	informace o výsledku operace OK = operace proběhla úspěšně ERR = operace se nezdařila
	`count`	počet online přátel
	`friend_login_n`	login přítele n
	`friend_lat_n`	zeměpisná šířka přítele n
	`friend_lon_n`	zeměpisná délka přítele n

Číslo zprávy:	**18**	
Popis:	**Získání azimutu a vzdálenosti vybraných přátel, kteří jsou online**	
Dotaz:	`18;latitude;longitude;`	
Popis parametrů:	`latitude`	uživatelova aktuální zeměpisná šířka
	`longitude`	uživatelova aktuální zeměpisná délka
Odpověď:	`18;status;count;friend_login_1;friend_azim_1;friend_dist_1;…;friend_login_n;friend_azim_n;friend_dist_n;`	
Popis parametrů:	`status`	informace o výsledku operace OK = operace proběhla úspěšně ERR = operace se nezdařila
	`count`	počet online přátel
	`friend_login_n`	login přítele n
	`friend_azim_n`	azimut přítele n z pozice uživatele
	`friend_dist_n`	vzdálenost uživatele od přítele n

Číslo zprávy:	**19**	
Popis:	**Získání seznamu účastníků výletu**	
Dotaz:	`19;`	
Odpověď:	`19;status;trippers_count;tripper_login_1;tripper_status_1;…;tripper_login_n;tripper_status_n;`	
Popis parametrů:	`status`	informace o výsledku operace OK = operace proběhla úspěšně ERR = operace se nezdařila
	`trippers_count`	počet účastníků
	`tripper_login_n`	login účastníka n
	`tripper_status_n`	stav účastníka n 0 = účastník je offline 1 = účastník je online

Číslo zprávy:	**20**	
Popis:	**Nastavení seznamu sledovaných účastníků**	
Dotaz:	`20;count;login_1;login_2;…;login_n;`	
Popis parametrů:	`count`	počet sledovaných účastníků
	`login_n`	login vybraného účastníka
Odpověď:	`20;status;`	
Popis parametrů:	`status`	informace o výsledku operace OK = operace proběhla úspěšně ERR = operace se nezdařila

Číslo zprávy:	**21**	
Popis:	**Získání azimutu a vzdálenosti vybraných účastníků, kteří jsou online**	
Dotaz:	`21;latitude;longitude;`	
Popis parametrů:	`latitude`	uživatelova aktuální zeměpisná šířka
	`longitude`	uživatelova aktuální zeměpisná délka
Odpověď:	`21;status;count;tripper_login_1;tripper_azim_1;tripper_dist_1;…;tripper_login_n;tripper_azim_n;tripper_dist_n;`	
Popis parametrů:	`status`	informace o výsledku operace OK = operace proběhla úspěšně ERR = operace se nezdařila
	`count`	počet online účastníků
	`tripper_login_n`	login účastníka n
	`tripper_azim_n`	azimut účastníka n z pozice uživatele
	`tripper_dist_n`	vzdálenost uživatele od účastníka n