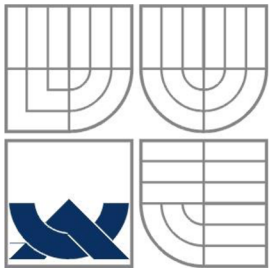
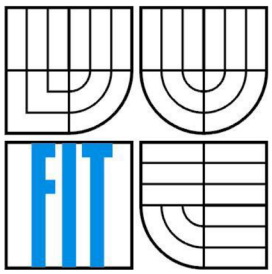


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

WEBOVÝ SYSTÉM PRO SPRÁVU GPS DAT V CLOUDOVÉM PROSTŘEDÍ

WEB-BASED GPS DATA MANAGEMENT SYSTEM IN A CLOUD ENVIRONMENT

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. VILIAM KASALA

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. RADEK BURGET, Ph.D.

BRNO 2015

Abstrakt

Tato práce se zabývá vývojem webového systému pro správu GPS dat v cloudovém prostředí. Práce seznamuje čtenáře s nejpoužívanějšími aplikačními rozhraními pro tvorbu vlastní mapové aplikace. Nabízí také porovnaní cloudových platforem Google App Engine a OpenShift Online. Dále se zabývá návrhem a implementací systému pro platformu OpenShift Online. Systém se zaměřuje na import tras z formátu GPX, správu tras, zobrazení tras na mapových podkladech Mapy.cz, vyhledávaní v trasách, tvorbu výškového profilu a různých statistik.

Abstract

This thesis focuses on development of web-based GPS data management system in a cloud environment. The thesis introduces the most used application interfaces for building custom mapping applications. It offers a comparison of cloud platforms such as Google App Engine and OpenShift Online. It also deals with the design and the implementation of system for OpenShift Online platform. The system focuses on importing tracks from GPX format, managing tracks, viewing tracks on map tiles from Mapy.cz, searching for tracks, creating an elevation profile and various statistics.

Klíčová slova

Webová aplikace, mapová aplikace, GPS, GPX, Google App Engine, Openshift, OpenLayers, Mapy.cz, Java EE, JAX-RS, JPA, AngularJS.

Keywords

Web application, map application, GPS, GPX, Google App Engine, Openshift, OpenLayers, Mapy.cz, Java EE, JAX-RS, JPA, AngularJS.

Citace

Kasala Viliam: Webový systém pro správu GPS dat v cloudovém prostředí, diplomová práce, Brno, FIT VUT v Brně, 2015

Webový systém pro správu GPS dat v cloudovém prostředí

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Ing. Radka Burgeta. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Viliam Kasala
26. května 2015

Poděkování

Rád bych poděkoval Ing. Radkovi Burgetovi za zajímavý námět a bezproblémovou spolupráci při tvorbě této práce.

Obsah

1 Úvod

Cieľom tejto diplomovej práce je preskúmať možnosti online mapových služieb a vývoja systémov v cloudových prostrediach ako Google App Engine a OpenShift. Na základe zistených informácií a systémových požiadaviek sme zvolili platformu na vývoj webového systému spravujúceho GPS dáta získané z GPS zariadení. Systém má slúžiť na zobrazenie prejdených trás na mapových podkladoch služby Mapy.cz, ich štatistík a ich výškových profilov v prehľadných grafoch. Systém má ďalej podporovať import trás zo súborov vo formáte GPX, zdieľanie trás medzi používateľmi, priradenie doplňujúcich údajov k trasám a vyhľadanie trás.

Práca je tematicky rozdelená do 9 kapitol. Kapitola 2 stručne charakterizuje GPS systém a formáty dát získaných z GPS zariadení.

Kapitola 3 zoznamuje čitateľa so základnými pojmami z oblasti kartografie, modelmi zemského povrchu a so súradnicovými systémami.

Kapitola 4 analyzuje možnosti online mapových služieb ako Google Maps a Mapy.cz. Ďalej táto kapitola porovnáva ich aplikačné rozhrania s knižnicou OpenLayers 3.

Kapitola 5 zoznamuje čitateľa s pojmom cloud computing, jeho výhodami, nevýhodami a jeho rozdelením. Ďalej táto kapitola predstavuje cloudové platformy Google App Engine a OpenShift a analyzuje možnosti vývoja so zameraním na programovací jazyk Java.

Kapitola 6 sa venuje analýze existujúcich aplikácií určených na zobrazovanie zaznamenaných trás. Ďalšia časť kapitoly je zameraná na špecifikáciu vyvíjaného systému.

Kapitola 7 zdôvodňuje výber cloudovej platformy a predstavuje návrh architektúry systému a jej jednotlivých častí.

Kapitola 8 popisuje implementáciu jednotlivých častí architektúry systému vrátane použitých technológií.

Kapitola 9 sa zaoberá testovaním a nasadením vyvíjaného systému do zvolenej cloudovej platformy.

Táto diplomová práca nadväzuje na semestrálny projekt, v rámci ktorého bola riešená teoretická časť práce. Kapitoly 2, 4 a 5 využívajú poznatky a zistenia zo semestrálneho projektu.

2 Globálny polohový systém

V tejto kapitole si stručne predstavíme GPS systém a jeho štruktúru. Budeme sa zaoberať tiež formátmi dát získaných z GPS zariadení.

Globálny polohový systém (angl. *Global Positioning System*, GPS) je globálny družicový systém, ktorý umožňuje používateľom presne určiť trojrozmernú polohu (zemepisná dĺžka, zemepisná šírka, nadmorská výška), rýchlosť pohybu a presný čas. Služby poskytované GPS sú dostupné kedykoľvek a kdekoľvek na zemskom povrchu a v jeho priľahlom okolí. Ide o systém, ktorý je schopný pracovať za každého počasia. Jediným obmedzením jeho využívania je priama viditeľnosť oblohy. Preto nie je možné využívať systém GPS napr. v podzemných priestoroch, v budovách alebo pod veľmi hustou vegetáciou [1].

2.1 História GPS systému

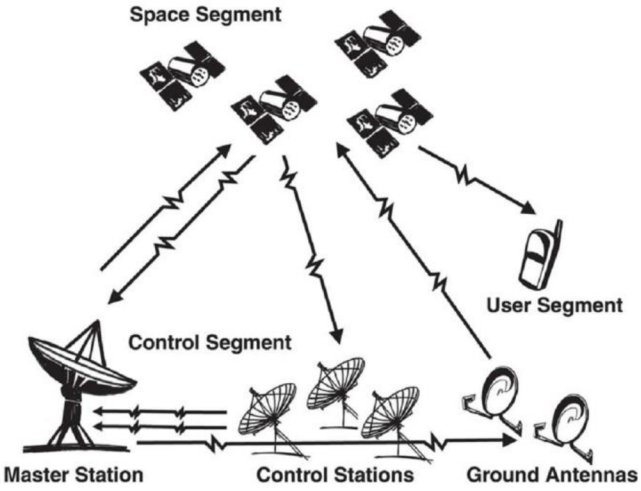
Vývoj systému GPS začal v roku 1973 pod názvom NAVSTAR (angl. *NAVigation with Satellite Timing And Ranging*) na podnet ministerstva obrany USA. Na začiatku bol tento systém určený výhradne pre účely armády USA. Až neskôr v 90. rokoch sa jeho použitie rozšírilo aj do verejnej sféry. Koncom 70. a v priebehu 80. rokov sa postupne v troch fázach začali vypúšťať satelity na obežnú dráhu Zeme. Počiatočný operačný stav (angl. *Initial Operational Capability*) pozostávajúci z dvadsiatich štyroch satelitov bol dosiahnutý 8. decembra 1993 a plný operačný stav (angl. *Full Operational Capability*) 17. júla 1995 [1].

2.2 Štruktúra GPS systému

Systém GPS sa delí na 3 základné segmenty – vesmírny, riadiaci a používateľský. Vesmírny segment (angl. *space*) pozostáva minimálne z dvadsiatich štyroch satelitov, pričom 21 satelitov je navigačných a 3 satelity sú aktívne záložné satelity. Všetky satelity sú umiestnené na šiestich obežných dráhach Zeme, ktoré majú voči Zemi nemennú polohu a nachádzajú sa vo výške 20180 km. Na každej obežnej dráhe sa nachádzajú štyri GPS satelity. Inklinácia (sklon obežnej dráhy) každého satelitu je 55° voči rovníku a doba obehu satelitu okolo Zeme je 11 hodín a 58 minút. Toto usporiadanie garantuje, že na ktoromkoľvek mieste na Zemi sú trvalo dostupné signály z minimálne štyroch satelitov počas celého dňa [1].

V prípade komunikácie satelitov s používateľmi (používateľský segment) je signál prenášaný len jedným smerom a to smerom k používateľom. Každý satelit komunikuje s pozemným riadiacim strediskom (riadiaci segment).

Riadiaci segment (angl. *control*) je zodpovedný za riadenie a bezporuchovú činnosť celého GPS. Tvorí ho hlavná riadiaca stanica (angl. *Master Control Station,* MCS), 4 monitorovacie stanice (angl. *Monitoring Stations*) rozložené po celej Zemi a 3 vysielacie stanice (angl. *Ground Antenna*) určené ku komunikácii so satelitmi. Monitorovacie stanice sú riadené pomocou MCS a ich činnosť spočíva v sledovaní satelitov, prijímaní satelitných údajov a následnom odovzdaní týchto údajov MCS. MSC zo získaných údajov vypočíta presné údaje o obežných dráhach (tzv. efemeridy) a korekcie atómových hodín pre každý satelit zvlášť. Vysielacie stanice minimálne raz za deň rozpošlú tieto údaje jednotlivým satelitom a tie následne vysielajú svoje efemeridy a čas do GPS prijímačov používateľov [1].



Obrázok 2.1 Segmenty globálneho polohového systému [2]

Používateľský segment (angl. *user*) pozostáva z GPS prijímača, jeho používateľov a vyhodnocovacích prístrojov a postupov. Pomocou týchto postupov GPS prijímače urobia predbežné výpočty polohy, rýchlosti a času. Na vyhodnotenie kompletného údaja o polohe tj. zemepisnej dĺžky, zemepisnej šírky a nadmorskej výšky, je potrebný príjem signálov minimálne zo 4 satelitov. 3 satelity sú dostatočné na výpočet zemepisnej šírky a zemepisnej dĺžky. Prijímače sa delia na [1]:

- **jednokanálové -** majú len jeden vstupný kanál, ktorý sa musí pri sledovaní viacerých satelitov prepínať.

- **viackanálové -** majú viac vstupných kanálov, vďaka čomu sú schopné súčasne sledovať viaceré satelity a zvýšiť presnosť výpočtu polohy.

2.3 Reprezentácia a formáty geografických dát získaných z GPS zariadení

V súčasnosti existuje na trhu veľké množstvo odlišných GPS zariadení, ktoré sa odlišujú veľkosťou, výrobcom a poskytovanou funkcionalitou. Zariadenia sú schopné postupne transformovať GPS údaje do geografických dát. Medzi najznámejšie reprezentácie geografických dát v GPS patrí bod záujmu (angl. *Point of Interest,* POI), bod cesty (angl. *Waypoint*), cesta(angl. *Route*) a trasa (angl. *Waypoint*).

POI reprezentuje mimoriadnu lokalitu alebo polohu na mape, ktorá má pre používateľa špecifický význam. Obsahuje v sebe informácie o čase, zemepisnej šírke, zemepisnej dĺžke a v určitých prípadoch aj o nadmorskej výške. POI bývajú často zobrazené na mape ako ikony. Príkladom POI sú napr. hotely, zámky, hrady, reštaurácie, múzea, čerpacie stanice, obchody, obľúbené miesta atď.

Body cesty sú podobné POI, avšak ukladajú aj informáciu o navigácií ako napr. smer jazdy.

Cesta reprezentuje naplánovanú trasu, ktorá sa využíva pre potreby navigácie. Je zložená z usporiadaného zoznamu bodov cesty, pričom jednotlivé body označujú zmenu smeru jazdy. GPS zariadenie pri navigácii danej cesty informuje o smere a vzdialenosti k ďalšiemu bodu cesty.

Celková prejdená trasa je súhrn automaticky zozbieraných údajov (POI) počas pohybu GPS zariadenia. Z týchto záznamov je neskôr možné zistiť dĺžku tejto trasy, rýchlosť a smer pohybu v jednotlivých úsekoch trasy.

Formáty geografických dát sa používajú k výmene týchto dát medzi GPS zariadeniami a softwarovými programami. Najuniverzálnejším a najpoužívanejším formátom je GPX. K ďalším formátom patria KML, TCX alebo CSV. Formáty bývajú medzi sebou často transformovateľné, môže však dôjsť k stratám určitých typov informácií.

TCX je dátový formát pre výmenu trás používaný zariadeniami od spoločnosti Garmin, v ktorom je trasa reprezentovaná ako aktivita a je k nej možné ukladať doplnkové informácie o kalóriách, tepe, cyklistickej a bežeckej kadencii. KML je formát určený k publikácii, distribúcií a vizualizácii geografických dát (bod, línia, plocha atď.). Hlavné využitie má v programoch pre 3D vizualizáciu geografických dát v programe Google Earth.

2.3.1 GPX formát

GPX (angl. *GPS eXchange format*) je textový XML (angl. *eXtensible Markup Language*) formát pre bezstratové uloženie GPS dát alebo pre výmenu GPS dát medzi aplikáciami a webovými službami na internete. Jedná sa o najbežnejší GPS formát pre softwarové aplikácie, GPS prijímače a webové služby. Ide o otvorený štandard, ktorý je dostupný zadarmo bez potreby platenia poplatkov za licencie [3].

GPX je podporovaný veľkými výrobcami GPS zariadení ako napr. *Garmin*, *TomTom* alebo *Magellan*. Taktiež je podporovaný rôznymi webovými službami určenými k zaznamenaniu cyklistických a bežeckých trás alebo tréningu. Takouto službou je napr. *MapMyRun*.

Štruktúra uložených informácií je definovaná pomocou XSD (angl. *XML Schema Definition*) schémy. Najnovšia GPX schéma vo verzii 1.1 bola vydaná 9. augusta 2004. GPX schéma definuje spoločnú množinu dátových značiek pre popis GPS a geografických dát uložených v XML. Okrem štandardizovanej XML schémy je možné definovať aj vlastné privátne objekty a atribúty vďaka podpore rozšírení. Tieto rozšírenia využíva napr. firma *Garmin* pre ukladanie adries, telefónnych čísel, teploty, hĺbky vody a iných údajov v závislosti na type a funkcionalite zariadenia [3, 4].

```
1.  <?xml version="1.0" encoding="UTF-8" standalone="no" ?>
2.  <gpx xmlns="http://www.topografix.com/GPX/1/1" …>
3.      <trk>
4.          <name>Track Showcase</name>
5.          <trkseg>
6.              <trkpt lat="49.5553325" lon="15.9382618">
7.                  <ele>666.46</ele>
8.                  <time>2013-11-10T08:00:24Z</time>
9.              </trkpt>
10.             <trkpt lat="49.5552385" lon="15.9382618">
11.                 <ele>872.93</ele>
12.                 <time>2013-11-10T08:00:26Z</time>
13.             </trkpt>
14.             <trkpt lat="49.545385" lon="15.9382618">
15.                 <ele>875.90</ele>
16.                 <time>2013-11-10T09:00:28Z</time>
17.             </trkpt>
18.         </trkseg>
19.     </trk>
20. </gpx>
```

Kód 2.1 Ukážka záznamu trasy v GPX formáte

Do formátu GPX je možné uložiť všetky reprezentácie geografických dát. V jednom GPX súbore môže byť uložených viacero trás, ciest a samostatných bodov cesty. Cesta sa skladá z viacerých bodov cesty a trasa sa skladá z viacerých segmentov trasy (angl. *track segment*), ktoré

obsahujú body cesty. Ak počas zaznamenávania trasy dôjde k výpadku GPS signálu, vytvorí sa nový segment trasy. Ukážka GPX formátu sa nachádza v kóde 2.1.

Cesta aj trasa obsahujú v GPX schéme rovnaké elementy ako pomenovanie, komentár, textový popis, poradové číslo alebo typ. Pre bod cesty sú v GPX schéme povinné atribúty pre zemepisnú šírku a zemepisnú dĺžku (kartografické súradnice). Šírka a dĺžka je vyjadrená v desatinných stupňoch využívajúcich WGS-84 (angl. *World Geodetic System*) súradnicový systém (viď. kapitola 3.2). Ďalej bod cesty obsahuje elementy ako nadmorská výška, časová značka (angl. *timestamp*), meno, textový popis, komentár, typ, počet družíc použitých k výpočtu polohy, presnosť a mnoho ďalších. Tieto elementy nie sú povinné. Nadmorská výška je vyjadrená v metroch a pre časovú značku sa využíva UTC (angl. *Universal Coordinated Time*), ktorý zodpovedá formátu ISO 8601 pre dátum a čas. Bod záujmu býva v GPX formáte reprezentovaný bodom cesty [5].

3 Kartografia

V tejto kapitole si najskôr predstavíme základné pojmy z oblasti kartografie. Ďalej sa pozrieme na matematické popisy (modely) zemského povrchu a na najznámejšie súradnicové systémy, s ktorými sa môžeme stretnúť pri práci s mapami.

3.1 Základné pojmy

Kartografia – je definovaná v [6] ako veda, technika a schopnosť navrhovať, zhotovovať a využívať mapy a mapám podobné zobrazenia.

Mapa – je definovaná v [6] ako zmenšené, zovšeobecnené a vysvetlené znázornenie objektov a javov na Zemi alebo vo vesmíre, zostrojené v rovine pomocou matematicky definovaných vzťahov.

Mierka mapy – v [6] je definovaná ako pomer zmenšenia mapy vyjadrený číselne v tvare *1:m*. Najčastejšie sa používa v zmysle dĺžkovej mierky mapy.

Zemepisné súradnice – podľa [6] slúžia na udávanie polohy bodov na zemskom povrchu. Bývajú určené dvomi sférickými polárnymi súradnicami – zemepisnou dĺžkou a zemepisnou šírkou.

Zemepisná dĺžka – v [6] je definovaná ako uhol medzi dvomi polrovinami, ktoré vychádzajú zo zemskej osi. Prvá polrovina prechádza bodom zvoleným ako základňa a druhá určovaným bodom. Spojnice bodov rovnakej zemepisnej dĺžky nazývame poludníky. Poludníky sa stretávajú v zemských póloch a sú číslovane od 0° do 180°.

Zemepisná šírka – je podľa [6] definovaná ako uhol medzi rovinou rovníka a normálou v určovanom bode. Rovník je hlavná kružnica, ktorá vznikla ako priesečník zemského povrchu a roviny prechádzajúca stredom Zeme kolmo k zemskej osi. Rovnobežky vzniknú spojením bodov rovnakej zemepisnej šírky. Číslujú sa od 0° do 90° smerom k pólom.

Kartografické zobrazenie (mapová projekcia) – je podľa [7] definované ako proces prevodu krivého zemského povrchu do zobrazovacej rovinnej plochy. Príkladom je zobrazenie bodu zemského povrchu v danom súradnicovom systéme do mapy (pixel na obrazovke).

Merkatorova projekcia – podľa [6] patrí medzi valcové zobrazenia, ktoré zobrazujú povrch sveta na plášť valca, ktorý je ovinutý pozdĺž hlavnej kružnice. Hlavnou nevýhodou tejto projekcie je veľké skreslenie. Túto projekciu používa väčšina dnešných online mapových služieb na zobrazovanie podkladových máp.

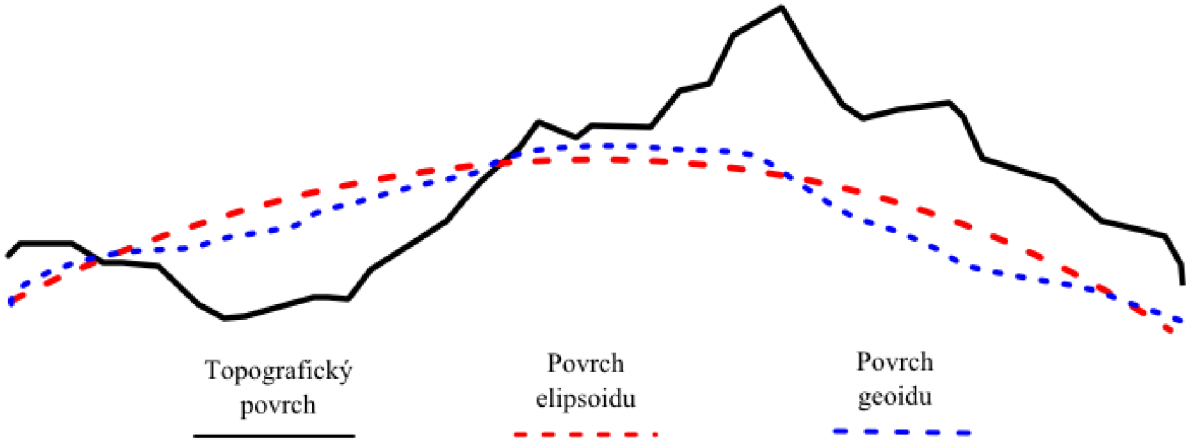
3.2 Model povrchu Zeme

K úplne presnému určeniu polohy na povrchu Zeme je nutný trojrozmerný priestor. Veľká členitosť a zvrásnenosť povrchu Zeme zvyšuje zložitosť topologických vzťahov medzi objektmi v priestore. Bežne sa preto v mapových aplikáciách stretávame s dvojrozmerným priestorom, ktorý predstavuje zjednodušený model povrchu Zeme [8].

Najjednoduchším modelom Zeme je povrch gule, ktorej tvar sa líši v sploštení Zeme na úrovni pólov. Preto bol zavedený ďalší, presnejší model Zeme – *geoid*. Jedná sa o plochu, ktorá odpovedá úrovni pokojnej strednej hladiny morí, pričom na určitých miestach zasahuje aj pod hladinu morí. Táto stredná hladina morí je v každom bode kolmá na smer gravitačnej sily. Zložitosť matematického popisu geoidu viedla k zavedeniu ďalšieho modelu – *referenčného (náhradného) elipsoidu*, ktorý je daný svojim stredom a dĺžkami kratšej a dlhšej poloosi. Elipsoid nahrádza goeoid s rôznou presnosťou v závislosti na voľbe parametrov dĺžky kratšej a dlhšej poloosi. Podľa rozsahu pokrytia územia sa elipsoidy rozdeľujú na [8]:

- **Lokálne** – sú určené k modelovaniu určitej časti zemského povrchu, čím zaručujú vysokú presnosť pri určovaní polohy. Pre Európu a Českú republiku sa používa *Besselov elipsoid*.

- **Globálne** – sú určené k modelovaniu celého zemského povrchu, pričom pri určovaní polohy dochádza k väčším nepresnostiam ako pri použití lokálneho. Najrozšírenejším globálnym elipsoidom je *WGS-84*, ktorý bol vyvinutý v roku 1984.



Obrázok 3.1 Vzťah modelov povrchu Zeme k zemskému povrchu [8]

3.3 Súradnicové systémy

Súradnicový systém je definovaný podľa Rapanta [8] nasledovne: „*Súradnicový systém je sada matematických pravidiel pre špecifikáciu spôsobu, akým sú súradnice priraďované k bodom v priestore.*“ Súradnicové systémy sú charakterizované referenčnou plochou (elipsoidom) a ich parametrami, typom kartografického zobrazenia, definíciou začiatku, definíciou súradnicových osí a jednotkami miery. Tieto systémy sú väčšinou medzi sebou vzájomne prevoditeľné. Podľa referenčného elipsoidu sa systémy rozdeľujú na lokálne (národné) a globálne. K najznámejším systémom v Českej Republike patrí:

- **WGS-84** – je globálny súradnicový systém typu šírka - dĺžka, ktorý sa využíva hlavne v spojení s GPS systémom. Dokáže určovať polohu s presnosťou na metre [7].

- **S-JTSK** – je lokálny súradnicový systém platný na území Českej a Slovenskej republiky, ktorý využíva Besselov elipsoid.

- **UTM –** je globálny súradnicový systém, ktorý rozdeľuje Zem do 60 zón. Každá z nich sa projektuje na valec [7]. Zóny majú pravouhlú súradnicovú sieť v metroch, pričom tieto súradnice sú platné len v rámci zóny. Česká a Slovenská republika patria do zón 33 a 34.

4 Online mapové služby

V tejto kapitole si najskôr veľmi stručne predstavíme technológiu skrývajúcu sa za poskytovaním mapových služieb. Následne si predstavíme aplikačné rozhrania (API, angl. *Application Programming Interface*) mapových služieb *Google Maps* a *Mapy.cz*, ktoré možno využiť k tvorbe vlastných odvodených mapových aplikácií. Predstavíme si aj knižnicu OpenLayers, ktorá tiež určená k tvorbe mapových aplikácií.

V súčasnosti existuje veľké množstvo online mapových služieb pre tvorbu vlastných mapových aplikácií. Služby poskytujú vlastné aplikačné rozhrania, ktoré slúžia k vytváraniu interaktívnych máp v prostredí webového prehliadača. Ponúkajú používateľom intuitívne grafické rozhranie na ovládanie pohybu mapy. Ďalej dodávajú ovládacie prvky slúžiace na výmenu mapových vrstiev, približovanie a odďaľovanie mapy, zobrazenie mierky mapy a výpisu licenčných informácií o zobrazovaných vrstvách.

Rozhrania uľahčujú prácu vývojárovi tým, že zapuzdrujú komunikáciu s mapovým serverom. Zavádzajú koncept tzv. mapových vrstiev (angl. *layers*), ktoré zjednocujú tematicky rovnaké geografické objekty do jedného celku a tým zjednodušujú prácu s objektmi. Poznáme rastrové a vektorové mapové podklady.

Rastrové vrstvy sú zložené z bitmapových obrázkov, ktoré primárne slúžia ako statické mapové podklady. Podklady sú pre každú úroveň priblíženia rozdelené na dlaždicovú mriežku (angl. *Tile Grid*), ktorá je reprezentovaná práve rastrovým obrázkom. Fyzicky bývajú dlaždice uložené v požadovanej mierke na mapovom serveri. API pri pohybe automatický generuje požiadavky na dlaždice, ktoré ešte neboli stiahnuté. Následne ich tiež automaticky zobrazuje v aplikácii. Vektorové vrstvy obsahujú geografické dáta, ktoré sa môžu dynamicky meniť.

Medzi najznámejších poskytovateľov mapových služieb v Českej republike patria *Google Maps, Yahoo! Maps, Bing Maps*, *Seznam Mapy.cz* a *OpenStreetMaps* (OSM). Všetky tieto služby okrem OSM poskytujú vlastné mapové podklady rôznych typov a vlastné API. OSM poskytuje iba vlastné mapové podklady, ktoré je možné zobraziť ľubovoľnými API alebo knižnicami pre zobrazovanie. V prípade splnenia licenčných podmienok je možné zobrazovať ľubovoľné mapové podklady pomocou knižníc s otvoreným zdrojovým kódom ako napr. *OpenLayers* alebo *Leaflet*, ktoré sa často využívajú v geografických informačných systémoch. V nasledujúcich podkapitolách si predstavíme:

- ***Google Maps API*** – ako zástupcu celosvetového poskytovateľa mapových služieb.
- ***Mapy API*** – ako zástupcu lokálneho poskytovateľa mapových služieb.
- ***OpenLayers*** – ako zástupcu vizualizačnej knižnice bez vlastných mapových podkladov.

4.1 Google Maps API

Google Maps je mapová webová služba zobrazujúca satelitné mapové podklady Zeme, ktorá bola do prevádzky uvedená v roku 2005 spoločnosťou Google. V tomto roku bolo do prevádzky uvedené aj *Google Maps API* slúžiace na vývoj vlastných mapových aplikácií. Tento produkt je dnes možné považovať za štandard medzi mapovými službami, pretože veľa súčasných mapových služieb sa práve ním inšpirovalo.

Google Maps poskytuje užívateľsky prívetivé rozhranie s možnosťou prepínania tematických máp, približovania a odďaľovania, pohybu po mape, plánovania cesty, geokódovania a spätného geokódovania. Väčšina týchto vymožeností je dostupných aj z Google Maps API, ktoré bolo na začiatku svojho vzniku určené len pre webové prehliadače. Neskôr s nástupom inteligentných telefónov sa Google Maps API rozšírilo aj na mobilné platformy ako iOS a Android. Google Maps API sa od svojho vzniku naďalej aktívne rozširuje a vyvíja, dôkazom čoho je aj rozsiahla dokumentácia, ukážky kódov a aktívne diskusné fórum. Momentálne sú vývojárovi k dispozícií nasledujúce rozhrania [9, 10]:

- **Google Maps JavaScript API v3** – rozhranie pre tvorbu mapových aplikácií pomocou jazyka JavaScript.
- **Google Places API** – rozhranie poskytujúce možnosť vyhľadania detailných informácií o rozličných geografických miestach.
- **Google Maps API Web Services** – rozhranie, ktoré sprostredkuje prístup k službám pre geokódovanie, plánovanie cesty, získanie výškových dát a získanie časových zón.
- **Google Maps Image API** – rozhranie umožňujúce vloženie panoramatických obrázkov ulíc do mapy (*Street View Image API*) alebo obrázku výseku mapy na webovú stránku (*Static Maps API*).

4.1.1 Mapové podklady a vrstvy

Google Maps dodáva mapové podklady s celosvetovým pokrytím. Podpora rôznych typov mapových podkladov a vrstiev sa líši v závislosti na lokalite. Medzi nezvyčajné mapové podklady patrí podpora pre Mars, Mesiac a pre oblohu Zeme. V rámci Google Maps API je možné zvoliť si zo štyroch typov mapových podkladov pre Zem a to [11]:

- **Cestná mapa** (angl. *roadmaps*) – základná mapa.
- **Satelitná mapa** (angl. *satellite*) – mapa so satelitnými snímkami Zeme.
- **Hybridná mapa** (angl. *hybrid*) – kombinácia cestných a satelitných máp.
- **Terénna mapa** (angl. *terrain*) – fyzická mapa, ktorá je vytvorená na základe informácií o teréne.

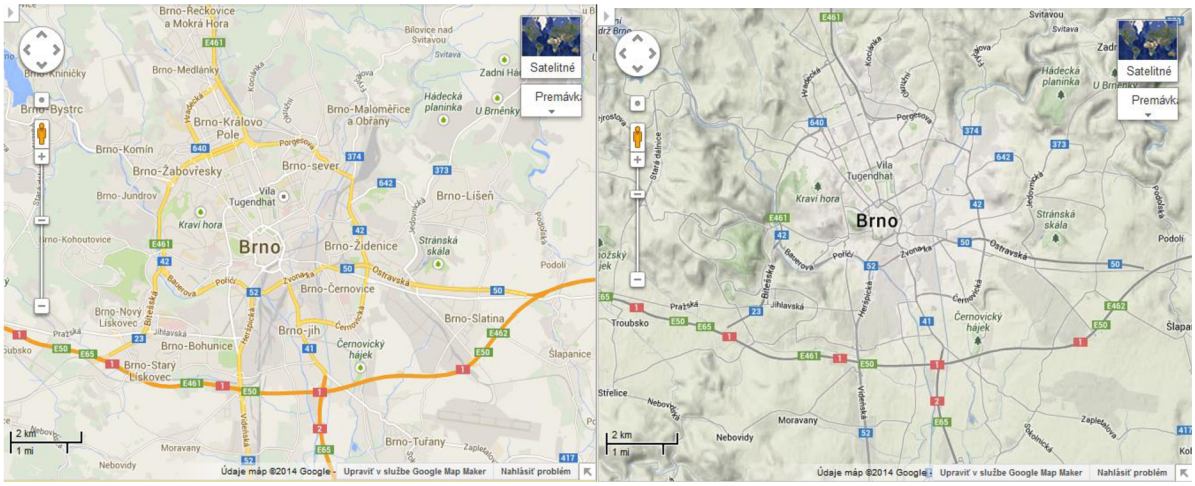
Služba ponúka aj niekoľko vlastných vektorových vrstiev na zobrazenie počasia, oblakov, dopravných spojov a dopravnej situácie na cestách. Okrem toho poskytuje aj vrstvu obsahujúcu cyklotrasy. Počet týchto cyklotrás je však v Českej republike veľmi nízky. Pre vlastné vektorové dáta sú k dispozícií vrstvy, ktoré dokážu automaticky rozparsovať a zobraziť dáta vo formátoch GeoRSS, KML, GeoJSON [12].

4.1.2 Licenčné podmienky a limity

K používaniu Google Maps API je potrebné vygenerovať unikátny kľúč (tzv. *API key*) pre doménu, z ktorej sa bude API využívať. Vďaka API kľúču je možné monitorovať spotrebu využívania služieb

API používanou aplikáciou. Bezplatne sa môže Google Maps API využívať pri splnení nasledujúcich podmienok [13]:

- Je zakázané prekrývanie alebo odstraňovanie mapových podkladov obsahujúcich logá spoločnosti Google.
- Je zakázané používanie služieb pre vytváranie podnikových (angl. *Enterprise*) aplikácii, pre sledovanie pohybu vozidiel alebo osôb v reálnom čase. Používanie takýchto služieb je možné až po zaplatení poplatkov.
- Výsledná aplikácia alebo služba musí byt verejná a bezplatná. Nemôže sa nachádzať na intranetovej sieti.
- Je zakázané používanie služieb Google Maps ako napr. geokódovanie bez použitia Google máp.
- Je zakázané poskytovať sexuálny obsah v odvodenej aplikácii spolu so službou.



Obrázok 4.1 Ukážka cestnej (vľavo) a terénnej (vpravo) mapy v Google Maps

4.2 Mapy API

Mapy.cz je mapová webová služba, ktorú vlastní spoločnosť Seznam. Ich *Mapy API* prešlo od svojho vzniku mnohými zmenami a rozšírilo sa o množstvo užitočných funkcií. Prevratnou verziou Mapy API je verzia 4.0, ktorá priniesla nové licenčné podmienky a zrušenie predchádzajúcich limitov. Vďaka ich detailným a rôznorodým mapovým podkladom patrí medzi najlepších poskytovateľov mapových služieb v Českej republike.

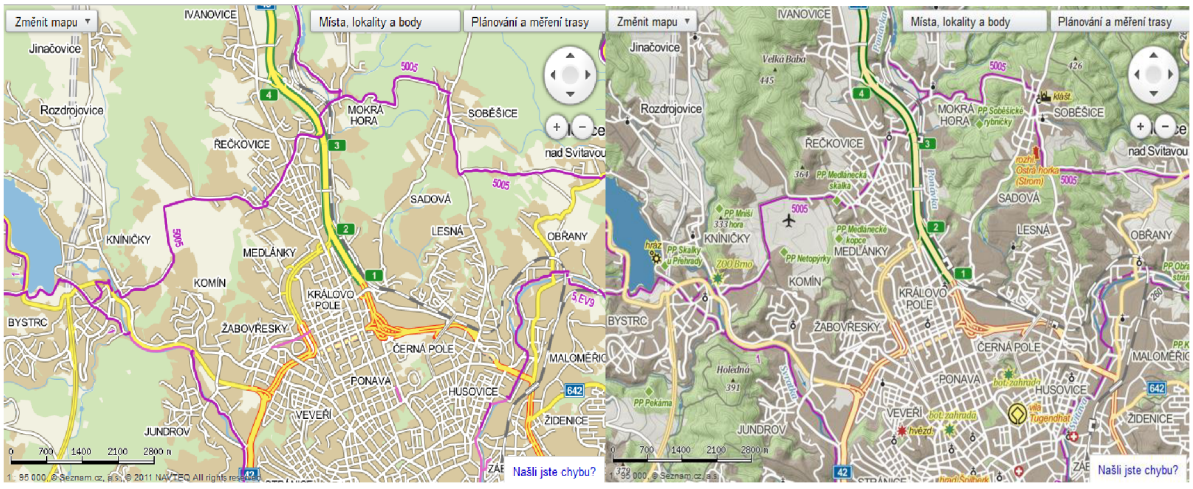
Rovnako ako Google Maps API, aj Mapy API poskytuje JavaScriptovú knižnicu, ktorá automaticky spravuje okno obsahujúce mapy a ponúka prvky na priblíženie a oddialenie a prepínanie podkladov a vrstiev. Nechýba ani podpora pre geokódovanie, spätné geokódovanie a prevody medzi súradnicovými systémami. API poskytuje dostatočnú dokumentáciu, ukážky kódov a diskusné fórum.

4.2.1 Mapové podklady a vrstvy

Jedná sa o veľmi prepracované a kvalitné mapy, ktoré sú pravidelné aktualizované. Sú zamerané hlavne na Českú a Slovenskú republiku. Pre Európu je dostupná základná mapa, ktorej úroveň je postačujúca na základnú navigáciu auta. Medzi mapové podklady schopné zobrazenia v Mapy API patria [14]:

- **Základná mapa** – zjednodušená mapa dopravných sietí, budov a rôznych bodov záujmu ako napr. zástavky, divadlá apod.
- **Letecká mapa** – letecké snímky realizované firmou Geodis.
- **Turistická mapa** – turistická mapa, ktorá obsahuje topografické vrstvy, výškopis, tieňovaný reliéf, turistické body záujmu a náučné chodníky.
- **Historická mapa** – historická mapa dnešného územia Českej Republiky z rokov 1836-1852.

Ďalej API obsahuje vektorové vrstvy ako *Turistické trasy*, *Cyklotrasy* alebo *Lyžiarske trasy*. Pre vlastné vektorové dáta sú k dispozícii vrstvy, ktoré dokážu automaticky rozparsovať a zobraziť dáta vo formátoch KML a GPX.



Obrázok 4.2 Ukážka základnej (vľavo) a turistickej (vpravo) mapy so zapnutými cyklotrasami v *Mapy.cz*

4.2.2 Licenčné podmienky a limity

Použitie *Mapy API* je úplne zdarma bez akýchkoľvek limitov alebo obmedzení (aj pre komerčné účely) pri splnení určitých podmienok, ktorými sú napr. [15]:

- Je zakázané brániť používateľovi v prístupe na stránky (nespoplatnené prihlasovanie na stránky je povolené).
- Je zakázané používanie API s inou službou alebo aplikáciou zaoberajúcou sa navigáciou alebo sledovaním vozidiel.

- Je zakázané používať API v spojení s inou službou alebo aplikáciou na poskytovanie dát na vykonávanie rôznych analytických, projekčných a developerských prác.

- Je zakázané poskytovať službu súvisiacu s kriminálnymi činmi, so sexuálnym obsahom, s hazardnými hrami alebo s obsahom podporujúcim predaj zákonom zakázaného tovaru.

- Je zakázané prekrývanie loga a textov dodávateľov mapových podkladov.

4.3 OpenLayers 3

OpenLayers je klientská JavaScriptová knižnica s otvorenými zdrojovými kódmi pre tvorbu interaktívnych webových máp, ktoré sú zobraziteľné v moderných webových prehliadačoch.

Knižnica sa objavila v roku 2006 ako alternatíva ku komerčným poskytovateľom mapových rozhraní. Avšak popularitu začala získavať až keď ju OSM projekt začal využívať pre zobrazovanie svojich mapových podkladov. Dnes ide o jednu z najprepracovanejších mapových webových knižníc, ktorá pracuje s geografickými informáciami a implementuje veľa štandardov vydaných OGC (angl. *Open Geospatial Consorcium*) [16].

OpenLayers sa aktuálne nachádza v stabilnej verzii číslo 3, ktorá prináša oproti predchádzajúcim verziám lepší výkon, príťažlivejšie zobrazovacie komponenty a API využívajúce najnovšie funkcie technológií HTML5 a CSS3 (napr. WebGL, Canvas). Medzi hlavné prednosti knižnice patrí [16]:

- podpora zobrazenia dát z rôznych zdrojov implementujúcich OGC štandardy, napr. WMS, WFS,

- podpora dlaždicových vrstiev od OSM, Bing, MapBox, Stamen, MapQuest ai.,

- podpora čítania/zápisu z/do rôznych dátových formátov ako napr. GeoJSON, TopoJSON, KML, GML, GPX atď.,

- podpora mapových projekcií a práce s nimi,

- podpora mobilných zariadení,

- rozšíriteľná paleta ovládacích prvkov mapy,

- otvorené zdrojové kódy,

- rozsiahla dokumentácia.

4.4 Porovnanie jednotlivých API

Skúmané API sú porovnané v tabuľke 1, ktorá prehľadne sumarizuje možnosti a funkcie vyššie popísaných rozhraní. Vhodnejšie funkcie pre potreby vyvíjaného systému sú pre každý riadok hrubo vyznačené.

Obrázok 4.3 Ukážka OSM mapových podkladov v OpenLayers 3

	Mapy API	*Google Maps API*	*OpenLayers 3*
Mapové podklady	**základné, letecké, historické a turistické**	základné, satelitné, hybridné a terénne	**vrstvy tretích strán**
Dostupné vektorové vrstvy	**cyklistické, turistické trasy**	dopravná situácia, počasie, verejná doprava	**vrstvy tretích strán**
Vlastné vrstvy	KML, GPX, WMS, WMTS, obrázok	KML, GeoRSS, WMS, GeoJSON	**GeoJSON, TopoJSON, WFS, WMS, GPX, KML, GML, obrázok ...**
Dostupnosť máp	Európa	**svet**	**svet**
Mapové podklady tretích strán	**áno**	**áno**	**áno**
Detailnosť máp v Českej republike	**kvalitné spracovanie Českej republiky**	uspokojivé spracovanie Českej republiky	**vrstvy tretích strán**
Dokumentácia a ukážky	uspokojivá	**výborná**	**výborná**
Vývojárska komunita	lokálna	**celosvetová**	**celosvetová**
Kapacitné obmedzenia	**žiadne**	25 000 načítaní máp za deň	**žiadne**
Vektorové trasy	**áno**	**áno**	**áno**
Geokóding	**áno**	**áno**	služby tretích strán
Spätný geokóding	**áno**	**áno**	služby tretích strán
Verzia API	4.11	3.19	3.4.0
Ďalšie služby	žiadne	**nadmorská výška, plánovanie trás, vzdialenosť medzi bodmi**	žiadne

Tabuľka 4.1 Porovnanie popísaných aplikačných rozhraní pre potreby vyvíjaného systému

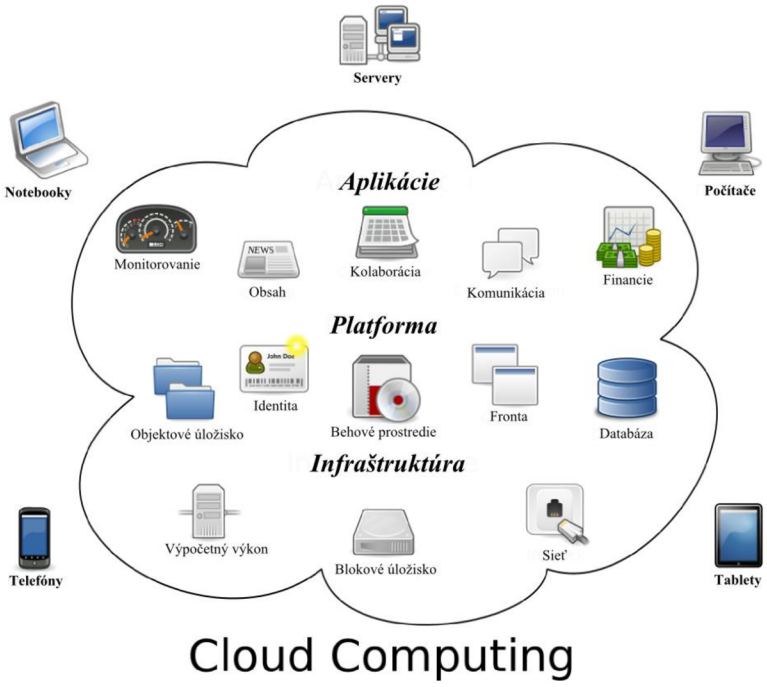
5 Cloud computing

V tejto kapitole si najskôr charakterizujeme cloud computing, jeho servisné modely a poskytovateľov. Následne si detailnejšie predstavíme predstaviteľov cloudu Google App Engine a OpenShift.

Informačné technológie sa veľmi rýchlo menia. S týmito zmenami prichádzajú aj nové modely architektúry a požiadavky na poskytovanie IT služieb. V poslednej dobe sa čoraz viac presadzuje technológia cloud computingu.

NIST (angl. *National Institue of Standards and Technology*) definuje Cloud Computing vo voľnom preklade nasledovne [17]: *„Cloud computing je model umožňujúci na vyžiadanie všadeprítomný a pohodlný sieťový prístup k zdieľanému združeniu konfigurovateľných výpočtových prostriedkov (napr. sietí, serverov, dátových úložísk, aplikácií a služieb), ktoré môžu byť rýchlo poskytované a uvoľňované s vynaložením minimálneho úsilia na správu alebo na interakciu poskytovateľa služby.“*

Jedná sa o nový typ poskytovania IT prostriedkov ako napr. hardvérovej infraštruktúry, softwarových programov, webových serverov, databázy a mnohých ďalších. V cloude sú všetky prostriedky zdieľané a v dobe nečinnosti môžu byť poskytnuté iným konzumentom. V dnešnej dobe sa táto technológia využíva pre kancelárske aplikácie (Google Documents, Microsoft Office 365), online operačné systémy (Chrome OS, iCloud), rôzne biznis riešenia alebo na riešenia distribuovaných výpočtov.



Obrázok 5.1 Znázornenie cloud computingu [18]

5.1 Výhody a nevýhody

K hlavným výhodám cloud computingu patrí [19]:

- **Cena –** konzument platí iba za spotrebu funkčných a dostupných aplikácií a služieb.

- **Flexibilita a škálovatelnosť –** možnosť pružnej reakcie na zmenu požiadaviek aplikácie. Konzument si môže kedykoľvek doobjednať nové alebo preobjednať aktuálne prostriedky v závislosti na jeho potrebách a tým znižovať náklady na správu a údržbu hardvéru a softvéru.

- **Samoobslužnosť –** štandardné služby sú poskytované v podobe servisného katalógu, z ktorého je možné objednávať samoobslužne. Cloud automaticky pripraví všetky prostriedky nutné k používaniu.

- **Prístup cez internet –** poskytované prostriedky sú dostupné odkiaľkoľvek cez internetové pripojenie.

- **Aktualizovateľnosť** – celý software je automaticky aktualizovaný.

Naopak k hlavným nevýhodám patrí to, že naše citlivé dáta nemáme pod fyzickou kontrolou, pretože sú uložené na serveroch v cloude. Ďalšou nevýhodou môže byť viazanosť na API platformy v dôsledku čoho môže migrácia služby k inému poskytovateľovi činiť veľké problémy.

5.2 Rozdelenie cloud computingu

Najčastejšie sa cloud computing rozdeľuje do dvoch skupín. Prvá skupina je nazvaná modely nasadenia a vyjadruje ako je cloud poskytovaný. Patria sem [17, 19]:

- **Privátne cloudy** (angl. *private clouds*) – poskytujú cloud, ktorý môže byť použitý výlučne jednou organizáciou. Služby nie sú dostupné mimo intranetové prostredie.

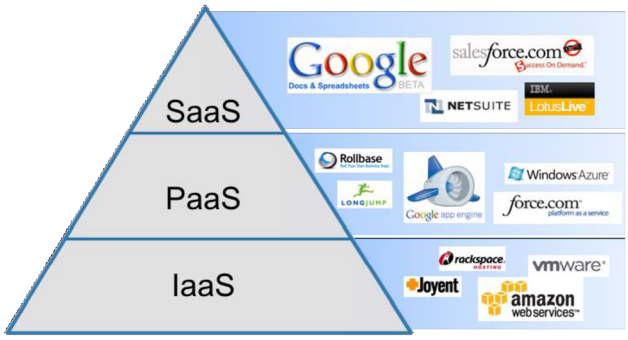
- **Komunitné cloudy** (angl. c*ommunity clouds*) – poskytujú cloud, ktorý môže využitý len špecifickou komunitou konzumentov z organizácií, ktoré majú rovnaké záujmy.

- **Verejné cloudy** (angl. *public clouds*) – poskytujú cloud ako verejnú službu. Službu môže využívať ktokoľvek.

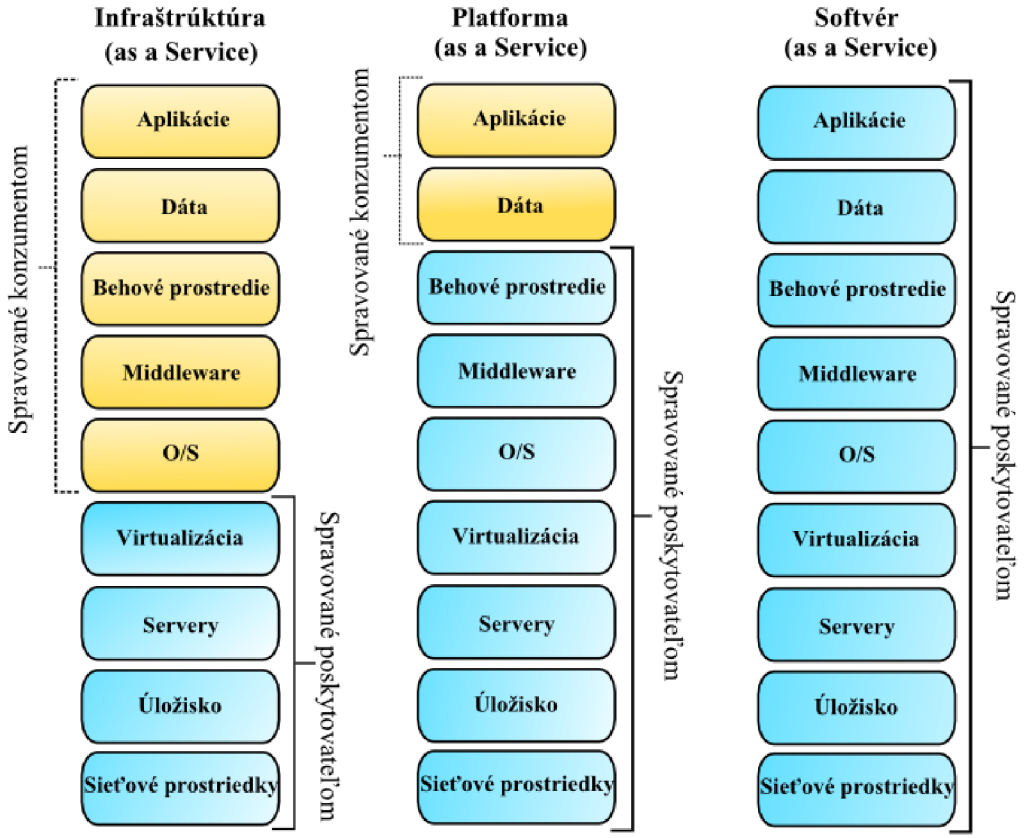
- **Hybridné cloudy** (angl. *hybrid clouds*) **–** jedná sa o kombináciu verejných, komunitných a privátnych cloudov.

Druhou skupinou je skupina nazvaná servisné modely. Patrí sem infraštruktúra ako služba (angl. *Infrastructure-as-a-Service*, IaaS), platforma ako služba (angl. *Platform-as-a-Service*, PaaS) a software ako služba (angl. *Software-as-a-Service*, SaaS). Jednotlivé servisné modely sú charakterizované v nasledujúcich podkapitolách. Obrázok 5.2 zobrazuje najznámejších poskytovateľov cloudu. Na obrázku 5.3 je zobrazená zodpovednosť konzumenta a poskytovateľa cloudu za jednotlivé prostriedky v servisných modeloch.

Existuje ešte celý rad ďalších kategórií ako napr. biznis procesy ako služba (angl. *Business Process as a Service*, BpasS), dáta ako služba (angl. *Data as a Service*, DaaS), sieť ako služba (angl. *Network as a Service*, NaaS) a iné [19].



Obrázok 5.2 Ukážka poskytovateľov cloudu pre jednotlivé servisné modely [20]



Obrázok 5.3 Obrázok zodpovedností konzumenta a poskytovateľa cloudu za jednotlivé prostriedky v servisných modeloch [21]

5.2.1 Infraštruktúra ako služba

IaaS je najzakladanejšou cloudovou službou. Poskytuje výpočtové, úložné, sieťové a iné bázové prostriedky, na ktorých je konzument schopný nasadenia a behu ľubovoľného softvéru ako napr. operačného systému, databázy, webového serveru alebo iných aplikácií. Hardvérové prostriedky sú už pripravené priamo na prenájom a použitie. Konzument sám nespravuje ani neriadi základnú infraštruktúru, má kontrolu iba nad operačným systémom, úložiskom a nasadenými aplikáciami. PaaS a SaaS využívajú pre svoj beh práve IaaS [19].

K jeho najznámejším zástupcom patria napr. služby *Amazon Web Services* alebo *Google Cloud Platform*.

5.2.2 Platforma ako služba

V *PaaS* je konzumentovi poskytnutá výpočtová platforma, na ktorú si konzument môže nasadzovať vlastné aplikácie. Aplikácie sú tvorené využitím programovacích jazykov, knižníc, služieb a nástrojov, ktoré poskytovateľ podporuje. Sú to konkrétne napr. Java EE aplikačné servery, HTTP servery a databázové servery. Konzument nemá prístup k riadeniu infraštruktúry (serveru, siete, operačného systému), ale má kontrolu nad svojou aplikáciou a jej konfiguračnými nastaveniami [17].

K najznámejším zástupcom PaaS patria služby *Google App Engine*, *Windows Azure* a *OpenShift*.

5.2.3 Softvér ako služba

SaaS poskytuje aplikáciu ako službu, pričom sa konzument nemusí zaujímať o jej technické prevedenie, tj. použitú infraštruktúru a platformu. Aplikácie sú prístupné z rôznych klientskych zariadení buď cez webové prostredie alebo programové rozhranie [17].

Príkladom takýchto služieb sú online emailové služby alebo online kancelárske aplikácie. K najznámejším zástupcom SaaS patria napr. *SalesForce* alebo *Google Apps*.

5.3 Google App Engine

Google App Engine (GAE) je súčasťou cloud platformy s názvom Google Cloud platforma (angl. *Google Cloud Platform*, GCP) od spoločnosti Google. Veľkou výhodou GCP sú otvorené API, ktoré umožňujú integráciu všetkých častí GCP a tým aj využitie všetkých jeho funkcií.

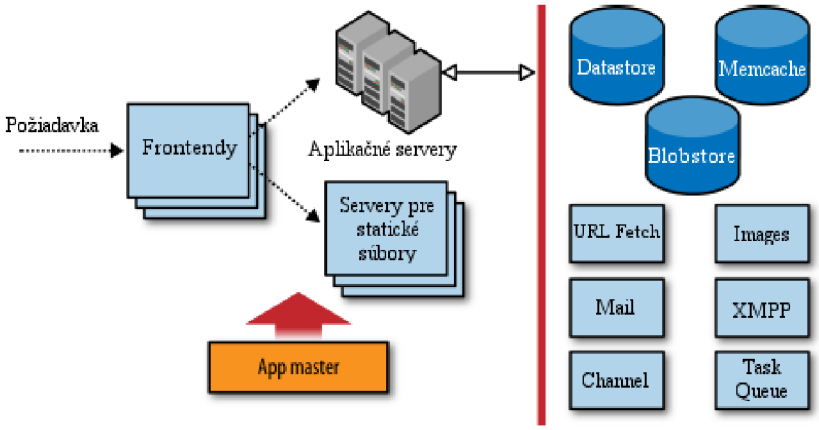
GCP poskytuje pre konzumenta produkty zo všetkých troch servisných modelov cloudu. Jednotlivé produkty sú stručne charakterizované v tabuľke 5.1.

GAE je webová hostingová služba zaobstarávajúca aplikačnú infraštruktúru na PaaS úrovni. Umožňuje vývoj webových aplikácií pomocou preddefinovaných knižníc a ich nasadenie na Google serveri. Poskytuje množstvo behových prostredí (angl. *runtime enviroments*) ako napr. Java, Go, PHP alebo Python prostredie. Tieto prostredia sú vytvorené tak, aby zabezpečili rýchly a bezpečný beh aplikácie bez vzájomného rušenia s inými aplikáciami v systéme. Aplikácie sú schopné automatického škálovania, vďaka čomu bežia spoľahlivo s veľkým množstvom dát bez zníženia výkonu [22, 23].

Celá architektúra GAE sa skladá z troch častí. Prvou časťou je behové prostredie, ktoré obsluhuje a riadi inštancie aplikácie. Druhou časťou je škálovateľné dátové úložisko, tzv. *Datastore*. Poslednou časťou sú škálovateľné služby. Jednotlivé časti architektúry sú charakterizované v nasledujúcich podkapitolách.

5.3.1 Behové prostredie

Behové prostredie je abstraktnou vrstvou nad operačným systémom, ktorá GAE umožňuje spravovať alokáciu zdrojov, výpočty, obsluhu požiadaviek, škálovanie a rozloženie záťaže. Funkcie, ktoré zvyčajne vyžadujú znalosť operačného systému sú poskytované službou mimo behového prostredia [22].



Obrázok 5.2 Architektúra platformy GAE [22]

Produkt	Model	Popis
Google Cloud Storage	IaaS	Služba poskytujúca trvalé a vysoko dostupné úložisko s jednoduchým prístupovým API.
Google Compute Engine	IaaS	Služba poskytujúca ľubovoľné virtuálne stroje v závislosti na potrebách konzumenta.
Google App Engine	PaaS	Služba poskytujúca prostredie pre prevádzku webových aplikácii a služieb postavených na technológiách danej platformy.
Cloud SQL	SaaS	Služba poskytujúca relačnú MySQL databázu.
Cloud Datastore	SaaS	Služba poskytujúca NoSQL databázu, ktorá sa automaticky škáluje podľa potrieb .
BigQuery	SaaS	Služba pre analýzu veľkých dát pomocou dotazov podobných SQL.
Prediction API	SaaS	Služba, ktorá dodáva REST (angl. *Representational State Transfer*) rozhranie obsahujúce Google algoritmy pre strojové učenie na analýzu dát a predpovedanie budúcich výsledkov.
Translate API	SaaS	Služba pre prekladanie textov do iných jazykov.
Cloud Endpoints	SaaS	Služba pre vytváranie RESTful služieb z programového kódu, ktorá je následne dostupná z rôznych klientov ako iOS, Android a JavaScript.

Tabuľka 5.1 Prehľad produktov poskytovaných Google Cloud platformou [24]

GAE po prijatí požiadavky najskôr identifikuje aplikáciu z doménového priestoru mien aplikácií a z viacerých Google serverov zvolí ten najvhodnejší, ktorý dokáže zaistiť rýchlu odpoveď. Následne je aplikácia nasadená do obmedzeného sandbox prostredia a uvedená do prevádzky. Kvôli úspore výpočtového výkonu a pamäti bežia aplikácie v prostredí dovtedy, kým aplikácia vykonáva nejakú činnosť [22].

Sandbox prostredie izoluje aplikáciu do jej vlastného bezpečného a spoľahlivého prostredia, ktoré zaisťuje, že aplikácie môžu vykonávať len také akcie, ktoré nezasahujú do výkonu a škálovateľností iných aplikácií. Prostredia sú nezávislé na hardvéri, operačnom systéme a fyzickej lokalite webového serveru. Sandbox prostredie má aj určité obmedzenia medzi ktoré patrí [25]:

- Požiadavky musia byť spracované do 60 sekúnd. Po ich vypršaní musí byť klientovi vrátená odpoveď a aplikácia musí byť odstránená z behového prostredia.

- Aplikácie nemôžu zapisovať dáta do súborového systému a ani nemôžu využívať JNI (angl. *Java Native Interface*).

- Aplikácia je dostupná len pomocou protokolu HTTP (alebo HTTPS) požiadaviek s dotazmi na štandardné porty.

5.3.2 Datastore

Väčšina dnešných webových aplikácií potrebuje ukladať a získavať informácie z databázového úložiska. GCP poskytuje pre tieto potreby nezávisle spoplatnené produkty Google Cloud SQL a Google Cloud Storage.

GAE ide svojou cestou. Prevádzkuje vlastnú NoSQL databázu nazývanú Datastore, ktorá poskytuje voľné kvóty (viď. kapitola 5.3.5) s dennými obmedzeniami. Datastore je objektové dátové úložisko, ktorého databázová štruktúra nemá schému. Poskytuje robustné, škálovateľné, vysoko dostupné úložisko s atomickými transakciami a dotazovacím systémom. Na rozdiel od klasických relačných databáz používa distribuovanú architektúru a odlišné vyjadrenie vzťahov medzi entitami. To mu dovoľuje vysokú flexibilitu, dostupnosť a automatické škálovanie pre veľké množstvo dát.

Štruktúra dát

K ukladaniu dát sa používajú tzv. entity (angl. *entities*). Každá entita má jednu alebo viacej vlastností (angl. *properties*). Každá vlastnosť je pomenovaná a má určenú hodnotu z možných predefinovaných dátových typov. Každá entita má pomenovaný druh (angl. *kind*), ktorý sa využíva pre potreby dotazov a vlastný kľúč, ktorý ju jedinečne identifikuje v rámci databázy. GAE vyžíva druh a kľúč entity na určenie uloženia entity v rámci kolekcie serverov. Veľkosť ukladaných entít do služby Datastore je maximálne 1 MB [22].

Entity v Datastore vytvárajú hierarchiu podobnú priečinkovej štruktúre známu zo súborových systémov. Pri ukladaní entity je možné vybrať jednu rodičovskú entitu a tým dosiahnuť vzťah rodič-dieťa. Toto zoskupenie entít sa nazýva entitná skupina (angl. *entity group*). Iným spôsobom vytvárania vzťahu medzi entitami je uloženie kľúča jednej entity do vlastnosti druhej entity. Tento spôsob je podobný cudzím kľúčom z relačnej databázy, avšak má určité nedostatky. Ani kľúč entity, ani druh entity a ani vzťah typu rodič-dieťa medzi entitami nemôže byť zmenený po tom, ako bola entita uložená do databázy [22, 26].

Na prvý pohľad sa toto uloženie dát podobá relačnej databáze, kde entity predstavujú tabuľky, druhy entity predstavujú riadky tabuľky a vlastnosti entity predstavujú stĺpce tabuľky. Nachádzame tu však aj viacero rozdielov. Prvým rozdielom je, že Datastore nepoužíva žiadnu schému pre štruktúru dát a preto dve entity rovnakého druhu nemusia mať zhodné vlastnosti a dokonca ani zhodný dátový typ pre určitú vlastnosť. Druhým rozdielom je odlišnosť medzi vlastnosťou entity a stĺpcom relačnej tabuľky. Entita je schopná ukladať kolekcie hodnôt do jednej vlastnosti, pričom tabuľka do stĺpca väčšinou nie [22].

Dotazy a indexy

Na získanie entít z Datastore sú podporované 2 typy dotazov. Prvým typom je dotaz vykonaný na základe kľúča a druhým je dotaz vykonaný na základe hodnôt vlastností. V dotazoch môžeme nastavovať [26]:

- **druh entity**, na ktorom sa dotaz vykoná.

- **filtre** pracujúce nad hodnotami vlastností, kľúčov a rodičov entít (tzv. ancestor dotazy),
- **zoradenie** výsledkov,
- **typ projekcie** navrátenej entity (celá entita, podmnožina entity, kľúč entity).

V predchádzajúcich možnostiach nastavenia nie je žiadna zmienka o tom, ako vykonávať zložité dotazy nad viacerými entitami mimo entitnej skupiny, nakoľko typy dotazov známe ako *join* dotazy z relačných databáz nie je možné vykonávať z dôvodu distribuovanej architektúry Datastore databázy.

Každý dotaz má odpovedajúci index. Pri vytváraní indexu dochádza počas kompilácie dotazu k jeho predspracovaniu. Vďaka indexom je každý dotaz iba jednoduchým čítaním z tabuľky predspracovaných hodnôt, pričom sa už iba vyberú odpovedajúce hodnoty na základe hodnôt filtru. Index robí dotaz pri čítaní hodnôt veľmi rýchlym, ale na druhej strane robia aktualizáciu dát zložitou a veľmi nákladnou [26].

Transakcie

Transakcie v Datastore sú veľmi podobné transakciám v relačných databázach. Datastore poskytuje ACID transakcie pomocou optimistického riadenia súbežnosti (angl. *optimistic concurrency control*). Každý pokus o vloženie, modifikáciu alebo vymazanie entity sa odohráva v kontexte transakcie, čím sa zachováva integrita dát. Transakcie pracujú nad entitnými skupinami [26].

Ak je potrebné vykonávať transakcie nad viacerými entitami patriacimi do rôznych entitných skupín, tak GAE poskytuje medziskupinové (angl. *cross-group*) transakcie. Takáto transakcia môže pracovať maximálne nad piatimi entitnými skupinami a vykonávať dotazy iba nad ich dátami. Pri používaní týchto transakcií sa môže prejaviť zvýšená latencia systému [26].

ORM rámce

Pre prácu s dátovým úložiskom poskytuje GAE iba nízkoúrovňové API, ktoré nie je veľmi pohodlné. DataNucleus, výrobca ORM (angl. *Object Relational Mapping*) nástrojov, preto dodáva pre GAE implementácie štandardných Java EE rámcov ako *Java Persitence API* (JPA) a *Java Data Objects* (JDO). Z dôvodu absencie štruktúry schémy databázy, sú žiaľ možnosti týchto rámcov veľmi obmedzené v porovnaní s ich implementáciami pre relačné databázy. JDO je navrhnuté hlavne pre prácu s postrelačnými databázami, a preto podporuje viac funkcií ako JPA.

Objectify[1] je veľmi jednoduchý rámec navrhnutý pre Datastore. Predstavuje akúsi medzivrstvu medzi nízkoúrovňovým API a JDO/JPA nástrojmi. Pomocou neho je možné používať všetky natívne vlastnosti Datastore, ktoré JDO/JPA neposkytuje. Je navrhnutý tak, aby nováčikovia boli okamžite schopný práce s GAE Datastore a dosiahli jeho maximálny potenciál.

5.3.3 Služby

Ako už bolo spomenuté, GAE poskytuje niekoľko automaticky škálovateľných služieb užitočných pre webové aplikácie. Behové prostredie poskytuje API, cez ktoré sú jednotlivé služby dostupné aplikácii. Služby sú spravované oddelene od inštancií aplikácie a bežia mimo sandbox prostredia. Medzi služby patrí [22]:

- **Memcache** – služba slúži ako krátkodobé dátové úložisko, ktoré ukladá dáta do pamäte miesto disku. Záznamy sú typu kľúč hodnota.

[1] https://github.com/objectify/objectify

- **URL Fetch** – služba dokáže pristupovať k zdrojom umiestneným na URL pomocou HTTP alebo HTTPS požiadaviek.

- **Mail** – služba, ktorá umožňuje prijímanie a posielanie emailových správ vrátane príloh.

- **OAuth –** služba využíva OAuth protokol, pomocou ktorého môžeme využívať iných poskytovateľov prihlasovacích služieb. Prístup k používateľským dátam iných aplikácií je možný bez toho, aby sme týmto aplikáciám poskytli nejaké údaje.

- **Google Cloud Endpoints –** služba, ktorá sa skladá z nástrojov a knižníc umožňujúcich generovanie aplikačných rozhraní a klientskych knižníc z GAE aplikácií. Vytvárajú tak aplikáciu dostupnú pre webových alebo mobilných klientov cez webovú službu [27]. Na obrázku 5.3 je možné vidieť základnú architektúru aplikácie pri využívaní GAE a koncových bodov.

- **XMPP** – služba, ktorá umožňuje prijímanie a zasielanie okamžitých správ aplikáciou nasadenou v GAE.

- **Blobstore** – služba, ktorá poskytuje úložný systém pre ukladanie veľkých dát ako sú napr. videá, hudba alebo obrázky.

- **Images** – služba pre spracovanie obrázkov, ktorá poskytuje jednoduché transformačné funkcie s obrázkovými dátami uloženými vo formáte JPEG alebo GIF.

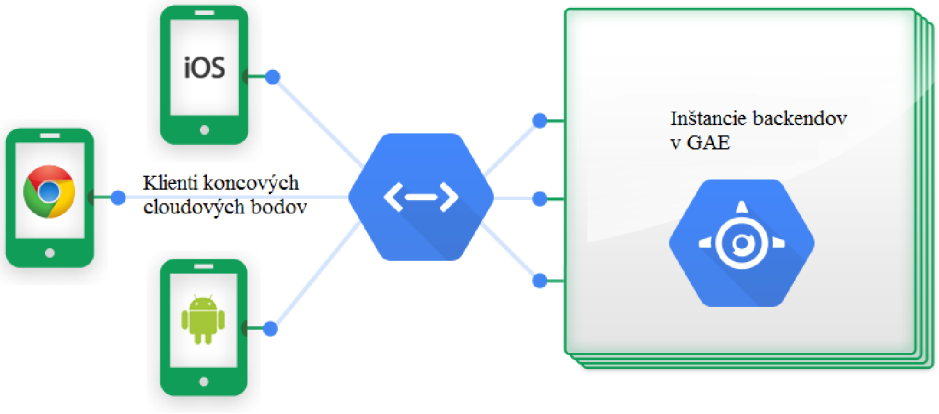
- **Task Queues** – služba obsahuje frontu úloh, ktoré sa vykonávajú na pozadí bez časového limitu obsluhy požiadavku.

- **Search –** služba poskytuje fulltextové vyhľadávacie služby nad indexovanými dokumentmi.

- **Users** – služba, ktorá dáva aplikáciám (napr.: Google Mail, Google Docs alebo Youtube) prístup k používateľskému prihlasovaciemu systému spoločnosti Google. Aplikácia môže povoliť používateľom prihlasovať sa do systému pomocou ich Google účtov, overiť ich identitu a získať tak informácie o emailovej adrese alebo používateľskom mene.

5.3.4 Poplatky a limity

GAE poskytuje bezplatné nasadenie a prevádzku aplikácie dovtedy, kým sa neprekročia určité limity. Limity sú nastaviteľné pre jednotlivé zdroje (šírka pásma alebo úložisko). Sú merané a účtované v gigabajtoch. V GAE sa nazývajú kvótami. Dostupné sú bezplatné, účtovateľné a bezpečnostné kvóty. Ako náhle je nejaká kvóta prekročená v rámci určitého časového obdobia, tak je aplikácia pozastavená. Predchádzajúci scenár platí iba pre bezplatné kvóty. Pre účtovateľné kvóty môže správca aplikácie nastaviť maximálny rozpočet, ktorý je aplikáciou možné vyčerpať za jeden deň. V každej aplikácií dochádza najskôr k čerpaniu bezplatných kvót. Po ich vyčerpaní bude aplikácia pracovať dovtedy, kým nedôjde k spotrebovaniu nastaveného maximálneho rozpočtu pre daný deň [28].



Obrázok 5.3 Základná architektúra koncových bodov [27]

Bezpečnostné kvóty sú nastavené fixne a nie je možné ich zvýšiť ani za prípadný poplatok. Chránia integritu GAE systému a zabezpečujú, že jedna aplikácia nevyčerpá všetky zdroje na úkor iných aplikácií. Medzi bezpečnostné kvóty patria [28]:

- **Denné kvóty** (angl. *daily quotas*) – sú resetované denne. Platené aplikácie môžu prekročiť tieto kvóty pokiaľ si nevyčerpajú rozpočet.

- **Minútové kvóty** (angl.*per-minute guotas*) – chránia aplikáciu pred rýchlym vyčerpaním zdrojov za veľmi krátky časový úsek a bránia ostatným aplikáciám zahltiť daný zdroj.

Platené aplikácie majú vyššie denné a minútové kvóty ako bezplatné aplikácie. Stručný prehľad najdôležitejších kvót je zobrazený v tabuľke 5.2.

5.3.5 Behové prostredie jazyka Java

GAE poskytuje viacero behových prostredí k vývoju v rôznych programovacích jazykoch. Jedným z nich je aj Java, ktorá umožňuje vyvíjať webové aplikácie s použitím štandardných Java technológií. Toto prostredie však podporuje len podmnožinu javového behového prostredia (angl. *Java Runtime Environment, JRE*), pretože mnoho tried JRE by nenašlo uplatnenie či už kvôli architektúre systému alebo sandbox prostrediu. Príkladom je napr. práca so súborovým systémom alebo programovým ukončovaním aplikácie.

Prostredie Javy poskytuje niektoré *Java Enterprise Edition* (Java EE) rámce k práci s webom, databázami a službami. Patria sem rámce ako *Java Servlets* (verzia 2.5), JDO, JPA, JAXB, JavaMail *a* JCache. Podpora týchto štandardou by mala uľahčovať migráciu aplikácií zo/do servlet kontajneru. Kvôli starším verziám rozhrania, obmedzenému JRE a architektúre systému (najmä databáze), môže migrovanie existujúcej aplikácie do bezplatnej verzie GAE spôsobovať značné komplikácie.

K lokálnemu vývoju a testovaniu aplikácie GAE dodáva Google vlastný webový server. Tento lokálny server dokáže simulovať prácu jednotlivých služieb a databázy Datastore. V tabuľke B.1 prílohy B sú zobrazené najznámejšie rámce schopné behu na platforme GAE. Zdroj informácií zobrazených v tabuľke už nie je momentálne dostupný.

Zdroje	*Štandardná bezplatná kvóta*		*Štandardná platená kvóta*		*Cena*
	Denný limit	*Maximálne množstvo*	*Denný limit*	*Maximálne množstvo*	
					Obecné limity
Šírka pásma In	1 GB	56 MB/minúta	žiadne	žiadne	
Šírka pásma Out	1 GB	56 MB/minúta	1 GB zadarmo; max 14 400 GB	10 GB/minúta	$0.12/GB
					Datastore
Množstvo dát	1 GB		1 GB zadarmo žiadne max		$0.18/GB/mesiac
Počet indexov	200		200		
Zápis operácie	50 000		neobmedzene		$0.06/100 000 operácií
Čítacie operácie	50 000		neobmedzene		$0.06/100 000 operácií
Malé operácie					bezplatné
					Mail služba
Volania API	100 volaní	32/minútu	1,7 milióna volaní	4 900/minútu	
Počet správ	100 správ	8/minútu	100 správ	5 100/minútu	$0.0001
					URL Fetch služba
Volania API	657 000 volaní	3 000/minútu	46 000 000 volaní	32 000/minútu	
					Task Queue služba
Volania API	100 000 volaní		1 000 000 000 volaní		
Počet uložených úloh	1 000 000		10 000 000 000		

Tabuľka 5.2 Základné kvóty a poplatky v GAE [28, 29]

5.4 OpenShift

OpenShift je hybridná PaaS platforma spoločnosti Redhat, ktorá sa snaží o zavedenie určitej miery štandardov do oblastí PaaS riešení. Vývojárom aplikácii umožňuje rýchly vývoj, nasadenie pomocou verziovacieho systému Git[2] a automatické škálovanie. Dnes existujčú tri rôzne verzie platformy OpenShift:

- **OpenShift Origin** – je bezplatná verzia platformy s otvorenými zdrojovými kódmi, ktorá beží na notebookoch, serveroch, verejných alebo privátnych cloudoch. Podporu a vývoj zabezpečuje komunita vývojárov. Táto verzia tvorí jadro ostatných verzií a preto je hlavný vývoj zameraný práve na ňu.

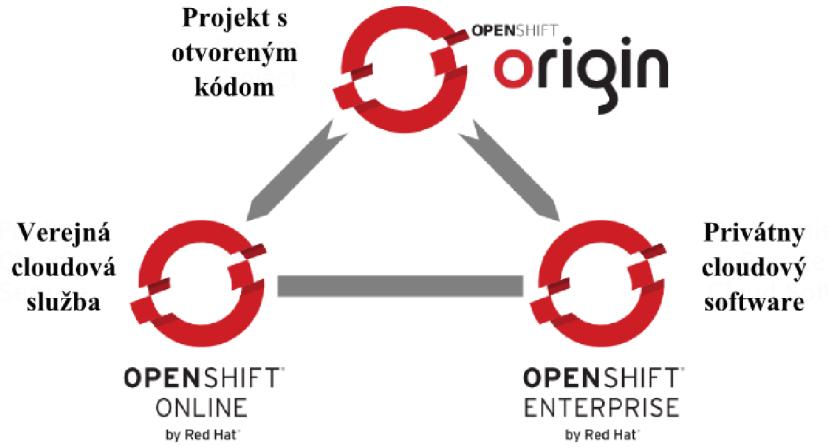
- **OpenShift Online** – je verejná verzia, ktorá je nasadená spoločnosťou Redhat vo verejnom cloude a to konkrétne na platforme Amazon Web Services. Spravovanie a riadenie serverov

[2] http://git-scm.com/

infraštruktúry je zabezpečené spoločnosťou Redhat, tak ako aj podpora, ktorá sa ale líši v závislosti na zakúpenom pláne (viď. kapitola 5.4.3).

- **OpenShift Enterprise** – je komerčný produkt určený k nasadeniu na privátne cloudy. Snaží sa zabezpečiť vysokú stabilitu. Podpora je dodávaná spoločnosťou Redhat. Práve kvôli stabilite nie sú dostupné všetky najnovšie *cartridge* (viď. kapitola 5.4.1), ktoré môžeme nájsť vo verzii OpenShift Origin alebo Online.

Medzi jednotlivými aplikáciami je možné presúvať aplikácie v prípade, že je *cartridge* v danej verzii dostupný. Ďalšie kapitoly budú zamerané na popis OpenShift Online verzie.



Obrázok 5.4 Vzťah medzi jednotlivými verziami platformy OpenShift [31]

5.4.1 Architektúra

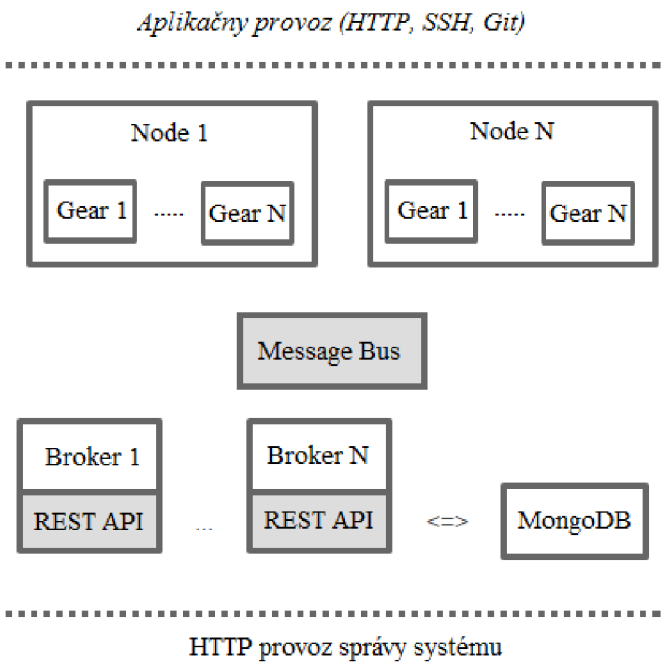
OpenShift (v dobe písania tejto diplomovej práce verzia 2) beží na operačnom systéme Red Hat Enterprise Linux. Na obrázku 5.5 sú zobrazené základné systémové komponenty systému OpenShift. Platforma je relatívne mladá a v súčasnej dobe ešte nie je zavedená slovenská ani česká terminológia väčšiny výrazov, preto budem používať pôvodné anglické pojmy.

Komunikáciu medzi *node* a *broker* servermi zabezpečuje *message bus* (zbernica správ), ktorá zaisťuje spoľahlivé doručenie správ. Node servery poskytujú virtualizované prostredie k nasadeniu aplikácií. Toto prostredie má pridelené určité hardvérové prostriedky a je spravované komponentou broker, ktorá je zodpovedná za prihlasovanie používateľov do systému, DNS (angl. *Domain Names Service*) alebo aplikačný stav. Broker je ovládateľný pomocou webovej konzoly, CLI (angl. *Command Line Interface*) nástrojov alebo REST rozhrania [32].

Ďalšou dôležitou komponentou je *gear*, ktorá predstavuje zabezpečený kontajner určený k behu aplikácií. Tento kontajner má pridelenú určitú množinu zdrojov node serveru ako CPU, RAM a úložisko. Aplikácia nemôže nikdy využívať viac zdrojov ako je alokovaných pre daný gear [32].

Poslednou komponentou je *cartridge*, ktorá je nasadzovaná do komponenty gear. Jedná sa o predpripravené zásuvné moduly s vlastnou adresárovou štruktúrou, behovým prostredím pre požadovaný programovací jazyk alebo pre poskytovanú službu. Cartrige môže byť webový rámec, databáza, monitorovacia služba alebo konektor k externým službám. Poznáme samostatný alebo vstavaný cartridge. Samostatný cartridge obsahuje programovacie jazyky alebo aplikačné servery slúžiace k obsluhe webového obsahu. Tento cartridge je postačujúci pre beh aplikácie. Vstavaný

cartridge poskytuje funkcionalitu k rozšíreniu aplikácie o služby, ktoré nedokážu bežať samostatne (napr. databázové služby) [30, 32].



Obrázok 5.5 Systémové komponenty cloudu OpenShift [31]

5.4.2 Dostupné technológie

K dispozícii je široká ponuka technológií dostupných vďaka komponente cartridge. Dostupnosť týchto technológií veľmi uľahčuje migráciu aplikácií práve do platformy OpenShift. Tabuľka 5.3 zobrazuje niektoré z dostupných technológii na OpenShift Online. V tabuľke je uvedená aj verzia aj typ technológie. Okrem predpripravených komponent cartridge je dostupný aj tzv. *diy* (angl. *Do It Yourself*) cartridge, ktorý poskytuje minimálnu stavebnú kostru. Rozšírenie a spravovanie takejto komponenty cartridge je ponechané na vývojárovi.

Technológia	Typ technológie	Verzia
Java	programovací jazyk	1.6, 1.7
PHP	programovací jazyk	5.3, 5.4
Python	programovací jazyk	2.6, 2.7. 3.3
Ruby	programovací jazyk	1.8, 1.9, 2.0
Perl	programovací jazyk	5.10
Tomcat	webový server	6, 7
JBoss AS/Wildfly	aplikačný server	7.1, 8.0
MongoDB	databáza	2.4
MySQL	databáza	5.1, 5.5
PostgreSQL	databáza	8.4, 9.2
Cron	nástroj	1.4
Jenkins	server určený k testovaniu	1.5

Tabuľka 5.3 Zoznam technológii dostupných na platforme OpenShift Online [33]

OpenShift Online ponúka vývojárovi obchod so službami iných spoločnosti, ktoré neponúka spoločnosť Redhat. Patria sem služby k správe emailov, správe cache pamäte, spoľahlivého doručenia správ, ďalej služby určené k záťažovému testovaniu aplikácie alebo poskytujúce iné typy databázových technológií.

Komponenty cartridge, ktoré dodávajú aplikačný server Wildfly alebo webový server Tomcat, zaručujú plnú podporu JRE a rámcov prostredia Java EE. Vývoj aplikácie preto môže prebiehať na lokálnych inštanciách serverov a až pri nasadení sa využije cloudová platforma. Vďaka širokej ponuke podporovaných technológií by prenesenie existujúcej aplikácie nemalo spôsobiť veľké komplikácie v porovnaní s platformou GAE.

5.4.3 Poplatky a limity

V platforme OpenShift online sú vyvinuté štyri typy komponenty gear, ktoré poskytujú rozdielnu úroveň výkonu a na základe toho sú aj rôzne spoplatnené. Tieto typy sú zobrazené v tabuľke 5.4. Tabuľka obsahuje množstvo alokovaných zdrojov, dostupnosť v regióne a ich cenu účtovanú za hodinu. OpenShift Online poskytuje bezplatný, bronzový a strieborný mesačný plán. Všetky plány v sebe zahŕňajú tri malé bezplatné typy komponenty gear, ktoré sú pozastavené v prípade 24 hodinovej nečinnosti. Ostatné tri typy sú dostupné len pre bronzový a strieborný plán, pričom k ich pozastaveniu nedochádza. Strieborný plán obsahuje oproti ostatným aj podporu od spoločnosti Redhat a 6GB úložisko pre každý typ.

Gear	CPU	RAM	Úložisko	Región	Cena
Malý	1x	512MB	1GB	USA	0.02$/hodina
Malý s vyšším CPU	2x	512MB	1GB	USA/Európa	0.025$/hodina
Stredný	2x	1GB	1GB	USA/Európa	0.04$/hodina
Veľký	4x	2GB	1GB	USA/Európa	0.08$/hodina

Tabuľka 5.4 Typy komponent gear v OpenShift Online

6 Analýza a špecifikácia

V tejto kapitole si predstavíme systém, ktorý je predmetom tejto diplomovej práce a ďalšie podobné existujúce aplikácie. Náš systém si neformálne špecifikujeme a jeho požiadavky zhrnieme do diagramu prípadov použitia.

6.1 Existujúce aplikácie

Z desktopových a webových aplikácií pracujúcich s formátom GPX sme vybrali dve, ktorým by sa vyvíjaná aplikácia mohla priblížiť. Obe aplikácie podporujú vizualizáciu GPX dát a generovanie doplňujúcich informácií. Prvou je desktopová aplikácia MyTourbook a druhou je webová aplikácia uTrack.

6.1.1 MyTourbook

MyTourbook je bezplatná desktopová aplikácia pre vizualizáciu, analýzu a spravovanie trás zaznamenaných pomocou GPS zariadení. Poskytuje veľké množstvo rôznych funkcií.

V prípade importu trás aplikácia podporuje veľké množstvo formátov získaných zo zariadení pre sledovanie a navigáciu. Medzi najznámejšie formáty patria GPX, TCX alebo CSV. Aplikácia dokáže zároveň exportovať údaje do GPX alebo TCX formátu.

Aplikácia štandardne používa OSM mapové podklady. Tieto podklady je možné kedykoľvek vymeniť za iné mapové podklady. Mapové podklady je tiež možné stiahnuť lokálne a používať aplikáciu bez pripojenia na internet. Nedávno začala aplikácia experimentálne podporovať aj 3D mapové podklady.

MyTourBook ponúka nástroje pre správu trás. V jednotlivých trasách je možné modifikovať geografické dáta, časové údaje alebo všeobecné informácie trasy. K zvoleným bodom trasy je možné priradiť obrázky. Podporované je aj spájanie viacerých trás do jednej trasy. Pokročilejšia funkcia tejto aplikácie dokáže na základe rozličných údajov rozdeliť trasu na menšie časti.

Ďalšou funkciou je vizualizácia trás na mape. Ukážka takejto vizualizácie trasy je zobrazená na obrázku 6.1. Trasa je do mapy vykreslená rôznymi farbami, ktoré predstavujú nadmorskú výšku na jednotlivých úsekoch trasy. Na trase sú taktiež vyznačené štartovacie body spolu s ďalšími doplnkovými informáciami v podobe značiek. Výškový profil pre trasu sa zobrazuje pod mapou.

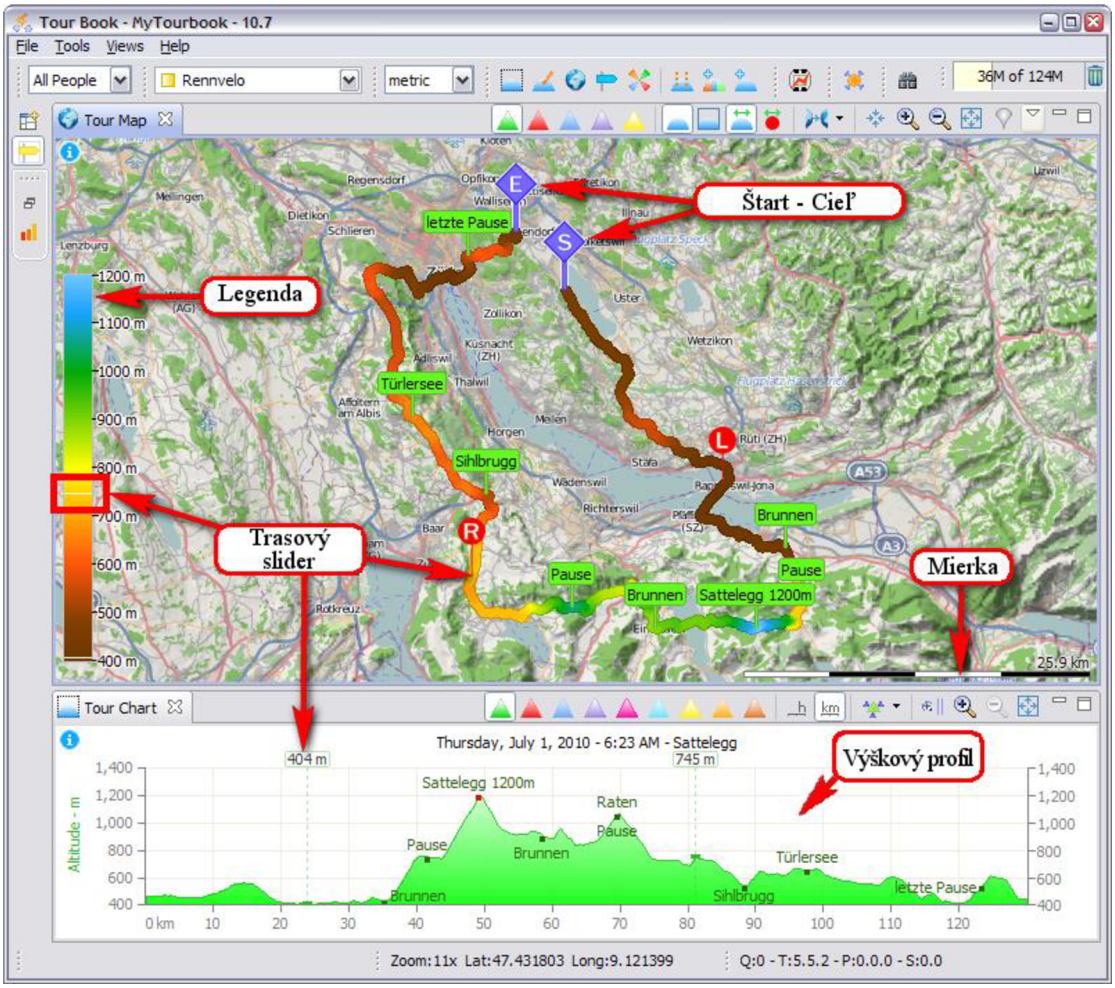
Analytické a štatistické funkcie patria medzi najzaujímavejšie dáta, ktoré je možné získavať z prejdených trás. MyTourbook vykresľuje tieto dáta v prehľadných grafoch. Aplikácia zobrazuje výškový profil trasy, počet najazdených kilometrov, rýchlosť a čas trasy. Sumarizovať tieto údaje môžeme zo všetkých trás v rôznych časových intervaloch, ktoré aplikácia povoľuje (deň, týždeň, mesiac, rok).

6.1.2 UTrack

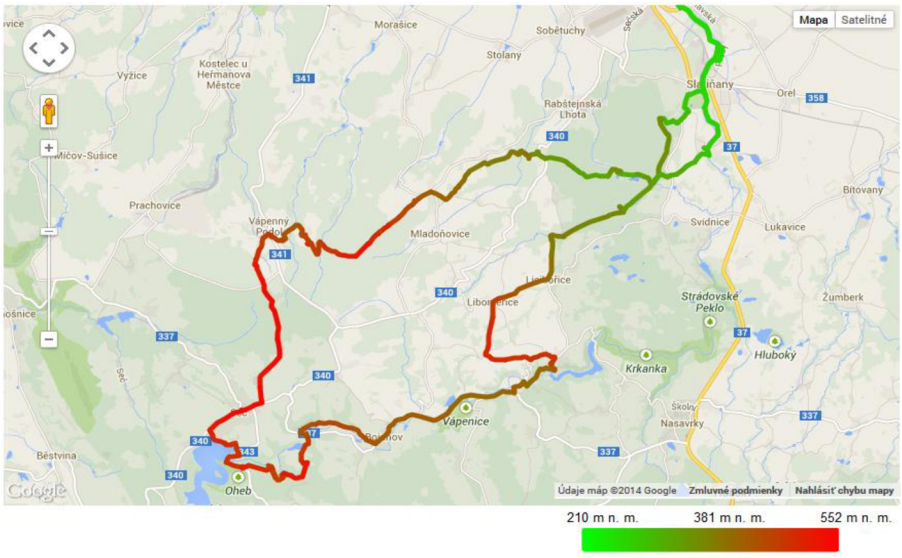
UTrack je bezplatná webová aplikácia slúžiaca na zobrazenie trasy na mape a generovanie rôznych štatistických údajov z prejdených trás uložených vo formáte GPX. Štatistické údaje je možné exportovať do PDF súboru.

Aplikácia na jednej webovej stránke zobrazuje vizualizáciu prejdenej trasy na mapovom podklade od Google Maps a štatistické informácie o výške, rýchlosti, čase a dĺžke trasy spolu

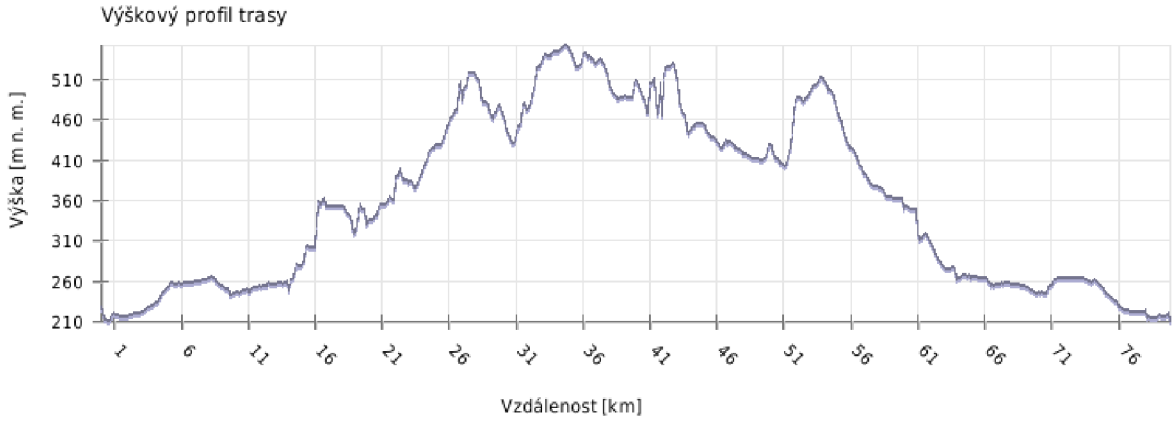
s výškovými, rýchlostnými, vzdialenostnými a časovými profilmi (viď. obrázok 6.3). Aplikácia zahŕňa farebné rozlíšenie nadmorskej výšky zobrazovanej trasy (viď. obrázok 6.2). Podrobné štatistické informácie získané analýzou prejdenej trasy sú zhrnuté v tabuľke 6.1.



Obrázok 6.1 Ukážka vizualizácie trasy v aplikácii MyTourbook



Obrázok 6.2 Ukážka vizualizácie trasy v aplikácii uTrack



Obrázok 6.3 Ukážka výškového profilu trasy použitého v aplikácii uTrack

Výškové štatistiky	Rýchlostné štatistiky	Časové štatistiky	Vzdialenostné štatistiky
Minimálna výška	Minimálna rýchlosť	Dátum trasy	Celková plošná vzdialenosť
Maximálna výška	Maximálna rýchlosť	Čas na začiatku trasy	Celková skutočná vzdialenosť
Priemerná výška	Priemerná rýchlosť stúpania	Čas na konci trasy	Vzdialenosť stúpania
Celkové stúpanie	Priemerná rýchlosť klesania	Celková doba trasy	Vzdialenosť klesania
Celkové klesanie	Priemerná rýchlosť na rovine	Celková doba stúpania	Vzdialenosť na rovine roviny
Počiatočná výška	Priemerná rýchlosť	Celková doba klesania	
Koncová výška		Celková doba roviny	

Tabuľka 6.1 Prehľadová tabuľka štatistických informácií o trase získavaných aplikáciou uTrack

6.2 Neformálna špecifikácia

Neformálna špecifikácia bola zostavená na základe komunikácie s vedúcim diplomovej práce, zadania práce a existujúcich aplikácií. Z funkcií existujúcich aplikácií nás oslovila hlavne možnosť získania rôznych štatistických informácií o trase, na ktoré sme sa zamerali aj pri tvorbe vyvíjanej aplikácie.

Systém bude z pohľadu používateľa umožňovať registráciu a prihlásenie do systému. Prihlásený používateľ si bude môcť spravovať svoj vlastný profil. V profile bude možnosť zadania štandardných atribútov ako meno, priezvisko, prihlasovacie meno, emailová adresa a fotografia. Hlavnou funkciou dostupnou pre prihláseného používateľa bude import trasy z formátu GPX. Pri importe trasy bude možné ukladať aj doplňujúce údaje ako názov, popis, viditeľnosť trasy pre iných používateľov a značky – tagy. Správa trás bude zahŕňať editovanie údajov a mazanie trás.

K jednotlivým trasám si používateľ bude môcť priradiť body záujmu s názvom, popisom, polohou a fotografiou. Body bude možné mazať. Trasy bude možné zobrazovať na mapových podkladoch služby Mapy.cz spolu s vypočítanými štatistikami a výškovým profilom. Podporovaná bude aj zmena mapových podkladov v rámci Mapy.cz. Tvorba štatistík bude zameraná na

zaznamenávanie výškových, rýchlostných, časových a vzdialenostných údajov, ktoré sú špecifikované v tabuľke 6.2.

Systém bude používateľovi okrem zobrazenia vlastných celkových štatistik umožňovať aj zobrazenie vlastných mesačných štatistík. Vlastné a verejné trasy bude možné vyhľadávať na základe tagov, časového úseku a dĺžky trasy.

Systém bude implementovaný pomocou programovacieho jazyka Java a jeho rámcov ako webová aplikácia s prívetivým a prehľadným grafickým užívateľským rozhraním. Nasadený a prevádzkovaný bude buď v bezplatnej verzii cloudovej platformy GAE alebo OpenShift Online.

Výškové štatistiky	Rýchlostné štatistiky	Časové štatistiky	Vzdialenostné štatistiky
Minimálna výška	Minimálna rýchlosť	Celková doba trasy	Celková skutočná vzdialenosť
Maximálna výška	Maximálna rýchlosť	Celková doba trasy v pohybe	Celková vzdialenosť stúpania
Celkové stúpanie	Priemerná rýchlosť	Celková doba trasy na rovine	Celková vzdialenosť klesania
Celkové klesanie		Celková doba stúpania	Celková vzdialenosť na rovine
		Celková doba klesania	

Tabuľka 6.2 Špecifikácia podporovaných štatistických údajov o trase

6.3 Analýza požiadaviek

Požiadavky na systém a jeho aktérov vyplývajúce z neformálnej špecifikácie sú zobrazené v diagrame prípadov použitia na obrázku 6.4. V systéme sa budú nachádzať nasledujúci aktéri:

- **Používateľ** – reprezentuje aktéra, ktorý má možnosť registrácie a prihlásenia.
- **Prihlásený používateľ** – reprezentuje hlavného aktéra, ktorý je prihlásený do systému. Dokáže spravovať svoje trasy, vlastný účet, prezerať si trasy, štatistiky, celkové a mesačné štatistiky, vyhľadávať trasy na základe rôznych kritérií a pridávať body záujmu k trasám.

Obrázok 6.4 Diagram prípadov použitia navrhovanej aplikácie

7 Návrh

Táto kapitola približuje obecný návrh architektúry systému, výber platformy a technológií, návrh jednotlivých častí systému a vysvetľuje ich úlohy v systéme.

7.1 Architektúra systému

Z požiadaviek na vyvíjaný systém vyplýva, že systém musí byť naprogramovaný ako webová aplikácia. Pri návrhu webovej aplikácie sa dnes využívajú dva základne prístupy a to architektúra s tenkým (angl. *thin*) alebo hrubým (angl. *thick*) klientom. Architektúra s tenkým klientom predstavuje zaužívaný klasický prístup, pri ktorom je celý stav aplikácie uložený na serveri a webové stránky sú renderované na strane serveru pri každej požiadavke.

Architektúra s hrubým klientom predstavuje moderný prístup, pri ktorom sa väčšina činností, ktoré sú v predchádzajúcom prístupe vykonávané na strane serveru, presúva na stranu klienta. Tento typ architektúry dekomponuje systém na dve samostatné časti.

Prvou časťou je webová služba, ktorá obsahuje biznis logiku, spravuje dáta uložené v databáze a poskytuje API rozhranie na ich získanie a modifikáciu. Na implementáciu takýchto webových služieb sa dnes veľmi často využíva architektúra REST kvôli jej jednoduchosti.

Druhú časť systému predstavuje hrubý klient, ktorý udržuje stav aplikácie, uchováva logiku užívateľského rozhrania a komunikuje s API webovej služby. K implementácii klienta vo webových prehliadačoch sa dnes používajú rámce, ktoré využívajú MVC (angl. *Model View Controller*) architektúru. Medzi takéto rámce patrí rámec AngularJS.

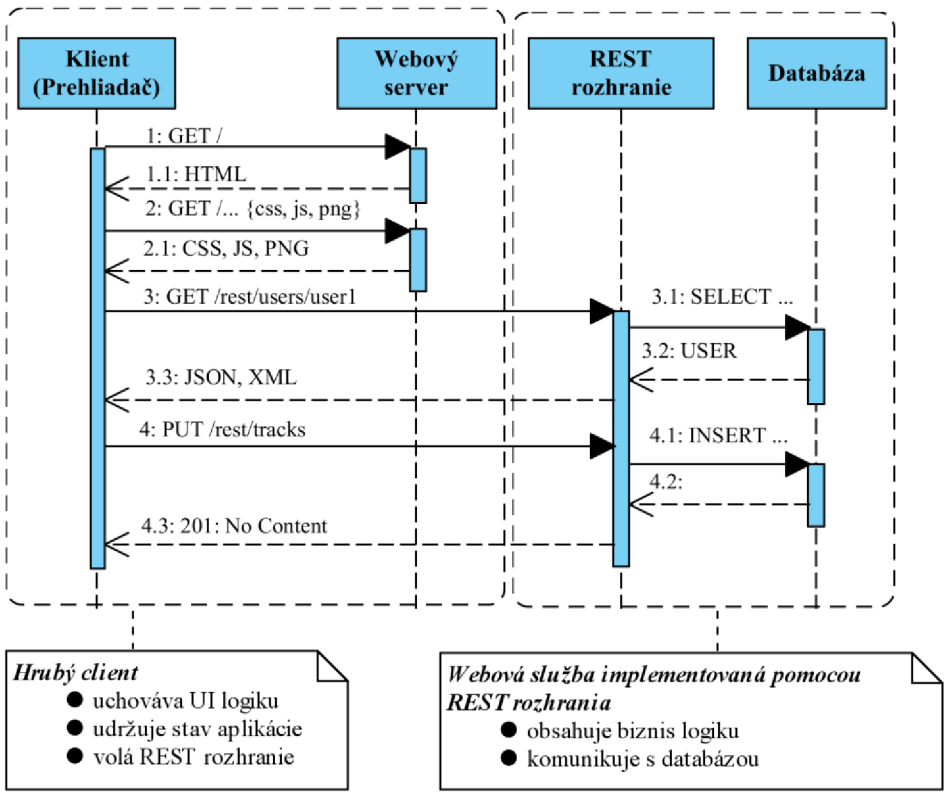
Pri tvorbe aplikácie sme sa rozhodli pre druhý prístup, pretože znižuje počet webových požiadaviek a tým aj spotrebu zdrojov cloudovej platformy. Architektúra navrhnutého systému spolu s jednoduchým znázornením komunikácie medzi jednotlivými časťami systému je znázornená na obrázku 7.1. Klient najskôr požiada webový server o zdroje potrebné k zobrazeniu stránky. Po načítaní zdrojov už server komunikuje pomocou AJAX (angl. *Asynchronous JavaScript and XML*) technológie len s rozhraním služby. Zdroje klienta a služba môžu byť združené v jednom logickom celku, vďaka čomu môžu byť nasadené na jednom serveri.

Popis návrhu jednotlivých častí systému sa nachádza v nasledujúcich podkapitolách.

7.2 Výber platformy

Ďalšie podkapitoly návrhu sa zameriavajú už na špecifickú platformu, preto je na tomto mieste potrebné objasniť výber vhodnejšej platformy pre vyvíjaný systém.

Najideálnejším riešením by bolo, keby bol systém spustiteľný na oboch platformách popísaných v kapitole 5. Architektúru s hrubým klientom a REST rozhraním je možné zrealizovať na obidvoch platformách. K implementácií REST rozhraní by sme mohli voliť rámec JAX-RS *Jersey*, ktorý je v špecifickej verzii dostupný v oboch platformách. Hlavný problém nastáva pri návrhu štruktúry databázy a použití spoločného ORM rámca. Takýmto spoločným rámcom je JPA, ktorý avšak v GAE prostredí nepodporuje všetky typy mapovania vzťahov medzi entitami práve kvôli špecifikám bezplatnej databázy. Z uvedeného dôvodu, by preto bolo veľmi komplikované navrhnúť spoločnú štruktúru databázy pre obe platformy.



Obrázok 7.1 Sekvenčný diagram vzájomnej komunikácie jednotlivých častí navrhnutého systému

V prípade zvolenia bezplatnej verzie platformy GAE by sa k implementácii REST rozhrania mohla využiť služba Google Cloud Endpoints s ORM rámcom Objectify. Tým by sa však systém stal závislým na obmedzenom JRE, špecifickej službe a databáze, čo by v budúcnosti mohlo značne komplikovať prenos systému na inú platformu.

OpenShift Online poskytuje v bezplatnej verzii oproti GAE viac typov databázových technológií a štandardné webové a aplikačné servery, ktoré úplne podporujú Java EE rámce. Pre PostgreSQL cartridge je dostupné rozšírenie *PostGIS*, ktoré dodáva dátové typy na uloženie priestorových objektov a podporuje priestorové dotazy nad týmito objektmi. V prípade využitia tejto platformy, bude prenos systému na inú platformu menej komplikovaný.

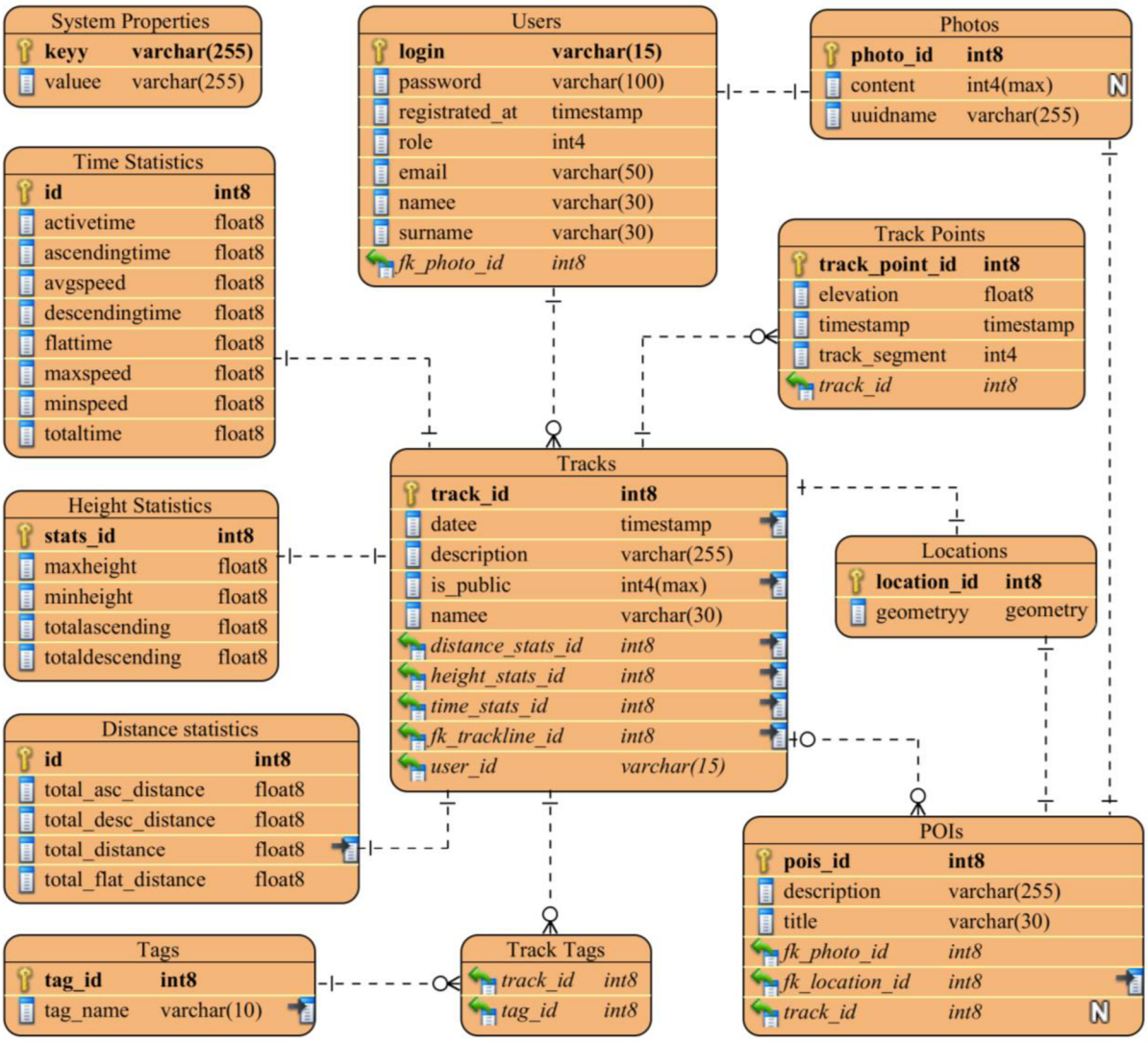
Na základe porovnania vyššie opísaných riešení bola vybratá platforma OpenShift Online. Z dostupných cartridgov sme zvolili databázu PostgreSQL s rozšírením PostGIS. Webová služba systému bude bežať na aplikačnom servery Wildfly kompatibilnom s Java EE 7 štandardom. Pre klienta aplikácie sme zvolili rámec AngularJS.

7.3 Návrh štruktúry dát

Navrhnutá štruktúra dát pre PostgreSQL databázu je dostupná vo forme entitno-relačného (ER) diagramu na obrázku 7.2. Diagram je odvodený z neformálnej špecifikácie a GPX formátu. Všetky entity okrem *Users* a *System Properties* obsahujú primárny číselný kľúč.

Pre správu používateľských účtov je navrhnutá entita *Users,* ktorej primárny kľúč je tvorený prihlasovacím menom. Entita obsahuje okrem kontaktných údajov používateľa, popísaných v neformálnej špecifikácii, aj rolu a čas registrácie. Každý používateľ má práve jednu fotografiu reprezentovanú entitou *Photo,* ktorej atribút *content* obsahuje fotografiu uloženú v databáze a atribút *uuidname* je univerzálny jedinečný identifikátor (UUID) generovaný systémom. UUID je pri prístupe

k fotografiám cez webovú službu použitý z bezpečnostných dôvodov. Jeho význam je vysvetlený v nasledujúcich kapitolách.



Obrázok 7.2 Štruktúra dát systému vo forme ER diagramu

Návrh štruktúry dát perzistentného nastavenia systému je reprezentovaný entitou *System Properties*, ktorá má atribúty kľúč a hodnota. Entita *Locations* je jediná entita s priestorovými dátami uloženými v atribúte *geometry*. Typ tohto atribútu je generický, čo znamená že je možné do neho uložiť akýkoľvek priestorový objekt (bod, lomenú čiaru, plochu alebo ich kolekcie).

Štruktúra dát ukladania trás z formátu GPX je mapovaná na entity *Tracks*, *Track Points* a *Locations*. Entita *Tracks* obsahuje atribúty názov, popis a dátum uloženia. Trasa v GPX môže mať niekoľko segmentov a každý segment má niekoľko bodov trasy. Body trasy sú preto reprezentované entitou *Track Points*, ktorá má atribút pre výšku a časovú značku. Na ukladanie segmentov trasy je vytvorený atribút *track_segment*, ktorý udáva poradové číslo v pôvodnej GPX trase. Vzťah medzi entitou *Track* a *Track Points* je modelovaný ako 1:N. Súradnice pre zemepisnú šírku a dĺžku sú spojené do jedného priestorového objektu reprezentovaného ako kolekcia lomených čiar (angl. *MultiLineString*), ktorý je uložený v entite *Locations*. Jedna entita *Tracks* má práve jednu kolekciu lomených čiar, a preto vzťah medzi entitou *Tracks* a *Locations* je modelovaný ako 1:1.

Entita *POIs* s atribútmi názov a popis predstavuje bod záujmu. Každý bod záujmu obsahuje práve jednu fotografiu a polohu. Preto má entita *POIs* vzťahy 1:1 s entitami *Photos* a *Locations*. Poloha bodu záujmu je reprezentovaná priestorovým objektom bod.

Tagy trás sú modelované v entite *Tags*, ktorá obsahuje atribút názov. Trasa môže mať priradených viac tagov, a preto je vzťah medzi týmito entitami modelovaný ako N:N s väzbovou entitou *Track Tags*.

Entity *Time Statistics, Distance Statisics, Height Statisics* modelujú všetky štatistické údaje o jednej trase. Preto medzi nimi a entitou *Track* existuje vzťah 1:1. Jednotlivé atribúty entít odpovedajú údajom uvedeným v tabuľke 6.2.

7.4 Návrh webovej služby

Webová služba systému spravuje dáta uložené v databáze a vo forme objektov ich poskytuje na rozhranie, ktoré je implementované architektúrou REST. Pre prenos štruktúrovaných objektov medzi webovou službou a klientom systému je zvolený formát JSON (angl. *JavaScript Object Notation*).

7.4.1 Architektúra

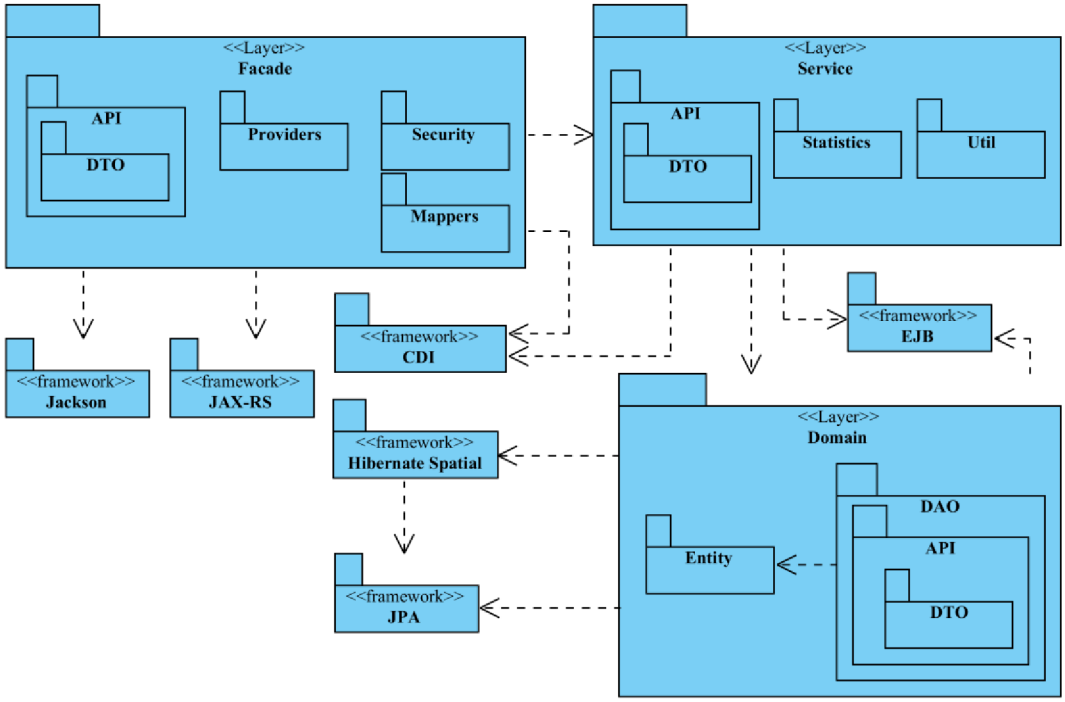
Architektúra webovej služby systému je dekomponovaná do troch spolupracujúcich vrstiev, ktoré sú zobrazené v diagrame balíčkov na obrázku 7.3. Návrh využíva princíp programovania do rozhraní, preto má každá vrstva balík *api*, ktorý obsahuje všetky rozhrania danej vrstvy využívané inými vrstvami. Okrem toho má tento balík ďalší balík *dto*, ktorý obsahuje triedy využívajúce návrhový vzor *Data Transfer Object* [34]. Jedná sa o objekty, ktoré v sebe zapuzdrujú dáta slúžiace k výmene medzi jednotlivými vrstvami. Všetky triedy v balíčkoch sú navrhnuté pomocou *objektovo-orientovanej* paradigmy. Medzi základné vrstvy v systéme patrí:

- **Doménová vrstva** – získava/modifikuje požadované dáta z/v databáze. Skladá sa z balíkov *entity* a *dto*. Balík *entity* realizuje mapovanie tabuliek databázy na triedy entít pomocou ORM techniky. Balík *dao*, pomenovaný podľa návrhového vzoru *Data Access Object* [35], obsahuje triedy sprostredkúvajúce komunikáciu s databázou na úrovni objektov.

- **Servisná vrstva** – obsahuje triedy implementujúce biznis logiku, pomocou ktorej sú volané metódy tried z doménovej vrstvy. V tejto vrstve je riešená aj správa transakcií. Vrstva obsahuje balík *statistics*, ktorý zapuzdruje triedy slúžiace na výpočet štatistických údajov z trasy. Vrstva má v balíku *api* ďalší balík *exceptions*, ktorý obsahuje výnimky vyhodené z tejto vrstvy.

- **Fasádna vrstva** – zabezpečuje adresovanie REST zdrojov, prijímanie požiadaviek na zdroje, spracovanie prijatých a odoslaných dát vrátane ich validácie a transformácie na objekty z balíka *dto*, odoprenie prístupu v prípade neprihláseného používateľa, volanie obslužnej funkcie zo servisnej vrstvy a odosielanie odpovedí spolu s dátami a návratovým kódom. V balíku *security* sa nachádzajú triedy riešiace bezpečnosť webovej služby.

Implementácie jednotlivých vrstiev používajú štandardné rámce z platformy Java EE dostupné na aplikačnom serveri Wildfly. ORM technika je zabezpečená vďaka rámcu JPA. Na tvorbu komunikačného rozhrania architektúrou REST je využitý rámec JAX-RS. Tieto rámce spolu s ďalšími použitými rámcami, ich významom a spôsobom použitia si predstavíme v kapitole zaoberajúcej sa implementáciou.



Obrázok 7.3 Návrhový diagram balíkov webovej služby systému

7.4.2 REST zdroje

Architektonický štýl REST predstavil Roy Fielding vo svojej dizertačnej práci [36]. REST je dátovo orientovaná architektúra rozhrania systému, ktorá poskytuje jednotný prístup k svojim zdrojom (angl. *resources*) adresovateľných pomocou unikátneho identifikátora URI (angl. *Uniform Resource Identifier*). Prístup k zdrojom je možný cez HTTP protokol a jeho metódy, ktoré sú známe ako CRUD:

- **Create** (vytvor) – vytvorenie nového zdroja pomocou metódy POST protokolu HTTP.
- **Retrieve** (získaj) – získanie zdroja pomocou metódy GET protokolu HTTP.
- **Update** (uprav) – úprava existujúceho zdroja metódou PUT protokolu HTTP.
- **Delete** (vymaž) – vymazanie existujúceho zdroja metódou DELETE protokolu HTTP.

V tabuľkách 7.1, 7.2, 7.3 a 7.4 sú popísané všetky REST zdroje webovej služby systému. Návrh zdrojov vychádza z identifikovaných entít ER diagramu. V jednotlivých tabuľkách je pre každý zdroj uvedená HTTP metóda, URI, parametre (pomocou *:uuid*), stručný popis a typ prístupu, ktorý udáva či používateľ musí byť pri prístupe k zdroju prihlásený alebo nie.

METÓDA	URI	PRÍSTUP
GET	*/rest/photos/:uuid*	Voľný
Vracia fotografiu na základe UUID.		

Tabuľka 7.1 REST zdroje určené k získaniu fotografií

METÓDA	URI	PRÍSTUP
POST	*/rest/users*	Voľný
Vytvorí nového používateľa. Slúži pre registráciu nového používateľa.		
GET	*/rest/users/:userId*	Prihlásený používateľ
Vráti detail profilu používateľa.		
PUT	*/rest/users/:userId*	Prihlásený používateľ
Upraví profil používateľa.		
PUT	*/rest/users/:userId/password*	Prihlásený používateľ
Upraví heslo špecifikovaného používateľa.		
PUT	*/rest/users/:userId/photo*	Prihlásený používateľ
Upraví alebo vytvorí fotografiu pre špecifikovaného používateľa.		
POST	*/rest/users/:userId/tracks*	Prihlásený používateľ
Vytvorí novú trasu pre špecifikovaného používateľa.		
GET	*/rest/users/:userId/tracks* Parametre: *fromDate, toDate, fromDistance, toDistance, tags, offset, limit*	Prihlásený používateľ
Vráti stránkovaný zoznam trás špecifikovaného používateľa na základe parametrov.		
GET	*/rest/users/:userId/tracks/:trackId*	Prihlásený používateľ
Vráti detail trasy špecifikovaného používateľa spolu so štatistikami a zoznamom bodov záujmu.		
PUT	*/rest/users/:userId/tracks/:trackId*	Prihlásený používateľ
Upraví údaje trasy.		
DELETE	*/rest/users/:userId/tracks/:trackId*	Prihlásený používateľ
Vymaže trasu špecifikovaného používateľa.		
POST	*/rest/users/:userId/tracks/:trackId/pois*	Prihlásený používateľ
Vytvorí nový bod záujmu pre špecifikovanú trasu.		
DELETE	*/rest/users/:userId/tracks/:trackId/pois/:poiId*	Prihlásený používateľ
Vymaže bod záujmu pre špecifikovanú trasu.		
GET	*/rest/users/:userId/statistics*	Prihlásený používateľ
Vráti celkové štatistiky trás špecifikovaného používateľa.		
GET	*/rest/users/:userId/statistics/:month*	Prihlásený používateľ
Vráti celkové štatistiky trás špecifikovaného používateľa za určitý mesiac.		

Tabuľka 7.2 REST zdroje určené k správe používateľov a ich trás

METÓDA	URI	PRÍSTUP
GET	*/rest/tracks/:id*	Prihlásený používateľ
Vráti detail trasy špecifikovaného používateľa spolu so štatistikami a zoznamom bodov záujmu.		
GET	*/rest/tracks* Parametre: *fromDate, toDate, fromDistance, toDistance, tags, offset, limit*	Prihlásený používateľ
Vracia stránkovaný zoznam zdieľaných trás na základe parametrov.		

Tabuľka 7.3 REST zdroje určené k získavaniu zdieľaných trás

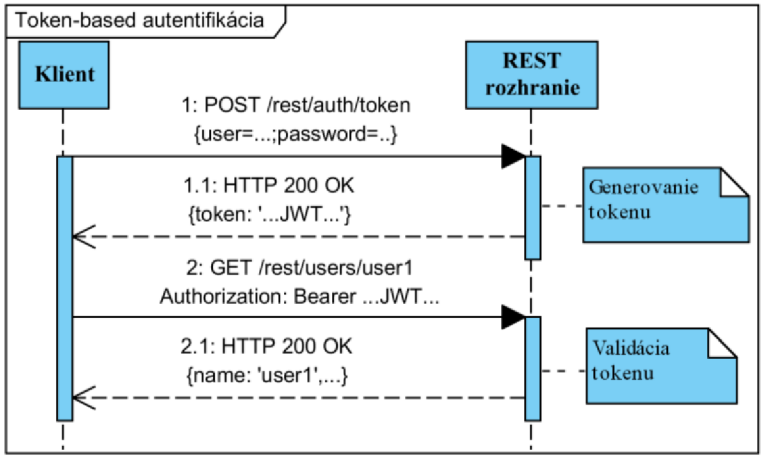
METÓDA	URI	PRÍSTUP
POST	/rest/auth/token	Voľný
Vracia prihlasovací token v prípade, ak sú zaslané prihlasovacie údaje valídne.		

Tabuľka 7.4 REST zdroje určené k autentifikácii používateľa

7.4.3 Zabezpečenie

K zabezpečeniu webovej služby je navrhnutá autentifikácia založená na tokene (angl. *token-based authentification*). Táto autentifikácia nepotrebuje uchovávať žiadny stav na strane serveru tak, ako je to v prípade tradičnej autentifikácie založenej na *cookies*.

Na obrázku 7.4 sa nachádza sekvenčný diagram zobrazujúci priebeh autentifikácie používateľa a získanie autentifikačného tokenu slúžiaceho na prístup k zdrojom webovej služby. Klient aplikácie najskôr zašle požiadavku spolu s prihlasovacími údajmi na zdroj, ktorý v prípade validity údajov vygeneruje a zašle token späť. Token sa následne zasiela v *authorization* hlavičke požiadavky na zabezpečený zdroj. Pri každej takejto požiadavke je na strane serveru overovaná validita a identita tokenu.



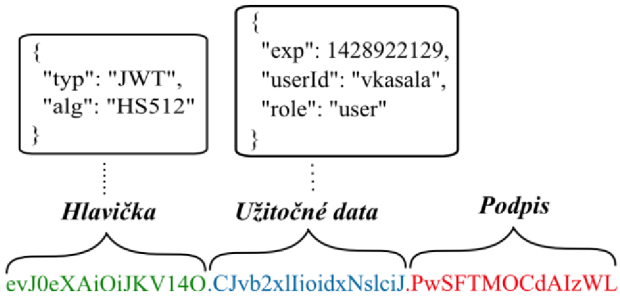
Obrázok 7.4 Sekvenčný diagram navrhovaného autentifikáčného mechanizmu webovej služby

Token je reprezentovaný pomocou *JSON Web Token* (JWT) [37], ktorého použitie je znázornené na obrázku 7.5. Skladá sa z troch častí oddelených bodkou, pričom každá časť je zakódovaná *Base64*[3] kódovaním. K týmto častiam patrí:

- **Hlavička** – nesie informácie o type použitého hešovacieho algoritmu,
- **Užitočné dáta** (angl. *payload*) – obsahuje užitočné dáta, ktoré chceme prenášať a ďalšie informácie o tokene. JWT špecifikácia definuje štandardné typy hodnôt (angl. *claims*) užitočných dát. Patrí sem napr. naším systémom použitý claim *exp*, ktorý udáva platnosť JWT tokenu.
- **Podpis** – je vytvorený hešovacou funkciou vyžívajúcou súkromný kľúč. Funkcii sú odovzdané spojené reťazce hlavičky a užitočných dát.

[3] Je kódovanie, ktoré prevádza binárne dáta na postupnosti ASCII znakov.

Webová služba prenáša v časti užitočných dát JWT tokenu jeho expiračný čas, rolu a prihlasovacie meno autentifikovaného používateľa. V záujme zabezpečenia dôvernosti komunikácie a odolnosti systému pred neoprávnením prístupom a získaním tokenu zo strany akékoľvek útočníka, by bolo vhodné pri komunikácii použiť šifrovaný protokol HTTPS (angl. *HyperText Transfer Protocol Secure*). V tejto práci tento protokol nebol použitý, pretože to nebolo stanovené ako cieľ tejto diplomovej práce.



Obrázok 7.5 Ukážka formátu JSON Web Token v systéme

7.5 Návrh užívateľského rozhrania webového klienta

Klient systému komunikuje s webovou službou za účelom získavania dát zobrazovaných vo webovom prehliadači. K implementácii klienta je zvolený rámec AngularJS a pre prácu s mapami je vybraná knižnica OpenLayers, ktorá podporuje veľké množstvo formátov slúžiacich na zobrazovanie priestorových dát. OpenLayers neposkytuje integráciu s mapovými podkladmi služby Mapy.cz, a preto je potrebné túto integráciu vyvinúť. Klient komunikuje s webovou službou pomocou JSON formátu z toho dôvodu je pre reprezentáciu priestorových dát vybraný formát GeoJSON, ktorý je podmnožinou JSON formátu.

Architektúru vrstiev webového klienta si popíšeme v implementácii, kde bude predstavený rámec AngularJS. V nasledujúcej časti tejto kapitoly sa zameriame na návrh užívateľského rozhrania klienta a jeho obrazoviek.

Užívateľské rozhranie klienta systému je inšpirované použitou šablónou *AdminLTE*[4]. Šablóna je responzívna, čo znamená, že ju je možné zobraziť aj na mobilných zariadeniach. Požiadavka na responzívnosť však nie je vyžadovaná zadaním práce a preto sa ňou v ďalšej časti nezaoberáme.

Každá obrazovka sa skladá z navigačného bočného menu, hlavného panelu, vyskakovacieho (pop-up) používateľského panelu a informačného okna. Bočné menu obsahuje položky určené k navigácii medzi jednotlivými obrazovkami systému a informačné okno zobrazuje správy o výsledkoch používateľských akcií. Používateľský panel slúži k zobrazeniu rýchleho náhľadu údajov prihláseného používateľa. Hlavný panel zobrazuje vždy iný obsah v závislosti na obrazovke. Dominantným prvkom hlavného panelu je vo väčšine obrazoviek mapa, ktorá je obsluhovaná ovládacími prvkami. Medzi tieto ovládacie prvky patrí okno k ovládaniu priblíženia/oddialenia mapy a k výberu typu mapových podkladov. Posledným prvkom, ktorý sa nachádza na každej obrazovke je tlačidlo slúžiace na ukrytie/zobrazenie bočného menu. Všetky popísané prvky rozhrania môžeme vidieť na obrázku 7.7.

[4] https://almsaeedstudio.com/AdminLTE

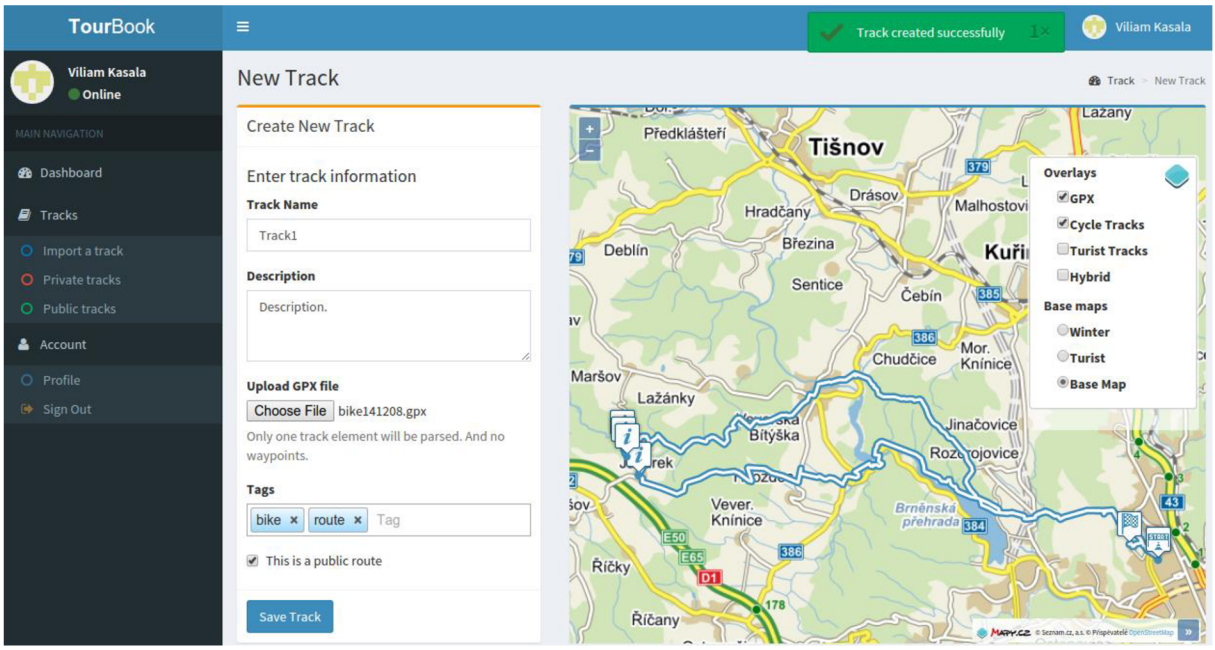
Klient systému je rozdelený na niekoľko hlavných obrazoviek, ktoré sa líšia obsahom hlavného panelu. Niektoré obrazovky využívajú dialógové okná k rozšíreniu funkcionality obrazovky. Medzi navrhnuté obrazovky klienta systému patrí:

- **Prihlasovacia obrazovka** – slúži k prihlasovaniu používateľov do systému. Obsahuje formulár pre vyplnenie prihlasovacích údajov. Po úspešnom prihlásení do systému je používateľ presmerovaný na svoju „Dashboard“ obrazovku.

- **Registračná obrazovka** – slúži k registrácii nových používateľov do systému. Obsahuje formulár na registráciu používateľa. Po úspešnej registrácii je používateľ presmerovaný na prihlasovaciu obrazovku.

- **„Dashboard” obrazovka** – slúži k zobrazeniu štatistík používateľa. Obsahuje panely slúžiace na zobrazenie celkových štatistík a mesačných štatistík. Panel mesačných štatistík má formulár, ktorým sa ovláda zobrazenie štatistík.

- **Obrazovka detailu používateľa** – slúži k správe vlastných údajov používateľa. Obsahuje panel k zobrazeniu údajov používateľa a fotografie. Súčasťou obrazovky sú aj tlačidlá vyvolávajúce dialógové okná na zmenu hesla, fotografie a údajov používateľov.



Obrázok 7.6 Obrazovka pridania novej trasy demonštrujúca navrhnuté užívateľské rozhranie

- **Obrazovka pridania novej trasy** – slúži k nahrávaniu novej trasy. Obsahuje mapu, ktorá zobrazuje trasu z vybraného GPX súboru a formulár s položkami pre zadanie názvu, stručného popisu, viditeľnosti, značiek a GPX súboru trasy. Po úspešnom uložení trasy je používateľ presmerovaný na obrazovku detailu práve ukladanej trasy.

- **Obrazovka detailu trasy** – slúži k spravovaniu a zobrazeniu trasy. Obsahuje panely so základnými údajmi, štatistikami a výškovým profilom trasy. Na obrazovke je dostupná aj mapa zobrazujúca trasu a jej body záujmu. Panel so základnými údajmi má tlačidlá

k vyvolaniu dialógových okien slúžiacich k editácii a mazaniu základných údajov a k vytváraniu a mazaniu bodov záujmu.

- **Obrazovka vyhľadania trás** – slúži na vyhľadávanie trás. Obsahuje formulár k zadaniu vyhľadávacích kritérií, panel na zobrazenie zoznamu nájdených trás a mapu zobrazujúcu počiatočné body trasy. Formulár má prvky slúžiace k vyhľadávaniu na základe tagov, rozsahu dátumov, rozsahu dĺžky trasy a typu trasy (zdieľaná/súkromná). Zoznam nájdených trás obsahuje tlačidlo, ktoré presmeruje používateľa na obrazovku detailu vybranej trasy. Pri pohybe kurzoru myši nad jednou trasou zo zoznamu sa na mape zobrazí práve táto trasa.

8 Implementácia

Implementácia webového systému pre správu GPS dát nadväzuje na kapitolu 7, ktorá dekomponuje systém na webovú službu a webového klienta a popisuje ich základnú funkcionalitu a správanie. V tejto kapitole sa zameriame na implementačné detaily a použité technológie jednotlivých častí.

8.1 Použité technológie

Táto sekcia má za cieľ vysvetliť technológie, ktoré sú použité k implementácii systému. Prezentované sú len kľúčové prvky týchto technológií z dôvodu rozsahového obmedzenia diplomovej práce. V prvej časti je predstavené databázové priestorové rozšírenie PostGIS. V ďalšej časti je popísaná platforma Java EE a jej jednotlivé technológie použité k implementácii webovej služby systému. Nakoniec je prezentovaný rámec AngularJS, ktorým je klient systému implementovaný.

8.1.1 PostGIS

PostGIS je rozšírenie určené pre objektovo-relačný databázový systém PostgreSQL, ktorý umožňuje prácu s priestorovými dátami. Do PostgreSQL je pridaná podpora pre priestorové dátové typy, priestorové indexy a priestorové funkcie. Rozšírenie implementuje geometrický objektový model OpenGIS konzorcia spolu so všetkými funkciami definovanými nad týmto modelom [38].

Pre stĺpec tabuľky sú dostupné priestorové dátové typy *GEOMETRY* a *GEOGRAPHY*. Jedná sa o typy generické, pretože môžu obsahovať všetky objekty definované z geometrického objektového modelu. Medzi základné geometrické objekty patrí napr. bod, krivka, lomená čiara a iné. Každý takýto stĺpec musí špecifikovať identifikátor súradnicového systému (SRID), v ktorom sú priestorové dáta reprezentované. Vďaka identifikátoru je možné transformovať dáta medzi rôznymi súradnicovými systémami.

8.1.2 Java EE 7

Java EE je súčasťou platformy Java určená pre vývoj a prevádzku webových a podnikových aplikácií, ktoré sú navrhnuté pomocou viacvrstvovej architektúry. Pre rôzne komponenty jednotlivých vrstiev aplikácie definuje platforma separátne API, ktoré sú popísané v špecifikáciách *Java Specification Request* (JSR).

Java EE 7 bola vydaná v roku 2013 a oproti verzií 6 poskytuje mnoho noviniek a zjednodušení vývojového procesu. Java EE 7 technológie použité na implementáciu webovej služby si bližšie opíšeme v nasledujúcom texte. Popis technológií vychádza z [39].

JPA

Java Persistence API 2.1, špecifikované v JSR 338, slúži k ORM mapovaniu tried entít na tabuľky relačnej databázy, k spravovaniu inštancií mapovaných tried entít, spravovaniu pripojenia k databáze a dotazovaniu sa nad databázou. Na mapovanie tried entít na tabuľky sa používa anotácia *@Entity* a na mapovanie atribútov entít na stĺpce tabuľky sa používajú anotácie *@Column, @Basic*. Trieda môže obsahovať len povolené typy atribútov, ak si s nimi nevystačíme, môžeme si vytvoriť vlastné mapovanie typu atribútu na stĺpec tabuľky. K vytvoreniu asociácií medzi entitami sa využívajú anotácie *@OneToOne*, *@OneToMany, @ManyToOne a @ManyToMany*.

Základné operácie s entitami zabezpečuje správca entít (trieda *EntityManager*), ktorý disponuje okrem iného operáciami uloženia, mazania a aktualizácie entity. K získaniu správcu manažéra sa používa anotácia *@PersitenceContext*. K dotazovaniu nad databázou je k dispozícií JPQL (angl. *Java Persistence Query Language*) jazyk alebo *Criteria API*. Obe sa dotazujú nad objektovým modelom, čím sa zvyšuje prenositeľnosť na rôzne databázové systémy.

V súbore ***persitence.xml*** je definovaná konfigurácia databázy, s ktorou správca entít pracuje. Technológia JPA je len špecifikáciou, pričom každá špecifikácia má viac implementácií. Wildfly aplikačný server poskytuje Hibernate implementáciu rámca JPA.

Bean Validation

Bean Validation, špecifikované v JSR 349, zavádza štandardný rámec pre validáciu Java tried. Obsahuje sadu anotácií, ktoré definujú validačné pravidlá tried, metód a atribútov. K overovaniu splnenia všetkých validačných pravidiel dochádzka pomocou validátorov. Medzi štandardné anotácie patrí napr. *@Null, @Min, @Max,@Size, @Pattern* atď.

Najčastejšie sa Bean Validation využíva v spolupráci s rámcom JPA k zaisteniu integrity dát. Ďalším typickým použitím je využitie v JAX-RS rámci na zaistenie validity vstupných dát.

JAX-RS

Java API for RESTful Web Services 2.0, špecifikované v JSR 339, slúži k rýchlemu a jednoduchému vývoju webových služieb podľa REST architektúry pomocou štandardizovaných anotácií. Anotácia *@Path* umožňuje mapovanie relatívnej zdrojovej cesty na triedy alebo jej metódy. K mapovaniu štandardných metód protokolu HTTP slúžia anotácie *@GET, @POST, @PUT a @DELETE*. Časti zdrojovej cesty je možné mapovať na parametre metód pomocou anotácií *@PathParam*, *@QueryParam*. Na určenie typu reprezentácie a formátu vstupných/výstupných dát sa používajú anotácie *@Produces* a *@Consumes*. Riadenie a správa samotnej komunikácie prebieha v réžii JAX-RS.

Poslednou dôležitou anotáciou JAX-RS, ktorú si popíšeme, je anotácia *@Provider*. Anotácia slúži k registrácii mapovania výnimiek na výstupné dáta, nových poskytovateľov obsahu a autentifikačných, autorizačných a iných filtrov modifikujúcich vstupné/výstupné dáta. Poskytovatelia obsahu zabezpečujú mapovanie vstupných/výstupných dát na ich asociované Java typy. Wildfly aplikačný server poskytuje RestEasy implementáciu JAX-RS.

EJB

Enterprise Java Beans 3.2 (EJB), špecifikované v JSR 345, sa používa na vývoj a nasadenie distribuovaných komponentových aplikácií. EJB komponentami sa zvyčajne implementuje biznis (aplikačná) vrstva. Implementácia biznis logiky pozostáva z množiny EJB komponent, ktoré sú spravované EJB kontajnerom. Kontajner spravuje ich životný cyklus, zaisťuje automatické transakcie a bezpečnosť, riadi súbežný prístup a vkladá závislosti pomocou návrhového vzoru *Dependency Injection* (DI). Biznis logika sa vkladá do metód EJB komponent.

EJB komponenty sa delia na *Session beans* a *Message driven beans*. *Session* komponenty sa ďalej delia na stavové (anotácia *@Stateful*), bezstavové (anotácia *@Stateless*) a singleton (anotácia *@Singleton*). Stavové EJB komponenty udržujú stav v rámci sedenia a bezstavové sú vytvárané kontajnerom pri každej požiadavke. Singleton EJB komponenta je založený na návrhovom vzore jedináčik (angl. *Singleton*).

CDI

Context and Dependency Injection for the Java EE Platform 1.1, (CDI) špecifikované v JSR 346, definuje typovo bezpečný mechanizmus injekcie závislostí. CDI sa využíva na prepojenie jednotlivých vrstiev Java EE aplikácií. Zavádza typovo bezpečnú injekciu založenú na anotáciách a definuje nové rozsahy platností injektovaných CDI komponent (*dependent*, *request*, *session*, *application*, *conversation, transactional scope*). Injektovať môžeme okrem CDI komponent aj EJB komponenty (anotácia @*Inject*) a zdroje Java EE (napr. anotácia @*PeristenceContext*).

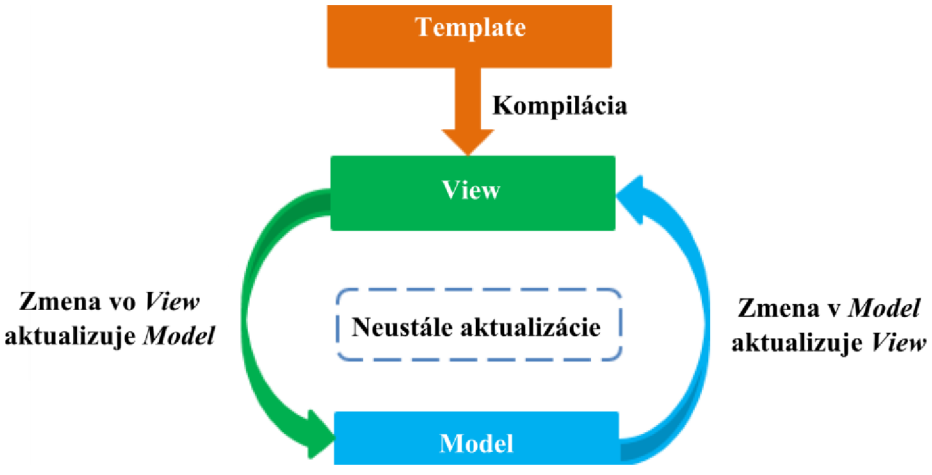
CDI komponentou môže byť akákoľvek POJO (angl. *Plain Old Java Object*) trieda, ktorá nie je abstraktná a finálna, neobsahuje finálne metódy a obsahuje bezparametrický konštruktor. CDI prichádza s ďalšími funkciami ako sú alternatívy, stereotypy, kvalifikátory injekcie, producery alebo systém udalostí. Wildfly aplikačný server poskytuje *Weld* implementáciu CDI.

8.1.3 AngularJS

AngularJS je webový aplikačný rámec jazyka JavaScript spravovaný firmou Google a skupinou vývojárov, ktorý sa snaží poskytnúť vývojárovi prostredie k budovaniu a testovaniu aplikácií bežiacich na jednej stránke (angl. *Single Page Applications,* SPA). Rámec využíva architektonický vzor MVC, ktorý prenáša na stranu klienta do webového prehliadača [40].

Šablóny vrstvy *view* (pohľad) písané v jazyku HTML, využívajú deklaratívny prístup programovania pomocou tzv. direktív (angl. *directive*), ktoré prepájajú vrstvy *view* a *model*. Ďalšou dôležitou časťou architektúry MVC sú kontroléry (angl. *controllers*), ktoré obsluhujú jednotlivé časti užívateľského rozhrania. Každý kontrolér je viazaný k určitému HTML elementu, v ktorom je vytvorený jeho kontext (*$scope*). Kontext slúži k previazaniu kontroléra a vrstvy *view*. V kontexte sú uložené dáta modelu. Kontext formuje hierarchickú štruktúru odpovedajúcu zanoreniu elementov kontroléru. To znamená, že zanorené kontroléry obsahujú odkaz na predka, pričom najvyšší kontext je označený ako koreňový (@*rootscope*) [40].

Technika obojstranného previazania dát (angl. *Two-Way Data Binding*) použitá v AngularJS predstavuje koncept obojstrannej automatickej synchronizácie dát medzi vrstvou *view* a *model*. Ilustrácia techniky je na obrázku 8.1 [41].



Obrázok 8.1 Two-Way Data Binding v AngularJS [41]

AngularJS umožňujú organizáciu aplikácie do logických celkov tzv. modulov. Aplikácia sa potom skladá z jadra rámca, vlastných modulov a potrebných prídavných modulov. K prídavným modulom patria napr. moduly zaoberajúce sa smerovaním na strane klienta (*ui-router*, *routes*),

komunikáciou so serverom (*http*, *resources*), internacionalizáciou a lokalizáciou aplikácie (*translate*). Existuje ešte veľké množstvo modulov vyvinutých externými vývojármi.

Moduly sa skladajú z kódu direktív, kontrolérov, služieb a iných. Služby (*services*) sú objekty využívajúce návrhový vzor jedináčik. To znamená, že v aplikácii existuje iba jedna inštancia objektu služby. K získaniu týchto služieb sa využíva návrhový vzor DI, ktorý rieši závislosti medzi jednotlivými komponentami systému a modulmi.

8.2 Implementácia webovej služby

Ako už bolo uvedené v kapitole 7.2, webová služba systému je trojvrstvová serverová aplikácia, ktorá spravuje a poskytuje dáta z databázy PostgeSQL. Databáza obsahuje rozšírenie PostGIS. Dáta sú poskytované vo forme JSON objektov cez rozhranie založené na REST architektúre. K reprezentácii priestorových dát trasy je použitý formát GeoJSON. Nasledujúce podkapitoly sú zamerané na spôsob implementácie jednotlivých vrstiev predstavených v kapitole 7.4.1 pomocou technológií Java EE 7 platformy.

8.2.1 Doménová vrstva

Doménová vrstva je zodpovedná za správu perzistených dát uložených v PostGIS. Vrstva využíva JPA rámec implementovaný pomocou Hibernate. Rámec JPA a ani jeho implementácia Hibernate nepodporujú prácu s priestorovými dátami a funkciami. Preto sa používa rozšírenie *Hibernate Spatial*[5]. Štruktúru doménovej vrstvy môžeme pozorovať v diagrame tried na obrázku C.1 v prílohe C.

Balík *entity*

Použitie JPA rámca v balíku *entity* spočíva hlavne v konfigurácií ORM mapovania tried na jednotlivé tabuľky databáze PostGIS. Triedy z tohto balíka budeme ďalej nazývať entitami. Všetky entity okrem *UserDetail* sú anotované pomocou anotácie *@Entity*, čím sú mapované práve na jednu tabuľku databázy. K mapovaniu atribútov entít na stĺpce tabuliek sa používa anotácia *@Column*. Integritné obmedzenia sú zaistené pomocou atribútov anotácie *@Column* a pomocou anotácií rámca *Bean Validation* ako *@NotNull, @Size*, atď. Anotácia *@Id* sa používa na mapovanie primárneho kľúča. Entity *SystemProperty* a *User* využívajú ako primárny kľúč Java typ *String*. Ostatné entity využívajú celočíselné primárne kľúče mapované na typ *long*. Vzťahy medzi entitami sú vyjadrené pomocou referencií alebo kolekcie referencií, ktoré sú anotované *@OneToOne*, *@OneToMany* a *@ManyToMany* anotáciami. Všetky referencie reprezentujúce vzťahy sú jednosmerné. Tvorba indexov a sekvencií určených ku generovaniu primárnych kľúčov je zabezpečená pomocou anotácie *@Index* a *@SequenceGenerator*. Špecifické JPQL dotazy sú uchované v entitách pomocou anotácie *@NamedQuery*. Ukážka mapovania entity *Location* na tabuľku je prezentovaná v kóde 8.1.

Všetky vzťahy entity *Track* majú nastavenú vlastnosť vzťahu *fetch* na *FetchType.LAZY*. Vďaka vlastnosti *fetch* môžeme kontrolovať, ktoré vzťahy sa majú načítať spolu s hlavnou entitou a kedy. Hodnota *Lazy* udáva, že vzťahy sa načítajú až pri prvom prístupe k dátam. Hodnota je však zvolená z dôvodu, že sa využíva špeciálny typ joinu *JOIN FETCH* v JPQL a Criteria API.

Mapovanie priestorových dát na atribút *geometry* entity *Location* zabezpečuje anotácia *@Type* z rozšírenia Hibernate Spatial. Priestorové dáta sú ukladané vo WGS-84 súradnicovom systéme.

[5] http://www.hibernatespatial.org/

```
1.  @Entity
2.  @SequenceGenerator(name = Location.ID_SEQ_NAME,
3.          sequenceName = Location.ID_SEQ_NAME,
4.          allocationSize = Location.ID_SEQ_ALLOC_SIZE)
5.  public class Location implements Serializable {
6.      @Id
7.      @Column(name = "location_id", nullable = false, unique = true)
8.      @GeneratedValue(generator = ID_SEQ_NAME,
9.                  strategy = GenerationType.SEQUENCE)
10.     private Long id;
11.
12.     @NotNull @Basic(optional = false)
13.     @Type(type = "org.hibernate.spatial.GeometryType")
14.     @Column(nullable = false, name = "geometryy")
15.     private Geometry geometry;
16.
17.     /*Gettery a Settery*/
18. }
```

Kód 8.1 Ukážka mapovania entity *Location* na tabuľku *Locations* pomocou rámca JPA

Balík *dao*

Balík *dao* obsahuje rozhrania a ich implementácie k manipulácii dát v databáze. Balík využíva bezstavové session EJB komponenty. Týmito komponentami sú triedy *TracksDaoJPA* a *GenericDaoJPA*, ktoré zdieľajú chovanie abstraktnej triedy *DaoJPA*. Kvôli programovaniu do rozhraní a typovo bezpečnej injekcii majú tieto triedy vyextrahované verejné metódy do rozhraní v balíčku *api*.

DaoJPA je generickou implementáciou návrhového vzoru DAO. Generická implementácia znamená, že nám stačia generické metódy jednej triedy k správe všetkých entít, pričom typ entity sa určí z predaného parametru. Trieda obsahuje generické metódy pre CRUD operácie (*create, find, update, delete*) a pre volanie JPQL dotazov s rôznymi parametrami (*findWithNamedQuery*). Tieto operácie delegujú predané argumenty na správcu entít, ktorý sa injektuje anotáciou *@PersistenceContext*. Správca entít je konfigurovaný pomocou súboru ***persistence.xml***, ktorý nastavuje typ transakčného spracovania na transakcie spravované kontajnerom (angl. *container-managed transaction*). Vďaka nemu sú transakcie riadené pomocou anotácií *@TransactionAttribute* a jej atribútov nastavených na metódach triedy *DaoJPA*. Väčšina metód využíva túto anotáciu s hodnotou atribútu nastavenou na *TransactionType.REQUIRED*, ktorý zabezpečí, že v prípade nepropagovania transakcie zo servisnej vrstvy sa metóda vždy vykoná v novej transakcii. Ukážku implementácie EJB komponenty *DaoJPA* je možné pozorovať v kóde 8.2.

Pre špecifické dotazy nad entitou *Track*, ktoré nie je možné implementovať pomocou generických metód, je vytvorená trieda *TracksDaoJPA*. Jej metódy využívajú *Criteria API* na budovanie efektívnejších špecifických dotazov zameraných na stránkovanie a filtráciu trás.

8.2.2 Servisná vrstva

Servisná vrstva je implementovaná pomocou bezstavových EJB komponent, ktorých metódy implementujú biznis logiku a riadia transakcie anotáciou *@TransactionAttribute*. Tieto metódy propagujú transakcie a delegujú akcie na metódy tried z doménovej vrstvy balíka *dao* alebo v niektorých prípadoch aj na metódy inej EJB komponenty. Metódy ešte pred samotnou delegáciou kontrolujú validitu predaných údajov. Ak dáta nie sú valídne, tak sú vrátené výnimky z balíka *exceptions*. Závislosť na používaných komponentách je vytvorená pomocou DI a anotácie *@Inject* rámca CDI. Diagram tried servisnej vrstvy je uvedený na obrázku C.2 v prílohe C. Obrázok zachytáva balíky, triedy, rozhrania a ich závislosti. V kóde 8.3 sa nachádza ukážka implementácie

servisnej triedy *TracksService* pomocou EJB rámca. Servisná vrstva obsahuje tieto najdôležitejšie triedy:

- ***AbstractGenericService*** – abstraktná trieda, ktorá zapuzdruje spoločné chovanie pre ostatné triedy. Obsahuje referenciu na *IGenericDaoJPA*. Ostatné triedy okrem triedy *TracksService* ju ďalej rozširujú.

- ***AuthService*** – zastrešuje autentifikačné služby. Obsahuje metódu *authenticate* na prihlásenie, *sign* na generovanie JWT autentifikačného tokenu a *verify* na overenie tokenu. Na prácu s JWT tokenmi sa využíva knižnica ***java-jwt***[6].

- ***SystemPropertiesService*** – zapuzdruje metódy na získanie hodnôt systémových nastavení. Medzi tieto nastavenia patrí tajný kľúč šifrovania tokenu, expiračný čas tokenu a maximálna veľkosť nahrávanej fotografie a GPX súboru. Token je platný 24 hodín a veľkosť fotografie a GPX súboru je z bezpečnostných dôvodov nastavená na 1MB.

- ***PhotosService*** – obsahuje metódy k vytvoreniu/editovaniu fotografie (*createOrUpdatePhoto*) a k vyhľadávaniu fotografie na základe UUID (*findPhotoByUUID*).

- ***StatisticsService*** – zapuzdruje metódy slúžiace na počítanie štatistík. Metóda *compute* vypočítava k trase všetky štatistiky, ktoré sú ukladané do databázy. K ich výpočtom sa využívajú triedy z balíka *statistics*. Haversinov[7] vzorec implementovaný v triede *GisUtils* je použitý na výpočet vzdialenosti medzi dvomi bodmi. Ostatné metódy slúžia k tvorbe mesačných a celkových štatistík, ktoré delegujú výpočet na pomenované JPQL dotazy používajúce agregačné funkcie. Práve kvôli celkovým a mesačným štatistikám sú štatistiky trasy ukladané do databázy.

- ***TracksService*** – zapuzdruje metódy pre správu trás. Metóda *createTrack* pri vytváraní novej trasy využíva k výpočtu štatistík injektovanú komponentu *StatisticsService*. Používateľské trasy je možné získať metódou *getUserTrack* a zdieľané trasy metódou *getSharedTrack*. Body záujmu trasy sú vytvárané a mazané metódami *addPoi* a *deletePoi*. K vyhľadávaniu používateľských trás slúži metóda *findUserTracksByParameters*. Implementácia metódy je zobrazená na riadku 20 v kóde 8.3. Vyhľadávacie parametre ako *offset*, *limit*, *tags*, *fromDate*, *toDate*, *toDistance* a *fromDistance* sa nachádzajú v DTO triede *TrackSearchParams*. Metóda *findUserTracksByParameters* najskôr zistí počet všetkých trás špecifikovaného používateľa, ktoré spĺňajú vyhľadávacie kritéria (riadok 11). V prípade, že nie je nájdená žiadna trasa, tak je vrátený prázdny zoznam trás. V opačnom prípade sa ďalej zisťujú primárne kľúče týchto trás (riadok 17), pričom počet vrátených kľúčov je obmedzený parametrami stránkovania. Nakoniec sa na základe zoznamu týchto kľúčov získajú samotné trasy, ktoré sa predávajú späť z metódy aj s počtom nájdených trás. Metóda *findSharedTracksByParameters* používa rovnaký princíp s tým rozdielom, že vyhľadáva iba v zdieľaných trasách. Tento princíp bol zvolený z dôvodu, že vykonať takýto dotaz pomocou jedného dotazu nie je možný kvôli existencii M:M vzťahu medzi entitami *Track* a *Tag*.

- ***UsersService*** – zastrešuje služby pre správu používateľov a ich údajov.

[6] https://github.com/auth0/java-jwt

[7] http://en.wikipedia.org/wiki/Haversine_formula

```java
public abstract class DaoJPA extends IDaoJPA {
    @PersistenceContext private EntityManager em; // Injection

    @Override public <T extends Serializable> T create(T t) {
        em.persist(t); return t; // Save entity
    }

    @Override @TransactionAttribute(TransactionAttributeType.SUPPORTS)
    public <T extends Serializable> T find(Class<T> type, Object id) {
        return (T) em.find(type, id); // Find entity
    }

    @Override @TransactionAttribute(TransactionAttributeType.SUPPORTS)
    public <T extends Serializable>
    List<T> findWithNamedQuery(String namedQueryName, Class<T> clazz) {
        return em.createNamedQuery(namedQueryName, clazz)
                 .getResultList();
    }
    // ...
}
```

Kód 8.2 Ukážka generickej implementácie návrhového vzoru DAO pomocou JPA rámca

```java
@Stateless @TransactionAttribute(TransactionAttributeType.REQUIRED)
public class TracksService implements ITracksService {

    @Inject private ITracksDaoJPA tDao;

    @Override @TransactionAttribute(TransactionAttributeType.REQUIRED)
    public PaginatedTrackListDTO findUserTracksByParameters(String userId,
        TrackSearchParams searchParams) {

        long count =
            tDao.searchUserTracksCountWithCriteria(userId, searchParams);

        if (count == 0) {
            return new PaginatedTrackListDTO(new ArrayList<Track>(), count);
        }
        List<Long> idsList =
            tDao.searchUserTracksIdsWithCriteria(userId, searchParams);
        return new PaginatedTrackListDTO(tDao.findTracksByIds(idsList), count);
    }
}
```

Kód 8.3 Ukážka implementácie triedy *TrackService* a jej metódy *findUserTrackByPrameters* pomocou EJB rámca

8.2.3 Fasádna vrstva

Fasádna vrstva tvorí komunikačné rozhranie poskytujúce prístup vonkajších klientskych aplikácií do webovej služby systému. Hlavnou funkciou fasádnej vrstvy je prijať dáta, validovať ich, delegovať ich na servisnú vrstvu a vrátiť výsledok zo servisnej vrstvy. Vrstva využíva k implementácii komunikačného rozhrania rámec JAX-RS. Diagram tried fasádnej vrstvy je znázornený na obrázku C.3 v prílohe C.

JAX-RS umožňuje oddeliť adresovanie zdrojov od kódu obslužnej metódy zdroja. K adresovaniu sa využíva rozhranie s anotáciami JAX-RS. Kód obslužných metód zdrojov je implementovaný až v triede implementujúcej rozhranie. Takýto prístup je využitý aj vo fasádnej vrstve.

V balíku *api* sú rozhrania *IUserResource*, *IPhotoResource*, *IAuthResource* a *ITracksResource*, ktorých metódy adresujú jednotlivé zdroje komunikačného rozhrania z kapitoly 7.4.2. V kapitole

7.4.2 je opísaný aj význam zdrojov (zodpovedajúcich metódam), preto sa nimi v tejto kapitole už nebudeme zaoberať. Príklad adresovania REST zdrojov k správe používateľov a ich trás je znázornený v kóde 8.4.

V balíku fasádnej vrstvy sú potom implementácie týchto rozhraní, ktoré obsahujú injektované triedy servisnej vrstvy.

```
1.  @Path('users')
2.  @Consumes(MediaType.APPLICATION_JSON)
3.  @Produces(MediaType.APPLICATION_JSON)
4.  public interface IUsersResource extends IRestFacade {
5.
6.      @Authenticated @Authorized
7.      @GET @Path('{userId:[a-zA-Z0-9]{5,15}}')
8.      Response findUser(@PathParam('userId') String userId);
9.
10.     @POST
11.     Response createUser(@Valid @NotNull NewUserDTO newUser);
12.
13.     @Authenticated @Authorized
14.     @DefaultContentTypeJson @MaxContentLength
15.     @POST @Consumes(MediaType.MULTIPART_FORM_DATA)
16.     @Path('{userId:[a-zA-Z0-9]{5,15}}/tracks')
17.     Response createTrack(@PathParam('userId') String userId,
18.                 @MultipartForm NewTrackFormDTO newTrackFormDTO);
19.     /*...*/
20. }
```

Kód 8.4 Ukážka spôsobu adresovania komunikačného rozhrania REST pomocou rámca JAX-RS

Poskytovatelia obsahu

V balíku *providers* sa nachádzajú poskytovatelia obsahu, ktorí sú potrební na beh webovej služby. JAX-RS rozpozná týchto poskytovateľov obsahu podľa anotácie *@Provider* a pridruží ich k určitým formátom na základe hodnôt atribútov anotácií *@Produces* a *@Consumes*.

Trieda *JacksonConfig* využíva k vstupnej/výstupnej transformácii objektov z/do formátu JSON rámec Jackson, ktorý je rozšírený o zásuvný modul realizujúci transformáciu priestorového typu z rámca Hibernate Spatial na formát GeoJSON a opačne. Anotácie Jackson určené k nastaveniu transformácie sú používané triedami balíka *dto*.

Metóda *createTrack* rozhrania *IUserTrack* prijíma formát *multipart/form-data*, ktorý obsahuje GPX súbor s trasami. Rámec Java Architecture for XML Binding (JAXB) je použitý k transformácii tohto GPX súboru na objekt Javy v triede *ValidationGpxTypeMessageBodyReader*.

Validácia vstupných dát

Validácia vstupných dát je zabezpečená buď regulárnym výrazom anotácie *@Path* (kód 8.4 riadok 7) alebo pomocou anotácií rámca Bean Validation (kód 8.4 riadok 11). Ten umožňuje nastavovať obmedzenia priamo na atribúty tried v balíku *dto*. Kontrola obmedzení je aktivovaná pomocou anotácie *@Valid* a dochádza k nej až po transformácií vstupných dát do Java objektu.

Metódy využívajúce iný formát vstupných dát ako JSON, vykonávajú validáciu vstupných dát programovo. Príkladom je napríklad metóda *createTrack*, ktorá k overeniu validity na základe schémy GPX súboru využíva triedu *ValidationGpxTypeMessageBodyReader*. Po dokončení tohto overenia metóda vymaže všetky nepotrebné údaje z objektu reprezentujúceho GPX súbor (ponechá iba jednu trasu v prípade viacerých) a kontroluje, či všetky body trasy obsahujú informácie

o zemepisnej šírke, dĺžke, nadmorskej výške a časovej značke. Nakoniec sa kontrolujú doplnkové údaje o trase.

Zabezpečenie

Fasádna vrstva implementuje autentifikáciu založenú na tokene pomocou triedy *AuthResource* a *JWTBearerTokenAuthenFilter*. Metóda *getAccessToken* triedy *AuthResource* vráti vygenerovaný autentifikačný JWT token. Trieda *JWTBearerTokenAuthenFilter* je filtrom rámca JAX-RS. Táto trieda zabezpečuje odopretie prístupu v prípade, že token zaslanej požiadavky nie je valídny. Ak je token valídny, tak nastavuje rámcu JAX-RS bezpečnostný kontext, ktorý je reprezentovaný triedou *JWTBearerTokenSecurityContext*. K mapovaniu filtra na metódu je vytvorená vlastná anotácia *@Authenticated*, ktorej použitie môžeme vidieť v kóde 8.4 na riadku 6 a 13.

Overeniu autorizácie používateľa dochádza pomocou JAX-RS filtra triedy *JWTBearerTokenAuthorFilter*. Ten overuje, či parameter *userId* nastavený na zabezpečenom zdroji je totožný s menom používateľa nastaveného v JWT tokene. Mapovanie filtra na metódu je dosiahnuté vlastnou anotáciou *@Authorized* (kód 8.4 riadok 12).

Ďalší bezpečnostný mechanizmus je implementovaný filtrom *MaxContentLengthFilter* a anotáciou *@MaxContentLength* (kód 8.4 riadok 13). Tento filter na základe nastavení systému overuje veľkosť GPX súboru a nahrávanej fotografie.

Posledný bezpečnostný prvok je použitý na metóde *getPhoto* triedy *PhotosResource*. Metóda vyhľadá fotografiu na základe UUID a vráti ju ako pole bytov (*byte[]*). Táto metóda však nie je zabezpečená autentifikačným filtrom kvôli klientovi systému, pretože ten k zobrazovaniu fotografie využíva kód 8.5. Tento kód nedovoľuje pripojiť JWT token do hlavičky požiadavky, pretože požiadavka na fotografiu je zaslaná v réžii webového prehliadača. Ak by sme fotografie vyhľadávali podľa celočíselného identifikátora, tak by sa mohla k fotografiám dostať akákoľvek neoprávnená osoba. Z toho dôvodu sa pri ukladaní fotografie vytvára UUID, ktoré už nie také jednoduché uhádnuť.

```
1.  <img ng-src="/rest/photos/1234-123123" >
```

Kód 8.5 Ukážka vloženia obrázku do webovej stránky pomocou HTML elementu *img* a AngularJS atribútu *ng-src*

8.3 Implementácia webového klienta

Webový klient predstavuje prezentačnú vrstvu systému, ktorá zobrazuje dáta webovej služby. Užívateľské rozhranie klienta je realizované pomocou šablóny AdminLTE, ktorá používa HTML a CSS súbory. Stav klienta je riadený a udržovaný klientskymi skriptami, ktoré sú vytvorené pomocou skriptovacieho jazyka JavaScript a jeho rámca AngularJS. Trasy sú vizualizované knižnicou OpenLayers 3 na mapových podkladoch služby *Mapy.cz*. Integrácia mapových podkladov je popísaná v podkapitole 8.4.3.

Webový klient je štruktúrovaný do modulov. Moduly klienta sú reprezentované balíkmi v diagrame balíkov na obrázku C.4 v prílohe C. Každý modul obsahuje súbor s príponou **.module.js*, ktorý špecifikuje závislosti modulu, vďaka čomu môžeme v module používať DI.

8.3.1 Moduly obrazoviek klienta

Moduly vyznačené svetlo hnedou farbou v sebe združujú skripty a šablóny, ktoré slúžia k nastaveniu a ovládaniu obrazoviek. Obrazovky boli popísané v kapitole 7.5. Názvy modulov spolu s názvami rodičovských modulov tvoria relatívnu cestu obrazovky, na ktorej je obrazovka dostupná v prehliadači. Napríklad obrazovka slúžiaca k prihláseniu má relatívnu cestu *auth/login*. Každý takýto modul obsahuje:

- šablóny s príponou **.tpl.html* – obsahujú HTML šablóny (pohľady) jednotlivých častí obrazovky a ich dialógových okien.

- skript s príponou **.routes.js* – obsahuje konfiguráciu stavu obrazovky smerovacieho modulu ui-router (viď. kapitola 8.4.4). V konfigurácii stavu je nadefinovaná hlavná šablóna, kontrolér a URL obrazovky. V niektorých prípadoch sú v stave zadefinované dáta, ktoré je potrebné získať ešte pred presmerovaním na obrazovku. Príkladom je napríklad získanie údajov trasy z webovej služby systému pred samotným zobrazením obrazovky detailu trasy.

- skript s príponou **.translate.js* – obsahuje internacionalizované texty obrazovky. Klient momentálne podporuje len anglický jazyk.

- skripty s príponou **Controller.js* – sú najdôležitejšími skriptami v moduloch. Každý skript obsahuje jeden kontrolér rámca AngularJS. Metódy kontroléra obsluhujú jednotlivé časti užívateľského rozhrania a delegujú používateľské akcie na služby z modulu *trackbook-api*, ktorý sprostredkúva komunikáciu s webovou službou systému.

```
1.  angular.module('trackbookApp.tracks').controller("TracksListController",
2.    function ($scope, tracksResource, ... /* DI of services */) {
3.      // Model for pagination panel.
4.      $scope.pagination = { limit: 5, offset: 1, totalItems: 0 };
5.      $scope.trackList = { data:[] }; // Model of found tracks.
6.      $scope.filterData = { distanceRange: [0, 100], dateRange: {} },
7.
8.      $scope.performSearch = function (newSearch) {
9.        $scope.toggleIsLoading(); // Signalize loading.
10.       // Create backend communication object.
11.       tracksHttp = tracksResource.getAllSharedTracks($scope.pagination.limit, ...);

12.       // Register handler methods.
13.       tracksHttp.success(function (data, status) {
14.         $scope.setTrackListData(data, newSearch); // Set tracks to model.
15.       }).error(function (data, status) {
16.         /* Inform about error */
17.       });
18.     };/* ... */
19. });
```

Kód 8.6 Ukážka implementácie kontroléra *TrackListController* z modulu *tracks*

Na obrázku 8.2 sú v module *tracks* pomocou tried zobrazené súbory a šablóny slúžiace k tvorbe obrazovky vyhľadávania trás. V kóde 8.6 sa nachádza ukážka implementácie kontroléra *TrackListController* zo skriptu *trackListController.js*. Na riadku 2 dochádza k DI injekcii služby *TracksResource* z modulu *trackbook-api*. Kontrolér obsahuje na riadku 4 až 6 modely dát, ktoré sú uložené v kontexte kontroléra. Riadok 8 ukazuje metódu *performSearch*, ktorá zasiela dáta na REST zdroj za účelom vyhľadania trás. V metóde na riadkoch 13 a 15 dochádza k registrácii obslužných

funkcií, ktoré spracujú výsledok vrátenej požiadavky a aktualizujú hodnoty modelu. Táto metóda je tiež definovaná na kontexte kontroléra, a preto ju je možné zavolať z pohľadu.

8.3.2 Zdieľané moduly

Modul *shared* obsahuje komponenty, ktoré sú zdieľané vo všetkých obrazovkách klienta. Tieto komponenty sú ďalej štruktúrované do ďalších podmodulov podľa funkcionality, ktorú vykonávajú. Funkcie modulov je možné odvodiť z názvu modulu. Najzaujímavejšie moduly sú *trackbook-api*, *security* a *ol-seznam*.

Modul *trackbook-api*

Komunikáciu klienta s webovou službou systému zaisťuje modul *trackbook-api* a jeho služby *UsersResource, AuthResource*, *TracksResource*. Tieto služby sú založené na službe *$http* rámca AngularJS alebo na službe *$upload* z modulu *angularFileUpload* (viď. kapitola 8.4.4). Obe služby poskytujú rozhranie k zaslaniu HTTP požiadavky na vzdialené dátové zdroje.

Hlavný význam služieb z balíka *trackbook-api* spočíva v tom, že ich funkcie abstrahujú komunikáciu s každým REST zdrojom webovej služby. Funkcie delegujú predané argumenty na vyššie zmienené služby, ktoré nastavia URL, metódu protokolu HTTP a formát zaslaného objektu na JSON alebo *multipart/form-data*. V kóde 8.7 je ukázaná implementácia služby *UserResource* a jej abstrahovaných funkcií. Integrácia služby s kontrolérom je ukázaná v kóde 8.6 riadok 13 až 17.

```
1.  angular.module('trackbookApp.shared.trackbook-api').factory("UsersResource",
2.    function ($http, apiPaths, $upload, /* DI of services */) {
3.      // Fetches shared track with given id.
4.      function getTrack(trackId) {
5.        return $http.get(
6.          apiPaths.tracksIdPath.replace(':trackId', trackId));
7.      }
8.      // Uploads new user track with GPX file.
9.      function uploadNewTrack(userId, file, trackMetaData) {
10.       return $upload.upload({
11.         url: apiPaths.usersIdTracksPath.replace(':userId', userId),
12.         fields: {track: trackMetaData},
13.         file: file
14.       });
15.     }
16.     /*...*/
17.     // Return singleton object.
18.     return { uploadNewTrack: uploadNewTrack, getTrack: getTrack,/*...*/};
19. });
```

Kód 8.7 Ukážka implementácia služby *UsersResource* modulu *trackbook-api*

Modul *ol-seznam*

Modul *ol-seznam* má direktívu *olSeznamMaps*, ktorá vkladá všetky vrstvy od *Mapy.cz* do mapy vytvorenej modulom *openlayers-directives*. Direktíva pridáva aj ovládací prvok určený k prepínaniu máp. Modul obsahuje aj službu *OlSeznamMapsHelpers*, ktorá implementuje pomocné funkcie zjednodušujúce prácu s mapou a jej vrstvami. K zaujímavým funkciám patrí napr. *addDataToLayer*, ktorá vkladá GeoJSON dáta do špecifikovanej vrstvy.

Modul *security*

Modul *security* obsahuje služby slúžiace k zabezpečeniu klienta. V module sa nachádzajú tieto hlavné služby *IdentityService, AuthInterceptor a AuthService*. *IdentityService* slúži ako úložisko JWT autentifikačného tokenu a identity prihláseného používateľa. Služba poskytuje funkcie k ich získaniu a nastaveniu. *AuthInterceptor* reprezentuje *$http* interceptor, ktorý pomocou metódy *request* zachytí HTTP požiadavky, pripojí k ním autentifikačný token zo služby *IdentityService* a odošle takto zmodifikované požiadavky ďalej na server. Metóda *responseError* naopak odchytáva odpovede požiadaviek a v prípade chybových kódov ako 401 a 403 odhlasuje používateľa.

AuthService zapuzdruje funkcie týkajúce sa autentifikácie klienta ako napr. *login*, *logout*, *isAuthenticated* a *loginFromLocalStorage*. Funkcia *login* najskôr získa autentifikačný token z webovej služby systému, ktorý uloží do služby *IdentityService*. V prípade, že sa používateľ nechce pri ďalšej návšteve stránky prihlasovať (funkcia klienta *rememberMe*), tak je tento autentifikačný token uložený aj do lokálneho úložiska webového prehliadača. Po úspešnom prihlásení funkcia *login* presmeruje používateľa na jeho „Dashboard“ obrazovku. Odhlásenie používateľa poskytované funkciou *logout* je veľmi jednoduché, pretože stačí zmazať autentifikačný token zo služby *IdentityService*, prípadne aj z lokálneho úložiska a nakoniec presmerovať používateľa na prihlasovaciu obrazovku klienta.

8.3.3 Mapové podklady Mapy.cz v OpenLayers 3

Keďže OpenLayers 3 nedodávajú predpripravené vrstvy slúžiace na zobrazenie mapových podkladov od služby Mapy.cz, je nutné vyvinúť ich integráciu. K implementácií takejto OpenLayers vrstvy je potrebné najskôr zistiť typ súradnicového systému Mapy.cz, jeho hranice a URL mapového serveru, ktoré slúži k získavaniu jednotlivých dlaždíc.

Hlavný rozdiel medzi Mapy.cz a inými globálnymi poskytovateľmi máp spočíva v tom, že Mapy.cz nevyužívajú Mercatorovú projekciu s WGS-84 súradnicovým systémom, ale svoju vlastnú projekciu nazvanú PP, ktorá využíva mierne modifikovaný súradnicový systém UTM v zóne 33. V oficiálnej databáze existujúcich súradnicových systémov má ich súradnicový systém vlastné označenie *SR-ORG:98* [42].

Schéma URL mapového serveru Mapy.cz poskytujúceho dlaždice mapových podkladov je zobrazená na obrázku 8.2. V zložených zátvorkách sú uvedené parametre, ktoré sa menia. Červený parameter *1-4* slúži k rozdeleniu záťaže na jednotlivé mapové servery. Zelený parameter *vrstva* identifikuje typ mapového podkladu. Povolené hodnoty sú napr. *base* (základná mapa), *turist* (turistická mapa), *tcyklo* (cyklotrasy), *ttur* (turistické trasy), *hybrid* (hybridná mapa). Modré parametre adresujú jednu dlaždicu na servery v určitom priblížení (angl. *zoom*). Rozsah hodnoty parametra *zoom* je 1 až 16. Parametre *x* a *y* predstavujú súradnice ľavého dolného rohu dlaždice v projekcii PP, ktoré sú posielané na server v hexadecimálnej sústave.

http://m{1-4}.mapserver.mapy.cz/{vrstva}/{zoom}_{x}_{y}

Obrázok 8.2 Schéma URL mapového serveru Mapy.cz

Implementácia integrácie sa nachádza v triede *ol.source.SeznamMaps*, ktorá rozširuje triedu *ol.source.TileImage*. V konštruktore vrstvy sa nastavuje projekcia *SR-ORG:98* pomocou knižnice *proj4js*[8], typ mapových podkladov služby Mapy.cz a objekt typu *ol.tilegrid.TileGrid*, ktorý rozdeľuje mapu na mriežku dlaždíc. Táto mriežka sa vypočítava na základe hraníc projekcie a rozlíšenia (angl.

[8] http://proj4js.org/

resolutions). Najdôležitejšou metódou triedy *ol.source.SeznamMaps* je *tileUrlFunction_*, ktorá vypočítava súradnice *x*, *y*, transformuje ich do hexadecimálnej podoby a nakoniec podľa schémy generuje URL, ktoré vracia.

8.3.4 Použité externé moduly AngularJS

Externé moduly AngularJS sú použité na uľahčenie implementácie webového klienta systému. Sú to zásuvné moduly vyvinuté nezávislými vývojármi. V implementácii klienta sú využité nasledujúce moduly:

- **ui-router** – je modul zabezpečujúci smerovanie (angl. *routing*) z jednej obrazovky na druhú v SPA aplikáciách. Smerovanie sa vykonáva na základe stavov, ktoré majú priradenú URL adresu. Zmenou stavu sa automaticky mení aj adresa URL. Hlavnou výhodou modulu je podpora zobrazenia viacerých stavov na jednej obrazovke a možnosť ich zanorenia.

- **ui-bootstrap** – predstavuje modul, ktorý obsahuje kolekciu ovládacích prvkov rámca Bootstrap implementovaných v AngularJS. Medzi ovládacie prvky patrí napríklad dialógové okno, stránkovací panel, progressbar, panel na výber dátumu, atď.

- **ng-translate** – je modul využitý k internacionalizácii textov všetkých obrazoviek a ich dialógových okien.

- **openlayers-directive** – slúži k integrácii knižnice OpenLayers 3 s rámcom AngularJS. Modul obsahuje direktívy, ktoré vytvárajú mapu a jej vrstvy, deklaratívne pomocou HTML. Modul je rozšírený o podporu zobrazenia vrstiev v rôznych projekciách, práve kvôli integrácii s mapovými podkladmi od Mapy.cz. Knižnica OpenLayers túto funkciu podporuje, ale openlayers-directive nie.

- **ngTagsInput** – umožňuje zobrazovanie, vkladanie a mazanie tagov v špeciálnom formulárovom prvku. Prvok je využitý napr. na obrazovke vytvorenia novej trasy.

- **n3-line-chart** – modul slúži na tvorbu čiarových grafov a v klientovi systému sa využíva k tvorbe výškového profilu na obrazovke detailu trasy.

- **angularFileUpload** – modul, ktorý umožňuje programovo vytvárať a zasielať súbory vo formáte *multipart/form-data*. K nahrávaniu GPX súboru a fotografie sa využíva práve tento modul, pretože štandardná služba *$http* túto funkciu nepodporuje.

- **angular-growl** – je modul, ktorý slúži k zobrazovaniu notifikačných správ. Správy môžu mať informatívny, výstražný alebo chybový charakter. Využíva sa k oboznámeniu používateľa o stave vykonaných akcií.

- **angular-loading-bar** – modul umožňuje vizualizovať stav požiadavky zaslanej službou *$http* pomocou lišty progresu (angl. *progressbar*) zobrazeného v hornej časti obrazovky.

- **angular-slidezilla** – modul obsahuje ovládací prvok posuvník (angl. *slider*), na ktorom je možné vybrať rozsah hodnôt. Je využívaný na zadávanie rozsahu dĺžky trás pri ich filtrácií.

- **localStorageModule** – modul obsahuje službu, ktorá pracuje s lokálnym úložiskom (angl. *LocalStorage*) webového prehliadača.

9 Testovanie a nasadenie

9.1 Testovanie

Prvotné testovanie prebiehalo počas vývoja jednotlivých častí aplikácie na lokálnom stroji s vlastným aplikačným serverom WildFly a PostGIS databázou. Testovanie prebiehalo v poradí, v akom boli jednotlivé časti systému vyvíjané.

V priebehu implementácie webovej služby bolo testované komunikačné rozhranie jednotlivých zdrojov. Požiadavky na jednotlivé zdroje boli generované manuálne pomocou nástroja REST Console bežiaceho vo webovom prehliadači Google Chrome. Testy sa zameriavali hlavne na vizuálnu kontrolu vrátených JSON objektov, návratových kódov a zabezpečenia jednotlivých zdrojov.

K testovaniu správneho importu GPX súboru bola zostavená testovacia množina súborov z GPS zariadení a aplikácií. Vedúci diplomovej práce dodal GPX súbory z neznámeho GPS zariadenia značky Garmin. Ďalšie GPX súbory pochádzali zo zariadenia Garmin Edge 500 a mobilných aplikácií ako MyTracks[9] a Endomodo[10]. Testovanie sa zameriavalo na kontrolu správneho mapovania GPX súboru na objekty doménovej vrstvy a tvorby štatistík. Systémom generované štatistiky sme porovnávali so štatistikami generovanými webovými aplikáciami ako sú Endomodo, UTrack a TrackProfiler[11]. Najviac sa naše štatistiky priblížili štatistikám získaných aplikáciou UTrack. Odlišnosť štatistík spočíva v nastavení rôznych prahových hodnôt využitých na výpočet jednotlivých štatistických údajov.

Počas implementácie webového klienta bolo zautomatizované testovanie jednotlivých obrazoviek pomocou rámca Arquillian[12] a jeho rozšírenia Arquillian Drone[13], ktoré umožňuje ovládať webový prehliadač a simulovať používateľské akcie z programového kódu jazyka Java. Testy sa zameriavali na základné akcie ako prihlásenie, registrácia, vyhľadávanie trás. Tieto testy boli vykonávané vo webovom prehliadači Google Chrome.

Zložitejšie akcie boli testované manuálne. Tieto testy sa zameriavali hlavne na vyplnenie formulárov s krajnými hodnotami. Následne sa vizuálne kontrolovali notifikačné správy zobrazované klientom. Príkladom krajných hodnôt boli GPX súbory a fotografie presahujúce 1 MB alebo existujúce prihlasovacie mená. Najdôležitejším manuálnym testovaním bolo testovanie obrazovky vyhľadania trás, pri ktorom sa najskôr importovali trasy spĺňajúce určité kritéria a následne sa podľa týchto kritérií vyhľadávalo a kontrolovalo, či boli vrátene správne trasy.

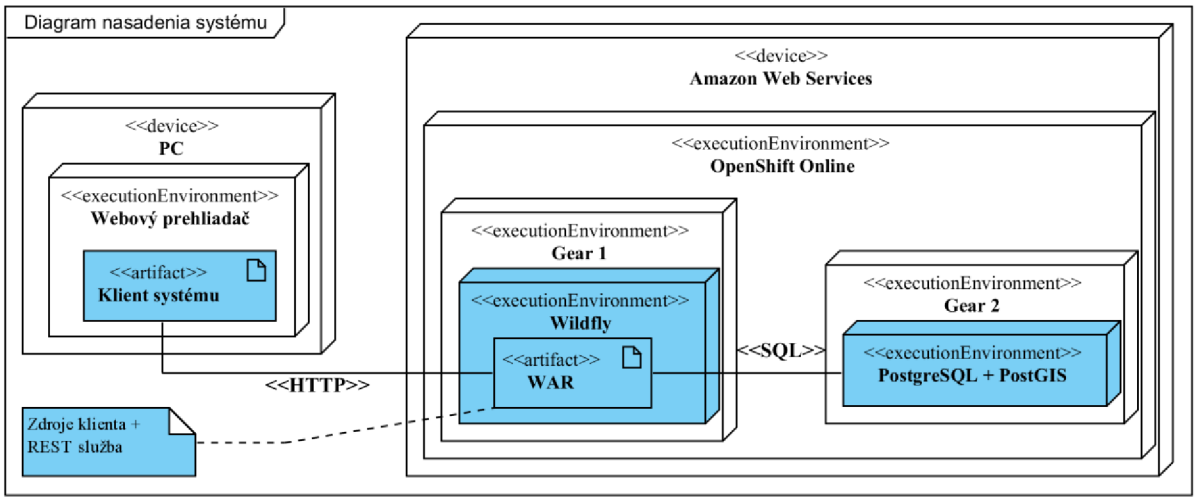
Po nasadení aplikácie na OpenShift Online bolo oslovených niekoľko priateľov so skúsenosťami s aplikáciami podobného zamerania. Okrem toho, že niektorí z nich otestovali základnú funkcionalitu systému, otestovali aj jeho funkčnosť na iných webových prehliadačoch ako Mozilla Firefox 38 a Internet Explorer 11. Po otestovaní bola získaná pozitívna spätná väzba aj s možnými návrhmi na rozšírenie.

[9] https://play.google.com/store/apps/details?id=com.google.android.maps.mytracks&hl=cs
[10] https://www.endomondo.com/
[11] http://www.trackprofiler.com/
[12] https://docs.jboss.org/author/display/ARQ/Introduction
[13] https://docs.jboss.org/author/display/ARQ/Drone

9.2 Nasadenie

Systém bol umiestnený do bezplatnej verzie platformy OpenShift Online a sprístupnený na adrese ***http://tourbook-vkasala.rhcloud.com***. K demonštračným účelom bol vytvorený účet s prihlasovacím menom *user1* a heslom *123456*.

Obe časti systému boli zabalené do súboru WAR (angl. *Web application ARchive*) a nasadené do WildFly aplikačného serveru bežiaceho v komponente gear. Systém využíva PostgreSQL databázu s rozšírením PostGIS. Nasadenie systému je zobrazené v diagrame nasadenia na obrázku 9.1.



Obrázok 9.1 Diagram nasadenia častí systému na zvolenej platforme

10 Záver

Cieľom diplomovej práce bolo preskúmať možnosti vývoja webového systému pre správu GPS dát v cloudovom prostredí. Najskôr sme preštudovali reprezentáciu a formáty geografických dát získaných zo zariadení GPS. Preskúmali sme aj online služby poskytujúce mapové podklady. Ďalej sme analyzovali cloudové platformy Google App Engine a OpenShift so zameraním na programovací jazyk Java. Okrem toho sme sa zoznámili s existujúcimi desktopovými a webovými aplikáciami podobného zamerania. Z týchto aplikácií boli prevzaté niektoré základné funkcie pre náš systém, ktoré sa premietli do špecifikácie jeho požiadaviek.

S ohľadom na zvolenú cloudovú platformu OpenShift Online sme navrhli architektúru systému a jeho častí. Podľa tohto návrhu sme implementovali systém s využitím platformy Java EE 7 a rámca AngularJS. Nakoniec sme otestovaný systém nasadili na cloudovú platformu OpenShift Online.

Systém spĺňa všetky špecifikované požiadavky zo zadania diplomovej práce. Je schopný zobrazenia a vyhľadávania trás a tvorby štatistík. K prejdeným trasám je možné získať výškové profily a priradiť doplňujúce údaje k miestam a trasám.

V budúcnosti by systém mohol byť rozšírený o tvorbu rýchlostných a vzdialenostných profilov. Na výpočet štatistík by mohli byť využité rôzne pokročilé nastavenia ako napr. voľba prahových hodnôt, využitie plávajúceho okna atď. Vďaka použitiu knižnice OpenLayers 3 by sa systém mohol rozšíriť o funkcie kreslenia a editovania trasy alebo o prepínanie medzi mapovými podkladmi rôznych dodávateľov. Systém nevyužíva plný potenciál priestorového rozšírenia PostGIS. Pri jeho plnom využití by v budúcnosti mohol byť systém rozšírený o priestorové dotazy nad trasami.

Práca priniesla nové poznatky o cloudových platformách a praktické skúsenosti v použití knižnice OpenLayers 3 a AngularJS, ktoré budem v budúcnosti naďalej využívať.

Literatúra

[1] RAPANT, P. *Družicové polohové systémy*. Vyd. 1. Ostrava: Vysoká škola báňská - Technická univerzita, 2002, 197 s. ISBN 80-248-0124-8.

[2] National Wildfire Coordinating Group: *Basic land Navigation* [online]. 2007. [cit. 2015-04-25]. Dostupné z URL: http://www.nwcg.gov/pms/pubs/475/PMS475.pdf

[3] TOPOGRAFIX. *GPX: the GPS Exchange Format* [online]. [cit. 2015-04-25]. Dostupné z URL: http://www.topografix.com/gpx.asp

[4] TOPOGRAFIX. *GPX for Developers* [online]. [cit. 2015-04-25]. Dostupné z URL: http://www.topografix.com/gpx_for_developers.asp

[5] TOPOGRAFIX. *GPX 1.1 Schema Documentation* [online]. [cit. 2015-04-25]. Dostupné z URL: http://www.topografix.com/gpx/1/1/

[6] ČAPEK, R., MUCHA, L., MIKŠOVSKÝ, M.: *Geografická kartografie*. Praha: Státní pedagogické nakladatelství Praha, 1992, ISBN 80-04-25153-6, 373 s.

[7] HRUBÝ, M.: *Geografické Informační Systémy*. Fakulta informačních technologií, VUT v Brně, 2006, 82 s.

[8] RAPANT, P.: *Geoinformatika a geoinformační technologie*. Ostrava: VŠB Technická Universita Ostrava, první vydání, 2006, ISBN 80-248-1264-9, 513 s.

[9] GOOGLE. *Google Developers: Google Maps API* [online]. 2015 [cit. 2015-04-28]. Dostupné z URL: https://developers.google.com/maps

[10] GOOGLE. *Google Developers: Google Maps JavaScript API v3* [online]. 2015 [cit. 2015-04-28]. Dostupné z URL: https://developers.google.com/maps/documentation/javascript/tutorial>

[11] GOOGLE. *Google Developers: Map Types* [online]. 2015 [cit. 2015-04-28]. Dostupné z URL: https://developers.google.com/maps/documentation/javascript/maptypes

[12] GOOGLE. *Google Developers: Layers* [online]. 2015 [cit. 2015-04-28]. Dostupné z URL: https://developers.google.com/maps/documentation/javascript/layers

[13] GOOGLE. *Google Developers: Google Maps/Google Earth APIs Terms of Service* [online]. 2014 [cit. 2015-04-28]. Dostupné z URL: https://developers.google.com/maps/terms

[14] SEZNAM. *Mapy.cz: Základní mapový podklad* [online]. 2015 [cit. 2015-04-28]. Dostupné z URL: http://napoveda.seznam.cz/cz/mapy/mapove-podklady/zakladni/

[15] SEZNAM. *Mapy.cz: Mapy API verze 4.8 – Hanzelka a Zikmund* [online]. 2013 [cit. 2015-04-28]. Dostupné z URL: http://api4.mapy.cz/

[16] SANTIAGO, A.: *The book of OpenLayers 3* [online]. LeanPub, 2014, Dostupné z URL: <https://leanpub.com/thebookofopenlayers3/read>

[17] NIST. MELL, P., GRANCE, T.: *The NIST Definition of Cloud Computing* [online]. [cit. 2015-04-28]. Dostupné z URL: http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf

[18] WIKIPÉDIA. *Cloud Computing* [online]. 2015 [cit. 2015-04-28]. Dostupné z URL: http://cs.wikipedia.org/wiki/Cloud_computing

[19] IBM. IBM: *IBM Cloud Computing* [online]. [cit. 2015-04-28]. Dostupné z URL: http://www-05.ibm.com/sk/cloud/

[20] GOOGLE. *HRÁČEK, F.: Cloud, jak ho chápeme v Googlu* [online]. 2011 [cit. 2015-04-28]. Dostupné na URL: <http://www.filiph.net/slides/idf-cloud/#slide11>

[21] DOCKSAI, R.: *Why 2013 Could be a Great Year for Cloud Computing* [online]. 2012 [cit. 2015-04-28]. Dostupné z URL: http://www.wfs.org/blogs/rick-docksai/why-2013-could-be-great-year-for-cloud-computing

[22] SANDERSON, D.: *Programming Google App Engine*. 2nd ed. Sebastopol, CA: O'Reilly, 2013. ISBN 978-1-449-39826-2.
[23] GOOGLE. *What Is Google App Engine?* [online]. 2015 [cit. 2015-04-28]. Dostupné z URL: https://cloud.google.com/appengine/docs/whatisgoogleappengine
[24] GOOGLE. *Google Cloud Platform* [online]. 2015 [cit. 2015-04-28]. Dostupné z URL: https://cloud.google.com/
[25] GOOGLE. *Java Runtime Environment* [online]. 2015 [cit. 2015-04-28]. Dostupné z URL: https://cloud.google.com/appengine/docs/java/
[26] GOOGLE. *Java Datastore API* [online]. 2015 [cit. 2015-04-28]. Dostupné z URL: https://cloud.google.com/appengine/docs/java/datastore/
[27] GOOGLE. *Google Cloud Endpoints* [online]. 2015 [cit. 2015-04-28]. Dostupné z URL: https://cloud.google.com/appengine/docs/java/endpoints/
[28] GOOGLE. *Google Developers: Paid Apps: Budgeting, Billing, and Buying Resources* [online]. 2015 [cit. 2015-04-28]. Dostupné z URL: https://cloud.google.com/appengine/pricing
[29] GOOGLE. *Google Developers: Quotas* [online]. 2015 [cit. 2015-04-28]. Dostupné z URL: https://cloud.google.com/appengine/docs/quotas
[30] POUSTY, S., MILLER, K.: *Getting started with OpenShift*. 2014, 88 s., ISBN 978-1-491-90143-4.
[31] Openshift. *Understanding OpenShift* [online]. 2014 [cit. 2015-04-28]. Dostupné z URL: https://www.openshift.com/products/architecture
[32] Openshift. *OpenShift Origin System Architecture Guide* [online]. 2015 [cit. 2015-04-28]. Dostupné z URL: http://docs.openshift.org/origin-m4/oo_system_architecture_guide.html
[33] Openshift. *Technologies* [online]. 2015 [cit. 2015-04-28]. Dostupné z URL: https://www.openshift.com/products/technologies
[34] FOWLER, M.: *Data Transfer Object* [online]. 2009 [cit. 2015-05-14]. Dostupné z URL: http://martinfowler.com/eaaCatalog/dataTransferObject.html
[35] Sun Microsystems, Inc.: *Core J2EE Patterns - Data Access Object* [online]. 2002 [cit. 2015-05-14]. Dostupné z URL: http://www.oracle.com/technetwork/java/dataaccessobject-138824.html
[36] FIELDING, R.: *Representational State Transfer (REST)* [online]. 2000 [cit. 2015-05-14]. Dostupné z URL: https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm
[37] JONES, M.: *JSON Web Token (JWT)* [online]. 2014 [cit. 2015-05-14]. Dostupné z URL: http://self-issued.info/docs/draft-ietf-oauth-json-web-token.html
[38] RAMSEY, P., WINSLOW, D.; GARNETT, J.: *Introduction to PostGIS* [online]. 2011 [cit. 2015-05-15]. Dostupné z URL: http://workshops.boundlessgeo.com/postgis-intro/introduction.html
[39] GUPTA, A.: *Java EE 7 Essentials*. Sebastopol, CA: O'Reilly Media, 2013, 343 s., ISBN 978-1449370176
[40] GREEN, B., SESHADRI, S.: *AngularJS*. Sebastopol: O'Reilly Media, 2013, 183 s., ISBN 978-144-9344-856.
[41] ANGULARJS: *Data Binding* [online]. 2015 [cit. 2015-04-28]. Dostupné z URL: https://docs.angularjs.org/guide/databinding
[42] Spatial Reference: *SR-ORG:98* [online], [cit. 2014-05-13]. Dostupné z URL: http://spatialreference.org/ref/sr-org/mapycz-projection/

Zoznam použitých skratiek a symbolov

API	*Application programming interface*
CDI	*Context and Dependency Injection for the Java EE Platform*
CLI	*Command Line Interface*
CSS	*Cascading Style Sheets*
CSV	*Comma Seperated Values*
DNS	*Domain Names Service*
EJB	*Enterprise Java Beans*
GAE	*Google App Engine*
GCP	*Google Cloud Platform*
GIS	*Geographic Information System*
GPS	*Global positioning system*
GPX	*GPS eXchange Format*
GML	*Geography Markup Language*
HTML	*HyperText Markup Language*
JAX-RS	*Java API for RESTful Web Services*
JDO	*Java Data Objects*
JPA	*Java Persitence API*
JPQL	*Java Persistence Query Language*
JRE	*Java Runtime Environment*
JSON	*JavaScript Object Notation*
JSR	*Java Specification Request*
JNI	*Java Native Interface*
JWT	*JSON Web Token*
KML	*Keyhole Markup Language*
MVC	*Model View Controller*
OGC	*Open Geospatial Consortium*
ORM	*Object Relational Mapping*
OSM	*OpenStreetMap*
POI	*Point of Interest*
REST	*Representational State Transfer*
S-JTSK	*Souřadnicový systém jednotné trigonometrické sítě katastrální*
TCX	*Training Center XML*
UML	*Unified Modeling Language*
URI	*Uniform Resource Identifier*
URL	*Unified Resource Locator*
UTM	*Universal Transverse Mercator Geographic Coordinate System*
UTC	*Coordinated Universal Time*
UUID	*Universally Unique Identifier*
WFS	*Web Feature Service*
WMS	*Web Map Service*
WMTS	*Web Map Tile Service*
WGS84	*World Geodetic System 1984*
XML	*eXtensible Markup Language*

Zoznam obrázkov

Zoznam tabuliek

Zoznam príloh

Príloha A: Obsah priloženého CD

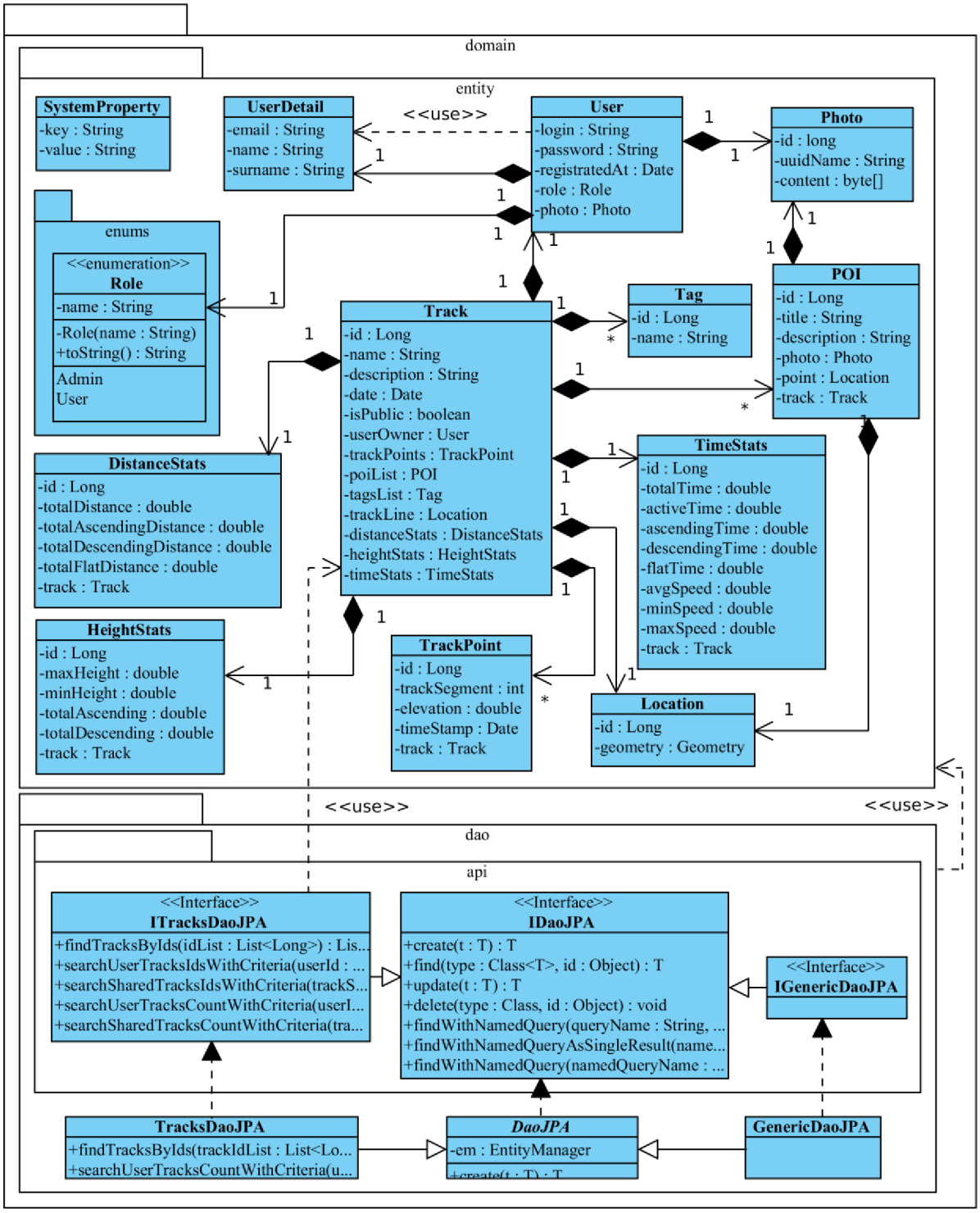
Cesta/súbor	Popis
sources	Priečinok so zdrojovými kódmi všetkých častí systému.
sources/trackbook	Priečinok so zdrojovými kódmi webovej služby a klienta.
sources/ol3-mapy.cz	Priečinok so zdrojovými kódmi integrácie OpenLayers a Mapy.cz.
sources/angular-directives-mapy.cz	Priečinok s upravenou verziou *openlayers-directive*.
dp.docx	Technická písomná správa vo formáte DOCX
dp.pdf	Technická písomná správa vo formáte PDF
Readme.txt	Súbor s popisom inštalácie.

Príloha B: Podporované rámce v platforme GAE

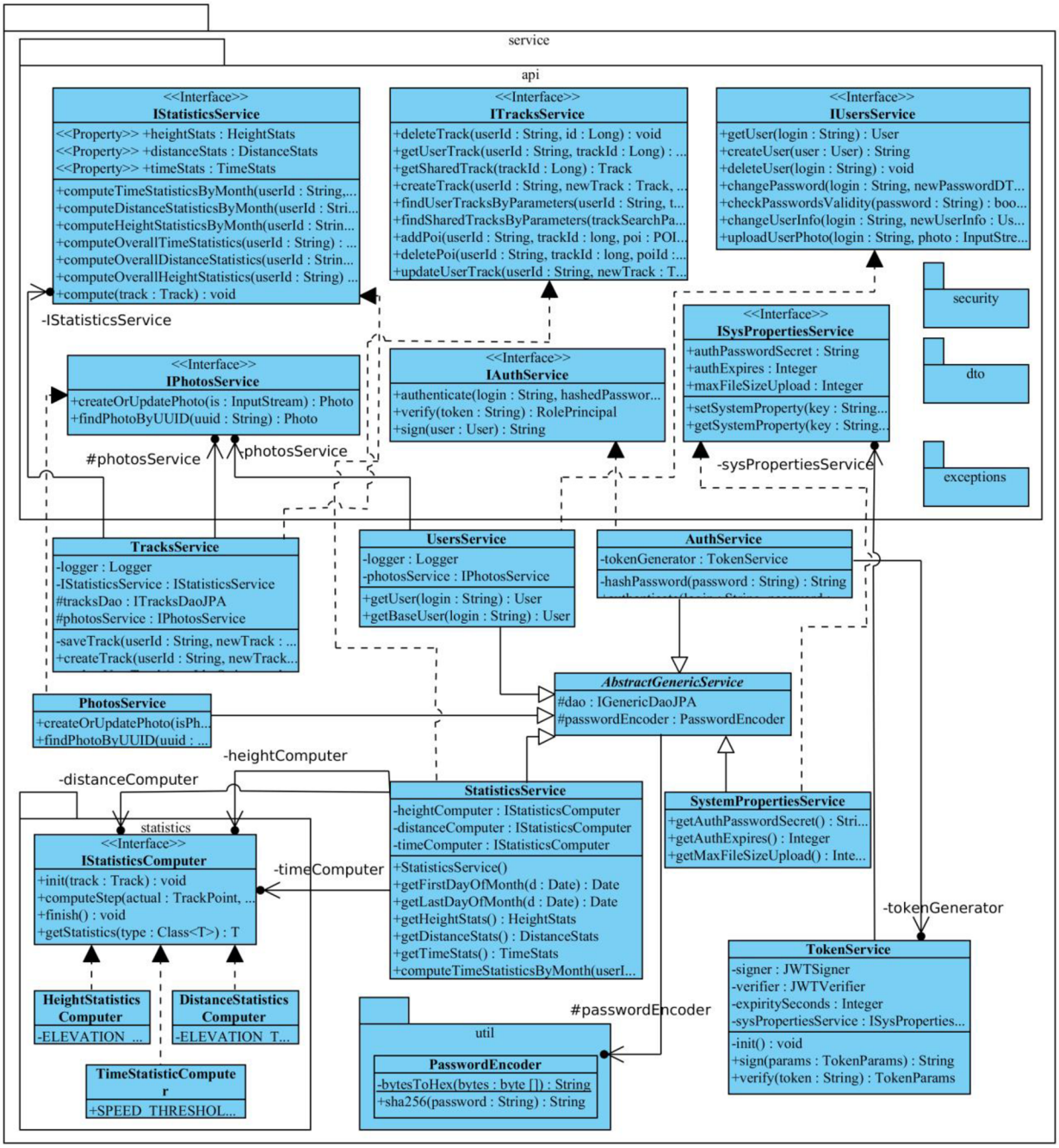
Rámec	Verzia	Kompatibilita	Popis
Restlet	2.0 M5+	Kompatibilný	Rámec pre REST aplikácie.
Jersey	1.1.5	Kompatibilný	Rámec pre REST aplikácie.
Spring MVC	2.5.6	Kompatibilný	MVC rámec pre vývoj webových aplikácii
Stripes		Kompatibilný	MVC rámec pre vývoj webových aplikácii.
Play Framework	n/a	Kompatibilný	Jednoduchý MVC rámec pre vývoj webových aplikácii.
MyFaces	1.1.6, 2.0.0-beta-3	Kompatibilný	Knižnica JSF komponent.
Spring Security		Polo kompatibilný	Rámec pre automatickú verifikáciu a autorizáciu aplikácie.
Wicket		Polo kompatibilný	Komponentový rámec pre tvorbu webových aplikácii.
Tapestry	5.0.18	Kompatibilný	Komponentový rámec pre tvorbu webových aplikácii.
Guice		Polo kompatibilný	Dependency Injection rámec.
Google Web Toolkit	All	Kompatibilný	Rámec pre tvorbu webových užívateľských rozhraní v Jave.

Tabuľka B.1 Tabuľka podporovaných rámcov v platforme GAE

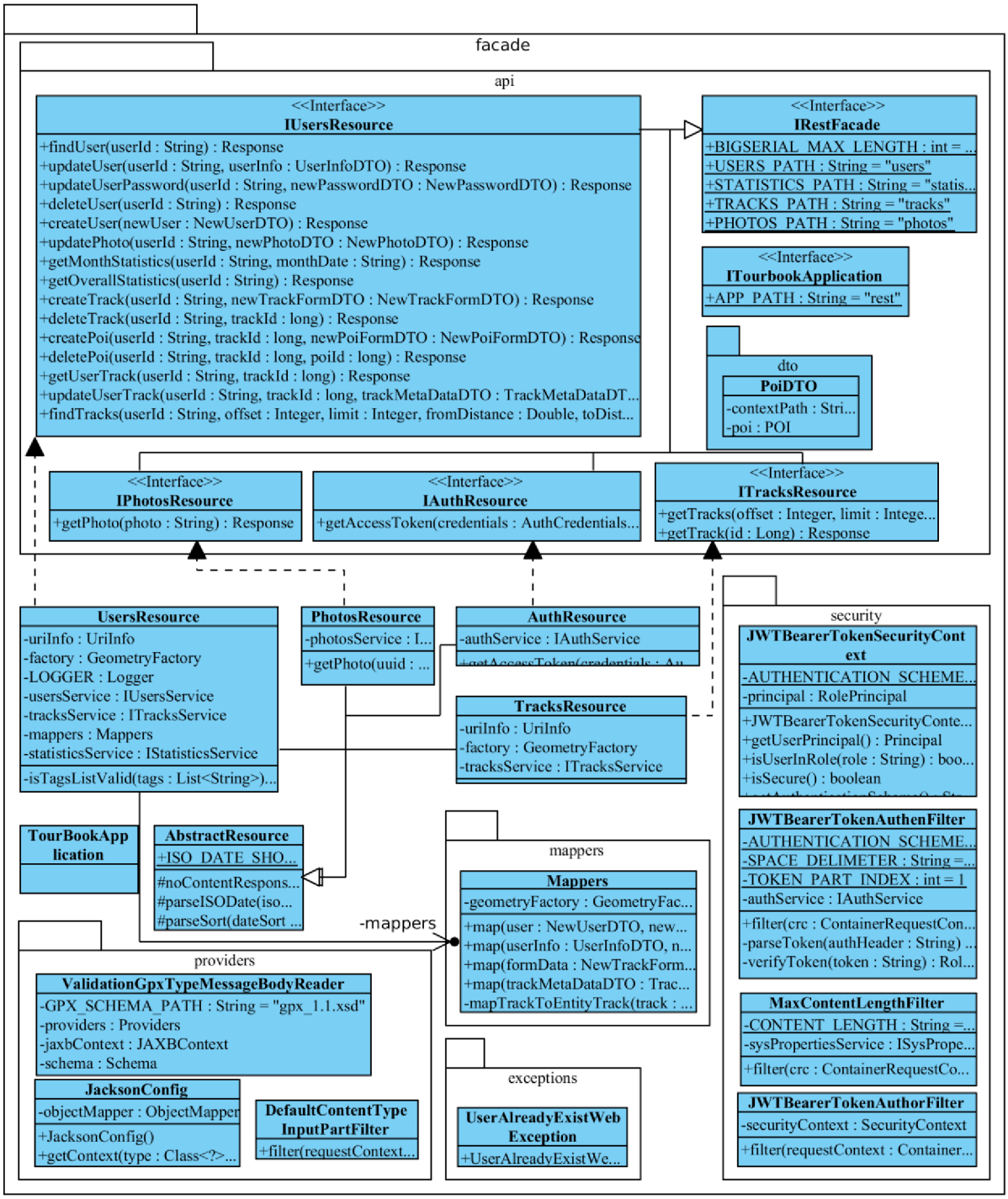
Príloha C: Implementačné diagramy



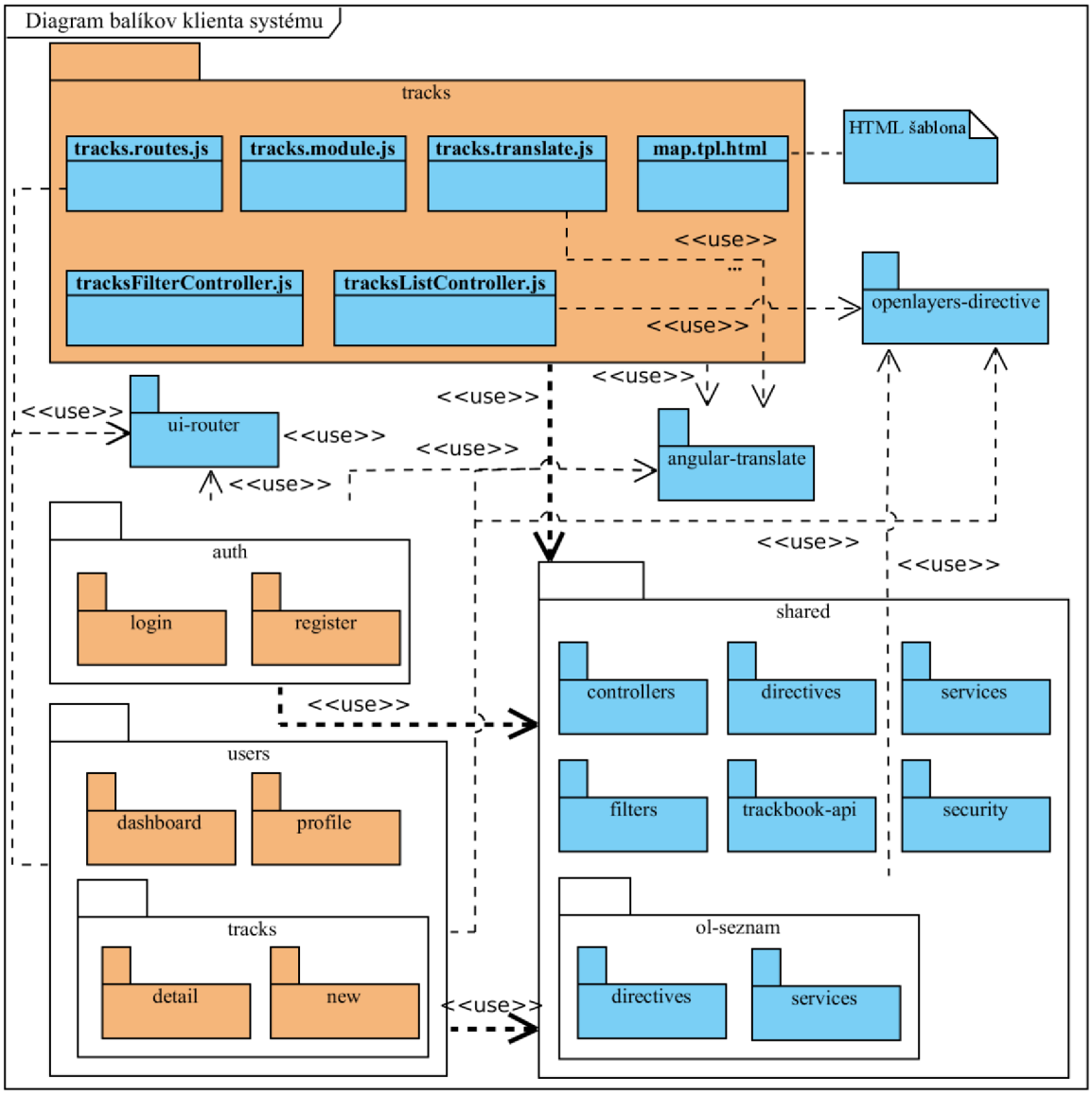
Obrázok C.1 Diagram tried doménovej vrstvy webovej služby systému



Obrázok C.2 Diagram tried servisnej vrstvy webovej služby

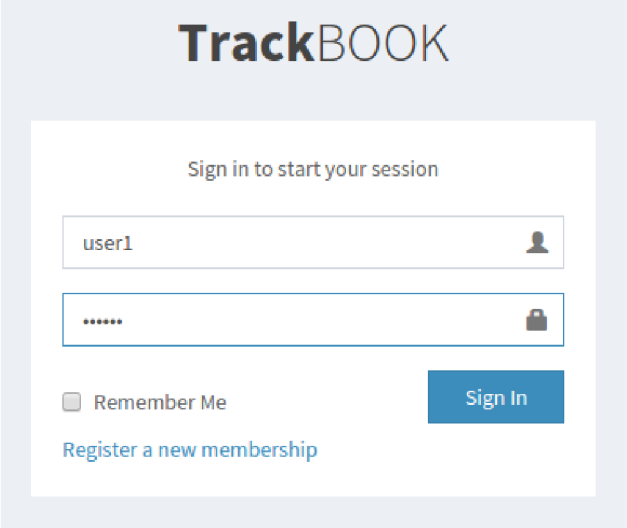


Obrázok C.3 Diagram tried fasádnej vrstvy webovej služby systému

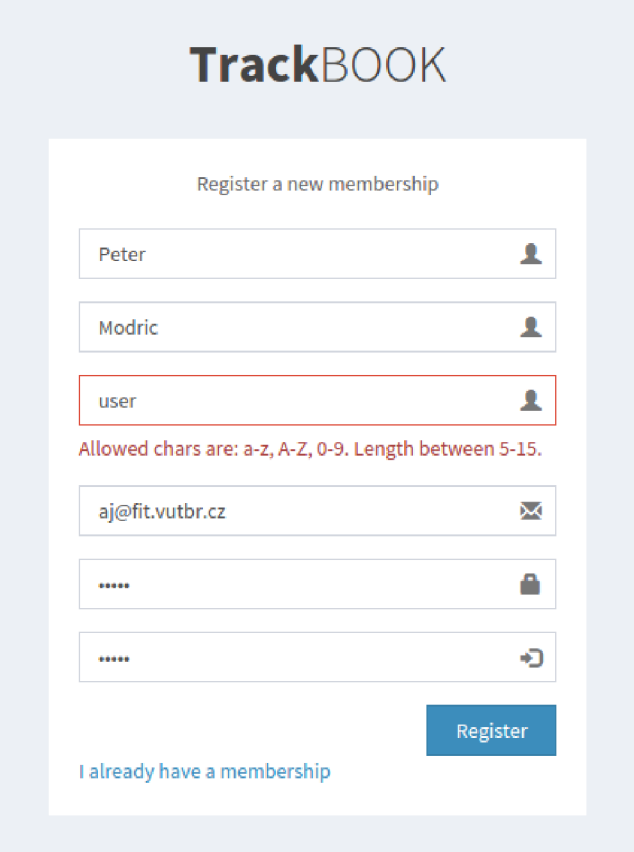


Obrázok C.4 Diagram balíkov jazyka UML zobrazujúci moduly webového klienta systému

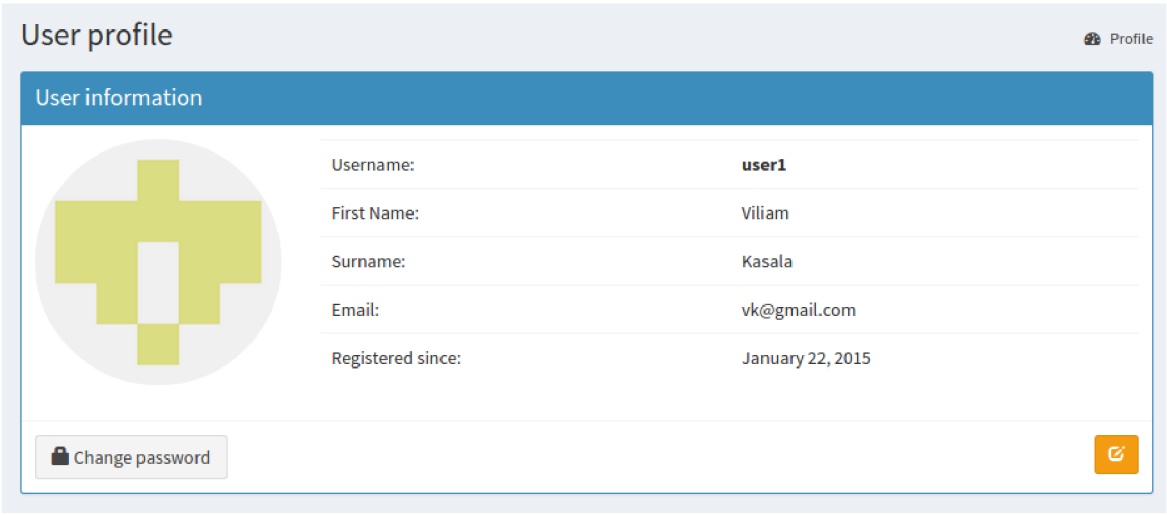
Príloha D: Snímky obrazoviek systému



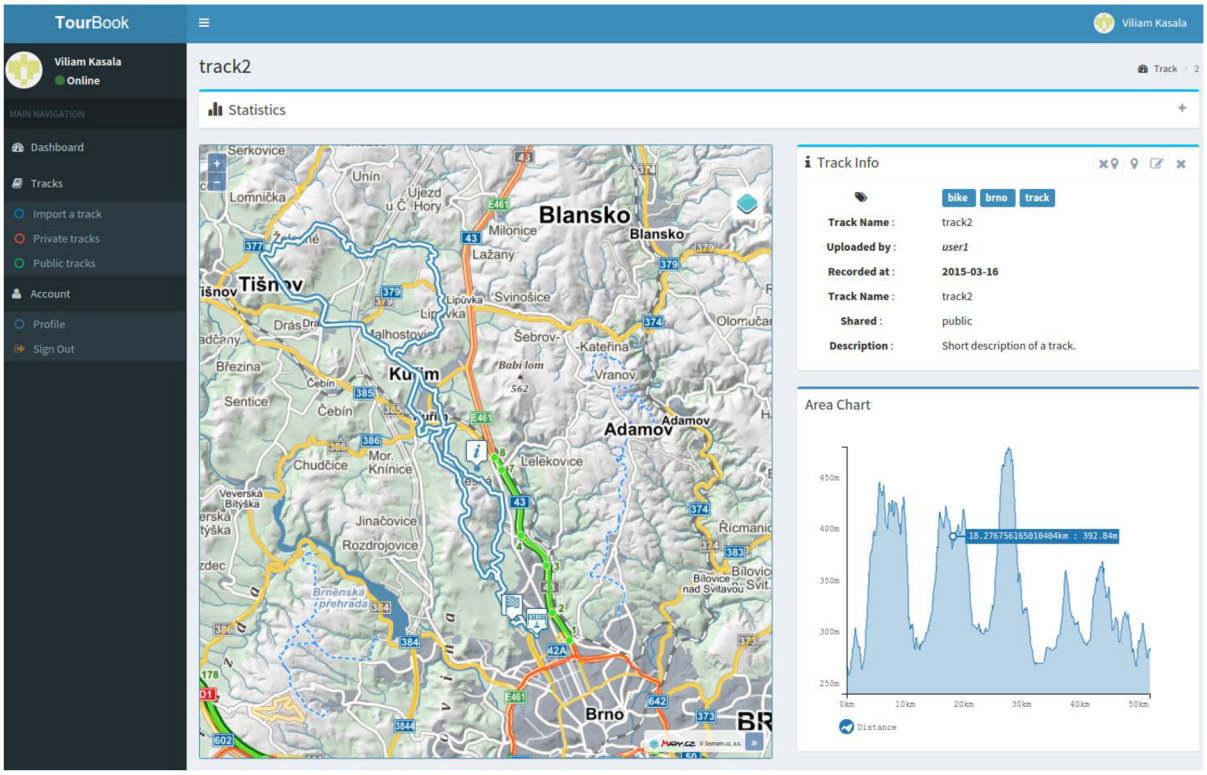
Obrázok D.1 Prihlasovacia obrazovka



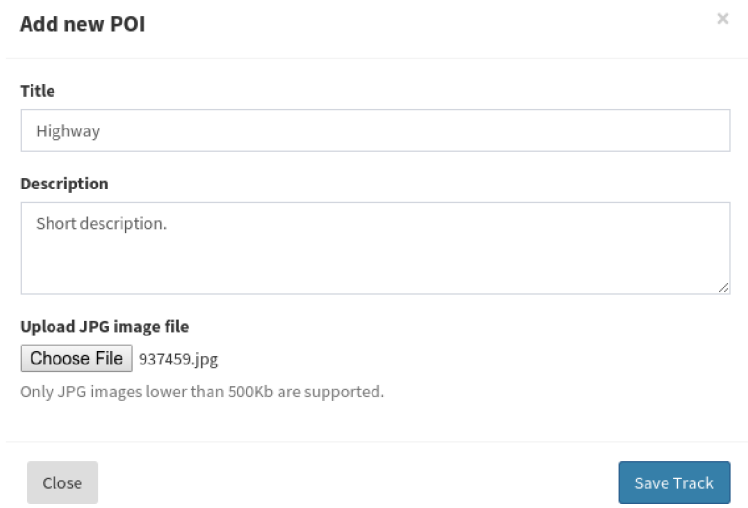
Obrázok D.2 Registračná obrazovka



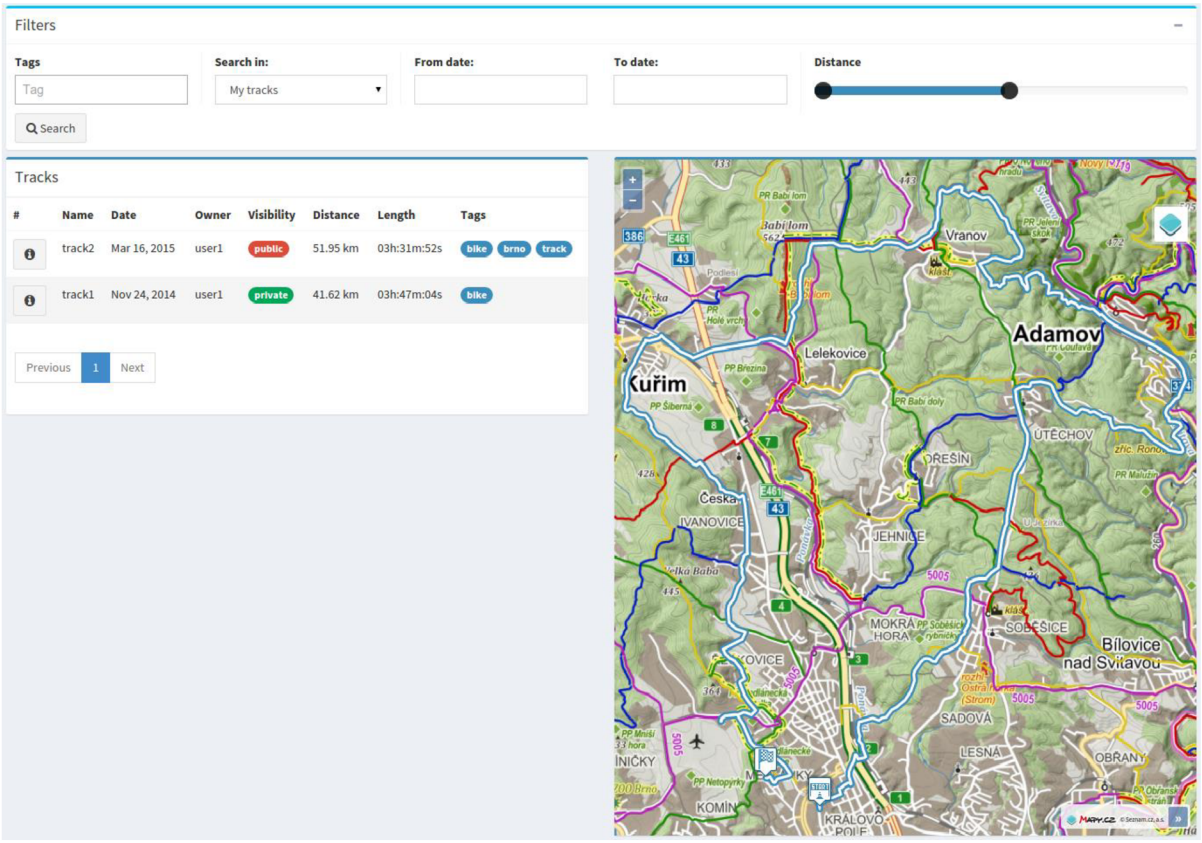
Obrázok D.3 Obrazovka profilu



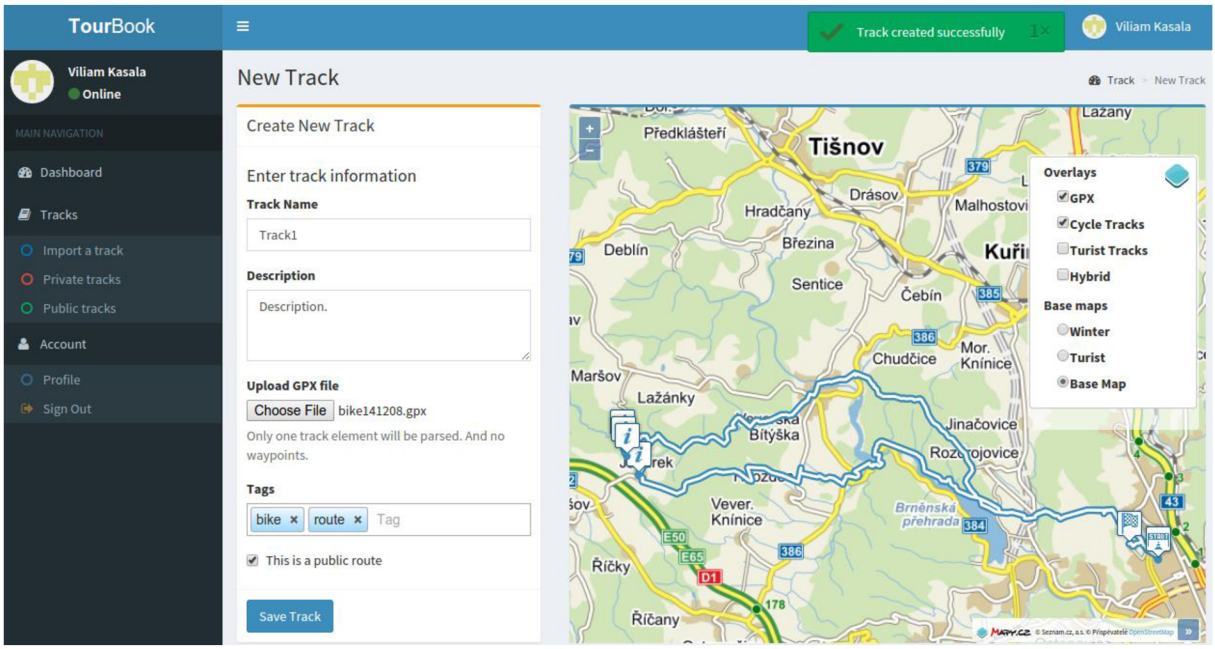
Obrázok D.4 Obrazovka detailu trasy



Obrázok D.5 Dialógové okno k vytvoreniu POI



Obrázok D.6 Obrazovka vyhľadania trás



Obrázok D.7 Obrazovka pridania novej trasy