České vysoké učení technické v Praze
Fakulta elektrotechnická
Katedra počítačové grafiky a interakce

Bakalárska práca

Systém pro automatické generování popisu cesty pro zrakově postižené

Jakub Bokšanský

Vedúci práce: Ing. Jan Vystrčil

Študijný program: Softwarové technologie a management, Bakalársky

Odbor: Web a multimedia

19. mája 2011

Poďakovanie

Ďakujem svojmu vedúcemu práce Ing. Janu Vystrčilovi a oponentovi Ing. Zdeňku Míkovci, Ph.D za cenné rady, motiváciu a konzultácie počas práce. Taktiež ďakujem všetkým v organizácii SONS, ktorí sa podieľali na príprave testovania a Otovi Procházkovi za vytvorenie prototypu, z ktorého práca vychádzala a za pomoc pri jeho nasadení do prevádzky.

Prehlásenie

Prehlasujem, že som prácu vypracoval samostatne a použil som iba podklady uvedené v priloženom zozname.
Nemám závažný dôvod proti použitiu tohto školského diela v zmysle §60 Zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon).

V Prahe dňa 19. 5. 2011 .....

Abstract

This work deals with a problem of generating city routes text description for visually impaired people. An existing application prototype is used as a basis and enhanced in two ways. Pathfinding system is added and quality of generated text is evaluated and improved. Final implementation is deployed on live server and tested with end users, which are navigation center operators in Czech blind united organisation.

Abstrakt

Práca pojednáva o probléme generovania popisu cesty mestom pre zrakovo postihnutých. Ako základ je použitý existujúci prototyp takej aplikácie a je zdokonalený v dvoch smeroch. Je pridaný systém na hľadanie trasy a je posúdená kvalita popisu, ktorú sa práca snaží zlepšiť. Výsledná implementácia je nasadená na verejne dostupnom serveri a otestovaná s koncovými užívateľmi, ktorými sú operátori navigačného centra v SONS (Sjednocená organizace nevidomých a slabozrakých).

Obsah

Zoznam obrázkov

Zoznam tabuliek

Kapitola 1

Úvod

1.1 Zrakovo postihnutý

Podľa [31] je „základnou podmienkou samostatného, bezpečného pohybu a možnosti orientácie zrakovo postihnutého človeka v prostredí dostatok informácií, ktoré má možnosť získať z prostredia. Tieto informácie musia byť dostupné a teda rešpektovať zdravotné postihnutie nevidiaceho alebo slabozrakého“.

Zrakom získa zdravý človek až 80% všetkých informácií. Každé poškodenie zraku obmedzuje možnosti vnímania reality a okolitého prostredia. U nevidiacich sa tak stretávame s lepšie rozvinutými ostatnými zmyslami, hlavne hmatom a sluchom. Základom pre fungovanie navigačného systému je získanie dostatočného množstva informácií o trase.

Obr. 1.1: Priechod pre chodcov s varovným a signálnym pásom v tvare písmena T

1.1.1 Získavanie informácií

Navigačný systém, ktorý by dokázal automaticky generovať popis musí vychádzať z rovnakých informácií ako zrakovo postihnutý, ktorý sa snaží v prostredí zorientovať. Informácie získava prostredníctvom hmatu, sluchu, čuchu a čiastočne zraku. Nie každé zrakové postihnutie spôsobuje úplnú stratu zraku, preto aj narušený zrak môže v obmedzenej miere pomôcť pri orientácii. Záleží na druhu a stupni postihnutia, ktorý zmysel nevidiaci používa viac, a ktorý menej. Je to u každého individuálne a preto by sme mu mali poskytnúť čo najviac informácií rôzneho charakteru. Je na každom jedincovi, ako tieto informácie dokáže využiť vo svoj prospech.

Informácie získané hmatom (taktylné informácie) sú všetky informácie, ktoré nevidiaci získava fyzickým dotykom s okolím. Najčastejšie bruškami prstov, dlaňou, chrbtom ruky, pri chôdzi chodidlom a slepeckou palicou. Dokáže tak zistiť typ podkladu, po ktorom chodí, steny, kraj chodníka, obrubník a pod. V exteriéri navyše môže využiť teplo od slnka, ktoré cíti na tvári.

Obr. 1.2: Varovný pás na zastávke MHD

Informácie získané sluchom sú minimálne tak dôležité ako hmatové. Nevidiaci pomocou nich dokáže určiť tvar a veľkosť priestoru (podchody, tunely), v ktorom sa nachádza (echolokácia - odraz zvuku od prekážok). Pomôžu mu aj zvuky áut, ozvučené semafory a kroky iných chodcov, ktorých sa môže pridržiavať.

Informácie získané zrakom Ak stretneme na ulici človeka s bielou palicou, je pravdepodobnejšie, že je to človek, ktorý má ešte zrakovú funkciu aspoň čiastočne zachovanú. Označujeme ich ako prakticky nevidiacich. Ich zrak je značne obmedzený alebo nespoľahlivý. Je však možné ho využiť pre zvýšenie ich vlastnej bezpečnosti a netreba ho podceňovať.

Informácie získané čuchom Jedná sa o charakteristické pachy, ktoré v meste nájde v blízkosti miest ako reštaurácie alebo parky. Aj tieto informácie môžu nevidiacemu pomôcť.

1.1.2 Zdroj informácií

Ako zdroj informácií bol zvolený projekt OpenStreetMap, viac o ňom pojednáva kapitola 1.5. Potom, ako získame potrebné informácie, musíme ich nevidiacemu podať v takej forme, aby ich mohol efektívne využiť. Na to je potrebné pochopiť techniky, ktoré používa k svojmu pohybu a orientácii.

1.2 Pohyb a orientácia nevidiaceho

Článok [30] uvádza najčastejšie techniky, ktoré nevidiaci využívajú pri pohybe v meste. Tou prvou je technika dlhej bielej palice. Okrem týchto existujú aj ďalšie spôsoby, viac informácii napr. v [11].

Obr. 1.3: Nebezpečne umiestnený označník zastávky

Trailing Pri tejto technike sa nevidiaci chrbtom ruky dotýka steny, popri ktorej chodí. Tento pohyb sa nepovažuje za veľmi bezpečný v exteriéri, pretože nevidiaci má len obmedzenú možnosť kontrolovať priestor pred sebou. Môže dôjsť aj k zraneniu ruky, ktorou sa

dotýka steny. Preto sa používa na pohyb v interiéri. Na zabránenie zranenia sa doporučuje používať horný alebo dolný bezpečnostný postoj [11]. Tieto postoje chránia nevidiaceho pred nárazom do prekážok. Nevidiaci si správnym držaním ruky kryje buď tvár a hlavu (horný bezp. postoj) alebo oblasť pásu (dolný bezp. postoj). Je možné ich používať súčasne.

Technika dlhej bielej palice Táto technika je najčastejšie používanou pre pohyb v meste, tiež má však svoje nedostatky. Pomocou bielej palice totiž nevidiaci nedokáže zistiť vyššie položené predmety (obr. 1.3). Jedná sa o telefónne automaty, dopravné značky, poštové schránky, smetné koše bez reliéfneho označenia a ďalšie podobné predmety.

Okrem týchto dvoch postupov môže nevidiaci využiť služby vodiaceho psa. Ten ale nerozhoduje o smere cesty (okrem vopred naučených trás) a nie je vhodný pre každého. Ideálnym riešením je doprovod sprievodcu. Tých je však málo a odkázanosť na cudziu osobu spôsobuje u nevidiaceho traumu z vlastného obmedzenia. Naše úsilie by malo preto smerovať k tomu, aby bolo prostredie navrhnuté s ohľadom na prístupnosť pre postihnuté osoby. Stávajú sa tak sebestačnými a značne to zvyšuje kvalitu ich života.

1.2.1 Prostredie vnímaním nevidiaceho

Prostredie je pre nevidiaceho možné vykresliť ako súbor orientačných bodov, vodiacich línií a orientačných znakov [29].

Obr. 1.4: Reklamný pútač ako dočasná prekážka v nedovolenej blízkosti vodiacej línie

Orientačné body sú všetky miesta, ktoré dokáže nevidiaci rýchlo identifikovať pomocou palice a nášľapom. Je možné ich ľahko rozlíšiť napr. podľa zvuku pri dotyku bielej palice, odporu proti pohybu, drsnosti povrchu atď. Ubezpečia nevidiaceho, že sa nachádza na známom mieste.

Vodiace línie Jedná sa o steny budov, obrubník, rozhranie chodníka a trávnika a ďalšie. Môžu byť aj umelo vytvorené na uľahčenie pohybu nevidiacich. Príklad na obrázku 1.4. Nevidiaci s vodiacou líniou udržuje stály kontakt.

Orientačné znaky sú skutočnosti, ktoré dokresľujú prostredie. Je to napr. typ povrchu chodníka, šum stromov, hluk z dopravy, zvuk potoka a pachy na ulici.

1.2.2 Orientácia v exteriéri

Je značne odlišná od orientácie v interiéri. Exteriér (mestské prostredie) obsahuje nebezpečenstvo v podobe áut, pracovných stavieb, neoznačených prechodov a ďalších. Okoloidúci narozdiel od ľudí v budove majú menší záujem o pomoc nevidiacemu tým, že by ho sprevádzali – kontakt na ulici je menej osobný.

Obr. 1.5: Varovný pás označujúci bezpečnostnú vzdialenosť od okraja nástupišťa

Riešenia, ktoré uľahčujú pohyb telesne postihnutým môžu paradoxne skomplikovať pohyb zrakovo postihnutým. Chodníky bez obrubníkov síce uľahčia pohyb imobilným ľuďom, ale nevidiaci prichádzajú o vodiacu líniu. V tomto prípade je dôležité zvoliť kompromis alebo nájsť také riešenie, ktoré vyhovuje obom.

Pre orientáciu v exteriéri sú typicky používané nasledovné prvky [13]:

- **Vodiace línie:** styk trávnika a chodníka, múr budovy, zábradlie, zvýšený obrubník. Obrubník chodníka pri vozovke pre autá sa nepovažuje za vodiacu líniu, aj keď ho tak často nevidiaci používajú. Pre nevidiacich je to životu nebezpečné a architekti by sa takému riešeniu pre nevidiacich mali vyhnúť.

- **Varovný pás:** má typické výstupky a musí byť farebne kontrastný k okoliu. Ohraničuje nebezpečné miesta a varuje pred vstupom do tohoto priestoru, viz obr.1.5 a obr.1.2.

- **Signálny pás:** Označuje významné miesta. Upozorňuje na zastávky MHD, prechody pre chodcov (obr. 1.1), vchody do nebytovej budovy, a podobne ...
- **Akustické majáky [28]:** svojim signálom označujú významné miesta. Zvyknú byť umiestnené aj pri vchodoch do významných budov, či v MHD. Staršie vydávali zvuk nepretržite, modernejšie typy sú ovládané vreckovou vysielačkou priamo na slepeckej palici. V MHD oznamujú majáky číslo, smer linky a nasledujúce stanice. Informačný maják v budove má zaznamenanú hlasovú informáciu o priestore, napr. „Vstupujete do haly. Prístup k nástupištiam je v priamom smere, na madlách sú označené čísla nástupíšť“. Na semaforoch rozlišujú červenú a zelenú.

1.3 Druhy zrakového postihnutia

To, ako sa zrakovo postihnutý dokáže orientovať v priestore závisí na druhu a stupni jeho zrakovej vady [14]. Kým niektoré vady umožňujú relatívne normálny život, iné spôsobujú neschopnosť rozoznávať predmety, tváre, či písmo. S ťažšími poruchami zraku človek dokáže vnímať už len v obmedzenej miere priestor, rozdiel medzi svetlom a tmou, alebo pri úplnej strate zraku nemá žiadne zrakové pocity. Zrakovo postihnutí netvoria jednoliatu skupinu, poruchy zraku sa delia podľa rôznych hľadísk [10]:

- Podľa doby vzniku
  - vrodené
  - získané
- Podľa druhu
  - slabozrakosť
  - zvyšky zraku
  - slepota
  - poruchy binokulárneho videnia
- Podľa dĺžky trvania
  - krátkodobé
  - opakujúce sa
  - dlhodobé

Zrakové poškodenie sa môže prejaviť v 3 smeroch:

- zníženie zrakovej ostrosti
- obmedzenie alebo poškodenie zrakového poľa
- porucha farebného videnia

Druhy zrakových porúch

Slabozrakosť je značné zníženie zrakovej ostrosti u obidvoch očí. Vzniká predovšetkým následkom očných ochorení alebo zranením zrakového orgánu. Spôsobuje pomalšie a nepresné (skreslené) vnímanie. Pri zrakovej práci sa takto postihnutí rýchlo unavia. Napriek oslabenému videniu ostáva zrakový analyzátor pri získavaní informácií vedúcim.

Zvyšky zraku Jedincov so zvyškami zraku je ťažké zaradiť do jednej zo skupín. Vymedzenie skupiny hraničí na jednej strane so slepotou a na druhej s normálnym videním. Niekedy sa označujú termínmi čiastočne vidiaci alebo prakticky slepí.

Slepota je úplná strata zrakových pocitov. Môže postihovať len jedno oko a môže byť vrodená alebo nadobudnutá (úrazom, chorobou, ...). Za slepcov sa označujú aj nevidiaci, ktorí rozlišujú len svetlo a tmu.

Poruchy binokulárneho videnia ako napríklad tupozrakosť, škúlenie a ďalšie tvoria najpočetnejšiu skupinu. Najcharakteristickejším dôsledkom je porucha vnímania priestoru a priestorových vzťahov.

1.4 SONS

Občianske združenie *Sjednocená organizace nevidomých a slabozrakých* (SONS) [6] pôsobí v ČR od roku 1996. Funguje na území celej Českej republiky. Pobočky a odborné strediská má vo väčšine okresov a dnes už združuje viac než 10 000 členov. Poskytuje informácie, rady, pomoc pri hľadaní zamestnania, cvičí vodiace psy, proste „učí žiť v tme“. Zastrešuje mnoho projektov, ktoré majú uľahčiť život nevidiacim, zoznam ukončených aj bežiacich projektov je na ich hlavnej stránke [6]. Jedným z nich je **navigačné centrum** [4].

1.4.1 Navigačné centrum

Je to malé call centrum, ktoré poskytuje nevidiacim viacero služieb. Jednou z nich je navigácia z miesta na miesto. Uľahčiť poskytovanie tejto služby je cieľom tejto bakalárskej práce.

Súčasný stav služby V súčasnosti funguje služba nasledovným spôsobom. Nevidiaci zavolá do call centra (res. pošle email) s žiadosťou o vytvorenie popisu cesty z jedného miesta na druhé. Ideálne aspoň 2 dni vopred, aby mali operátori dosť času na tvorbu popisu. Operátori pomocou máp, satelitných snímkov a služby Google Street View vytvoria popis. Nevidiacemu potom operátor popíše telefonicky cestu. Ak je to možné, snaží sa využiť linky MHD. Musí taktiež vyhovieť špeciálnym požiadavkám nevidiaceho a vyhnúť sa miestam pre neho nebezpečných.

Stav služby po nasadení vytváranej aplikácie Vyhľadávanie kvalitných mapových podkladov a tvorba popisu je časovo náročná. Cieľom projektu je uľahčiť prácu operátorom tým, že zautomatizujeme tento proces. V hotovej aplikácii operátor vyberie na mape trasu a systém automaticky vygeneruje jej popis. Od operátora stačí, aby ho skontroloval a upravil do lepšie čitateľnej podoby.

Takýto systém bol navrhnutý a jeho prototyp implementoval Ota Procházka vo svojej diplomovej práci [15]. **V mojej bakalárskej práci sa zameriam na ďalšie rozvíjanie tohto prototypu**.

1.5 OpenStreetMap

Projekt OpenStreetMap (OSM) sa venuje vytvoreniu voľne šíriteľnej mapy sveta. Založil ho v júli r. 2004 Steve Coast [8], v tom čase frustrovaný tým, že neexistuje mapa sveta, ktorá by nebola zaťažená obmedzujúcimi licenciami. Projekt OSM je založený na kolektívnej spolupráci komunity a na koncepcii open source. Užívatelia môžu editovať mapové dáta a tým sa podieľať na vytvorení celosvetovej mapy.

Obr. 1.6: Logo projektu OpenStreetMap

Licencia pre OSM dáta

OSM mapy sú dostupné pod licenciou CC-BY-SA 2.0 [9]. Jedná sa copyleftovú licenciu navrhnutú neziskovou spoločnosťou Creative Commons určenú pre umelecké a iné autorské diela. Dáva právo dielo použiť a modifikovať s tým, že musí byť uvedený autor diela a nesmie sa ďalej distribuovať pod inou licenciou. Môže sa distribuovať pod rovnakou alebo podobnou licenciou a to aj za účelom predaja.

Súčasný stav projektu sleduje [25]. Od leta roku 2004, kedy bolo zmapované prvé miesto zanesené do OSM (Regents Park, Londýn) sa mapa rozrástla a dnes (Apríl 2011) pokrýva väčšinu osídlených území sveta. K tomu pomohli aj štáty, napr. Kanada, ktorých vlády poskytli svoje geografické údaje na použitie v OSM a dobrovoľníci, ktorí prispievajú k tvorbe mapy. Dobrovoľníkov, ktorí nejakým spôsobom prispeli do OSM sa nazbieralo už vyše 320 000.

Najjednoduchším spôsobom, ako prispieť je pomocou GPS zmapovať určitú oblasť, potom spracovať a nahrať tieto údaje. Návod, ako sa zapojiť je na stránke [26] v anglickom jazyku.

1.5.1 Technické pozadie OSM

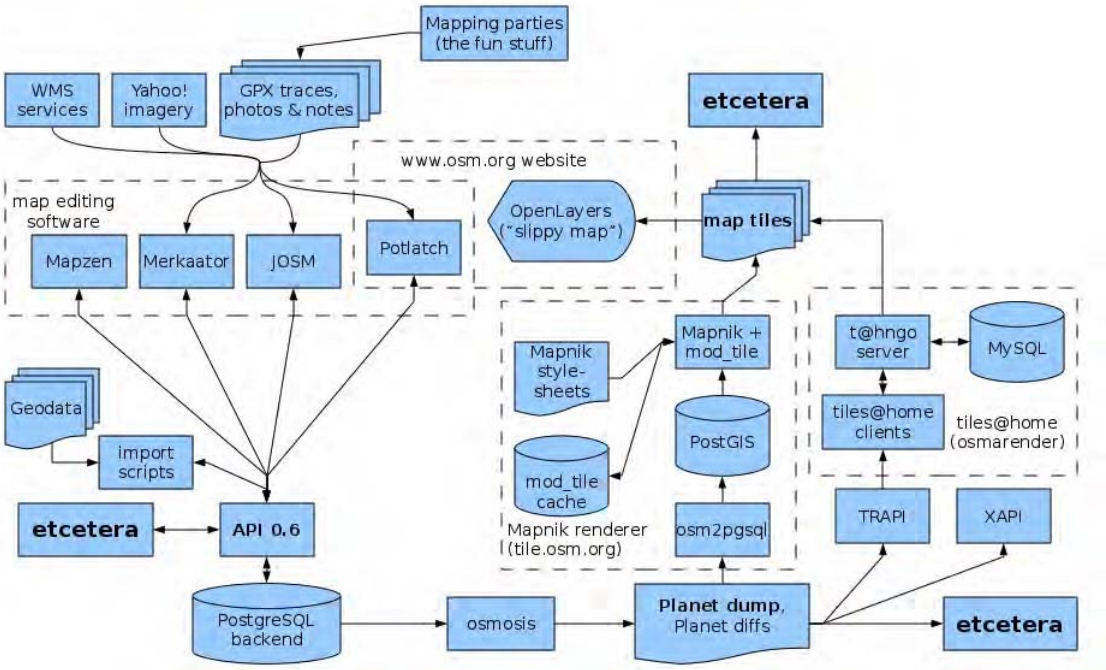
Projekt sa skladá z mnohých komponentov, ktoré zobrazuje diagram na obr. 1.8. Dôležitými komponentmi pre našu aplikáciu sú:

Obr. 1.7: Postup zapojenia sa do projektu OpenStreetMap

OSM API - Je aplikačné rozhranie dostupné ako webová služba. Umožňuje sťahovať a nahrávať nové dáta.

Aplikácia Mapnik - Konzolový generátor tiles pre slippy map (pozri kap. 1.5.5). Z OSM údajov vyrendruje obrázky pre mapu, ktorú je možné zobraziť na webovej stránke a prehliadať v rôznej mierke.

Aplikácia JOSM - Desktopová aplikácia napísaná v Jave, ktorá umožňuje editovať mapy, pracovať s GPS dátami a nahrať zmeny na OSM server. Viac na [21].



Obr. 1.8: Diagram komponentov OpenStreetMap

Výhodou, ktorou open source projekt disponuje je možnosť vyvíjať nové komponenty, prípadne vyvinúť vlastnú komponentu ako náhradu už existujúcej. Vymenované komponenty

sú len jedny z možných, ktoré sa dajú použiť. Na generovanie tiles je možné použiť napr. aplikáciu Osmarender [18] a na editovanie mapy napr. online editor Potlach [19].

1.5.2 OSM API

V súčasnosti vo verzii 0.6, OSM API [17] je webová služba, ktorá sprístupňuje aplikačné rozhranie projektu OSM na Internete. Na jeho použitie nie je potrebná registrácia. Jednotlivé metódy sú namapované na URL, ktoré stačí zavolať správnym HTTP dotazom.

Na vyskúšanie stačí obyčajný internetový prehliadač. Otvorením URL `http://api.openstreetmap.org/api/0.6/capabilities` získame informácie o schopnostiach a nastaveniach OSM API servera. Pre vývojové účely sa nedoporučuje používať hlavný server **api.openstreetmap.org**, ale testovací **api06.dev.openstreetmap.org**.

Ako formát odpovedí sa používa XML a štandardné čísla HTTP odpovedí. To znamená, že v kontexte tohoto aplikačného rozhrania neznamená dobre známy návratový kód 404 (not found) neexistujúcu stránku, ale že danému dotazu nezodpovedá žiaden prvok v databáze.

1.5.3 Dátová vrstva OSM

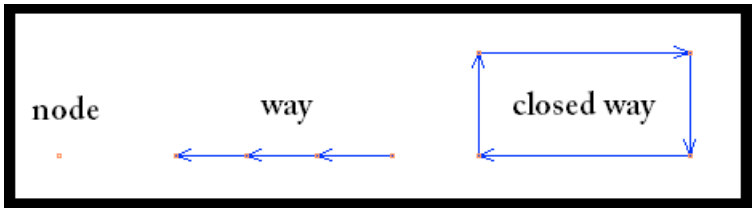
Mapa sa skladá z mapových **elementov**, každý z nich môže byť bližšie určený **tagmi**. Tagy určujú, čo element znamená a aké sú jeho vlastnosti.

Elementy mapy Dôležitým aspektom z hľadiska technického riešenia generátora popisu je spôsob, akým je mapa uložená. Dáta, ktoré dokážeme z OSM API dostať sú súhrnne označované ako **data primitives** [24]. Tie sa delia na:

Node - uzol – základný stavebný prvok mapy, predstavuje bod. Môže to byť bod na ceste, alebo samostatne stojaci bod.

Way - cesta – je usporiadané spojenie aspoň dvoch bodov. Reprezentujú cesty, chodníky, železničné trate, ale aj ohraničené oblasti (parky, jazerá, ...). Môžu byť uzavreté (vznikne uzavretá oblasť).

Relation - vzťah – pomocou neho sa vyjadrujú vzťahy medzi uzlami alebo cestami. Skupina ciest môže reprezentovať linku električky, cyklistický chodník alebo iné, viac na [23].



Obr. 1.9: Grafické znázornenie OSM elementov uzol, cesta a špeciálny prípad uzavretej cesty

Tagy Tagy nie sú dátové elementy, ale sú neodlučiteľnou súčasťou mapových dát. Pomocou tagov je možné z abstraktného bodu, alebo cesty vytvoriť telefónnu búdku, poštovú schránku, diaľnicu, železnicu alebo kultúrne pamiatky. Stačí použiť správny tag alebo kombináciu tagov.

Kombinácia tagov môže bližšie určiť, o aký objekt sa jedná. Tag **highway** = **crossing** priradený ceste určuje, že sa jedná o prechod pre chodcov. V kombinácii s tagom **crossing** = **uncontrolled** vznikne prechod neriadený žiadnym svetelným značením.

Jeden tag je teda kombinácia **kľúč** = **hodnota**. Príklad XML reprezentácie jednosmernej ulice s názvom Clipstone Street pomocou tagov ukazuje obrázok 1.10.

1.5.4 Nekonzistencia tagov v OSM

Problém vzniká, keď narazíme na objekt, pre ktorý neexistuje žiaden tag (resp. nevieme zistiť, aký tag by sme mali použiť). OSM tento problém rieši tak, že umožňuje užívateľom vložiť akýkoľvek tag, ktorý nemusí zodpovedať žiadnemu štandardu. Tak vzniká nekonzistentné značenie. Pre rovnaký objekt používajú rôzni užívatelia rôzne tagy. Existuje zaužívaná a doporučená množina tagov pre najbežnejšie objekty, viz. [20], ale jej použitie nie je vynucované.

```
<way id="5090250" visible="true" timestamp="2009-01-19T19:07:25Z"
     version="8" changeset="816806" user="Blumpsy" uid="64226">
    <nd ref="822403"/>
    <nd ref="21533912"/>
    <nd ref="821601"/>
    <nd ref="21533910"/>
    <nd ref="135791608"/>
    <nd ref="333725784"/>
    <nd ref="333725781"/>
    <nd ref="333725774"/>
    <nd ref="333725776"/>
    <nd ref="823771"/>
    <tag k="highway" v="unclassified"/>
    <tag k="name" v="Clipstone Street"/>
    <tag k="oneway" v="yes"/>
</way>
```

Obr. 1.10: XML reprezentácia cesty s množinou bodov a tromi tagmi.

1.5.5 Tiles a slippy map

Slippy map [27] je systém použitý na zobrazenie mapy na webovej stránke, viz obr. 1.11. Je to klasický systém, aký poznáme napríklad z Google Maps.

Mapu môžeme posúvať, zväčšovať a programovo ovládať pomocou Javascriptu. Po technickej stránke je riešená pomocou Javascriptu a technológie AJAX. Pomocou AJAX dokáže slippy map načítať zo servera jednotlivé časi mapy vtedy, keď na ne presunieme pohľad.

Obr. 1.11: Slippy map používaná na hlavnej stránke OSM.

Tieto časti mapy označujeme ako **tiles**. Sú to obrázky, z ktorých poskladáme mapu tým, že ich umiestnime vedľa seba. Musia byť vopred vygenerované pre každú oblasť mapy a pre každú možnú mierku.

1.6 Existujúci prototyp aplikácie na generovanie popisu

Základom, z ktorého bude moja práca vychádzať je funkčný prototyp aplikácie na generovanie popisu. Vzhľadom k tomu, že prototyp implementuje mnoho funkcií a vyriešil mnoho problémov s komunikáciou so zdrojmi mapových dát, s nasadením na serveri a ďalšie, je vhodným základom pre ďalšiu prácu. Hotové riešenie prototypu obsahuje:

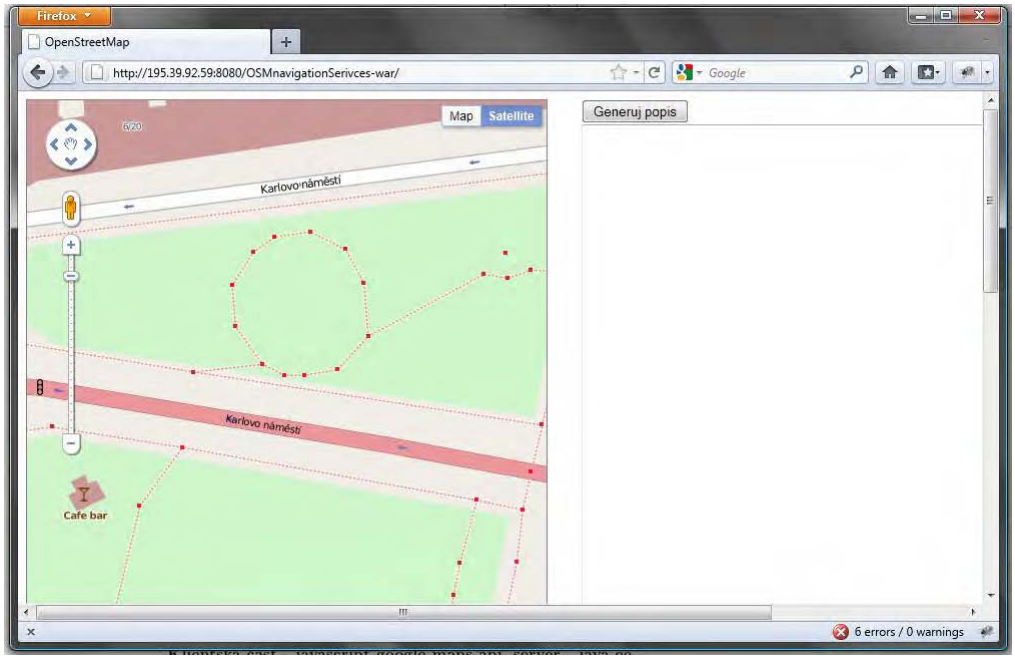
- **Server** ako obraz disku (image) virtuálneho stroja určeného pre virtualizačný nástroj VMWare. Image obsahuje funkčnú inštaláciu všetkých komponentov potrebných na prácu s OpenStreetMap.
- **Klientskú aplikáciu OSMNavigationServices** Je to Enterprise Java aplikácia, vyvíjaná ako projekt v NetBeans. Na spustenie tejto aplikácie je potrebný aplikačný server Glassfish (je súčasťou inštalácie NetBeans) alebo podobný server.

Server Návod na spustenie serveru vo VMWare je v prílohe F. Takýto typ nasadenia je vhodný napr. na vývojové účely. Po spustení je k dispozícii OSM API, nástroj na generovanie mapových podkladov Mapnik a databáza PostGIS[1] s geografickými dátami. Podrobný

[1]PostGIS je rozšírenie pre databázu PostgreSQL na prácu s geografickými dátami

model servera a popis všetkých nainštalovaných komponentov pokrýva [15].

Aplikácia OSMNavigationServices je typická Enterprise Java aplikácia. Obsahuje serverovú časť (Enterprise Java Beans kontajner) a klientskú časť (Web application kontajner). Serverová časť beží na aplikačnom serveri[2] a klientská časť v internetovom prehliadači s ňou komunikuje prostredníctvom webových služieb.



Obr. 1.12: Webová stránka klientskej časti aplikácie

1.6.1 Typický prípad použitia prototypu

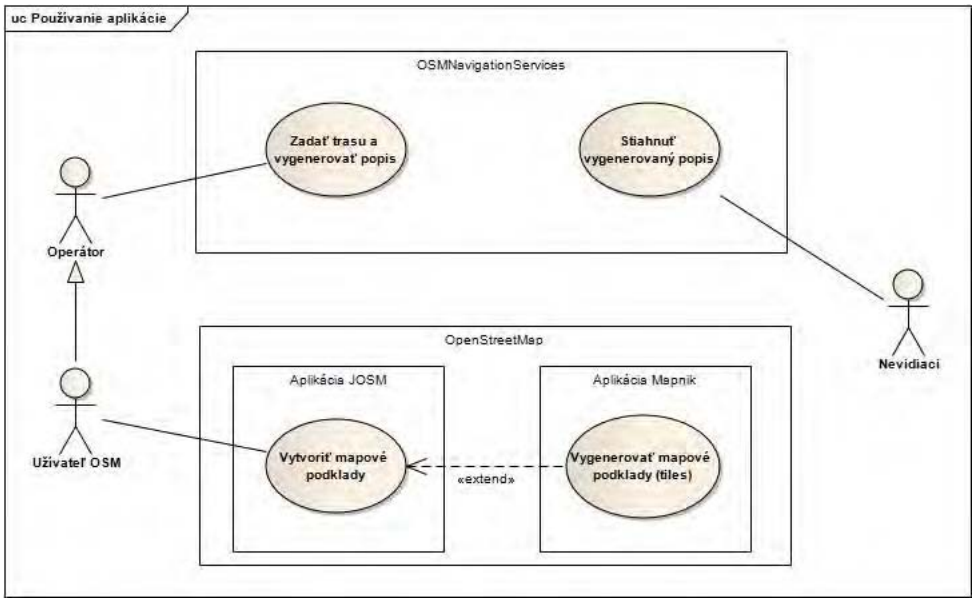
Táto časť popíše všetky hlavné komponenty prototypu pomocou prípadu použitia (use case), ktorý je na obr. 1.13. Tento prípad zobrazuje typickú situáciu, akým sa bude aplikácia reálne používať a preto sa do neho zapoja všetky dôležité časti aplikácie.

Klientská časť je webová aplikácia (obr. 1.12). Realizuje zadávanie trasy do systému a umožňuje stiahnuť vygenerovaný popis. Mapa je zobrazená pomocou Google Maps API. Nezobrazuje sa ale mapa, ktorú poskytuje Google. Používa sa mapa uložená na našom OSM serveri. Užívateľ v nej vyberie trasu, nastaví parametre generovania trasy a nechá vygenerovať popis. V tomto kroku sa zapojí serverová časť aplikácie.

V serverovej časti prebieha vytvorenie popisu cesty podľa zadanej trasy. Tu bude ďalej implementovaný systém pre automatické vyhľadávanie trasy.

Predpokladom pre fungovanie systému je existencia mapových podkladov v systéme OpenStreetMap. V prípade, že mapové podklady ešte neexistujú, alebo nie sú dostatočne

[2] Ten však nemusí bežať na tom istom fyzickom stroji ako beží server s OSM API a ďalšími službami



Obr. 1.13: Typický prípad použitia aplikácie na generovanie popisu

podrobné, môže ich operátor (alebo každý užívateľ OSM) vytvoriť. Na to je potrebné vykonať dva kroky. Zadať mapové dáta do OSM a vygenerovať mapové podklady (tiles). Tieto kroky je možné vykonať pomocou dvoch aplikácií:

JOSM Desktopová aplikácia JOSM je voľne šíriteľný editor OSM máp. Ako podklady pre kreslenie máp môže zobraziť satelitné snímky, je ale potrebné dávať pozor na ich licenciu. Niektoré snímky totiž nemôžu byť použité ako podklady na kreslenie máp. Viac o tejto problematike pojednáva zdroj [15] na str. 9. Po zmapovaní oblasti a zanesení do OSM tento program dokáže nahrať dáta na OSM server.

Mapnik Generátor mapových podkladov
Návod, ako tieto kroky správne vykonať v prototype našej aplikácie je v prílohe D.

1.6.2 Vývojové nasadenie prototypu

Pre vývoj aplikácie je server spustený na lokálnom počítači. Aplikácia je spustená v NetBeans, ktorý obsahuje inštaláciu serveru Glassfish. Pre testovanie s užívateľmi bude použitý server umiestnený priamo v SONS. Model nasadenia v SONS je uvedený v prílohe C.

Kapitola 2

Popis problému

Cieľom bakalárskej práce je identifikovať nedostatky existujúceho prototypu a odstrániť ich. Vychádzam pri tom z návrhov a nedostatkov, ktoré popisuje Ota Procházka vo svojej diplomovej práci [15] a z vlastných testov prototypu. Po odstránení týchto nedostatkov by mal byť prototyp pripravený na ďalší vývojový cyklus (testovanie – analýza – implementácia).

2.1 Nedostatky prototypu - riešené problémy

Prototyp aplikácie obsahuje problémy, ktoré je potrebné odstrániť, aby mohla byť aplikácia nasadená do prevádzky. Niektoré problémy sú nedostatky, ktorými trpí kvalita popisu. Ďalšie problémy sú spôsobené tým, že ešte neboli implementované niektoré funkcie, napríklad routing.

2.1.1 Routing

Jedným z najväčších nedostatkov prototypu je nevhodný postup, akým operátor zadáva do mapy trasu, ktorej popis chce vygenerovať. Operátor kliká na jednotlivé body, ktorými má trasa prechádzať, pričom naráža na nasledovné problémy:

- Je možné omylom zadať trasu, ktorá nevedie po chodníku (t.j. po ceste pre vozidlá, električkovej linke, železnici alebo diaľnici), viz obr. 2.1.
- Optimálnu (najkratšiu, najbezpečnejšiu, ...) trasu musí operátor odhadnúť a zadať „manuálne“
- Systém nespojí dva body po chodníku, ale spojí ich priamo – vzdušnou čiarou. Je potrebné zadať každý bod zvlášť tak, aby cesta viedla po chodníku.

Postup je tak zdĺhavý a náchylný na chyby. Odstránenie týchto nedostatkov je kľúčové pre používanie a testovanie systému s užívateľmi. Požadované riešenie by malo odstrániť uvedené problémy a zároveň implementovať systém výberu trasy tak, aby bral ohľad na preferencie zrakovo postihnutého užívateľa.

Obr. 2.1: Trasa pretínajúca vozovku pre autá, správne by mala viesť podchodom

Optimálna trasa v takom prípade neznamená vždy najkratšiu trasu. Je potrebné vyhýbať sa miestam, ktoré by mohli byť pre nevidiaceho nebezpečné, aj za cenu dlhšej trasy. Podrobne sa tomuto problému venovala kapitola 1.2 o orientácii nevidiacich. Na riešenie sú tak kladené nasledovné funkčné požiadavky. Riešenie musí:

- nájsť optimálnu trasu medzi dvoma zadanými bodmi
- ak je to možné, použiť chodník pre chodcov
- umožniť operátorovi nastaviť preferencie[1] pri hľadaní trasy.
- dostatočne rýchla odozva systému pri hľadaní trasy

Ďalšou nefunkčnou požiadavkou je integrácia do prototypu aplikácie. Aj keď by bolo možné použiť niektorú externú webovú službu, žiadna z nich zatiaľ neimplementuje navigáciu pre nevidiacich. Ďalším dôvodom je, že prototyp využíva vlastný zdroj OSM dát. V prípade externej služby, ktorá používa hlavný OSM server by sme museli riešiť problém s nekonzistentnými dátami.

2.1.2 Kvalita popisu

Súčasná podoba generovaných popisov nie je ideálna pre použitie bez dodatočných úprav. Nemôžeme predpokladať, že strojovo generovaný text bude dosahovať kvalitu textu vytvoreného človekom. Avšak mnohé z nedostatkov, ktoré generátor textu má, je možné odstrániť. Zjednoduší a zrýchli sa tak úprava výsledného textu operátorom.

[1] Vymedzenie miest, ktorým sa chce nevidiaci vyhnúť

Aby bolo možné začať riešiť tento problém, bolo potrebné identifikovať čo najviac problémov, ktoré popisy obsahujú. Generovaním popisov rôznych trás sa podarilo identifikovať 6 rôznych typov problémov (findings). Pre každý problém boli zvolené viaceré prípady (test case), kedy sa chyba prejaví, aby bolo možné porovnať popis pred a po zavedení riešenia a zistiť tak zlepšenie.

Problémy generátora, ktoré sa podarilo identifikovať:

- **Finding 1** – Nevhodné dĺžky segmentov
- **Finding 2** – Duplicitné zobrazenie objektov na trase
- **Finding 3** – V texte sú uvedené aj zbytočné informácie, ktoré sa netýkajú navigácie
- **Finding 4** – Nerozpoznané objekty a ich interpretácia
- **Finding 5** – V texte sa objavuje pokyn na pokračovanie o nula metrov
- **Finding 6** – Problém s otočením nevidiaceho, keď je začiatok trasy na križovatke

Finding 1 – Nevhodné dĺžky segmentov

Generátor rozdeľuje postup tým istým smerom na veľký počet malých úsekov (alebo príliš dlhý úsek nerozdelí). Stane sa tak, že nevidiaci dostane pokyn pokračovať vpred o malú vzdialenosť (1–5 metrov) niekoľko krát po sebe (obr. 2.2). Taký krátky postup by sa dal nahradiť jedným dlhším segmentom.

Opačným extrémom je pokyn na pokračovanie o rádovo desiatky metrov. Nevidiaci takú dlhú vzdialenosť nedokáže odhadnúť a potrebuje ju rozdeliť na menší počet úsekov.

```
Napříč Vaší cestou vede park. Podloží pod Vámi je chodník, cesta mírně stoupá.
Následující postup: Jděte rovně 25.0 metrů

Otočte se velmi mírně vpravo. Vlevo od Vás je nyní význačné místo typu
fountain. Následující postup: Pokračujte rovně 5.0 metrů.

Následující postup: Pokračujte rovně 5.0 metrů.

Následující postup: Pokračujte rovně 4.0 metry.

Následující postup: Pokračujte rovně 4.0 metry.
```

Obr. 2.2: Popis s nevhodne rozdelenými segmentami

Finding 2 – Duplicitné zobrazenie objektov na trase

V popisoch sa objavuje situácia, kedy sa popis jedného objektu objaví dva krát po sebe.

```
Napříč Vaší cestou vede jednosměrná hlavní třída, název Karlovo náměstí,
provoz tramvají; jednosměrná hlavní třída, název Karlovo náměstí, provoz
tramvají. Následující postup: Jděte rovně 47.0 metrů.
```

Obr. 2.3: V popise je hlavná ulica uvedená dva krát hneď po sebe

Finding 3 – V texte sú uvedené aj informácie, ktoré sa netýkajú navigácie

Generátor pridáva do textu aj informácie, ktoré nemajú pre navigáciu zmysel. Situácia vzniká tak, že generátor interpretuje všetky tagy, ktoré sú s objektom na mape zviazané. Takže aj tie, ktoré označujú autora objektu, použitý program na vytvorenie objektu a pod.

```
Vpravo od Vás je stěna domu. Napříč Vaší cestou vede {[created_by:Potlatch 0.9c;
type:multipolygon; ]}; Následující postup: Jděte rovně 21.0 metrů
```

Obr. 2.4: Popis s uvedenou zbytočnou informáciou - názvom mapovacieho programu

Finding 4 – Nerozpoznané objekty a ich interpretácia

Objekty, ktoré generátor popisu nepozná, interpretuje vo forme, ako v ukážke na obr. 2.5. Nikdy nebude možné „naučiť“ program rozoznávať všetky objekty. Súvisí to s tým, že OSM nedefinuje pevnú množinu tagov, pozri kapitolu 1.5.4 o nekonzistencii tagov.

Program však musí poznať najzákladnejšie objekty, ktoré sa často vyskytujú – reštaurácie, zastávky MHD, vchody do metra, rôzne obchody, telefónne búdky a ďalšie. Program by bolo vhodné neskôr rozšíriť o možnosť, aby operátor mohol „naučiť“ generátor správne interpretovať nové tagy.

```
Vlevo od Vás je {[name:Karlovo náměstí, Metro B; railway:subway_entrance;
created_by:JOSM; ]}; dům č.p.293/11; {[name:Karlovo náměstí, Metro B;
railway:subway_entrance; ]}; dům č.p.293/13; {[shop:chemist; name:dm; ]};
{[shop:supermarket; name:Albert; ]}
```

Obr. 2.5: Popis s nezrozumiteľne uvedenými informáciami

Finding 5 – V texte sa objavuje pokyn na pokračovanie o nula metrov

Tento problém vzniká tým, že dĺžka úseku sa zaokrúhli smerom nadol. Pre úseky kratšie než 0.5 m vznikne dĺžka úseku 0 metrov. Pokyn na prejdenie 0 metrov nemá žiaden zmysel a informácie, ktoré predchádzajú tento pokyn môžu byť pripojené k nasledujúcemu pokynu.

Finding 6 – Problém s otočením nevidiaceho, keď je začiatok trasy na križovatke

Ak trasa začína na križovatke – generátor vybere jeden možný smer, z ktorého by mohol užívateľ prísť a podľa toho generuje trasu. Popis trasy z obr. 2.7 začína pokynom „otočte se

```
Vpravo od Vás je jednosměrná ulice, název Odborů, počet jízdních pruhů 1,
parkování one transversal. Následující postup: Jděte rovně 0.0 metry

Otočte se vpravo. Vpravo od Vás je nyní stěna domu...
```

Obr. 2.6: Pokyn na pokračovanie o nula metrov

vpravo“, pretože generátor predpokladá príchod z ulice Odborů. To však nemusí byť pravda. Pop je uvedený na obr. 2.6.

Obr. 2.7: Trasa, na ktorej vzniká popis s pokynom pokračovať nula metrov

2.2 Existujúce riešenia routingu

V rámci projektu OpenStreetMap prebieha niekoľko projektov, ktoré sa venujú routingu. Bohužiaľ len málo sa ich venuje navigácii pre handicapovaných a žiaden pre zrakovo postihnutých[2]. Aktuálny stav týchto projektov monitoruje stránka [22].

2.2.1 Accessible Routing

Projekt sa momentálne nachádza v štádiu prototypu [16] a možnosť navigácie pre nevidiacich je uvedená ako nadchádzajúca funkcia. Väčšina funkcionality je stále v štádiu návrhu, zvyšná v štádiu prototypu. Tento projekt začal vznikať len nedávno a je ešte len v rannom štádiu.

[2]Existujú projekty, ktoré sa venujú kompletnému riešeniu generovania popisu pre nevidiacich, napríklad projekt Lorodux. Nás ale zaujíma routingová webová služba, ktorú by sme mohli využiť.

2.2.2 OpenRouteService

Webová stránka tohoto projektu [5] ponúka jednoduchú aplikáciu na vyhľadávanie najkratšej, respektíve najrýchlejšej trasy pre autá. Neplánuje sa implementovať vyhľadávanie trasy pre nevidiacich. Túto funkcionalitu má pokrývať projekt Accessible Routing. Svedčí o tom aj fakt, že odkazy na „routing pre handicapovaných“ sú nasmerované na stránku projektu Accessible Routing.

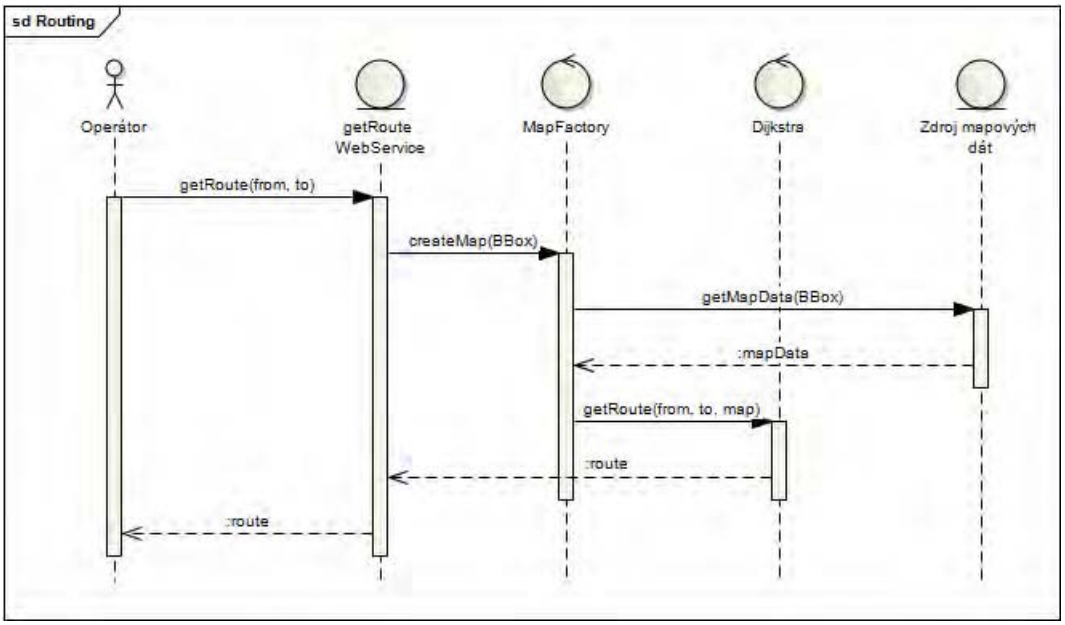
Kapitola 3

Analýza a návrh riešenia

3.1 Riešenie Routingu

Na zjednodušenie analýzy a implementácie je úloha, ktorú routing vykonáva popísaná ako **dotaz** od operátora systému. Ten sa dá rozdeliť na tri kroky, ktoré sa vykonajú v nasledujúcom poradí.

1. Vygenerovanie mapy pre vyhľadávací algoritmus
2. Vyhľadanie optimálnej cesty v mape
3. Zobrazenie cesty v klientskej časti aplikácie



Obr. 3.1: UML sekvenčný diagram dotazu na nájdenie trasy v mape

Riešením každého kroku samostatne nezávisle na ďalších vznikne modulárne riešenie. Takéto riešenie je výhodné z hľadiska testovania, keďže každú časť je možné testovať samostatne a v prípade potreby je možné jednotlivé časti nahradiť novou implementáciou bez väčších zásahov do systému.

3.1.1 Voľba vyhľadávacieho algoritmu

Aby sme mohli v mape vyhľadávať, musíme nad mapou spustiť vyhľadávací algoritmus. Keďže mapa sa dá reprezentovať ako cyklický graf, kde hrany sú ulice a uzly križovatky, je možné použiť jeden z algoritmov na prehľadávanie grafu. Medzi dvoch populárnych zástupcov týchto algoritmov patria algoritmy Dijkstra a A*. Problematiku výberu algoritmu a dátového zdroja (OSM) zoširoka pokrýva bakalárska práca [12].

Algoritmus Dijkstra Je zástupca neinformovaného prehľadávania do šírky [32]. Z toho vyplývajú dve zásadné vlastnosti, ktoré ho odlišujú od iných algoritmov. Beh Dijkstrovho algoritmu zachycuje pseudokód na obr. 3.2. Popis jeho vlastností:

- **Neinformovaný** – rozhodnutie, ktorým smerom „objavovať“ trasu skôr sa neriadi žiadnou informáciou, postupuje všetkými smermi rovnako rýchlo.
- **Do šírky** – graf prehľadáva tak, že najprv objavuje všetkých susedov aktuálne prehľadávaného uzla, potom prejde do prvého objaveného suseda a ten sa stáva aktuálne prehľadávaným uzlom. Tento krok sa opakuje, kým neprehľadá celý graf.

Z týchto dvoch vlastností vyplýva, že v každom kroku je v každom smere rovnako vzdialený (počtom uzlov) od počiatočného uzla. Logickým vylepšením algoritmu by bolo, aby postupoval smerom k cieľu väčšou rýchlosťou. To dokáže zabezpečiť algoritmus A*, ktorého popis nasleduje nižšie, alebo to dokážeme zabezpečiť pomocou **bounding boxu**, tak ako to ukazuje obr. 3.3. Tým, že počiatočný a cieľový uzol vymedzujú vyhľadávaciu oblasť, vždy budú v „rohoch“ oblasti. Takto jednoducho je zabezpečené, že sa oblasť bude prehľadávať rýchlejšie k cieľu, než iným smerom. Nevýhodou je, že bounding box nemusí trasu vôbec obsahovať, touto problematikou sa zaoberá kapitola 3.1.4.

Algoritmus A* A* je rozšírením algoritmu Dijkstra o použitie heuristky. Spadá tým do kategórie informovaného prehľadávania. Účelom heuristickej funkcie je vybrať najvhodnejší uzol na expandovanie. Týmto spôsobom algoritmus dospeje rýchlejšie k riešeniu. Podmienkou je ale použitie heuristiky, ktorá je vhodná pre daný problém[1]. Zároveň táto heuristika nesmie byť výpočtovo tak náročná, že by sa vyplatilo expandovať viac uzlov. Ak použijeme dobrú heuristiku, A* všeobecne expanduje menej uzlov, aby dosiahol cieľ. Je teda výpočtovo a pamäťovo menej náročný.

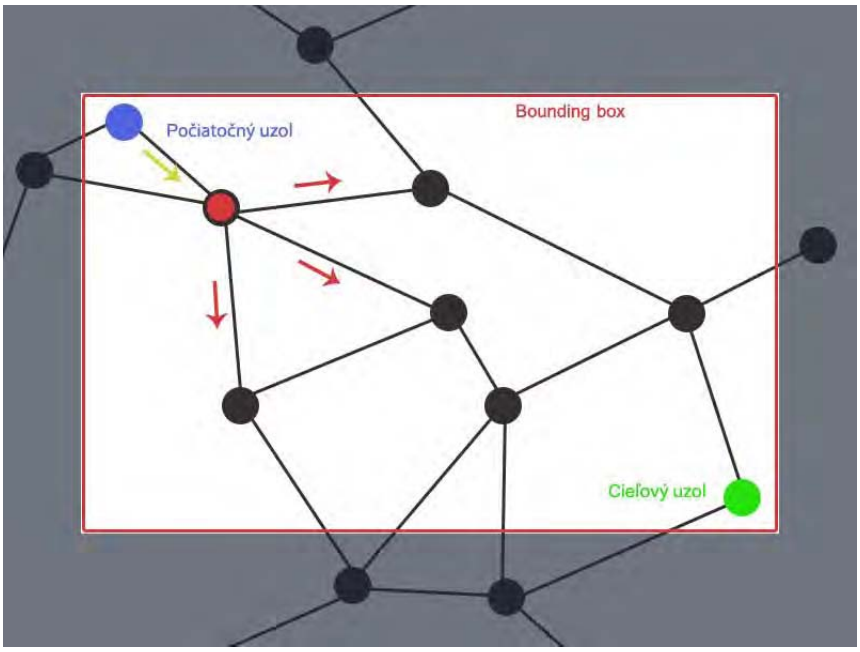
Z uvedeného popisu vyplýva, že A* je z hľadiska pamäťovej a výpočtovej zložitosti lepšia voľba než Dijkstrov algoritmus. Pre naše potreby je ale nevhodný. Na vytvorenie heuristiky pre A* musíme poznať odhad zostávajúcej vzdialenosti do cieľa. Práve odhad zostávajúcej vzdialenosti nie je z nášho grafu možné jednoducho zistiť (štandardne sa používa vzdušná vzdialenosť do cieľa).

[1] Nie pre každý problém je možné vytvoriť heuristiku

$$
\begin{array}{l}
\textsc{Dijkstra}(G, w, s) \\
\quad \text{INITIALIZE-SINGLE-SOURCE}(G, s) \\
\quad T \leftarrow \emptyset \\
\quad Q \leftarrow V[g] \\
\quad \textbf{while } Q! = \emptyset \\
\qquad \textbf{do} \begin{cases} u \leftarrow \text{EXTRACT-MIN}(Q) \\ S \leftarrow s \cup u \\ \textbf{for each } v \in \text{NEIGHBOURS}(u) \\ \quad \textbf{do } \text{RELAX}(u, v, w) \end{cases}
\end{array}
$$

Obr. 3.2: Pseudokód Dijkstrovho algoritmu

My totiž nehľadáme najkratšiu cestu, ale cestu ktorá spĺňa určité preferencie. Tie spočívajú v tom, že uzol, ktorý sa snažíme obísť ohodnotíme vysokou cenou. Rozloženie a cena týchto uzlov v grafe sa nedá „odhadnúť“. Neexistuje technika, ktorou sa dá predvídať architektúra mesta. Táto komplikácia spôsobuje, že algoritmus A* by mohol podľa štandardnej heuristiky vybrať uzol, ktorý je k cieľu bližšie, ale nie je vhodnou voľbou, pretože jeho nasledovník je uzol ohodnotený vysokou cenou.



Obr. 3.3: Bounding box ohraničujúci vyhľadávaciu oblasť

3.1.2 Backtracking algoritmus

Pre vyhľadávacie algoritmy je typické, že ich výstupom nie je cesta k cieľu, ale len logická hodnota, či graf cieľový uzol obsahuje[2]. Nás, ale zaujíma práve táto cesta. Tento problém rieši tzv. **backtracking**.

Dijkstrov algoritmus modifikujeme tak, aby si pri každom expandovaní nového uzlu zapamätal uzol, z ktorého sa do neho dostal. Pomocou toho sme schopní po dosiahnutí cieľového uzlu spätne zistiť trasu, po ktorej sme sa do neho dostali.

3.1.3 Preferencie

Dôležitou požiadavkou na routing je možnosť nastaviť preferencie hľadania trasy. Operátor určí prvky na mape, ktorým sa nevidiaci snaží vyhnúť. Často sa jedná o schody, prechody bez semaforu a podobné miesta, viz obr. 3.4. Na implementáciu tejto funkcie bol navrhnutý nasledovný systém.

Operátor pre každý taký prvok nastaví penalizáciu – dĺžku v metroch[3], ktorú je nevidiaci ochotný prejsť navyše, aby sa vyhol danému miestu. Miesta sa určia pomocou OSM tagov. Je na implementácii užívateľského rozhrania, či operátor bude zadávať priamo OSM tagy, alebo bude mať k dispozícii formulár s vymenovanými prvkami mapy, pre ktoré bude nastavovať penalizácie. Vhodná kombinácia oboch by bola pravdepodobne ideálnym riešením.

3.1.4 Vytvorenie dát pre vyhľadávací algoritmus

V tejto časti je využitý zdroj mapových dát. Získame z neho dáta, ktoré nám dokáže poskytnúť pre zadaný bounding box. Tieto sú potom spracované tak, aby nad nimi mohol Dijkstrov algoritmus vyhľadávať s ohľadom na preferencie.

Zdroj mapových dát je OpenStreetMap server. Do úvahy pripadajú 2 možnosti, ako ho použiť.

1. Použiť externý OSM server.
2. Použiť lokálny OSM server a PostGIS databázu.

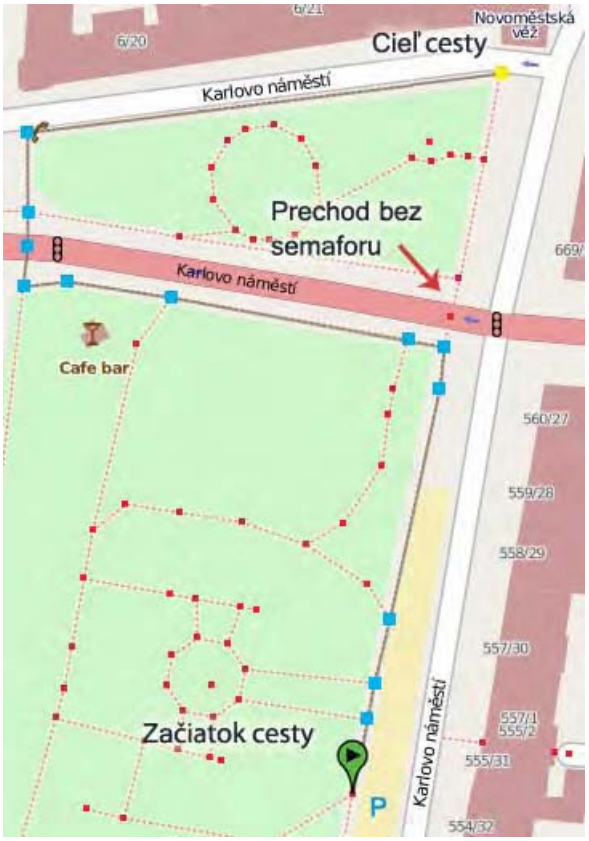
Externý server Použitie externého servera je lepšie z hľadiska aktuálnosti údajov. Takéto riešenie by navyše odľahčilo záťaž lokálneho servera. Nevýhodou je, že k dátam môžeme pristupovať jedine prostredníctvom OSM API, ktoré neumožňuje pokročilú prácu s geografickými dátami, napríklad výber všetkých elementov mapy v bounding boxe.

Lokálny server Prototyp aplikácie pre generovanie trasy využíva kombináciu lokálneho OSM servera a PostGIS databázy. Výhodou tohoto riešenia je, že PostGIS databáza umožňuje prácu s geografickými dátami.

Pri príprave dát pre Dijkstrov algoritmus využijeme výhodu PostGIS databázy. Z databáz potrebujeme získať oblasť ohraničenú zadaným bounding boxom, v ktorom sa má

[2] Výstupom Dijkstrovho algoritmu je navyše ohodnotenie grafu, ktoré každému uzlu priradí najkratšiu vzdialenosť od počiatočného uzla.

[3] Táto dĺžka sa v grafe pre Dijkstrov algoritmus prejaví ako cena za prechod uzla.



Obr. 3.4: Trasa, ktorú našiel algoritmus zohľadňujúci preferencie – obchádza prechod bez semaforu

vyhľadávať. PostGIS umožňuje vrátiť všetky geografické dáta, ktoré spadajú do štvoruholníkovej oblasti ohraničenej dvoma bodmi (bounding box). Pretože PostGIS neobsahuje všetky dáta (neobsahuje napr. tagy, ktoré bližšie určujú cesty a body) spojíme ich s dátami z OSM servera. Tak máme pripravené mapové podklady určenej oblasti pre ďalšie spracovanie.

Spracovanie dát

Mapové podklady je potrebné spracovať do grafu, nad ktorým dokáže Dijkstrov algoritmus vyhľadávať. Hrany grafu sú ohodnotené vzdialenosťou uzlov, ktoré spájajú. Vygenerovanie takéhoto grafu prebieha v 2 krokoch:

1. Stiahnutie potrebných dát z databáz
2. Vytvorenie grafu pre Dijkstrov algoritmus zo stiahnutých dát

V prvom kroku načíta z databázy všetky uzly a cesty nachádzajúce sa v zadanom bounding boxe. V druhom kroku tieto dáta prevedie do grafu. V tomto kroku je možné aplikovať preferencie nasledovne. Keď narazíme na prvok, ktorý obsahuje tag zaťažený penalizáciou - pripočítame k cene hrany, ktorá do neho vedie výšku tejto penalizácie.

Bounding box

Plocha bounding boxu je určená počiatočným a koncovým bodom. Táto plocha je potom zväčšená v každom smere o určitú hodnotu. Ideálna veľkosť je veľmi závislá na charaktere mapy. Čím menší bounding box, tým rýchlejšie sa trasa nájde, ale zvyšuje sa pravdepodobnosť, že trasu vôbec nebude obsahovať. Veľký bounding box spôsobí, že vyhľadávanie trvá dlhšie. Riešením je umožniť zadať bounding box ako parameter, ktorý môže používateľ routingu ľubovoľne zvoliť. Tým, že veľkosť bounding boxu nie je naprogramovaná uvonatvrdo vznikajú nasledujúce výhody.

- Operátor môže upraviť jeho veľkosť tak, aby mu vyhovovala.
- Klientská aplikácia môže implementovať gradientné vyhľadávanie – ak sa v malom bounding boxe trasa nie je, postupne sa zväčšuje, kým sa v ňom trasa nenájde.
- Jednoduchým testovaním len v rámci klientskej časti je možné zistiť optimálnu veľkosť bounding boxu, ktorá má byť prednastavená.

Vygenerovaný graf sa použije len raz a vytvára sa pre každý dotaz na routing znovu. Lepším riešením by bolo použiť už raz vygenerovaný graf opakovane. Narazíme ale na mnoho problémov:

- Mapové dáta mohli byť od posledného hľadania aktualizované.
- Dijkstrov algoritmus by vyhľadával aj trasu mimo určený bounding box (spojením grafu súčasného hľadania s predošlými). Aj keď to nie je nevyhnutne problém, spôsobí to nepredvídateľné správanie algoritmu - výsledky by boli závislé na predošlých hľadaniach.
- Preferencie, ktoré operátor zadáva môžu pri každom dotaze iné.

Posledný bod predstavuje najväčší problém. Jediná možnosť ako ho odstrániť je uložiť si tagy mapových elementov rovno do grafu. Vznikla by tak lokálna kópia mapy v pamäti servera. To však nie je potrebné. Našťastie takúto kópiu si za nás vytvára samo OSM API, ktoré cachuje všetky sťahované dáta. Navyše by sme celý graf museli prejsť, skontrolovať na penalizácie a prideliť ceny uzlom. Výpočtovo by to bolo rovnako náročné ako vytvoriť mapu znovu.

3.1.5 Komunikácia s klientskou aplikáciou

Ideálnou formou, ako zabezpečiť komunikáciu medzi klientskou a serverovou časťou systému je pomocou webovej služby (web service). Serverová a klientská časť aplikácie tak budú na sebe nezávislé. Získame tým výhody Service Oriented Architecture.

- Jednotlivé webové služby je v prípade potreby možné jednoducho nahradiť novou verziou bez zásahu do iných častí systému

- A/B testovanie – ak potrebujeme zistiť, ktorý algoritmus (A alebo B) je lepší, spustíme ich súčasne na dvoch webových službách. Niektorým klientom prezentujeme webovú službu A, ostatným B (stačí vymeniť URL služby). Podľa toho, ktorí budú spokojnejší vyberieme lepší algoritmus.

Protoyp aplikácie už obsahuje webové služby:

- **generateDescription** – vygeneruje popis zadanej trasy
- **getNearestPOI** – vráti súradnice najbližšieho bodu na mape vzhľadom k zadaným súradniciam
- **getDescriptionOfNearestPoint** – vráti popis najbližšieho bodu (Popis obsahuje presné súradnice bodu a tagy z OSM databázy s ním spojené)

3.2 Optimalizácie generátora popisu

Podľa zistených problémov s kvalitou popisu v kapitole 2.1.2 boli navrhnuté nasledovné riešenia. Každé riešenie (solution) rieši jeden problém (finding). Číslami solution zodpovedajú findingom.

Solution 1 – Nevhodné dĺžky segmentov

Riešením problému s rozdeľovaním úsekov je algoritmus, ktorý rozdelí príliš dlhé úseky na kratšie a spojí príliš krátke úseky do primerane dlhých. Ako úsek sa chápe časť textu, ktorá popíše prostredie a na konci vydá pokyn. Pokyn môže byť na pokračovanie nejakým smerom alebo otočenie sa. Týchto úsekov je v popise trasy niekoľko za sebou.

Algoritmus nesmie zasahovať do úsekov, ktoré podávajú informácie - upozorňujú na prechod, zmenu prostredia a pod. Využíva fakt, že vygenerovaný popis obsahuje mnoho úsekov, ktoré nepodávajú žiadnu informáciu (len pokyn na pokračovanie určeným smerom). Tieto je možné spojiť do jedného úseku.

Užívateľ má mať možnosť ovplyvňovať jeho chovanie. Na to boli navrhnuté dva parametre, ktorými sa algoritmus riadi – **treshold** a **desiredLength**[4] . Parameter treshold definuje maximálnu dĺžku úseku, ktorú môžeme akceptovať nerozdelenú. Parameter desiredLength stanovuje dĺžku úseku, ktorá je pre užívateľa optimálna. Na takúto dĺžku sa budú spájať/rozdeľovať nevhodne dlhé úseky.

Požiadavky na algoritmus:

1. Úseky dlhšie než treshold rozdeliť na úseky dlhé desiredLength.
2. Úseky kratšie než desiredLength, ktoré nasledujú za sebou zoskupiť do jedného úseku s dĺžkou desiredlength (Neplatí pre úseky obsahujúce informácie).
3. Posledný úsek môže byť dlhší než desiredLength, ale nesmie byť dlhší než treshold.
4. Ak úsek obsahuje informácie, musí ich algoritmus zachovať bez zmeny.

[4]Pre správne fungovanie algoritmu musí byť hodnota desiredLength nižšia než treshold

Solution 2 – Duplicitné zobrazenie objektov na trase

Tento problém je spôsobený postupom, ktorý generuje popis. Dva rôzne elementy OSM mapy, ktoré majú podobné tagy sa môžu prejaviť rovnakým textovým popisom. Riešenie tohoto problému je jednoduché. Stačí upraviť cyklus, ktorý vytvára výsledný popis.

Tento cyklus iteruje všetky položky, ktoré má do popisu zahrnúť. Stačí kontrolovať, či sa popis aktuálnej položky nerovná popisu predošlej. Ak áno – ignoruje aktuálny popis. Tak zabezpečíme, že bezprostredne za sebou nebudú nikdy nasledovať dva rovnaké popisy.

Solution 3 – V texte sú uvedené aj zbytočné informácie, ktoré sa netýkajú navigácie

Tieto informácie sú tagy mapových elementov prevedené na text. K tomu, aby sme zbytočné informácie odstránili stačí určiť tagy, ktoré nemajú orientačný význam a ignorovať ich. Jedná sa predovšetkým o tagy:

- **created_by** označujúci program použitý na jeho vytvorenie
- **layer**, ktorý určuje vrstvu, v ktorej sa element nachádza
- **type=„multipolygon“**

Systém by malo byť možné jednoducho rozšíriť o nové tagy, ktoré má ignorovať. Ideálne by ich mohol operátor pridávať z klientskej aplikácie.

Solution 4 – Nerozpoznané objekty a ich interpretácia

Jediným možným riešením je naučiť generátor popisu čo najviac elementov a tagov, ktoré dokáže správne interpretovať. Tak ako v predošlom riešení by mal byť systém ľahko rozšíriteľný o nové tagy. Zoznam často sa vyskytujúcich tagov a príklad ich správnej interpretácie ukazuje tabuľka 3.1.

Solution 5 – V texte sa objavuje pokyn na pokračovanie o nula metrov

Riešenie tohoto problému je jednoduché – stačí tento pokyn odstrániť z popisu úseku. Ak úsek obsahoval ďalšie informácie, algoritmus spájania úsekov sa postará o to, aby boli pripojené k nasledujúcemu úseku.

Solution 6 – Problém s otočením nevidiaceho, keď je začiatok trasy na križovatke

Tento problém stačí vyriešiť upozornením operátora, že začiatok popisu by mal otočiť nevidiaceho tak, aby pokračoval správnym smerom. To je úkon, ktorý sa nedá vynechať pri žiadnom popise, ale tam kde má možnosť vydať sa viacerými smermi je obzvlášť potrebný.

OSM Tag	Správny popis
[shop:travel_agency; name:Siam travel;]	Cestovní kancelář, název Siam Travel
[shop:computer; name:MC computer;]	Obchod s počítači
[shop:wine; name:Víno;];	Vinárna
[name:Albertov; railway:tram_stop;];	Zastávka tramvaje Albertov
[name:Karlovo náměstí, Metro B; railway:subway_entrance;]	Vchod do metra, linka B, stanice Karlovo Náměstí
[shop:chemist; name:dm;];	Drogerie DM
[shop:supermarket; name:Albert;];	Supermarket Albert
[amenity:post_box;];	Poštovní schránka
[amenity:telephone;];	Telefonní automat
[parking:transversal;];	Parkování příčně
[parking:crosswise;];	Parkování příčně
[parking:lengthwise;]	Parkování podélně
[surface:paving_stones;];	Dlažební kostky
[shop:outdoor; name:Humi;];	Outdoor shop Humi
[amenity:bank];	Banka
[amenity:atm;];	Bankomat
[parking:both;];	Parkování na obou stranách
[shop:hairdresser; name:Kadeřnictví solarium;];	Kadeřnictví
[landuse:village_green;]	Trávnatá plocha
[name:Alois Jirásek; historic:memorial;];	Památník Alois Jirásek
[amenity:bar;];	Bar
[amenity:parking;];	Parkoviště
[amenity:toilets;];	Veřejné WC

Tabuľka 3.1: Zoznam najčastejších tagov v mestskom prostredí

Kapitola 4

Realizácia

4.1 Implementácia Routingu

Routingový systém je implementovaný pomocou celej škály technológií. Základom je Enterprise Java aplikácia v rámci ktorej beží SOAP webová služba. Dotazy na routing jej posiela klientská časť aplikácie bežiaca v internetovom prehliadači.

Celý projekt je teda rozdelený na dve časti - serverovú a klientskú. Implementácia routingu zasahuje do obidvoch častí. Jej úlohou je zabezpečiť, aby server úspešne obslúžil dotaz na nájdenie trasy od klientskej aplikácie. Klientská aplikácia musí odpoveď spracovať a zobraziť operátorovi.

Implementácia sa dá popísať ako realizácia tohoto dotazu. V nasledujúcom texte je popísaná implementácia jednotlivých krokov dotazu.

4.1.1 Priebeh dotazu na routing

Priebeh dotazu je vo forme sekvenčného diagramu zobrazený na obr. 3.1. Dotaz sa začína tým, že operátor vyberie počiatočný a koncový bod. Potom automaticky prebehne volanie webovej služby routingu. Toto volanie prebieha v nasledovných krokoch:

1. Parametre hľadania sú v klientskej aplikácii sformátované do JSON reťazca.
2. Klientská aplikácia zavolá webovú službu routingu, ako parameter použije JSON reťazec z predošlého kroku.
3. Metóda bežiaca na serveri sa spojí s PostgreSQL databázou a OSM API a stiahne potrebné dáta pre Dijkstrov algoritmus.
4. Dijkstrov algoritmus vyhľadá trasu.
5. Výstup predošlého kroku je sformátovaný do JSON reťazca a odoslaný späť klientskej aplikácii.
6. Odpoveď je spracovaná Javascriptom a pomocou Google Maps API [1] zobrazená operátorovi na mape.

Na vygenerovanie popisu tejto trasy je totiž potrebné odoslať celú túto trasu (súradnice jej bodov) späť na server službe **generateDescription**. Vyznačená trasa je preto celá udržiavaná v pamäti klientskej aplikácie (aj mimo Google Maps API). Uložením trasy u klienta zabezpečíme, že nie je potrebné znova volať webovú službu routingu a potom službu generovania popisu. Tieto dva systémy sú na sebe nezávislé.

4.1.2 Webová služba routingu

Do prototypu bola pridaná služba s názvom **getRoute**. Je komunikačným rozhraním routingu. Na korektné vstupné hodnoty vráti zodpovedajúci výstup – nájdenú trasu.

Vstupné parametre služby

Nasledovné údaje je potrebné odoslať webovej službe ako parametre.

- Zemepisné súradnice počiatočného bodu (**lat1**, **lon1**)
- Zemepisné súradnice koncového bodu (**lat2**, **lon2**)
- Zemepisné súradnice 2 bodov ohraničujúcich bounding box, v ktorom sa trasa bude vyhľadávať (**bblat1**, **bblon1** a **bblat2**, **bblon2**)
- Zoznam preferencií (**penalizations**)

Zoznam preferencií sa a posiela ako JSON reťazec. Ten je zložený z položiek (preferencií), z ktorých každá vyjadruje penalizáciu pre jeden element mapy.

Každá preferencia je určená tromi kombináciami kľúč-hodnota. Dve z nich sú povinné a korektný JSON ich musí obsahovať.

- Povinný kľúč **key** a hodnota určujúca názov tagu, na ktorý sa penalizácia aplikuje.
- Povinný kľúč **distance** a jeho hodnota určujúca výšku penalizácie.
- Nepovinný kľúč **attribute**, ktorého hodnota určuje atribút OSM tagu, na ktorý má byť penalizácia aplikovaná. Ak nie je uvedené, aplikuje sa penalizácia na všetky tagy so zodpovedajúcim názvom. Nezáleží teda na hodnote atribútu.

Ukážka korektného JSON reťazca na obr. 4.1 obsahuje 2 penalizácie. Prvá stanovuje penalizáciu 50 m pre každé význačné miesto (amenity) typu *parking*, t.j. parkovisko. Druhá stanovuje penalizáciu 100 m pre akýkoľvek element mapy, ktorý obsahuje tag s názvom *railway*. Nezáleží pri tom na atribúte tohoto tagu, môže to byť železničná trať, alebo električkové koľaje.

Výstup webovej služby

Výstupom tejto služby je vždy zoznam bodov, ktoré tvoria nájdenú trasu vrátane počiatočného a koncového bodu. V prípade, že trasa v mape (alebo v časti mapy ohraničenej bounding boxom) neexistuje, vráti zoznam obsahujúci iba počiatočný a koncový bod. Bod je v zozname reprezentovaný jeho zemepisnými súradnicami. Zoznam je sformátovaný do JSON reťazca, jeho podoba je na obr. 4.2.

```
{
        "penalizations":
        [
             {
                 "key": "amenity",
                 "attribute": "parking",
                 "distance": "50",
             },
             {
                 "key": "railway",
                 "distance": "100",
             },
        ]
}
```

Obr. 4.1: Príklad JSON reťazca so zoznamom preferencií

4.1.3 Načítanie mapových dát

Mapové podklady sa získajú spojením dát z dvoch databáz. Našim cieľom je získať všetky body, ktoré potrebujeme a cesty, ktoré ich spájajú. Zoznam ciest, ktoré bounding box obsahuje získame jedným SQL dotazom do Postgis databázy (obr. 4.3).

Potom použijeme tento zoznam a z OSM API získame podrobnejšie informácie o týchto cestách. Tieto informácie obsahujú zoznam uzlov každej cesty. Ďalšími dotazmi do OSM API získame konkrétnu zemepisnú polohu každého uzla. Tu vzniká problém s veľkým množstvom dotazov do OSM API.

Komunikácia s OSM API prebieha pomocou HTTP protokolu. Každému volaniu funkcie OSM API zodpovedá jeden HTTP dotaz. Na zjednodušenie implementácie je použitá knižnica LibOSM2 [3] pre Javu. Tá mapuje HTTP dotazy na metódy v Jave.

Optimalizácia dotazov do OSM API V našej situácii, kde pre každý bod chceme zistiť jeho súradnice podľa jeho ID čísla, musíme poslať HTTP dotaz `GET /api/0.6/node/#id`. Pre veľké množstvo bodov je to ale značne časovo náročné.

Riešením je pripraviť si vopred zoznam všetkých bodov, pre ktoré potrebujeme zistiť súradnice a potom zavolať jeden krát funkciu `GET /api/0.6/nodes?nodes=123,456,789`. Tá vráti súradnice všetkých zadaných bodov naraz. Na prípravu tohto zoznamu musíme iterovať všetky nájdené cesty a spojiť zoznamy ich uzlov do jedného.

Popis dôležitých OSM API metód

GET /api/0.6/node/#id – Vráti XML reprezentáciu uzla

GET /api/0.6/node/#id/ways – Vráti všetky cesty, na ktorých daný uzol leží

GET /api/0.6/nodes?nodes=#id1,#id2,#id3 – Vráti XML reprezentáciu viacerých uzlov

```
{
    "points":
    [
        {"lat":50.077927, "lon":14.419642},
        {"lat":50.077755, "lon":14.419644},
        {"lat":50.0777, "lon":14.4201565},
        {"lat":50.077713, "lon":14.420341},
        {"lat":50.077694, "lon":14.4204},
        {"lat":50.077694, "lon":14.420452},
        {"lat":50.077705, "lon":14.420541},
        {"lat":50.077763, "lon":14.420623}
    ]
}
```

Obr. 4.2: JSON reťazec, ktorý je výstupom služby routingu

GET /api/0.6/ways?ways=#id1,#id2,#id3 – Vráti XML reprezentáciu viacerých ciest

```
SELECT osm_id FROM planet_osm_roads WHERE ST_Intersects(way,
    ST_Transform(ST_GeomFromText('POLYGON((14.419993125891779 50.078567779564764,
      14.421187675499823 50.078567779564764, 14.421187675499823 50.07711001584539,
      14.419993125891779 50.07711001584539, 14.419993125891779 50.078567779564764))',
      4326), 900913));
```

Obr. 4.3: Postgis SQL dotaz na všetky cesty v zadanom bounding boxe

4.1.4 Výmenný formát dát

Ako výmenný formát dát medzi klientskou a serverovou časťou bolo potrebné zvoliť jeden zo zaužívaných formátov. Medzi najznámejšie patrí XML a JSON.
Vlastnosti formátu XML:

+ široko používaný štandardizovaný formát

+ ľahko čitateľný pre ľudí

– **obtiažna implementácia v Javascripte, odlišnosti kódu pre rôzne prehliadače**

Vlastnosti formátu JSON:

+ úspornejší zápis než XML (z hľadiska veľkosti textu)

+ **jednoduché použitie v Javascripte - je možné pracovať s JSON reťazcom ako s poľom objektov**

– ťažšie čitateľný pre ľudí

– menej rozšírený

Pre komunikáciu bol zvolený formát JSON, hlavne pre jeho jednoduchšiu implementáciu v klientskej časti aplikácie (Javascript). V serverovej časti aplikácie (Java) je možné pracovať s JSON reťazcami pomocou štandardného balíka `org.json`. Pre pokročilú prácu s reťazcami, serializáciu Java objektov na JSON reťazce a späť je možné použiť voľne dostupnú knižnicu GSON [2].

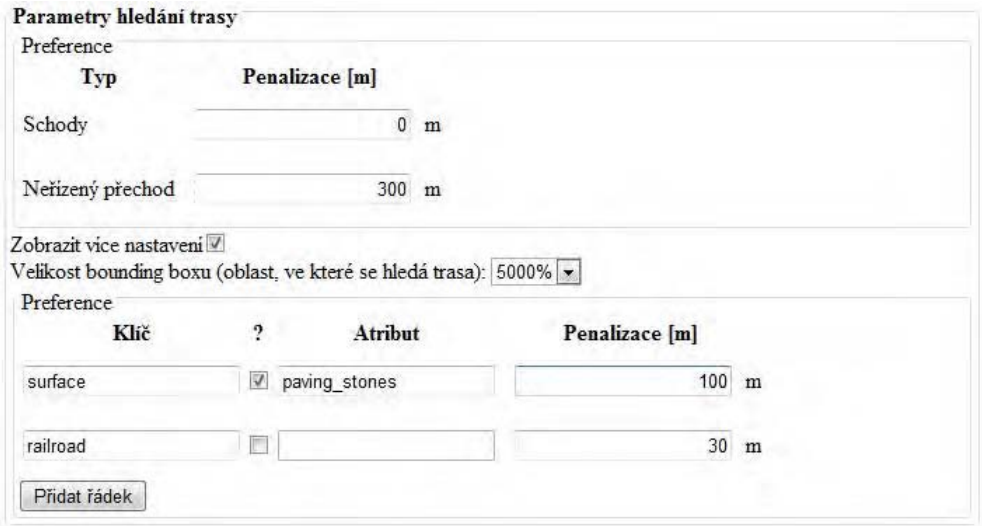
4.1.5 Klientská aplikácia

Prototyp klientskej aplikácie obsahuje jednoduché grafické rozhranie, viz obr. 1.12. Hlavným prvkom je mapa vytvorená pomocou Google Maps API. Pod mapou je formulár, ktorý umožňuje zadať preferencie hľadania.

Z hľadiska routingu je v klientskej časti dôležité, aby implementovala volanie webovej služby getRoute. Na toto volanie je použitá knižnica **Javascript SOAP Client** [7]. Tá používa technológiu AJAX na asynchrónne volanie webovej služby.

Spomínané knižnice vykonajú väčšinu práce, stačí ich vhodne využiť. Google Maps API nám po kliknutí do mapy poskytne zemepisné údaje tohoto bodu – vyšle udalosť *click*. Po odchytení aspoň dvoch týchto udalostí za sebou, máme zemepisné súradnice počiatočného a koncového bodu, medzi ktorými chceme nájsť trasu. Pomocou SOAP Client knižnice zavoláme webovú službu getRoute.

Služba berie ako vstupné parametre aj preferencie hľadania. Tie získame z HTML formuláru na obr. 4.4 a sformátujeme do JSON reťazca (viz obr. 4.1).



Obr. 4.4: Formulár na zadávanie penalizácií. Umožňuje zadávať aj OSM tagy.

Odpoveď webovej služby (JSON reťazec so zoznamom bodov) nie je potrebné nijak parsovať. Javascript dokáže s JSON reťazcom pracovať, ako by to bol objekt, t.j. pristupovať k

jeho premenným pomocou bodkovej notácie (dot notation) známej z objektového programovania. Zo zoznamu bodov vytvoríme Google Maps API objekt `google.maps.Polyline` a ten sa zobrazí na mape.

Jedinou úpravou, ktorá je vykonávaná so zoznamom bodov je odstránenie slučiek. Pri zadávaní trasy sa môže stať, že operátor zadá trasu, ktorá sa kríži s už existujúcou trasou. Nevidiaci by sa tak vrátil na miesto, kde už bol. Takúto slučku program identifikuje a odstráni.

Kapitola 5

Testovanie

5.1 Ciele testovania

Ciele testovania sú v zásade dva – overiť bezchybnosť implementácie a otestovať kvalitu implementovaných funkcií. Prvým cieľom sa bude zaoberať testovanie bez užívateľa, na druhý cieľ využijeme možnosť pracovať s koncovými užívateľmi systému – operátormi navigačného centra SONS. Z výsledkov testovania s užívateľmi budú potom vyvodené návrhy na zlepšenie funkcionality a doporučí sa ďalší vývoj systému.

5.2 Testovanie bez užívateľa

Tento typ testovania bol použitý počas implementácie systému na vyhľadávanie trasy. Dijkstrov algoritmus bol testovaný štýlom **Test-driven development**, t.j. najprv boli napísané **unit testy** na overenie funkčnosti, až potom samotná implementácia.

To isté platí pre druhý prípad použitia unit testov – pre algoritmus spájania a rozdeľovania úsekov trasy.

Pomocou NetBeans IDE je možné unit testy jednoducho vytvoriť a spustiť. Výsledky týchto testov zobrazuje screenshot z IDE na obr. 5.1.

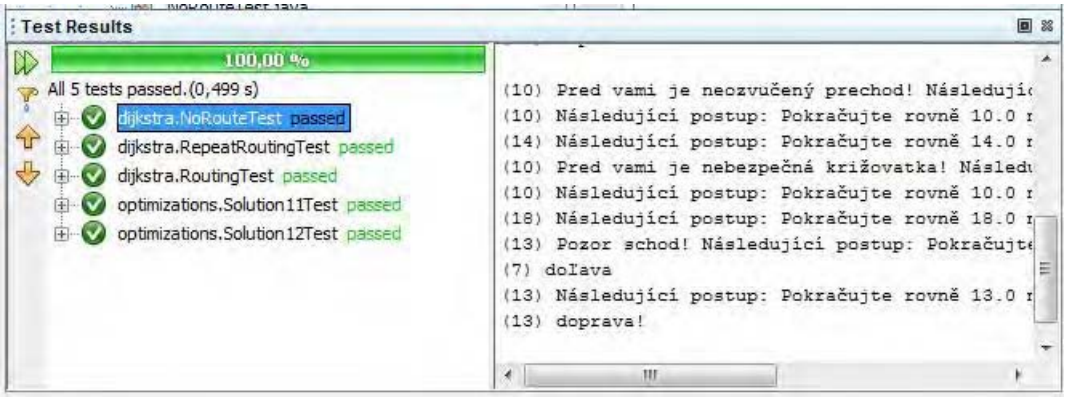
5.2.1 Unit testy pre Dijkstrov algoritmus

Testy sú vytvorené tak, aby overili základnú funkčnosť algoritmu, aj špeciálne prípady použitia:

1. **RoutingTest** – overí, či algoritmus nájde trasu. Trasa musí byť optimálna.

2. **RepeatRoutingTest** – testuje situáciu, kedy je trasa vyhľadávaná opakovane. Algoritmus sa musí korektne inicializovať znovu.

3. **NoRouteTest** – testuje špeciálny prípad, kedy cesta grafom medzi zadanými bodmi neexistuje. Algoritmus to musí detektovať a musí sa korektne ukončiť.



Obr. 5.1: Výsledky unit testov v prostredí NetBeans

5.3 Testovanie s užívateľom

Systém bol testovaný s jeho užívateľmi – operátormi navigačného centra. Nebol testovaný so zrakovo postihnutým (aj ten je užívateľom systému), pretože funkcionalita, ktorá bola do prototypu pridaná sa týka hlavne operátorov.

Priebeh testov sa riadi prípadom použitia z obr. 1.13. Podľa neho boli pripravené testy na prácu s aplikáciou JOSM a na „naklikanie“ trasy do systému. Namiesto vyskúšania vygenerovaného popisu s nevidiacim boli pripravené testy, v ktorých má operátor upraviť vygenerovaný popis tak, aby ho mohol prezentovať nevidiacemu. Využijeme tak jeho skúsenosti a porovnaním upraveného s originálnym popisom zistíme prípadné nedostatky.

5.3.1 Príprava testov

Testom predchádzali prípravné práce, hlavne príprava hardvéru. Bol nainštalovaný server, na ktorom beží testovaná aplikácia. Tak je možné simulovať podmienky, ktoré budú mať užívatelia pri reálnom nasadení systému. Ďalšími prípravnými krokmi bolo vytvorenie dotazníkov a zadaní testov, ktoré sú uvedené v prílohe G.

Forma testovania

Testovanie sa začína úvodným rozhovorom. Operátor je zoznámený s cieľom testu, testovaným systémom a stručne je popísaný priebeh testov. Je ubezpečený, že testovanie je anonymné, a že netestujeme jeho, ale aplikáciu.

Nasleduje predtestový dotazník. Pomocou neho zistím, ako je operátor schopný pracovať s PC a jeho znalosť angličtiny. Tieto informácie sú potrebné, aby bolo možné zistiť dôvod prípadných problémov s riešením úlohy. Môže byť problém v aplikácii, alebo je problémom neznalosť angličtiny u operátora.

Potom sú operátorovi prezentované zadania testov. Operátor je poučený o tom, že zadania testov obsahujú názvy ulíc a miest, ktoré musí nájsť na mape. Aby sa ušetril čas, môže sa opýtať zadávateľa na ich umiestnenie na mape. Operátor by mal úlohy splniť a rozmýšľať pri ich plnení nahlas. V prípade, že nie je schopný dokončiť úlohu, zadávateľ mu pomôže.

Po teste má operátor možnosť vyjadriť sa k testu a aplikácii a vyplní potestový dotazník.

5.3.2 Priebeh testovania

Testovanie prebiehalo v SONS priamo na pracovisku navigačného centra. Operátori použili vlastné pracovné počítače a tak boli vytvorené rovnaké podmienky ako v reálnej prevádzke systému.

5.3.3 Účastníci testu

Pri testovaní sa postupne vystriedali 4 operátori navigačného centra. Každý z nich mal podobné výsledky predtestového dotazníku. Všetci pracujú s počítačom a elektronickými mapami prakticky denne a každý sa dorozumie anglicky.

Navyše sa už každý z nich stretol s našou aplikáciou – počas testovania predošlej implementácie. Nemali teda problém zvládnuť základnú prácu s aplikáciou a označovaním trasy.

Operátori sa teda zamerali hlavne na skúmanie vygenerovaného popisu.

5.4 Výsledky testovania

Z testov vyplývajú nasledujúce zistenia. Niektoré boli pozorované pri práci operátorov, ďalšie operátori vyjadrili ako návrhy na zlepšenia. K nim ich inšpirovala práca s podobnými mapami, hlavne Google Maps a Mapy.cz. V nich používajú funkcionalitu, ktorú by radi videli aj v našom systéme.

Zistenia sú rozdelené podľa funkcionality, ktorej sa týkajú.

5.4.1 Routing

Táto funkcia bola prijatá kladne. Prejavili sa ale vážne nedostatky v nastavovaní preferencií vyhľadávania.

Nastavenie preferencií vyhľadávania Systém, akým sa zadáva považujú operátori za komplikovanejší, než trasu nájsť ručne. Dôvodom je, že musia „odmerať“ dĺžku, akú je nevidiaci ochotný prejsť navyše. Problém by sa dal vyriešiť tak, že operátori nebudú zadávať dĺžku, ale len hodnotu áno/nie, ktorá nastaví preddefinovanú dĺžku sama.

Možnosť vypnúť routing Problém vznikol, keď na mape nebol zanesený prechod, ktorý v skutočnosti existuje. Routing ho obišiel a nebolo možné zadať trasu vedúcu po ňom. Musí existovať jednoduchá možnosť routing vypnúť.

Možnosť začať trasu mimo chodník Niektorí operátori sa snažili začať trasu priamo na mieste, kde sa nevidiaci mal nachádzať, napr. v reštaurácii. Routing im ale povolil zadať len trasu začínajúcu až na chodníku. Bolo by dobré, keby routing našiel trasu k najbližšiemu chodníku.

Mazanie trasy Na vymazanie trasy po bodoch nie je vhodné tlačítko **Backspace**, pretože v internetových prehliadačoch je to presun na predošlú stránku. V prípade, že kurzor nie je nastavený na mapu, backspace nefunguje tak, ako by mal.

Záver z týchto testov je, že funkcia sa osvedčila a stačí upraviť jej ovládanie, aby bola prakticky použiteľná. Operátori ju považujú za prínos, ak sa vyrieši zadávanie preferencií.

5.4.2 Kvalita popisu

Čo sa týka kvality popisu, mali operátori rôzne názory. Je zaujímavé, že niektorí uviedli ako najväčší prínos to, že je veľmi podrobný a zároveň tvrdili, že je zbytočné, až kontraproduktívne uvádzať tak veľa informácií. Z testovania vyplývajú nasledovné zistenia.

Vodiace línie Problém sa prejavil na ulici, ktorá viedla mierne do oblúka. Systém vygeneroval popis, ktorý poslal nevidiaceho o 4 metre vpred a potom mu prikázal otočiť sa mierne vpravo. To nasledovalo mnoho krát po sebe. Lepšie by bolo, keby systém dokázal identifikovať nejakú vodiacu líniu – stenu domu, obrubník, zábradlie a nevidiacemu prikázal „držte sa steny domu“.

S týmto problémom súvisí ďalší, týka sa pokynov „otočte sa mierne vľavo“ a „otočte sa veľmi mierne vľavo“, ktoré sa objavujú v popise trasy. Takýto pokyn nemá pre nevidiaceho veľký zmysel, pretože bez ďalšieho upresnenia ho nemôže ani približne vykonať.

Mnoho informácií Napriek tomu, že informácie je možné „odmazať“, popis stále obsahuje informácie, ktoré sa nebudú dať nijak využiť. Jedná sa o popisné čísla domov (ak to nie je cieľom trasy), niektoré názvy ulíc, linku metra a počet jazdných pruhov.

Popis konca úseku Nevidiaci nedostáva žiaden popis miesta, ktoré má dosiahnuť. To sa týka konca jednotlivých úsekov, aj celej trasy. Je nevyhnutné, aby nevidiaci vedel, že dosiahol určitý bod trasy, a že má pokračovať ďalším úsekom.

Rozdelenie úsekov Najväčší problém predstavovalo nevhodné rozdelenie úsekov. Úsek je v súčasnosti automaticky vytvorený medzi dvomi bodmi chodníka tak, ako je to zadané v mape OSM. To nie je v súlade s tým, ako vytvárajú popis navigátori. Úsek by mal existovať medzi dvomi rohmi ulice, prerušiť ho môže len prechod, alebo schody alebo podobná prekážka. Zároveň musí existovať popis konca tohoto úseku.

Ak je popis bez zjavného dôvodu prerušený uprostred chodníka, ktorý pokračuje rovno, je to pre nevidiaceho problém – tradične je popis úseku prerušený len tam, kde je prekážka, prípadne kde sa musí otočiť.

Niektorí operátori si želali „obrátiť chronológiu“ popisu tak, aby na začiatku bol pokyn, potom popis konca a potom popis vecí, ktoré cestou minie. Príklad takého popisu je na obr. 5.4. Nepredstavuje to však veľký problém.

Električky Bolo by dobré, keby systém uvádzal, po ktorej ruke má nevidiaci električkovú trať. Podľa zvuku električky sa môže nevidiaci dobre orientovať.

Prechod pre chodcov Popis prechodu v súčasnosti uvádzaný ako „. . . prechod pretína ulicu . . .“ nahradiť vetou „. . . prechod vedie cez ulicu . . .“.

Jednosmerné ulice Popis jednosmerných ulíc by mal uvádzať aj smer ulice vzhľadom k pohybu nevidiaceho. Keď bude vedieť, ktorým smerom sa na ulici pohybujú autá, môže jednoducho určiť, ktorým smerom sa má vybrať.

Stena domu Hlášku, že vedľa nevidiaceho vedie stena domu to uvádza aj v prípade, že stena je dosť ďaleko od chodníka, viz obr. 5.2 a popis na obr. 5.3.

Obr. 5.2: Trasa, ktorej popis obsahuje stenu, ktorá je ďaleko od chodníka

`Otočte se velmi mírně vlevo. Vpravo od Vás je nyní bar, název Cafe bar; stěna domu. Následující postup: Pokračujte rovně 46.0 metrů.`

Obr. 5.3: Popis trasy obsahuje stenu, ktorá je ďaleko od chodníka

Poradie objektov v popise Popis uvádza objekty nachádzajúce sa na danom úseku trasy v pseudonáhodnom poradí. Správne musia byť uvedené v takom poradí, v akom okolo nich nevidiaci prejde. Príklad tohoto problému ilustruje obr. 5.5 a zodpovedajúca trasa na obr. 5.6. Čínsky bufet je v skutočnosti ešte pred reštauráciou Ridge Back.

```
...

Pokračujte rovno 60 metrov, potom sa na rohu ulíc otočte vpravo. Cestou
miniete reštauráciu.

Choďte rovno 15 metrov a prejdite cez prechod so svetelnou signalizáciou.

Za prechodom pokračujte rovno a držte sa steny vpravo. Po 20 metroch je
vchod do budovy...
```

Obr. 5.4: Správne vytvorený popis úsekov

5.4.3 Iné nedostatky

Meranie dĺžky na mape Operátori by radi mali možnosť merať vzdušnú vzdialenosť dvoch bodov na mape, tak ako to funguje na portáli Mapy.cz. Vzhľadom k tomu, že routing vyhľadá optimálne dlhú trasu a generátor popisu uvádza presné vzdialenosti nemusí byť táto funkcia vôbec potrebná.

Tvorba chodníkov Na využitie systému musia byť do systému zadané chodníky a prechody. Operátori sa zhodli na tom, že nemajú čas a chuť venovať sa tejto činnosti. Stálo by za to vytvoriť systém, ktorý by tento proces uľahčil. Bolo by možné napríklad využiť existujúce cesty pre vozidlá, vedľa nich často vedie chodník.

Mapové podklady Operátori považovali za samozrejmosť, že namiesto vygenerovanej mapy budú zobrazené satelitné snímky. Je potrebné vygenerovať len priesvitnú mapu s bodmi a chodníkmi a pod ňou zobrazovať satelitné snímky.

GUI klientskej aplikácie Operátori používajú pri práci prehliadač Opera. Klientská aplikácia musí byť otestovaná a funkčná aj v tomto prehliadači.

5.4.4 Záver testovania

V závere testovania uvádzam tabuľku 5.1, kde sú vybrané problémy zoradené od najzávažnejšieho. Tieto by mali byť vyriešené ako prvé. Zároveň usudzujem, že nedostatky routingu je možné jednoducho vyriešiť návrhom kvalitného GUI pre klientskú aplikáciu.

Problémy označené závažnosťou 5 a 4 sú kritické a spôsobujú chyby v popise trasy. Je potrebné ich v každom prípade odstrániť. Nižšie hodnotené problémy sú doporučenia pre ďalší vývoj a problémy ohodnotené 1 sú len drobné nedostatky.

Obr. 5.5: Trasa, ktorej popis zobrazuje prvky mapy v nesprávnom poradí

`Vpravo od Vás je dům č.p.290/16; restaurace Ridge back; dům č.p.288/17; restaurace, název Čínský bufet; stěna domu. Vlevo od Vás je jednosměrná hlavní třída, název Karlovo náměstí, provoz tramvají. Podloží pod Vámi je chodník. Následující postup: Jděte rovně 10.0 metrů.`

Obr. 5.6: Popis trasy zobrazuje prvky mapy v nesprávnom poradí

Závažnosť	Problém	Poznámka
5	**Rozdelenie úsekov**	
5	**Popis konca úseku**	
4	**Poradie objektov v popise**	
4	**Stena domu**	
3	**Vodiace línie**	
2	**Mapové podklady**	
2	**GUI klientskej aplikácie**	Zahŕňa problémy spojené s routingom
2	**Električky**	
1	**Mnoho informácií**	
1	**Prechod pre chodcov**	

Tabuľka 5.1: Problémy zistené testovaním zoradené podľa závažnosti

Kapitola 6

Záver

Cieľom projektu bolo vyvinúť systém, ktorý uľahčí prácu operátorom navigačného centra SONS[1]. Bol využitý prototyp existujúceho systému na generovanie popisu pre nevidiacich, ktorý bol zdokonalený. Práca sa zamerala na dva smery – implementovať algoritmus vyhľadávania trasy a zvýšiť kvalitu vygenerovaného textu. V tejto fáze bol úspešne využitý otvorený projekt OpenStreetMap ako zdroj dát.

Ďalšia fáza práce bola venovaná nasadeniu prototypu do prevádzky. Hotový hardvér a softvér bol nasadený v SONS a je dostupný operátorom navigačného centra, aj vzdialeným užívateľom na Internete. Značne sa tak zjednoduší ďalší vývoj a testovanie, pretože odpadne zdĺhavý proces inštalácie potrebného vybavenia. Pre vyskúšanie hotovej aplikácie stačí navštíviť `http://195.39.92.59:8080/OSMnavigationSerivces-war/`.

Implementovaná funkcionalita bola otestovaná s koncovými užívateľmi – operátormi navigačného centra.

6.1 Doporučenie ďalšieho vývoja

Na základe testovania a skúseností nadobudnutých pri práci na projekte je možné navrhnúť mnoho zlepšení a doporučiť, akým smerom by sa mal vývoj uberať.

6.1.1 Užívateľské rozhranie

Pre lepšie využitie aplikácie je potrebné navrhnúť a otestovať grafické užívateľské rozhranie (GUI). Nejde len o komfort používania a estetickú stránku, ale je potrebné implementovať niektoré funkcie tak, aby ich dokázal používať netechnický užívateľ. Jedná sa o nasledovné funkcie:

- **Automatické nastavenie veľkosti bounding boxu.** Bounding box, v ktorom sa má vyhľadávať nemusí byť nastavený pevne, ale môže sa postupne zväčšovať kým, sa v ňom nenájde trasa. O tomto probléme pojednáva kapitola 3.1.4.

[1] Sjednocená organizace nevidomých a slabozrakých

- **Zadávanie preferencií vyhľadávania.** V súčasnej podobe GUI operátor zadáva priamo OSM Tagy. Tento postup nie je vhodný pre užívateľa bez znalosti XML a OSM dátových typov. Lepšie riešenie je ponúknuť mu nastavenia pre konkrétne objekty ako schody, semafory a ďalšie.
- **Zobrazenie satelitných snímok** Miesto mapy si operátori želajú vidieť satelitné snímky obohatené o chodníky a body, po ktorých môžu vyznačovať trasu.

6.1.2 Vkladanie nových dát

Pre vkladanie nových zmapovaných oblastí sa používa aplikácia JOSM, ktorej výstup sa exportuje do OSM API a PostGIS databázy. Tento postup sa po technickej stránke ukázal ako dobré riešenie. Je ale potrebné tento postup zautomatizovať.

Ideálne by mal program JOSM exportovať dáta priamo do našej aplikácie. To je možné dosiahnuť pomocou pluginu do JOSM, ktorý by exportoval dáta, vygeneroval mapové podklady a nasadil tento výstup na server. Vytvoriť takýto plugin je vhodným pokračovaním práce na projekte. Celý postup na vloženie nových dát je uvedený v prílohe D.

6.1.3 Zabezpečenie serveru a vzdialená správa

Server, na ktorom bežia všetky potrebné služby je momentálne nasadený na verejnej IP adrese. Je potrebné ho skontrolovať na bezpečnostné riziká a prípadne zabezpečiť tento server. Taktiež zakázať vzdialený prístup k službám a databázam, u ktorých nie je potrebný.

V rámci vzdialenej správy je potrebné vyriešiť jednoduché nasadzovanie nových verzií prototypu na aplikačný server Glassfish.

6.1.4 Nasadenie serveru

Problém s nasadením serveru vzniká pri jeho presune na inú IP adresu. V serverovej časti aplikácie (Java), aj klientskej časti (Javascript) sú URL databáz a webových služieb napísané priamo do zdrojového kódu. Po presune je potrebné všetky URL prepísať, skompilovať zdrojový kód a nasadiť novú verziu na aplikačný server.

Najlepším riešením je, aby IP adresy boli nastavené pri spustení aplikácie automaticky. Ak to nebude možné, mali by sa dať aspoň jednoducho nastaviť na jednom mieste. Zoznam URL, ktoré je potrebné meniť je v prílohe B.

6.1.5 Využitie liniek MHD

Pri generovaní a vyhľadávaní trasy na väčšiu vzdialenosť by systém mohol využívať linky MHD.

6.1.6 Systém pre vkladanie reprezentácie OSM tagov

Aby operátori nemuseli čakať na novú verziu systému v prípade, že ho potrebujú rozšíriť o nové OSM tagy, mali by mať možnosť vkladať ich sami. Keď systém narazí na nový tag (ktorý nevie interpretovať), mal by operátorovi ponúknuť možnosť napísať jeho interpretáciu, ktorú si systém zapamätá.

6.1.7 Pomalé generovanie popisu

Vygenerovanie popisu trvá v súčasnosti príliš dlho. Odozva systému je aj pre krátke trasy rádovo v desiatkach sekúnd. Problém je spôsobený veľkým počtom dotazov do databázy a HTTP dotazov do OSM API. Vhodnou optimalizáciou by bolo možné tento počet znížiť.

6.1.8 Zlepšenie kvality generovaného popisu

Problémy, ktoré pokrýva kapitola 5.4 je potrebné vyriešiť, aby mali operátori menej práce s úpravou popisu. Jedná sa aj o kritické problémy, konkrétne napríklad problém s popisom koncového úseku trasy.

Literatúra

[1] Google Maps JavaScript API V3.
http://code.google.com/intl/sk/apis/maps/documentation/javascript//.

[2] GSON knižnica.
http://code.google.com/p/google-gson/.

[3] Knižnica LibOSM2.
http://wiki.openstreetmap.org/wiki/Traveling_salesman#LibOSM.

[4] Navigačné centrum SONS.
http://navigace.sons.cz/navigacni-centrum.html.

[5] Projekt Open Route Service.
http://www.openrouteservice.org/.

[6] Webová stránka organizácie SONS.
http://www.sons.cz/index.php.

[7] M. Casati. Javascript SOAP Client.
http://www.guru4.net/articoli/javascript-soap-client/en/.

[8] S. Coast. Osobná stránka zakladateľa projektu OpenStreetMap Stevea Coasta.
http://www.stevecoast.com/talks.html.

[9] Creative Commons. Licencia CC-BY-SA 2.0.
http://creativecommons.org/licenses/by-sa/2.0/.

[10] Eva Halásová. Skúsenosti s integráciou zrakovo postihnutých v Základnej škole pre nevidiacich a slabozrakých internátnej v Levoči.

[11] Eva Halásová, Viera Kamenická, Šarlota Múdra. Ja to zvládnem sám - Metodická príručka nácviku priestorovej orientácie, samostatného pohybu a sebaobslužných činností zrakovo postihnutých detí, 2005.

[12] L. Fedorová. Bakalárska práca Vyhľadávanie najkratších ciest nad dátami z OpenStreetMap, 2009.

[13] Kubišová, Harušťák. Taktylné prvky pre orientáciu nevidiaceho.
http://architektonickebariery.sk/architektonicke-bariery/taktylne-prvky/vytvaranie-vodiacich-linii.

[14] Mgr. V. Vallová. Charakteristika zrakového postihnutia.
http://www.pistalka.proxia.sk/Article:12.

[15] O. Procházka. Diplomová práca Systém pro tvorbu popisu cesty pro zrakově postižené, 2010.

[16] Projekt OpenStreetMap. Accessible Routing.
http://wiki.openstreetmap.org/wiki/Accessible_Routing.

[17] Projekt OpenStreetMap. Aplikačné rozhrani osm api 0.6.
http://wiki.openstreetmap.org/wiki/API_v0.6.

[18] Projekt OpenStreetMap. Aplikácia Osmarender.
http://wiki.openstreetmap.org/wiki/Osmarender.

[19] Projekt OpenStreetMap. Aplikácia Potlach.
http://wiki.openstreetmap.org/wiki/Potlatch.

[20] Projekt OpenStreetMap. Dohodnutá množina tagov.
http://wiki.openstreetmap.org/wiki/Map_Features.

[21] Projekt OpenStreetMap. Editor JOSM.
http://josm.openstreetmap.de/.

[22] Projekt OpenStreetMap. Online Routers.
http://wiki.openstreetmap.org/wiki/Routing/OnlineRouters.

[23] Projekt OpenStreetMap. Popis mapového elementu relation - vzťah.
http://wiki.openstreetmap.org/wiki/Relations.

[24] Projekt OpenStreetMap. Popis mapových dát.
http://wiki.openstreetmap.org/wiki/Data_Primitives.

[25] Projekt OpenStreetMap. Postup zmapovania sveta.
http://wiki.openstreetmap.org/wiki/Places, stav z 24. 4. 2011.

[26] Projekt OpenStreetMap. Príručka pre začínajúceho užívateľa.
http://wiki.openstreetmap.org/wiki/Beginners%27_guide.

[27] Projekt OpenStreetMap. Slippy map.
http://wiki.openstreetmap.org/wiki/Slippy_Map.

[28] Roman Martinovič. Akustické majáky.
http://architektonickebariery.sk/architektonicke-bariery/akusticke-prvky.

[29] Roman Martinovič. Dôležité poznatky o samostatnom pohybe a orientácii zrakovo postihnutých.
http://architektonickebariery.sk/pohyb-a-orientacia-zp/dalsie-dolezite-poznatky-o-samostatnom-pohybe-a-orientacii-zp.

[30] Roman Martinovič. Pohyb a orientácia nevidiacich.
http://architektonickebariery.sk/pohyb-a-orientacia-zp.

[31] Roman Martinovič. Získavanie informácií pre orientáciu nevidiacich. `http://architektonickebariery.sk/ziskavanie-informacii`.

[32] R. L. R. Thomas H. Cormen, Charles E. Leiserson and C. Stein. *Introduction to Algorithms, Second Edition*. The MIT Press, 2nd edition, 2001.

Príloha A

Zoznam použitých skratiek

OSM Open Street Maps

SONS Sjednocená organizace nevidomých a slabozrakých

API Application Programming Interface

GUI Graphical User Interface

HTML HyperText Markup Language

HTTP Hypertext Transfer Protocol

ID Identification

IDE Integrated Development Environment

IP Internet Protocol

JSON JavaScript Object Notation

SOAP Simple Object Access Protocol

URL Uniform Resource Locator

XML Extensible Markup Language

Príloha B

Zoznam pevne nastavených URL

Súbor	**URL** **popis URL**
`OSM.java`	`"http://192.168.23.128:20001/api/0.6"` Odkaz na umiestnenie OSM API.
`controller.js`	`"http://192.168.23.128/kn/tiles/"` Umiestnenie tiles, ktoré budú zobrazené na mape.

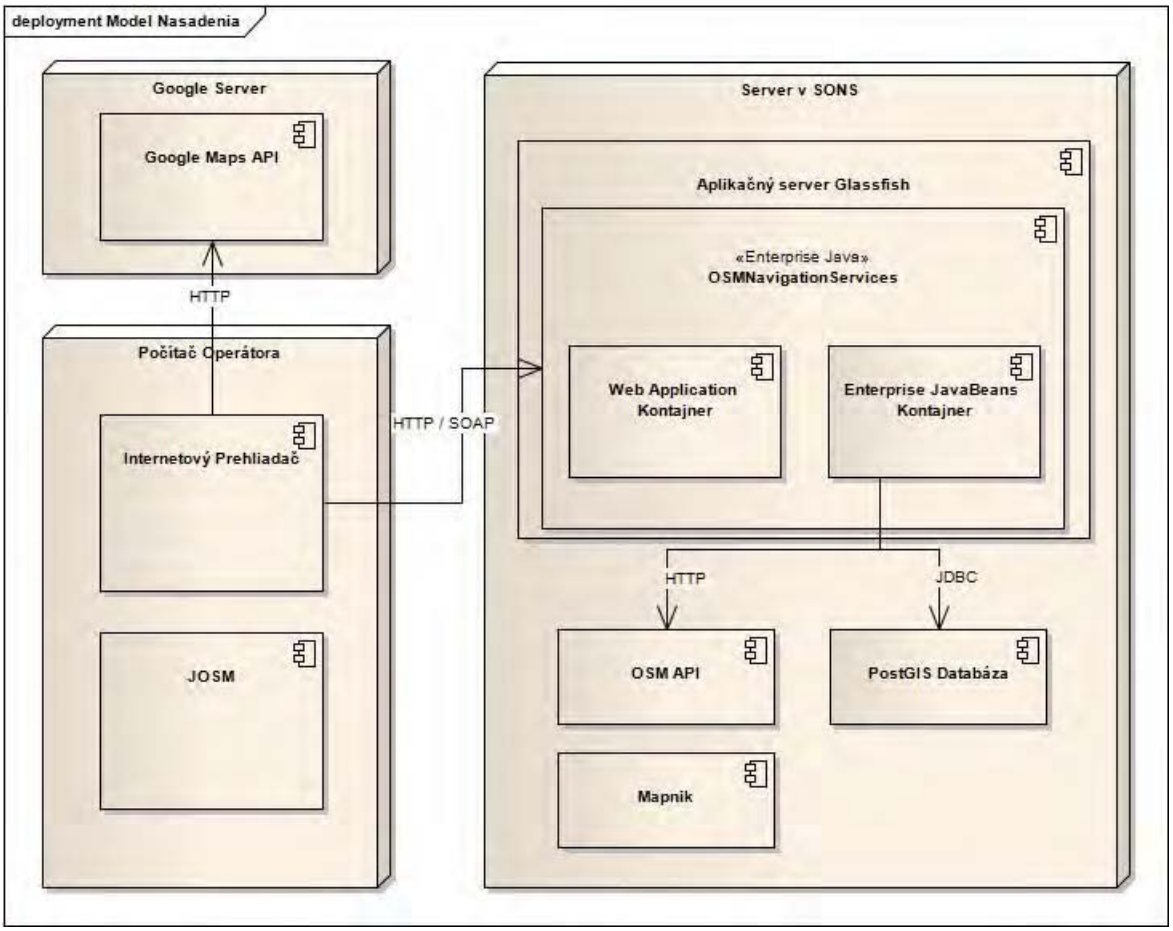
`descriptionWS.js`
`nearestPOIDescriptionWS.js`
`nearestPOIWS.js`
`routeWS.js`
	`"http://localhost:8080/DescriptionerService/`
	`/Descriptioner"`
	Adresa WSDL popisovača webových služieb na serveri Glassfish.

Tabuľka B.1: Zoznam pevne nastavených URL

Uvedené URL sú závislé na IP adresách príslušných služieb a serverov a mali by sa dať nastaviť aj mimo zdrojového kódu.

Príloha C

Model nasadenia



Obr. C.1: Model nasadenia aplikácie

Server v SONS je nasadený na verejnej IP na Internete. Vďaka tomu je možné vyskúšať aplikáciu otvorením URL adresy `http://195.39.92.59:8080/OSMnavigationSerivces-war/` v internetovom prehliadači.

Príloha D

Návod na vloženie nových mapových dát

Na vloženie nových mapových dát do systému je potrebné vykonať nasledujúce kroky:

1. Zmapovať oblasť pomocou programu JOSM.
2. Mapové dáta nahrať na OSM server a do PostGIS databázy.
3. Vygenerovať nové mapové podklady

D.1 Zmapovanie oblasti pomocou JOSM

Pred začatím práce je potrebné v JOSM **nastaviť URL adresu nášho vývojového OSM API**. Tým zabezpečíme, že JOSM nahrá dáta na správny server. Inak by použil hlavný OSM server.

Potom môžeme vytvoriť mapu. Pre uľahčenie práce je možné stiahnuť už existujúcu mapu danej oblasti z OSM servera. Túto oblasť teraz editujeme. Môžeme zobraziť satelitné snímky ako mapové podklady, pozor ale na ich licenciu.

Po skončení úprav **nahráme na server upravené dáta** a **exportujeme dáta z JOSM do súboru** s príponou osm.

D.2 Nahranie dát na server

Súbor s mapovými dátami presunieme na náš vývojový server. Teraz použijeme program osm2pgsql na nahratie dát do PostGIS databázy. **V konzole serveru sputíme skript** z obr. D.1. Heslo do PostGIS databázy je `osmosmosm`.

```
osm@osm:~/osm2pgsql$ ./osm2pgsql --slim -S./default.style -H127.0.0.1
-P5432 -Uosm -W -dgis ../data.osm
```

Obr. D.1: Skript pre nahranie OSM dát do PostGIS databázy

Ďalej je potrebné obnoviť dáta aktívnej vrstvy v PostGIS databáze. Tým zabezpečíme, že bude možné klikať na interaktívne body na mape. Je potrebné spustiť SQL skript z obr. D.2. Na to môžeme použiť buď grafický program ako PGAdmin, alebo konzolový nástroj psql, ktorý spustíme príkazom z obr. D.3.

```
--vymaže existující body z vrstvy
delete from points;
--vloží body z chodníků
select pointsToTable(way) from planet_osm_line WHERE
highway='footway';
--vloží místa zájmu
INSERT INTO points (id, osm_id, amenity, way) SELECT
nextval('seq_points'), osm_id, amenity, way FROM
planet_osm_point where amenity is not null;
--vloží adresy domů
INSERT INTO points (id, osm_id, house_num, way) SELECT
nextval('seq_points'), osm_id, "addr:housenumber", way
FROM planet_osm_point where "addr:housenumber" is not null;
INSERT INTO points (id, osm_id, shop, way) SELECT
--vloží obchody
nextval('seq_points'), osm_id, shop, way FROM
planet_osm_point where shop is not null;
--vloží zastávky tramvají
INSERT INTO points (id, osm_id, railway, way) SELECT
nextval('seq_points'), osm_id, railway, way FROM
planet_osm_point where railway is not null;
```

Obr. D.2: Skript pre vygenerovanie dát aktívnej vrstvy v databáze PostGIS

```
osm@osm:~/$ psql -f <NÁZOV SÚBORU> gis
```

Obr. D.3: Príkaz na spustenie SQL skriptu zo súboru v databáze s názvom gis

D.3 Vygenerovanie mapových podkladov

Teraz stačí vygenerovať nové tiles pomocou programu Mapnik. Skriptom z obr. D.4 spustíme generovanie tiles. Pred generovaním sa musíme ubezpečiť, že zložka `~/mapnik-tools/mapnik /scripts/knp/tiles`, do ktorej sa budú ukladať je prázdna.

Na koniec presunieme vygenerované tiles zo zložky `~/mapnik-tools/ mapnik/scripts/knp/ tiles` do `/var/www/kn/tiles`. Na to môžeme použiť program Midnight Commander.

```
osm@osm:~/mapnik-tools/mapnik/scripts/knp$ source set-mapnik-env_KN_p
osm@osm:~/mapnik-tools/mapnik/scripts/knp$ ./customize-mapnik-map_KN_p >../../osm.xml
osm@osm:~/mapnik-tools/mapnik/scripts/knp$ ./generate_KN_points
```

Obr. D.4: Skript pre vygenerovanie mapových podkladov programom Mapnik

Príloha E

Nasadenie Java aplikácie na aplikačný server

Tento postup sa využije hlavne pri nasadzovaní novej verzie aplikácie. Postup je možné vykonať v grafickej konzole servera Glassfish, alebo jednoduchší postup pomocou konzoly. Postup pre konzolu je nasledujúci:

1. Nahrať EAR (Enterprise Application) archív na disk na serveri. Na to je možné použiť napríklad obľúbený SCP klient WinSCP. Súbor OSMnavigationSerivces.ear sa štandardne nachádza v zložke dist NetBeans projektu.
2. v konzole servera spustiť skript z obr. E.1.

```
osm@osm:~/$ cd /home/glassfish/glassfish3/glassfish/bin/
osm@osm:~/$ sudo ./asadmin start-domain domain1)
osm@osm:~/$ sudo ./asadmin undeploy OSMnavigationSerivces
osm@osm:~/$ sudo ./asadmin deploy <CESTA K SUBORU OSMnavigationSerivces.ear>
```

Obr. E.1: Postup na nasadenie noveh verzie aplikácie na aplikačný server

Príloha F

Návod na inštaláciu lokálneho vývojového servera

Po nainštalovaní bude k dispozícii server, na ktorom bežia všetky potrebné aplikácie na generovanie trasy. Inštalácia je rozdelená na 2 kroky:

1. Inštalácia virtuálneho Ubuntu servera
2. Inštalácia Javovskej aplikácie OSMNavigationServices

*Inštalácia virtuálneho Ubuntu servera

Na CD prílohe je obraz disku virtuálneho servera pre VMWare. Prvým krokom, ako ho spustiť je inštalácia programu VMWare Server, ktorý je voľne dostupný pre študentov.

1. Stiahnuť a nainštalovať **VMWare Server**. (Je potrebná minimálne verzia 2.0)
2. Spustiť **VMware Server Home Page** (inštalátor vytvorí odkaz na ploche) – týmto odkazom sa spúšťa VMWare konzola
3. Prihlasovacie meno a heslo sú rovnaké ako prihlasovacie údaje do operačného systému
4. Po úspešnom prihlásení je otvorená VMWare konzola

VMWare vytvoril na disku adresár **Virtual Machines**.

5. Do tohto adresára je potrebné skopírovať zložku **Ubuntu 9.10 OSM server** z priloženého CD, ktorá obsahuje image servera
6. Vo VMWare konzole v ponuke **Commands** zvoliť **Add Virtual Machine To Directory**
7. V ponuke vybrať **Ubuntu 9.10 OSM server** a zvoliť OK
8. Ak sa VMWare dotáže, či bol image skopírovaný alebo presunutý je potrebné vybrať **presunutý (moved)**. V prípade, že je po inštalácii server na sieti nedostupný. Je potrebné vrátiť sa k tomuto kroku a zvoliť druhú možnosť.

Ďalej je potrebné v hostiteľskom operačnom systéme nakonfigurovať sieťové pripojenie, ktoré vytvoril VMWare. Postup pre MS Windows:

9. **Ovládací Panel** > **Centrum sietí** > **Spravovať sieťové pripojenia**
10. Pravým kliknúť na sieťové pripojenie s popisom **VMware Virtual Ethernet Adapter for VMnet8** a zvoliť **Vlastnosti**
11. Vybrať **IPv4 Protocol** a kliknúť na **Vlastnosti**
12. Vyplniť IP adresu **192.168.23.1**, maska **255.255.255.0**.

Teraz je možné spustiť virtuálny server z VMWare konzoly a vyskúšať sieťové pripojenie k nemu.

13. Vo VMWare konzole vybrať **Ubuntu 9.10 OSM server** a kliknúť na tlačítko **Start**
14. V internetovom prehliadači otvoriť adresu **http://192.168.23.128/kn/tiles** (ak server funguje načíta sa stránka bez problémov)

F.1 Inštalácia aplikácie OSMNavigationServices

1. Nainštalovať **NetBeans** s pribaleným serverom **GlassFish**
2. Z adresy `http://jdbc.postgresql.org/` stiahnuť **JDBC PostgreSQL driver** (vo formáte .jar) a tento súbor nahrať do lib adresára serveru glassfish (napr. `C:\Program Files\glassfish-3.0.1\glassfish\domains\domain1\lib`
3. Otvoriť projekt **OSMNavigationServices** v NetBeans, kliknúť na neho pravým a zvoliť **Deploy**
4. V NetBeans prejsť na záložku **Services**, viz. obr. F.1

Obr. F.1: Záložka Services v IDE NetBeans

5. V záložke **Services** nájsť **Servers -> GlassFish**, kliknúť pravým a zvoliť **Start server**

6. Po štarte serveru kliknúť na neho pravým a zvoliť **View Admin Console**

V internetovom prehliadači sa otvorí administrátorská konzola serveru Glassfish. V nej je potrebné nakonfigurovať pripojenie k Postgres databáze

7. V menu **Resources -> JDBC -> Connection Pools** vybrať **New** (vytvorí nový connection pool). Je potrebné vyplniť:
   - Názov: **osm_pool**
   - Resource type: **javax.Sql.DataSource**
   - Database Vendor: **Postgresql**
8. V menu kliknúť na **Resources -> JDBC -> Connection Pools -> osm_pool** a v záložkách hore vybrať záložku **Additional Properties**
9. Tabuľku vyplniť tak ako na obr. F.2

Additional Properties (13)

Add Property | Delete Properties

Name	Value
DatabaseName	gis
User	osm
LogLevel	0
Password	osm
Ssl	false
ServerName	192.168.23.128
ProtocolVersion	0
TcpKeepAlive	false
SocketTimeout	0
PortNumber	0
LoginTimeout	0
UnknownLength	2147483647
PrepareThreshold	5

Obr. F.2: Údaje pripojenia k databáze v konzole Glassfish

10. V menu **Resources -> JDBC -> JDBC Resources** vybrať **New**. Je potrebné vyplniť:
    - JNDI Name: **jdbc/osm_postgis**
    - Pool: **osm_pool**

Inštalácia je hotová a teraz je možné v NetBeans spustiť aplikáciu. Pre správne fungovanie musí byť spustený VMWare Server a musí byť dostupné pripojenie na Internet.

Príloha G

Zadania testov

G.1 Predtestový dotazník

1. Používate v práci alebo doma počítač?
   (a) Áno, každý deň
   (b) Áno, nepravidelne
   (c) Len občas
   (d) Nie
2. Používate internetové stránky zobrazujúce elektronické mapy (Google maps, Mapy.cz, Seznam mapy...)?
   (a) Áno
   (b) Nie, ale viem, že existujú
   (c) Nepoznám také stránky
3. Načo tieto mapy používate (na navigáciu v meste/na cestách, pri práci, zaujímajú vás satelitné snímky)?
4. Používate radi elektronické mapy?
   (a) Áno, sú lepšie než klasické
   (b) Nie, radšej sa im vyhýbam
5. Využívate rozšírené funkcie elektronických máp, ako napr. hľadanie ulíc, vytvorenie najkratšej trasy, . . . ?
   (a) Áno
   (b) Nie, ale viem, že to moderné mapy dokážu
   (c) Nie, myslím, že to nie je potrebné
   (d) Nepoužívam tieto mapy
6. Aká je vaša znalosť angličtiny?

(a) Dohovorím sa plynule anglicky

(b) Dohovorím sa

(c) Len krátke frázy/slová (výrazy používané v PC)

(d) Nehovorím po anglicky

G.2 Zadanie úloh

Inštrukcie Testy Vás nezdržia viac než 30 minút. Ak narazíte na problém, zadávateľ testu Vám pomôže. Netestujeme Vás, ale aplikáciu. Úlohy obsahujú názvy miest a ulíc na KN, ktoré budete musieť nájsť na mape. Nezdržiavajte sa ich hľadaním – opýtajte sa zadávateľa, kde sa nachádzajú. Pomôžete nám, ak budete nahlas rozmýšľať.

G.2.1 Testovanie JOSM

Test Case 1.1

Zadanie testu Prezrite si program JOSM. V okne vidíte mapu Karlovho Námestia, zelenými čiarami sú reprezentované chodníky, žltými značkami body. Každý bod alebo chodník má určité vlastnosti. Vyberte si ľubovoľný chodník a zistite jeho vlastnosti.

Potom skúste do mapy pridať telefónnu búdku. Postup je nasledovný: pridajte bod, kde chcete búdku postaviť a nastavte mu vlastnosť *amenity* s hodnotou *telephone*.

Počiatočný stav Otvorený program JOSM s načítanou mapou Karlovho námestia.

Koncový stav Užívateľ prečítal vlastnosti a pridal do mapy telefónnu búdku.

Ideálny priebeh Užívateľ klikne na ľubovoľný chodník a z panelu vlastnosti prečíta jeho vlastnosti. Potom vyberie nástroj *Kresliť body* na paneli s nástrojmi vľavo a klikne na mapu. Klávesou *Escape* zruší pridávanie ďalších bodov. Vybratý nástroj zmení na nástroj *výberu* a ním vyberie novovytvorený bod. Na paneli vlastnosti klikne na tlačidlo *Pridať* a v dialógovom okne vyplní hodnoty podľa zadania.

Účel testu Zoznámenie užívateľa s programom a mapou - bodmi, chodníkmi a vlastnosťami.

Test Case 1.2

Zadanie testu Presuňte sa na križovatku ulíc Myslíkova a Na Zderaze. Ulica Myslíkova má v mape zakreslený len chodník na jednej strane ulice. Všimnite si prechod na druhú stranu blízko križovatky.

Dokreslite do mapy tento prechod a zároveň chodník na druhej strane ulice. Prechod je vytvorený rovnako ako chodník, má ale nastavené zvláštne vlastnosti. Začnite kresliť prechod a postupujte ďalej po druhej strane ulice. Stačí po najbližšiu križovatku. Aby ste mohli nastaviť zvlášť vlastnosti pre prechod a chodník, musíte *rozdeliť* cestu v bode, kde sa dotýkajú. Vlastnosti prechodu sú *highway = crossing* a *crossing=uncontrolled*, vlastnosti chodníka *highway = footway*.

Počiatočný stav Pokračuje z predošlého testu.

Koncový stav Je pridaný prechod cez ulicu Myslíkova a druhá strana tejto ulice.

Ideálny priebeh Užívateľ vyberie nástroj *Kresliť body* a klikne na bod, ktorý už v mape existuje (tam, kde má začínať prechod). Postupne kreslí body po požadovanej trase. Ukončí kreslenie klávesou *Escape*. Označí vytvorený bod na druhom konci prechodu a z panela nástrojov vyberie *Rozdeliť cestu vo zvolenom bode*. Označí prechod a pridá vlastnosti postupom z predošlej úlohy. Vlastnosti chodníka sú nastavené implicitne.

Účel testu Zistiť náročnosť vkladania chodníkov a nastavenie potrebných vlastností.

G.2.2 Testovanie routingu

Test Case 2.1

Zadanie testu Predstavte si, že sa nevidiaci nachádza na rohu ulíc Myslíkova a Odborů. Snažíte sa ho priviesť pred reštauráciu Ridge Back na Karlovom Náměstí. Na mape vyznačte trasu medzi týmito dvomi bodmi. Popis vygenerovať nemusíte. Trasa sa vyznačuje klikaním na červené body na chodníkoch.

Počiatočný stav Je otvorený internetový prehliadač s testovanou aplikáciou, mapa je nastavená vo východiskovej polohe (na sever KN).

Koncový stav Bola označená trasa medzi zadanými bodmi, užívateľ využil automatické nájdenie trasy.

Ideálny priebeh Užívateľ na mape vyhľadá križovatku spomínaných ulíc a klikne na bod v blízkosti tejto križovatky. Potom vyhľadá reštauráciu Ridge Back a klikne na bod v jej blízkosti. Systém sám vyhľadá najkratšiu trasu medzi zadanými bodmi a zvýrazní ju.

Účel testu Zoznámiť užívateľa s možnosťou automaticky vyznačovať trasu medzi bodmi na mape.

Test Case 2.2

Zadanie testu Označenú trasu celú vymažte. Presuňte sa na Resslovu ulicu. Zhruba v jej strede sa nachádza Krčma u parašutistů. Vyznačte trasu od tohoto miesta k reštaurácii Stone (nachádza sa na križovatke ulíc Náplavní a Zahořanského).

Počiatočný stav Pokračuje sa od predošlej úlohy.

Koncový stav Užívateľ vyznačil trasu popísanú v zadaní.

Ideálny priebeh Užívateľ klikne na tlačidlo *Vymaž vybranou cestu.* Potom nájde Krčmu u parašutistů a klikne na bod v jej blízkosti. Pravdepodobne klikne na bod pri reštaurácii Stone, systém ale trasu nenájde a spojí body priamo. Užívateľ zmaže posledný bod, alebo celú trasu. Potom začne znovu tak, že pri vyznačovaní trasy klikne na viacero bodov, ktoré sa nachádzajú na požadovanej trase.

Účel testu Vyskúšať, ako sa užívateľ vyrovná so zlyhaním automatického vyznačovania trasy.

Test Case 2.3

Zadanie testu Od reštaurácie Stone sa vydajte na sever a zabočte doľava. Vyznačte túto trasu na mape. Keď dorazíte na križovatku na nábreží Vltavy, otočte sa a vráťte sa späť až k reštaurácii Lemon Leaf (na križovatke ulíc Myslíkova a Na Zderaze).

Počiatočný stav Pokračovanie predošlej úlohy.

Koncový stav Je označená trasa od reštaurácie Stone k reštaurácii Lemon Leaf.

Ideálny priebeh Užívateľ označí tak ako v predošlých testoch trasu ku križovatke. Potom klikne na bod pri reštaurácii Lemon Leaf (Nie je potrebné zmazať žiaden bod). Vyznačí sa požadovaná trasa.

Účel testu Zistiť, ako využije užívateľ schopnosť algoritmu "vrátiť sa"cestou, ktorou prišiel. Užívateľ by nemal vymazať žiaden bod.

Test Case 2.4

Nastavenie testu Je potrebné nastavenie testu - zväčšiť bounding box na 5000 %.

Zadanie testu Nevidiaci sa teraz nachádza na chodníku pred parkoviskom fakultnej nemocnice. Snaží sa dostať pred Novoměstskú Věž. Všimnite si, že musí prejsť cez ulicu, pričom môže použiť dva prechody. Jeden je bez svetelnej signalizácie (na kratšej trase). Druhý je ďalej, vľavo a má svetelnú signalizáciu. Nevidiaci si želá použiť prechod so signalizáciou.

Vyznačte trasu, pričom využite *parametre hľadania trasy*, ktoré nájdete pod mapou. Potom stačí označiť počiatočný a koncový bod trasy, systém nájde požadovanú trasu sám.

Počiatočný stav Pokračovanie predošlej úlohy.

Koncový stav Je označená popísaná trasa, t.j. obchádza prechod bez svetelnej signalizácie.

Ideálny priebeh Užívateľ vyplní pod mapou testové pole s parametrom *Neřízený přechod*, je potrebná hodnota aspoň 200 m. Potom klikne na počiatočný bod pri parkovisku a koncový bod pri Novoměstskej Věži.

Účel testu Zistiť, ako dokáže užívateľ využiť nastavenie preferencií vyhľadávania trasy.

G.2.3 Testovanie kvality trasy

Test Case 3.1

Zadanie testu Na mape si vyznačte ľubovoľnú trasu. Nechajte vygenerovať popis. Tento popis preskúmajte a snažte sa ho prepísať do formy, ktorú by ste prezentovali nevidiacemu.

Počiatočný stav Je otvorená mapa, nie je vyznačená žiadna trasa.

Koncový stav Užívateľ poskytne vygenerovaný popis, ktorý upravil pre reálne použitie.

Ideálny priebeh Užívateľ si vyznačí dlhšiu trasu, z ktorej vygeneruje popis a potom v textovom bloku prepíše trasu tak, aby bola použiteľná pre nevidiaceho.

Účel testu Skontrolovať optimalizácie trasy. Zistiť, ktoré optimalizácie fungujú dobre, a ktoré užívateľ prepisoval znovu. Objaviť ďalšie nedostatky generovaného popisu.

Test Case 3.2

Zadanie testu Všimnite si ponuku *Parametry generování trasy* pod mapou. Umožňuje nastaviť spájanie a rozdelovanie úsekov vo vygenerovanom popise podľa dĺžky. Skúšajte generovať popis pre rôzne trasy a sledujte, ako sú úseky rozdelené. Vyberte si jednu trasu a experimentujte s nastavením týchto parametrov kým, nebudete spokojní s rozdelením úsekov. Tento popis preskúmajte a snažte sa ho prepísať do formy, ktorú by ste prezentovali nevidiacemu.

Počiatočný stav Rovnaký ako v predošlom teste.

Koncový stav Je vygenerovaný popis k vyznačenej trase. Parametre generátora trasy sú nastavené tak, aby vyhovovali užívateľovi.

Ideálny priebeh Užívateľ vyznačí trasu, na ktorej chce parametre skúšať a vygeneruje jej popis. Tento krok môže opakovať, kým nedostane popis, ktorý mu vyhovuje. Potom sa zameria na parametre, ktoré upravuje a po úprave nechá znovu vygenerovať popis. Keď je s popisom spokojný, test končí

Účel testu Zistiť, ako dokáže užívateľ využiť možnosti nastavenia generátora trasy. Zistiť, či budú pre neho užitočné. Objaviť ďalšie nedostatky generovaného popisu.

G.3 Potestový dotazník

1. Chceli by ste pri práci používať systém tvorby popisu, ktorý ste práve vyskúšali?
   (a) Áno
   (b) Asi áno
   (c) Nie
   (d) Rozhodne nie
2. Čo myslíte, že je najväčšia výhoda tohoto systému?
3. Tvorba mapových podkladov (chodníkov) je kľúčová pre rozvoj systému. Boli by ste ochotní venovať tejto činnosti potrebný čas?
   (a) Áno
   (b) Asi áno
   (c) Nie
   (d) Rozhodne nie
4. Ako hodnotíte kvalitu vygenerovaného popisu? V čom sú podľa vás nedostatky?
5. Považujete systém hľadania trasy za užitočný?
   (a) Áno, funguje výborne
   (b) Áno, ale má určité nedostatky
   (c) Nie, iba to komplikuje

Príloha H

Obsah priloženého DVD

```
.
|-- Ubuntu 9.10 OSM server        - zložka s obrazom virtuálneho servera
|-- OSMnavigationSerivces         - NetBeans projekt s vyvíjanou aplikáciou
|-- Latex_sources                 - Zdrojové kódy textu BP
'-- boksajak_2011bach.pdf         - text BP v PDF formáte
```