VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Absolvování individuální odborné praxe

Individual Professional Practice in the Company

2017 Martin Ondo-Eštok

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Martin Ondo-Eštok**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Absolvování individuální odborné praxe
Individual Professional Practice in the Company

Jazyk vypracování: čeština

Zásady pro vypracování:

1. Student vykoná individuální praxi ve firmě: KVADOS, a.s.
2. Struktura závěrečné zprávy:
   a) Popis odborného zaměření firmy, u které student vykonal odbornou praxi a popis pracovního zařazení studenta.
   b) Seznam úkolů zadaných studentovi v průběhu odborné praxe s vyjádřením jejich časové náročnosti.
   c) Zvolený postup řešení zadaných úkolů.
   d) Teoretické a praktické znalosti a dovednosti získané v průběhu studia uplatněné studentem v průběhu odborné praxe.
   e) Znalosti či dovednosti scházející studentovi v průběhu odborné praxe.
   f) Dosažené výsledky v průběhu odborné praxe a její celkové zhodnocení.

Seznam doporučené odborné literatury:

Podle pokynů konzultanta, který vede odbornou praxi studenta.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Jan Kožusznik, Ph.D.**

Konzultant bakalářské práce: Ing. Radek Garzina

Datum zadání: 01.09.2016

Datum odevzdání: 28.04.2017

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 11. dubna 2017 .....

Souhlasím se zveřejněním této bakalářské práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v bakalářských programech VŠB-TU Ostrava.

V Ostravě 11. dubna 2017

KVADOS®
SOFTWARE SOLUTIONS
KVADOS, a.s.
Pivovarská 4/10, 702 00 Ostrava 1
IČ: 25826654 DIČ: CZ25826654

.....

Chcel by som sa touto cestou poďakovať môjmu vedúcemu bakalárskej práce Ing. Janovi Kožuszníkovi, Ph.D. za cenné rady, a čas venovaný konzultáciám, a Ing. Radkovi Garzinovi, Ph.D. za odborné vedenie mojej praxe. Ďalej by som sa chcel poďakovať mojej rodine a priateľke, za morálnu podporu počas celej doby štúdia.

Abstrakt

Táto bakalárska práca popisuje priebeh absolvovania individuálnej odbornej praxe v spoločnosti Kvados a.s.. Cieľom práce bol vývoj geoinformačného systému, pre poskytovanie máp a výpočet trás, s možnosťou rozšírenia o vlastné mapové vrstvy. V prvej časti popisujem fungovanie daného geoinformačného systému, spôsoby získania dát a úvodné nastavenia. V ďalších kapitolách sa venujem implementácií samotného systému a android klienta, ktorý bude konzumovať vytvorené služby, s využitím open sourcových nástrojov.

Kľúčové slová: výpočet trasy, mapové dlaždice, prevod adries, gis, android

Abstract

This bachelor thesis describes the process of absolving individual practical experience in the company Kvados a.s. The aim of the thesis was the development of the geoinformation system for providing maps and for calculation of routes with the possibility of extension for custom map layers. In the first chapter I describe the functioning of the geoinformation system, methods of obtaining data and initial settings. The following chapters deal with the implementation of the system itself and the Android client that will consume created services with usage of the open source tools.

Key Words: route calculation, map tiles, address conversion, gis, android

Obsah

Zoznam použitých skratiek a symbolov

OGC	–	Open Geospatial Consortium
HTTP	–	Hypertext Transfer Protocol
WCS	–	Web Coverage Service
WFS	–	Web Feature Service
WMS	–	Web Map Service
TMS	–	Tile Map Service
SOA	–	Service-oriented architecture
TCP	–	Transmission Control Protocol
WCF	–	Windows Communication Foundation
GIS	–	Geographic information system
API	–	Application programming interface
SDK	–	Software development kit
TFS	–	Team Foundation Server
SRID	–	Spatial Reference System Identifier
EPSG	–	European Petroleum Survey Group

Zoznam obrázkov

Zoznam tabuliek

Zoznam výpisov zdrojového kódu

1 Úvod

V čase voľby tém bakalárskych prác, som pôsobil v spoločnosti Kvados a.s. ako stážista. Vo firme sa mi páčilo natoľko, že som sa rozhodol v nej vykonávať aj bakalársku prácu formou praxe. Jednou z vypísaných tém firmy, bol aj projekt so zameraním na geotechnológie, na ktorý som sa po konzultácií so súčasným vedúcim vo firme prihlásil.

Cieľom tejto práce bolo vytvoriť geoinformačný systém, ktorý by generoval mapové podklady formou dlaždíc, a zároveň poskytoval funkcionalitu, pre výpočet najkratšej cesty medzi dvoma bodmi. Pri práci na tomto projekte, bolo potrebné zanalyzvať dostupné technológie, a zvoliť tie, ktoré by poskytovali dostatočnú variabilitu, a možnosť bezplatného využitia pre komerčné účely.

S geoinformačnými technológiami som sa nikdy pred tým nestretol, a preto som tento projekt bral aj ako osobnú výzvu naučiť sa niečo nové, a následne tieto získané znalosti, aj rovno aplikovať v praxy.

Počas celej svojej odbornej praxe, som pracoval na tomto projekte. Značnú časť času zabralo štúdium technológií a postupov využívaných v tejto oblasti. Po získaní základných vedomostí, som začal na implementácií samotného systému. Pri riešení konkrétných úloh na danom projekte, sa tieto znalosti postupne prehlbovali.

V prvej kapitole popisujem základnú architektúru, princípy fungovania systému, a úvodnú konfiguráciu. V ďalších kapitolách sa venujem samotnej implementácií systému. Na konci práce som uviedol záver, kde som zhodnitil celú prax, a zhrnul využité vedomosti získane počas štúdia, a vedomosti, ktoré mi počas práce chýbali, a bolo potrebné sa ich doučiť.

2 Predstavenie spoločnosti

2.1 O firme

Ostravská spoločnosť Kvados a.s. sa od roku 1992 považuje za rešpektovaného výrobcu a dodávateľa vlástných softwarových riešení, služieb a poradenstva. Ako jedna z mála spoločností na českom trhu informačných a komunikačných technológií, je certifikovaná podľa medzinárodných noriem ISO. Spoločnosť zamestnáva okolo 150 IT špecialistov, ktorý robia svojú prácu naozaj dobre, čoho dôkazom sú spokojný klienti v celej Európe. Kvados a.s. je tiež držiteľom niekoľkých ocenení ako Microsoft Industry Awards, Microsoft Technology Awards, a mnoho ďalších [1].

2.2 Moje zaradenie vo firme

Začiatkom roka 2015 som sa začal uvažovať nad brigádou, aby som popri štúdiu získal aj cennú prax. Prezeral som si Ostravské IT firmy, z ktorých ma najviac zaujala spoločnosť Kvados a.s. O pár dni neskôr, som sa zúčastnil veľtrhu pracovných príležitosti, kde som sa zhodou okolností zoznámil s Ing. Gabrielou Wojaczkovou zo spoločnosti Kvados a.s., ktorá ma pozvala na pohovor do sídla firmy.

Po úspešnom pohovore, som nastúpil na pozíciu programátor - stážista, a zároveň sa stal členom .NET týmu, ktorý pozostával zo siedmich odborníkov. Tým viedol Ing. Radek Garzina, Ph.D., ktorý sa neskôr stal aj mojím konzultantom odbornej bakalárskej praxe. Do konca leta 2016, som primárne pracoval na vývoji informačného systému Extranet, postaveného na platforme .NET, ktorý slúžil na elektronickú evidenciu úloh. Od začiatku septembra som sa naplno podieľal na zadaní bakalárskej práce.

3 Použité technológie

3.1 Geoserver

Geoserver je open source mapový server, ktorý využíva otvorených štandardov OGC, napísaný v jazyku Java [2]. Umožňuje užívateľom upravovať a zdieľať geografické dáta. Pre daný projekt som si zvolil Geoserver práve vďaka obrovskej užívateľskej komunite a skvelej flexibilite. Navyše poskytuje RESTfull rozhranie, pomocou ktorého s ním môžu klienti komunikovať na základe HTTP správ. Geoserver taktiež obsahuje niekoľko typov OGC služieb, ako WCS, WFS, WMS.

3.2 PostgreSQL

PostgreSQL je jeden z najpokročilejších open source systémemov na správu objektovo-relačných databáz. Vzhľadom k obrovskému množstvu rozšírení a open source licencii, je často používaný na výskumné účely, ale taktiež aj v komerčnej sfére [3]. PostgreSQL beží na mnoho moderných operačných systémoch, vrátane Windows, MacOS a Linuxových distribúcií. Využíva pokročilé techniky a algoritmy, ktoré zabezpečujú veľmy dobrý výkon, aj pri veľkom množstve dát. Podporuje niekoľko typov indexov vrátane GIST, ktorý sa hodí práve pre indexovanie geopriestorových dát.

3.3 PostGIS

PostGIS je rozšírením objektovo-relačného databázového systému PostgreSQL o podporu geografických a priestorových dát [4]. Pridáva funkcie pre prácu s geografickými dátmi, a taktiež špeciálne geometrické dátové typy.

3.4 PgRouting

Jedná sa o ďalšie open source rozšírenie PostgreSQL, ktoré poskytuje smerovaciu funkcionalitu nad geografickými dátami. Knižnica pgRouting obsahuje algoritmy pre výpočet trás, na základe rôznych parametrov [4]. Medzi najpoužívanejšie algoritmy tejto knižnice patrí Dijsktrov algoritmus, A-Star Algoritmus, a mnoho ďalších.

3.5 Windows Comunication Foundation

WCF je framework určený pre vytváranie aplikácií, založených na SOA architektúre. Podporuje zasielanie správ založených nielen na HTTP, ale aj TCP a iných protokolov [5]. Služba vytvorená pomocou WCF, odposlúchavá komunikáciu na definovaných koncových bodoch, na ktoré klienti zasielajú správy.

3.6 OpenStreetMap

OpenSteetMap je projekt zameraný na poskytovanie bezplatných geografických dát komukoľvek, kto ich potrebuje. Projekt vznikol pretože väčšina dostupných bezplatných máp podliehala prísnym právnym, alebo technickým obmädzeniam. V súčastnosti má tento projekt viac ako tri milióny prispievateľov po celom svete [6].

3.7 Android

Android je softwarová platforma vyvinutá pre mobilné telefóny, tablety, alebo iné zariadenia. Od roku 2005 ju vlastní gigantická spoločnosť Google, ktorá z nej vybudovala najrozšírenejšiu mobilnú platformu na trhu (85% mobilných zariadení) [7].

3.8 OsmDroid

OsmDroid je knižnica pre plaformu Android, ktorá umožnuje prácu s geodatami, a taktiež poskytuje vlastnú MapView komponentu. Veľmi často sa využíva s knižnicou OsmBonuspack, ktorá rozširuje OsmDroid o ďalšie užitočné triedy [8].

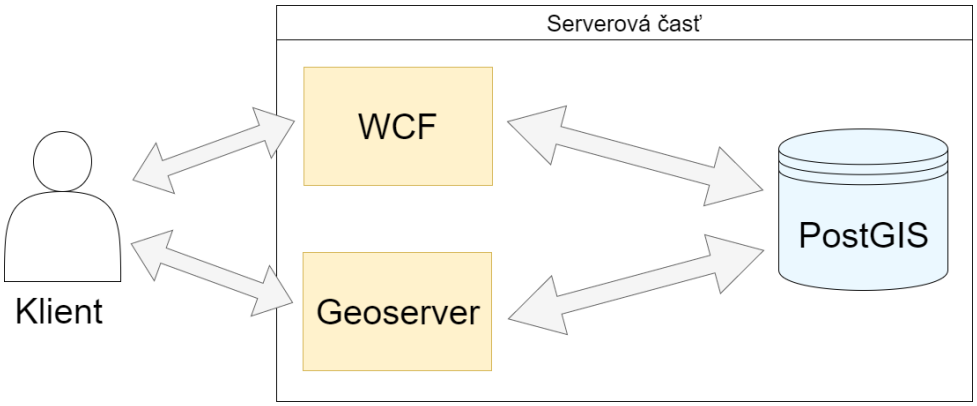
4 Geoinformačný system

4.1 Architektúra systému

Väčšina dnešných enterprise systémov je založená na komunikácií klient – server, a tak to bude aj pri tomto systéme. Klienta bude v mojej práci predstavovať jednoduchá Android aplikácia, ktorá bude zasielať HTTP správy na server.

Serverová časť bude priamo obstarávať požiadavky klienta. Budú ju tvoriť rozhrania dvoch služieb:

1. WCF služba, ktorá bude vykonávať preklad poštových adries na geografické súradnice
2. Služby mapového server, ktorý bude poskytovať podkladové mapy formou WMS služby, a trasu medzi zadanými bodmi formou WFS služby



Obr. 1: Znázornenie komponent systému

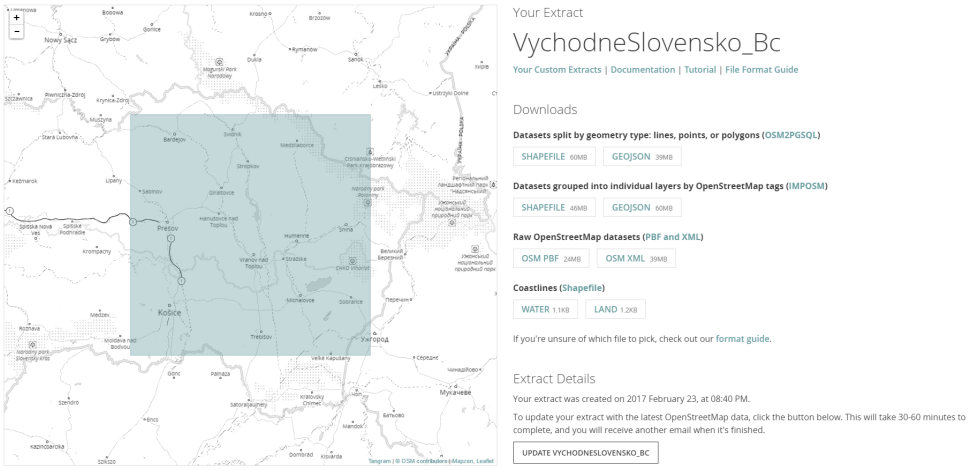
4.2 Získanie dátovej sady OpenStreetMaps

Keďže OpenStreetMap je bezplatný a voľne šíriteľný projekt, ktorý nepodlieha žiadnym obmädzeniam, existuje viacero spôsobov ako získať požadované dátové sady. Tie sú dostupne vo viacerých formátoch. Najbežnejšie dva, s ktorými som pracoval vo svojej práci sú OSM XML a PBF format.

OSM formát predstavuje nekomprimovaný XML súbor, ktorý tvoria elementy ako nodes, ways a relations [9]. PBF je skomprimovaný binárny súbor, ako alternatíva k OSM. Umožňuje niekoľkonásobné rýchlejší zápis, a je zhruba o polovicu menší [10].

Dáta celého sveta, sú k dispozícií na oficiálnej stránke OpenStreetMap a sú denne aktualizované. Dátová sada mapových podkladov celej planéty, má vyše 55 GB v OSM formáte, alebo 35 GB v PBF formáte. Nie vždy sú potrebné dáta celého sveta, preto existjú rôzne servery, alebo API, z ktorých sa dá stiahnúť ľubovoľná oblasť. Medzi najpoužívanejšie API patrí napríklad Overpass [11].

Druhou možnosťou sú servery, kde je možné ručné zvolenie požadovanej oblasti, alebo aj možnosť výberu z predpripravených dátových sád. Veľmi často využívaným príkladom, je server Geofabrik, kde sú jednotlivé dátové sady zorganizované podľa regiónov [12].



Obr. 2: Užívateľské rozhranie servera Mapzen

Pre nižšie hardvérové nároky na spracovanie dát, som sa rozhodol demonštrovať svojú prácu na menšej oblasti, konkrétne oblasti Východného Slovenska. Pre získanie dát som sa rozhodol využíť server Mapzen [13], ktorý tiež ponuká možnosť ručného zvolenia požadovanej oblasti, a navyše výslednu oblasť dokáže exportovať do rôznych formátov (PBF, geoJSON, shapefiles, a pod.).

Po zvolení požadovanej oblasti, mapzen vygeneruje zvolenú dátovú sadu, čo trvá zopár minút. Idealný formát pre tento projekt je nekompriovaný OSM súbor, keďže nástroj pre tvorbou sieťovej topológie iný formát nepodporuje. Zvolil som PBF formát, ktorý neskôr je možné konvertovať na klasický OSM, pomocou nástroja osmconverter [14].

4.3 Nasadenie Geoservera

Na začiatku je potrebné zvoliť správny servlet kontainer pre hosťovanie Geoservera. V praxy sa často využíva aplikačný server Apache Tomcat, ktorý zabezpečí jednoduchšie aktualizácie, riešenie problémov a škálovateľnosť [15]. Po inštalácií príslušnej verzie Tomcatu, stačilo nakopírovať webový archív Geoserva do adresára webapps a nakonfigurovať webový kontainer.

Veľmi doležitá je pravé optimalizácia JVM. Pri generovaní mapových podkladov sa stávalo, že Geoserver zamŕzal a nepodával dostatočný výkon. To sa mi podarilo vyriešiť zväčšením kapacity haldy JVM. Štandardnou hodnotou JVM haldy zvykne byť 64 MB, čo je pre generovanie máp nepostačujúce. Vyriešenie tohto problému zabezpečí nasledujúci príkaz, ktorý nastavý veľkosť haldy na 2 GB:

```
set JAVA_OPTS=%JAVA_OPTS% -Xmx2048m -Xmx2048m
```

Výpis 1: Príkaz pre konfiguráciu haldy JVM

V bin adresári Tomcatu je potrebné vytvoriť súbor setenv.bat, do ktorého bude tento príkaz vložený. Súbor bude automatický spúštaný pri štarte servera súborom startup.bat. Následne sa bude možné do aplikácie sa dostať pomocou url: http://localhost:8080/geoserver/web/

4.4 Príprava Databázy

Pred samotnou prácou, je potrebné si vytvoriť databázu, pre využívané geodata. Pre správu systému PostgreSQL, som využíval grafické prostredie nástroja PgAdmin, alebo občas príkazový riadok psql shellu.

Po vytvorení databázy, som z praktických dôvodov vytvoril novú schému, pretože naimportované dáta budu v dvoch rôznych topológiách. Prvú schému som využil pre dáta na renderovanie, a druhú sieťovú topológiu pre výpočet trás.

Na záver je potrebné pridať knižnice, ktoré danú databázu rozšíria o funkcie pre prácu s geopriestorovými datami, fulltextovým vyhľadávaním, a podobne. Jednotlivé knižnice popíšem neskôr pri ich využití.

```
CREATE EXTENSION postgis;
CREATE EXTENSION pgrouting;
CREATE EXTENSION hstore;
CREATE EXTENSION unaccent;
```

Výpis 2: Skripty pre import potrebných knižníc

5 Renderovanie mapových podkladov

5.1 Osm2pgsql

Osm2pgsql je konzolový nástroj, slúžiaci pre načítavanie OpenStreetMap dát do PostgreSQL databázy [16]. Schéma vytvorená týmto nástrojom je ideálna pre renderovanie máp, pretože v tabuľkách sú obsiahnuté všetky dôležité stĺpce, ktoré budem neskôr využívať v mapách pri vykresľovaní.

5.2 Príprava dát pre renderovanie

Ďalšou výhodou osm2pgsql je, že k načítaniu dát potrebuje konfiguračný súbor s príponou *.style, v ktorom sú definované názvy stĺpcov, ktoré chceme importovať [16]. Tento súbor je priložený k inštalašnému balíku nástroja. V tomto súbore je možné pridať potrebné stĺpce, repsektíve nejaké odobrať. Tieto naimportované dáta, bude potrebné neskôr spracovať aj pre geocoding. Preto je nutné pridať do súboru stĺpce, reprezentujúce názov ulice, mesta, číslo domu a krajinu:

Tabuľka 1: Dodatočné atribúty pre konfiguračný súbor osm2pgsql

OsmType	Tag	DataType	Flags
node,way	addr:country	text	linear
node,way	addr:city	text	linear
node,way	addr:street	text	linear
node,way	addr:streetnumber	text	linear
node,way	addr:housenumber	text	linear

Po pridaní požadovaných stĺpcov môžeme spustiť program nasledujúcim príkazom:

```
osm2pgsql -c -C 3000 -d slovakia -U postgres --hstore -S osm2pgsql.style
    slovakia_bc.osm.pbf
```

Výpis 3: Skript pre spustenie osm2pgsql

Program načíta požadovanú dátovú sadu, a spustí parsovanie dát, ktoré môže trvať niekoľko minút, vzávislosti na veľkosti súboru. Vyparsované dáta následne vloží do vytvorených tabuliek, a vytvorí nad nimi indexy. Každá z tabuliek obsahuje typ dát, definovaný typom ich geometrie (LINESTRING, POINT, POLYGONS a pod.). Zoznam vytvorených tabuliek je následovný:

- planet_osm_line - Geometrický typ tejto tabuľky je Linestring, takže tabuľka reprezentuje objekty, ktoré tvoria na mape čiary. Napríklad cesty, železnice, riečne toky a podobne.
- planet_osm_roads - Má rovnaký geometrický typ ako planet_osm_line, takže obsahuje dáta rovnakého typu. Narozdiel od spomínanej tabuľky, obsahuje dáta, ktoré sú vhodné pre renderovanie pri nízkom leveli priblíženia.

- planet_osm_point - Obsahuje konkrétne vrcholy z osm súboru, ktorych tag je zahrnutý v konfiguračnom súbore. Geometrický typ je Point, takže v tejto tabuľke sú body, ktoré ukazujú konkretné miesta objektov, ako napríklad vchody budov, polohy miest, historické pamiatky, konkrétne obchody v budovách a tak ďalej.
- planet_osm_polygon - Podobne ako tabuľka planet_osm_point, tiež ma dáta ktorých tagy su obsiahnuté v konfiguračnom súbore, no sú typu Polygon. Polygon je množinou bodov, ktorých spojením vzniknú súvisle plochy. Napríklad ihriská, vodné nádrže, budovy a podobne.

Po vygenerovaní tabuliek, je potrebné pridať do Geoservera dátovy zdroj. Ako je možné vydieť, Geoserver podporuje rôzne formáty dát od shapefiles, až po externú WMS službu. Tento zoznam je možné rozšíriť, pomocou komunitných pluginov. Vygenerované dáta sú uložené v PostgreSQL databáze s rozšírením PostGIS, preto je potrebné zvoliť túto možnosť. Prepojenie s databázou je jednoduchý krok, pri ktorom stačí vyplniť parametre pripojenia, a geoserver načíta zadanú schému. Osm2pgsql nepodporuje voľbu schémy, do akej ma dáta naimportovať, a všetko vkladá do schémy public.

5.3 Nasadenie SLD štýlov

Mapy aké poznáme vo výslednej podobe, sú vo väčšine tvorené z viacerých vrstiev, kde každá vrstva predstavuje iný typ dát [15]. Väčšinou sa jedná o základne typy ako points, lines a polygons. Geopriestorové datá, neobsahujú žiadnú vlastnú vizuálnu zložku. Na to, aby tieto data boli vyobrazené, potrebujeme mať pre jednotlivé vrstvy definované štýly, ktoré na ne môžeme aplikovať. V geoserveri sú štýly definované pomocou SLD jazyka, ktorý je súčasťou štandardu OGC. SLD je značkovací jazyk na báze XML. Môžeme pomocou neho definovať farby, hrúbku, veľkosť písma, a mnoho ďalšieho.

```
<FeatureTypeStyle>
 <Rule>
  <ogc:Filter>
   <ogc:PropertyIsEqualTo>
    <ogc:PropertyName>highway</ogc:PropertyName>
    <ogc:Literal>primary_link</ogc:Literal>
   </ogc:PropertyIsEqualTo>
  </ogc:Filter>
  <LineSymbolizer>
   <Stroke>
    <CssParameter name="stroke">#bbbbba</CssParameter>
   </Stroke>
  </LineSymbolizer>
```

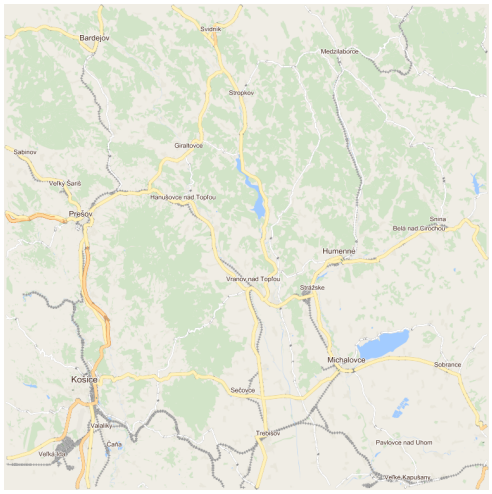
```
    </Rule>
  </FeatureTypeStyle>
```

Výpis 4: Ukážka SLD štýlu

Pre vytvorenie dobre vyzerajucích štýlov, som využil vstavaný nástroj, ktorý sa vyskytuje v novších verziách Geoservera. Umožňuje vykreslenie štýlu v reálnom čase na ľubovoľnej vrstve. Vďaka tomu je písanie štýlov vcelku pohodlné.

Ako som spomínal, dobre vyzerajúce mapy, sú tvorené z viacerých vrstiev, kde každú z vrstiev definuje vlastný štýl. SLD štýly nám umožnujú pomocou atribútov tabuliek a ich hodnôt definovať podmienky, na základe ktorých môžeme spresniť vykreslenie konkrétnych objektov vo vrstve, skrývať a zobrazovať objekty pri rôznych úrovniach priblíženia a podobne. Ukážkový SLD štýl vo výpise č. 4 definuje sivú čiaru pre atribút highway, ktorý nadobúda hodnotu primary_link.

Na internete sú dostupné rôzne štýly, a taktiež vďaka SLD štýlom, si môžu vytvoriť programátori mapy podľa vlastných predstáv. Pri tvorbe mojích štýlov, mi veľmi pomohla dokumentácia k OpenStreetMaps atribútom, kde sú jednotlivé tagy popísané [17]. Vytvoril som štýl pre každú tabuľku, vytvorenú nástrojom osm2pgsql. Jednotlivé štýly priložím k bakalárskej práci ako prílohu.



Obr. 3: Výsledná mapa po aplikovaní SLD štýlov

5.4 Vytvorenie mapových vrstiev pre WMS

WMS je OGC štandardom, pre publikovanie máp, na základ HTTP správ [18]. Po prevolaní WMS služby, server vracia mapové dlaždice v požadovanom formáte, v mojom prípade png. Predanými parametrami je potrebné definovať ohraničenie oblasti, pre ktorú chceme mapové podklady získať, názov vrstvy, gridset, formát výstupu, a ďalšie parametre, ktorými chceme definovať výstup. Štýly ktoré som vytvoril, obsahujú cesty bielej farby, preto pri volaní som ešte

nastavil parameter bgcolor, ktorý definoval farbu pozadia, a parameter transparent, ktorým som vypol priehľadnosť dlaždíc.

Na to, aby bolo možné prevolávať naimportované vrstvy, je potrebné ich najsamprv zjednotiť. Geoserver umožňuje funkcionalitu pre zjednocovanie viacerých vrstiev, a následne túto skupinu prevolávať jedným identifikátorom. Ja som si tieto vrstvy zjednotil do skupiny Slovakia v takomto poradí:

1. planet_osm_polygon
2. planet_osm_roads
3. planet_osm_line
4. planet_osm_point

Následne je potrebné nechať vypočítať ohraničenie, a zadať koordinačný systém, ktorý je v mojom prípade EPSG:900913. Po uložení bude možné danú službu prevolávať aj pre vrstvu Slovakia.

5.5 Cacheovanie máp

Geoserver obsahuje vstavaný modul GeoWebCache, slúžiaci pre cacheovanie mapových podkladov. Slúži ako proxy medzi Geoserverom a klientom, eliminuje redundantné requesty pre generovanie dlaždíc a podobne [19].

GeoWebCache umožnuje cacheovať mapové podklady za behu. Cache je pri vrstvách defaultne povolená, preto stačí pozmeniť url koncového bodu tak, že sa pridá do url tento reťazec /gwc/service/wms, a na koniec parameter tiles=true. Tento proces však volania na server príliš nezrýchli, pretože vygenerované dlaždice sa musia ukladať.

Ďalšou možnosťou ako využiť GeoWebCache, je predgenerovať mapové podklady na lokálne úložisko. Na stránke spravovania cache vrstiev, je potrebné pri požadovanej vrstve zvoliť možnosť seed. Následne sa otvorí administračná stránka GeoWebCache pre danú vrstvu, kde sa zvolí formát mapových dlaždíc, požadovaný GridSet, úrovňe levelu pre ktoré sa maju dlaždice vygenerovať, poprípade ohraničenie požadovanej oblasti. Po spustení generovania, sa mapové dlaždice štrukturovane uložia na lokálny server (LEVEL/X/Z.png), odkiaľ sa budú predávať klientovi. Tento spôsob zrýchli poskytovanie dlaždíc klientovi, keďže sa dlaždice nemusia za každým volaním generovať nanovo. Nevýhodou však je zaberanie miesta na disku, čo pri vačších oblastiach môže byť problém.

Keďže som zvolil v štýle bielu farbu ciest, bolo potrebné pridať parametre, ktoré by definovali s akou farbou pozadia sa majú dlaždice generovať. Preto v nastaveniach cacheovania požadovanej vrstvy, je potrebné pridať taktiež parametre bgcolor a transaprent, a nastaviť im požadované hodnoty.

6 Preklad adries na súradnice

6.1 Geocoding

Preklad poštových adries sa v geoinformatike nazýva aj geocoding [20]. Jedná sa o procedúru, kde sa adresy z bežne používaného textového formátu, prevádzajú na geografické súradnice, ktoré reprezentujú konkrétne miestno na zemskom povrchu. Opačnú funkciu zastáva takzvaný reverse geocoding, ktorý prevádza geografické súradnice späť na poštovú adresu [20].

Na internete existuje viacero vyhľadávacích nástrojov ktoré geocoding, či už klasický alebo reverzný poskytujú. Jednou z nich je služba Nominatim, ktorá je medzi programátormi geoinformačných systémov veľmi obľúbená, pretože taktiež ako zdroj dát využíva práve dátovú sadu OpenStreetMaps. Prístup k tejto službe je sprostredkovaný cez REST API.

Tento projekt vyžadoval čo najmenšiu závislosť na službách tretích strán. Preto bolo potrebné postaviť vlastné API, ktoré by sprostredkovávalo preklad adries zložených z mesta, ulice a čísla domu, a nahradilo do istej miery spomínaný Nominatim.

6.2 Zjednotenie adries do tabuľky

Pred vytvorením samotného API, je potrebná istá predpríprava dát. V kapitole venovej renderovaniu dát, som spomínal potrebné zmeny konfiguračného súboru pre osm2pgsql. Išlo o pridanie stĺpcov, ktoré by obsahovali dáta potrebné na geocoding. Konkrétne sa jedná o stĺpce predstavujúce adresu (kód krajiny, názov mesta, názov ulice a čísla budovy).

Importované adresy sa nachádzajú v tabuľkách planet_osm_polygon a planet_osm_point. Tieto tabuľky okrem mnou pridaných atribútov, obsahujú mnoho dalších stĺpcov, ktoré su potrebné pre renderovanie. Pre geocoding su tieto ostatné stĺpce kontraproduktívne, a zbytočne by spomaľovali vyhľadávanie. Preto som sa rozhodol extrahovať dáta do samostatnej tabuľky, ktorá bude určená pre výlučne pre geocoding. Tabuľku som si vytvoril selectom, v ktorom som zjednotil dáta z oboch tabuliek, nakoľko každá z tabuliek obsahuje adresy pre vlastné špecifické objekty.

Požadovaná tabuľka by mala obsahovať kód krajiny, názov mesta, názov ulice, číslo ulice, zemepisnú šírku (latitude) a zemepisnú dĺžku (longitude). Zemepisnú šírku a dĺžku je možné ľahko získať z hodnoty geometrického stĺpca. Rozšírenie PostGIS obsahuje mnoho funkcií pre prácu s priestorovými dátami, medzi ktoré patria aj ST_X pre zemepisnú dĺžku a ST_Y pre zemepisnú šírku [4]. Obe funkcie majú vstupný geometrický parameter typu Point. Keďže geometrický typ tabuľky planet_osm_polygon je Polygon, vyžaduje menšie úpravy. Ideálnou funkciou pre tento problém je ST_Centroid(geometry) [4]. Funkcia ma ako vstup geometrický typ, z ktorého vypočíta stredové súradnice.

Pre uloženie súradníc v štandardizovanom formáte WSG 84, som sa rozhodol geometrické typy adries transformovať pomocou funkcie ST_Transform(geometry, EPSG). Na výsledný chod

aplikacie to nemá žiaden dopad, nakoľko dnešné knižnice pre prácu s geopriestorovými datami podporuju viacero projekcií [4].

```
CREATE TABLE OSM_ADDRESSES AS
    SELECT
          "addr:country" AS country,
          "addr:city" AS city,
          "addr:street" AS street,
          "addr:streetnumber" AS streetnumber,
          "addr:housenumber" AS housenumber,
          ST_X(ST_TRANSFORM(ST_CENTROID(way), 4326)) AS lon,
          ST_Y(ST_TRANSFORM(ST_CENTROID(way), 4326)) AS lat
    FROM planet_osm_polygon
    WHERE ("addr:city" <> '') AND ("addr:streetnumber" <> '' OR "addr:
       housenumber" <> '')
    UNION
    SELECT
          "addr:country" AS country,
          "addr:city" AS city,
          "addr:street" AS street,
          "addr:streetnumber" AS streetnumber,
          "addr:housenumber" AS housenumber,
          ST_X(ST_TRANSFORM(way, 4326)) AS lon,
          ST_Y(ST_TRANSFORM(way, 4326)) AS lat
    FROM planet_osm_point
    WHERE ("addr:city" <> '') AND ("addr:streetnumber" <> '' OR "addr:
       housenumber" <> '')
```

Výpis 5: Skript pre vytvorenie tabuľky so zjednotenými adresami

6.3 Vytvorenie indexu tabuľky

Tabuľka naplnená adresami bude vo výsledku mať takmer 100 000 záznamov, nad ktorými bude fulltextové vyhľadávanie. Pri väčších oblastiach bude toto číslo mnohonásobne vyššie, preto je potrebné využiť indexy.

Databázové indexy slúžia na zrýchlenie prístupu k dátam. V prípade veľkého množstva dát, zrýchlenie predstavuje aj niekoľko desiatok percent. Pri menších tabuľkách, bude však zrýchlenie zanedbateľné. Indexy je vhodné použiť nad stĺpcami, na základe ktorých sa vykonáva vyhľadávanie, respektíve triedenie dát. V tomto prípade týmto kritériám podliehajú všetky stĺpce s výnimkou súradníc.

Pre pohodlnejšie vyhľadávanie, som sa rozhodol rozšíriť fulltextové vyhľadávanie o funkciu, ignorujúcu diakritiku. Rozšírenie unaccent ktoré som pridal na začiatku do databázy, obsahuje funkciu s rovnakým názvom, ktorá vracia reťazec bez diakriky. Indexy môžu obsahovať aj podmienky, poprípade volanie iných funkcií, preto som sa rozhodol túto funkciu v mojom indexe využiť.

Funkcie priamo volané v indexoch, musia niesť označenie IMMUTABLE [3], čo znamená, že tieto funkcie nemôžu modifikovať databázu, a pri rovnakých vstupných parametroch, stále vrátia rovnaký výstup. Funkcia unaccent má označenie STABLE [3], takže nie je možné ju v indexe priamo volať. Preto som vytvoril vlastnú immutable funkciu, ktorá vo svojom tele, predá parametre funkcii unaccent. Predané parametre (v mojom prípade hodnoty stĺpcov) sú zjednotené do reťazca s medzerami, a vrátené ako reťazec bez diakritiky.

```
CREATE OR REPLACE FUNCTION ft_unaccent(text, VARIADIC text[])
RETURNS text LANGUAGE sql IMMUTABLE AS 'SELECT unaccent(array_to_string($2, $1)
    )';
```

Výpis 6: Skript pre vytvorenie immutable unaccent funkcie

Následne je možné bez problémov túto funkciu použiť aj pri vytvorení indexu, spolu s funkciou to_tsvector, ktorá je určená na konverziu dát pre fulltextové vyhľadávanie.

```
CREATE INDEX ft_addresses_idx ON osm_addresses USING GIN (
to_tsvector('simple', ft_unaccent(' ', country, city, street, streetnumber,
    housenumber)));
```

Výpis 7: Skript pre vytvorenie indexu nad tabuľkou s adresami

6.4 Fulltextové vyhľadávanie

Fulltextové vyhľadávanie je technika vyhľadávania dokumentov, na základe výrazov prirodzeného jazyka [21]. Jednotlivé výsledky je možne zoraďovať podľa podobnosti s hľadanými výrazmi. V PostgreSQL je základnou jednotkou fulltextového vyhľadávania dokument, ktorý môže reprezentovať nejaký súvislý text (v mojom prípade sú to adresy). Dokumenty sú rozparsované na kľúčové slová asociované s ich rodičovským dokumentom, na základe ktorých sa následne porovnávajú dokumenty s hľadanými výrazmi.

Základnými funkciami pri fulltextovom vyhľadávaní v PostgreSQL sú funkcie to_tsvector a planto_tsquery. Funkcia to_tsvector [21] rozdelí porovnávaný dokument na lexéma, a pridá k nim čísla reprezentujúce ich polohu v reťazci. Tento výstup potom porovnáva s výrazom vytvoreným funkciou to_tsquery [21], ktorá rozdelí hľadaný reťazec na samostatné slová spojené logickými operátormi.

V mojom prípade sú spojene logickým operátorom AND pretože hľadám adresu ktorá obsahuje všetky zadané slová. Obe funkcie ako majú ako vstupný parameter názov slovníka pre

ignorovanie takzvaných stop words. Jedná sa o často používané slová ako napríklad predložky, spojky a podobne, ktoré budú z vyhľadávanie vyradené. Pri vyhľadávaní adries nie je slovník nutný, preto som využil simple.

Ako poslednú vec ktorú je potrebné implementovať, zoradenie výsledkov na základe podobnosti. Môže nastať situácia, že v tabuľke sa budú nachádzať mestá, kedy názov prvého, bude podmnožinou toho druhého. Napríklad Trhovište a Dolné Trhovište. Keďže funkcia vráti stále len jeden najpodobnejší výsledok, je potrebné ošetriť možné komplikácie. O to sa postará funkcia ts_rank [22], ktorá ohodnotí jednotlivé porovnania na základe podobnosti dokumentu s porovnávaným výrazom. Vstupnými parametrami funckie je tsvector dokumentu, porovnávaný výraz v query formáte a voliteľný parameter určujúci spôsob normalizácie. Spôsob normalizácie je veľmi dôležitý, pretože je potrebné rozlišovať dĺžku porovnávaného dokumentu. Ja som funkcii nastavil normalizáciu s číslom 1, čo znamená, že vypočítané ohodnotenie na záver predelí súčtom jednotky, a logaritmu dĺžky dokumentu.

```
CREATE OR REPLACE FUNCTION ft_geocode(text) RETURNS SETOF osm_addresses AS $$
    SELECT * FROM osm_addresses
    WHERE to_tsvector('simple', ft_unaccent(' ', country, city, street,
        streetnumber, housenumber))
    @@ plainto_tsquery('simple', ft_unaccent(' ', $1))
    ORDER BY ts_rank(to_tsvector('simple', ft_unaccent(' ', country, city,
        street, streetnumber, housenumber)),
    plainto_tsquery($1), 1) DESC
    LIMIT 1;
$$ LANGUAGE SQL;
```

Výpis 8: Skript pre vytvorenie ft_geocode

6.5 Implementácia WCF služby

WCF je výborná technológia na vývoj SOA služieb bežiacích v prostredí Windows [5]. S touto technológiou som mal viacero pozitívnych skúsenosti na projektoch, na ktorých som pracoval v minulosti. Úlohou bolo, aby serverová časť aplikácie pracovala na platforme .NET, čím sa WCF stala pre mňa najideálnejším riešením.

WCF som využil na vytvorenie webového rozhrania, ktoré bude sprostredkovávať HTTP požiadavky klientov na preklad adries na súradnice. Funkcionalita tejto služby bude založená na jednoduchom volaní databázovej funkcie ktorú som pre túto úlohu vytvoril. Funkcia ako parameter prijíma reťazec reprezentujúci adresu na preklad. Táto adresa je následne predaná sql funkcii. Pre prístup k PostgreSQL databáze som využil open source modul Npgsql dostupný v správcovi balíčkov nuget.

```
public Address Geocode(string address)
        {
            Address result = new Address();
            string connection = $"Server=127.0.0.1;Port=5432;User Id=postgres;
                Password=postgres; Database=slovakia";
            NpgsqlConnection conn = new NpgsqlConnection(connection);
            conn.Open();
            NpgsqlCommand command = new NpgsqlCommand("select * from geocode(
                @address)", conn);
            command.Parameters.AddWithValue("@address", address);
            NpgsqlDataReader dr = command.ExecuteReader();

            if (dr.Read())
            {
               result.Country = dr["country"].ToString();
               result.City = dr["city"].ToString();
               result.Street = dr["street"].ToString();
               result.StreetNumber = dr["streetnumber"].ToString();
               result.HouseNumber = dr["housenumber"].ToString();
               result.Lon = Decimal.Parse(dr["lon"].ToString());
               result.Lat = Decimal.Parse(dr["lat"].ToString());
            }
            conn.Close();
            return result;
        }
```

Výpis 9: Funkcia WCF služby pre geocoding

7 Výpočet trás

7.1 Osm2pgrouting

Jedná sa o ďalší konzolový nástroj, ktorý slúži na importovanie dát do PostgreSQL [23]. Narozdiel od osm2pgsql vytvára topológiu, ktorá je určená pre trasovanie.

7.2 Príprava dát pre trasovanie

Osm2pgrouting podporuje len súbory formátu OSM. Preto je potrebné, pred spustením programu konvertovať dátovú sadu vo formáte PBF [10] na OSM formát [9]. Je potreba mať na pamäti, že výsledný súbor bude mať väčšiu veľkosť. Pre konverziu dát sa hodí spomínaný osmconverter [14], ktorý je možné stiahnúť z wiki stránok OpenStreetMap, poprípate github. Taktiež je založený len na príkazoch z konzole, a je veľmi jednoduchý na obsluhu. Prevod súboru sa zabezpečí týmto príkazom:

```
osmconvert64 slovakia_bc.osm.pbf > slovakia_bc.osm
```

Výpis 10: Príkaz pre prevod PBF do OSM

```
<?xml version="1.0" encoding="UTF-8"?>
  <type name="highway" id="1">
    <class name="motorway" id="101" priority="1.0" maxspeed="130" />
    <class name="motorway_link" id="102" priority="1.0" maxspeed="130" />
    <class name="motorway_junction" id="103" priority="1.0" maxspeed="130" />
    <class name="trunk" id="104" priority="1.05" maxspeed="110" />
    <class name="trunk_link" id="105" priority="1.05" maxspeed="110" />
    <class name="primary" id="106" priority="1.15" maxspeed="90" />
    <class name="primary_link" id="107" priority="1.15" maxspeed="90" />
    <class name="secondary" id="108" priority="1.5" maxspeed="90" />
    <class name="secondary_link" id="109" priority="1.5" maxspeed="90"/>
    <class name="tertiary" id="110" priority="1.75" maxspeed="90" />
    <class name="tertiary_link" id="111" priority="1.75" maxspeed="90" />
    <class name="residential" id="112" priority="2.5" maxspeed="50" />
    <class name="living_street" id="113" priority="3" maxspeed="20" />
    <class name="service" id="114" priority="2.5" maxspeed="50" />
    <class name="unclassified" id="117" priority="3" maxspeed="90"/>
    <class name="road" id="100" priority="5" maxspeed="50" />
  </type>
```

Výpis 11: Konfiguračné xml pre import ciest

Program spustíme týmto príkazom:

```
osm2pgrouting -f slovakia_bc.osm -c mapconfig_for_cars.xml --dbname slovakia -U
    postgres -scheme pgr
```

Výpis 12: Príkaz pre spustenie osm2pgrouting

Po spustení začne program parsovať jednotlivé cesty, ktoré následne uloží do vytvorených tabuliek do schémy ktorú som sin a začiatku pripravil. Zoznam vytvorených tabuliek je následovný:

- ways_vertices_pgr
- ways
- relations_ways
- osm_nodes
- osm_relations
- osm_way_types
- osm_way_classes

Z vygenerovaj topológie sú kľúčové dve tabuľky. Tabuľka ways_vertices_pgr obsahuje vrcholy grafu, ktorá obsahuje vrcholu vytvoreného grafu. S touto tabuľkou sa bude pracovať pri hľadaní koncových bodov požadovanej trasy. Po zvolení vrcholov, bude potrebné nájsť medzi týmito vrcholmi hrany grafu, ktorých spojením vznikne trasa. Hrany grafu sa nachádzajú v tabuľke ways. Jednotlivé hrany obsahujú viacero atribútov, ako napríklad ohodnotenie, ktoré sa bude využívať pri hľadaní najkratšej cesty, id koncových vrcholov z tabuľky ways_vertices_pgr, názvy daných úsekov ak sú k dispozícií a mnoho ďalšieho. Neskôr sa pri práci s touto databázou môžu hodiť ešte tabuľky osm_way_types a osm_way_classes, ktoré uchovávajú typy ciest na základe priloženého konfiguračnéo súboru.

7.3 Hľadanie najkratšej trasy

Ako som spomínal, knižnica pgRouting obsahuje mnoho zaujímavých funkcií pre trasovanie, medzi ktoré patri aj implementácia Dijkstrovho algoritmu [24]. Tento algoritmus vyhľadáva najkratšiu cestu v grafe s nezáporným ohodnotením hrán, vďaka čomu je najpoužívanejším algoritmom v sieťových topológiach ciest. Dijkstrov algoritmus implementuje funkcia pgr_dijkstra [24], ktorú je možne použiť pre orienované, ale aj neorientované grafy. Signatúra tejto funkcie je niekoľko krát preťažená. Základná hlavička vyzerá takto: pgr_dijkstra(edges_sql, start_vid, end_vid, directed)

- edges_sql – Predstavuje select nad tabuľkou s hranami grafu, ktorého výstup bude obsahovať stĺpce pre unikátne id hrany (id), stĺpce reprezentujúce id počiatočného (source) a cieľového (target) vrcholu hrany, a stĺpec ktorý bude obsahovať ohodnotenie hrán (cost). Aliasy v zátvorkách su povinné.
- start_vid – Id nášho počiatočného vrcholu
- end_vid – Id nášho cieľového vrcholu
- directed – Logické označenie, či sa jedná o orientovaný, alebo neorientovaný graf. Štandardná hodnota je true.

Parametre start_vid a end_vid môžu byť v prípade potreby nahradené množinami počiatočných a cieľových vrcholov.

```
SELECT d.seq, w.length_m, w.name, w.the_geom FROM pgr_dijkstra('
    SELECT gid AS id,
         source,
         target,
         cost,
         reverse_cost,
         the_geom
        FROM pgr.ways',
    30623,
    30593, directed := true) d
join pgr.ways w on d.edge = w.gid
order by seq
```

Výpis 13: Využitia pgr_dijkstra nad aktuálnymi dátami

Čísla 30623 a 30593 predstavujú id počiatočného a cieľového vrcholu z tabuľky ways_vertices_pgr. Pri volaní tejto funkcie je potrebné tieto id nejakým sposobom získať na základe súradníc. Na to som využil vnorený select, ktorý pomocou distančného operátora <-> [4] ktorý slúži pre indexové vyhľadávanie najbližších susedov (KNN). Operátor sa využíva pri ORDER BY klauzule, ktorá usporiada porovnávané záznamy od najmenšej vzdialenosti po najdlhšiu. Pri porovnávaní sa berú do úvahy vzdialenosti medzi stredovými bodmi dvoch geometrií. Vytvorená WCF služba mi poskytne súradnice hľadaného bodu, ktoré následne porovnávam s vrcholmi v tabuľke ways_vertices_pgr.

```
SELECT id
FROM pgr.ways_vertices_pgr
ORDER BY the_geom <-> st_setsrid(st_point(21.8984329, 48.7571546), 4326)
LIMIT 1
```

Výpis 14: Príklad skriptu pre nájdenie najbližšieho vrcholu

Skript využíva navyše funckie ST_Point a ST_SetSRID, ktoré zo zadaných súradnic vytvoria geometriu, a nastavia jej požadovaný koordinačný systém, čo v tomto prípade je EPSG:4326.

7.4 Trasová vrstva pre WFS

Pre vytvorenie trasovej vrstvy je potrebné opäť využiť webové rozhranie Geoservera. Keďže sieťovú topológiu som si nechal vygenerovať do samostatnej schémy, musel som do Geoservera pridať nový dátovy zdroj reprezentujúci danú schému. Konfigurácia pripojenia je rovnaká ako v prípade renderovania. Jediné čo sa zmení, bude názov schémy zadanej v konfiguračnom panely.

Po pridaní nové zdroja sa opäť zobrazí možnosť s publikovaním jednotlivých tabuliek. Požadovaná trasa sa nenachádza v žiadnej zo statických tabuliek, preto je potrebne zvoliť možnosť konfigurácie vlastného SQL View, ktoré sa bude dynamicky meniť v závislostí na predaných súradniciach. Po zadaní názvu SQL View, je potrebné vytvoriť SQL skript, ktorého výstup bude obsahovať atribút geometrie, na základe ktorého, bude možné zobraziť tieto dáta. Pre tento účel som využil skript, ktorý som opísal v predchádzajúcej kapitole. Pozostáva z volania funkcia pgr_dijsktra [24] pre výpočet najkratšej trasy, a vnorených selectov pre zistenie najbližších vrcholov grafu, na základe predaných súradnic.

Geoserver obsahuje veľmi užitočné vylepšenie, ktorým je vytvorenie parametrizovaného SQL View [25]. To znamená, že SQL skript ktorý sa vytvorí v tomto view, môže obsahovať lubovoľné parametre, ktorých hodnoty sa predaju službe pri jej volaní. V tomto prípade tu budú dve dvojice súradníc bodov, medzi ktorými je potrebné nájsť cestu. Parametre som si pomenoval LON1 a LAT1 pre počiatočný bod, a LON2 a LAT2 pre cieľový bod. Pre identifikovanie parametrov v skripte, je potrebné jednotlivé parametre obaliť znakom %.

```
SELECT e.seq, e.path_seq, e.node, e.cost, e.agg_cost, w.name, w.the_geom, w.x1,
     w.y1, w.x2, w.y2 FROM pgr_dijkstra('
    SELECT gid AS id,
         source,
         target,
         cost_s AS cost,
         reverse_cost_s AS reverse_cost
        FROM pgr.ways',
    (SELECT id FROM pgr.ways_vertices_pgr
   order by the_geom <-> st_setsrid(st_point(%LON1%, %LAT1%),4326)
   LIMIT 1),
    (SELECT id FROM pgr.ways_vertices_pgr
   ORDER BY the_geom <-> st_setsrid(st_point(%LON2%, %LAT2%),4326)
   LIMIT 1),
```

```
directed := true)e
JOIN pgr.ways w on w.gid = e.edge
ORDER BY e.seq
```

Výpis 15: Skript pre parametrizované SQL View

Po zadaní skriptu, je potrebné zadať tieto parametre do tabuľky s parametrami, alebo ich pridať automaticky kliknutím na odkaz pod textovým poľom. Vstupné parametre je dôležité zabezpečiť regulárnym výrazom, a tým zabrániť SQL Injection. Predávané súradnice budú desatiné čísla, preto som tieto parametre ošetril výrazom, ktorý prepúšťal len čísla v tomto formáte. Taktiež je nutné nastaviť defaultnú hodnotu, pre inicializáciu postačí aj nula. Pre použitie týchto parametrov stačí pridať do requestu parameter s názvom viewparams, ktorý reprezentuje list dvojíc, vo formáte kľúč:hodnota, kde kľúč je názov parametra zabaleného percentami, a hodnota bude predávaná súradnica. Napr.: *viewparams=LON1:21.926;LAT1:48.746;LON2:21.922;LAT2:48.757*

Po aktualizovaní parametrov je potrebné nastaviť typ geometrie a SRID. Je možné využiť automatického nastavenia, no nie vždy funguje správne. Preto je lepšie tieto hodnoty zadať manuálne. Geometria v topológií vygenerovanej nástrojom osm2pgrouting je 4326, a typ Linestring. Po zadaní hodnôt sa môže SQL View uložiť. Následne Geoserver zobrazí konfiguráciu vrstvy. Vrstvy v Geoserveri zobrazia maximálne takú veľkú oblasť, aké ohraničenie zvolíme v nastaveniach. Keďže naša oblasť sa bude dynamicky meniť v závislosti na parametroch, je dobré počítať s najväčšou možnou oblasťou. Preto som si ohraničenie nechal vygenerovať Geoserverom z natívneho SRS, čo zabezpečí pokrytie požadovanej oblasti.

WFS služba poskytuje pre výstup viacero formátov ako napríklad JSON, GML, KML a podobne [26]. Výstupný parameter je možné definovať v requeste danej služby pomocou parametra: outputFormat=<format> Zvolil som si format JSON ktorý v dnešnej dobe jeden z najpoužívanejších formátov pri multiplaformom prenose dát. Z JSON vychádza aj formát, typický pre prenos geografických dát nazývaný GeoJSON. Existujú rôzne variácie GeoJSON-a, ktoré popisujú vektorové dáta. Každý element obsahuje objekt geometrie z tejto množiny:

- Point
- LineString
- Polygon
- MultiPoint
- MultiLineString
- MultiPolygon

Tieto objekty obsahujú jednu, alebo viac dvojíc súradníc. Tiež zvyknú obsahovať objekt popisujúci rôzne vlastnosti objektu, ako názov, id, a podobne.

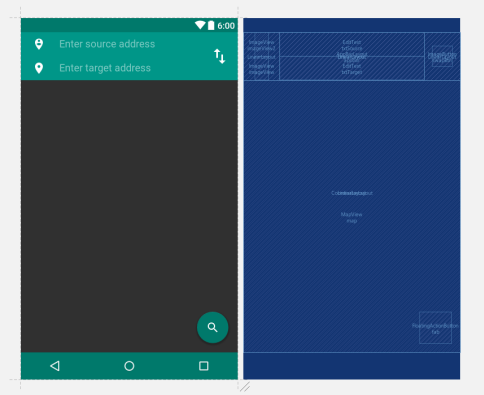
8 Android klient

8.1 Nastavenie projektu

Pred samotným vývojom bolo potrebné pridať do projektu pár knižníc pomocou buildovacie nástroja Gradle. Konkrétne sa jednalo o knižnice Osmdroid [8] a OsmBonusPack [28], ktoré obsahujú prostriedky pre prácu s geodatami na platforme android. Treťou knižnicou ktorú som využil bol HTTP klient okHTTP [27].

8.2 Návrh layoutu

Hlavnou úlohou android klienta, je demonštrácia vytvorených služieb. Preto som sa rozhodol zvoliť minimalistické užívateľské rozhranie, pozostávajúce len z pár základných komponent. V hornej časti aplikácie som umiestnil 2 textové polia, pre cieľové body trasy. Pre zlepšenie užívateľského pôžitku, som k spomínaným textboxom pridal tlačítko, pre prehodenie počiatočnej adresy s cieľovou. Pre vyobrazenie trasy slúži plávajúce tlačítko v pravom dolnom rohu. Zvyšok plochy zaberá MapView z knižnice Osmdroid, ktoré nahrádza natívnu komponentu MapView z Android SDK. Vďaka nej, je možné pre zobrazenie mapových podkladov využiť aj iné zdroje, ako napríklad WMS, alebo TMS služby od súkromných providerov.



Obr. 4: Layout aplikácie

8.3 Konzumácia WMS služby

Volanie WMS služby nastáva hneď po spustení aplikácie. MapView z knižnice OsmDroid som nastavil stredový bod a úroveň priblíženia, ktoré sa inicializujú hneď na začiatku. Na základe týchto údajov sa načíta podkladová mapa aplikácie.

Pre konzumáciu bolo potrebné implementovať vlastnú triedu, ktorá bude dediť z triedy OnlineTileSourceBase z knižnice OsmDroid. Táto trieda bude obstarávať volanie WMS služby s korektnými parametrami, a vypočítavať hraničné body požadovanej plochy.

8.4 Získanie koncových bodov trasy

Pre tento účel som využil vytvorenú WCF službu, ktorá sprostredkováva geocoding. Užívateľ po zadaní koncových adries trasy, prevolá WCF službu ktorá mu vráti objekty hľadaných adries, vrátane ich súradnic. Textové znenie získaných adries, som nastavil do hľadáčikov aplikácie. Vďaka tomu užívateľ uvidí presne znenie adries, ktoré tvoria hľadané záchytné body trasy.

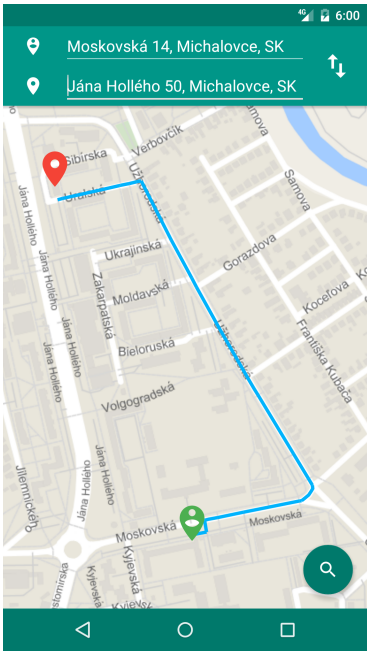
8.5 Konzumácie WFS služy

Po získaní súradnic koncových bodov, som implementoval volanie WFS služby Geoservera. Toto volanie nasleduje bezprostredne po získaní výsledkov z WCF služby. Výsledkom tejto služby bude geoJSON hľadanej trasy.

Pomocou inštancie triedy KmlDocument z knižnice OsmBonuspack [28], som vyparsoval geodata potrebné pre vykreslenie trasy. Následne som z danej inštancie vytvoril objekt FolderOverlay, ktorý som vložil do listu mapových vrstiev komponenty MapView z knižnice OsmDroid.

Ďalšími vrstvami ktoré som do tohto zoznamu pridal, boli ukazatele na hľadané miesta. Výsledna trasa vedie len po pozemných komunikáciach, nesmeruje priamo na hľadaný bod. Preto pre lepšiu orientáciu slúžia vytvorené ukazatele pomocou objektov triedy Marker.

MapView umožnuje aj priblížiť užívateľskú mapu na špeicifckú oblasť. Preto som využil funkciu triedy KmlDocument, ktorá dokáže vypočítať ohraničenie oblasti, ktorú daná vrstva pokrýva. Vďaka tomu som získal hranice oblasti, do ktorej zasahuje výsledna cesta, a predal tieto hraničné body MapView.



Obr. 5: Finálna aplikácia

9 Záver

9.1 Uplatnené vedomosti získane počas štúdia

Škola mi poskytla mnoho zaujímavých, a taktiež užitočných predmetov ktoré som dokázal využíť počas svojej odbornej bakalárskej praxe. Za kľúčove predmety ktorých vedomosti som mohol aplikovať počas praxe boli Programovacie jazyky 2 a Architekrúra .NET, ktoré mi poskytli hlbšie znalosti v programovaní aplikácií pre platformu .NET. Neskôr som na tieto vedomosti naviazal v predmete Vývoj Informačných Systémov, ktorý mi umožnil porozumieť enterprise systémom a osvojiť si aplikáciu návrhových vzorov. Síce som počas štúdia nemal predmet ktorý by sa priamo venoval databázovému systému PostgreSQL, dokázal som využíť vedomosti nadobudnuté v predmete Databázové a informačné systémy. Tieto predmety mi pomohli hlavne počas vývoja serverovej časti aplikácie. Taktiež kladne hodnotím predmety Tvorba aplikací pro mobilní zařízení 2 a Programovacie jazyky 2, ktoré mi poskytli základy Javy a tvorby aplikácií pre mobilnú platformu Android, čo som využil pri vývoji klientskej aplikácie.

9.2 Chybajúce znalosti počas vykonávania bakalárskej praxe

Za najvačší vedomostný nedostatok počastej odbornej bakalárskej praxe považujem znalosti týkajúce sa vývoja Geoinformačných technológií. Doteraz som sa taktiež v praxy nestretol s databázovým systémom PostgreSQL. Dokázal som síce skúsenosti pri práci s Oracle databázami, no tie postačovali len do istej miery, a bolo potrebné sa chýbajúce veci doučiť.

9.3 Zhrnutie

Prax v spoločnosti Kvados a.s., považujem za neoceniteľnú životnú skúsenosť, ktorá mi určite pomôže v profesnom raste. Počas odbornej praxe som sa stretol s mnoha novými technológiami, ku ktorým by som sa počas školy nedostal. Taktiež som si zlepšil analytické a programátorské schopnosti, a osvojil si prácu s verzovacím systémom TFS.

Odbornú prax hodnotím vcelku pozitívne, a som rád, že som si zvolil práve takúto formu bakalárskej práce.

Literatúra

[1] Informace o společnosti. In *Kvados, a.s.*. [online]. [cit. 2016-11-10]. Dostupné z: https://www.kvados.cz/o-spolecnosti/

[2] About. In *GeoServer*. [online]. [cit. 2016-11-10]. Dostupné z: http://geoserver.org/about/

[3] OBE, Regina - HSU, Leo. *PostgreSQL: Up and Running*. [cit. 2016-11-15]. United States of America: O'Reilly Media, Inc., 2012. ISBN 978-1-449-32633-3

[4] OBE, Regina - HSU, Leo. *PostGIS in action*. [cit. 2016-11-15]. Manning Publications Co., 2012. ISBN 9781617291395

[5] VOGEL, Peter. 2011. WCF and Service-Oriented Architectures. In *Visual Studio Magazine*. [online]. [cit. 2016-11-10]. Dostupné z: https://visualstudiomagazine.com/Articles/2011/06/01/pcnet_WCF-and-SOA.aspx

[6] About OpenStreetMap. In *OpenStreetMap Wiki*. [online]. [cit. 2016-11-16]. Dostupné z: http://wiki.openstreetmap.org/wiki/About_OpenStreetMap

[7] Android, the world's most popular mobile platform. In *Android Developer*. [online]. [cit. 2016-11-16], Dostupné z: https://developer.android.com/about/android.html

[8] OsmDroid. In *GitHub, Inc.*. [online]. [cit. 2016-11-16]. Dostupné z: https://github.com/osmdroid/osmdroid

[9] OSM XML. In *OpenStreetMap Wiki*. [online]. [cit. 2016-11-18]. Dostupné z: http://wiki.openstreetmap.org/wiki/OSM_XML

[10] PBF Format. In *OpenStreetMap Wiki*. [online]. [cit. 2016-11-18]. Dostupné z: http://wiki.openstreetmap.org/wiki/PBF_Format

[11] Overpass API. In *OpenStreetMap Wiki*. [online]. [cit. 2016-11-18]. Dostupné z: http://wiki.openstreetmap.org/wiki/Overpass_API

[12] About Geofabrik. In *Geofabrik GmbH*. [online]. [cit. 2016-11-18]. Dostupné z: https://www.geofabrik.de/geofabrik/geofabrik.html

[13] Metro Extracts. In *Mapzen*. [online]. [cit. 2016-11-25]. Dostupné z: https://mapzen.com/documentation/metro-extracts/

[14] Osmconvert. In *OpenStreetMap Wiki*. [online]. [cit. 2016-11-25]. Dostupné z: http://wiki.openstreetmap.org/wiki/Osmconvert

[15] HENDERSON, Colin. *Mastering GeoServer*. [cit. 2016-11-27]. Birmingham: Packt Publishing Ltd, 2014. ISBN 978-1-78328-769-7

[16] Osm2pgsql. In *OpenStreetMap Wiki*. [online]. [cit. 2016-11-29]. Dostupné z: `http://wiki.openstreetmap.org/wiki/Osm2pgsql`

[17] TagInfo. In *OpenStreetMap*. [online]. [cit. 2017-01-29]. Dostupné z: `https://taginfo.openstreetmap.org/`

[18] Web Map Service. In *Open Geospatial Consortium*. [online]. [cit. 2017-01-29]. Dostupné z: `http://www.opengeospatial.org/standards/wms`

[19] IACOVELLA, Stefano. *Geoserver Cookbook*. [cit. 2017-02-05]. Birmingham: Packt Publishing Ltd, 2014. ISBN 978-1-78328-961-5

[20] Google Maps Geocoding API. In *Google Developer*. [online]. [cit. 2017-02-05], Dostupné z: `https://developers.google.com/maps/documentation/geocoding/intro`

[21] Full Text Search Introduction. In *PostgreSQL: Documentation*. [online]. [cit. 2017-02-10], Dostupné z: `https://www.postgresql.org/docs/9.6/static/textsearch-intro.html`

[22] Controlling Text Search. In *PostgreSQL: Documentation*. [online]. [cit. 2017-02-10], Dostupné z: `https://www.postgresql.org/docs/9.6/static/textsearch-controls.html`

[23] osm2pgrouting - Import OSM data into pgRouting Database. In *PgRouting: PgRouting Tools*. [online]. [cit. 2017-02-10]. Dostupné z: `http://pgrouting.org/docs/tools/osm2pgrouting.html`

[24] pgr_dijkstra. In: *PgRouting Manual*. [online]. [cit. 2017-02-10]. Dostupné z: `http://docs.pgrouting.org/latest/en/pgr_dijkstra.html#pgr-dijkstra`

[25] SQL Views. In *Geoserver 2.10.x: User Manual*. [online]. [cit. 2017-02-11]. Dostupné z: `http://docs.geoserver.org/stable/en/user/data/database/sqlview.html`

[26] WFS Reference. In *Geoserver 2.10.x: User Manual*. [online]. [cit. 2017-02-11]. Dostupné z: `http://docs.geoserver.org/latest/en/user/services/wfs/reference.html`

[27] OkHTTP. In *GitHub, Inc.*. [online]. [cit. 2017-02-18]. Dostupné z: `http://square.github.io/okhttp/`

[28] OsmBonuspack. In *GitHub, Inc.*. [online]. [cit. 2017-02-18]. Dostupné z: `https://github.com/MKergall/osmbonuspack`

A Štruktúra príloh na CD