VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

KONTROLNÍ APLIKACE PRO TRASY KČT V OSM

MANAGEMENT APPLICATION FOR TOURISTIC ROUTES IN OSM

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — PATRIK LEV
AUTHOR

VEDOUCÍ PRÁCE — Ing. TOMÁŠ KAŠPÁREK
SUPERVISOR

BRNO 2017

Zadání bakalářské práce/18334/2016/xlevpa00

Vysoké učení technické v Brně - Fakulta informačních technologií

Centrum výpočetní techniky Akademický rok 2016/2017

Zadání bakalářské práce

Řešitel: **Lev Patrik**
Obor: Informační technologie
Téma: **Kontrolní aplikace pro trasy KČT v OSM**
Management Application for Touristic Routes in OSM
Kategorie: Informační systémy

Pokyny:

1. Nastudujte systém mapování turistických tras v projektu OSM.
2. Zjistěte současné možnosti ruční a (polo)automatické kontroly tras KČT v datech OSM.
3. Navrhněte postupy pro automatizovanou kontrolu stavu tras v realitě a v datech OSM.
4. Realizujte (rozšiřte stávající) aplikaci umožňující kontrolu tras KČT, uvažujte různé role uživatelů (turista poskytující jednoduché ale aktuální informace z terénu, pokročilý uživatel mapující v OSM, programátor využívající data z OSM).
5. Proveďte testování a zhodnoťte výsledky.

Literatura:

- OpenStreeMap (http://www.openstreetmap.org)
- Cyklotrasy v ČR v OSM (http://wiki.openstreetmap.org/wiki/Cyklotrasy_v_ČR)
- Turistické mapováni v OSM (http://wiki.openstreetmap.org/wiki/WikiProject_Czech_Republic/OTM_značkový_klíč)
- Learn OSM (http://learnosm.org)
- dle pokynů vedoucího

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 až 3.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Kašpárek Tomáš, Ing.**, CVT FIT VUT
Datum zadání: 1. listopadu 2016
Datum odevzdání: 17. května 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav informačních systémů
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Dušan Kolář
vedoucí ústavu

Abstrakt

Cílem této bakalářské práce je návrh a realizace aplikace pro automatickou kontrolu turistických tras spravovaných Klubem českých turistů v datech projektu OpenStreetMap. V této technické zprávě je popsán projekt OpenStreetMap a jeho způsob komunitního mapování, ukládání a distribuce OSM dat. Pro mapování turistických tras na území České Republiky slouží značkový klíč OpenTrackMap. V následující části jsou popsány možnosti manuální a automatické kontroly OSM dat. Součástí technické zprávy je návrh aplikace pro kontrolu OSM dat. V závěru téhle práce je popsána implementace aplikace a manuál k aplikaci.

Abstract

The goal of this bachelor thesis is a design and realization of an application, which would automatically check the hiking trails managed by Czech Tourist Club (Czech: Klub českých turistů, KČT) in the OpenStreetMap (hereinafter OSM) project's data. In this technical report is described the OSM project and its way of community mapping, data storage and the OSM data distribution. To map the hiking trails in the Czechia we use so called OpenTrackMap. In the following part are described the possibilities of the manual and automatic OSM data check. The part of the technical report is a design of the OSM data checking application. In the end of this work is described the implementation of the application and there is also attached the manual for the application.

Klíčová slova

OpenStreetMap, KČT, kontrola turistických tras, PostgreSQL databáze, PostGIS, PHP, MVC architektura,

Keywords

OpenStreetMap, KČT, tourist routes control, PostgreSQL database, PostGIS, PHP, MVC architecture

Citace

LEV, Patrik. *Kontrolní aplikace pro trasy KČT v OSM.* Brno, 2017. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Kašpářek Tomáš.

Kontrolní aplikace pro trasy KČT v OSM

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Tomáše Kašpárka. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

. . . . . .
Patrik Lev
9. května 2017

Poděkování

Tímto bych rád poděkoval vedoucímu této bakalářské práce, panu Ing. Tomáši Kašpárkovi, za odbornou pomoc, podněty a čas věnovaný při řešení této bakalářské práce.

Obsah

Kapitola 1

Úvod

Mapové podklady již od počátku lidstva hrají svou podstatnou roli při osídlování či dobývání nových území. Mapové podklady byli vytvářeny za pomocí vzpomínek vracejících se vojáků z bojů nebo za tímto účelem vyslání objevitelé. Získané informace byly dále šířeny pro příští generace za pomocí rytin, maleb, ústního vyprávění a v pozdějších dobách i záznamem na papír. Kvalitu a přesnost map tedy spočívala pouze na přesnosti vzpomínky nebo přesnosti zakreslení daného geografického prvku. Geografické prvky lze zakreslit různými způsoby, mezi něž patří obrázky, zástupné symboly nebo text. Pro sjednocení vyjádření geografických prvků je potřebné stanovit pravidla a dbát na jejich dodržování. S rostoucím množstvím lidí zaznamenávajících geografické prvky se snižuje přesnost mapy, jež je způsobená odchýlením se od stanovených pravidel. Pro zachování věrohodnosti mapy je nezbytné kontrolovat stanovená pravidla.

Způsob mapování v projektu OpenStreetMap (OSM) lze přirovnat výše popsanému principu. O tvorbu mapy se starají samotní uživatelé, kteří za pomocí moderních technologií zaznamenávají geografické prvky do digitální podoby. S narůstajícím počtem uživatelů tvořících mapu je potřebné kontrolovat dodržování stanovených pravidel. Dodržování pravidel lze kontrolovat ručně za pomocí uživatelů, nebo automatizovaně za pomocí programového vybavení. V této bakalářské práci se zabývám možnostmi automatické kontroly OSM dat.

Obecné informace o projektu OpenStreetMap jsou popsány v úvodní kapitole. V úvodní kapitole této práce jsou dále uvedeny jednotlivé prvky OSM dat, možností ukládání OSM dat a způsoby distribuce OSM dat. V následující části této kapitoly jsou uvedeny projekty, které využívají OSM data. Konkrétním příkladem využití projektu OSM může být mapování turistických tras na území české republiky. Popis jednotlivých prvků sloužících pro popis turistických tras je uveden v závěru této kapitoly. V kapitole č. 3 jsou podrobněji popsány příčiny nutnosti kontrolovat OSM dat a možnosti jak tato data kontrolovat. Součástí této kapitoly je výčet projektů zaměřujících se na poloautomatickou kontrolu OSM dat. Kapitola č. 4 je zaměřena na stanovování cílů této práce, definici jednotlivých etap vývoje a jejich popis. V následující kapitole je popsán způsobe realizace aplikace pro automatickou kontrolu turistických tras. V této kapitole jsou popsány problémy a možnosti řešení problémů, které se objevily při vývoji aplikace. Závěrem této práce je obsažen manuál k výsledné aplikaci.

Kapitola 2

OpenStreetMap

V nadcházející kapitole jsou uvedeny obecné informace o projektu OpenStreetMap, kterému je větší část téhle technické zprávy věnována. Pro pochopení proč je třeba provádět testování OpenStreetMap dat, je potřeba si vysvětlit jak jsou data vytvářena a jakým způsobem zaznamenávána.

2.1 0 OpenStreetMap

Cílem projektu OpenStreetMap (OSM)[1] je tvorba svobodných geografických dat. Geografická data jsou následně využita k vizualizaci do podoby topografických map. Data jsou vytvářena za pomocí samotných uživatelů. Zdrojem dat může být záznam z GPS přijímačů nebo obkreslováním leteckých map uživatelem. OSM data jsou šířena pod svobodnou licencí Open Database License[2].

2.2 Základní primitiva

- Uzel (`node`)– Bod lokalizovaný v souřadném systému[15]. Uzly se využívají k popisu bodového prvku, jako je památník nebo rozcestník.
- Cesta (`way`) – Cesta je tvořena z posloupnosti uzlů[7]. Popisuje liniové prvky. Mezi liniové prvky lze zařadit řeky, cesty, ulice. Pokud první a poslední bod je umístěn na totožném místě, jedná se tzv. o polygon. Polygony se značí lesy, louky, parky nebo vodní plochy.
- Relace (`relation`) – Využívá se pro seskupení uzlů a cest do jednoho celku[6]. Seskupení může mít mezi sebou vztah logický nebo geografický. Jednotlivé členy relace tvoří uspořádaný seznam.
  - `route` – Seskupuje primitiva typu `way` a opatřuje je tagem s číslem silnice, cyklostezky nebo číslem linky veřejné hromadné dopravy.
  - `street` – Zaobaluje všechny prvky tvořící ulici. Mezi prvky může patřit např. cesty, domy, adresy domů.
  - `multipolygon` – Slouží pro zaznamenání složitějších ploch, které v sobě obsahují díry popsané polygonem.

[1] Stránky projektu OpenStreetMap: http://www.openstreetmap.org/
[2] Licence ODBL: https://opendatacommons.org/licenses/odbl/summary/

- Značka (`tag`) – Značku lze přiřadit ke všem výše popsaným prvkům[14]. Značka je tvořena ze dvou prvků ve tvaru `<klíč>=<hodnota>`. Např. značka `maxspeed=50` vyjadřuje omezení rychlosti u relace typu `way`. Hodnota značky může být tvořen z čehokoli, avšak jsou uživateli zavedené konvence pro daný účel. Zde nastává úskalí v podobě neznalosti patřičné konvence nebo nevhodnosti použití zvolené značky.

2.3 Formát dat

Projekt OpenStreetMap má početnou skupinu editorů, kteří vytvářejí mapu. Aktuálně je do projektu zapojeno více jak 3,5 miliónu registrovaných uživatelů[13]. S množstvím uživatelů roste i množství ukládaných a přenesených dat. Pro distribuci svobodných dat je zapotřebí stanovit mechanismus, jak data efektivně šířit. V následující části jsou uvedeny 3 formáty pro distribuci OSM dat.

2.3.1 OSM XML

OSM XML formát je založen na průmyslovým standartu XML[3], který je lidsky čitelný. Formát je nositelem nejen samotných dat, ale popisuje i strukturu souboru. Je proto vhodný pro přenos dat mezi různými systémy.

Výhodou tohoto formátu je jeho čitelnost. Pro editaci není potřeba soubor nijak přeformátovat čí jinak upravovat. XML formát je zavedený průmyslový standart. Existuje mnoho prostředků pro práci s formátem XML nezávisle na projektu OSM. Mezi nevýhody formáty bych uvedl jeho paměťovou náročnost. Z paměťové náročnosti vyplývá i časová náročnost při načítání a ukládání souboru. Pro zmenšení paměťové náročnosti lze textový formát dále komprimovat za pomocí nástrojů gzip. V následující ukázce stojí za povšimnutí v prvním řádku XML definice. V následujícím řádku je element s definicí OSM dat. Element obsahuje 3 potomky. Potomci reprezentují datová primitiva popsaná výše.

Listing 2.1: Ukázka OSM XML souboru.

```
<?xml version="1.0" encoding="UTF-8"?>
<osm version="0.6" generator="CGImap 0.0.2">
    <node id="261728686" lat="54.0906309" lon="12.2441924" />
    </node>

    <way id="26659127">
        <nd ref="292403538"/>
        <nd ref="298884289"/>
        <tag k="highway" v="unclassified"/>
    </way>

    <relation id="56688" >
        <member type="node" ref="294942404" role=""/>
        <tag k="name" v="Kustenbus Linie 123"/>
    </relation>
</osm>
```

[3] https://www.w3.org/XML/

2.3.2 PBF Format

PBF Formát je binární formát využívající kompresi pro zmenšení velikosti výsledného souboru. Pro editaci je zapotřebí specializované prostředky pro načtení vstupního souboru. Referenčním projektem, který implementuje formát pbf, je projekt Osmosis[4]. Výhodou popisovaného formátu je rychlé zpracování, několika násobně menší paměťová náročnost. Pro práci s OSM daty nepotřebujeme vždy použít všechny data pro celou planetu. Formát umožňuje výřez určité geografické oblasti.

2.3.3 OsmChange

5.3.4 Pro aktualizaci dat je nevhodné vždy stahovat znovu celý balík OSM dat. Vhodnější je stáhnout pouze změny, které byli provedeny od posledního importu dat a provedené změny přidat k balíku dat. K tomuhle účelu byl navrhnut OsmChange[5] formát, který je využíván programy osmosis a osmconvert. OsmChange je svou strukturou podobný formátu OSM XML. OsmChange popisuje změny mezi dvěma verzemi OSM dat. V ukázce níže stojí za povšimnutí elementu `modify`, který značí, že obsažené prvky byli změněny. Podobně jako element `modify` je možné použít elementy `delte` a `create`.

Listing 2.2: Ukázka OsmChange souboru, který modifikuje uzel.

```
<osmChange version="0.6" generator="acme osm editor">
    <modify>
        <node id="1234" changeset="42" version="2"
              lat="12.1234567" lon="-8.7654321">
            <tag k="amenity" v="school"/>
        </node>
    </modify>
</osmChange>
```

2.4 Projekty

Na datech z projektu OpenStreetMap je postaveno mnoho projektů. Jednotlivé projekty se odlišují svým zaměřením, zobrazením nebo použitými prostředky.

2.4.1 OpenStreetMap.org

Webová aplikace projektu OpenStreetMap. Za pomocí ovládacích prvků lze vybrat mezi 4 typy mapových vrstev. Mezi nabízené vrstvy patří standartní, cyklistická, dopravní a hamanitární mapa. Pro snadnou editaci OSM dat je součástí rozhraní i možnost zobrazení historie úprav a editace.

2.4.2 MTBMap.cz

MTBMap.cz[5] projekt poskytuje volně dostupnou mapu Evropy určenou pro jízdu na horském kole a pro pěší turistiku. Otevřený projekt tvořený českou komunitou. Umožňuje vyhledávání s definicí povrchu a obtížnosti. Na stránkách projektu lze stáhnout týdenní exporty vygenerované mapy.

[4] http://wiki.openstreetmap.org/wiki/Osmosis
[5] Stránky projektu MTBMap.cz: http://mtbmap.cz/

2.4.3 GNOME Maps

GNOME Maps[6] je desktop aplikace určená pro operační systém Linux s grafickým prostředím Gnome. Umožňuje zobrazit pouze online mapy. Přidanou hodnotou k zobrazení map je vyhledávání cílů a navigace. Aplikace je v raném stadiu vývoje. S postupem vývoje budou přidány další funkční prvky.

2.4.4 OsmAnd

OsmAnd[7] je mobilní aplikace určená pro operační systémy Android a Apple iOS. Aplikace neobsahuje pouze mapy, ale i další funkcionalitu využívající data OpenStreetMap. Aplikace umožňuje stáhnout mapu offline a využívat jí pro další použití. Mezi příklady použití offline mapy bych uvedl možnost offline navigace nebo zobrazení veřejné hromadné dopravy.

2.5 Mapování turistických tras v OpenStreetMap

Turistické trasy jsou popsány relací obsahující značku `route` s hodnotou `foot` nebo `hiking`. Hodnoty `foot` a `hiking` lze považovat jako ekvivalentní. V české republice jsou turistické trasy značeny a spravovány Klubem český turistů[8]. Turistické trasy v České republice jsou zaznamenány jako relace skládající se z částí cest, uzlů a rozšířené o specifické značky. Značkový klíč OpenTrackMap definuje pravidla pro mapování turistických a cykloturistických tras v české republice[11]. Popis jednotlivých specifických značek je uveden níže.

2.5.1 tag `type`

Značka `type` slouží pro popis typu relace. Pro turistické trasy nabývá hodnoty `route`. Tato hodnota značky se používá k popisu jakékoliv cesty.

2.5.2 tag `route`

Slouží pro popis fyzicky neexistující dopravní trasy[6]. Pro turistické trasy se využívají tyto hodnoty:

- `foot` – turistická stezka
- `ski` – lyžařská stezka
- `bicycle` – cykloturistická stezka
- `wheelchair` – stezka pro vozíčkáře
- `horse` – stezka pro jezdce na koni

2.5.3 tag `network`

Značku `network` lze použít k různému účelu. Účel použití závisí na lokalitě výskytu popisovaného prvku. Značka se může použít pro popis systému silnic, tras veřejné dopravy, cykloturistických tras nebo turistických tras. Pro popis turistických tras v České republice se používají následující hodnoty:

[6] Stránky projektu GNOME Maps: https://wiki.gnome.org/Apps/Maps
[7] Stránky projektu OsmAnd: http://osmand.net/
[8] Klub českých turistů: http://www.kct.cz/

- `iwn` – využívá se pro mezinárodní dálkové trasy[11].
- `nwn` – využívá se pro dálkové trasy[11].
- `rwn` – využívá se pro regionální trasy[11].
- `lwn` – využívá se pro místní trasy[11].

2.5.4 tag `operator`

Značka `operator` se používá pro vyznačení názvu společnosti, osoby nebo jiné entity, která provozuje daný objekt na mapě[9]. Turistické trasy v České republice spravuje Klub českých turistů.

2.5.5 tag `kct_<barva>`

`<barva>` je proměnná část klíče. Proměnná část může nabývat hodnot `red`, `green`, `blue` nebo `yellow`. Hodnota značky upřesňuje značku `route`.

Používané hodnoty pro značení turistických tras:

- `ski` – lyžařská stezka
- `bicycle` – cykloturistická stezka
- `wheelchair` – stezka pro vozíčkáře
- `horse` – stezka pro jezdce na koni
- `major` – pásové značení
- `local` – místní značení
- `learning` – naučná stezka
- `ruin` – odbočka ke zřícenině
- `spring` – odbočka ke studánce/pramenu
- `interesting_object` – odbočka k jinému zajímavému objektu
- `peak` – odbočka k vrcholu nebo vyhlídce

2.5.6 tag `osmc:symbol`

Slouží jako pomocná značka pro snadné vykreslování turistických tras. Značku lze použít ke kontrole značek popisujících turistickou trasu.

2.5.7 tag `complete`

Slouží pro označení zmapovanosti relace. Pokud je relace kompletně zmapována, značka nabývá hodnoty `yes`. Tag `complete` může nabývat hodnot `yes/no`.

2.5.8 tag `abandoned`

Slouží pro označení prvků, které jsou opuštěné, ale stále jsou viditelné v krajině. Takové prvky jsou užitečné pro navigaci v krajině. Opravení prvků není plánováno. Tag může nabývat hodnoty `yes/no` nebo může být použit jako prefix jiné značky[8].

Kapitola 3

Možnosti kontroly OSM dat

S růstem počtu uživatelů a vložených dat roste i míra inkonzistence dat. Do projektu se zapojuje více uživatelů, kteří nejsou seznámeni s pravidly a konvencemi projektu OpenStreetMap. Vložené data je zapotřebí kontrolovat, ať už ručně za pomocí uživatelů nebo automaticky za pomocí vytvořeného programu.

3.1 Manuální metody kontroly

Mezi manuální metody lze zařadit takové metody, u kterých je zapotřebí interakce uživatele. Tato metoda závisí na počtu uživatelů vyskytujících se v dané lokalitě. V oblastech s menším výskytem obyvatel, a tedy i menším počtem uživatelů OpenStreetMap, trvá kontrola delší časový interval, než ve srovnání s oblastí s větším počtem uživatelů.

Jednou z metod manuální kontroly dat lze zařadit kontrolu dat za pomocí jiných zdrojů. Jako zdroj pro ověření lze použít jiné projekty podobného zaměření, jako je nesvobodný projekt Google Maps. Data z Google Maps lze použít pouze pro kontrolu nebo pro zorientování v dané oblasti. Data nelze použít přímo z důvodu nekompatibility licence Google Maps a projektu OpenStreetMap. Další z možností kontroly dat je pomocí leteckých map. Polopropustná vrstva mapy je podložena vrstvou s leteckým snímkem oblasti. Za pomocí letecké mapy lze kontrolovat obrysy objektů nebo linie cest.

Další z možností manuální kontroly dat je za pomocí znalosti daného prostředí uživatelem. Uživatel kontroluje daný prvek podle svých znalostí, popřípadě svých poznámek. Přesnost metody závisí na uživateli. Metodu lze zpřesnit za pomocí GPS záznamu při fyzickém průchodu dané trasy. Při použití GPS záznamu záleží na přesnosti daného GPS zařízení.

3.2 Automatické metody kontroly

Automatické metody kontroly jsou takové, u kterých není zapotřebí ke kontrole uživatel. Časová složitost takové kontroly může být v řádech setina až po desítky minut. Kontroluje se, zda jsou dodržena stanovená pravidla. Výhodou takového testu je šíře kontroly. Lze jednoduše otestovat všechny prvky pro celou planetu, avšak lze kontrolovat i určitou geografickou část.

Přiřazování značek k jednotlivým prvkům má svá pravidla. Jednotlivé značky je zapotřebí kontrolovat, zda dodržují určená pravidla. Příkladem takového pravidla může být

hodnota značky `complete`, která může nabývat hodnot `yes` nebo `no`. Pro kontrolu obsahu značek lze využít aplikaci Osmose.

Mezi jeden ze speciálních případů automatické kontroly může patřit kontrola lokálních pravidel. Pro české prostředí platí určitá pravidla pro značení turistických tras. Tato pravidla jsou popsána projektem OpenTrackMap (OTM)[1]. Turistické trasy kontroluje aplikace OsmHiCheck[2], kterou vytvořil pro bakalářskou práci student FIT VUT Petr Švaňa[25].

3.3 Osmose

Projekt Osmose[3] je zaměřen na automatickou kontrolu OSM značek pro celou planetu. Jednotlivé analýzy jsou řazeny do skupin podle zaměření dané analýzy. Hlavním prvkem celé aplikace je mapa, která je překryta vrstvou s chybnými značkami. Značky lze zobrazit i v tabulkovým výpisu.V projektu lze analyzovat relace. Analýza relací je umožněná pouze jednotlivě po zadání ID relace. Analýzu dat lze provádět i pro data přidaná nebo upravená od daného uživatele. Uživatele lze zvolit podle jména, pod kterým se registroval do projektu OpenStreetMap. Pro snadné zpracování nalezených chyb jinými projekty lze chyby exportovat v mnoha formátech (HTML, JOSM, OSM, RSS, GPX, Json). V aplikaci je možné zobrazit statistické údaje. Statistika je dostupná pro jednotlivé regiony v tabulkovým výpisu. Rozhraní aplikace je lokalizované do několika jazykových mutací. Jednou z obsažených jazykových mutací je i čeština.

3.4 OsmHiCheck

4.1 Projekt OsmHiCheck[4] je úzce zaměřen pro kontrolu pravidel OpenTrackMap. Projekt umožňuje zobrazit prvky nevyhovující kontrole umístěné v mapě nebo v tabulkovým výpisu. Mapové rozhraní umožňuje zadávat a upravovat data rozcestníků. Pro vložení a úpravu dat z jiného projektu je umožněno nahrát data ve strojově čitelném formátu JSON. Projekt OsmHiCheck obsahuje následující kontroly:

- **Nevyplněné tagy** – Analyzuje, zda jsou zadány všechny značky definované podle OTM značkového klíče. Analýza je prováděna nad daty uloženými v databázi.
- **Hodnoty tagů ve špatném formátu** – Projekt OpenTrackMap definuje u některých značek i možné hodnoty, které daná značka může obsahovat. Tato pravidla nemusí být známa pro všechny uživatele, proto je třeba hodnoty OTM značek analyzovat.
- **Konflikt hodnot tagů** – V projektu OpenTrackMap jsou definice některých značek vzájemně obsahem totožné. Příkladem takových značek mohou být značky `kct_<barva>` a `osmc:symbol`. Další z možností výskytu konfliktu může nastat u značek popisujících typ trasy. Typ trasy popisují značky `route` a `kct_<barva>`.
- **Kontrola rozcestníků** – Ryze českým projektem, který je tvořen uživateli, je projekt PhotoDB[10]. Cílem projektu je vytvořit databázi geolokalizovaných fotografií rozcestníku a informačních tabulí. Za pomocí nahraných fotografií lze ověřovat data

[1] http://wiki.openstreetmap.org/w/index.php?oldid=1452611
[2] Stránky projektu OsmHiCheck: http://osm.fit.vutbr.cz/OsmHiCheck/
[3] Stránky projektu Osmose: http://osmose.openstreetmap.fr/
[4] Stránky projektu OsmHiCheck: http://osm.fit.vutbr.cz/OsmHiCheck/

uložená v OpenStreetMap. Příkladem kontroly může být jméno dané lokality, přesné umístění rozcestníku nebo kontrolu sousedního cíle uvedeného na směrnici rozcestníku.

V projektu OsmHiCheck lze analyzovat zda jsou k rozcestníku přiřazeny fotografie. Rozcestníky jsou evidovány svým referenčním číslem, které je uvedeno na tabulce s jménem lokace. Referenční číslo je třeba analyzovat, zda je uvedeno v datech OSM. Po nahrání fotografie je přiřazen k fotografii rozcestník. Při nedokončení přidání rozcestníku může existovat mnoho fotografií zabírající kapacitu disku zbytečně. Pro kontrolu takových fotografií je v projektu OsmHiCheck dostupný výpis nepoužitých fotografií.

Kapitola 4

Návrh aplikace

Pro návrh aplikace je třeba správně pochopit danou problematiku. Můj návrh vychází z již dříve realizované práce OsmHiCheck. Z projektu OsmHiCheck přebírá zkušenosti, možnosti testování a část databázové struktury.

4.1 Aktuální řešení – OsmHiCheck

Pro vyvarování se stejných nedostatků a chyb, je potřebné aplikaci OsmHiCheck řádně analyzovat a nalezené nedostatky a chyby řešit. Níže je uvedena část nedostatků a chyb, které je potřebné v navrhované aplikaci opravit.

Při procházení rozhraní aplikace OsmHiCheck lze narazit na 3 typy rozhraní. Mezi tato rozhraní patří rozhraní mapy, analýza turistických map a analýza rozcestníků. Pro přístup k jednotlivým rozhraní je potřebné sestoupit vždy na úvodní stránku. Rozhraní analýz turistických tras a analýz rozcestníku jsou rozdílné, avšak mají podobnou funkcionalitu. Při procházení rozhraním novým uživatelem, může být uživatel dezorientovaný. Aplikace by měla rozhraní se stejnou funkcionalitou sjednotit, tak aby se nový uživatel lehce orientoval.

Pro grafické zobrazení statistik slouží v aplikaci liniový graf. Graf je dostupný pouze pro relace s chybějícími tagy a relace s chybami. Pro zobrazení liniového grafu je zapotřebí ukládat data do databáze. Při nahlédnutí do databázové struktury lze najít dvě databázové tabulky pro uložení statistik – `gp_stats` a `stats` plnící obdobný problém. Pro uložení počtu prvků vyhovujících danému testu slouží samostatný atribut v databázové tabulce. Při přidání nebo odebrání typu analýzy je zapotřebí pozměnit databázovou strukturu. Se změnou databázové struktury souvisí i změna implementace aplikace.

Při zobrazení webové stránky na mobilním zařízení se aplikace nepřizpůsobí velikosti zobrazovacího zařízení. Text načtené webové stránky je příliš malý, na to aby byl čitelný. Čitelné a lehce ovladatelné rozhraní aplikace na mobilním zařízení je důležité pro mapování v terénu.

Při procházení aplikace OsmHiCheck lze narazit na odkazy směřující na adresu `localhost`. Adresa `localhost` je adresa směřující na počítač, odkud byla adresa zavolána. Tato adresa se využívá při vývoji. Při přenosu aplikace z vývojového serveru na produkční je zapotřebí tuto adresu zaměnit za adresu používající produkčním serverem.

Součástí URL adresy je i část udávající jméno souboru a cestu k souboru, který má danou akci vykonat. Pro zvýšení bezpečnosti je vhodné schovat adresářovou strukturu aplikace. Pro snadnou zapamatovatelnost URL adresy je vhodné používat krátké adresy s minimem speciálních znaků.

Pro vývoj a přidání nové analýzy do aplikace nejsou dostupné žádné nástroje, které by urychlily vývoj a přidání analýzy. Mezi základní nástroje ulehčující práci patří nástroj pro přístup k databázi, načítání konfigurace nebo zaznamenávání chyb.

K databázovému systému je omezen počet spojení, které mohou být ve stejný časový okamžik aktivní. K databázi je vhodné se připojit až v případě potřeby databáze. Nástroj pro přístup k databázi by měl zaručit pouze jedno aktivní spojení k databázi pro danou instanci aplikace. V aplikaci není obsažen nástroj zprostředkovávající výše popsané vlastnosti. Při vyčerpání všech spojení je další pokus o spojení neúspěšný.

Pro vývoj webových stránek je využíváno několik odlišných technologií. Tyto technologie je vhodné navzájem oddělit. Oddělením různých technologií přináší do aplikace vyšší přehlednost, modularitu, znovupoužitelnost. V aplikaci jsou různé technologie použity dohromady. V jednom souboru dochází k řízení aplikace, analýze prvků a výpis prvků pomocí HTML. Při potřebě zaměnit výpis prvků z HTML na JSON, bude zapotřebí přepsat větší část aplikace. V aplikaci je vhodné oddělit analýzu, řízení programu a výpis do samostatných částí.

4.2 Specifikace etap vývoje aplikace

Vývoj aplikace byl rozdělen do dvou etap.

- **Framework**
- **Kontrola OSM dat**

4.3 Framework

První etapa vývoje je zaměřena na vývoj frameworku. Hlavním úkolem frameworku je řízení běhu aplikace, obsluha požadavků, ošetření vstupů a výstupů. Framework zpřístupňuje nástroje pro práci s databází, obsluhu chyb a výjimek nebo nastavení parametrů aplikace. Framework je v aplikaci základním kamenem, nad kterým je postavena celá aplikace. Při špatném návrhu frameworku může být ovlivněna celková kvalita výsledné aplikace.

Na trhu je dostupných mnoho PHP frameworků. Použití hotového frameworku jsem zavrhl a vyvinul jsem vlastní řešení. Důvody nepoužít framework je několik. Dostupné frameworky obsahují mnoho komponent, které v této aplikaci nebude využito. Příkladem takových komponent může být komponenta pro výpis, řízení a zpracování odeslání formuláře. S rozsáhlostí frameworku souvisí i zvýšení složitosti vývoje. Pro začátek vývoje je zapotřebí mít základní znalosti o frameworku.

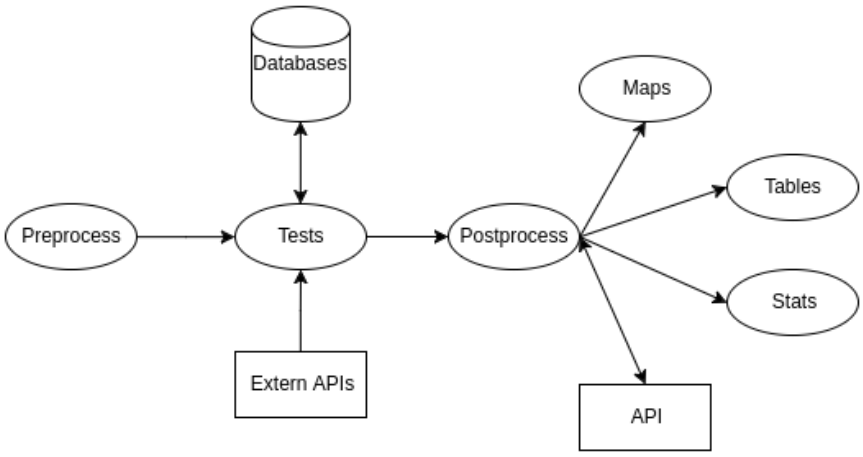
Webové aplikace jsou specifické z hlediska použitých technologií. Při vývoji aplikace se využívá kombinaci několika různých technologií. Příkladem kombinace použitých technologií může být PHP, PostgreSQL, HTML, CSS a JavaScript. Použitím kombinace technologií přináší úskalí. Oddělením jednotlivých technologií se zvýší modulárnost a přehlednost aplikace. Pro dosažení požadovaných vlastností byla zvolena softwarová architektura MVC. Zvolená architektura ovlivnila adresářovou strukturu aplikace.

4.4 Kontrola OSM dat

Po vyvinutí a otestování frameworku, který tvoří základní část celé aplikace, vývoj pokračoval v 2. etapě. 2. etapa je zaměřena na kontrolu OSM dat. Pro návrh architektury kontroly

OSM dat je potřebné danou problematiku pochopit. K pochopení principů kontroly OSM dat posloužily aplikace Osmose 3.3 a OsmHiCheck 4.1.

Navržené řešení se skládá z několika částí, lišící se svojí funkcí.



Obrázek 4.1: Architektura

Z diagramu výše, je patrné, že se skládá ze tří částí (preprocess, tests, postprocess), které volají nebo mohou být volány dalšími částmi aplikace.

4.4.1 Preprocess

Preprocessor slouží k přípravě testů, nastavení potřebných parametrů a následně dané testy spouští. Součástí preprocessoru může být i stáhnutí dat z jiných zdrojů. Preprocessor může být spouštěn při načtení stránky nebo periodicky v daný čas (např. pomocí softwarového daemona CRON).

4.4.2 Tests

Kontroly lze začlenit podle jejich charakteristiky do dvou skupin. Pozorováním spotřebovaných zdrojů a časového intervalu, které jsou pro kontrolu potřeba, lze zařadit do jedné ze skupin, které jsou popsány níže.

Realtime test

Kontrola OSM dat se provede u realtime testů vždy při potřebě znát výsledek. Výsledek u takového testu je vždy přesný. Ve většině případu realtime testů jsou pro vykonávání potřeba pouze data uložená v databázi. Samotná kontrola je založena na položení vhodného SQL příkazu databázi.

Příklady realtime testů:

- hodnota network nekoresponduje s hodnotou kct_*:.
- typ cesty v rozporu s ostatnimi u tagu osmc:symbol, kct_* nebo route:.

- hodnota network u cyklotrasy nekoresponduje s cislem v ref:.
- barva cesty v rozporu u osmc:symbol a kct_*:.
- chybné hodnoty tagu network, complete, osmc:symbol:, route:, kct_*:.
- chybějící hodnoty tagu network:, complete:, osmc:symbol:, destinations:.

Longtime test

Jsou to takové testy, u kterých je jejich vykonání náročné a není vhodné je spouštět při každém načtení stránek. Nemusí se jednat pouze o časovou náročnost ale i o vyšší nároky na paměť či výpočetní kapacitu. Testy budou spouštěny periodicky v určitý čas. Test se bude vykonávat pro celou oblast testování (například pro celou Českou Republiku). Výsledek testu je permanentně uložen v databázi. Při potřebě výsledku testu se distribuují data uložená z databáze. Nevýhodou takového řešení může být neaktuálnost výsledku. Tuto nevýhodu lze odstranit ručním spouštěním testu.

Příklady longtime testů:

- kontrola fotky rozcestníku a její polohy.
- kontrola kvality fotek rozcestníku (barevný tón, světelnost, ...).
- kontrola sousedních regionů a jejich vzájemný překryv (kolize).
- vyhledání sousedních regionů a děr mezi regiony.
- vyhledání osamocených cest bez napojení na další cestu.
- porovnání entit, relací, bodů s mapou z jiného zdroje, kontrola vzájemné odchylky dvou entit, relací, bodů.
- statistické testy (délka silnic, počet silnic, počet budov, ...).
- vyhledávání velkých ploch bez dalších entit.
- vyhledání zastavěných ploch bez vyznačených hranic budov.
- kontrola propojení pro Wikipedia a Wikimedia.
- kontrola GPS souřadnice v EXIFu fotky s umístěním fotky.

Extern API

Prováděné testy mohou vyžadovat data uložená mimo aplikaci. Jeden z příkladů externího zdroje může být databáze informací o fotkách rozcestníků. Pro kontrolu rozcestníků se stáhne za pomocí API potřebné informace a provede se kontrola s OSM daty uloženými v databázi. Dalším zdrojem informací je webová encyklopedie Wikipedia.

4.4.3 Postprocess

Postprocessor transformuje dodané data z testů a zpřístupňuje je pro další zpracování. Způsoby jakým mohou být data zpracována jsou popsány níže.

Maps

V tomto rozhraní je zobrazena mapa se základními ovládacími prvky potřebnými k ovládání mapy. Mapa je překryta poloprůhlednou vrstvou zobrazující prvky nevyhovující provedené kontrole. Součástí mapy je i ovládací prvek pro výběr zobrazované vrstvy.

Tables

V rozhraní tabulek lze zobrazit všechny typy kontrol. U jednotlivých kontrol je vyjádřeno procentuálně kolik prvků danou kontrolou neprošlo. Z výpisu všech kontrol lze přistoupit k detailu kontroly. V detailu kontroly jsou vypsány všechny prvky, které dané kontrole nevyhovovaly.

Stats

Pro zobrazení průběhu chyb v čase je v aplikaci přístupné rozhraní s grafickým zobrazením. Vhodným prvkem pro grafickou reprezentaci je lineární graf. Pro zobrazení grafu je potřebné periodicky ukládat data do databáze. Ukládaná data se skládají z počtu celkem testovaných prvků, počtu nevyhovujících prvků a data pořízení.

API

Pro zpřístupnění dat navazujícím projektům je součástí aplikace rozhraní, poskytující výsledky kontrol. Výsledky mohou být distribuovány za pomocí souboru, který bude pravidelně aktualizován. Pro formát souboru je zvolen strojově čitelný formát JSON nebo XML.

4.4.4 Ověření správnosti

Po návrhu aplikace je zapotřebí vytvořit sadů referenčních testů, které má dané řešení vykonávat a aplikovat je. Při aplikaci testů je zapotřebí sledovat, zda navržené řešení neomezuje provádění referenčních testů. Pokud řešení omezuje provádění referenčních testů, je zapotřebí se zamyslet, zda tento typ testů je třeba uvažovat a přepracovat navržené řešení, nebo daný referenční test vypustit.

4.4.5 Možnost rozšíření aplikace

Možnost rozšíření aplikace je systém upozornění, který by kontroloval rozdíl počtu navrácených bodů, uzlů či relací než v předchozím provádění testu. Pokud by rozdíl byl větší než zadaná odchylka, systém by vygeneroval upozornění. Upozornění by byla zasílána pomocí emailu zodpovědným osobám nebo zobrazován výpis na stránkách aplikace. Výše popsaný systém je vhodný pro včasné zachycení velkého nárůstu počtu chyb.

Kapitola 5

Implementace aplikace

Aplikace je implementována jako webová stránka. Vývoj aplikace započal vývojem frameworku a základních komponent aplikace. V této kapitole jsou uvedeny implementační detaily a problémy, které se při vývoji objevily.

Pro vývoj aplikace byli zvoleny následující technologie:

- **PHP7** – objektově orientovaný programovací jazyk
- **PostgreSQL** – objektově relační databáze
- **HTML5** – značkovací jazyk
- **CSS3** – kaskádové styly
- **JavaScript** – objektově orientovaný programovací jazyk

5.1 Architektura aplikace

Pro architekturu aplikace byl zvolen softwarový návrhový vzor MVC. Webová aplikace se skládá z několika na sobě samostatných částí tvořených z různých technologií. Pro přehlednost a snadnou rozšiřitelnost aplikace je vhodné použít prostředky, které aplikaci rozdělí do oddělených částí.

Aplikace je složena z tříd reprezentující logiku aplikace, databáze a operacích nad databází. Pro reprezentaci obsahu je zapotřebí vrstva, která vytvoří výslednou stránku. Pro řízení aplikace je zapotřebí vrstva, která spojuje předchozí dvě vrstvy a řídí jejich činnost.

5.1.1 MVC

Architektura MVC rozděluje aplikaci do 3 samostatných vrstev, tak že jednotlivé vrstvy se vzájemně minimálně ovlivňují. Architektura je logicky dělena na datový model, řídící logiku a uživatelské rozhraní[1].

Model

Model reprezentuje informace aplikace a logiku, která nad těmito daty provádí operace. Model si můžeme představit jako databázi a objekty, které tyto data zapouzdřují do logických celků.

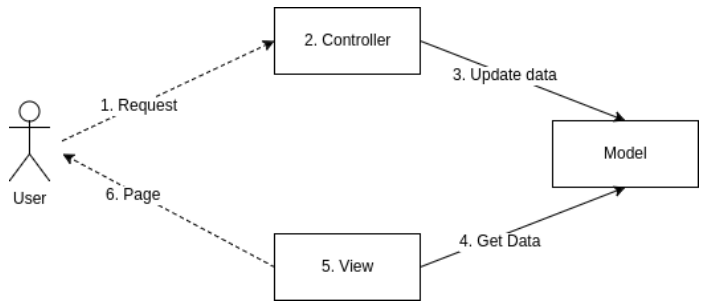
View

View neboli pohled transformuje dodané data od modelu do přívětivé formy pro zobrazení dat uživateli. Pohled obsahuje pouze nezbytnou logiku pro zobrazení dat. Vícero pohledů může využívat jeden model.

Controller

Controller zajišťuje provedení událostí přijaté od uživatele a rozhodne o dalším pokračování ve vykonávání. Jeden z modelů je i router controller, který parsuje HTTP adresu a volá podle zadané adresy patřičný controller.

Životní cyklus stránky



Obrázek 5.1: Architektura MVC

1. Uživatel vytvoří požadavek, který je zaslán controlleru.
2. Požadavek zachytí controller, rozhodne v dalším pokračováním.
3. Controller upraví data v modelu, pokud je to nezbytné.
4. Controller předá řízení aplikace pohledu. Pohled načte odpovídající model.
5. Pohled zpracuje data a sestaví výslednou stránku.
6. Stránka je odeslána zpět uživateli.

5.2 Adresářová struktura

Ve výchozím nastavení je umožněno klientovi přistupovat ke všemu souborům a složkám. Klient může tedy nekontrolovaně přistupovat a vykonávat programy, které jsou ve složkách obsaženy. Pro zvýšení bezpečnosti je zapotřebí zajistit nepřístupnost programové části z klienta.

Jeden ze způsobů zamezení přístupu je využít možností operačního systému a jeho nastavení přístupových práv k souborům a složkám. Pro zabezpečení souborů a složek odebereme přístupová práva danému klientovy. Odebráním přístupových práv znemožníme přístup k souborům a složkám i webovému serveru, který dané soubory potřebuje k vykonávání. Vhodnější způsob je využít možností webového serveru, který umožňuje zakázat přístup

klientů k určité části stromové struktury adresářů. Omezit přístup k souborům a složkám lze nastavit přímo v konfiguračním souboru webového serveru. Avšak tahle možnost nastavení omezení přístupu klienta je nepraktická. Při změně webového serveru je zapotřebí mít na paměti nutnost provést patřičné nastavení v konfiguračním souboru webového serveru. Z tohoto důvodu je lepší mít konfiguraci webového serveru přímo v aplikaci. Vhodným způsobem je provést konfiguraci za pomocí souboru `.htaccess`, přes který lze nastavit konfiguraci webového serveru[19].

Listing 5.1: Direktiva pro zamezení přístupu k souborům a složkám klientovi.

```
Order Allow,Deny
Deny from all
```

Výše popsané nastavení je aplikováno pro adresář `app`. Obsahem složky je programová část aplikace. Přímým spuštěním skriptů může dojít k vyvolání chyb. V horším případě může být porušena integrita uložených dat, nebo k úniku citlivých dat z konfiguračních souborů.

```
Osm
├── app
│   ├── classes
│   │   ├── exceptions
│   │   └── tests
│   ├── config
│   ├── controller
│   └── view
├── bin
├── init
└── www
    ├── css
    ├── data
    ├── fonts
    ├── images
    ├── js
```

Obrázek 5.2: Stromová struktura adresářů aplikace

Podadresáře složky `app` tvoří programovou část aplikace. Ve složce `classes` jsou uloženy pomocné třídy a samotné testy. V architektuře MVC odpovídá složka `classes` svojí funkcionalitou modelům. Obsah složky `view` je tvořen pohledy ve formátu `phtml`. Pro řízení běhu aplikace slouží kontrolery, které jsou umístěny ve složce `controller`. Konfigurační soubory jsou uloženy v podadresáři `config`.

Pro přímé načtení souborů a spouštění skriptů je určena složka `bin`. V případě potřeby přístupu k souborům, které jsou umístěny v chráněné části (`app`), lze přistupovat za pomocí nástrojů jazyka PHP. Vhodným nástrojem pro načtení obsahu souboru je vestavěna funkce `file_get_contents`. Pro připojení PHP skriptu je vhodné použít funkci `include` nebo `required`. Chování funkcí `include` a `required` se liší v případě neúspěchu načtení souboru. Při nenačtení souboru za pomocí funkce `required` končí vykonávání programu. V případě použití funkce `include` se pokračuje v provádění programu dále.

Adresářová struktura je navržena tak, aby oddělila datovou část od programové části. Datová část je obsažena ve složce `www`. Výjimku tvoří podadresář `js`, ve kterém jsou uloženy JavaScript soubory. JavaScript soubory se spouští až na straně klienta a tudíž jsou

pro webový server reprezentovány jako běžné datové soubory. Další výjimkou je soubor `index.php`. Klientovi je umožněno komunikovat pouze s rozhraním aplikace přes přístupový bod. Přístupovým bodem v aplikaci je reprezentován souborem `index.php`.

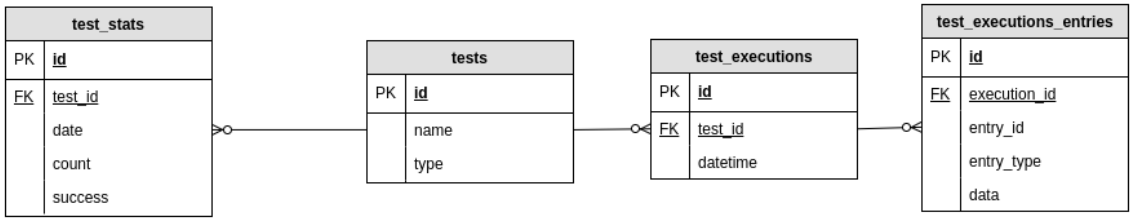
5.3 Databázová vrstva

Pro permanentní uložení dat je využit databázový systém PostgreSQL[1]. Pro snadnou manipulaci s geografickými objekty je využito rozšíření databáze PostgreSQL PostGIS[2]. Rozšíření PostGIS přináší možnost ukládání geometrické typy jako jsou uzly, polygony nebo řetězce bodů. Mezi další funkce, které jsou nezbytné pro práci s geometrickými typy, patří funkce pro určení vzdálenosti dvou geometrických prvků, délka linie nebo funkce pro určení výměry ploch.

5.3.1 ER diagram

Pro uložení informací o vykonávání jednotlivých testů je zapotřebí navrhnout databázovou strukturu. Ve struktuře je zapotřebí uložit informace o jednotlivých testech. Každý test může obsahovat několik variací testů, které testují podobný fakt. Jednotlivé varianty testů je potřeba uložit v databázové struktuře. Některé testy jsou svojí charakteristikou nevhodné provádět při každém spuštění. Výsledek takového testu je zapotřebí uložit do databáze. Společně s testem, kterým byl proveden je zapotřebí uložit i časové razítko vykonání testu. Časové razítko je zapotřebí pro určení, zda daný test je aktuální a dané hodnoty jsou platné, nebo test již aktuální není a data uložená v databázi jsou již neplatná. Při provádění testu se vyhodnocují jednotlivé entity, které je zapotřebí evidovat, zda danému testu vyhovují. Jednotlivé entity mohou pro svoji činnost vyžadovat zaznamenání dodatečných informací. Dodatečnou informací mohou být např. čísla rozcestníků. Pro záznam statistik a jejich následné zobrazení za pomocí liniového grafu je zapotřebí evidovat počet entit, které danému testu vyhovují a také počet celkem testovaných entit. K počtu vyhovujících entit a celkového počtu testovaných entit je zapotřebí uložit i datum, kdy byl záznam pořízen. Pokud jsou již hodnoty v daný den uloženy, hodnoty boudou aktualizovány.

Z výše uvedeného popisu problému byl navrhnut Entity-relationship diagram, který znázorňuje množiny a kardinalitu vztahů jednotlivých množin.Ve schématu 5.3 je hlavním prvkem množina `tests`, která obsahuje položky jméno a variaci daného testu. Jednotlivé spouštění testů se uchovávají v množině `test_executions`, která obsahuje položku pro zaznamenání časového razítka. Jednotlivé entity se vyskytují v množině `test_executions_entries`.



Obrázek 5.3: ER diagram

[1]Stránky projektu PostgreSQL: https://www.postgresql.org/
[2]Stránky projektu PostGIS: http://postgis.net/

5.3.2 Vytvoření databázové struktury

Před nahráním OSM dat je potřebné připravit prostředí pro uložení dat. Pro aplikaci je zvolen databázový systém `PostgreSQL` s rozšířením `postgis`. Rozšíření `postgis` je nezbytné pro databázové schéma `pgsnapshot`. Pro vytvoření schématu lze použít importní SQL skript, který je distribuovaný s nástrojem `Osmosis`[25].

Pro vytvoření potřebné databázové struktury uchovávající data pro kontrolu prvků je součástí aplikace přiložen importní skript. Importní skript je obsažen ve složce `init` soubor `dump.sql`. OSM data jsou uloženy společně v jedné databázi s daty výsledky testů.

5.3.3 Import OSM dat

Po úspěšné konfiguraci databáze lze nahrát OSM data. Import kompletních OSM dat je časově náročný. Pro úsporu času lze importovat pouze změny, které byli provedeny. Nejdříve je však nutné vložit všechna OSM data do databáze.

- **A) Stáhnutí dat.** Aplikace je zaměřena pouze na kontrolu dat v České Republice. Denně aktualizovaná data pro tuto zemi lze stáhnout z datového skladu serveru geofabrik.de[3]. Formát staženého souboru volíme nejmenší nabízený – `pbf`[25].

- **B) Konverze formátu.** Pro změnu formát z binárního `pbf` na textový `o5m` lze použít nástroj `osmconvert`. Změna formátu je provedena pro vyšší efektivitu následujícího kroku[25].

```
osmconvert downloaded_dump.osm.pbf converted_dump.o5m
```

- **C) Odfiltrování nepotřebných dat.** V aplikaci jsou využita pouze data popisující turistické trasy. Nepotřebná data odfiltrujeme.

```
osmfilter converted_dump.o5m --keep-ways="highway="
--keep="operator=cz:KCT" --out-o5m > filtered_dump.o5m
```

- **D) Konverze formátu zpět[25].**

```
osmconvert filtered_dump.o5m first_import.pbf
```

- **E) Import do databáze[25].**

```
osmosis --rb first_import.pbf --wp
database=... user=... password=...
```

5.3.4 Periodická aktualizace OSM dat

Prvotní import OSM dat do databáze je časově náročný. Pro snížení časové náročnosti lze vložit do databáze pouze změny, které byli provedeny od poslední aktualizace databáze.

- **A) Stažení aktualizace.** Pro vytvoření `OsmChange` je zapotřebí uchovat původní soubor, který byl importován. Původní soubor lze po tomhle kroku smazat a označit soubor stažený v tomhle kroku jako původní. Program `osmupdate` shromažďuje změny provedené v různých časových intervalech.

[3] http://download.geofabrik.de/

```
osmupdate converted_dump.o5m u_converted_dump.o5m
```

- **B) Odfiltrování nepotřebných dat[25].**

```
osmfilter u_converted_dump.o5m --keep-ways="highway="
--keep="operator=cz:KCT" --out-o5m > u_filtered_dump.o5m
```

- **C) Vytvoření OSM change souboru[25].**

```
osmconvert filtered_dump.o5m u_filtered_dump.o5m --diff
-o=u_database.osc
```

- **D) Aktualizace databáze[25].**

```
osmosis --rxc file=u_database.osc --wpc
database=database=... user=... password=...
```

5.4 Hezké URL

Pro přístup k jednotlivým webovým stránkám je zapotřebí zadat URL adresu. Adresa URL se skládá z následujících částí: protokol, doména, cesta. Protokol a doména jsou pro aplikaci neměnné konstanty. V aplikaci se mění část URL za doménou – cesta. Cesta se skládá z částí složka, soubor, parametry.

Listing 5.2: Ukázka běžných adres.

```
http://www.domena.cz/index.php?mod=tables&action=missing&type=dest
```

Při použití běžné URL odkrýváme adresářovou strukturu aplikace. Potencionálnímu útočníkovy URL prozradí, který soubor operaci zpracovává a ve které složce je soubor umístěn. Další nevýhodou použití takové adresy je nepřehlednost. Běžný uživatel nerozumí, co adresa vyjadřuje a jak je členěna.

Listing 5.3: Ukázka hezké url.

```
http://www.domena.cz/tables/missing/destinations
```

I při neznalosti dané aplikace lze přímo z URL odvodit, co daná stránka vykonává. Použití metody hezké URL se zvýší přehlednost pro uživatele tak i pro vyhledávače. Zobrazovaná URL adresa je kratší a lépe zapamatovatelná pro běžného uživatele. Pro potencionální útočníky je skrytá vnitřní adresářová struktura aplikace. Před potencionálním útočníkem je schováno jméno souboru, který akci zpracovává.

Pro implementaci hezké URL je zapotřebí upravit nastavení webového serveru. Pro ovlivnění nastavení webového serveru je třeba vložit do souboru `.htacess` pravidla. Aplikace poté zpracuje URL adresu z globálního pole `$_SERVER`.

Listing 5.4: Pravidla pro hezké URL

```
RewriteEngine On
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule ^(.*)$ /www/$1
```

5.5 Ošetření chyb a vyjímek

Programovací jazyk PHP je jazyk, který má za sebou 22 roků vývoje. Za tu dobu se změnila paradigma jazyka z čistě procedurálního na kombinaci objektového a procedurálního paradigma[3]. Způsob vyhodnocování chybového stavu operace u procedurálního paradigma, je různé, jak u paradigma objektového. U procedurálního paradigma se využívá návratového stavu funkce, popřípadě chybového hlášení. U objektového paradigma při chybě nebo neočekávaném stavu operace se vyvolá výjimka.

5.5.1 Chyby v PHP

V PHP může být vyvolána chybová hlášení několika úrovní. Daná úroveň chyby je definovaná konstantou. Níže přehled nejčastěji vyvolaných úrovní chyb[24].

- `E_NOTICE` - Potencionální chyba. Program se vykonává dále.
- `E_PARSE` - Syntaktická chyba. Daný program se vůbec nespustí.
- `E_WARNING` - Chyba, pokračuje se dále ve vykonávání aplikace.
- `E_EROR` - Fatální chyba, po které nelze dále pokračovat. Vykonávání programu se ukončí.
- `E_ALL` - Všechny úrovně chyb.

Zobrazení chyb na produkční serveru je nevhodné. Chybová hlášení nejsou pochopitelná pro uživatele dané aplikace. V některých případech může být zobrazení chyb i nebezpečné. Ve vygenerovaném chybovém hlášení může být obsažena část kódu, ve kterém jsou uloženy citlivé údaje. Například při chybném volání PHP funkce `pg_connect` pro připojení k databázi PostgreSQL, jsou zobrazeny citlivé údaje pro připojení k databázi.

Další z problémů je vyvolání chyby úrovně `E_ALL`, popřípadě jiné úrovně chyby, která ukončí provádění programu. Ve výchozím stavu se nelze zotavit z chyby a danou chybu zaznamenat nebo odeslat administrátorovy dané webové stránky. Vyvolání chyby a ukončení vykonávání programu, může také způsobit nekonzistenci zpracovávaných dat. Při sekvenčním zpracováváním po vyvolání chyby a ukončení programu, zůstane část dat nezpracovaná.

Zobrazení chyb a jejich zaznamenání do souboru lze ovlivnit nastavením patřičných konstant v konfiguračním souboru, v souboru .htaccess nebo přímo v programu pomocí příslušných funkcí, které nastaví konstantu. Pro nastavení konfigurace z programu lze zavolat funkci `ini_set;`, kterou lze ovlivnit hodnotu konstant uvedených níže [23].

- `display_errors` - konstanta k zobrazení/schování chybových hlášení. Na produkčním serveru je vypisování chyb vypnuté.
- `error_reporting` - konstanta pro nastavení úrovně vypisování chyb.

I když je možno nastavit zobrazení chyb a jejích zaznamenání do souboru, z vyvolané chyby se nelze nijak zotavit, nebo zaslat chybu emailem administrátorovi. Další z nepříjemností jsou nové konstrukce jazyka PHP, kde se chybové stavy ošetřují za pomocí vyvolání výjimek. Pro sjednocení a čistější kód lze převést chybu na výjimku za pomocí PHP funkce `set_error_handler`. Funkce `set_error_handler` volá callback funkci, ve které dojde ke zpracování chyby a vyvolání výjimky. Pro rozeznání, co výjimku vyvolalo, se vyvolá objekt třídy, který dědí od třídy `Exception`.

5.5.2 Vyjímky v PHP

V moderních objektově orientovaných jazycích se využívají pro ošetření chybových stavů výjimky[2]. V bloku `try` se provede kód, který je rizikový a může vyvolat výjimku, která je následně zachycena v bloku `catch`. `Catch` blok může být obsažen násobně s rozdílným typem parametru. Jako poslední by měl být uveden blok `catch` s parametrem typu `Exception`. Efektivnější řešení je nastavit callback pomocí funkce `set_exception_handler`. V callback funkci dojde podle nastavené úrovně běhu aplikace obsluha. Pokud aplikace běží v produkční verzi, výjimka se zaznamená a zobrazí se chybová stránka. Při běhu aplikace ve vývojové verzi je chyba vypsána na výstup.

5.6 Konstanty

V jazyce PHP jsou dostupné dva způsoby definic konstant, klíčové slovo `const`[21] a definice pomocí vestavěné funkce `define()`[22]. Hlavní rozdíl mezi konstantou definovanou za pomocí funkce a konstantou definovanou pomocí klíčového slova je rozsah viditelnosti konstanty a době definice konstanty[4]. Při použití funkce `define()` je obsah konstanty definován až při vykonávání programu. Při použití klíčového slova `const` je obsah konstanty definován již při překladu programu. Další z rozdílu je rozsah platnosti konstanty. Konstanta definovaná klíčovým slovem `const` je platná pouze v dané třídě a potomcích této třídy. Konstanta definovaná za pomocí funkce `define()` je dostupná v celé aplikaci. Tohoto chování lze vyžít např. pro definování URL adres serveru. Přehled předdefinovaných konstant lze najít v kapitole 6.2.

5.7 Responzivní design

S nástupem mobilních zařízeních umožňujících prohlížet webové stránky, je zapotřebí přizpůsobovat zobrazovaný obsah. Při zobrazení webové stránky na mobilním zařízení nepoužívající responzivní design obsah je širší než šíře zobrazovací plochy. Dochází tak ke skrytí části obsahu webové stránky ležící mimo zobrazovací plochu. Dalším z problémů může nastat na stránkách, které obsahují velké volné plochy. Při zobrazení na mobilním zařízení s malou zobrazovací plochou se zobrazí volná plocha. Nepozorný uživatel si nevšimne možnosti stránku posunout horizontálně a následně takovou stránku opustí.Uživatelé mobilních zařízeních jsou zvyklý posouvat obsah pouze ve vertikálním směru. Při možnosti posouvat obsah horizontálním směrem, může docházet ke snížení jednoduchosti ovládání aplikace.Zobrazovaný obsah se přizpůsobuje šířce zobrazovací plochy. Pokud prvky nelze zobrazit užší, přeskládají se z horizontální linie do linie vertikální. Při vývoji responzivního webu můžeme použít dva základní přístupy:

- **Desktop first** – Při vývoji webové stránky začíná vývoje od největšího možného rozlišení. Typicky největšího rozlišení dosahuje stránka při zobrazení na desktopu. Po vytvoření webové stránky pro zobrazení zaměřeném na desktop pokračuje vývoj stránky určené na menší rozlišení. Rozložení stránky zobrazující se na mobilních zařízeních se vyvíjí naposled. Problém může nastat při obsahu, který nelze zúžit nebo přeskládat z horizontální linie do linie vertikální. Typicky k tomuto chování dochází u složitých grafických návrhů, kde při zmenšení dochází k deformaci.

- **Mobile first** – Vývoj probíhá z opačného konce než u `desktop first`. Vývoj začíná u zobrazení pro mobilní zařízení. Prvky, které tvoří webovou stránku, je zapotřebí si více promyslet. Použitím této metody vzniká přehlednější kód.

Při vývoji aplikace byl uplatněn `desktop first` přístup. Aplikace neobsahuje složitou grafikou a nenastává tedy problém, jak grafiku zobrazit pro mobilní zařízení.

Pro usnadnění práce je vhodné použít CSS knihovnu, která zprostředkovává základní prvky. Mezi tyto prvky stojí za zmínku grid systém, konzistence CSS nebo media query. Pro snadnou definici rozložení stránky je vhodné využít grid systém. Grid systém dělí obsah do sloupců a řádků o proměnné šířce. Sloupce přizpůsobují svojí šířku podle šíře obrazovky. Při zobrazení stránky na menším zařízení se sloupce a řádky přeskupí. Grid systém je vhodné použít pro urychlení vývoje a zvýšení konzistence obsahu při zobrazení stránky na různých zařízeních. Ve výchozím stavu zobrazují různé prohlížeče elementy HTML různě. Pro zvýšení cross-browser konzistence je vhodné použitým elementu přidat výchozí styl.

5.7.1 W3.CSS

Projekt W3.CSS[4] si zakládá na rychlém a jednoduchém vývoji[17]. Komponenty frameworku W3.CSS jsou tvořeny pouze kaskádovými styly. Použitím pouze kaskádových stylů se sníží paměťové velikost a prostředky potřebné pro načtení frameworku. Framework neobsahuje žádné JavaScript skripty. Absence JavaScript skriptů přináší výhodu možnosti použít knihovnu v prostředí, kde je vypnutý nebo zcela chybí JavaScript. Pro zajištění konzistence výchozích stylů v různých prohlížečích je obsažena knihovna `Normalize.js`[5].

5.7.2 Bootstrap

Bootstrap[6] je balík nástrojů ucelených v jedné knihovně. Pro snadnou orientaci v prvcích je zpřístupně rozsáhlá dokumentace. Pro snadné rozmístění prvků na stránce je součástí knihovny grid systém. Součástí projektu jsou i komponenty pro vykreslení slideru, vyskakovací okna, ikony, tlačítka. S množstvím obsažených nástrojů a komponent roste i velikost a nároky potřebné pro načtení frameworku. Pro snížení velikosti a nároků lze knihovnu přizpůsobit k potřebám projektu.

Pro dosažení responzivního designu je v aplikaci využita knihovna Bootstrap. Knihovna Bootstrap byla zvolena z důvodu její rozšířenosti použití, rychlostí vývoje a jednoduchém použití knihovny. Další aspektem proč použít Bootstrap je rozsáhlá kolekce předefinovaných prvků použitelných v aplikaci. Obsahuje navíc i JavaScript prvky, které jsou nezbytné pro aplikaci. Mezi takové prvky se řadí prvek `collapse`.

5.8 Statistiky

Po dokončení jednotlivých dílčích testů se do databáze ukládají statistky o provedeném testu. Do databáze se ukládají informace o datu, počtu testovaných prvků a počtu prvků, které daným testem prošly. Pokud se již dané informace v databázi pro daný den vyskytují, dané informace se aktualizují. Testy jsou automatizovaně spouštěny minimálně jednou denně za pomocí služby Cron.

[4] Stránky projektu W3.CSS: https://www.w3schools.com/w3css/
[5] Normalize.js https://necolas.github.io/normalize.css/
[6] Stránky projektu Bootstrap: http://getbootstrap.com/

5.8.1 Grafy

Pro grafické znázornění statistiky byl zvolen spojnicový graf, z kterého je patrný průběh počtu chyb v čase. Pro generování grafů jsou dostupné dvě možnosti generování. Jednou z možností je generování grafů na straně serveru za pomocí PHP knihovny. Tahle možnost přináší výhodu v konzistenci grafů napříč všemi zařízeními a použitými prohlížeči. Kompatibilita je dosažena vygenerováním grafu jako obrázek a následném zobrazením v prohlížeči. Po vygenerování je možné obrázek uložit pro pozdější použití. Mezi nevýhody téhle možnosti je vyšší zátěž na straně serveru, omezené možnosti interaktivity grafu[18]. Ze zástupců knihoven, které generují grafy na straně severu je PHP knihovna pChart[7].

pChart

- Svobodná licence GNU GPLv3
- Dostupnost více typů grafů
- Závislost na dalších PHP knihovnách (GD library)

Možnost generovat grafy na straně serveru byla zavrhnuta z důvodu navýšení zátěže severu a nutnost instalovat další rozšíření PHP knihovny. Další z možností je generování grafů na straně klienta za pomocí JavaScript knihovny. Tahle možnost přináší snížení zátěže serveru při generování grafů. Výpočetní výkon, který by byl potřebný pro generování grafů, je spotřebováván na straně klienta. Jako nevýhodu bych uvedl možné problémy s konzistencí grafů na různých zařízeních. Výsledný vzhled může být ovlivněn použitým prohlížečem. Při použití JavaScript knihovny graf umožňuje interaktivnost s uživatelem. Je možné například skrýt některou z datových linií nebo při najetí na danou datovou linii zobrazit hodnotu v daném bodě. Jedna z knihoven, která je běžně používaná pro vykreslování grafů je knihovna Google Charts[8].

Google Charts

- vykreslení jako SVG obrázek
- Rozsáhlá množina typů grafů
- Graf není responzivní
- Umístění JS souborů na serverech Google
- Nesvobodná licence

Využití knihovny Google Charts byla zavrhnuta z důvodu nesvobodné licence a umístění JavaScript knihovny pouze na serverech Google. Pro vykreslení grafů byla zvolena JavaScript knihovna Chart.js [9].

[7] Stránky projektu pChart: http://www.pchart.net/
[8] Stránky projektu Google Charts: https://developers.google.com/chart/
[9] Stránky projektu Chart.js: http://www.chartjs.org/

Chart.js

- vykreslení na HTML5 prvek `canvas`
- Responzivní graf
- Open source licence (MIT licence)
- Omezené množství typů grafů (6 typů)

Knihovna Chart.js přináší potřebnou funkcionalitu, vhodnou licenci a jednoduchost implementace. Pro vykreslení grafu je do dané stránky vložen HTML prvek `canvas` a po načtení dané stránky je za pomocí JavaScriptu vyvolán požadavek pro přenos dat ze serveru. Data potřebná pro vykreslení grafu jsou přenášena asynchroním požadavkem ve formátu `JSON`. Po úspěšném přijmutím dat klientem dojde k vykreslení grafu na straně klienta. Při využití asynchroního načítání dat ze serveru dochází k zvýšení počtu požadavků, které musím daný server zpracovat.

Kapitola 6

Manuál k výsledné aplikaci

6.1 Konfigurační soubory

Pro snadnou konfiguraci aplikace při vývoji a následném nasazení na produkční server aplikace obsahuje dva druhy konfiguračních souborů plnící stejný účel. Na vývojovým serveru se využívají soubory pojmenované ve tvaru `jméno_souboru.local.json`, pro produkční server slouží konfigurační soubory pojmenované ve tvaru `jmeno_souboru.json`. Konfigurační soubory jsou uloženy ve strojově čitelném formátu `JSON`. `JSON` formát byl vybrán z důvodu vyšší rychlosti načtení a ukládání dat, nižší paměťové náročnosti a možnosti uložit nebo načíst objekt [16]. Konfigurační soubory jsou uloženy ve složce `app/config/`

Listing 6.1: Ukázka konfiguračního souboru ve formátu JSON.

```
{
"DB_SERVER": "localhost",
"DB_USER": "postgres",
"DB_PASSWORD": "password",
"DB_DATABASE": "database",
"DB_SCHEME": "hicheck",
"TIME_ZONE" : "Europe/Prague"
...
}
```

Pro snadnou manipulaci a zajištění načtení správné konfigurace odpovídající prostředí spouštění aplikace slouží třída `Configuration`. Při neexistenci konfiguračního souboru nebo nenačtení konfiguračního souboru se vyvolá výjimka typu `ConfigurarionException`.

Pro přístup k datovým položkám slouží metoda `get()`.Při neexistenci datové položky v konfiguračním souboru metoda vrací hodnotu `NULL`. Jednotlivé položky v konfiguračním souboru lze zanořovat, avšak přístup k zanořeným položkám za pomocí metody `get()` možný není. Pouze prvky na 1. úrovni jsou přístupné za pomocí metody `get()`.

6.2 Předdefinované konstanty

Pro snadnější ladění a testování aplikace je přístupná v celé aplikaci konstanta `LOCAL`.

- `LOCAL` - Konstanta obsahující informaci, zda aplikace běží na vývojovým nebo produkčním serveru.

Pro usnadnění nasazení nebo přesun aplikace na jiný server jsou předdefinovány konstanty obsahující absolutní cesty a URL adresy.

- `PAGE` - Konstanta obsahuje URL adresu, kde je aplikace spuštěna.
- `BASE` - Absolutní cesta root složky aplikace.

Umístění uživatelem nahraných soubor a exportů z testů lze ovlivnit následujícími konstantami.

- `DATA_PAGE` - URL adresa složky, ve které jsou umístěny nahrané soubory.
- `DATA_BASE` - Absolutní cesta složky, ve které jsou umístěny nahrané soubory.
- `EXPORT_PAGE` - URL adresa složky pro soubory exportované z testů.
- `EXPORT_BASE` - Absolutní cesta složky pro soubory exportované z testů.

Další konstanty jsou dostupné v deklaracích tříd, které jsou přístupné pouze v dané třídě nebo v potomcích této třídy. Jednou z takových konstant je konstanta `DISTANCE`, která je definována ve třídě `GuidepostsTest`. Konstanta určuje maximální vzdálenost rozcestníku od uzlu.

6.3 Přidání nové stránky

V aplikaci je uplatněna MVC architektura s jednosměrným statickým směrováním. Pro přidání jednoduché stránky je zapotřebí vytvořit nový controller, který dědí od abstraktní třídy `Controller`.

6.3.1 Controller

Všechny controllery jsou umístěny ve složce `app/controller/`. Vytvořený controller implementuje metodu `control(array $url)`, která je volána po vytvoření objektu a rozhoduje o dalších akcích aplikace. Z dříve uvedené metody jsou načítány další objekty (modely), které reprezentují datovou vrstvu aplikace. Pro předání dat z modelu do šablony slouží třídní proměnná `$data`, která je reprezentována datovým typem `array`. Pro definici jména šablony slouží třídní proměnná `$view`.

6.3.2 View

Šablony jsou umístěny ve složce `app/view/`. Pro vypsání dat, které jsou předány šabloně controllerem je zapotřebí šablonovací jazyk. Jazyk PHP již v sobě obsahuje šablonovací jazyk[20]. Obsah šablony je tvořen pouze nezbytnou logiku pro výpis obsahu. Logiku aplikace je vhodné přenést do modelu.

6.3.3 Router

Aby bylo možné zavolat daný controller, je zapotřebí přiřadit URL k danému controlleru. K tomuhle účelu slouží speciální controller `RouterController`. Parametry zadané URL adresy (`POST`, `GET`) je vhodné zpracovat až v přiřazeným controlleru.

6.4 Přidání nového testu

Soubory s jednotlivými testy jsou uloženy ve složce `/app/classes/tests`. Pro přidání testu je nutné implementovat abstraktní třídu `Test`. V této třídě jsou dostupné i další pomocné metody.

6.4.1 Longtime test

Longtime test se lišší od běžného testu svojí časovou náročností vykonávání. Typicky se takový test vykonává pouze 1x denně. Po vykonání testu jsou prvky uloženy do databáze. Pro uložení do databáze je přístupná třídní metoda `saveEntry`. Pro periodické spouštění longtime testů je zapotřebí přidat potřebné záznamy do controlleru `LongTestController`.

6.5 CRON

Pro provedení analýz jednotlivých testů je zapotřebí spouštět je periodicky v určitém časovém okamžiku. Při provádění analýz se uloží výsledek longtimetestů do databáze pro pozdější použití. Po vykonání jednotlivých analýz se provede vyhodnocení a uložení statistik do databáze. Pro konfiguraci daemona `CRON` je připraven inicializační skript. Skript `cron.sh` je umístěn ve složce `init`.

6.6 Uložení statistik

V kapitole 5.8 je obecný popis, jak lze statistiku graficky zobrazit za pomocí grafů. Grafy slouží pro vizuální reprezentaci statistik. V grafech lze velmi rychle dohledat přibližnou hodnotu nebo zkontrolovat průběh. Aby bylo možné grafy zobrazit, je zapotřebí data permanentně uchovat do databáze. Pro uložení dat slouží třídní metoda `saveStats` ve třídě `Tests`, kde parametry téhle metody jsou tvořeny z čísla testu, počtu testovaných prvků celkem a počtu prvků, které vyhověly danému testu. Statistiky se evidují pro každý test v určitý den. Aby nevznikaly duplicitní záznamy pro daný test v jeden den, je zřízen v databázi unikátní klíč na množině `test_id` a `date`. Při vkládání nového záznamu do databáze a výskytu konfliktu se záznam aktualizuje.

Kapitola 7

Závěr

Cílem této bakalářské práce je nastudovat systém mapování turistických tras v projektu OSM, zjistit možnosti automatické kontroly KČT v datech OSM, navrhnout postup pro automatizovanou kontrolu stavu tras v realitě a v datech OSM a realizovat aplikaci umožňující kontrolu tras KČT.

Pro začátek práce na této bakalářské práci pro mne znamenal obeznámit se s typy elektronických map a způsoby jejich tvorby. Pro pochopení proč je potřebné automaticky kontrolovat OSM data, jsem se seznámil jakým způsobem jsou mapy OpenStreetMap tvořeny. Po seznámení se s obecnými informacemi o projektu OSM, jsem analyzoval způsob ukládání a distribuce OSM dat. Hlavní pozornost při studii OSM dat byla zaměřena na jednotlivé prvky tvořící datovou strukturu. Při přidávání nebo upravování prvků uživatelem mohou být porušeny pravidla a konvence projektu OSM. Dodržování pravidel lze kontrolovat ručně, kde porušení pravidel detekuje uživatel a automatická kde porušení pravidel odhalí vytvořená aplikace. V této technické zprávě se zabývám možnostmi automatické kontroly OSM dat, mezi které patří i kontroly turistických tras. Po studii principů OSM projektu se má pozornost zaměřovala na aplikace Osmose a OsmHiCheck, které implementují (polo)automatické kontroly OSM dat. Tato aplikace vychází z projektu OsmHiCheck. Aplikace z projektu OsmHiCheck přebírá zkušenosti a způsob aktualizace OSM dat.

Automatické kontroly byli implementovány jako webová aplikace skládající se ze 3 základních částí. Hlavní částí aplikace je mapa s vrstvami zobrazující jednotlivé chyby. Pro výpis všech kontrol slouží tabulková část aplikace. V této části lze zobrazit i detail kontroly, který vypíše všechny prvky neodpovídající dané kontrole. Poslední částí jsou statistiky. Statistiky slouží pro rychlý přehled průběhu kontrol v čase. Pro zobrazení statistiky byl zvolen grafický prvek liniový graf. Vytvořená aplikace splňuje základní stanovené cíle. Mezi tyto cíle patří modularita, jednoduchost, rychlost vývoje a bezpečnost. V aplikaci se nepodařilo dosáhnout poloautomatické kontroly dat.

Po provedení jednotlivých analýz prvků a ruční kontrole uživatelem mohou nastat dvě možnosti. Jednou z možností které mohou nastat je, že nevyhovující prvek je uveden správně. Takový prvek je třeba označit jako správný a nadále ho již neanalyzovat. Druhá situace může být opačná. Analyzovaný prvek obsahuje chybu, která je zapotřebí opravit. Rozšíření aplikace by umožňovalo základní úpravy prvků. Provedené úpravy by byly provedeny i v OSM datech a provedené změny by byly synchronizovány se servery projektu OpenStreetMap. Při úpravě dat je zapotřebí autentizovat daného uživatele. Vhodným mechanismem pro autentizaci uživatele je využít autentizačního systému projektu OpenStreetMap a technologie OAuth[12].

Dalším možným pokračováním ve vývoji této aplikace je přidání dalších analýz. Při přidávání nových analýz se mohou vyskytnout nové skutečnosti, které bude v aplikaci potřeba změnit. Touto skutečností může být např. obsah databáze. V databázi jsou uloženy pouze data turistických tras. Dalším směrem, kterým se vývoj aplikace může ubírat, je analýza zvyklostí jiných zemí. Při podpoře zvyklostí jiných zemí bude potřebné zavést podporu více jazykových mutací rozhraní.

Literatura

[1] Bernard, B.: *Úvod do architektury MVC*. Květen 2009, [Online; navštíveno 18.03.2017].
URL https://www.zdrojak.cz/clanky/uvod-do-architektury-mvc/

[2] From Wikipedia, the free encyclopedia: *Exception handling*. [Online; navštíveno 16.03.2017].
URL https://en.wikipedia.org/wiki/Exception_handling

[3] Lockhart, J.; Sturgeon, P.: *Language Highlights*. [Online; navštíveno 15.03.2017].
URL http://www.phptherightway.com/#programming_paradigms

[4] NikiC: *define() vs const*. 2015, [Online; navštíveno 16.04.2017].
URL http://stackoverflow.com/questions/2447791/define-vs-const

[5] OpenStreetMap Wiki contributors: *OsmChange*. 2016, [Online; navštíveno 17.04.2017].
URL http://wiki.openstreetmap.org/wiki/Tags

[6] OpenStreetMap Wiki contributors: *Relation*. 2016, [Online; navštíveno 17.04.2017].
URL http://wiki.openstreetmap.org/wiki/Relation

[7] OpenStreetMap Wiki contributors: *Way*. 2016, [Online; navštíveno 17.04.2017].
URL http://wiki.openstreetmap.org/wiki/way

[8] OpenStreetMap Wiki contributors: *Cs:Key:abandoned:*. 2017, [Online; navštíveno 11.05.2017].
URL http://wiki.openstreetmap.org/w/index.php?title=Cs:Key:abandoned:&oldid=1463020

[9] OpenStreetMap Wiki contributors: *Cs:Key:operator*. 2017, [Online; navštíveno 10.05.2017].
URL http://wiki.openstreetmap.org/w/index.php?title=Cs:Key:operator&oldid=1458813

[10] OpenStreetMap Wiki contributors: *Cs:PhotoDB*. 2017, [Online; navštíveno 22.04.2017].
URL http://wiki.openstreetmap.org/w/index.php?title=Cs:PhotoDB&oldid=1441525

[11] OpenStreetMap Wiki contributors: *Cs:WikiProjekt Česko/OTM značkový klíč*. Květen 2017, [Online; navštíveno 11.05.2017].
URL http://wiki.openstreetmap.org/w/index.php?title=Cs:WikiProjekt_%C4%8Cesko/OTM_zna%C4%8Dkov%C3%BD_kl%C3%AD%C4%8D&oldid=1452611

[12] OpenStreetMap Wiki contributors: *OAuth*. 2017, [Online; navštíveno 22.04.2017].
URL http://wiki.openstreetmap.org/w/index.php?title=OAuth&oldid=1429477

[13] OpenStreetMap Wiki contributors: *Stats*. 2017, [Online; navštíveno 19.04.2017].
URL http://wiki.openstreetmap.org/w/index.php?title=Stats&oldid=1446588

[14] OpenStreetMap Wiki contributors: *Tags*. 2017, [Online; navštíveno 17.04.2017].
URL http://wiki.openstreetmap.org/wiki/Tags

[15] OpenStreetMap Wiki contributors: *Way*. 2017, [Online; navštíveno 17.04.2017].
URL http://wiki.openstreetmap.org/wiki/node

[16] Refsnes Data: *JSON vs XML*. [Online; navštíveno 10.04.2017].
URL https://www.w3schools.com/js/js_json_xml.asp

[17] Refsnes Data: *What is W3.CSS?* [Online; navštíveno 24.04.2017].
URL https://www.w3schools.com/w3css/default.asp

[18] Singhal, V.: *4 Best Chart Generation Options with PHP Components*. 2015, [Online; navštíveno 15.04.2017].
URL https://www.sitepoint.com/4-best-chart-generation-options-php-components/

[19] The Apache Software Foundation: *Access Control*. [Online; navštíveno 16.04.2017].
URL https://httpd.apache.org/docs/2.2/howto/access.html

[20] The PHP Group: *Alternative syntax for control structures*. [Online; navštíveno 29.04.2017].
URL http://php.net/manual/en/control-structures.alternative-syntax.php

[21] The PHP Group: *Class Constants*. [Online; navštíveno 16.04.2017].
URL http://php.net/manual/en/language.oop5.constants.php

[22] The PHP Group: *define*. [Online; navštíveno 16.04.2017].
URL http://php.net/manual/en/function.define.php

[23] The PHP Group: *ini_set*. [Online; navštíveno 16.03.2017].
URL http://php.net/manual/en/function.ini-set.php

[24] The PHP Group: *Predefined Constants*. [Online; navštíveno 16.03.2017].
URL http://php.net/manual/en/errorfunc.constants.php

[25] Švaňa, P.: *Kontrolní aplikace pro trasy KČT v OSM, bakalářská práce*. FIT VUT v Brně, 2015.

Přílohy

Příloha A

Obsah CD

Adresářová struktura CD:

- `/src/` – zdrojové soubory aplikace
- `/src/init/` – zdrojové soubory pro import a nastavení
- `/thesis/BP_xlevpa00.pdf` – text technické zprávy ve formátu PDF
- `/git/` – GIT repositář s kompletní historií vývoje