VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

WEBOVÝ EDITOR GPX DAT

WEB EDITOR FOR GPX DATA

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **PETR HENDRYCH**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. RADEK BURGET, Ph.D.**
SUPERVISOR

BRNO 2020

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav informačních systémů (UIFS) Akademický rok 2019/2020

Zadání bakalářské práce

Student: **Hendrych Petr**
Program: Informační technologie
Název: **Webový editor GPX dat**
Web Editor for GPX Data
Kategorie: Databáze

Zadání:

1. Seznamte se s datovými formáty pro reprezentaci geografických dat se zaměřením na formát GPX.
2. Prostudujte existující technologie pro tvorbu webových aplikací s tlustým klientem v jazyce JavaScript.
3. Navrhněte aplikaci pro interaktivní editaci dat GPX s možností jak interaktivní editace na mapě, tak i ruční editace dat (bodů, segmentů, tras, apod.)
4. Implementujte navrženou aplikaci v jazyce JavaScript s využitím vhodných podpůrných technologií.
5. Proveďte testování vytvořeného řešení.
6. Zhodnoťte dosažené výsledky.

Literatura:

- Gutmans, A., Rethans, D., Bakken, S.: Mistrovství v PHP 5, Computer Press, 2012
- GPX 1.1 Schema Documentation, http://www.topografix.com/GPX/1/1/

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 až 3

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Burget Radek, Ing., Ph.D.**
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2019
Datum odevzdání: 28. května 2020
Datum schválení: 16. října 2019

Abstrakt

Cílem této bakalářské práce je vytvoření webového editoru, prostřednictvím kterého bude uživatel moct zobrazit a upravit své zaznamenané trasy ve formátu GPX. U každé trasy může uživatel změnit pozici vybraného bodu, přidat další nebo smazat jednotlivý bod či vybranou skupinu. Klientská strana aplikace je napsána v jazyce Javascript za využití knihovny React. Serverová část je implementována pomocí frameworku Django a modulu GeoDjango. Komunikace mezi jednotlivými částmi zajišťuje Django REST framework. Výsledky této práce umožňují uživatelům jednoduše upravovat své trasy ve webovém prohlížeči.

Abstract

The aim of this bachelor thesis is to create a web editor, which will allow the user to view and edit their recorded routes in GPX format. For each route, the user can adjust the position of the selected point, add another or delete an individual point or a selected group. The client part of the application is written in Javascript using the React library. The server part is implemented using the Django framework and the GeoDjango module. Communication between these parts is provided by the Django REST framework. The results of this work allow users to easily edit their tracks in a web browser.

Klíčová slova

Javascript, React, Redux, SCSS, CSS, Bootstrap, Python, Django, GeoDjango, Django REST Framework, PostgreSQL, PostGIS, Leaflet, OpenStreetMap

Keywords

Javascript, React, Redux, SCSS, CSS, Bootstrap, Python, Django, GeoDjango, Django REST Framework, PostgreSQL, PostGIS, Leaflet, OpenStreetMap

Citace

HENDRYCH, Petr. *Webový editor GPX dat.* Brno, 2020. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Radek Burget, Ph.D.

Webový editor GPX dat

Prohlášení

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

.....
Petr Hendrych
27. května 2020

Poděkování

Rád bych poděkoval mému vedoucímu práce panu Ing. Radku Burgetovi, Ph.D. za odborné vedení a věcné připomínky.

Obsah

Kapitola 1

Úvod

Zaznamenávání tras pomocí GPS[1] přístrojů začíná být s nynějším vzestupem užívání chytrých zařízení, jako jsou například hodinky nebo náramky, stále běžnější záležitostí. Tyto trasy je možno poté exportovat do různých datových formátů, které slouží pro přenos mezi jednotlivými zařízeními. GPS zařízení fungují na základě přijímaní signálů vysílaných z družic, které obíhají kolem země. Pokud se se zařízením dostaneme do míst, kde je špatný signál, tak vznikají různé odchylky, které mohou ve výsledné trase působit rušivým dojmem.

Pro účely zobrazení tras existuje řada nástrojů. Ať už se jedná o desktopové aplikace nebo online webové. Většina z nich umožňuje i editaci tras, avšak u žádné jsem nenalezl možnost výběru bodů ve zvolené oblasti na mapě.

Cílem této bakalářské práce je vytvořit webovou aplikaci pro snadnou úpravu geografických dat a to se zaměřením na datový formát GPX[2]. Editor by měl poskytnout možnost jak interaktivní editace trasy přímo na mapovém podkladu, tak i ruční editaci dat. Jedná se například o úpravu polohy jednotlivých bodů trasy, jejich přidávání nebo odebírání či selekce vybraného úseku trasy.

Tato práce je rozdělena do jednotlivých kapitol a podkapitol. V kapitole 2 se přiblíží obecně GPS, souřadnicové systémy a datové formáty, které se v této oblasti využívají. Zvláštní pozornost se zde věnuje formátu GPX, se kterým tato aplikace pracuje. Následující kapitola 3 pojednává o již existujících aplikacích, které řeší tuto problematiku. Lze zde najít zástupce desktopových i webových aplikací. Kapitola 4 přibližuje nynější moderní technologie pro tvorbu webových aplikací. Návrh aplikace a použité technologie lze najít v kapitole 5. Předposlední kapitola 6 je rozdělena do dvou částí a popisuje samotnou implementaci aplikace. Závěrečné shrnutí práce se nachází v kapitole 7.

[1] Global positioning systém - Globální polohový systém
[2] GPS Exchange Format

Kapitola 2

GPS a datové formáty geografických dat

GPS[1] je systém, díky němuž zařízení mohou lokalizovat svou polohu kdekoli na Zemi. Jedná se o celosvětový radionavigační systém skládající se nyní ze 31 satelitů a jejich pozemních stanic. Zpočátku byl systém GPS určen pro vojenské účely, což se změnilo v roce 2000, kdy byl systém zpřístupněn veřejnosti. [9]

GPS poskytuje speciální satelitní signály, které přijímací zařízení zpracovávají. Takové přijímače sledují přesnou polohu s nadmořskou výškou a také mohou vypočítat rychlost a čas. Pro stanovení polohy zařízení je nutné, aby byly získány signály alespoň ze čtyř satelitů a jejich vzdálenost od zařízení. Díky tomu mohou přijímače určit zeměpisnou šířku, zeměpisnou délku, nadmořskou výšku a čas. Přijímače tyto signály registrují v časových intervalech, čímž vzniká výsledná trasa. [4]

2.1 Souřadnicové systémy

Souřadnicový systém je schopen vyjádřit polohu každého bodu území, na němž je definován nanejvýše třemi číselnými údaji [22]. Ve 2D prostoru se jedná o zeměpisnou šířku a zeměpisnou délku. Při práci v 3D prostoru se přidá třetí hodnota, a to nadmořská výška.

Zeměpisné souřadnice (šířku a délku) je nutno vztahovat vůči určitému referenčnímu tělesu. Nejčastěji se využívá elipsoid. Dalšími parametry, které určují daný souřadnicový systém, jsou: poloha nultého poledníku a jednotky úhlové míry (nejčastěji stupeň). [22]

2.1.1 WGS84

V současnosti se mezi nejvíce využívané souřadnicové systémy zeměpisných souřadnic řadí WGS84[2]. Jde o globální systém, jenž pro své účely používá například NATO, a GPS využívá právě tento systém. [22]

Poloha pro tento systém vychází ze zeměpisných souřadnic a výšky. Zeměpisná délka a šířka se udávají ve stupních a minutách. Šířka nabývá hodnot $0° - 90°$ na sever a jih od rovníku. Délka nabývá $0° - 180°$ na západ a východ od nultého poledníku. Výška se udává v metrech nad referenčním elipsoidem. Jedná se o pravotočivou soustavu souřadnic.

[1] Global Positioning System
[2] World Geodetic System 1984

V online mapách se nejčastěji používá systém rovinných souřadnic jako Web Mercator. Při něm se jako referenční těleso bere koule se stejným poloměrem jako hlavní poloosa elipsoidu, který využívá WSG84. Tento fakt přináší drastické zkreslení v oblastech kolem pólů. Avšak toto zkreslení lze v online mapách kompenzovat proměnlivým měřítkem. [22]

2.1.2 ETRS89

Základním systémem pro Evropu je ETRS89[3]. Vychází z globálního elipsoidu GRS80, který je svými parametry velice blízký elipsoidu, jenž využívá systém WSG84. V tomto systému se Euroasijská deska považuje za statickou. Stejně jako WSG84 používá zeměpisné i pravoúhlé souřadnice. Nad tímto systémem je vytvořena řada národních systémů. [22]

2.1.3 S-JTSK

V České republice je pro civilní účely používán systém S-JTSK[4]. Systém je sestaven použitím Křovákova zobrazení (dvojité kuželové zobrazení) nad Besselovým elipsoidem. Cílem tohoto zobrazení bylo zavést takový pravoúhlý souřadnicový systém, který by zahrnoval celé tehdy Československo v prvním kvadrantu, a tudíž by byly obě souřadnice kladné. [22]

2.2 Datové formáty

V následující sekci přiblížím některé datové formáty, které se používají pro zaznamenání a především přenos mezi různými zařízeními.

2.2.1 KML

KML [14] je formát souboru používaný k zobrazení geografických dat v aplikacích typu Google Earth. Jedná se o značkovací jazyk založený na standardu XML[5]. Všechny tagy rozlišují velké a malé písmena a musí se vyskytnout přesně jak je uvedeno v KML referenci.

Po XML hlavičce je uvedena KML deklarace výchozího jmenného prostoru s příslušnou URI adresou. V ukázce 2.1 lze vidět jednoduchý příklad KML souboru, který obsahuje jeden `<Placemark>` element, jenž obsahuje jeden bod specifikující jeho polohu.

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2">
  <Document>
        <Placemark>
          <name>New York City</name>
          <description>New York City</description>
          <Point>
                <coordinates>-74.006393,40.714172,0</coordinates>
          </Point>
        </Placemark>
  </Document>
</kml>
```

Výpis 2.1: Ukázka jednoduchého KML souboru obsahujícího jeden bod.

[3] European Terrestrial Reference System 1989
[4] Systém jednotné trigonometrické sítě katastrální
[5] eXtensible Markup Language

2.2.2 GPX

Formát GPX [12] je jednoduchý datový formát taktéž založený na jazyku XML. Slouží pro výměnu GPS dat mezi aplikacemi a webovými službami. GPX definuje společnou sadu datových značek pro popis geografických dat v XML.

Jak uvádí dokumentace [11], kořenem této stromové struktury je element `<gpx>`. Tento element má dva povinné atributy, kde prvním je `version`, který udává verzi schématu, jenž daný soubor používá, kde aktuální verze je 1.1. Druhý atribut je `creator`, což je název aplikace, zařízení nebo webové služby, která daný soubor vytvořila. Dále se zde vyskytuje deklarace implicitního jmenného prostoru, díky kterému se zabrání vzniknutí konfliktů při spojení různých formátů, které obsahují elementy se stejným názvem.

Následuje element `<metadata>`. Ten uchovává doplňující informace o tomto souboru jako je například autor, čas vytvoření nebo klíčová slova. Tato část je volitelná a nemusí se vyskytovat v každém souboru.

Poté se může vyskytnout jeden ze tří typů: `<wpt>` (*waypoint*), `<rte>` (*route*) nebo `<trk>` (*track*). Jediný soubor může obsahovat více typů, které nemusí být stejného druhu. Tato práce se zaměřuje pouze na `<trk>`, jehož strukturu nyní přiblížím.

Tento typ reprezentuje trasu - seřazené pole bodů, které popisují cestu. Trasa se skládá ze segmentů - `<trkseg>`, kde každá neprázdná trasa musí obsahovat minimálně jeden segment.

Segmenty obsahují seznam bodů - `<trkpt>`, které jsou logicky seřazeny. Pokud GPS přijímač ztratí signál nebo se zařízení vypne, tak se segment ukončí. Po znovu nalezení signálu se začne vytvářet nový segment. V rámci této práce bylo po dohodnutím s vedoucím práce panem Burgetem rozhodnuto, že pokud trasa obsahuje více segmentů, tak se tyto segmenty spojí do jednoho. Jelikož pokud nastane skutečnost, že se v trase vytvoří více segmentů, tak to většinou bývá tak, že tyto segmenty se nacházejí blízko sebe a tudíž jejich spojení nijak výrazně nenarušuje strukturu zaznamenané trasy.

Nejdůležitější částí jsou jednotlivé zaznamenané body – `<trkpt>`. Tyto elementy obsahují dva povinné atributy. Jedná se o `lat`, což je souřadnice zeměpisné šířky a `lon`, která udává hodnotu zeměpisné délky. Jak bylo zmíněno v sekci 2.1, tak tyto hodnoty se udávají podle souřadnicového systému WSG84 a většinou jsou omezeny na šest desetinných míst, protože větší přesnost už na mapových podkladech nehraje roli. Dále může tento element obsahovat další dva volitelné elementy, které přidávají další informace ohledně daného bodu. Jejich pořadí je pevně stanovené schématem.

Prvním je `<ele>`, který obsahuje nadmořskou výšku udávanou v metrech. Ne všechny GPS zařízení tuto možnost poskytují, proto je tento element volitelný.

Druhým elementem, který poskytuje doplňující informace ohledně bodu je `<time>`. Jak vychází z názvu, tak tento element uchovává informaci, kdy byl daný bod zaznamenán. Čas je vztahován k UTC[6] a reprezentován ve formátu ISO 8601[7].

Příklad GPX souboru lze vidět na následující ukázce 2.2.

[6] Coordinated Universal Time

[7] https://cs.wikipedia.org/wiki/ISO_8601

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<gpx
  xmlns="http://www.topografix.com/GPX/1/1"
  xmlns:gpxx="http://www.garmin.com/xmlschemas/GpxExtensions/v3"
  xmlns:gpxtpx="http://www.garmin.com/xmlschemas/TrackPointExtension/v1"
  creator="Oregon 400t" version="1.1"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.topografix.com/GPX/1/1
  http://www.topografix.com/GPX/1/1/gpx.xsd
  http://www.garmin.com/xmlschemas/GpxExtensions/v3
  http://www.garmin.com/xmlschemas/GpxExtensionsv3.xsd
  http://www.garmin.com/xmlschemas/TrackPointExtension/v1
  http://www.garmin.com/xmlschemas/TrackPointExtensionv1.xsd"
>
  <metadata>
    <link href="http://www.garmin.com">
      <text>Garmin International</text>
    </link>
    <time>2009-10-17T22:58:43Z</time>
  </metadata>
  <trk>
    <name>Example GPX Document</name>
    <trkseg>
      <trkpt lat="47.644548" lon="-122.326897">
        <ele>4.46</ele>
        <time>2009-10-17T18:37:26Z</time>
      </trkpt>
      <trkpt lat="47.644548" lon="-122.326897">
        <ele>4.94</ele>
        <time>2009-10-17T18:37:31Z</time>
      </trkpt>
      <trkpt lat="47.644548" lon="-122.326897">
        <ele>6.87</ele>
        <time>2009-10-17T18:37:34Z</time>
      </trkpt>
    </trkseg>
  </trk>
</gpx>
```

Výpis 2.2: Ukázka obsahu volně dostupného GPX souboru obsahující jednu krátkou trasu.

2.2.3 GeoJSON

Dalším formátem, který se využívá pro přenos geografických dat je GeoJSON. Jak již vyplývá z názvu, tak tento formát využívá struktury JSON[8]. Právě tato vlastnost je velkou výhodou, jelikož formát JSON podporuje velké množství programovacích jazyků, tak se stává ideálním formátem pro výměnu geografických dat.

GeoJSON definuje několik typů JSON objektů a způsob jakým jsou kombinovány k reprezentaci geografických objektů, jejich vlastnosti a prostorový rozsah [5].

Jak uvádí dokumentace [5], GeoJSON objekt může reprezentovat prostorovou oblast (`Geometry`), prostorově ohraničenou entitu (`Feature`) či seznam entit (`FeatureCollection`). GeoJSON podporuje následující typy geometrie: `Point`, `LineString`, `Polygon`, `MultiPoint`, `MultiLineString`, `MultiPolygon` a `GeometryCollection`. Každý objekt může obsahovat další doplňující informace v objeku `properties`.

Příklad GeoJSON souboru můžeme vidět na ukázce 2.3.

```
{
   "type": "FeatureCollection",
   "features": [{
       "type": "Feature",
       "geometry": {
           "type": "Point",
           "coordinates": [102.0, 0.5]
       },
       "properties": {
           "prop0": "value0"
       }
   }, {
       "type": "Feature",
       "geometry": {
           "type": "LineString",
           "coordinates": [
               [102.0, 0.0],
               [103.0, 1.0],
               [104.0, 0.0],
               [105.0, 1.0]
           ]
       },
       "properties": {
           "prop0": "value0",
           "prop1": 0.0
       }
   }]
}
```

Výpis 2.3: Ukázka jednoduchého GeoJSON objektu.

[8] JavaScript Object Notation

Kapitola 3

Existující služby pro úpravu tras

S rozšířením chytrých zařízení, jenž dokážou zaznamenávat zeměpisnou polohu, a tudíž vytvářet záznamy různých aktivit, se vyskytla také potřeba dané záznamy zobrazit (většinou na mapových podkladech) nebo přenášet mezi dalšími zařízeními. Služby, které toto poskytují, můžeme rozdělit do dvou kategorií a to desktopové aplikace, díky nímž je možnost upravovat trasy v offline režimu, a online webové aplikace, které přinášejí další možnosti jako je například sdílení na různých sociálních sítích.

3.1 Desktopové aplikace

I když zadání poukazovalo na tvorbu webové aplikace, tak nelze opomenout i tuto možnost úpravy dat. Jak již bylo zmíněno, hlavní výhodou desktopových aplikací je možnost práce offline. Jako zástupce jsem vybral GPS Track Editor[1].

GPS Track Editor

Jedná se o velice komplexní editor pouze pro operační systém Windows, který mimo základního zobrazení tras na různých mapových podkladech, ze kterých si může uživatel vybrat, také přináší širokou škálu editačních možností.

Řadí se mezi ně:

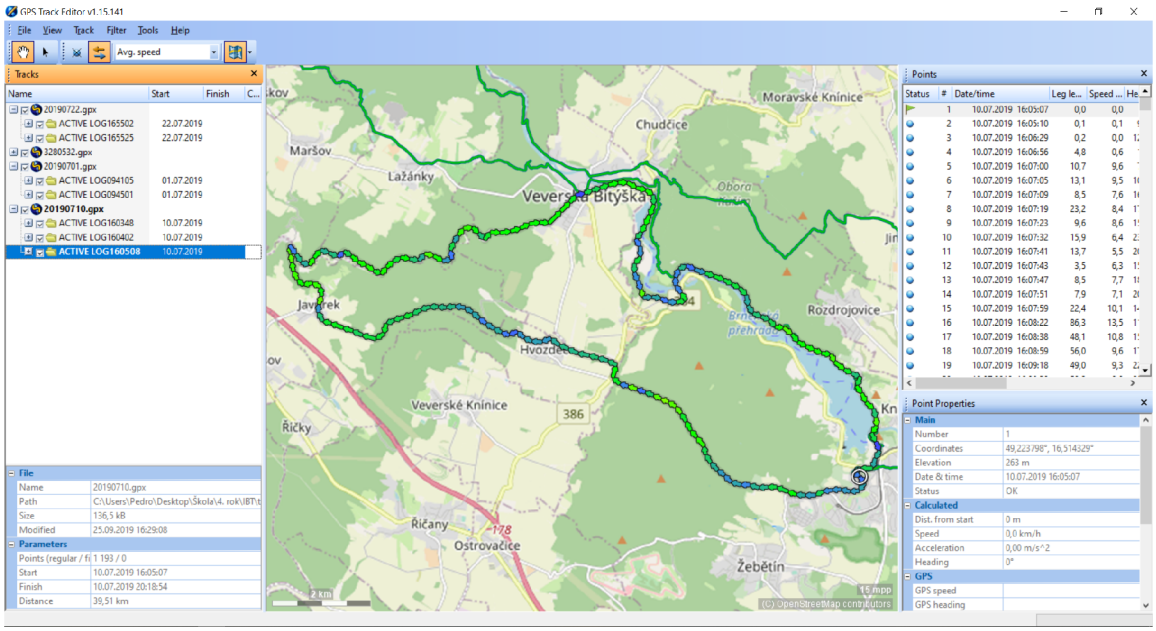
- Úprava polohy bodu na mapě
- Přidání a smazání bodu
- Spojení více tras do jedné
- Filtrace bodů dle zadaných parametrů
- Automatická oprava odchylek

Jak můžeme vidět na obrázku 3.1 aplikace je rozdělena do tří hlavních sekcí. Vlevo se nachází postranní panel, který obsahuje informace ohledně nahraných souborů a jejich trasy. Dále zde nalezneme obecné informace o souboru a parametry zvolené trasy.

V pravém postranním panelu se nachází seznam jednotlivých bodů. Pod tímto segmentem se uživatel může dozvědět další doplňující informace ohledně zvoleného bodu.

[1] http://www.gpstrackeditor.com/

Prostřední část je vyhrazena pro mapu. Na mapě můžeme zobrazit více tras najednou a je na výběr z různých režimů zobrazení tras. Například zobrazení šipek pro vyznačení směru zaznamenání nebo barevné rozlišení podle dalších parametrů.



Obrázek 3.1: Ukázka z aplikace GPS Track Editor

3.2 Webové aplikace

Pro potřeby editace tras lze na internetu najít celá řada aplikací. Často můžeme nalézt webové stránky, které jsou přímo spojené s mobilními aplikacemi, jenž tyto data zaznamenávají. Jedná se například aplikace Sport Tracker[2] nebo Strava[3]. Tyto aplikace většinou slouží k analýze sportovních aktivit, uchovávání statistik či sdílení na sociálních sítích.

GPX Editor

Prvním zástupcem, kterého jsem vybral je GPX Editor[4]. Editor je optimalizovaný především záznamy typu `track` a soubory ve formátu GPX. Aplikaci je možné využívat zdarma s velmi omezenými možnostmi nebo se stát odběratelem a za poplatek získat přístup ke všem editačním prostředkům. Náhled aplikace lze vidět na obrázku 3.2.

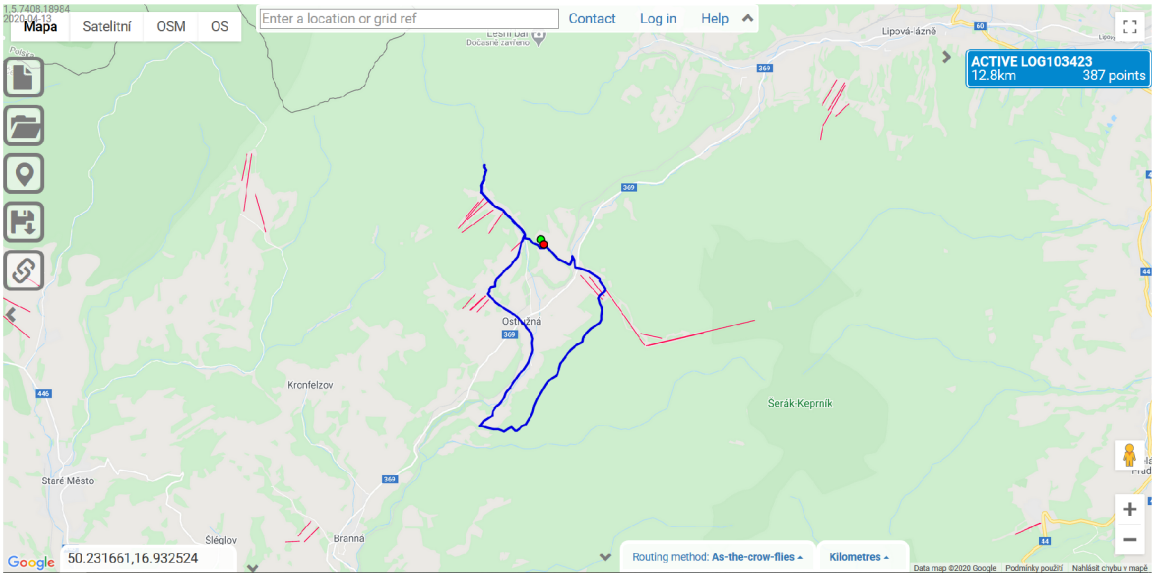
Verze zdarma

Verze zdarma umožňuje pouze zobrazení jedné trasy nebo vytvoření zcela nové trasy. Dále je ještě dostupné stažení trasy, a to ve formátu GPX. Aplikace také využívá různých Google doplňků jako je například vyhledávač či Street View[5].

[2] https://www.sports-tracker.com/
[3] https://www.strava.com/
[4] https://www.gpxeditor.co.uk/map
[5] https://www.google.com/intl/cs_cz/streetview/

Placená verze

Při placení ceny dvaceti liber ročně se nám otevře plná verze aplikace a s tím i možnost editace tras. Přidávají se akce jako například úprava bodů na mapě, rozdělování trasy na menší segmenty nebo ukládání do jiných formátů jako je TCX apod.



Obrázek 3.2: Ukázka z online aplikace GPX Editor

WTracks

Druhým zástupcem, který stojí v této oblasti za zmínku, je WTracks[6]. Oproti dříve popsanému GPX Editoru se jedná o open-source projekt a přináší větší možnost editace tras bez placení jakýchkoli poplatků.

Mimo základní širokou škálu editačních funkcí aplikace přináší podporu nahrání vlastních API klíčů od jiných služeb, které základní funkcionalitu rozšiřují. Taktéž aplikace podporuje trasy, které obsahují více segmentů a nabízí po nahrání souboru jejich spojení.

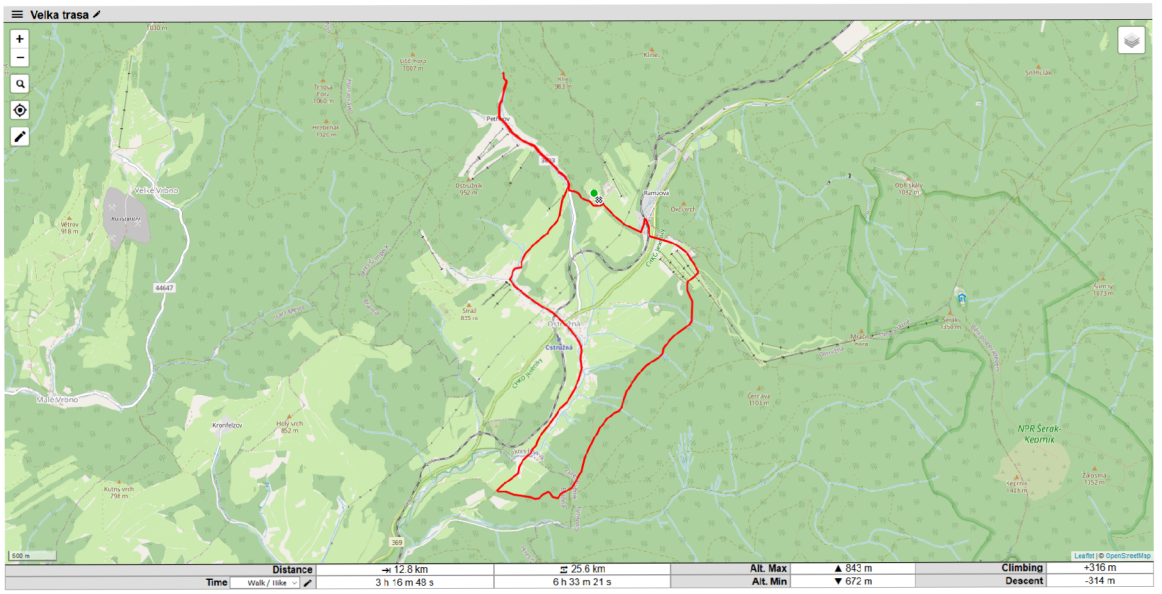
Ve spodním panelu přináší WTracks přehledné informace ohledně nahrané trasy, kdy se po kliknutí na tento panel zobrazí výškový profil nahrané trasy. Dále, mimo základní informace jako je délka trasy, maximální a minimální nadmořskou výšku, přináší také volbu vypočtení délky trvání trasy, a to v závislosti na vybrané možnosti. Řadí se zde chůze, běh, jízda na kole nebo plavání.

Úprava bodů zde probíhá pouze interaktivně přímo na mapě. Po přepnutí do editačního režimu se zde zobrazí každý bod dané trasy a pokud trasa obsahuje velké množství bodů, tak lze pocítit ztráta ve výkonu aplikace. Přidání bodu zde probíhá po výběru značky mezi dvěma již existujícími body trasy. Mimo tyto editační funkce aplikace také podporuje vytvoření zcela vlastní trasy.

[6] https://opoto.github.io/wtracks/

Dále aplikace přináší do jisté míry automatizovanou editaci trasy v závislosti na parametrech zadaných uživatelem. Jedná se o kompresi trasy, kdy budou odstraněny body, které se nacházejí v určité vzdálenosti od sebe, aniž by došlo ke ztrátě tvaru trasy. Další možností je odstranění určitého počtu bodů od začátku či konce trasy.

Ukázku aplikace lze vidět níže na obrázku 3.3



Obrázek 3.3: Ukázka z aplikace WTracks

Kapitola 4

Technologie pro tvorbu webových aplikací

V této kapitole jsou popsány různé technologie, které slouží k vytvoření webových aplikací. Na výběr existuje nepřeberné množství technologií a tato kapitola přiblíží jednotlivé části, které vedou k tvorbě webové aplikace. Kapitola je rozdělena do tří hlavních částí – klientská část, serverová část a databázová vrstva.

4.1 Klientská strana

Jedná se o tvorbu uživatelského rozhraní. K této práci lze využít klasických technologií jako jsou HTML[1] a CSS[2], avšak v zadání práce bylo uvedeno využití technologií v jazyce JavaScript. Vybral jsem tudíž nejznámější frameworky a knihovny k tomuto určené, které dále více přiblížím.

4.1.1 Angular

Původně AngularJS je populární webový framework sloužící k tvorbě single-page aplikací. Za vývojem tohoto frameworku je postavena společnost Google a první oficiální vydání se datuje k říjnu 2010 [6]. V roce 2016 byla vydána nová verze – Angular 2, kde vypadlo z názvu spojení JS. Tato verze a její následující se označují pouze prostým Angular.

Pro práci s Angularem se používá jazyk TypeScript. Jedná se o nadstavbu jazyka JavaScript, který jej rozšiřuje o statické typování, moduly a další. Jelikož webové prohlížeče nedokážou zpracovat TypeScript, tak musí dojít ke kompilaci do čistého JavaScriptu. Díky tomu je jakýkoli kód napsaný v JavaScriptu plně funkční i v jazyce TypeScript.

Samotný Angular má implicitně zabudované velké množství funkcí jako je například směrování. Dále je možnost využít Angular CLI[3]. Tento nástroj přináší řadu užitečných příkazů, které usnadňují práci s frameworkem.

[1] Hypertext Markup Language
[2] Cascading Style Sheets
[3] Command-line interface

Řadí se mezi ně například [13]:

- `ng new [název]` - inicializace nové aplikace
- `ng generate component [název]` - vytvoření nové komponenty
- `ng serve` - spuštění development serveru

Angular je postaven na takzvaných komponentách. Pokud se využije k vytvoření nové komponenty CLI, tak to nám automaticky vygeneruje novou složku s názvem komponenty, kde se nachází základní soubory pro danou komponentu. Jedná se o HTML soubor, který slouží jako šablona, CSS soubor pro styly, TypeScript soubor, který se stará o logiku komponentu.

4.1.2 React

Další velmi populární možností na tvorbu webových aplikací je React. Na rozdíl od Angularu se jedná o JavaScriptovou knihovnu. Originálně knihovnu napsal Jordan Walke a nyní za vývojem stojí společnost Facebook. První oficiální využití proběhlo v roce 2013 speciálně pro produkty od Facebooku [6].

Komponenty

Komponenty jsou nedílnou součástí každého ze zmíněných frameworků a není tomu jinak ani u Reactu. Umožní nám rozdělit uživatelské rozhraní do nezávislých, znovupoužitelných částí. V Reactu jsou komponenty popsány takto: „Konceptuálně můžeme komponenty přirovnat k JavaScriptovým funkcím. Přijímají libovolné vstupy nazývané `props` a vracejí React elementy popisující, co by se mělo na obrazovce objevit.“ [10]

Funkcionální komponenty

Klasické JavaScriptové funkce mohou být v Reactu komponentami. Takováto funkce má právě jeden argument, a to `props`, který je objekt s předávanými daty [10]. Návratová hodnota funkce je React element.

```
function Welcome(props) {
  return <h1>Hello, {props.name}</h1>;
}
```

Výpis 4.1: Příklad funkcionální komponenty

Třídní komponenty

Druhou možností jak vytvořit komponenty je pomocí ES6[4] `class` syntaxe. Tyto komponenty byly dlouho jedinou možností pro využití `state`, avšak ve verzi 16.8 byly vydány takzvané *hooks*, které umožňují toto využít i ve funkcionálních komponentech.

[4] ECMAScript 6

```
class Welcome extends React.Component {
  render() {
    return <h1>Hello, {this.props.name}</h1>;
  }
}
```

Výpis 4.2: Příklad třídní komponenty

State versus Props

Tyto dva JavaScriptové objekty jsou velice důležitými prvky Reactu. Je důležité znát rozdíl mezi nimi a kdy a na co je použít. Jak uvádí blog [3] rozdíl je následující.

`Props` je zkratka pro „properties“. `Props` se předávají do komponentů a můžeme je přirovnat k argumentům, které se předávají funkcím. Důležitým pravidlem je, že hodnoty `props` by se neměly nijak měnit. Jedná se tedy o neměnný objekt. Komponenty, které využívají pouze `props` jsou takzvané „pure components“.

`State` stejně jako `props` uchovává informace o komponentě. Objekt se inicializuje uvnitř komponenty v konstruktoru a obsahuje privátní data dané komponenty. Rozdílem od `props` se možnost změny těchto dat. Hodnota nelze být změněna přiřazením, ale je nutné zavolat funkci `setState()`.

JSX

Jedná se o rozšíření syntaxe pro jazyk JavaScript. Využívá se v komponentech pro definování uživatelského rozhraní a velmi se podobá značkovacímu jazyku HTML. Uvnitř JSX můžeme s využitím složených závorek používat proměnné, podmíněné výrazy v podobě ternárních operátorů nebo JavaScriptové operátory a výrazy.

React DOM

React DOM[5] je rozšiřující balíček, který poskytuje specifické metody pro práci s DOM [10].

Mezi nejdůležitější metodu se řadí `render()`. Tato metoda vykreslí React element do DOM v rámci daného kontejneru a vrátí referenci na komponentu [10]. Pokud je již element vykreslen, tak využije algoritmu pro rozeznání změn a aktualizuje se obsah.

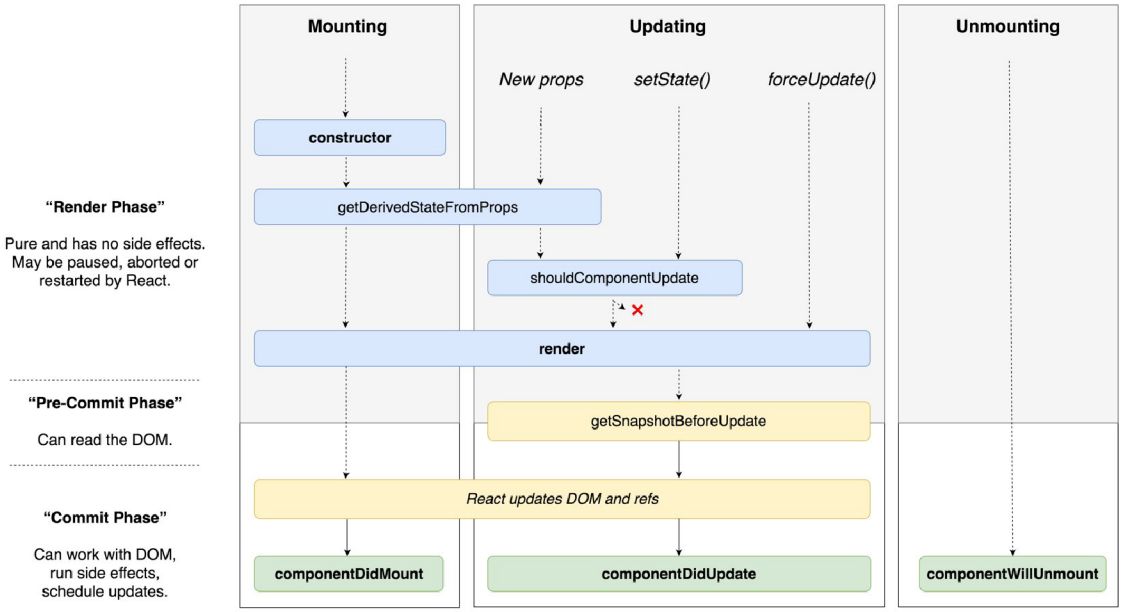
React taktéž využívá konceptu takzvaného Virtual DOM. Jedná se o „virtuální“ reprezentaci uživatelského rozhraní uloženého v paměti a synchronizováno s „reálným“ DOM. V Reactu je virtuální DOM spojen s React elementy, což jsou objekty reprezentující uživatelské rozhraní [10].

Metody životního cyklu komponenty

Následující text je převzat ze stránky [18]. Přehledný diagram cyklu lze vidět na obrázku 4.1. Každá komponenta má svůj vlastní životní cyklus a lze jej rozdělit do tří částí:

- Nasazení komponenty
- Aktualizace
- Zánik komponenty

[5] Document Object Model



Obrázek 4.1: Diagram životního cyklu komponenty s jednotlivými metodami. Převzato z [1]

`render()`

Jedná se o nejvíce používanou metodu a vyskytuje se ve všech třídních komponentech, jelikož je jedinou metodou, která je povinná. Jak vychází z názvu, tak tato metoda se stará o vykreslení samotné komponenty. Metoda se volá při vzniku komponenty a při její aktualizaci. Návratová hodnota je JSX nebo `null` pokud se nemá nic vykreslit. Metoda také musí dodržet, že při stejných vstupech vrací stejný výstup – tzv. „pure“ funkce.

`componentDidMount()`

Tato metoda se volá poté co komponenta byla nasazena v uživatelském rozhraní (vytvořen odkaz na uzel v DOM). Toto je dobré místo pro vykonání API[6] požadavků, které mají za úkol načíst data ze serveru.

`componentDidUpdate()`

Je metoda, která se invokuje při změně. Nejčastějším případem je aktualizace DOM v reakci na změnu `props` nebo `state`. Pokud chceme v této návaznosti změnit `state` je důležité vždy porovnat předchozí stav. Pokud se tato vlastnost nedodrží, tak při použití `setState()` může nastat nekonečné zacyklení.

`componentWillUnmount()`

Je poslední metodou, která často vyskytuje v komponentech. Metoda je volána předtím než je komponenta „zničena“. Využívá se k různým uklízecím akcím jako jsou například ukončení časovačů nebo uvolnění mezipaměti.

[6] Application Programming Interface

4.1.3 Vue

Jako posledního zástupce v této kategorii zmíním je Vue.js. Vue bylo vyvinuto bývalým pracovníkem Googlu Evanem You v roce 2014. Během posledních tří let tento framework zaznamenal výrazný posun v popularitě a to i přesto, že za vývojem nestojí žádná velká firma. Jelikož vznikl z výše zmíněných nejpozději, tak mohl čerpat z jejich silných stránek, které poté zakomponoval do své implementace. [6]

Stejně jako React Vue využívá konceptu virtuálního DOM nebo hlavní zaměření na základní knihovnu. Pro využití směrování nebo globálního stavu aplikace je nutné doinstalování různých balíčků.

S Angularem Vue sdílí podobnost v oddělení HTML a JavaScriptového kódu uvnitř jednotlivých komponent. Stejně jako Angular využívá direktiva a má vlastní CLI.

Mezi výhody, které Vue do světa frameworků přináší patří jeho vysoká flexibilita – pro nejjednodušší implementaci Vue do již existující webové aplikací stačí do HTML souboru přidat skript, který obsahuje CDN[7] link s Vue core skriptem. Pokud vezmeme v potaz pouze základní knihovnu tak je Vue vůči ostatním frameworkům velice kompaktní. Jelikož React přináší novou syntaxi v podobě JSX a Angular je velký balík se všemi nástroji již v základu a navíc využívá TypeScript, tak se dá konstatovat, že Vue má z výše zmíněných technologií nejmírnější křivku učení.

4.2 Serverová strana

Pro vývoj backendové částí je na výběr ze široké škály technologií a programovacích jazyků. Je to dané díky tomu, že řada programovacích přináší podporu pro tvorbu webových aplikací či dokonce přímo specializované frameworky. V této části jsou popsány dva jazyky s přiblížením jejich frameworků.

4.2.1 PHP

PHP[8] je široce užívaný skriptovací jazyk zaměřen především na tvorbu webových aplikací s možností vložení přímo do HTML kódu. Kód se vkládá mezi značky `<?php` a `?>` díky kterým pozná PHP parser, kde se tento kód pro zpracování nachází. [2]

PHP je dynamicky typovaný jazyk. Avšak tento fakt se s poslední verzí 7.4, vydanou v listopadu 2019, změnil, kdy byly přidány typové vlastnosti proměnných [15]. Díky jeho popularitě je tento jazyk na většině serverů podporován a je nezávislý na platformě. Tudíž lze kód jednoduše přenášet mezi jednotlivými operačními systémy.

Frameworků pro tvorbu webových aplikací v PHP je nepřeberné množství. Vybral jsem nejznámější a ty následně více přiblížím dle článku [16].

Laravel

Tento název se většině lidí vybaví jako první, když se řekne PHP framework. Samotná syntaxe vypadá velice elegantně a zároveň je jednoduchá na pochopení. Již vestavěné funkce pro práci se směrováním, management uživatelů atd. Umožňuje asynchronní běh požadavků na pozadí což zvyšuje výkon. Základní knihovna lze snadno rozšířit o další užitečné doplňky, které jsou potřeba. Díky velice aktivní komunitě je možné snadno dohledat všechny potřebné informace.

[7] Content Delivery Network
[8] PHP: Hypertext Preprocessor

CodeIgniter

CodeIgniter je framework, který využívá architekturu MVC[9]. To znamená, že CodeIgniter používá různé komponenty pro řešení specifických úkolů, což je velice rozšířené mezi developery. Mezi výhody se řadí jeho celková jednoduchost a díky podrobné dokumentaci dojde k rychlému vývoji aplikací.

Symfony

Symfony se řadí mezi nejvýznamnější PHP frameworky díky svému rozdělení na flexibilní komponenty, které umožňují, pokud nechceme využít plnohodnotný framework, ale vybrat pouze funkce, které potřebujeme. Taktéž obsahuje již zabudovanou testovací funkcionalitu.

4.2.2 Python

Jedná se o interpretovaný jazyk. Díky vysoko-úrovňovým vestavěným datovým strukturám kombinovanými s dynamickým typováním se stává atraktivním jazykem pro velice rychlý vývoj aplikací, využití pro skriptování nebo spojení již existujících komponent dohromady [17]. Interpret a rozsáhlá základní knihovna je dostupná pro všechny platformy tudíž je zajištěna přenositelnost mezi operačními systémy.

Postupem času vznikly 3 hlavní verze Pythonu:

- Python 1 - vydán roku 1991 a dnes se již nepoužívá
- Python 2 - původně vydán v roce 2000 a oficiální podpora již skončila
- Python 3 - je aktivní verzí vydanou roku 2008

Python v posledních letech velice rychle nabíral na popularitě a nyní podle TIOBE indexu, který indikuje popularitu programovacích jazyků dle četnosti jejich vyhledávání na internetu, se vyskytuje na předních příčkách vedle Javy a C.

Tento jazyk se řadí spíše k netypické volbě pro tvorbu webových aplikací, avšak s vývojem jednotlivých frameworků nelze tuto možnost opomenout.

Flask

Flask je velice jednoduchý framework často nazývaný „micro framework“. Původně byl vyvinut jako wrapper pro šablonový jazyk Jinja[10] a knihovnu Werkzeug[11]. Díky své jednoduchosti má velice nízkou křivku učení, avšak je dostatečný i pro tvorbu větších projektů.

Tento framework nepřináší všechny funkce přímo v základu, ale je na vývojáři, které balíčky a knihovny využije, popřípadě doinstaluje. [19] Chybí zde také nativní podpora databázové vrstvy. Díky tomuto dostává vývojář úplnou volnost nad výběrem té, která se pro jeho projekt nejvíce hodí.

Django

Druhým velice populárním frameworkem v jazyce Python je Django. Využívá přístupu „batteries included“, neboli již v základu zahrnuje podporu všech potřebných funkcí pro tvorbu webových aplikací [7].

[9] Model View Controller
[10] https://jinja.palletsprojects.com/en/2.11.x/
[11] https://werkzeug.palletsprojects.com/en/1.0.x/

Řadí se mezi ně například:

- autentizace uživatelů
- šablony, směrování a pohledy
- admin stránka
- robustní zabezpečení
- podpora různých databází
- a mnohem více

Modely

Jedná se o třídy, které specifikují zdroj informací ohledně dat. Uchovávají nezbytné položky a chování uložených dat. Obecně každý model se dá přirovnat k samostatné tabulce v databázi. [7]

Django obsahuje přímo zabudovaný ORM[12], který vytváří vysokou úroveň abstrakce nad relační databází, čímž nám umožňuje pracovat s daty v jazyce Python namísto SQL[13]. Pro srovnání se lze podívat na výpis 4.3, který je v Pythonu a k němu ekvivalentní SQL kód v ukázce 4.4. Pokud primární klíč není definován, tak Django automaticky přidá do každého modelu řádek s názvem ID, který bude tuto funkci plnit.

```python
from django.db import models

class Person(models.Model):
    first_name = models.CharField(max_length=30)
    last_name = models.CharField(max_length=30)
```

Výpis 4.3: Příklad vytvoření modelu `Person` v Djangu, který obsahuje jméno a příjmení.

```sql
CREATE TABLE myapp_person (
    "id" serial NOT NULL PRIMARY KEY,
    "first_name" varchar(30) NOT NULL,
    "last_name" varchar(30) NOT NULL
);
```

Výpis 4.4: Ekvivalentní kód pro vytvoření tabulky `Person` jako v příkladě 4.3

[12] Object-Relational Mapper
[13] Structured Query Language

Pohledy

Z anglického *views* jsou dalším z klíčových komponent pro tento framework. Pohledy přijímají webové požadavky a vracejí webové odpovědi. Většinou se v nich uchovává nezbytná logika pro vytvoření odpovědí. Proto se zde využívá načítání objektů z databáze a jejich případná modifikace. Odpověď může být různého druhu a to například HTML obsah, webová stránka, přesměrování, návratové kódy, obrázky, XML dokumenty atd. [7]

Django původně používalo funkce pro tvorbu pohledů, ale postupem času přišlo s využitím tříd. Tento posun přinesl ulehčení tvorby pohledů díky vyšší míře templatizace funkčnosti.

Administrační rozhraní

Jednou z nejsilnějších stránek Djanga je automatické vygenerování administračního rozhraní. Tato funkce načítá metadata z vytvořených modelů a poskytuje rychlé rozhraní zaměřené na modely, kde důvěryhodní uživatelé mohou spravovat obsah. [7]

Pro přihlášení do rozhraní je zapotřebí, aby uživatel měl nastavený atribut `is_superuser` nebo `is_staff` na hodnotu `True`. Další možností je vytvoření účtu „superuser“ pomocí obslužného programu příkazové řádky `manage.py`.

4.3 Databáze

Databáze slouží k ukládaní perzistentních dat, které jsou dále využívány různými aplikacemi. Výběr správné databáze pro ukládání prostorových dat, mezi které se GPS data řadí, je velice důležitým krokem. I když v dnešní době již většina databází tuto možnost podporuje pomocí různých rozšíření, tak je podstatné zjistit, které operace nám dovolují s daty provádět.

4.3.1 MySQL

MySQL se řadí mezi nejpopulárnější databázový systém vůbec. Díky vlastnostem jako je volná šiřitelnost, jednoduchá implementace na všech operačních systémech a své rychlosti se řadí mezi nejpopulárnější databáze současnosti.

Co se týče podpory pro prostorové databáze, tak MySQL nepřináší tolik možností jako například PostGIS, o kterém se dozvíme v další kapitole. Podporuje ukládání většiny prostorových datových typů, avšak poskytuje pouze základní operace s těmito daty.

4.3.2 PostgreSQL

Další možností je PostgreSQL. Jedná se o objektově-relační databázový systém. Kromě toho, že se jedná o takzvaný „open source“, tak pro PostgreSQL existuje velké množství různých rozšíření. Na poli ohledně ukládání geografických dat se díky rozšíření PostGIS řadí mezi nejlepší volbu.

PostGIS

Tato nadstavba přináší podporu geografických objektů a základní verzi PostgreSQL rozšiřuje o prostorovou databázi. PostGIS přináší nové datové typy, `Geometry` a `Geography`. Díky těmto typům můžeme do databáze ukládat objekty jako jsou například:

- `Point` - pro uložení bodu s jeho zeměpisnou šířkou a délkou
- `LineString` - sekvence zeměpisných souřadnic nebo objektů typu `Point`
- `Polygon` - vytváří se stejně jako `LineString` avšak s podmínkou, že první a poslední hodnota musí být stejná
- `Collections` - jedná se o kolekcí, které sdružují jednoduché geometrie do sad, mezi které patří `MultiPoint, MultiLineString, MultiPolygon, GeometryCollection`, kde v poslední ze zmíněných se jedná o heterogenní kolekci jakékoli geometrie včetně jiných kolekcí

Kapitola 5

Návrh aplikace

Následující kapitola se věnuje bližší specifikaci požadavků na aplikaci, popisuje různé mapové podklady a knihovny, které se využívají pro interaktivní práci s mapami. Na konci jsou uvedeny technologie, které byly pro tuto práci vybrány.

5.1 Specifikace zadání

Ze zadání a po pozdější domluvě vyplynuly následovné požadavky na aplikaci.

Bude se jednat o webovou aplikaci postavenou na architektuře **klient-server**. S aplikací budou moct pracovat **pouze registrovaní** uživatelé. Systém bude podporovat nahrávání zaznamenaných tras pouze pomocí **.gpx** souborů. Frontendová část aplikace bude napsána v jazyce **JavaScript**.

Editace tras, což je stěžejní část aplikace, bude moct být provedena jak **interaktivně** na mapě, tak i **ručně** zadáním souřadnicových hodnot. Uživatel si bude smět si vybrat **jednu** ze svých nahraných tras, která se mu následně promítne na zvolený mapový podklad. Takto vybrané trase bude možné **upravit polohu** jednotlivých bodů, či body **smazat** (a to i více najednou). Následně, po dokončení úprav, bude možné **stáhnout** trasu ve formátu GPX.

Na základě daných požadavků byl sestaven diagram případů užití, které lze vidět na obrázku 5.1

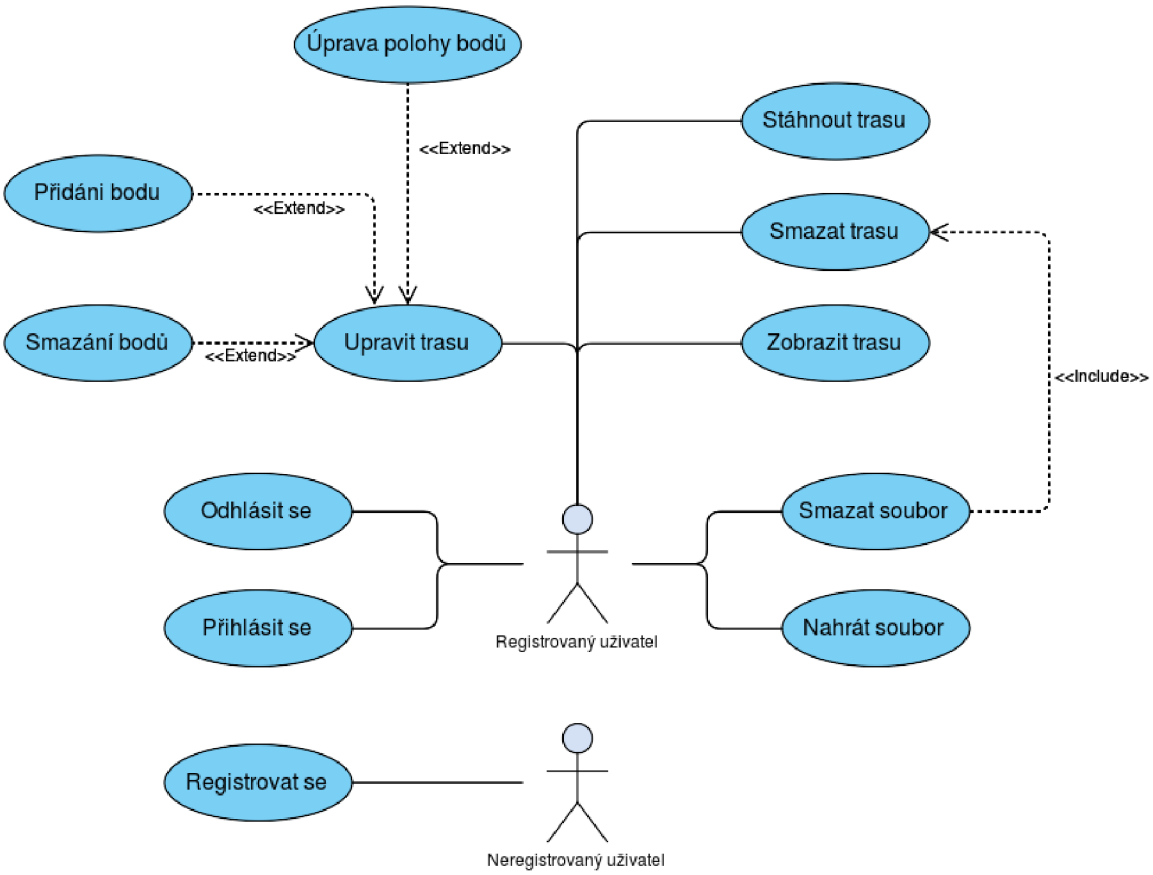
5.2 Mapové podklady

Mapových podkladů existuje velké množství a vývojář si zde může vybrat takový, který nejvíce vyhovuje pro dané užití. Některé služby poskytují data volně za podmínky uvedení zdroje, jiné vyžadují registraci a získání osobního API klíče.

5.2.1 Mapy.cz

Jedná se o velice populární mapový nástroj v České republice. Původně byly určeny pouze pro Českou republiku, avšak postupem času s pomocí ostatních volně dostupných mapových serverů rozšířily své působení prvně na Evropu a nakonec na celý svět.

Tato webová služba nabízí mimo základní vrstvy také leteckou nebo turistickou, která zvýrazňuje turistické trasy, či přidává různé turistické památky. Samozřejmostí je plánovaní tras, či vyhledávání míst s využitím našeptávače.



Obrázek 5.1: Diagram případů užití

Mapy.cz také poskytují vlastní aplikační rozhraní, které slouží pro práci s těmito mapovými podklady. Přináší tedy všechny nástroje, které obsahuje samotná webová služba. Mezi zajímavé funkce API patří zobrazení přímo GPX souborů.

Jelikož se tato služba zaměřuje primárně na české uživatele, tak chybí větší podpora v JavaScriptových frameworcích, které jsou zmíněny v sekci 4.1. Menší komunita okolo této služby, tudíž horší dohledávání různých informací při problémech a primární zaměření map na Českou republiku vedlo k rozhodnutí tuto možnost nevyužít.

5.2.2 Google Maps

Google hraje na poli internetu obrovskou roli a není tomu výjimka ani u mapových služeb. Jelikož za touto službou stojí velká společnost, tak tyto mapové podklady obsahují všechny nejnovější trendy a funkce. Jako první přišly s funkcí panoramatického zobrazení, takzvaného *Street View*.

Podpora zobrazení GPX souborů je zde velice omezená a je doporučeno jej převést do formátu KML, který vyvíjí samotný Google.

Pro použití služeb je nutno se zaregistrovat a získat unikátní API klíč. Nevýhodou, kterou tato služba disponuje, je, že Google přidal omezení na počet požadavků denně a to na 25 000 až 28 000 přístupů měsíčně. Po překročení těchto hodnot se služba zpoplatňuje.

Ačkoli API od Googlu je velice kvalitně zpracované a funkce, které poskytuje jsou opravdu rozsáhlé, tak nevýhody, které jsou spojené se smluvními podmínkami a omezením přístupů zdarma po určitém limitu, vedly k zavrhnutí této možnosti.

5.2.3 OpenStreetMap

OpenStreetMap se řadí mezi *otevřené data*, tím pádem je možné je používat pro libovolné účely, pokud se uvede autorství. Na rozdíl od Google Maps je tato služba řízena komunitou, která dobrovolně přidává a udržuje data aktuální.

OSM[1] je zaměřena primárně na využívání datových serverů, které poskytují mapové podklady. Mezi výběrem schází letecké snímky a panoramatický pohled, avšak tyto mírné nedostatky jsem neshledal jako stěžejní k vytvoření zadané práce.

Služba neobsahuje vlastní API pro interaktivní práci s vytvořenými objekty jakou jsou trasy či různé prostorové výběrové útvary. K této práci je nutné využít jiných JavaScriptových knihoven, o kterých se zmíním v následující sekci.

Díky těmto skutečnostem a možnosti využití těchto podkladů bez jakýchkoli nutných registrací a podobných podmínek, jsem se rozhodl pro tuto volbu.

5.3 Knihovny pro práci s mapou

Se zvolením mapových podkladů OpenStreetMap bylo dalším úkolem zvolit vhodnou knihovnu. Těchto JavaScriptových knihoven, které slouží pro vytvoření mapy na webových stránkách a interaktivní práci s objekty na zvolených mapových podkladech existuje velké množství. Po prozkoumání různých možností se výběr zúžil na dvě velice populární knihovny, které nyní přiblížím.

5.3.1 OpenLayers

První vybranou je open-source knihovna OpenLayers. Tato knihovna poskytuje velmi bohaté API, díky kterému lze vytvořit od jednoduchých interaktivních map s několika vektorovými útvary či markery až po velice komplexní mapové aplikace.

OpenLayers při inicializaci mapy také využívá projekci mapy. Typ projekce se může zvolit, ale jako výchozí volba se používá *Web Mercator projection (EPSG:3857)*. Zajímavou vlastností této knihovny je fakt, že souřadnice udává v pořadí zeměpisná délka, zeměpisná šířka. Toto může řadu vývojářů při prvním použití zaskočit a je nutné si na tento typ zápisu zvyknout. Také je zde přímá podpora zobrazení GIS[2] formátů jako jsou například GeoJSON, KML, GPX atd.

Samotná knihovna má velice bohatou online dokumentaci, kde lze nalézt velké množství příkladů. API dokumentace je velice kvalitně strukturována, avšak kvůli její velikosti se v ví ze začátku trvá vyznat.

5.3.2 Leaflet

Leaflet je momentálně jedničkou mezi open-source JavaScriptovými knihovnami pro tvorbu interaktivních map. Jde o lehkou knihovnu a její použití je velice jednoduché. To ovšem neznamená, že funkcionálně zaostává za dříve zmíněnou knihovnou OpenLayers.

[1] OpenStreetMap
[2] Geographic information system

Knihovna obsahuje všechny potřebné funkce pro tvorbu mapové aplikace. Mimoto stojí za knihovnou široká komunita a mnoho doplňků, které rozšiřují základní verzi. Z podpory GIS formátů Leaflet zaostává za OpenLayers, avšak již zmíněné doplňky tuto mezeru doplňují.

Rozhodujícím faktorem při výběru knihovny bylo, že pro Leaflet byla vytvořena abstrakce v podobě komponentů pro React – React-Leaflet[3]. Ta má také velice kvalitní dokumentaci, upravenou pro využití v Reactu i s jednotlivými příklady.

5.4 Vybrané technologie

V této sekci je popsáno, které technologie, jenž byly popsány v kapitole 4, jsem zvolil pro implementaci výsledné aplikace. U každé části je popsán důvod vybrání a případně využití dalších rozšíření k jednotlivým technologiím. Přehledný diagram výsledného návrhu architektury s jednotlivými technologiemi lze vidět na konci této sekce na obrázku 5.2.

5.4.1 Frontend

Pro tvorbu uživatelského rozhraní jsem vybíral ze tří velice populárních frameworků, které jsou popsány v sekci 4.1. Jelikož rozdíly ve výkonu jednotlivých frameworků jsou v dnešní době takřka minimální, tak mezi rozhodující faktory patřili:

- podpora knihovny Leaflet, která byla vybrána pro práci s mapou
- technologie pro práci s frameworkem
- případné dřívější zkušenosti s daným frameworkem

Po pročtení různých článků, jenž tyto možnosti porovnávaly jsem dospěl k následujícímu rozhodnutí.

Angular

Angular již v základu přináší většinu potřebných funkcí pro vytvoření aplikace. Toto může být na jednu stranu výhodou, ale na druhou stranu to přináší vysokou křivku učení, kde pro začátek práce s frameworkem je potřeba nastudovat jeho architekturu. Dalším faktorem je jeho implementace a programování v jazyce TypeScript, kde je nutné se doučit jeho syntaxi.

Co se týče podpory knihovny Leaflet, tak pro Angular byl vytvořen balíček `ngx-leaflet`[4] Tento balíček poskytuje flexibilní komponenty pro integraci Leafletu do Angularu. Avšak schází zde podrobnější API dokumentace s jednotlivými příklady. Z těchto důvodů jsem tuto možnost nezvolil.

Vue

Vue je na rozdíl od Angularu velice lehkým frameworkem. Po instalaci obsahuje pouze základní funkcionalitu a ostatní balíčky se musí doinstalovat. Co má naopak společného je oddělení HTML šablony a JavaScriptové logiky jednotlivých komponent a tudíž vytváří přehlednější prostředí. Díky těmto faktorům je Vue snadno pochopitelný a křivka učení začíná pozvolna a mírně.

[3] https://react-leaflet.js.org/en/
[4] https://github.com/Asymmetrik/ngx-leaflet

Pro využití tohoto frameworku není potřeba dostudovávat žádné další speciální věci, poněvadž se pracuje pouze s HTML, CSS a JavaScriptem.

K Vue byla vytvořena knihovna Vue Leaflet[5], která obaluje Leaflet a usnadňuje práci s ní. Knihovna má velmi dobře propracovanou dokumentaci s celou řadou příkladů ke komponentům.

Tento framework patřil mezi horké kandidáty, avšak předchozí zkušenost s Reactem rozhodla pro jeho volbu.

React

Jak bylo nastíněno, vítězem v této kategorii se stala knihovna React. Ačkoli by se jako nevýhoda mohla zdát skutečnost, že na rozdíl od předchozích technologií přináší jiný přístup k vytváření aplikací (vše je psáno v JavaScriptu, naučení syntaxe JSX a tak dále), tak dřívější práce, které jsem již vytvořil s touto knihovnou, tyto překážky eliminovaly.

Potřebnou funkcionalitu, která není v základu Reactu, lze snadno doinstalovat pomocí různých balíčků, které jsou aktivně vyvíjeny a spravovány širokou komunitou.

Jak bylo již dříve zmíněno, tak pro React byl vytvořen wrapper právě pro práci s knihovnou Leaflet. Díky tomu je možné přímo používat Reactovské komponenty, což velice usnadňuje práci v tomto prostředí. Dokumentace k tomuto wrapperu je opravdu obsáhlá, přehledná a lze zde najít široká škála příkladů.

Pro tuto práci byla doinstalována celá řada balíčků. Mezi nejvýznamnější patří:

- `react-router-dom`[6] - poskytuje základní směrovací funkce; viz 6.2.2
- `redux`[7] - balíček pro správu globálního stavu aplikace; blíže popsáno v sekci 6.2.3
- `redux-thunk`[8] - middleware pro Redux, který přináší asynchronní vykonávání akcí
- `axios`[9] - jednoduchý HTTP klient založený na objektech typu *Promise*

5.4.2 Backend

Pro serverovou část bylo rozhodování primárně mezi PHP a Pythonem. Úkolem serverové části aplikace je v podstatě pouze zprostředkování komunikace mezi frontendovou částí a databází. Tudíž samotná implementace má podobu pouze aplikačního rozhraní bez zobrazovací funkce.

Jelikož se jedná o návrh velice jednoduché serverové části a všechny možnosti podporují potřebnou funkcionalitu, tak hlavním faktorem pro výběr jazyka byly osobní preference.

PHP

Využití PHP je na poli tvorby serverových částí webových aplikací stále na prvním místě. Od verze 5 podporuje objektové programování a ve verzi 7 prošlo PHP refaktorizací a výrazně se zrychlilo.

[5] https://vue2-leaflet.netlify.app/
[6] https://reacttraining.com/react-router/web/guides/quick-start
[7] https://redux.js.org/introduction/getting-started
[8] https://github.com/reduxjs/redux-thunk
[9] https://www.npmjs.com/package/axios

V PHP je napsaná řada populárních frameworků, které plně dostačují pro danou práci, avšak rozšiřující balíčky, které přinášejí podporu práce s prostorovými databázemi nejsou na úrovni, která je u Django.

Python

Zde zapracovala osobní preference, kde mnohem čistší syntaxe, jednoduché doinstalování balíčků a podpora virtuálních prostředí byly faktory pro zvolení tohoto jazyka. Tudíž zbývalo pouze vybrat mezi Flask a Django. Po prozkoumání jednotlivých možností jsem se rozhodl pro Django.

Django přináší veškeré potřebné věci pro vytvoření serverové části. Dalším plusem bylo již integrované objektově-relačnímu mapování, díky němuž bylo možné psát SQL dotazy přímo v Pythonu. Zásadním faktorem pro zvolení tohoto frameworku však byla nadstavba GeoDjango[10], která z klasického Django vytvoří profesionální geografický framework.

Pro tvorbu API bylo využito Django REST frameworku[11], které jak vyplývá z názvu je přímo určeno pro tento framework.

5.4.3 Databáze

Zde po zvolení GeoDjango jako frameworku pro tvorbu serverové části a přečtení tutoriálu, zůstali na výběr čtyři možnosti: PostgreSQL (s PostGIS), MySQL, Oracle a SQLite.

Oracle

Oracle přináší kvalitní podporu pro ukládání prostorových dat, avšak nutnost registrace a placení poplatků byly dostatečně negativní faktory, které rozhodly o poohlédnutí se za jinými databázemi.

SQLite

Tato databáze zakládá na své kompaktnosti a není tomu ani jinak u nadstavby SpatialLite, která ji rozšiřuje o prostorové data. Kvůli zachování malé velikosti tak databáze nepodporuje různé funkcionality a ani se neřadí mezi nejrychlejší. Schází zde taktéž prostorové indexy, které se v ostatních databázích využívají k rychlejšímu vyhledávání mezi velkým množstvím dat. Z těchto důvodů jsem tuto možnost nezvolil.

MySQL

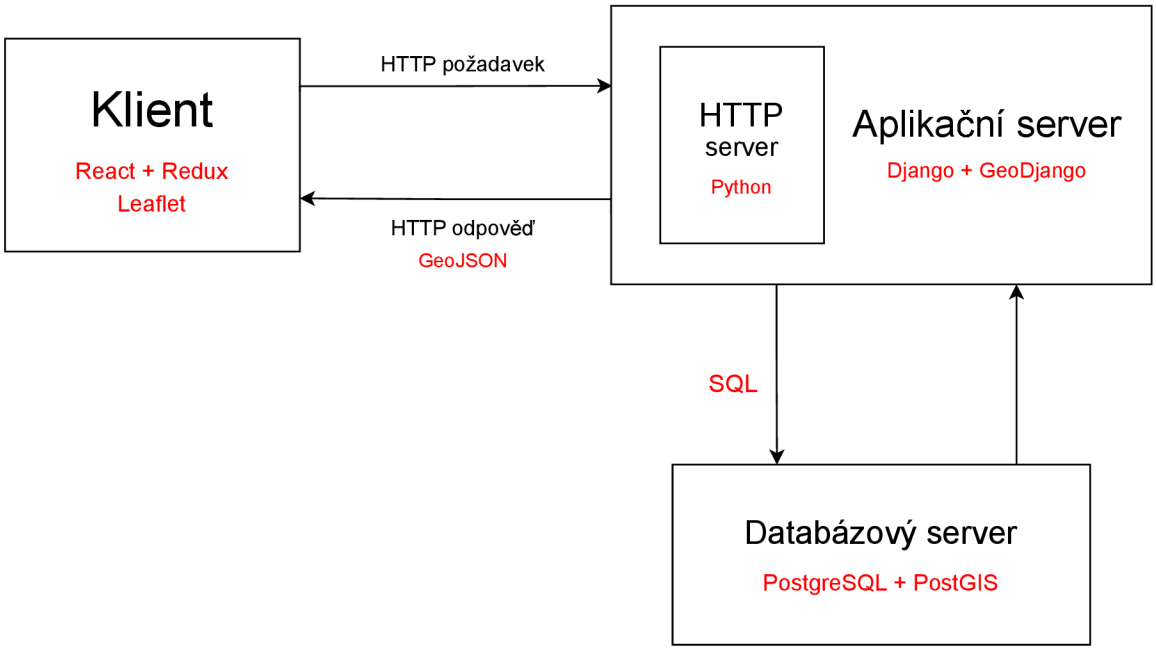
MySQL je velice populární volbou a od verze 5 začalo podporovat prostorová data. Po pročtení doporučení jsem narazil na varování, že právě tato databáze má v Django limitace. Mezi ně patří horší podpora pravých prostorových indexů, kde se dává na výběr mezi rychlým vyhledáváním a integritou dat. Další omezení se nacházelo v menší podpoře prostorových funkcí. Tím pádem zůstala poslední možnost.

PostGIS

Momentálně nejlepší možnost pro tvorbu prostorových databází. GeoDjango tuto možnost doporučuje, jelikož PostGIS je nejvíce vyspělou open-source databázi, která je velmi bo-

[10] https://docs.djangoproject.com/en/3.0/ref/contrib/gis/
[11] https://www.django-rest-framework.org/

hatá na funkce právě pro práci s prostorovými daty [8]. Dal jsem tedy na radu oficiální dokumentace a zvolil ji za svůj databázový základ mé práce.



Obrázek 5.2: Návrh architektury s doplněním použitých technologií

5.5 Vzhled aplikace

V neposlední řadě bylo potřeba také navrhnout grafické uživatelské rozhraní. Důležitým faktorem bylo vytvoření dostatečně velké plochy pro vlastní mapu, kde se budou zobrazovat jednotlivé trasy. Jako inspirace byly vybrány stránky mapových služeb od Googlu[12] a Seznamu[13]. Podle těchto velice populárních služeb jsem se rozhodl rozdělit stránku podobně, na dvě hlavní části – oblast pro mapu a postranní panel.

Pro oblast mapy nebylo navrženo nic speciálního. Jediné co stojí za zmínku je přidání tlačítka, které bude mít za funkci vycentrování mapy, aby se zobrazila celá trasa. V pozdější fázi při implementaci bylo doplněno další tlačítko, které zobrazuje přehlednou uživatelskou příručku pro editační práci.

Postranní panel byl rozdělen do dvou částí. Header, kde bylo plánováno místo pro tlačítka na přihlášení/odhlášení nebo registrace. Zbylá část panelu je využita pro zobrazení jednotlivých bodů, jejich souřadnic a inputů pro jejich změnu. Návrh GUI lze vidět na obrázku 5.3.

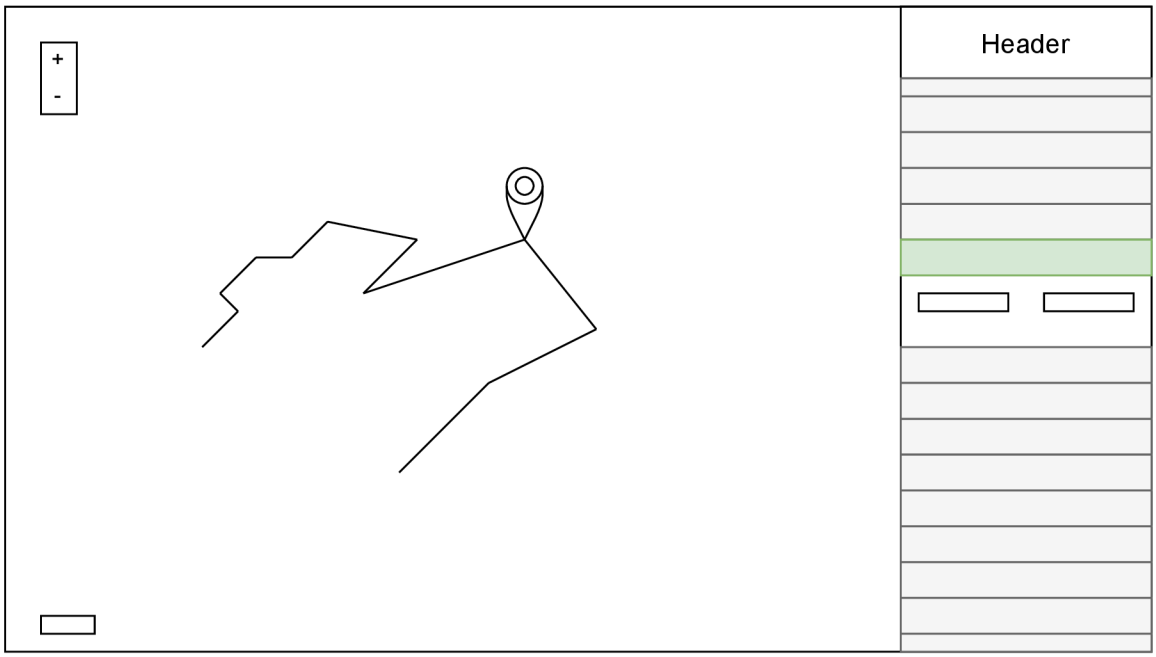
CSS a preprocesor SCSS

Pokud chceme, aby naše stránky vypadaly dobře, tak je třeba sáhnout k jazyku CSS, který se o toto stará. Kaskádové styly ve své poslední verzi CSS3 nabízí opravdu široké možnosti formátování. Populárními zástupci jsou zde `display-flex` a `display-grid`, které

[12] https://www.google.com/maps
[13] https://mapy.cz/

ulehčují rozložení prvků na stránce. Překážkou, která se u využívání nejmodernějších CSS technologií musí překonávat, je podpora prohlížečů. Při větších projektech se počet stylů zvyšuje a pokud se nedodržuje nějaká struktura, tak se zhoršuje jejich údržba. To bylo jedním z důvodů proč vznikly různé preprocesory pro tento jazyk, kde jeden z nich jsem využil v této práci.

Jedná se o preprocesor SCSS. Preprocesory obecně rozšiřují klasické CSS například o proměnné, mixiny, možnost rozdělení stylů do jednotlivých modulů nebo vnořování. Díky těmto vlastnostem se kód zpřehlední a ulehčí případné změny.



Obrázek 5.3: První návrh GUI

Bootstrap

Jde o velice užitečnou sadu nástrojů, díky kterým se stylování webových stránek stává jednodušším. Bootstrap[14] je open-source a obsahuje již předdefinované styly, které lze ve svých projektech využít. Pro lepší používání v Reactu byl vytvořen framework React Bootstrap[15], který přetvořil jednotlivé Bootstrap komponenty do React komponent, avšak stále ponechává kontrolu nad nimi ve vývojářových rukou.

[14] https://getbootstrap.com/
[15] https://react-bootstrap.github.io/

Kapitola 6

Implementace

V této kapitole je popsána samotná implementace klíčových částí aplikace. Kapitola je rozdělena do tří částí a to serverová část, klientská část a testování.

6.1 Serverová část

Jelikož serverová část aplikace je v podstatě implementována pouze jako aplikační rozhraní pro komunikaci mezi databází a klientskou částí, tak se tato sekce věnuje především této problematice. Taktéž jsou zde informace ohledně databáze a její tabulky.

6.1.1 Struktura

Pro vytvoření projektu se typicky využívá příkaz `django-admin startproject` *`name`*. Tento skript obstarává automatické vygenerování kódu, který definuje Django projekt.

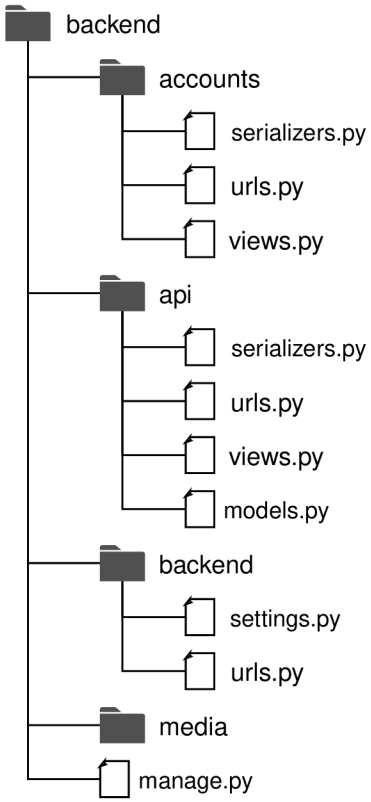
Mezi důležité vygenerované soubory patří:

- `manage.py`, který slouží jako CLI utilita pro administrativní úkoly jako jsou například spuštění serveru, či nahrávání Django modelů do databáze.
- `settings.py` zde se nachází všechny důležité nastavení jako je databáze, nainstalované aplikace v rámci projektu, nastavení REST frameworku, či nastavení cesty pro lokální ukládání nahraných souborů
- `urls.py` soubor, kde se sdružují veškeré deklarace koncových URL adres

Dále se v Djangu vytvářejí takzvané aplikace, což by se dalo popsat jako Pythonovské balíčky, které poskytují určitou sadu funkcí pro daný projekt. V tomto projektu byly vytvořeny dvě aplikace:

  - `accounts`, aplikace obsahující všechny potřebné věci ohledně uživatelských účtů
  - `api`, v podstatě nejdůležitější aplikace celého projektu definující celé aplikační rozhraní

V každé aplikaci lze najít soubory, které neobsahují žádný kód, což je z důvodu, že Django při vytvoření nové aplikace vygeneruje soubory podle jeho konvence. Důvodem je fakt, že každá aplikace je plně nezávislá a pouze rozšiřuje funkcionalitu projektu. Jednotlivé aplikace nemusí být vázány pouze na jeden projekt, ale jsou přenositelné. Ve výsledné struktuře 6.1 jsou zahrnuty pouze soubory, které obsahují důležitý kód v rámci projektu.



Obrázek 6.1: Adresářová struktura backendu s důležitými soubory jednotlivých složek.

6.1.2 Aplikační rozhraní

Aplikační rozhraní (dále jen API) je stěžejní částí celé serverové části. V této sekci je přiblížen Django REST framework, směrování a přehled všech API funkcionalit.

Django REST framework

Jako první je třeba zmínit Django REST framework[1], který byl pro tuto práci zvolen. Tento framework přináší sadu nástrojů, která vytváření API velice zjednodušuje. Taktéž přináší možnost prohlížení API přímo na webu, což pro developerské účely je velice výhodné.

Serializéry, které se v API využívají, umožňují komplexní data jako jsou objekty z databáze či modelu, na nativní datové typy Pythonu, které lze dále jednoduše převést do XML nebo JSONu.

Django REST přináší takzvané *generic views*, které slouží jako zkratka k běžnému schématu použití pohledů. Související pohledy lze dát do jedné třídy a kombinovat jejich logiku. Pro tento projekt byl zvolen *ModelViewSet*.

Typické nastavení těchto pohledů obsahuje:

- `queryset`, reprezentující kolekci objektů z databáze
- `serializer_class`, poskytující validace, serializaci výstupu a deserializaci vstupu

[1] https://www.django-rest-framework.org/

Směrování

Směrování je velice důležitou částí každého aplikačního rozhraní. Slouží k vytvoření koncových URL bodů, kam se zasílají jednotlivé dotazy.

O směrování se v Djangu starají soubory `urls.py`, kde hlavní soubor, který sdružuje všechny směrování od ostatní aplikací, se nachází ve složce `backend`. Směrování hlavního URL souboru lze vidět na výpisu 6.1.

```
urlpatterns = [
    re_path(r'^admin/', admin.site.urls),
    re_path(r'^api/', include("api.urls")),
    re_path(r'^auth/', include("accounts.urls"))
]
```

Výpis 6.1: Hlavní rozdělení URL adres použitých ve výsledné aplikaci.

Při zaslání dotazu na server se Django podívá právě do tohoto souboru, z něhož vyhodnotí, který kód bude proveden. Sekvenčně se prochází jednotlivé položky v `urlpatterns`, což by mělo obsahovat funkce `path` nebo `re_path`. Tyto funkce obsahují dva povinné argumenty, kde prvním je cesta. U `re_patch` lze využít, pro vytvoření řetězce, regulárních výrazů. Druhým argumentem je pohled, který daný požadavek obslouží. Zde jsou dvě možnosti. Buď parametr obsahuje přímo pohled, který požadavek zpracuje nebo `include()` funkci, která provede import cesty do zadaného URLconf modulu.

Pro směrování dotazů se u pohledů typu *ViewSet* využívá `router` namísto klasického `path` nebo `re_path`. Router poskytuje automatické URL směrování, kde pomocí `id` hodnot jednotlivých instancí vytváří jejich URL detaily.

Soubory

Pohled `FileViewSet` sloužící pro zpracování a uložení souborů, je v aplikaci velice důležitým místem, jelikož právě zde, po jeho nahrání, probíhá parsování a vytvoření instancí jednotlivých tras. Po obdržení dotazu a gpx souboru se spustí následující sekvence událostí.

V první řadě se pomocí serializéru zkontroluje jestli v požadavku přišly správná data. Pokud jsou data validní, uloží se objekt do databáze. Následuje volání funkce, která obstarává parsování souboru.

Funkce nese název `create_track_instances` a vyžaduje dva argumenty – gpx soubor a instanci právě vytvořeného souboru. Pro samotné parsování souboru jsem využil balíčku *gpxpy*[2]. Na začátku každé trasy se vytvoří nová instance, kam se po rozparsování souboru uloží načtené hodnoty. Pokud takto vytvořená instance obsahuje validní data, tak je uložena do databáze.

API metody

Přehled API metod s jejich koncovými body pro práci se soubory. Všechny metody vrací kód 403, pokud nejsou poskytnuty ověřovací údaje uživatele.

[2] https://github.com/tkrajina/gpxpy

`GET /api/files`

POPIS: získání seznamu nahraných souborů přihlášeného uživatele

OPRÁVNĚNÍ: přihlášený uživatel

NÁVRATOVÉ KÓDY: 200 úspěch, 403 nevalidní token

ODCHOZÍ DATA: Seznam JSON objektů jednotlivých souborů

```
[
    {
        "id": 6,
        "title": "wiki.gpx",
        "gpx_file": "http://127.0.0.1:8000/media/uploaded_gpx_files/
            wiki.gpx",
        "owner": 9
    },
    {
        "id": 8,
        "title": "ACTIVE LOG101345.gpx",
        "gpx_file": "http://127.0.0.1:8000/media/uploaded_gpx_files/
            ACTIVE_LOG101345.gpx",
        "owner": 9
    }
]
```

Výpis 6.2: Příklad odpovědi pro získání seznamu nahraných souborů od přihlášeného uživatele.

`POST /api/files`

POPIS: nahrání nového souboru od přihlášeného uživatele

OPRÁVNĚNÍ: přihlášený uživatel

NÁVRATOVÉ KÓDY: 201 vytvořeno, 400 špatný typ souboru, 403 nevalidní token

PŘÍCHOZÍ DATA: multipart/form-data ve tvaru: soubor, název

ODCHOZÍ DATA: JSON objekt vytvořeného souboru v podobně dle výpisu 6.2

`GET /api/files/<int:id>`

POPIS: získání detailu souboru zadané pomoci identifikátoru

OPRÁVNĚNÍ: přihlášený uživatel

NÁVRATOVÉ KÓDY: 200 úspěch, 403 nevalidní token, 404 soubor nenalezen

ODCHOZÍ DATA: Detail jednoho objektu souboru v podobě dle na výpisu 6.2

`DELETE /api/files/<int:id>`

POPIS: smazání souboru specifikovaného pomocí identifikátoru

OPRÁVNĚNÍ: majitel souboru

NÁVRATOVÉ KÓDY: 204 žádný obsah, 403 nevalidní token, 404 soubor nenalezen

Trasy

Jak bylo řečeno v předchozí sekci, tak instance tras vznikají při nahrání souboru. Tato API část, mimo automaticky vygenerovaných koncových bodů, obsahuje dva speciální a to pro stáhnutí trasy a získání pouze části trasy závislé na předaném ohraničujícím rámečku.

Pro zpětné vytvoření gpx souboru při požadavku na stažení byl vybrán balíček *yattag*[3], který velice jednoduše umožňuje vytvářet soubory značkovacích jazyků přímo v Pythonu. Jak je popsáno v podsekci 2.2.2, tak soubor může obsahovat metadata. V našem případě, se zde využijí informace o uživateli a přidají se elementy `<author>` se jménem uživatele a jeho emailovou adresou. Druhým elementem je `<time>`, který udává čas stažení souboru.

Další významnou funkcionalitou, kterou server poskytuje při dotazu na trasy, je získání pouze části trasy v závislosti na zaslaných souřadnicích. Server obdrží v dotazu seznam dvou souřadnic, pomocí kterých vytvoří polygon představující plochu, v rámci kterého chce uživatel zjistit, které body trasy se tam nacházejí. Z polygonu se poté vytvoří takzvaná `PreparedGemoetry`, která výrazně urychluje vykonávání porovnávacích metod. Z instance dané trasy se vybere položka `geometry`, což je v našem případě `LineString`. Poté se v cyklu nad vytvořeným polygonem volá metoda `contains()`, níž se jako parametr předává postupně každý bod trasy. Pokud bod leží uvnitř vymezené plochy, uloží se jeho index. Po projití celé trasy vznikne pole indexů, které se odešle zpět uživateli.

Pro serializaci instance trasy jsem využil balíčku *django-geojson-serializer*[4], který přidáním dekorátoru, kde jako parametry se zadávají `geometry` objekty daného modelu, k danému serializéru modelu. Ostatní objekty modelu se při serializaci dají do `"properties"`. Dále, při obdržení požadavku na aktualizaci trasy, serializér kontroluje, zda všechny souřadnice jsou validní.

API metody

Níže nalezneme přehled koncových bodů s jejich metodami, které souvisejí s dotazy na trasy. Všechny metody vrací kód 403 pokud nejsou poskytnuty ověřovací údaje uživatele.

`GET /api/tracks`

POPIS: získání seznamu všech tras přihlášeného uživatele

OPRÁVNĚNÍ: přihlášený uživatel

NÁVRATOVÉ KÓDY: 200 úspěch, 403 nevalidní token

ODCHOZÍ DATA: seznam JSON objektů jednotlivých tras; znázorněno na výpisu 6.3

[3] https://github.com/leforestier/yattag
[4] https://pypi.org/project/django-geojson-serializer/

```
[{
    "id": 16,
    "name": "ACTIVE LOG101345",
    "gpx_file": 8
},
{ "id": 13,
    "name": "Example GPX Document",
    "gpx_file": 6
}]
```

Výpis 6.3: Příklad odpovědi pro získání seznamu všech tras od přihlášeného uživatele.

`GET /api/tracks/<int:id>`

POPIS: získání detailu trasy v podobně GeoJSONu, specifikovaného pomocí identifikátoru

OPRÁVNĚNÍ: přihlášený uživatel

NÁVRATOVÉ KÓDY: 200 úspěch, 403 nevalidní token, 404 trasa nenalezena

ODCHOZÍ DATA: GeoJSON objekt dotazované trasy

```
{
    "type": "Feature",
    "geometry": {
        "type": "LineString",
        "coordinates": [
            [50.302147, 17.159159],
            [50.301889, 17.158339]
        ]
    },
    "properties": {
        "id": 18,
        "name": "ACTIVE LOG112135",
        "elevations": [
            "4.4600",
            "4.9400"
        ],
        "times": [
            "2009-10-17T18:37:26+00:00",
            "2009-10-17T18:37:31+00:00"
        ],
        "gpx_file": 10
    }
}
```

Výpis 6.4: Ukázka GeoJSON odpovědi na dotaz pro získání specifické trasy.

`PUT /api/tracks/<int:id>`

POPIS: aktualizace hodnot trasy zadané pomocí identifikátoru

OPRÁVNĚNÍ: majitel trasy

NÁVRATOVÉ KÓDY: 200 úspěch, 400 nevalidní hodnoty souřadnic, 403 nevalidní token, 404 trasa nenalezena

PŘÍCHOZÍ DATA: GeoJSON objekt trasy v podobně 6.4

ODCHOZÍ DATA: uložený GeoJSON objekt dané trasy v podobě 6.4

`DELETE /api/tracks/<int:id>`

POPIS: smazání trasy zadané pomocí identifikátoru

OPRÁVNĚNÍ: majitel trasy

NÁVRATOVÉ KÓDY: 204 žádný obsah, 403 nevalidní token/zakázaný přístup, 404 trasa nenalezena

`GET /api/tracks/<int:id>/download`

POPIS: stažení trasy, zadané pomocí identifikátoru, v podobě GPX souboru

OPRÁVNĚNÍ: majitel trasy

NÁVRATOVÉ KÓDY: 200 úspěch, 403 nevalidní token/zakázaný přístup, 404 trasa nenalezena

ODCHOZÍ DATA: XML string GPX souboru s danou trasou

`POST /api/tracks/<int:id>/partition`

POPIS: získání indexů bodů, které se nachází ve vybrané ploše

OPRÁVNĚNÍ: majitel trasy

NÁVRATOVÉ KÓDY: 200 úspěch, 403 nevalidní token/zakázaný přístup, 404 trasa nenalezena

PŘÍCHOZÍ DATA: JSON objekt obsahující pole dvou souřadnic pro vytvoření vybrané plochy pro porovnávání

ODCHOZÍ DATA: pole indexů jednotlivých bodů

```
{
    "indexes": [125,126,127,130,241,242,243]
}
```

Výpis 6.5: JSON objekt obsahující seznam indexů bodů ve vybrané ploše.

Uživatelé

Po diskuzi s vedoucím práce panem Burgetem bylo rozhodnuto, že plnohodnotně využívat aplikací budou moct pouze přihlášení uživatelé. Tudíž pro veškerou práci s trasami či soubory je nutné se prvně přihlásit nebo zaregistrovat. Samotné Django přináší již předpřipravené modely pro uživatele, tudíž pro implementaci stačilo postupovat podle dokumentace. Veškeré zabezpečení jako je šifrování hesel či tokenů jsou implementovány přímo ve frameworcích. Samotná implementace přihlašování, registrace a odhlašování v jisté míře kopíruje tutoriál [21].

K autentizaci uživatelů byl využit Django-Rest-Knox[5]. Tento nástroj vytváří každému uživateli při přihlášení nebo registraci unikátní token. Tento token je uložen zašifrovaný v databázi a je u něj uvedena časová známka, kdy jeho platnost vyprší. Uživatel poté, při každém požadavku na server, přiloží do hlavičky v položce `Authorization` svůj token ve tvaru: `Token` *`token`*. Jeho platnost se při každém požadavku automaticky prodlužuje a základní délka života je nastavena na 30 minut. Při odhlášení se token z databáze odstraní.

API metody

Přehled API metod s jejich koncovými body zajišťující autentizaci.

`GET /auth/user`

POPIS: získání informací přihlášeného uživatele

OPRÁVNĚNÍ: přihlášený uživatel

NÁVRATOVÉ KÓDY: 200 úspěch, 403 zakázaný přístup/neplatný token

ODCHOZÍ DATA: JSON objekt obsahující `id, username, email`

`POST /auth/login`

POPIS: přihlášení uživatele

OPRÁVNĚNÍ: každý uživatel

NÁVRATOVÉ KÓDY: 200 úspěch, 400 chybné údaje

PŘÍCHOZÍ DATA: JSON objekt obsahující uživatelské jméno a heslo

ODCHOZÍ DATA: JSON objekt obsahující: `id, username, email` a `token`

```
{
    "user": {
        "id": 9,
        "username": "jim",
        "email": "jim@gmail.com"
    },
"token": "206e68dedde01fbf1573bdfc433e2815b8967974e2213eede61c59903b047822"
}
```

Výpis 6.6: JSON objekt obsahující informace ohledně uživatele a jeho vygenerovaný token

[5] https://james1345.github.io/django-rest-knox/

`POST /auth/register`

POPIS: zaregistrování nového uživatele

OPRÁVNĚNÍ: každý uživatel

NÁVRATOVÉ KÓDY: 201 vytořeno, 400 chybné údaje

PŘÍCHOZÍ DATA: JSON objekt obsahující email, heslo a přihlašovací jméno

ODCHOZÍ DATA: stejný JSON objekt jako ve výpisu 6.6

`POST /auth/logout`

POPIS: odhlášení uživatele

OPRÁVNĚNÍ: přihlášený uživatel

NÁVRATOVÉ KÓDY: 204 žádný obsah, 403 neplatný token

6.1.3 Databáze

Jak bylo dříve zmíněno, tak jako databáze byla zvolena PostgreSQL. Zde po zprovoznění bylo potřeba rozšířit její základní funkcionalitu o PostGIS. To se provedlo v `psql shell` pomocí příkazu: `CREATE EXTENSION postgis;`, avšak ještě před tímto bylo potřeba doinstalovat veškeré potřebné a doporučené knihovny, které PostGIS vyžaduje. Další podstatnou částí bylo v Django projektu doinstalovat *psycopg2*[6] modul, který se používá jako databázový adaptér pro Python. Veškerý výčet potřebných knihoven a postup celé instalace lze nalézt v dokumentaci [7].

Nastavení databáze v Django projektu je umístěno v souboru `settings.py`. Názvy tabulek se generují podle toho, v jaké aplikaci se modely vyskytují. V našem případě se jedná o strukturu: `api_model`.

Schéma databáze

Každý vytvořený model odpovídá jedné tabulce v databázi. Jelikož frameworky generují tabulky, které jsou pro jejich základní funkci nutné automaticky, tak bylo potřeba vytvořit pouze tabulky pro soubory a trasy. Jejich přehled lze vidět na ER diagramu 6.2.

Tabulka `api_gpxfile`

Tato tabulka uchovává potřebné informace ohledně nahraných souborů. Primární klíč zde implicitně doplňuje Django v podobě `id`. Pro automatickou inkrementaci hodnoty je využito PostgreSQL sekvencí. Dále obsahuje sloupec `owner`, což je cizí klíč vůči tabulce `auth_user` a uchovává id vlastníka souboru. `Title` uchovává název trasy a `gpx_file` cestu k nahranému souboru.

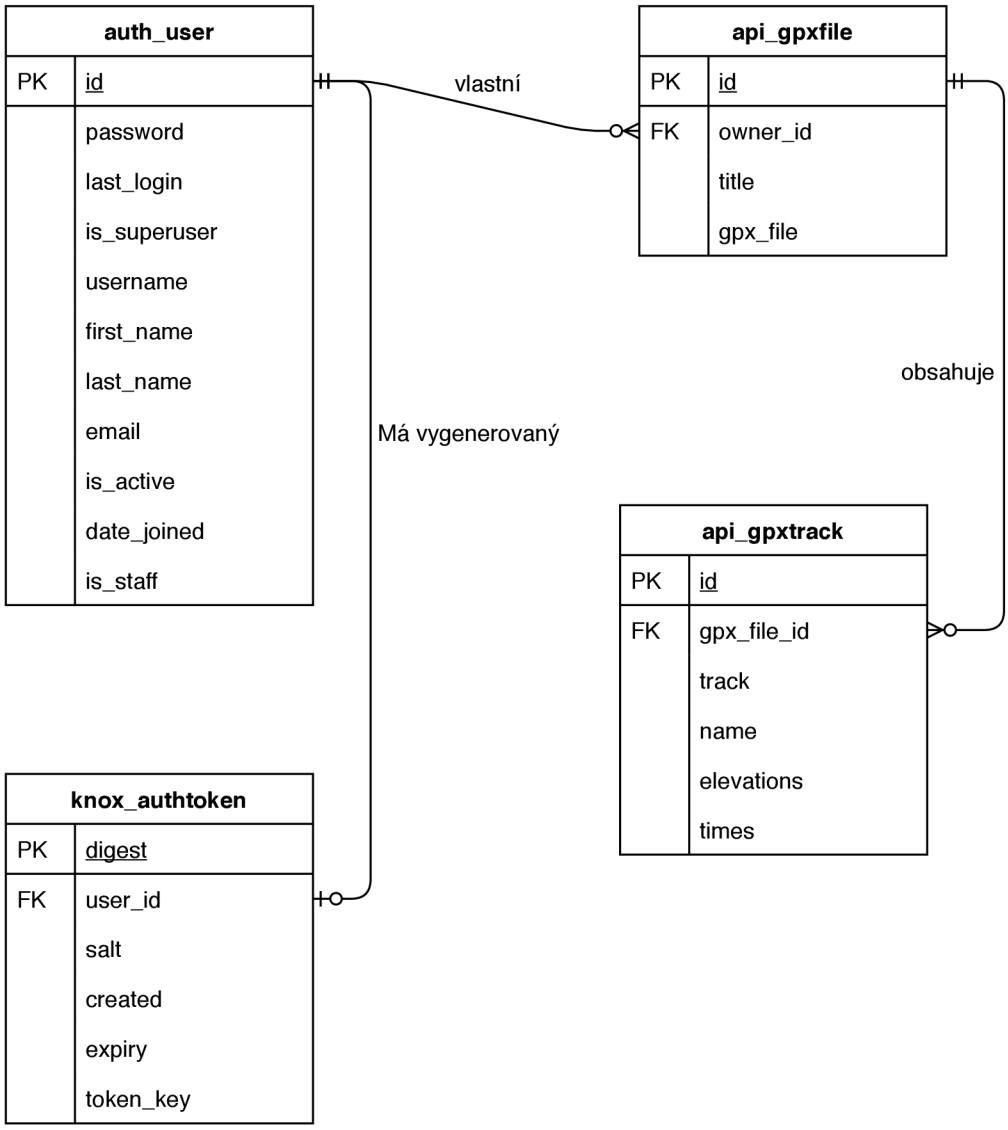
[6] https://www.psycopg.org/

Tabulka `api_gpxtrack`

Jednotlivé trasy se ukládají právě v této tabulce. Primární klíč, stejně jako v předchozím případě, implicitně přidává Django. Cizí klíč `gpx_file` vytváří odkaz na `id` tabulky `api_gpxfile`.

Zbylé sloupce jsou následující:

- `track`, uchovávající trasu pomocí typu *geometry*, přesněji se jedná o `LineString`
- `name`, název trasy
- `elevations`, pole hodnot nadmořské výšky bodů dané trasy
- `times`, pole hodnot zaznamenaných časů jednotlivých bodů trasy



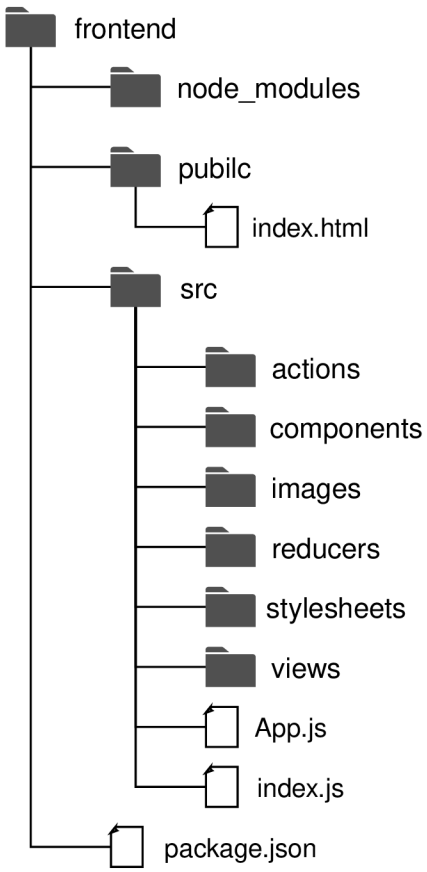
Obrázek 6.2: Entitně-vztahový diagram významných tabulek databáze.

6.2 Klientská část

Tato sekce pojednává o implementaci klientské části webové aplikace. Jak bylo zmíněno v sekci 5.4, tak pro tuto tvorbu byla vybrána knihovna React, která je blíže popsána v sekci 4.1.2. Přiblíží se zde struktura frontednové části, dále doinstalování všech potřebných balíčků a jejich využití v aplikaci. Poslední část se bude věnovat možnostem editace trasy. Stejně jako v serverové části, tak kód, který zajišťuje přihlašování, odhlašování a registraci uživatelů, následuje online tutoriál [21].

6.2.1 Struktura

Na obrázku 6.3 lze vidět adresářová struktura frontendové části aplikace. Jako první se zde nachází adresář *node_modules*, který obsahuje veškeré nainstalované balíčky třetích stran. Reactem automaticky vygenerovaná složka *public* obsahuje základní HTML šablonu, která se pro projekt využívá. Složka *src* obsahuje veškeré zdrojové kódy. Ty jsou podle funkčnosti rozděleny do jednotlivých podadresářů. V poslední řadě se zde nachází soubor *package.json* uchovávající metadata ohledně aplikace a seznam nainstalovaných balíčků.



Obrázek 6.3: Adresářová struktura frontendu. Obsah jednotlivých `src` složek není pro kompaktnost vyjmenován.

Node Package Manager

Jde o správce JavaScriptových balíčků třetích stran. Open-source balíčky, které se v tomto systému nacházejí lze díky tomuto nástroji velice jednoduše stáhnout a využít ve vlastních projektech. Pro doinstalování balíčků se využívá příkazu `npm install [-g] název`. Seznam veškerých využitých balíčků je definován v souboru `package.json`. Díky tomuto nástroji není potřeba při přenosu systému na jiné zařízení přenášet i veškeré balíčky. Cílový uživatel jednoduše ve složce, kde se nachází soubor `package.json` spustí skript `npm install`, který stáhne veškeré balíčky, jenž jsou v souboru uvedeny.

6.2.2 React

Vytvoření React aplikace obvykle probíhá pomocí Node Package Manager (dále pouze NPM) příkazem `npx create-react-app app-name`. To nám nainstaluje základní balíčky potřebné pro práci s Reactem a vytvoří výchozí soubory.

Základní souborem aplikace je `index.js`, který se nachází ve složce *src*. Zde se definuje cílový HTML element ze souboru `index.html` ve složce *public*, kam se vykresluje výsledná aplikace, a Redux *store*, který je blíže popsán v podsekci 6.2.3.

Komponenty

Jak bylo řečeno v sekci 4.1.2, tak React tvorbu uživatelského rozhraní dělí do komponent. V aplikaci lze komponenty nalézt ve složkách *components* a *views*.

Adresář *views* obsahuje čtyři soubory, kde každý představuje vzhled určité stránky. Veškeré pohledy se využívají v komponentě `App.js`, která je přiblížena dále v podsekci ohledně směrování.

- `MainView.js`, hlavní stránka aplikace
- `ErrorView.js`, 404 stránka pro neplatné URL adresy
- `LoginView.js`, formulář pro přihlášení do aplikace
- `RegisterView.js`, formulář pro registraci uživatele

Složka *components* uchovává zbylé komponenty, které se převážně využívají pro zobrazení `MainView.js`. Pojmenování souborů logicky odpovídá jejich funkcionalitě.

Nachází se zde:

- `Alerts.js`, vyskakující okno, pro oznámení uživateli ohledně chyb
- `Input.js`, input element jednotlivých souřadnic
- `Map.js`, komponenta zajišťující veškerou práci s mapou, blíže popsáno v sekci 6.2.4
- `TrackInfo.js`, komponenta obsahující veškeré informace ohledně vybrané trasy
- `SideBar.js`, postranní panel zobrazující jednotlivé trasy uživatele
- `SideBarHeader.js`, hlavička postranního panelu sloužící pro přihlášení/registraci a nahrání souborů
- `SideBarMenu.js`, menu při zobrazení detailu trasy

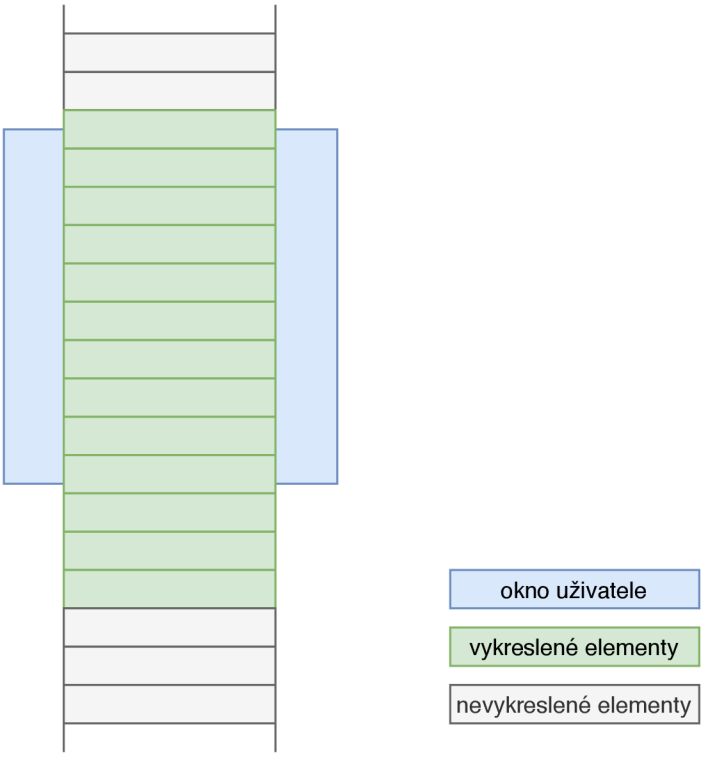
- `SideBarCard.js`, detail vybraného bodu se seznamu
- `SideBarPoints.js`, kontejner pro seznam všech bodů zvolené trasy, viz níže

`SideBarPoints.js`

Této komponentě se bude věnovat trochu větší pozornost. Trasy mohou obsahovat klidně i několik tisíc bodů a vykreslovat tolik elementů do DOM by bylo velice neefektivní. Pro tyto účely existuje balíček `react-virtualized`, který jsem využil.

K vykreslení seznamu, se využívá komponenta `List`, které se předá pomocí `props` výška a šířka každého generovaného řádku, celkový počet řádků a funkce, která je zodpovědná za vykreslení elementu. Pomocí těchto parametrů je možné vypočítat kolik elementů se aktuálně vleze na obrazovku uživatele. Když uživatel začne scrollovat, tak to komponenta zaregistruje a začnou se vykreslovat elementy nové. Řádky, které mají být momentálně zobrazeny, komponenta kalkuluje z počtu odscrollovaných pixelů, velikosti zobrazovacího okna, výšky jednoho elementu, aktuálně zobrazených elementů a celkového počtu. Jelikož zaregistrování scrollování a samotné vykreslení elementu trvá nějakou dobu, tak se zde přidává parametr `overscanRowCount`, jenž udává kolik řádků navíc by se mělo vykreslit. Vykreslování je znázorněno na obrázku 6.4.

Každý element bodu, jsem se rozhodl implementovat jako „accordion“ neboli rozbalovací nabídky, které po kliknutí zobrazí více informací (v našem případě inputy pro ruční editaci souřadnic a případně čas či nadmořskou výšku bodu). Kvůli této vlastnosti bylo nutné využít další komponenty z balíčku, které se starají o automatické počítání šířky a výšky elementu a při jejich změně rekalkuluje elementy k zobrazení.



Obrázek 6.4: Znázornění vykreslování pomocí `react-virtualized`. Parametr pro vykreslení řádků navíc nastaven na hodnotu 3.

Směrování

Pro směrování v Reactu bylo potřeba doinstalovat balíček `react-router`[7]. Jak bylo dříve nastíněno tak, směrování na různé stránky se odehrává v komponentně `App.js`. Dále je popsána typická struktura směrování v Reactu.

Jednotlivé cesty k podstránkám se definují v komponentě `<Route/>`, která vyžaduje dva parametry.

- `path`, popisuje cestu, kam se má výsledná komponenta vykreslit
- `component`, React komponenta, která se při shodě cesty vrátí

Tyto cesty se typicky obalují komponentou `<Switch>`, která prochází postupně jednotlivé cesty a při první shodě vykreslí danou komponentu. Zakázání shody podstránek daného řetězce se udává pomocí parametru `exact`, kdy se hledá pouze přesně zadaný řetězec. Příklad směrování v naší aplikaci lze vidět na výpisu 6.7. Díky vlastnosti, že cesty se porovnávají sekvenčně, tak poslední cesta je definována jako "∗", která se shoduje se všemi cestami. Pokud shoda nastane až zde, tak to znamená, že se uživatel snažil přistoupit na stránku, která neexistuje a tudíž se mu vrátí náš pohled se 404 hláškou.

```
<BrowserRouter basename={process.env.PUBLIC_URL}>
    <Switch>
        <Route exact path="/" component={MainView} />
        <Route exact path="/login" component={LoginView}/>
        <Route exact path="/register" component={RegisterView}/>
        <Route path="*" component={ErrorView}/>
    </Switch>
</BrowserRouter>
```

Výpis 6.7: Ukázka směrování v naší aplikaci.

6.2.3 Redux

Redux je stavový kontejner pro JavaScriptové aplikace. Umožňuje nám mít jeden globální stav nad celou aplikací, ke kterému má, pomocí funkce `connect()`, přístup jakákoli komponenta. Redux lze popsat ve třech základních principech. Přehledný diagram jak funguje Redux cyklus lze vidět na obrázku 6.5.

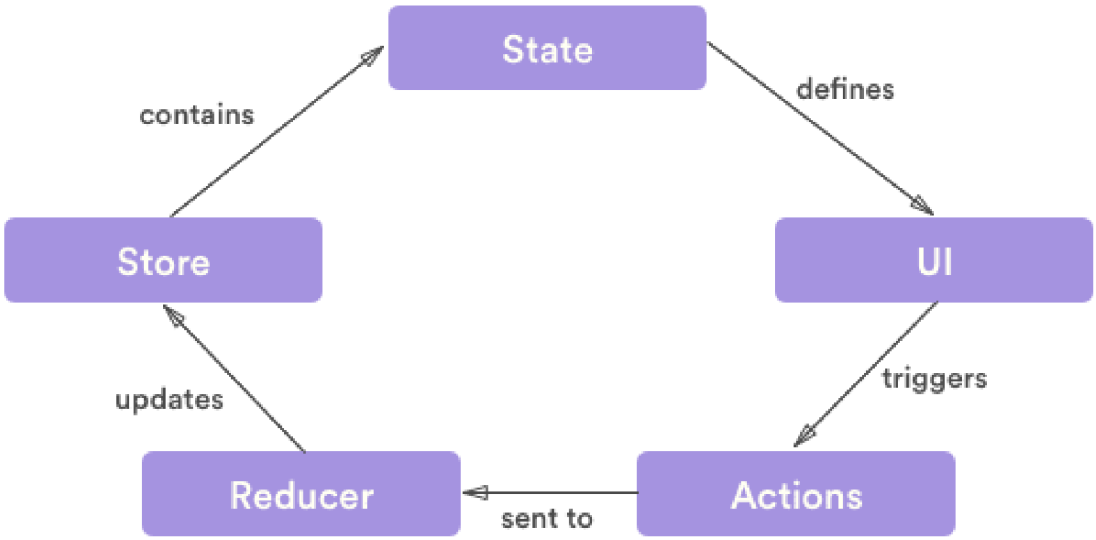
Store

Jedná se o objekt, ve kterém jsou uložená všechna data ohledně aplikace a v celé aplikaci se nachází pouze jeden. V naší aplikaci se v tomto objektu uchovávají následující data:

- `selectedIndex`, index vybraného bodu trasy
- `bounds`, pole souřadnic pro zaslání serveru při vybrání pouze části trasy

[7] https://github.com/ReactTraining/react-router/tree/master/packages/react-router-dom

- `auth`, veškeré informace ohledně uživatele
- `errors`, objekt obsahující status a zprávu pokud server vrátí chybu
- `files`, všechny informace ohledně souborů od uživatele
- `tracks`, veškeré informace ohledně tras od uživatele
- `partition`, pole indexů vybrané části trasy



Obrázek 6.5: Přehled jak funguje Redux cyklus. Převzato z: [20]

Akce

Pokud chceme data ve `store` změnit, využije se zavolání akce. Akce je objekt, jenž musí obsahovat atribut `type`, jímž se akce identifikuje. Dále obsahuje data určené k vykonání změny. K odeslání akce se využívá metoda `dispatch()`. Všechny akce v této práci, jsou rozděleny do souborů podle jejich funkce a umístěny ve složce *actions* dle adresářové struktury 6.3.

Reducery

Reducer je poslední důležitou částí pro Redux. Jedná se o funkci, jenž dostane na vstupu aktuální stav aplikace a akci z metody `dispatch()`. Zde identifikuje typ akce a provede změnu dat stavu aplikace, kterou následně vrátí. Stejně jako akce, tak i reducery mají svůj vlastní adresář, kde jsou rozděleny podle funkcí.

6.2.4 Mapa, zobrazení a editace trasy

Stěžejní část práce spočívala v implementaci mapy, zobrazování a upravování tras na ní. Jak bylo řečeno v sekci 5.3, jako knihovna byla vybrána Leaflet. Samotná inicializace mapy nepředstavovala díky podrobné dokumentaci žádný problém.

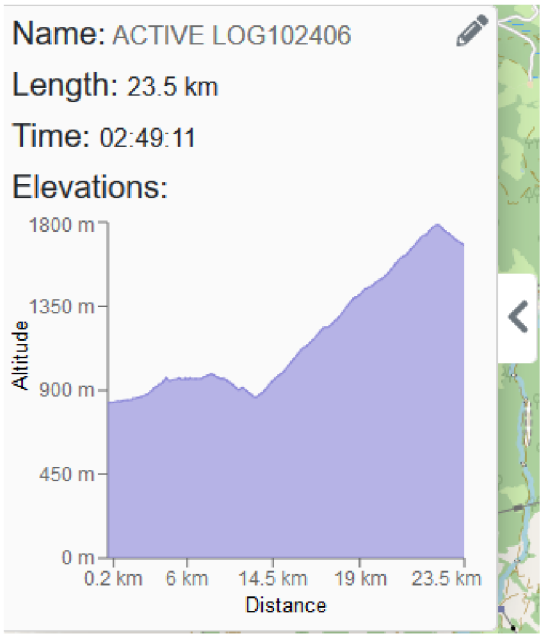
Samotná mapa se chová stejně jako jakákoli jiná online mapa, se kterou se uživatel může na internetu setkat. Specifikem implementované mapy je potlačení přiblížení pomocí dvojitého kliknutí a jeho nahrazení voláním akce přidání značky. Více informací ohledně této funkcionality je popsáno v podsekci selekce části trasy.

Zobrazení trasy

Po výběru trasy ze seznamu dojde k jejímu vykreslení na mapě. Původně bylo zamýšleno využít komponenty `<GeoJSON/>`, protože server při požadavku zasílá právě tento typ formátu a Leaflet tuto možnost přináší. Z pozdějšího průzkumu ale vyplynulo, že se zvolená komponenta nepřekresluje při změně souřadnic a bylo nutné od této varianty upustit. Výsledkem je volba komponenty `<Polyline/>`, které se jednoduše do parametru `positions` předá celé pole jednotlivých souřadnic z globálního stavu a pokud se tyto hodnoty nějakým způsobem změní, tak je trasa automaticky překreslena s novými souřadnicemi.

Informace ohledně trasy

Informace, kterých lze ohledně trasy získat, existuje celá řada. Pro naše účely jsem se rozhodl zobrazit název trasy, který může uživatel změnit, délku trasy, časový interval a graf s výškovým profilem trasy. Poslední dvě zmíněné informace jsou zobrazeny pouze za podmínky, že nahraná trasa tyto informace obsahuje. Celkový čas se vypočítává přímo ze získaných bodů, tudíž startovní čas udává první zaznamenaný bod a cíl poslední. Nevýhoda tohoto způsobu je, že pokud uživatel upraví zvolenou trasu tím, že ji výrazně zkrátí smazáním bodů uprostřed trasy celková doba bude stále stejná, protože první a poslední bod zůstal nezměněn. Pro vykreslení výškového profilu bylo využito knihovny Recharts[8]. Příklad zobrazení informací ohledně trasy lze vidět na obrázku 6.6.



Obrázek 6.6: Přehled informací ohledně zobrazené trasy.

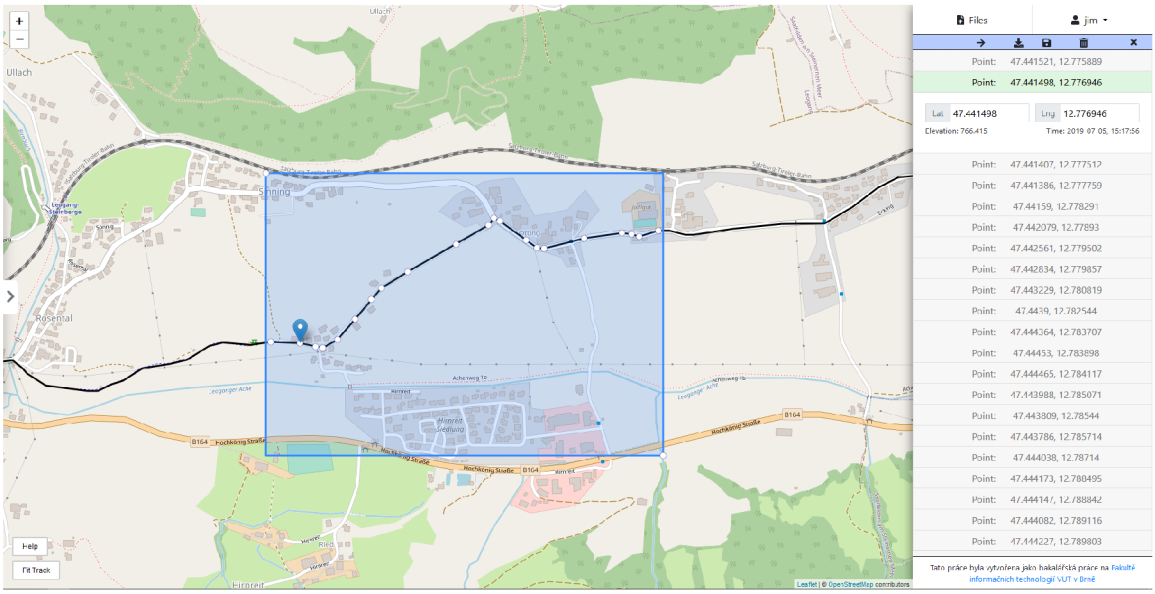
[8] https://recharts.org/en-US

Výběr části trasy

Z důvodu zkrácení doby vyhledávání konkrétního bodu trasy v dlouhém seznamu je pro uživatele připravena možnost výběru pouze části trasy. Tato funkce po výběru oblasti zobrazí uživateli pouze body, které se tam nacházejí, a tím výrazně zrychlí jejich případnou editaci.

Výběr plochy probíhá pomocí dvojitého kliknutí kamkoliv na mapu. Po prvním dvojkliku se objeví značka, která představuje souřadnici pro vytvoření obdélníku. Přidáním druhé souřadnice druhým dvojklikem vznikne na mapě obdélník, který zobrazuje vybranou plochu. Při této události se zároveň zašle požadavek na server ohledně získání indexů bodů, které se v tomto obdélníku nacházejí. Bližší informace ohledně odeslaných a návratových datech lze najít v podsekci API 6.1.2.

Po obdržení odpovědi se překreslí postranní panel, kde se zobrazí pouze body, které se ve vybrané ploše nacházejí. Takto vybranou plochu je možné na mapě dále upravovat a při každém posunu je požadavek na server znovu odeslán. Kromě přepsání souřadnic v panelu se uživateli na mapě také zobrazí uzly těchto bodů, které je možné využít ke změně jejich polohy. Blíže je tato funkce popsána podsekci níže. Zobrazení výběru lze vidět na obrázku 6.7.



Obrázek 6.7: Zobrazení při výběru pouze části trasy.

Přidání bodu

Klasickou funkcionalitou, jenž se u editačních nástrojů tras vyskytuje, je přidání bodu. V aplikaci je tato funkce řešena pomocí kliknutí na zobrazenou trasu. Z kliknutí se získají geografické souřadnice daného místa a ty se pomocí Leaflet funkce `.latLngToLayerPoint()` převedou na promítnuté souřadnice vůči levému hornímu rohu aktuálně zobrazené části mapy. Tato hodnota se společně s objektem trasy předá funkci `.closestPoint()`, což je lehce modifikovaná funkce Leafletu. Veškeré výpočty vykonává funkce od Leafletu a vrací nejbližší bod z trasy a jeho index. Poté se tento bod převede zpátky na geografické souřadnice a vloží se na daný index do globálního stavu trasy.

Pokud trasa obsahuje doplňující informace, jako například nadmořskou výšku a časovou známku, bylo po domluvě s vedoucím práce panem Burgetem rozhodnuto následovně: Jelikož se nepředpokládá, že by nově přidané body nějak extrémně vybočovaly z trasy, bude se hodnota nadmořské výšky kopírovat z předchozího bodu. Časový údaj pak bude doplněn vypočtením průměru z časů předchozího a následujícího bodu.

Změna polohy bodu

Jak vyplývá ze zadání, uživatel by měl mít možnost upravovat body jak interaktivně na mapě, tak i manuálně.

Právě pro manuální úpravu byl vytvořen boční panel, kde nám po zobrazení trasy vyjede nový kontejner, který tyto body obsahuje. V části o jednotlivých komponentách výše, jsem implementaci již trochu nastínil. Jedná se o seznam rozbalovacích nabídek, kde jsou v záhlaví uvedeny souřadnice bodu a po rozkliknutí se nám zobrazí dva interaktivní inputy – jeden pro zeměpisnou šířku a druhý pro délku. Oba mají omezení ohledně vstupních dat pomocí regulárních výrazů, avšak stále může uživatel zadat hodnoty mimo jejich validní interval. Pokud toto nastane, zvýrazní se chybné hodnoty červeně a zobrazí se informativní hláška, která nabádá uživatele pro zadání hodnoty ve validním rozmezí. Pokud uživatel zkusí takto chybně upravenou trasu uložit, vrátí se mu chybový stav a akce se neprovede.

Pro interaktivní úpravu přímo na mapě se využívají značky (komponenta `<Marker>`). Tyto značky se objeví na mapě ve dvou případech. První možností je kliknutím uživatele na libovolný bod ze seznamu v bočním panelu, druhá možnost je spojena s funkcí zobrazení pouze části trasy, která je popsána výše. Obě fungují stejně tím způsobem, že je každé značce nastavena vlastnost `draggable` na hodnotu `true`, čímž se umožní s nimi pohybovat po mapě. Leaflet u značek také podporuje událost `onDragEnd`, která je spuštěna ve chvíli, kdy uživatel pustí značku. Této skutečnosti je využito a v tuto chvíli se volá funkce, která obstará získání polohy značky a následnou aktualizaci globálního stavu.

Takto upravená trasa se nachází pouze na klientské straně a pro uložení na server je nutné tyto změny uložit. Pokud se uživatel rozhodne jít zpět na výběr trasy bez uložení, tak jsou tyto změny zahozeny.

Mazání bodů

Mazání bodů je další funkcí, kterou tento editor přináší. Mazání se využívá například pokud máme více bodů shluknuto blízko sebe a tudíž jejich výpovědní hodnota je téměř nulová. Body lze v aplikaci smazat třemi způsoby.

První možností je při najetí na rozbalovací nabídku bodů v postranním panelu, kde se objeví ikona popelnice. Po kliknutí na ikonu se předá funkci `deletePoint()` index daného bodu. Tato akce provádí změnu pouze na lokálním zařízení a pro nahraní do databáze je potřeba trasu uložit.

Druhá volba je kliknutí ikony koše v horním menu postranního panelu. Po kliknutí se u každého bodu objeví zaškrtávací políčko. Poté co uživatel vybere body, které si přeje smazat, klikne opět na ikonu koše (v této chvíli zvýrazněna červeně), načež se objeví dialogové okno, zda si opravdu přeje tyto body smazat. Důvodem je, že tato akce má dopad i na serverovou část a nová trasa je uložena do databáze. Při tomto způsobu výběru nechybí políčko pro zaškrtnutí všech položek trasy.

Interaktivní mazání bodů přímo na mapě probíhá pouze, když uživatel vybere část trasy. V tomto režimu jsou při dvojitém kliknutí na jednotlivé značky dané body smazány.

6.3 Testování

Testování aplikace probíhalo převážně ručně. Zobrazení aplikace bylo vyzkoušeno v nejpopulárnějších webových prohlížečích mezi, které patří Google Chrome, Mozilla Firefox a Safari. Pro testování API byla využita aplikace *Postman*[9]. Tato aplikace umožňuje širokou konfiguraci HTTP dotazů a přehledné zobrazení odpovědí.

Jelikož se nepředpokládá, že by uživatelé upravovali své trasy na mobilních zařízení, tak testování responzivity probíhalo pouze do rozlišení menších notebooků. K těmto účelům byl využit vestavěný responzivní mód v prohlížeči Mozilla Firefox, kde se nachází již předdefinové velikosti různých zařízení.

6.3.1 Testování uživatelů

Pro testování uživatelů byly vytvořeny jednoduché aplikační testy, které lze najít v adresáři `accounts` v souboru `tests.py`. Tyto testy, kontrolují správnost přihlášení, registraci a získání informací ohledně přihlášeného uživatele. V těchto testech probíhá převážně kontrola návratových kódů.

Důležitou částí, u uživatelů a celkového zabezpečení, bylo řádné otestování autorizačního tokenu. Testovaly se zcela neplatné tokeny, užití po vypršení jejich platnosti, či použití svých validních tokenů k editaci tras, které uživatel nevlastní.

6.3.2 Nahrávání souborů

Testování nahrání GPX souborů a jejich zpracování bylo provedeno na široké škále vzorků a to jak na volně dostupných souborech, souborech zprostředkovaných od vedoucího práce pana Burgeta, či po prozkoumání schématu, vlastnoručně vyrobených simulačních souborech.

Mezi vzorky se nacházely soubory s velkým počtem zaznamenaných bodů (několik tisíců), soubory obsahující více tras, trasy s více segmenty, prázdné trasy, či trasy bez názvů. Dále, po nechtěném zvolení jiného souboru, bylo nahrávání omezeno pouze na soubory, které mají příponu gpx.

6.3.3 Validace stažených souborů

Aplikace přináší možnost stáhnutí trasy, s čímž bylo nutné otestovat validitu takto vytvořených souborů. Podle oficiálních stránek GPX[10] jsem k validaci využil Xerces XML parser[11]. Tento nástroj obsahuje obslužný program příkazové řádky *SAXCount.exe*, kde se k validaci souboru využívá následující příkaz:

```
SaxCount.exe -v=always -n -s -f my_gpx_file.gpx
```

Pomocí tohoto programu byl zjištěn chybný zápis emailu uživatele v metadatech souboru. Po opravení této chyby se docílila potřebná validita.

Dodatečná kontrola také proběhla v podobě nahrání souborů do již existujících editorů. Všechny editory uvedeny v kapitole 3 tyto soubory úspěšně zpracovali a zobrazili.

[9] https://www.postman.com/
[10] https://www.topografix.com/gpx_validation.asp
[11] https://xerces.apache.org/xerces-c/

Kapitola 7

Závěr

V rámci této bakalářské práce vznikla webová aplikace pro editaci geografických dat zaznamenaných pomocí souborů ve formátu GPX. Po prozkoumání různých možností a požadavků byly k implementaci vybrány moderní technologie pro tvorbu webových aplikací jako je knihovna React či framework Django. Díky této volbě bylo zaručena svižnost celé aplikace, což je při práci s trasami obsahujícími i několik tisíc bodů velice důležité.

Po přihlášení a nahrání souborů je uživateli umožněno vybrat si jednu ze svých tras, zobrazit si ji na mapě, zjistit její přehledné informace jako je výškový profil, délka či dobu trvání a následně ji dle potřeb upravovat. Uživatel má zde na výběr z několika funkcí, které jsou typické pro úpravu tras. Patří zde mazání a přidávání bodů, jejich posouvání, či změna názvu trasy. Úpravy lze provádět jak přímo interaktivně na mapovém podkladu, tak ručně zadáváním hodnot v seznamu bodů v bočním panelu. Svou upravenou trasu si může uživatel opět stáhnout ve formátu GPX. Takto stažené soubory mohou být opět využity k přenosu mezi dalšími zařízeními.

Případné rozšíření této práce by mohlo implementovat automatizované úpravy trasy dle různě zvolených parametrů. Jako příklad lze uvést třeba odstranění bodů, jenž se nacházejí do určité vzdálenosti mezi sebou. Další možností, kterou by bylo zajímavé přidat, je automatické přichycení bodů trasy k nejbližší cestě.

Práce s geografickými daty byla pro mne úplnou novinkou a byla potřeba dostudovat veškeré potřebné informace spojené s touto problematikou. To stejné platí i o tvorbě aplikačního rozhraní a technologiemi s tímto spojené. Tento fakt se projevil k závěru práce, kdy bych implementoval určité části trochu elegantněji. Na druhou stranu mi tato práce přinesla obrovské zkušenosti na poli vývoje webových aplikací, které určitě zužitkuji ve své budoucí profesní kariéře.

Literatura

[1] Abramov, D. *I just made this diagram of modern React lifecycle methods. Hope you'll find it helpful!* [online]. 2018 [cit. 2020-5-1]. Dostupné z: https://twitter.com/dan_abramov/status/981712092611989509/photo/1.

[2] Achour, M., Betz, F., Dovgal, A., Lopes, N., Magnusson, H. et al. What is PHP? *PHP manual* [online]. [cit. 2020-5-2]. Dostupné z: https://www.php.net/manual/en/.

[3] Bain, L. *ReactJS: Props vs. State* [online]. 2016 [cit. 2020-4-30]. Dostupné z: https://lucybain.com/blog/2016/react-state-vs-pros/.

[4] Bertagna, P. *How does a GPS tracking system work?* [online]. 2010 [cit. 2020-5-7]. Dostupné z: https://www.eetimes.com/how-does-a-gps-tracking-system-work.

[5] Butler, H., Daly, M., Doyle, A., Gillies, S., Hagen, S. et al. *The GeoJSON Format* [Internet Requests for Comments]. RFC 7946. RFC Editor, August 2016.

[6] Daityari, S. *Angular vs React vs Vue: Which Framework to Choose in 2020* [online]. Aktualizováno 4. 4. 2020 [cit. 2020-4-28]. Dostupné z: https://www.codeinwp.com/blog/angular-vs-vue-vs-react/.

[7] Django Software Foundation. *Django documentation* [online]. [cit. 2020-5-3]. Dostupné z: https://docs.djangoproject.com/en/3.0/.

[8] Django Software Foundation. GeoDjango Installation. *Django documentation* [online]. [cit. 2020-5-13]. Dostupné z: https://docs.djangoproject.com/en/3.0/ref/contrib/gis/install/.

[9] Dolejš, J. *Jak funguje zaměření polohy pomocí GPS?* [online]. 2015 [cit. 2020-5-7]. Dostupné z: https://www.svetandroida.cz/gps-princip/.

[10] Facebook, Inc. *React documentation* [online]. [cit. 2020-4-29]. Dostupné z: https://reactjs.org/docs/.

[11] Foster, D. *GPX 1.1 Schema Documentation* [online]. [cit. 2020-4-22]. Dostupné z: https://www.topografix.com/GPX/1/1/.

[12] Foster, D. *What is GPX?* [online]. [cit. 2020-4-22]. Dostupné z: https://www.topografix.com/gpx_for_developers.asp.

[13] Google. CLI Overview and Command Reference. *Angular docs* [online]. [cit. 2020-4-28]. Dostupné z: https://angular.io/cli.

[14] Google. KML Documentation Introduction. *Keyhole Markup Language* [online]. Aktualizováno 31. 10. 2018 [cit. 2020-4-22]. Dostupné z: https://developers.google.com/kml/documentation/kml_tut.

[15] Hujer, M. *Jaké novinky přinese PHP 7.4* [online]. 2019 [cit. 2020-5-2]. Dostupné z: https://www.zdrojak.cz/clanky/jake-novinky-prinese-php-7-4/.

[16] Morris, W. *8 Best PHP Frameworks for Web Developers* [online]. 2019 [cit. 2020-5-2]. Dostupné z: https://www.hostinger.com/tutorials/best-php-framework.

[17] Python Software Foundation. *What is Python? Executive Summary* [online]. [cit. 2020-5-3]. Dostupné z: https://www.python.org/doc/essays/blurb/.

[18] Ravichandran, A. *React Lifecycle Methods – A Deep Dive* [online]. 2018 [cit. 2020-5-1]. Dostupné z: https://programmingwithmosh.com/javascript/react-lifecycle-methods/.

[19] Ronacher, A. *Flask* [online]. [cit. 2020-5-3]. Dostupné z: https://palletsprojects.com/p/flask/.

[20] Sharma, R. *ABC of Redux* [online]. 2018. Aktualizováno 10. 10. 2018 [cit. 2020-5-17]. Dostupné z: https://dev.to/radiumsharma06/abc-of-redux-5461.

[21] Traversy, B. *Full Stack React, Redux, Django* [online]. 2019. Dostupné z: https://www.youtube.com/playlist?list=PLXE2Bj4edhg5fnlk8C8e-aEONQNPPuqNp.

[22] Šimbera, J. *Souřadnicové systémy.* Výukový list 1.0. Praha: Přírodovědecká fakulta Univerzity Karlovy, 2018. 21 s. Dostupné z: https://www.natur.cuni.cz/geografie/geoinformatika-kartografie/ke-stazeni/projekty/moderni-geoinformacni-metody-ve-vyuce-gis-kartografie-a-dpz/souradnicove-systemy/.