VŠB TECHNICKÁ UNIVERZITA OSTRAVA
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Webová aplikace pro správu GPX souborů

Web Application for Managing GPX Files

Vojtěch Janásek

Bakalářská práce

Vedoucí práce: Ing. Jan Janoušek

Ostrava, 2023

VŠB TECHNICKÁ UNIVERZITA OSTRAVA | FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Č.j.VSB/22/078994
vsboee220260C9

Zadání bakalářské práce

Student: **Vojtěch Janásek**

Studijní program: B0613A140014 Informatika

Téma: Webová aplikace pro správu GPX souborů
Web Application for Managing GPX Files

Jazyk vypracování: čeština

Zásady pro vypracování:

Cílem práce je vytvořit webovou aplikaci pro správu GPX souborů postavenou na frameworku Angular. Aplikace bude svou funkčností pokrývat celý proces vytváření, editace, vizualizace, importu a exportu GPX souborů.
Všechny trasy a informace o nich, jako je nadmořská výška, délka jednotlivých úseků, atd. budou vizualizovány v rámci mapy. Aplikace bude umožňovat automatické doplnění chybějících informací o trase z veřejně dostupných API.

Hlavní body zadání:
1. Seznámení se s moderními technologiemi pro tvorbu webových aplikací a frameworkem Angular.
2. Návrh a implementace webové aplikace pro správu GPS souborů.
4. Výkonnostní testování.
3. Uživatelské testování a zhodnocení dosažených výsledků.

Seznam doporučené odborné literatury:

[1] FAIN, Yakov a Anton MOISEEV. Angular Development with TypeScript. 2. Manning Publications, 2018. ISBN 9781617295348.
[2] SAVKIN, Victor a Jeff CROSS. Essential Angular. Packt Publishing Ltd, 2017. ISBN 9781788291040.
[3] SAVKIN, Victor a Jeff CROSS. Angular Router. Packt Publishing, 2017. ISBN 9781787287150.

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

Vedoucí bakalářské práce: **Ing. Jan Janoušek**

Datum zadání: 01.09.2021

Datum odevzdání: 30.04.2023

Garant studijního programu: doc. Mgr. Miloš Kudělka, Ph.D.

V IS EDISON zadáno: 07.11.2022 12:18:12

Abstrakt

Cílem této bakalářské práce je vytvořit webovou aplikaci na správu a editaci GPX souborů. Na začátku této práce jsou popsány moderní technologie a frameworky na tvorbu webových aplikací. Následně jsou popsány využité technologie pro implementaci aplikace. V další části je seznámení se strukturou aplikace a jsou detailně vysvětleny jednotlivé funkce na editaci GPX souboru. Následuje uživatelské a výkonnostní testování aplikace.

Klíčová slova

Angular; GPX soubor; webová aplikace; TypeScript

Abstract

This bachelor thesis aims to implement web application for managing and editing GPX file. In the begining of the thesis reader is introduced to modern technologies and frameworks for creating web applications. Used technologies in implementation are described in the next part of the thesis. Than the reader is acquainted with structure of the application and functions for editing GPX files. After that the procedure and result of user testing is written down. The last part of the work is dedicated to performance testing of the application.

Keywords

Angular; GPX file; web application; TypeScript

Poděkování

Rád bych na tomto místě poděkoval vedoucímu bakalářské práce panu Ing. Janu Janouškovi, za pomoc při řešení problémů na konzultacích a za množství rad, které mi při psaní této práce poskytl.

Obsah

Seznam použitých zkratek a symbolů

HTML	–	Hyper Text Markup Language
CSS	–	Cascading Style Sheets
JS	–	JavaScript
TS	–	TypeScript
API	–	Application Programming Interface
UI	–	User Interface
GPX	–	GPS exchange Format
RxJS	–	Reactive Extensions Library for JavaScript
URL	–	Uniform Resource Locator
SQL	–	Structured Query Language
NoSQL	–	Not only SQL
GPS	–	Global Positioning System
GUI	–	Graphic User Interface
OOP	–	Object-oriented programming
JSX	–	JavaScript XML
Wi-Fi	–	Wireless Fidelity
SPA	–	Single-page application
DI	–	Dependency injection
DOM	–	Document Object Model
MVVM	–	Model–view–viewmodel
XML	–	Extensible Markup Language
JSON	–	JavaScript Object Notation

Seznam obrázků

Kapitola 1

Úvod

S nástupem GPS zařízení a chytrých mobilních telefonů se rozšířilo využívání aplikací pro navigaci, plánování tras nebo jejich realizaci v dopravních prostředcích, na kole či pěšky.

Uživatel si může vybrat ze široké nabídky nejen aplikací, ale i zařízení, na kterých jsou mapy zobrazovány. Většina aplikací má možnost exportovat data do souboru s formátem GPX. Tento formát zajišťuje kompatibilitu s různými aplikacemi i zařízeními, ale liší se způsobem ukládání dat. Je možné ukládat pouze jednotlivé body naplánované cesty nebo ukládat podstatně detailnější informace. Formát GPX poskytuje mnohem více možností.

Cílem této práce je navrhnout webovou aplikaci, která bude určena pro skupinu uživatelů, kteří potřebují, aby jejich exportované GPX soubory obsahovaly detailnější informace a mohly být exportovány do chytrých hodinek nebo GPS zařízení.

Tato práce je rozdělena do pěti kapitol. První kapitola seznamuje čtenáře s moderními technologiemi a frameworky užitečnými pro tvorbu webových aplikací. Druhá kapitola se věnuje technologiím, které byly použity při implementaci aplikace. Třetí kapitola se detailně zabývá chodem aplikace. Je zde popsána struktura aplikace a jsou vysvětleny konkrétní funkce pro editace souboru. Čtvrtá kapitola popisuje výkonnostní testování aplikace. Poslední kapitola je věnována uživatelskému testování, kde je uživatelské rozhraní testováno pomocí scénářů.

Kapitola 2

Moderní technologie pro tvorbu webových aplikací

2.1 Proč vytvářet webové aplikace

Webové aplikace jsou softwarové aplikace, ke kterým má uživatel přístup pomocí intranetu nebo internetu. V dnešní době je využívání webových aplikací značně rozšířené - od komunikačních aplikací jako jsou sociální sítě, až po komplexní aplikace, například internetové bankovnictví. Jsou vyvíjeny pomocí webových technologií např. HTML, CSS a JavaScript.

Jedním z hlavních důvodů, proč vytvářet webové aplikace, je nezávislost na platformě, což znamená, že je lze používat na jakémkoli zařízení s webovým prohlížečem, bez ohledu na operační systém. Jsou přístupné širokému spektru uživatelů, bez potřeby instalace speciálního softwaru.

Webové aplikace jsou obvykle hostovány na webovém serveru a přistupuje se k nim pomocí URL (Uniform Resource Locator). Uživateli pouze stačí zadat URL aplikace a webový prohlížeč zobrazí požadovanou aplikaci.

2.2 Vybrané moderní technologie

Tato kapitola se zabývá nejčastěji používanými technologiemi pro tvorbu webových aplikací.

2.2.1 HTML5

HTML5 [1][2][3] je značkovací jazyk pro tvorbu webových stránek. Jedná se o nejnovější verzi HTML. Při tvorbě webové aplikace se v dnešní době nejde téměř bez HTML obejít, stává se z něj standard pro tvorbu webových aplikací. Níže jsou uvedeny zajímavé funkce a technologie v HTML5.

1. **Sémantické elementy** jsou značky, které pomáhají definovat strukturu a obsah webové stránky a umožňují internetovým vyhledávačům pochopit kontext a organizaci stránky. Mezi základní elementy patří:

   - <header> - definuje část záhlaví webové stránky,
   - <nav> - používá se pro navigaci na webové stránce,
   - <article> - slouží k definování samostatného obsahu, jako je například článek v novinách,
   - <footer> - používá se k definování části zápatí webové stránky,
   - <figure>, <figcaption> - definuje obrázky a jejich popisky.

2. **Podpora zvuku a videa:** HTML5 má vestavěnou podporu pro přehrávání videa a zvuku, a to odstraňuje potřebu instalace pluginů třetích stran (např. Flash). Pro vkládání zvuku nebo videa se používají elementy <audio> a <video>. Element <source> lze použít k určení více zdrojů v různých formátech, přičemž prohlížeč vybere ten nejvhodnější zdroj. Součástí HTML5 jsou také mediální elementy, které navíc zahrnují rozhraní API pro ovládání přehrávání zvuku a videa, jako je přehrávání, pauza, hledání a vizualizace aktuálního času a délky trvání.

3. **Canvas:** Element <canvas> v HTML5 dává možnost dynamického vykreslování 2D tvarů a bitmapových obrázků. Na webové stránce dovoluje přidat kreslicí plátno, na které se kreslí pomocí JavaScriptu. To přináší možnost vytváření komplexní grafiky, animací a interaktivních vizualizací zobrazovaných v prohlížeči v reálném čase.

4. **Webové úložiště:** HTML5 obsahuje nová rozhraní API pro ukládání dat na straně klienta, která zrychlí a zefektivní webové aplikace. Pro tato webová úložiště existují dva různé mechanismy pro ukládání dat: LocalStorage a SessionStorage. LocalStorage je úložiště, které umožňuje trvalé uložení dat na zařízení uživatele a uživatel k nim může přistupovat i po zavření prohlížeče. SessionStorage je úložiště a na rozdíl od LocalStorage ukládá data pouze pro jedno sezení neboli relaci. Data jsou smazána, jakmile uživatel zavře prohlížeč. Uložená data, jako jsou uživatelské preference, data formulářů na straně klienta, umožňují snížit potřebu časté komunikace se serverem a díky tomu může být webová aplikace rychlejší a pohotovější.

5. **IndexedDB:** [4] je API databáze umožňující ukládání velkých objemů strukturovaných dat na straně klienta, urychluje vyhledávání pomocí použití indexů a nabízí možnost fungování v režimu offline. Na rozdíl od jiných řešení pro ukládání dat na straně klienta, jako jsou LocalStorage nebo SessionStorage, je IndexedDB NoSQL databází. To znamená, že používá systém ukládání dat pomocí dvojice klíč-hodnota. Klíče musí být jedinečné a jejich datové typy mohou být čísla, řetězce, pole i datumy. Hodnoty mohou být navíc oproti klíčům i objekty z JavaScriptu.

API pro IndexedDB je asynchronní. Každá operace s databází se provádí na pozadí, přičemž neblokuje hlavní vlákno UI. Nedochází tak k sekání uživatelského rozhraní při práci s velkými objemy dat.

6. **Geolocation API** umožňuje webovým aplikacím přistupovat k informacím o poloze zařízení uživatele (zeměpisná šířka a délka) a používat získané informace k poskytování služeb, např. mapování a předpověď počasí. Polohu je možné získat pomocí GPS senzoru v zařízení, IP adresy nebo informací o síti Wi-Fi. Tuto funkci podporují téměř všechny webové prohlížeče a digitální zařízení. Aplikace nemá automaticky přístup k informacím o poloze, uživatel musí s jejím používáním souhlasit. Tuto funkci lze použít pro různé typy aplikací, jako je sledování polohy, navigace, sociální sítě a reklamy založené na poloze.

7. **Validace formulářů:** HTML5 přináší nové možnosti validace formulářů, které zkracují čas potřebný na vývoj webových aplikací a zároveň zlepšují uživatelský dojem z aplikace. Validace definuje kritéria, která vyplněný formulář musí splňovat před odesláním na server. To zahrnuje např. kontrolu povinných polí, ověření e-mailových adres a zajištění, že čísla jsou v požadovaném formátu. HTML5 obsahuje nové atributy formulářů, mezi základní patří:

   - required - označuje povinné pole formuláře, které musí být vyplněno před odesláním,
   - pattern - umožňuje zadat regulární výraz pro kontrolu vstupu,
   - min a max - používá se k nastavení platných rozsahů pro numerické vstupy,
   - type - umožňuje určit typ vstupu, jako je e-mail, číslo nebo datum.

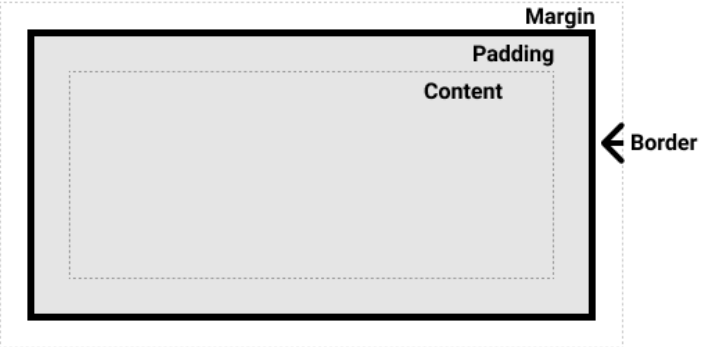
2.2.2 CSS3

CSS3 [3] umožňuje stylování webových stránek pro zvýšení jejich atraktivity a přehlednosti. CSS3 je založeno na takzvaných selektorech, které umožňují vybrat prvky na stránce podle určitých kriterií a potom tyto prvky ostylovat. CSS3 přidalo spoustu nových funkcí a níže budou vysvětleny některé z nich:

1. **Box Model** [5] je model rozvržení, který používá CSS k výpočtu velikosti a pozice prvků na webové stránce. Box model definuje, jak se vypočítává šířka a výška prvku a jakým způsobem se rozděluje prostor kolem prvku.

   Skládá se ze čtyř částí: content, padding, borders a margins. Content je oblast, kde se nachází vlastní obsah prvku. Padding je vnitřní okraj prvku, vyhrazuje prostor mezi content a border. Border je vizuální ohraničení kolem prvku. Margin je vnější okraj, rozšiřuje prostor okolo prvku a používá se k oddělení jednoho prvku od druhého.

   Nová vlastnost *box-sizing*, která byla zavedena v CSS3, definuje způsob výpočtu šířky a výšky prvku. V prohlížečích je výchozí hodnota nastavena na *content-box*, která narozdíl od *border-box* nezahrnuje padding a border do celkové šířky a výšky prvku.



Obrázek 2.1: Vizualizace box modelu[5]

Dalšími vlastnostmi jsou *box-shadow*, umožňující přidávat stíny k prvkům, a *border-radius*, který vytváří zaoblené rohy prvků.

2. **2D a 3D transformace** umožňují manipulovat s pozicí, velikostí a tvarem prvků na webové stránce. Metody transformace:

   - transform - provádí operace jako otáčení, naklonění, změna velikosti a přesun prvků,
   - rotate - otáčí prvek o daný úhel kolem jeho středového bodu,
   - skew - používá se k naklonění prvku horizontálně nebo vertikálně o zadaný úhel,
   - scale - zvětšuje nebo zmenšuje velikosti prvku v poměru k původním rozměrům,
   - translate - posouvá prvek horizontálně nebo vertikálně na dané souřadnice,
   - transform-origin - specifikuje bod počátku pro transformace.

   CSS3 také představila sadu 3D transformačních metod - rotateX, rotateY a rotateZ. Tyto metody otočí prvek kolem dané osy o určitý úhel.

   Všechny tyto metody, spojené s možností animovat transformace pomocí transition a animation, pomáhají vytvářet dynamické a vizuálně zajímavé efekty na webové stránce.

3. **Flexbox Layout** [6] umožňuje bez použití statických hodnot pro pozici prvků vytvářet flexibilní a responzivní rozvržení. Poskytuje způsob, jak zarovnat a rozdělit prostor mezi prvky v kontejneru, i když je jejich velikost dynamická.

   Flexbox vytvoří flexibilní kontejner, ve kterém položky zarovnává podle os do řad nebo sloupců. Mezi hlavní funkce patří:

   - flex-direction - definuje hlavní osu a směr, podle které se prvky za sebe vkládají,
   - flex-wrap - definuje, kdy by se měly položky zalomit, to znamená vykreslit na další řádek,

- justify-content - používá se k zarovnání položek podél hlavní osy,
- align-items - zarovnává položky podél vertikální osy,
- align-self - umožňuje individuálním položkám odlišné zarovnání oproti ostatním položkám v kontejneru.

Flexibilní položky mohou také dostat proporcionální velikost pomocí vlastností flex-grow, flex-shrink a flex-basis.

Flexbox Layout poskytuje jednodušší řešení pro vytváření responzivních rozvržení založených na mřížce, bez použití složitějších systémů CSS grid.

4. **Grid Layout** [7] je systém rozvržení prvků založený na dvourozměrné mřížce skládající se z řádků a sloupců.

   Mřížkový kontejner se vytváří pomocí vlastnosti *display: grid* a řádky a sloupce se definují přes *grid-template-rows* a *grid-template-columns*. Polohu prvků v mřížce zajišťují *grid-row* a *grid-column*. K zarovnání a dalším úpravám prvků slouží:

   - grid-gap - přidává mezery mezi řádky a sloupci,
   - grid-auto-flow - automaticky nastavuje pozice prvku v mřížce,
   - grid-template-areas - umožňuje pojmenovat oblasti mřížky a prvky se poté mohou umístit do těchto oblastí pomocí vlastnosti grid-area,
   - justify-items a align-items - zarovnává prvky v mřížce podél osy řádku a sloupce,
   - justify-content a align-content - zarovnává celé mřížky podél osy řádku a sloupce.

   Grid Layout je vhodnější využít při zarovnávání prvků ve 2D, zatímco Flexbox Layout je lepší volbou při zarovnání v jednom směru.

2.2.3 JavaScript

JavaScript [8] je dynamický objektově orientovaný programovací jazyk, který se stal jedním z nejoblíbenějších programovacích jazyků na světě. Původně byl vyvinut v polovině 90. let Brendanem Eichem u společnosti Netscape Communications Corporation a od té doby se stal nezbytným nástrojem pro vývojáře webů.

JavaScript umožňuje vytvářet dynamické webové stránky s interaktivními prvky GUI např. s posuvníky obrázků a interaktivními mapami. JavaScriptový kód se obvykle spouští přímo v prohlížeči, uživatelé tak mohou s webovou stránkou pracovat bez potřeby obnovování stránky.

JavaScript je také vhodný pro použití v serverových aplikacích, příkladem jsou Node.js a Apache. Pomocí JavaScriptu lze vytvořit kompletní webové aplikace od front-endu po back-end.

Desktopové a mobilní aplikace se dnes neobejdou bez JavaScriptu, který je možné rozšířit o širokou škálu knihoven a frameworků (AngularJS, React a Vue.js), které usnadňují vývoj složitých aplikací.

Jednou z hlavních vlastností JavaScriptu je podpora objektově orientovaného programování. Podporuje tvorbu tříd, objektů a dědičnosti, a to umožňuje vytvářet opakovaně použitelný a modulární kód. Také podporuje funkcionální programování a ve verzi Javascriptu z roku 2017 byly přidány důležité funkce jako async a await.

JavaScript může být použit k vytváření jednoduchých skriptů i velkých aplikací. Může být použit v různých prostředích, jako jsou webové prohlížeče, servery a dokonce i na desktopu. Je také interpretovaným jazykem, což znamená, že před spuštěním ho není třeba kompilovat.

2.2.4 TypeScript

TypeScript je programovací jazyk a je nadstavbou JavaScriptu. Vnáší do OOP přehlednost zejména zavedením kontroly datových typů. TypeScript byl poprvé vydán v roce 2012 společností Microsoft a od té doby jeho obliba vzrostla, zejména mezi projekty pro vývoj webových aplikací velkého měřítka.

Přechod z již stávajícího JavaScriptového projektu na TypeScript je výhodný v tom, že kód JavaScriptu je platný TypeScriptový kód, díky tomu je možné postupně přijmout TypeScript do kódového základu v JavaScriptu, bez nutnosti přepisování již stávajícího kódu do TypeScriptu. Navíc je možné použít libovolné knihovny z JavaScriptu v TypeScriptu.

2.2.4.1 Rozdíly mezi JavaScriptem a TypeScriptem

Klíčovým rozdílem mezi JavaScriptem a TypeScriptem je, že JavaScript postrádá **systém typování proměnných**. JavaScript je dynamicky typovaným jazykem, což znamená, že typ proměnné se určuje během běhu a může se kdykoliv změnit. TypeScript je na druhé straně staticky typovaný jazyk, což znamená, že typ proměnné musí být určen při kompilaci a nemůže se změnit, ale je možné využít datový typ *any* pro přidání proměnné bez předem určeného datového typu. Tyto dodatečné informace o typech usnadňují práci s projekty velkého měřítka.

Dalšími rozdíly jsou:

- **Syntaxe** v TypeScriptu se více podobá tradičním objektově orientovaným programovacím jazykům, jako je Java, zatímco JavaScript má syntaxi, která se více blíží funkcionálním programovacím jazykům, jako je Python nebo Ruby.

- **Nástroje:** TypeScript má oproti JavaScriptu lepší podporu pro moderní integrované vývojové prostředí (IDE) a editory, které mohou nabízet dodatečné funkce jako doplnění kódu, zvýraznění chyb a refaktorování.

- TypeScript podporuje koncepty **objektově orientovaného programování**, jako jsou třídy, rozhraní a dědičnost, což usnadňuje psaní strukturovaného a udržitelného kódu. JavaScript na druhou stranu má omezenou podporu pro koncepty objektově orientovaného programování, což znesnadňuje psaní strukturovaného a udržitelného kódu.

2.3 JavaScript frameworky

JavaScript frameworky jsou sbírkou knihoven obsahujících kód napsaný v JavaScriptu, které usnadňují práci softwarovým vývojářům. Každý JavaScriptový framework nabízí předpřipravené a naprogramované části aplikace, které jsou užitečné pro různé úseky v softwarovém vývoji, což velice šetří čas.

JavaScriptové frameworky poskytují strukturu pro kód, takže je důležité, aby před začátkem tvorby nové aplikace se vývojař naučil se strukturou pracovat. JavaScriptový framework poskytuje šablonu, podle které je vhodné se řídit a využít už předpřipravené části aplikace, které nabízí.

Většina frameworků je open-source, což znamená, že jsou neustále zlepšovány komunitou vývojářů a většinou bývají dobře odladěné a často aktualizované. Také nabízejí možnost si libovolně upravit jakoukoli jeho část. Mezi nejrozšířenější JavaScript frameworky patří AngularJS, React, Vue.js.

2.3.1 Rozdíl mezi JavaScript framework a JavaScript knihovnou

JavaScriptové frameworky a knihovny jsou dva různé typy nástrojů, které se často používají při vývoji webových aplikací. Oba poskytují předem napsaný kód, který může být použit k usnadnění vývoje a zefektivnění práce. Používají se však různými způsoby. JavaScriptová knihovna je sbírka funkcí, které lze volat z vlastního kódu pro provedení specifických úkolů. Knihovny se obvykle zaměřují na konkrétní funkčnost. Příklady populárních JavaScriptových knihoven jsou jQuery a Lodash. JavaScriptový framework je komplexnější sadou nástrojů, která poskytuje strukturu pro vývoj webových aplikací. Zahrnuje sadu knihoven, stejně jako dodatečný kód, který pomáhá organizovat a strukturovat vytvářený kód. Frameworky jsou navrženy tak, aby usnadňovaly vývoj složitých aplikací.

2.3.2 Proč si zvolit JavaScript framework

Je důležité si uvědomit, že použití frameworků může pomoci usnadnit a urychlit vývoj aplikací, ale jejich použití není nezbytné pro všechny projekty. Vývojáři by měli zvážit přínosy a možná omezení jednotlivých frameworků a zvolit ten, který nejlépe vyhovuje potřebám jejich projektu.

2.3.3 Angular

Angular [9][10] je přední open-source framework pro programování všestranných jednostránkových webových aplikací, který používá jazyk TypeScript. V dnešní době Angular vyvíjí společnost Google.

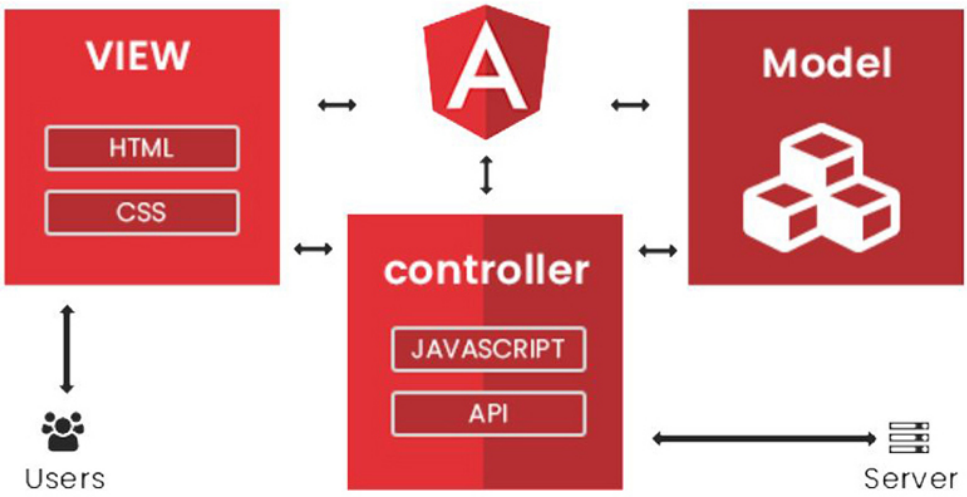
Základní stavební prvek Angularu jsou komponenty. Aplikace se skládá z více komponent, kde každá komponenta plní svoji funkci a navzájem se spolu doplňují a spolupracují.

Každá komponenta se skládá z:

- HTML šablony, která deklaruje, co se vykresluje na stránce,
- TypeScript třídy, která definuje chování,
- CSS selektoru, který definuje, jak se komponenta používá v šabloně,
- Volitelné CSS styly aplikované na šablonu.

TypeScript třída obsahuje dekorátor *@Component*, který definuje chování a logiku komponenty. Pro manipulaci se třídou se používají vlastnosti a metody. Třída také umožňuje komunikaci mezi ostatními komponentami pomocí importu Angular služeb. Díky tomu může třída přistupovat i k metodám a funkcím jiných komponent.

Každá třída komponenty prochází tzv. "životním cyklem"(stavy životního cyklu např. vytvořena, aktualizována nebo zničena). Jedná se o série metod, ke kterým se v konkrétních bodech existence komponenty přistupuje. Metody jsou volány v konkrétních bodech životního cyklu komponenty.



Obrázek 2.2: Ukázka MVC [9]

Angular využívá vzor **Model-View-Controller (MVC)**, který definuje logiku rozdělení aplikace do tří samostatných komponent: modelu, pohledu a kontroleru. V modelu by měla být zpraco-

vávána a uložena data. Pohled zobrazuje UI uživateli, kde data pro zobrazení a následnou aktualizaci UI pochází z modelu. Kontroler spravuje komunikaci mezi modelem a pohledem. Umožňuje poslat data zadaná uživatelem z pohledu do modelu, a při změně dat v modelu aktualizuje pohled. Logika MVC usnadňuje pochopení, testování a údržbu aplikace.

Angular **Dependency Injection (DI)** je mechanismus, kterým Angular automaticky poskytuje komponentě závislosti, které potřebuje k fungování. Komponenta spoléhá na externí logiku (služby nebo objekty), bez vědomosti o tom, jak se daná logika tvoří. DI usnadňuje testování a správu aplikace.

2.3.4 React

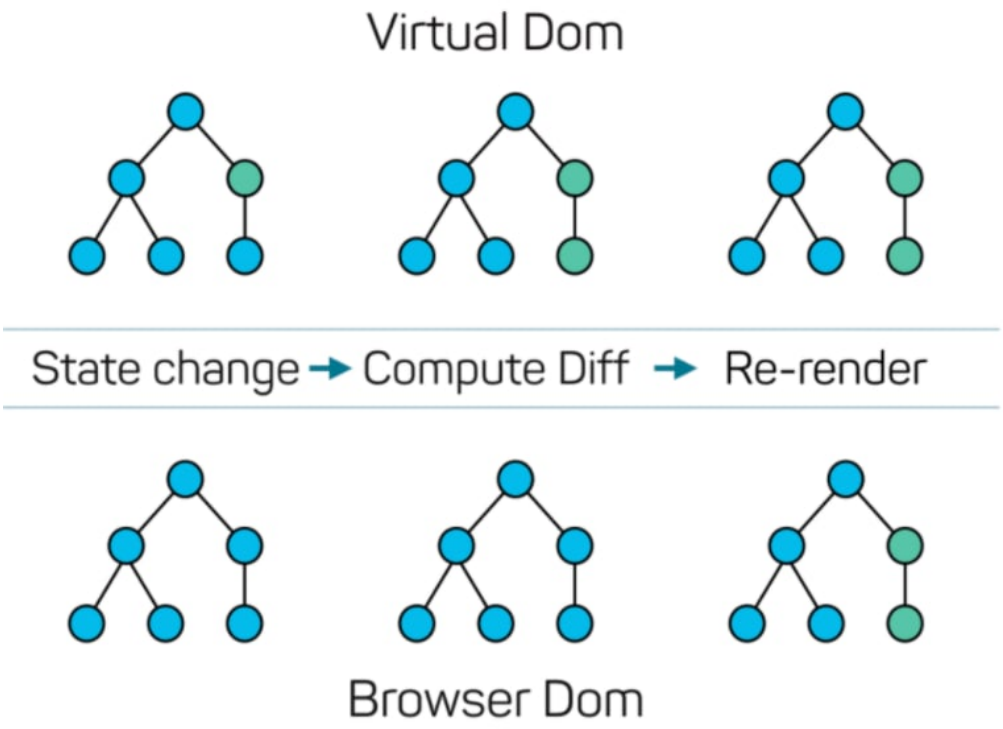
React [11] je JavaScriptový open-source framework pro vytváření front-endu webových aplikací. Byl vyvinut a je spravován společností Meta (v minulosti Facebook). React se snaží zjednodušit údržbu aplikace, např. pomocí vytváření opakovaně použitelných komponent UI. Je vhodný pro tvorbu jednostránkových aplikací (SPA), protože dokáže optimálně zpracovávat rychle se měnící data. Jeho výhodou je i rozsáhlý výběr dostupných nástrojů a knihoven pro implementaci složitých webových aplikací.

Také podobně jako Angular používá React komponenty, které jsou obvykle psány v JavaScriptu a JSX. Komponenty mohou být definovány pomocí třídy nebo funkce a mohou přijímat vlastnosti (props) a stav jako vstupy. Až je komponenta definována, je pomocí JSX nebo metody render() vykreslena do DOM. Vykreslená komponenta vrátí strom dalších komponent a prvků, které tvoří uživatelské rozhraní. Poté proběhne aktualizace DOM, kde se aktualizují pouze části, které se změnily.

Ve Reactu může být komponenta buď "stateful"nebo "stateless". **Stateful** komponenta si udržuje svůj vlastní stav a může se sama aktualizovat v reakci na událost (např. interakce uživatele). Stateful komponenty jsou definovány pomocí třídy a mohou přijímat data od jiných komponent pomocí *props*. **Stateless** komponenta přijímá všechna svá data jako *props* a neudržuje svůj vlastní stav. Tyto komponenty jsou definovány jako JavaScriptové funkce. Mohou pouze přijímat a zobrazovat data a nemají schopnost se samy aktualizovat. Nejčastěji se stateless komponenty používají k vykreslování UI a stateful komponenty řídí stav a data aplikace.

React pro vykreslování používá **JSX (Javascript XML)**, který je rozšířením JavaScriptu, a poskytuje způsob, jak jednoduše strukturovat vykreslení komponenty se syntaxí podobnou HTML. JSX umožňuje psát prvky HTML a JavaScriptu dohromady, tím vytváří jednodušší a čitelnější kód.

Virtual DOM (Document Object Model) DOM reprezentuje stav uživatelského rozhraní aplikace, přičemž kdykoli dojde ke změně stavu uživatelského rozhraní, DOM se aktualizuje. Obvykle tato změna stavu ovlivňuje skutečný (real) DOM, což vede k pomalému výkonu. Skutečné DOM mají obvykle mnoho komponent uživatelského rozhraní s mnoha prvky, které je třeba neustále znovu vykreslovat.



Obrázek 2.3: Jak funguje Virtuální DOM [12]

Virtuální (virtual) DOM funguje mnohem lépe než skutečný DOM. Při změně stavu v aplikaci se mění, virtuální DOM, který se efektivně aktualizuje místo skutečného DOM, čímž se zvýší efektivita. Virtuální DOM vypočítá nejlepší možný způsob, jak provést změny uživatelského rozhraní ve skutečném DOM přičemž skutečný DOM by musel aktualizovat každý prvek zvlášť.

React používá **Dependency Injection (DI)** a na rozdíl od Angularu React tuto funkci nemá zabudovanou. React používá techniku nazývanou "props drilling"k předání dat a závislostí mezi komponenty. Každá závislost musí být ručně předávána přes několik úrovní komponent. Řešením tohoto problému může být použít externí knihovny jako react-redux k efektivnější správě stavu a závislostí.

2.3.5 Vue.js

Vue.js [13] je open-source JavaScriptový framework pro tvorbu UI a single-page aplikací (SPA). Jednou z hlavních výhod Vue.js je integrace architektury Model-View-ViewModel (MVVM). Vzor MVVM je způsob rozdělení logiky aplikace do tří odlišných komponent:

- Model - spravuje data a logiku aplikace,
- View - zobrazuje UI, které se stará o zobrazení dat pro uživatele,

- ViewModel - funguje jako prostředník mezi modelem a pohledem, to znamená, že převádí data z modelu do pohledu.

Ve Vue.js je ViewModel implementován jako instance Vue, která slouží k interakci mezi modelem a pohledem. Instance Vue je zodpovědná za:

- sledování a následnou reakci na změny dat v modelu,
- aktualizaci pohledu s aktuálním stavem dat z modelu,
- zpracování uživatelského vstupu a provedení změn dat v modelu.

Syntaxe založená na šablonách definuje pohled a popisuje, jak by se měla data zobrazovat. Instance Vue poté spojuje pohled s daty modelu a automaticky aktualizuje pohled při každé změně dat.

Zásadní vlastností Vue.js jsou jeho directives, které se používají k úpravě chování prvků HTML. Přidávají se spolu s jakýmikoli elementy HTML do šablony Vue. Na základě implementace funkce *directive* se pomocí těchto speciálních atributů provádí různé druhy akcí. Vue.js podporuje mnoho vestavěných directives, ale vývojáři si mohou také vytvořit vlastní. Jde je např. použít na skrývání/zobrazování prvků, přidávání/odebírání prvků atd.

Vue.js má vestavěnou **Dependency Injection (DI)** stejně jako Angular. Systém DI ve Vue.js poskytuje přístup komponentám ke sdíleným službám, přičemž je jeho použití jednoduché a přehledné. V Angularu je DI složitější a flexibilnější, nepoužívá se jen k poskytnutí přístupu k sdíleným službám, ale také k řízení životního cyklu těchto služeb a k řešení dalších typů závislostí.

Vue.js používá také **Document Object Model (DOM)** k reprezentaci struktury komponent aplikace. Aktualizace DOM probíhá v reálném čase v závislosti na stavu aplikace. Podobně jako v Reactu i ve Vue.js se používá virtuální DOM, který je paměťovou reprezentací skutečného DOM. Pokud se změní stav aplikace, aktualizuje se virtuální DOM, a až následně se porovnává virtuální DOM se skutečným DOM, kde se aktualizují pouze prvky, u kterých došlo ke změně. Tento proces se nazývá "re-rendering"a je mnohem efektivnější než aktualizace celého DOM při každé změně.

Zatímco React a Vue.js využívají virtuální DOM, Angular používá skutečný DOM s detekcí změn pro aktualizaci pohledu. Angular má také syntaxi založenou na šabloně podobně jako Vue a React, ale navíc obsahuje systém pro *data binding*, který umožňuje snadné vázání dat na prvky a řízení chování prvků v závislosti na stavu aplikace.

Kapitola 3

Technologie použité při implementaci

3.1 GPX soubor

GPX (GPS Exchange Format) [14] je speciální formát souboru pro ukládání geolokačních dat mezi GPS zařízeními a aplikacemi. Byl vytvořen, aby poskytl standardní formát pro komunikaci s geolokačními daty mezi různými zařízeními. Mezi tři hlavní prvky GPX souboru patří:

- **Body (waypoints)** - jednotlivé body na mapě s přesným umístěním. Často se používají k označení bodů zájmu, jako jsou parkoviště, kempy, vrcholové značky nebo jiná místa. Každý bod může obsahovat informace, jako je zeměpisná šířka, délka, nadmořská výška, čas a další metadata data (např. jméno bodu, typ aktivity a poznámky uživatele). Každý bod může být označený specifickým symbolem ze sady ikon, pro lepší orientaci uživatele na mapě.

- **Cesta (route)** je sérií bodů propojených na mapě, které představují plán uživatele. Takto naplánovaná cesta může být uložena a načtena do GPS zařízení pro pozdější navigaci. Cesty často používají nadšenci pro outdoorové aktivity, jako jsou turistika, cyklistika a běh, k plánování svých výletů nebo k navigaci v neznámém terénu. Cesty mohou být rovněž použity při plánování jízdy na kole nebo autem. Obsahují informace, jako je jméno cesty, typ aktivity a body, které tvoří naplánovanou cestu.

- **Trasa (track)** je řadou propojených bodů zaznamenaných GPS zařízením. Na rozdíl od naplánované cesty jde již o uskutečněnou trasu. Zaznamenává GPX data (zeměpisná šířka, délka, nadmořská výška), pohybové aktivity osoby (túra, jízda na kole, běh a pod.) a poskytuje další užitečné informace (ujetá vzdálenost, celkové převýšení, průměrná rychlost, čas a poznámky uživatele). Trasa se dále dělí na segmenty (tracks segments), které slouží k rozdělení trasy na více sekcí. Nejčastěji se používají, když GPS zařízení na určitou dobu přestane zaznamenávat data, například při přestávce během aktivity. V těchto případech je trasa rozdělena do několika segmentů, přičemž každý segment představuje souvislou část cesty, kterou osoba absolvovala.

Toto umožňuje přesnější analýzu činnosti, jako je například ujetá vzdálenost, celkové stoupání a klesání a průměrná rychlost.

3.2 Angular Material

Angular Material [15] je knihovna UI, která poskytuje sadu předpřipravených UI komponent pro framework Angular. Je vytvořena a udržována týmem Angularu a je navržena tak, aby pomohla vývojářům rychle vytvářet responzivní, mobilní webové aplikace s konzistentním vzhledem a chováním.

Angular Material poskytuje širokou škálu UI komponent, včetně tlačítek, menu, formulářů, dialogových oken a dalších. Tyto komponenty jsou navrženy s ohledem na principy Material Design, což je designový jazyk vytvořený společností Google, který umožňuje vytvořit moderní a intuitivní uživatelské rozhraní.

3.3 Leaflet

Leaflet [16] je open-source JavaScriptová knihovna, která umožňuje vykreslit interaktivní mapy na webových stránkách. Je flexibilní, snadno použitelný a lehce přizpůsobitelný, což ho činí ideální volbou pro širokou škálu aplikací pro webové mapování. Je podporován ve všech moderních webových prohlížečích.

Leaflet pracuje s různými vrstvami map. Zobrazuje základní vrstvy map (např. dopravní, turistické a satelitní mapy) od oblíbených poskytovatelů, jako OpenStreetMap a Mapbox, a zároveň dokáže tyto vrstvy překrývat značkami a vyskakovacími okny. Knihovna podporuje zobrazení vektorových dat (GeoJSON, TopoJSON a KML), což umožňuje do map přidávat vlastní data. Leaflet zahrnuje podporu pro geokódování, routování a další běžné funkce mapování. Výhodou Leafletu je jeho rozšiřitelnost o širokou škálu pluginů, které interaktivní mapy obohatí např. o teplotní mapy, vizualizaci dat a 3D mapování.

Leaflet poskytuje řadu ovládacích prvků pro interakci s mapami umožňujícími zoom a posun, změnu měřítka, výběr vrstev, *click* a *hover* události. Pomocí Leafletu lze snadno přidávat vlastní interaktivitu do map, jako je například zvýrazňování prvků na mapě při kliknutí nebo zobrazování dodatečných informací v pop-upu.

Kromě mapování Leaflet podporuje přidávání markerů (bodů) a dalších typů překryvů do map. Markerům je možné upravit ikonu a přidat pop-upy. Skupiny markerů je možné seskupit do markerových clusterů za účelem zlepšení výkonu při zobrazování na mapě. Další typy překryvů, jako jsou polylines a polygons, slouží k vykreslení úseček a tvarů na mapě.

Do Leafletu je také možné přidat vlastní pluginy a integrace, které rozšíří možnosti dané knihovny.

3.4 OpenStreetMap API

OpenStreetMap [17] API je volně stažitelná a editovatelná API poskytující mapy světa. Umožňuje vývojářům vkládat mapy do webových, mobilních a desktopových aplikací a dále je editovat. OpenStreetMap nabízí několik API:

- XAPI - poskytuje přístup k celé databázi OpenStreetMap,
- Overpass API - umožňuje efektivnější způsob dotazování na specifická data z databáze,
- Nominatim API - nabízí služby geokódování a reverzního geokódování (konverze adres a názvů míst na souřadnice zeměpisné šířky a délky a naopak),
- Static Maps API - obsahuje statické obrázky map.

OpenStreetMap také poskytuje nástroje pro editaci (jednoduchý způsob úprav - iD editor, pokročilé editační funkce JOSM editor).

Data OpenStreetMap jsou neustále vylepšována a aktualizována globální komunitou, čímž poskytují bohatý a rozmanitý zdroj geolokačních dat.

3.5 Mapbox API

Mapbox API [18] poskytuje komplexní platformu pro vytváření vlastních online map s širokým spektrem funkcí a nástrojů, sloužící pro potřeby vývojářů, datových vědců a odborníků na GIS(geographic information system).

Mapbox nabízí několik API:

- Maps API - poskytuje přístup k mapám Mapbox a umožňuje přidávat vlastní styly a značky do svých map,
- Geocoding API - konvertuje adresy a jména míst na souřadnice zeměpisné šířky a délky,
- Directions API - generuje trasy pro jízdu autem, na kole a pěší cesty,
- Static Images API - obsahuje statické obrázky map,
- Datasets API - umožňuje práci s velkým množstvím geoprostorových dat
- Tilesets API - umožňuje přidat vlastní data do mapy.

Kromě API Mapbox poskytuje řadu nástrojů pro přizpůsobování map a přidávání interaktivity. Mapbox GL JS je JavaScriptová knihovna pro tvorbu vlastních map a Mapbox Studio slouží pro návrh vlastních stylů map. Dále Mapbox nabízí mnoho pluginů pro práci s populárními mapovacími frameworky, jako je Leaflet a OpenLayers.

3.5.1 Implementace Directions API do navrhované aplikace

Directions API [19] bylo použito pro vyhledávání cesty mezi zadanými body.

```
let url = 'https://api.mapbox.com/directions/v5/mapbox/' + this.routingSelect + '/
    ' + coordinatesStr + '?geometries=geojson&access_token=pk.
    eyJ1IjoiZ3B4YXBwIiwiYSI6ImNsYWgxdTFybzA3YmszcW1xbzlsaWdwNWQifQ.
    rNa3brIV1D9eor_7gzjRmQ'
```

Listing 3.1: Příklad použití Directions API

V proměnné *routingSelect* je uložena informace o způsobu dopravy (auto, kolo nebo chůze). Proměnná *coordinatesStr* obsahuje souřadnice bodů, které se vkládají jako řetězec longitude,latitude oddělené středníkem. API je možné použít až po registraci a vytvoření klíče, který je vložen na konec odkazu. Výstupem z API je objekt ve formátu JSON. Tento objekt obsahuje informace jak o vzdálenosti, tak i o výchozím bodu, koncovém bodu a bodech mezi nimi.

3.6 Leaflet canvas overlay

Leaflet canvas overlay [20] je JavaScriptová knihovna, která umožňuje kreslit vlastní grafiku, animace nebo jiný dynamický obsah přímo na mapu v Leafletu. Tato vrstva je založena na plátně obsaženém v HTML5, a poskytuje flexibilní způsob tvorby interaktivních animovaných grafik v prohlížeči.

S touto knihovnou je možné vytvářet vlastní vizualizace mapy, které nejsou možné se standardními vrstvami v Leafletu, jako jsou například heatmapy, složité vektorové grafiky nebo animace dat v reálném čase.

Pro vytvoření kreslící vrstvy na plátno je třeba definovat vlastní Leaflet vrstvu a implementovat potřebné metody pro kreslení a aktualizaci obsahu plátna. Příklad inicializace plátna a následné aktualizace, kde aktualizace je prováděna pokaždé, když je posunuta myš:

```
function drawingOnCanvas(o: any) {
        console.log(o);
        lng_min = o.bounds._southWest.lng
        lng_max = o.bounds._northEast.lng
        lat_min = o.bounds._southWest.lat
        lat_max = o.bounds._northEast.lat
        width = o.size.x
        height = o.size.y
        canvas = o.canvas;
        var ctx = o.canvas.getContext('2d');
        ctx.clearRect(0, 0, o.canvas.width, o.canvas.height);
        ctx.fillStyle = "rgba(255,116,0, 0.2)";
```

```
        ctx.beginPath();
    };
```

Listing 3.2: Inicializace HTML canvas

```
this.mymap.addEventListener('mousemove', (y:any) => {
          let p = (y as any).layerPoint;
          var ctx = canvas.getContext('2d');
          ctx.fillStyle = "rgba(255,116,0, 0.2)";
          ctx.beginPath();
          ctx.setLineDash([5, 15]);
          ctx.moveTo( (e.latlng.lng - lng_min) * width / (lng_max - lng_min), (
             height - ((e.latlng.lat - lat_min) * height / (lat_max - lat_min))));
          ctx.lineTo(y.containerPoint.x, y.containerPoint.y);
          ctx.clearRect(0, 0, canvas.width, canvas.height);
          ctx.stroke();
    });
```

Listing 3.3: EventListener na pohyb myši s přepočtem geolokačních souřadnic na souřadnice canvasu

3.7 Leaflet.contextmenu

Leaflet.contextmenu [21] je plugin pro JavaScriptovou knihovnu Leaflet, který umožňuje přidat kontextové menu pro libovolné prvky na mapě. Mezi tyto prvky se řadí body na mapě (waypoints),trasy(tracks) a cesty(routes), stejně jako další prvky, jako jsou kruhy, obdélníky a obrázky. Kontextové menu se otevírá stiskem pravého tlačítkem myši. Podle toho, na který prvek mapy bylo kliknuto, kontextové menu nabídne seznam funkcí.

3.7.1 Implementace Leaflet.contextmenu do navrhované aplikace

Při vytvoření mapy pomocí knihovny Leaflet se specifikuje základní kontextové menu. Ukázka vytvoření základního kontextového menu, které se zobrazí při kliknutí na jakoukoli část mapy i na body, trasy nebo cesty.

```
    this.mymap = L.map('map', {
      contextmenu: true,
      contextmenuWidth: 180,
       contextmenuItems: [
      { text: 'Add Waypoint',
      callback: (e:any) => { this.addWtpOnRightClick(e);},
```

```
    icon: '../../assets/images/baseline_place_black_24dp.png',},
    {separator: true},
    { text: 'Add Track',
     callback: (e:any) => { this.onValChangeTrk(true); this.trkArr.push([e.
         latlng.lat, e.latlng.lng])},
    icon: '../../assets/images/baseline_timeline_black_24dp.png'},
    { text: 'Add TrackAuto',
     callback: (e:any) => { this.onValChangeTrkAuto(true); this.trkArr.push([e.
         latlng.lat, e.latlng.lng])},
    icon: '../../assets/images/baseline_find_replace_black_24dp.png'},
    {separator: true},
    { text: 'Add Route',
     callback: (e:any) => { this.onValChangeRte(true); this.rteArr.push([e.
         latlng.lat, e.latlng.lng])},
    icon: '../../assets/images/baseline_route_black_24dp.png'},
    {separator: true},
    { text: 'Exit drawing',
     callback: (e:any) => { this.exitDrawing(); },
    icon: '../../assets/images/baseline_close_black_24dp.png'},

  ],
    editable: true,
    center: [50.177, 14.365],
    zoom: 8,
    layers: this.normMap
  });
```

Listing 3.4: Vytvoření mapy a specifikace základního kontextového menu

Dále je také možné přidat kontextové menu pro prvky na mapě. Funguje to obdobně jako se základním menu. Při tvorbě prvků se povolí kontextové menu a specifikují se možnosti v něm. Při kliknutí na takto vytvořený prvek se zobrazuje vždy základní kontextové menu rozšířené o možnosti definované pro daný prvek.

```
let marker = L.marker([this.lat,this.lon],
        {
        icon: this.iconWtpDef,
        //@ts-ignore
        contextmenu: true,
        contextmenuWidth: 180,
```

```
        contextmenuItems: [
        { text: 'Delete Waypoint',
        callback: (e:any) => { this.delWtpByClick(e); }, index: 1,
        icon: '../../assets/images/baseline_close_black_24dp.png'},
        { text: 'Move Waypoint',
        callback: (e:any) => { this.moveWtp(e); }, index: 2,
        icon: '../../assets/images/baseline_open_with_black_24dp.png'},
        { text: 'Copy of Waypoint',
        callback: (e:any) => { this.copyWaypoint(e); }, index: 3,
        icon: '../../assets/images/content_copy_FILL_24pd.png'},
        { text: 'Move to another GPX',
        callback: (e:any) => { this.moveToGpx(newObj, 'wtp'); }, index: 4,
        icon: '../../assets/images/baseline_move_black_24dp.png'}

      ]
      }
      ).on('dblclick', (e:any)=> {this.loadGPXService.editObjByDblClick({obj:e.
          sourceTarget.myData, type:"wtp"})}).addTo(this.mymap);
```

Listing 3.5: Přidání kontextového menu při vytvoření bodu na mapě

3.8 IndexedDB API

IndexedDB [4] je API databáze, která umožňuje webovým aplikacím ukládat a načítat data lokálně. Poskytuje způsob, jak spravovat strukturovaná data, jako jsou páry klíč-hodnota pro aplikace, které potřebují pracovat offline nebo s velkými objemy dat.

Příklad vytvoření IndexedDB:

```
var request = window.indexedDB.open("db_gpx", 1);
request.onupgradeneeded = function (event) {
    var db = event.target.result;
    var objectStore = db.createObjectStore("db_gpx_object", { keyPath: "id",
        autoIncrement:true })
}
```

Listing 3.6: Ukázka vytvoření databáze IndexedDB

V tomto příkladu je otevřeno spojení s novou nebo existující databází pojmenovanou 'db_gpx' s číslem verze 1. V callbacku je vytvořeno objektové úložiště pojmenované 'db_gpx_object'. Inde-

xedDB použije vlastnost 'id' každého objektu jako unikátní identifikátor tohoto objektu v úložišti. Také je nastaven autoIncrement, který zajistí, že klíče se budou generovat automaticky.

V dalším příkladu je vidět metoda přidávání záznamu do databáze.

```
const request = await db.transaction([this.storage], "readwrite").objectStore(this
    .storage).put(db_object);
```

Listing 3.7: Příklad vytvoření nového záznamu do databáze IndexedDB

Pro přidání je možné použít funkci *add()* nebo *put()*. Rozdíl je v tom, že funkce *add* vždy přidá nový záznam, zatímco funkce *put* záznam vytvoří pouze tehdy, když není obsažen v databázi, v opačném případě záznam zaktualizuje. Pro ověření, zda je záznam už v databázi, se musí do funkce *put* přidat parametr klíče.

Pro získání dat z databáze se používá obdobná komunikace jako u předešlých funkcí. Funkce pro získání dat jsou *getAll()* nebo *get()*. Funkce *get* musí mít parametr, který určí klíč v databázi. Pro získání všech dat z databáze se použije funkce *getAll*.

Kapitola 4

Návrh a implementace webové aplikace

V této kapitole je popsána webová aplikace pro správu GPX souborů. Cílem je seznámení s vnitřní strukturou aplikace, a dále s komponenty tvořící aplikaci a se systémem komunikace mezi komponentami.

4.1 Struktura webové aplikace

- app - kořenová aplikace
  - load_gpx - zajišťuje nahrání souboru, komunikaci s databází, parsování souboru, komunikaci mezi komponenty map a tree
    - gpx-parser-builder - parsuje gpx soubor
  - map - komponenta se stará o vykreslování a editaci mapy
    - L.CanvasOverlay - vykreslení čárkované čáry před mapou při vytváření nového tracku nebo routy
    - leaflet-contextmenu - menu pro editaci spouštěné pravým tlačítkem myši
    - leaflet.heightgraph - vizualizace nadmořské výšky ve vybraném tracku nebo routě
    - move-to-gpx-dialog - dialog se zobrazí při přesouvání tracku nebo routy do jiného gpx souboru
  - treePrint - komponenta vykresluje stromovou strukturu a edituje data uvnitř této struktury
    - menu - nabízí možnosti editace skrze dialogy
      - edit-menu - nabízí možnosti editace jednotlivých částí gpx souboru jako jsou metadata, tracky, routy a waypointy
      - tree-menu - nabízí možnosti editace gpx souboru(vykreslení, smazání, přejmenování a uložení)

    - dialogy - dialogy pro editaci
        - file-rename - dialogy pro změnu názvu
        - metadata - dialogy pro editaci metadat
        - waypoint - dialogy pro editaci waypointu
        - track - route - dialogy pro editaci tracku nebo routy
        - track segment - dialogy pro editaci track segmentu
    - struktura tree - zajišťuje editaci datové struktury

4.2 Popis webové aplikace

4.2.1 Základní struktura

Po načtení aplikace se inicializují komponenty *app*, *map* a *tree*. Každá z těchto komponent zajišťuje konkrétní funkce aplikace.

Komponenta *app* se stará o vytvoření připojení k databázi a načtení dat z databáze do datové struktury. Po načtení se spustí metoda ze servisní třídy load-gpx.service.ts, která začne načítat data z databáze IndexedDb. Pokud není databáze na lokálním úložišti ještě vytvořena, program vytvoří novou databázi. V databázi jsou data uložena v objektu, který obsahuje informaci o primárním klíči, a soubor ve formátu GPX. Při postupném nahrání celé databáze se parsujou GPX soubory na objekty. Pro tento objekt je použito datové rozhraní *GpxNode*, které je vytvořeno tak, aby se s objektem nemuselo manipulovat při vykreslování do stromové struktury stylované pomocí knihovny Angular Material. Pro efektivnější a rychlejší načítání jsou soubory parsovány pouze částečně. Při parsování jsou získána pouze data nutná k tomu, aby šel vykreslit strom v uživatelském rozhraní, ale nejsou parsována data týkající se třeba jednotlivých souřadnic bodů, cest nebo tras. Při používání aplikace se soubory doparsují v případě nutnosti použití pro zobrazení nebo editaci. Po naparsování souboru do objektu je poslán do komponenty *tree* skrze BehaviorSubject z javaskriptové knihovny RxJS.

Komponenta *tree* při inicializaci zajišťuje uložení naparsovaných objektů do datové struktury TREE_DATA a následné vykreslení těchto dat do UI. První krok, který se stane při načtení aplikace je, že komponenta naváže spojení tak, že odebírá BehaviorSubject propojení z ostatních komponent. Dále čeká, až budou poslány naparsované objekty. Tyto objekty používají datové rozhraní *GpxNode*.

```
export interface GpxNode {
    name: string;
    id?: any;
    obj?: any;
    type?: string;
    children?: GpxNode[];
```

```
    isShown?: boolean;
    selected?: boolean;
    parsed?: boolean ;
}
```

Listing 4.1: Datové rozhraní GpxNode

V tomto rozhraní se nacházejí informace o unikátním *id*, které propojuje objekt s databází pro případy uložení nebo odstranění. Pole objektů *children* je nutné pro použití na vykreslení UI. Angular material komponenta *Tree* pomocí této struktury vykreslí zanořený seznam. V poli *children* jsou data ohledně jednotlivých podkategorií, jako Metadata, Waypoints, Routes, Tracks. Každá z těchto podkategorií může obsahovat další pole *children* pro větvení například na jednotlivé body na mapě. V proměnné *obj* jsou uloženy konkrétní části naparsovaných dat. Např. v konkrétní cestě v této proměnné budou uloženy informace o jednotlivých souřadnicích.

Tyto naparsované objekty jsou přidány do datové struktury TREE_DATA. Tato struktura je pole objektů, kde se v průběhu chodu aplikace provádějí veškeré změny v aplikaci. Toto pole je následně vykresleno na UI pomocí již zmiňované knihovny Angular material, která obsahuje komponentu *Tree*.

Komponenta *map* podobně jako *tree* inicializuje spojení pomocí BehaviorSubject s komponenty. Hlavní částí inicializace je nastavení mapy vykreslené pomocí knihovny Leaflet. Při nastavení mapy se musí specifikovat vrstvy, které má mapa obsahovat a také poskytovatel map. Rovněž se nastavuje základní kontextové menu (ukázka výše). V inicializaci *map* komponenty se specifikuje metoda, která se volá při kliknutí na mapu.

4.2.2 Nahrání souboru

Nový GPX soubor se přidává pomocí tlačítka. Po kliknutí se otevře okno, kde si uživatel vybere, který soubor by chtěl nahrát. Nahrávání souboru je pomocí HTML input. O nahrání se stará komponenta *app* a po nahrání souboru je volána metoda servisní třídy load-gpx.service.ts, kde se předá načtený soubor. Metoda vytvoří nový záznam v databázi IndexedDb, kde je uložen celý soubor ve formátu GPX. Záznamy v databázi mají svůj unikátní klíč id a obsah souboru. Po uložení do databáze je soubor parsován na objekt s datovým rozhraním GpxNode. Poté je soubor parsován. Rozdílem oproti inicializaci aplikace je, že zde se soubor parsuje už na kompletní objekt. Je to tím, že zde už nehrozí velké čekání, protože je parsován pouze jediný objekt. Objekt je dále poslán do komponenty *tree* skrze BehaviorSubject, kde je za pomoci metody přidán do datové struktury TREE_DATA a je aktualizováno vykreslení stromové struktuře v UI.

4.2.3 Vykreslení GPX souboru

Pro vykreslení GPX souboru si uživatel v UI zvolí z vykresleného seznamu na pravé straně a klikne na název souboru. Po kliknutí se spustí asynchronní metoda komponenty *tree*, kde se ověří, zda je objekt plně parsován. Pokud není, doparsuje se, přičemž metoda čeká na dokončení parsování, než pokračuje dál. Metoda předává objekt celého GPX souboru do komponenty *map*, kde metoda vytvoří jednotlivé prvky mapy. Tyto prvky z knihovny Leaflet obsahují *markers* a *polylines*. Pro každý prvek se také nastavuje Leaflet.contextmenu, kde se navíc každému typu prvku přiřazují specifické operace, které lze nad prvkem realizovat. *Markers* jsou použity pro vykreslení bodů na mapě (waypoints), kde každý bod musí obsahovat souřadnice a také je možné, aby obsahoval informaci o tom, jakou ikonou se zobrazí. Název ikony je uložen v souboru GPX, podle názvu je dosazena ikona uložená v aplikaci. Pro ikony je použit seznam od společnosti Garmin. Pokud bod nemá specifikovanou ikonu, je vykreslen základní ikonou.

```
let marker = L.marker([node.children![1]?.children![i].obj.$.lat, node.children
    ![1]?.children![i].obj.$.lon],
    {
        icon: iconWtp,
        contextmenu: true,
        contextmenuWidth: 180,
        contextmenuItems: [
        { text: 'Delete Waypoint',
        callback: (e:any) => { this.delWtpByClick(e); }, index: 1,
        icon: '../../assets/images/baseline_close_black_24dp.png'}
        ]
    }
    ).on('dblclick', (e:any)=> {this.loadGPXService.editObjByDblClick({obj:e.
        sourceTarget.myData, type:"wtp"})}).addTo(this.mymap);
    (marker as any)['myData'] = node.children![1]?.children![i].obj;
```

Listing 4.2: Ukázka vytvoření bodu v knihovně Leaflet

Polylines vykreslují trasy(tracks) a cesty(routes). Trasy i cesty musí obsahovat pole souřadnic. Toto pole musí být nejdříve vytvořeno z objektu, protože struktura dat souřadnic v objektu je rozdílná. Je možné přidat volitelnou informaci o barvě, kterou budou trasy nebo cesty vykresleny. Barva je uložena v souboru GPX jako rozšíření trasy nebo cesty. Po kompletním zobrazení se mapa zaostří na vykreslené prvky. Zaostření je docíleno tak, že v průběhu přidávání prvku se všechny souřadnice ukládají do pole, které je pak předáno funkci *fitBounds* z Leafletu, ta zobrazí veškeré body, tak aby se všechny zobrazily, ale nebyla mapa zbytečně oddálená.

4.2.4 Přidání prvků na mapu

Na mapu lze přidat několik prvků, jsou to waypoints, tracks a routes. Přidání probíhá skrze kontextové menu. **Bod na mapě**(waypoint) se umístí tam, kde bylo stisknuto pravé tlačítko a byla vybrána akce Add Waypoint. Do metody komponenty *map* vstupují informace, na jakých souřadnicích bylo tlačítko stisknuto. Tyto souřadnice se pošlou do komponenty *tree*, kde je vytvořen nový objekt pro bod na mapě a je vložen do konkrétního GPX objektu, uloženém ve struktuře TREE_DATA. Objekt s bodem je zpět poslán do komponenty *map* kde se pomocí něho přidá nový bod na mapu.

Uživatel **trasu** vytvoří tak, že postupně přidává body na mapu. Tyto body jsou propojené úsečkou mezi sebou. Přidání trasy probíhá po zvolení v kontextovém menu možnosti Add Track. Tato akce aktivuje metodu, kde je inicializována knihovna L.CanvasOverlay. Knihovna vytvoří před mapou HTML canvas a ten je použit pro vykreslování čárkované čáry mezi posledním zvoleným bodem a kurzorem myši. Ukázka implementace je výše v textu. V průběhu vytváření zvolí uživatel libovolný počet bodů a když se rozhodne přestat vytvářet trasu, zvolí v kontextovém menu možnost Exit drawing. Souřadnice bodů v průběhu vytváření jsou ukládány do pole. Při každém přidání nového bodu se překreslí trasa na mapě o nově přidaný bod a na konci se toto pole se všemi souřadnicemi pošle do komponenty *tree*, kde je vytvořen objekt a přidán na správné místo do struktury TREE_DATA. Nově vytvořený objekt trasy se přepošle zpět do komponenty *map*, kde je následně vytvořena a vizualizována nová *polyline*.

Na mapu lze přidat i prvek **cesty** a **automaticky plánovaná trasa**. Tyto prvky se principem přidání ze strany uživatele velice podobají trase. Hlavním rozdílem pro automaticky plánovanou trasu je, že propojení mezi body, které vybral uživatel, není úsečkou, ale používá se API Mapbox pro nalezení trasy. V aplikaci je tedy oproti trase přidána funkce na komunikaci s API, která vrací body, přes které trasa povede.

Cesta se od automaticky plánované trasy liší tím, že uživatel si může vybrat, zda se všechny cesty budou zobrazovat tím způsobem, že budou body spojené mezi sebou pouze úsečkami nebo bude mezi body vyhledána trasa. Při zakládání cesty tedy záleží na tom, jakým způsobem se cesty právě vykreslují. Pro změnu vykreslování cest má uživatel k dispozici tři možnosti. Vybrat si může z menu umístěného na mapě.

Každé nově vytvořené entitě je přidána do Leafletu proměnná, která obsahuje celý objekt dané entity. To znamená, že po vykreslení a následném vybrání prvku je okamžitě k dispozici objekt tohoto prvku. Tento objekt je pak použit při editaci k identifikaci entity, uložené ve struktuře TREE_DATA.

4.2.5 Editace GPX souboru

Velká část možností editace souboru GPX je v kontextovém menu. Tyto funkce se převážně týkají bodů na mapě, tras a cest.

- Funkce **Delete** umožňuje vymazat vybraný prvek z mapy. Do funkce vstupuje objekt prvku, který uživatel zvolil a je poslán z komponenty *map* do komponenty *tree*. Zde je pomocí objektu a informace, které GPX soubory jsou vykresleny, vyhledán správný soubor ve struktuře TREE_DATA. Následně je objekt ze souboru odstraněn.

- **Continue Track/Route** - tato funkce umožňuje uživateli pokračovat v již vytvořené cestě nebo trase. Uživatel tuto funkci aktivuje a pokračuje obdobně jako při vytváření nového prvku mapy. Pokračovat je možné v obou směrech a to tak, že program vyhodnocuje, kterému bodu uživatel klikl blíž. Do funkce vstupuje objekt editovaného prvku z mapy. Z objektu se uloží souřadnice prvního a posledního bodu do proměnných. Tyto souřadnice jsou použity k vyhodnocení, kterým směrem bude uživatel prvek rozšiřovat.

```
let distanceFirst = (newPoint[0] - this.startPoint[0])** 2 + (newPoint
    [1] - this.startPoint[1])** 2
let distanceLast = (newPoint[0] - this.endPoint[0])** 2 + (newPoint[1]
     - this.endPoint[1])** 2
```

Listing 4.3: Ukázka výpočtu vzdálenosti nově vytvořeného bodu s prvním a posledním bodem trasy nebo cesty

  Po výpočtu se podle vzdálenosti určí, za který bod bude uživatel přidávat další body. Nově přidané body se ukládají do pole a až uživatel skončí, jsou přiřazeny do pole souřadnic celého prvku a odeslány do komponenty *tree* k aktualizaci hlavní struktury TREE_DATA.

- **Add Track/Route point** je funkce pro přidání bodu na trase nebo cestě. Tento bod je přidán na místo na úsečce prvku, kam uživatel klikl. Do funkce jsou předány souřadnice nového bodu a jsou přidány do pole souřadnic prvku.

- **Split Track/Route** - tato funkce rozdělí trasu nebo cestu na dvě entity téhož typu. Uživatel zvolí kde se prvek rozdělí. Na tomto místě je vytvořen nový bod jako u funkce *Add Track/Route point* a jeho souřadnice s objektem prvku jsou poslány do komponenty *tree*. Zde je pole původního objektu zkráceno. Je vytvořen nový prvek se začátkem v nově přidaném bodu a zbytkem pole bodů původního prvku.

- Funkce **Show/Hide Track/Route points** vykreslí ikonu bodu nad každým bodem trasy nebo routy. To umožňuje manipulovat s těmito body individuálně, tedy je možné je mazat, posouvat a kopírovat. Při opakovaném použití se ikony bodů skryjí.

- **Reverse Track/Route** prohodí pořadí bodů trasy nebo cesty. Tato funkce prohodí pořadí pole bodů daného prvku. Změny jsou aplikovány podobně jako u funkcí pro editace výše.

- **Copy of Waypoint/Track/Route** zkopíruje objekt z uživatelem vybraného prvku a uloží jej jako nový prvek do hlavní struktury TREE_DATA. Metoda uložení je podobná jako u funkcí výše.

- **Move Waypoint/Track/Track segment/Route to other GPX file** umožňuje kopírovat prvky mezi soubory GPX. Po výběru prvku se uživateli zobrazí seznam souborů GPX a vybere do kterého by chtěl nakopírovat prvek. Poté se objekt prvku pošle do komponenty *tree* s informací o indexu GPX souboru, do kterého bude prvek přidán.

- **Show graph of Track/Route** je funkce pro zobrazení výškového grafu ze souřadnic prvku. Nejdříve je získáno pole souřadnic z objektu prvku. V komponentě *map* jsou souřadnice formátovány do formátu, se kterým pracuje *Open-Elevation API*. (Ukázka práce s API je výše v textu.) Data jsou po návratu z api formátována do pole. V poli se data ukládají tímto způsobem *[longitude, latitude, elevation]*. Toto pole je vloženo do nově vytvořeného *geoJsonu*. Je to z toho důvodu, že pro vizualizaci je použita knihovna *Leaflet.Heightgraph*, která umožňuje vytvořit graf výškového profilu trasy a přijímá jako vstup *geoJson*. Posledním krokem je pak přidání vytvořeného grafu do mapy.

```
 const FeatureCollections = [{
"type": "FeatureCollection",
"features": [{
    "type": "Feature",
    "geometry": {
        "type": "LineString",
        "coordinates": geoJson
    },
   "properties": {
        "attributeType": 3
    }
}],
"properties": {
    "Creator": "OpenRouteService.org",
    "records": 1,
    "summary": "Steepness"
}
}];
```

Listing 4.4: Ukázka formátování dat do geoJsonu

- Funkce **Move Waypoint** umožňuje uživateli kliknout na libovolný bod a změnit jeho pozici na mapě. Po spuštění funkce se vytvoří *HTML canvas*, na kterém se na pozici myši vykresluje základní ikona bodu. Uživatel poté vybere novou pozici bodu. Z místa, kam uživatel umístil bod, jsou získány souřadnice, které jsou použity pro aktualizaci souřadnic bodu. Tato aktualizace probíhá obdobně jako u prvků zmíněných výše v této kapitole.
- **Transfer Route to Track** je funkce pro přeměnu cesty na trasu. Původní cesta je zachována a je vytvořena nová trasa z bodů původní cesty. Uživatel si také může vybrat typ dopravy (auto, kolo, chůze) a to stejně jako při změně vykreslení cesty. První krok vytváření trasy je vygenerování pole souřadnic za pomoci *MapBox API*. Vstupem API jsou původní body cesty. Výstupem API je pole souřadnic, ze kterého je vytvořena cesta stejným způsobem, jak bylo zmíněno na začátku kapitoly.

Po dokončení těchto funkcí jsou data uložena do datové struktury TREE_DATA. Pro zobrazení změn v mapě se editovaný objekt a všechny další objekty, které jsou na mapě vykreslené pošlou do komponenty *map*. V této komponentě se postupně překreslí všechny body, trasy a cesty.

Kapitola 5

Uživatelské testování

V této kapitole je popsáno, jakým způsobem bylo uživatelské testování provedeno. Velkým přínosem uživatelského testování je odhalení, zda je UI správně navrženo a je pochopitelné pro uživatele, kteří se s ním setkávají poprvé.

Testování se zúčastnilo pět uživatelů, kterým byly předloženy stejné úkoly rozdělené do tří scénářů. Scénáře jsou navrženy tak, aby byla otestována téměř celá aplikace. Průběh, jak uživatelé mohli postupovat, je popsán v příloze této práce. Postup uživatelů se může mírně lišit od vzorového postupu, protože v aplikaci jde místy dosáhnout stejného výsledku více způsoby. Pokud se však postup uživatele výrazně liší, je nutné vyhodnotit z jakého důvodu, s čím měl uživatel potíže a jakým způsobem by šla konkrétní funkce vyřešit lépe.

5.1 Scénář 1

V tomto scénáři má uživatel za úkol založit nový GPX soubor a má do tohoto souboru vytvořit novou cestu(route) z libovolného místo do Ostravy. Poté má změnit barvu této cesty a prodloužit ji až do Opavy. Dalším krokem je přejmenování cesty na libovolný název. Posledním úkolem je z cesty vytvořit trasu(track) a změnit zobrazování veškerých cest tak, aby se přepočítaly pro jízdu na kole. Tyto změny pak uživatel uloží.

5.1.1 Výsledek scénáře 1

Tento scénář dokončili úspěšně všichni uživatelé. Většině uživatelů dělalo problém pochopit, jak by měli označit GPX soubor, aby do něj mohli přidávat nové prvky. To bylo částečně způsobené tím, že v aplikaci byl nahrán pouze jeden GPX soubor a uživatelé tak nevěděli, proč by si měli soubor zvolit. Uživatelům také trvalo, než našli možnost změnit vykreslování cest na mapě.

5.2 Scénář 2

Uživatel si v tomto scénáři zvolí soubor, s kterým chce dále pracovat. Do tohoto souboru vloží bod na mapě (waypoint). Dále bod posune na jinou pozici, než bod původně vytvořil a změní ikonu bodu zobrazujícího se na mapě. Dále vytvoří trasu a po vytvoření posune jeden z bodů na jinou pozici. Poté rozdělí trasu na dva segmenty a změní barvu jednoho z nich. Následuje vykreslení výškového grafu. Nakonec uživatel odstraní soubor, na kterém pracoval.

5.2.1 Výsledek scénáře 2

Tento scénář dokončili úspěšně všichni uživatelé. Zpočátku někteří uživatelé hledali nastavení pouze v kontextovém menu na mapě, než přišli na to , že ve stromové struktuře se nachází nastavení prvků. Také uživatelům chvíli trvalo, než zjistili, že po vytvoření trasy se první bod vytvoří na místě, kde bylo kliknuto pravým tlačítkem. Uživatelé rovněž nevěděli, proč se trasa dělí na segmenty a cesta ne. Toto bylo způsobeno do jisté míry neznalostí uživatelů o GPX souborech.

5.3 Scénář 3

Uživatel při tomto scénáři začne importem předpřipraveného souboru z adresáře uloženého na jeho počítači. Soubor přejmenuje a vykreslí jeho obsah na mapu. Vytvoří novou automaticky naváděnou trasu (TrackAuto). Dále odstraní libovolný prvek z mapy a přesune vybranou trasu do jiného GPX souboru. Scénář uživatel ukončí stažením souboru do paměti počítače.

5.3.1 Výsledek scénáře 3

Uživatelé se už díky předchozím scénářům naučili s aplikací pracovat, a tak i tento scénář dokončili úspěšně všichni uživatelé. Pouze u jednoho uživatele nastalo drobné nepochopení pro vykreslování trasy s automatickým naváděním. Způsobilo to pojmenování této volby a to na *Add TrackAuto*.

5.4 Výsledky uživatelského testování

Ze získaných výsledků vyplývá, že uživatelům chvíli trvalo, než dokázali s aplikací bez problémů pracovat. Všechny funkce uživatelům nebyly okamžitě jasné, ale po jejich vyzkoušení pochopili, jak je používat. Ze začátku měli uživatelé problém s tím, že po kliknutí pravým tlačítkem a následném výběru přidání trasy nebo cesty se první bod uložil na místo, kde si menu rozklikli, ale brzy si na tento postup zvykli. Některým uživatelům činilo potíže rozdělení úprav aplikace, konkrétně na vizuální úpravy v mapě a na úpravy nastavení jednotlivých prvků v stromové struktuře. Také uživatelé chvíli přicházeli na to, jak označit soubor pro editaci. Řešení (kliknutí na název souboru

nebo v menu u názvu souboru) je jednoduché, ale při prvním otevření neznámé aplikace to nemusí být okamžitě zřejmé.

5.5 Změny vyplývající z uživatelského testování

Navrhované změny vycházejí z požadavků uživatelů, kteří se zúčastnili testování. Do aplikace bude přidán stručný návod pro snadnější orientaci uživatele při prvním spuštění aplikace. Další úpravou bude oprava názvu funkce na změnu vykreslování cest podle zvoleného typu dopravy na pochopitelnější název. Na horní část hlavní obrazovky přibude tlačítko na ukončeni kreslení.

Kapitola 6

Výkonnostní testování

Tato kapitola je zaměřená na výkonnostní testování aplikace. Při implementaci aplikace bylo zjištěno, že se aplikace značně zpomaluje s přibývajícím počtem souborů v ní uložených. Jednalo se hlavně o načtení aplikace, kde docházelo k parsování veškerých souborů uložených v databázi. Řešením tohoto problému je soubory při načítání aplikace parsovat pouze částečně a teprve až při nutnosti zobrazení konkretní soubor naparsovat celý.

Dalším problémem bylo vykreslování čárkované čáry na mapu při tvorbě cesty nebo trasy. Při vykreslování přímo do Leafletu nastaly potíže se sekáním aplikace, a to z toho důvodu, že se při každém pohybu myši musela překreslit celá mapa. Řešením bylo vytvořit HTML canvas před mapou a kreslit tuto čárkovanou čáru na něj. Tím, že mapa používá jiné souřadnice něž canvas, musejí se tyto souřadnice přepočítávat.

Dále v kapitole je otestován rozdíl mezi částečným a plným parsováním GPX souborů.

6.1 Testování rozdílu mezi částečným a plným parsování GPX souborů

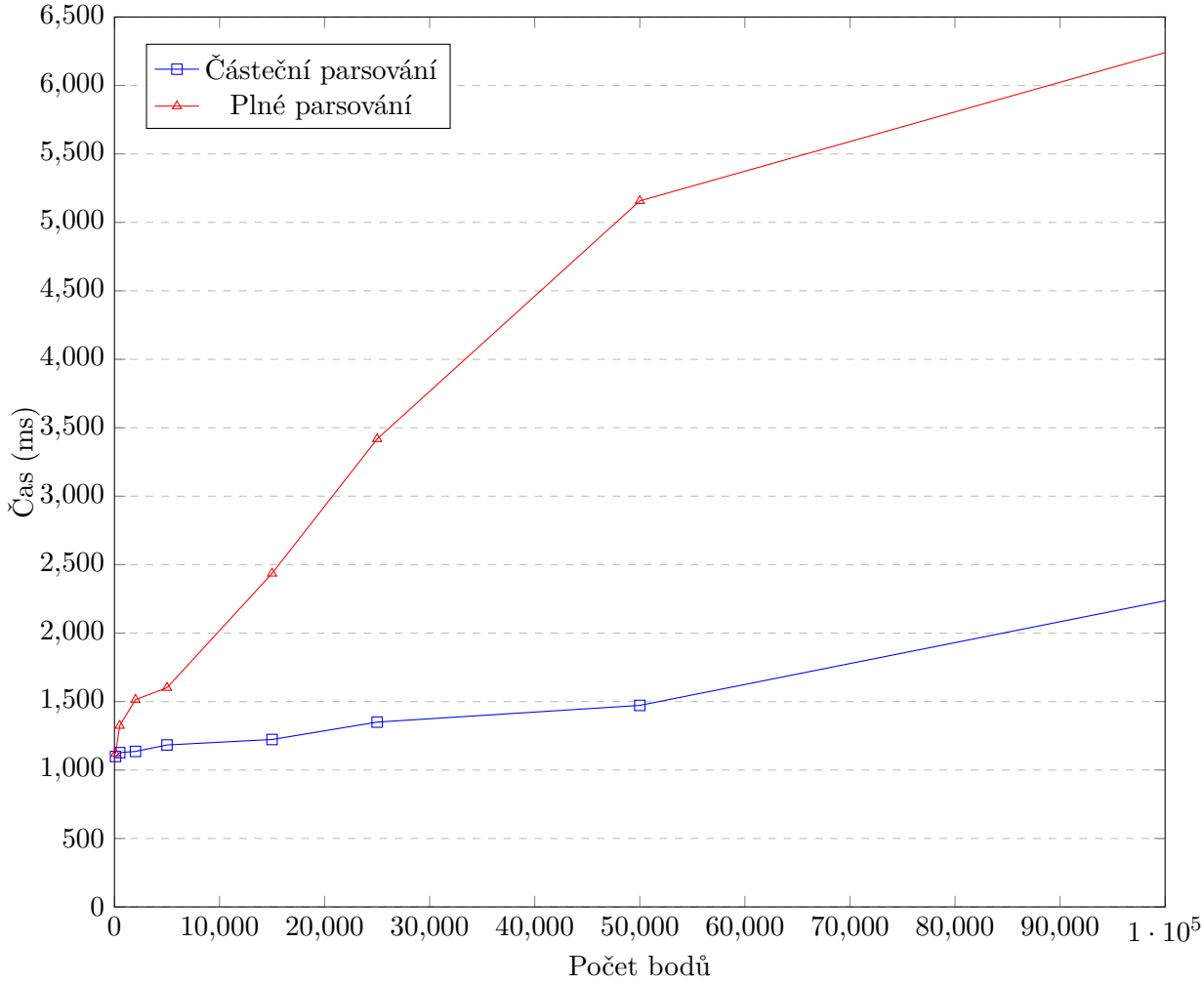
Při testování byly použity vývojářské nástroje v prohlížeči Chrome. Byl měřen celkový čas, za který se aplikace načetla. Měření může ovlivnit výkon počítače, takže všechny testy byly měřeny na stejném stroji. Stolní počítač, na kterém se testy prováděly, má parametry:

- Procesor: čtyř jádrový Intel Core i5-4460 3.20GHz
- Grafická karta: Nvidia GTX 960 2GB VRAM
- RAM paměť: 16 GB DDR3

6.1.1 Test 1

V tomto testu se zjišťovalo, jak moc efektivně si dokážou oba přístupy poradit s jedním souborem. Tento soubor obsahoval jednu trasu, která se postupně zvětšovala. Počet bodů trasy začínal na 100

a končil na 500 000 bodech. Výsledek je zobrazen v grafu. Pro přehlednější vizualizaci je graf omezen na 100 000 bodů a na 6500 ms.



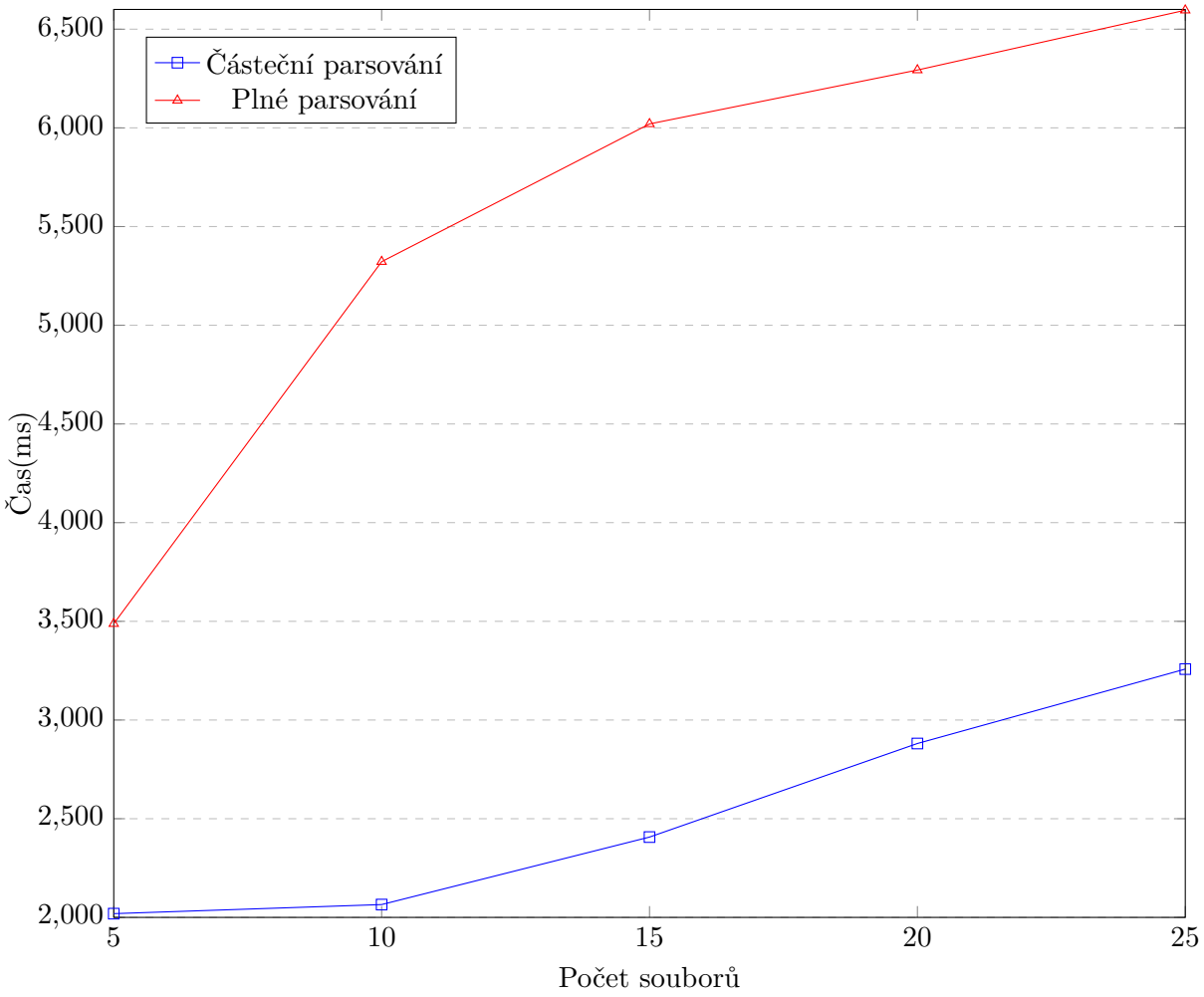
Časové rozdíly narůstaly s přibývajícími body:

- 200 000 bodů: částečné parsování 2 857 ms, plné parsování 10 068 ms
- 500 000 bodů: částečné parsování 5 637 ms, plné parsování 70 470 ms

Částečné parsování je při jakémkoliv počtu bodů efektivnější než plné parsování. Je to způsobené tím, že při částečném parsování se parsuje pouze název souboru a názvy cest, tras a bodů na mapě, zatímco při plném parsování se zpracovává kompletně celý soubor.

6.1.2 Test 2

Ve druhém testu byla zjišťována efektivita obou přístupů parsování při načítání většího počtu souborů. Test byl proveden s opakovaným vložením jednoho souboru, který obsahoval tři body na mapě, jednu cestu o 10 bodech a trasu s 5000 body. Testovalo se v intervalu od 5 až po 25 souborů.



S přibývajícím počtem souborů se aplikace zpomaluje, ale i přesto je částečné parsování rychlejší až o 50% než plné parsování.

Kapitola 7

Závěr

Cílem této bakalářské práce bylo zpracovat webovou aplikaci pro správu a editaci GPX souborů. Tato aplikace byla vytvořena za pomoci frameworku Angular. V práci byly použity knihovny pro vizualizaci a editaci GPX souborů.

V první řadě bylo potřeba se seznámit s možnostmi vizualizace mapy ve webové aplikaci. Po výběru byla vytvořena aplikace a bylo zprovozněno načítání a následné parsování GPX souboru. Následovala implementace vykreslování GPX souborů do mapy a funkcí pro editaci. Pro vytvoření přehledného uživatelského rozhraní byly provedeny některé změny vycházející z uživatelského testování.

V průběhu vytváření této bakalářské práce jsem se seznámil s tvorbou webových aplikací pomocí Angular frameworku, a také jsem získal vědomosti o použití a tvorbě GPX souborů. Nejvíce jsem se naučil o vizualizaci mapy a její editaci, zde jsem měl předtím jen základní znalosti.

Webovou aplikaci by bylo užitečné v budoucnu rozšířit o možnost sdílet vytvořené soubory mezi uživateli. Také by bylo možné vytvořit mobilní aplikaci s uloženými mapami v zařízení telefonu pro offline použití.

Literatura

1. *HTML5 Basics For Everyone Tired Of Reading About Deprecated Code* [online]. [cit. 2023-04-26]. Dostupné z: `https://html.com/html5/`.
2. *Top 10 new features of HTML5* [online]. [cit. 2023-04-26]. Dostupné z: `https://www.geeksforgeeks.org/top-10-new-features-of-html5/`.
3. DUCKETT, Jon. *HTML and CSS: Design and Build Websites*. Wiley, 2011. ISBN 9781118008188.
4. *IndexedDB API* [online]. [cit. 2023-04-26]. Dostupné z: `https://developer.mozilla.org/en-US/docs/Web/API/IndexedDB_API`.
5. *The box model* [online]. [cit. 2023-04-26]. Dostupné z: `https://developer.mozilla.org/en-US/docs/Learn/CSS/Building_blocks/The_box_model`.
6. *A Complete Guide to Flexbox* [online]. [cit. 2023-04-26]. Dostupné z: `https://css-tricks.com/snippets/css/a-guide-to-flexbox/`.
7. *Learn CSS Grid* [online]. [cit. 2023-04-26]. Dostupné z: `https://learncssgrid.com/`.
8. DUCKETT, Jon. *JavaScript and jQuery: Interactive Front-End Web Development*. Wiley, 2014. ISBN 9781118531648.
9. *Who Uses Angular in 2023? 12 Global Websites Built With Angular* [online]. [cit. 2023-04-26]. Dostupné z: `https://www.trio.dev/blog/companies-use-angular`.
10. VICTOR SAVKIN, Jeff Cross. *Essential Angular*. Packt Publishing Ltd, 2017. ISBN 9781788291040.
11. *Angular vs. React in 2023: Side-By-Side Comparison* [online]. [cit. 2023-04-26]. Dostupné z: `https://www.trio.dev/blog/angular-vs-react`.
12. *ReactJS - Hng dn tân th (P1)* [online]. [cit. 2023-04-26]. Dostupné z: `https://viblo.asia/p/reactjs-huong-dan-tan-thu-p1-m68Z0oW9KkG`.
13. *What Is Vue.js? The Pros and Cons of Vue.js in 2023* [online]. [cit. 2023-04-26]. Dostupné z: `https://www.trio.dev/blog/why-use-vue-js`.
14. *GPX* [online]. [cit. 2023-04-26]. Dostupné z: `https://docs.fileformat.com/gis/gpx/`.
15. *Angular Material Tutorial* [online]. [cit. 2023-04-26]. Dostupné z: `https://www.javatpoint.com/angular-material`.

16. *Leaflet API reference* [online]. [cit. 2023-04-26]. Dostupné z: https://leafletjs.com/reference.html.

17. *Develop* [online]. [cit. 2023-04-26]. Dostupné z: https://wiki.openstreetmap.org/wiki/Develop.

18. *Web Services APIs* [online]. [cit. 2023-04-26]. Dostupné z: https://docs.mapbox.com/api/overview/.

19. *Directions API* [online]. [cit. 2023-04-26]. Dostupné z: https://docs.mapbox.com/api/navigation/directions/.

20. *Leaflet Canvas Overlay* [online]. [cit. 2023-04-26]. Dostupné z: https://gist.github.com/sabman/ac90bdfad0971a468fb36736db655554.

21. *Leaflet.contextmenu* [online]. [cit. 2023-04-26]. Dostupné z: https://github.com/aratcliffe/Leaflet.contextmenu.

Příloha A

Řešení po krocích pro scénář 1

1. Kliknutí na tlačítko *New GPX file*.
2. Kliknutí na název nově vytvořeného souboru, aby byl soubor vybrán k editaci.
3. Kliknutí na mapu pravým tlačítkem myši tam, kde bude cesta začínat; vybrání možnosti *Add Route*, přidání bodů kliknutím levého tlačítka myši a ukončení kreslení tlačítkem *Exit drawing*, které je umístěno v menu po kliknutí pravým tlačítkem do mapy.
4. Rozbalení stromové struktury pro daný GPX soubor a vybrání nově přidané routy v záložce *Route*.
5. Kliknutí na tlačítko u cesty pro zobrazení menu a zvolení *Settings*.
6. Kliknutí na políčko s názvem *Line color*, vybrání nové barvy a kliknutí na tlačítko *Save*.
7. Najetí myší na cestu, kliknutí pravým tlačítkem myši, zvolení volby *Continue Route* a přidání nových bodů a ukončení kreslení tlačítkem *Exit drawing*.
8. Najetí myší na cestu, kliknutí pravým tlačítkem myši, zvolení volby *Transfer Route to Track*.
9. V stromové struktuře vybrání cesty, zobrazení nastavení po kliknutí na menu a zvolení *Settings*. Vepsání nového názvu do políčka *Name* a uložení.
10. Kliknutí na volbu *Automatic routing* ve vrchní části mapy a vybrání *Bike*.
11. Ve stromové struktuře kliknutí na menu vedle názvu souboru a vybrání *Save file*.

Příloha B

Řešení po krocích pro scénář 2

1. Kliknutí na název souboru, aby byl vybrán pro editaci.
2. Kliknutí pravým tlačítkem myši do mapy na místo, kde bude bod vytvořen po vybrání možnosti *Add Waypoint*.
3. Kliknutí pravým tlačítkem na bod, zvolení možnosti *Move Waypoint* a kliknutí levým tlačítkem pro vybrání nové pozice bodu.
4. Kliknutí na menu tlačítko vedle názvu bodu ve stromové struktuře a vybrání *Settings*. Zvolení nové ikony ze seznamu *Waypoint icon* a uložení změn.
5. Kliknutí na mapu pravým tlačítkem myši v místě, kde bude trasa začínat, vybrání možnosti *Add Track*, přidání bodů kliknutím levého tlačítka myši a ukončení kreslení tlačítkem *Exit drawing*, které je umístěno v menu po kliknutí pravým tlačítkem do mapy.
6. Kliknutí pravým tlačítkem na trasu vybrání *Show/Hide Track points*. Po zobrazení vybrání konkrétního bodu, kliknutí pravým tlačítkem a zvolení možnosti *Move Waypoint*.
7. Kliknutí na trasu pravým tlačítkem v místě, kde bude trasa rozdělena a vybrání možnosti *Split Track*.
8. Kliknutí na tlačítko u segmentu trasy pro zobrazení menu a zvolení *Settings*.
9. Kliknutí na políčko s názvem "Line color", vybrání nové barvy a kliknutí na tlačítko *Save*.
10. Kliknutí na trasu pravým tlačítkem a vybrání možnosti *Show graph of Track*.
11. Ve stromové struktuře kliknutí na menu vedle názvu souboru a vybrání *Delete file*.

Příloha C

Řešení po krocích pro scénář 3

1. Kliknutí na tlačítko *Import file*.
2. Ve stromové struktuře kliknutí na menu vedle názvu souboru a vybrání *Rename file*, napsání nového názvu a uložení.
3. Kliknutí na název souboru, aby byl vybrán pro editaci.
4. Kliknutí na mapu pravým tlačítkem myši v místě, kde bude trasa začínat, vybrání možnosti *Add TrackAuto*, přidání bodů kliknutím levého tlačítka myši a ukončení kreslení tlačítkem *Exit drawing*, které je umístěno v menu po kliknutí pravým tlačítkem do mapy.
5. Najetí myší na cestu, kliknutí pravým tlačítkem myši, zvolení volby *Delete Route*.
6. Najetí myší na trasu, kliknutí pravým tlačítkem myši, zvolení volby *Move Track to other GPX* a zvolení souboru, kam se trasa kopíruje.
7. Ve stromové struktuře kliknutí na menu vedle názvu souboru a vybrání *Download file*.