UNIVERZITA PARDUBICE
Fakulta elektrotechniky a informatiky

Návrh a implementace webové aplikace pro vyhodnocení polohy mobilních objektů

Jan Holec

Bakalářská práce
2018

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: **2016/2017**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Jan Holec**
Osobní číslo:	**I14097**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Návrh a implementace webové aplikace pro vyhodnocení polohy mobilních objektů**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování:

Cílem práce je návrh a implementaci webové aplikace pro vyhodnocení polohy mobilních objektů se zaměřením na pohyb závodníků na trati.
V teoretické části práce bude proveden přehled současného uplatnění lokalizace polohy při pořádání sportovních událostí
V praktické části se student zaměří na vlastní návrh a implementaci webové aplikace, která bude umožňovat vyhodnocovat polohu mobilních objektů. Navržená aplikace bude pracovat na databázi, do které budou informace o poloze zasílány externím HW zařízením.
Výsledná aplikace bude otestována v reálném provozu.

Rozsah grafických prací:

Rozsah pracovní zprávy:

Forma zpracování bakalářské práce: **tištěná**

Seznam odborné literatury:

ULLMAN, L. PHP a MySQL. Brno, Computer Press Books, 2004

BORONCZYK, Tim. PHP 6, MySQL, Apache: vytváříme webové aplikace. Brno: Computer Press, 2009. ISBN 978-80-251-2767-4.

Vedoucí bakalářské práce: **Ing. Jan Fikejz, Ph.D.**
Katedra softwarových technologií

Datum zadání bakalářské práce: **31. října 2016**

Termín odevzdání bakalářské práce: **12. května 2017**

Univerzita Pardubice Česká republika

L.S.

Ing. Zdeněk Němec, Ph.D.
děkan

Mgr. Josef Horálek, Ph.D.
vedoucí katedry

V Pardubicích dne 31. března 2017

Prohlášení autora

Prohlašuji, že jsem tuto práci vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Pardubicích dne 7. 5. 2018 Jan Holec

Poděkování

Rád bych poděkoval především panu Ing. Janu Fikejzovi, Ph.D. za odborné vedení během celého procesu zpracování Bakalářské práce. Dále chci poděkovat své rodině a přátelům za vyjádřenou podporu.

Anotace

Práce je zaměřena především na návrh a implementaci webové aplikace pro monitorování polohy osob při sportovních aktivitách. Teoretická část se zabývá možnostmi Geolokace s důrazem na globální satelitní navigační systémy. Dále je provedena rešerše nad vybranými projekty z oblasti sledování polohy při sportovních aktivitách. Závěr teoretické části obsahuje přehled a krátký popis nejvyužívanějších technologií pro psaní webových aplikací. Praktická část je zaměřena na implementaci aplikace. Aplikace je rozdělena na Klientskou část, která je vytvořena v Javascriptovém frameworku Vuejs a na serverovou část, pro kterou je použit mikro-framework Lumen. V práci je také popsána implementace funkcí pro zpracování GPS dat. Mezi tyto funkce patří například zjemnění, nebo aproximace dat na stanovenou dráhu závodu.

Klíčová slova

GPS, SPA, geolokace, webová aplikace, Vuejs, Lumen

Title

Design and implementation of web application for evaluating the location of mobile objects

Annotation

The document is mainly focused on the design and implementation of web application for tracking the position of people in sports activities. The theoretical part deals with the possibilities of geolocation with an emphasis on global satellite navigation systems. In addition, a search is made of selected position monitoring projects for sporting activities. Conclusion of the theoretical part contains overview and short description of the most used technologies for creating web applications. The practical part is focused on implementation of the application. The application is divided into the Client part, which is created in the Javascript framework Vuejs and the server part is built by the microframework Lumen. The thesis also describes the implementation of GPS data processing functions. These include, for example, refinement or approximation of data to a specified race course.

Keywords

GPS, SPA, geolocation, web application, Vuejs, Lumen

Obsah

Seznam zkratek

AJAX	Asynchronous JavaScript and XML
API	Application Programming Interface
CSS	Cascading Style Sheets
DBAL	Database Abstraction Layer
DCL	Data control language
DDL	Data definition language
DDoS	Denial of service
DML	Data manipulation language
DOM	Document Object Model
ES5	ECMAScript 5
ES6	ECMAScript 6
GLONASS	Globalnaja navigacionnaja sputnikovaja sistěma
GNSS	Global Navigation Satellite Systém
GPRS	General Packet Radio Service
GPS	Global Positioning Systém
GSM	Global System for Mobile Communications
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
JS	JavaScript
JSON	JavaScript Object Notation
JSX	JavaScript Syntax eXtension
MVC	Model-view-controller
MVVM	Model–view–viewmodel
MWW	Model-View-Whatever
NPM	Node package manager
PHP	Hypertext Preprocessor
REST	Representational State Transfer
SASS	Syntactically Awesome Style Sheets
SEO	Search Engine Optimization
SOAP	Simple Object Access Protocol
SPA	Single-page application
SQL	Structured Query Language
URL	Uniform Resource Locator
XML	Extensible Markup Language
XML-RPC	Extensible Markup Language - Remote procedure call
XSS	Cross-site scripting

Seznam obrázků

Úvod

Technologie pro zaznamenávání polohy dnes již nejsou používané pouze pro armádní, nebo vědecké účely, jak tomu bylo v minulosti. V posledních letech byla tato technologie natolik posunuta kupředu že je dnes možné určit polohu za pomocí nejen specializovaného zařízení, ale také pomocí chytrého telefonu, nebo třeba fitness náramku. Vzhledem k dostupnosti se začala geolokace využívat v celém spektru lidských činností. Jednou z oblastí využívající lokalizaci jsou i sportovní události, kde je například možné monitorovat pohyb závodníků.

Na začátku teoretické části se práce zabývá, problematikou určování polohy se zaměřením na použití při sportovních událostech. Jako první je uveden úvod do problematiky včetně přehledu a porovnání technologií, které jsou pro tento účel využívány.

V druhé části pak práce obsahuje rešerši čtyřech existujících projektů. Je zde uveden jejich krátký popis a zhodnocení.

Třetí část je věnována vývoji moderních webových aplikací. V rámci tohoto uvedení je vysvětleno, co znamenají pojmy SPA, REST API, imutabilita, nebo reaktivní aplikace. Dále bude uveden přehled existujících technologií včetně krátkého popisu a porovnání výhod a nevýhod, které tyto technologie přináší. Na závěr teoretické části jsou popsány technologie použité pro realizaci praktické části.

Praktická část si bere za úkol vytvoření vlastní webové aplikace pro grafické zobrazení polohy osob při sportovních událostech. Aplikace vznikla v nepřímé návaznosti na jinou bakalářskou práci zabývající se návrhem zařízení, které periodicky zaznamenává informace o své poloze. Data ze zařízení jsou odesílána a ukládána do databáze. Podrobnou dokumentaci a další informace o zařízení je možné nalézt v práci s názvem GPS záznamník polohy od Roberta Břízy.

Primárním úkolem bylo vytvořit webovou aplikaci pracující nad daty z tohoto zařízení a zpřístupnit je tak běžnému uživateli. Nejedná se ovšem o pouhé zobrazení polohy. V aplikaci je možné přehrávat záznam ze sportovní události, nebo sledovat právě probíhající závod. K dispozici jsou také statistické údaje jednotlivých týmů jako je např. průměrná rychlost, nebo převýšení.

Aplikace je navržena s důrazem na moderní a pro uživatele příjemné grafické rozhraní. Zároveň zde jsou využity technologie, které umožňují vysoký výkon a plynulý běh aplikace a zpříjemňují tak práci uživatele s aplikací. Při návrhu je myšleno na minimalizaci nutného síťového provozu mezi aplikací a webovým serverem. Některé funkce aplikace bude možné využívat i bez připojení k internetu.

1 Současné využití geolokace při sportovních událostech

Od roku 1990 se pro geolokaci začal používat globální navigační satelitní systém (GNSS) GPS, který umožňuje určit polohu nezávisle na denní době, nebo počasí. Není divu, že se tento systém rozšířil do mnoha lidských činností včetně sportovních událostí a sportu obecně. Netrvalo dlouho a podle vzoru GPS začaly vznikat podobné systémy. I když se nové systémy poměrně rychle začleňují do běžného provozu, nejpoužívanějším navigačním systémem pro civilní využití stále zůstává systém GPS. Hlavní důvodem, proč jsou satelitní navigační systémy tolik využívané je integrace přijímačů GNSS do téměř každého chytrého telefonu. Vedle navigačních systémů se dnes pro měření rychlosti pohybu po předem definované trase využívá tzv. systém bran. [1]

1.1 Přehled lokalizačních systémů

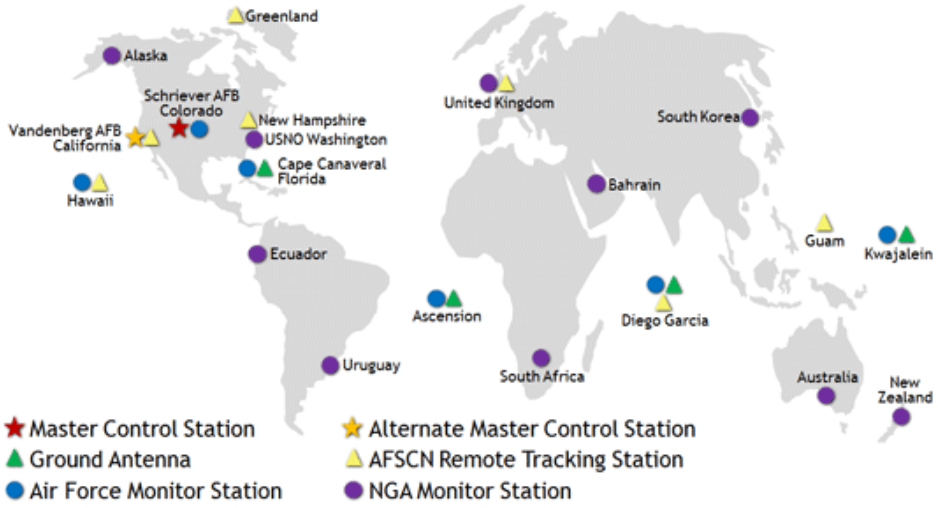
1.1.1 GPS

GPS je označení pro navigační systém, který dokáže určit polohu kdekoli na povrchu země. Projekt GPS vznikl v roce 1973 jako projekt Ministerstva obrany USA. Začátkem 90. let kongres USA schválil užívání GPS pro civilní účely. Současná přesnost určení polohy pro civilní využití se pohybuje do 5 m. V budoucnu je plánována modernizace, která by měla určení polohy zpřesnit až na 30 cm. Pro určení polohy je zapotřebí signálu nejméně ze 3 družic. Systém GPS bývá rozdělován na tři podsystémy. [1], [2]

Kosmický podsystém: Je tvořen 31 družicemi, 7 družic slouží jako záložní. Na obrázku 1 je zobrazena družice systému GPS. Družice obíhají kolem země ve vzdálenosti 20 200 km po 6 oběžných drahách. Vybavení družic zahrnuje především přijímač, vysílač a atomové hodiny dále je družice vybavena systémy pro navigaci a případně i systémy pro speciální určení. Družice komunikují s řídícím centrem na zemi, kterému poskytují informace o své poloze a stavu svých systémů. [2], [3]

Obrázek 1 – Družice systému GPS, zdroj [5]

Řídící podsystém: Řídící systém monitoruje stav každé družice a sdílí informace o její poloze ostatním družicím. Řídící systém je sestaven z hlavní řídící stanice v Colorado Springs, záložní hlavní řídící stanice, 16 monitorovacích stanic a 11 řídících stanic. Na obrázku 2 je možné vidět mapu zobrazující jednotlivé stanice. [4], [5]



Obrázek 2 - Pozice kontrolních center systému GPS, zdroj [3]

Uživatelský podsystém: Systém tvořený přijímači GPS signálu. Přijímačem může být specializované zařízení, nebo třeba chytrý telefon. [5]

1.1.2 GLONASS

GLONASS je původně Sovětský a nyní Ruský navigační systém, velice podobný systému GPS. Kosmický podsystém zde tvoří 24 družic stejně jako je tomu u systému GPS. Družice se pohybují ve vzdálenosti 19 100 km. Vývoj systému GLONASS začal ještě zhruba o 10 let dříve, než vývoj GPS ale z důvodu špatného financování byl spuštěn až v roce 1982. Od jeho spuštění byl GLONASS udržován mimo provoz. Kolem roku 2001 začalo Rusko s jeho modernizací a postupným uvedením do aktivního využívání. V roce 2010 GLONASS pokryl celé území Ruska a o rok později dosáhl globálního pokrytí. Přesnost určení polohy se uvádí do 4,5 m. Systém GLONASS je možné kombinovat se systémem GPS a dosáhnout tak přesnosti až 2,5 m. V budoucnosti se plánuje zpřesnění na 0,6 m. [1], [6]

1.1.3 GALILEO

GALILEO je Evropský navigační systém. Jedná se o první navigační systém určený pouze pro civilní potřeby. Jeho plánované využití je cíleno především na všechny druhy dopravy, ale bude možné jej využívat i pro jiné oblasti. Kompletní satelitní systém by se měl skládat z 30 družic z toho budou 3 družice záložní a 27 jich bude operativně využíváno. Družice budou zemi obíhat ve vzdálenosti 23 222 km a jejich dráhy budou s rovinou rovníku svírat 65°. Přesnost systému by se měla pohybovat nejvýše do 1 m. GALILEO je funkční teprve z části, jeho plné dokončení je plánováno na rok 2020. [7]

Druhy služeb poskytované systémem GALILEO:

- základní služba – veřejně dostupný signál,
- komerční služba – komerčně zaměřená služba, bude využívat komerční kódování a o dva signály více oproti základní službě. Komerční kódování bude spravováno GALILEO operátorem a vyžadováno na straně přijímače,
- veřejně regulovaná služba – služba poskytovaná státem vybraným osobám. Bude se jednat především o bezpečnostní složky státu,
- vyhledávací a záchranná složka – služba zaměřená na nouzovou lokalizaci. [7]

1.1.4 Beidou

Beidou je Čínský navigační systém, který se svým návrhem velice podobá systémům GPS a GLONASS.

Beidou-1: První verze Beidou. Poskytovala signál ze 3 družic a pokrytí celého území Číny a několika přilehlých států.

Beidou-2: Beidou-2 je plánovaná modernizace Beidou-1. Oproti Beidou-1 bude tvořen 35 družicemi a svým signálem pokryje celý svět. Plánované dokončení se uvádí na rok 2020.

V současné době je Beidou provozován s 10 družicemi a jeho přesnost by se měla pohybovat do 10 m pro nešifrované veřejně dostupné připojení. Pro šifrované připojení se přesnost pohybuje kolem několika cm. S touto přesností by tak Beidou-2 mohl v budoucnu jednoduše nahradit systém GPS. [1], [8]

1.1.5 GSM určení polohy

Lokalizace podle mobilní sítě je alternativní metoda ke Globálním navigačním systémům. Největší výhodou GSM lokalizace je možnost jejího využití kdekoliv v dosahu mobilní sítě. Tato metoda funguje uvnitř budov i v podzemí na rozdíl od GNSS, které potřebují mít výhled na oblohu. Nejčastěji se tak GSM určení polohy používá v kombinaci s GNSS, a to především tehdy kdy není možné polohu určit pomocí GNSS.

Cell ID: Představuje číslo buňky mobilní sítě. Pomocí čísla buňky je mobilní operátor schopný omezit polohu zařízení pouze na území jedné buňky. Přesnost se bude lišit podle hustoty pokrytí a bude se pohybovat od 100 m do několika kilometrů.

Timing advance: Je metoda, která umí určit přibližnou vzdálenost zařízení od vysílače na základě rychlosti šíření signálu mezi zařízením a vysílačem. Přesnost polohy určené touto metodou se pohybuje okolo 500 m. Pokud by bylo možné využít signál z více vysílačů může se určená poloha přiblížit až k přesnosti několika metrů.

Angle of Arrival: Metoda založena na směrových anténách. Základem metody je měření úhlu přijímaného signálu. Úhel je měřen na straně zařízení, nebo v základnové stanici. Výpočtem vznikají přímky a poloha zařízení je určena jako jejich průsečík.

Enhanced Cell Global Identity: Tato metoda využívá metod Cell ID a Timing advance a rozšiřuje je o sílu signálu. Podle síly signálu je metoda schopna určit nejpravděpodobnější výskyt zařízení. Přesnost metody se pohybuje od 50 m do 500 m, nebo od 250 m do 8 km podle síly mobilního signálu. [39]

1.1.6 Systém bran

Při některých sportovních událostech není nutné znát přesnou polohu závodníka po celou dobu závodu. Právě pro tyto účely slouží systém bran. Na dráze se rozmístí libovolný počet kontrolních bran. Závodník u sebe musí nést identifikační čip pomocí kterého brána pozná o kterého závodníka se jedná. Při závodě závodníci probíhají jednotlivými branami a ty zaznamenávají časy průběhu. U každého závodníka známe časy v jednotlivých kontrolních bodech závodu. Lze tak určit například průměrnou rychlost mezi jednotlivými body, nebo průběžné pořadí.

Při porovnání se satelitními navigačními systémy popsanými výše je zde hlavní výhodou výrazné snížení množství dat a tím i náročnosti pro jejich zpracování. Další výhoda je bezpochyby absence chyby při určení polohy sledovaného objektu. Data o časech a rychlosti jsou o velikost této chyby přesnější. Nepřetržité sledování polohy je také velice náročné na energii a velice rychle vybijí baterii zařízení. Při závodech trvajících i několik dní se toto může jevit jako neřešitelný problém. K mírnému zlepšení dochází, pokud zařízení vysílá data o poloze ve větších intervalech například v rozmezí 5 minut. Hlavní nevýhodou systému bran je absence informací o poloze sledovaného objektu. Není tak možné zpětně modelovat jeho pohyb.

1.2 Rešerše vybraných existujících systémů

1.2.1 FollowMe

FollowMe je projekt, který umožnuje pronájem, nebo koupi sledovacího zařízení. Hlavními klienty projektu FollowMe jsou pořadatelé sportovních událostí. Za pomoci sledovacího zařízení je možné pozorovat závodníky a mít tak přehled o jejich poloze na trati. To je užitečné nejen pro sledování průběhu závodu a zobrazování průběžných výsledků, ale také pro případné krizové situace. Zařízení využívá systému GPS a v předem určených intervalech odesílá informace o poloze na server. Interval je možné nastavit na jednu, nebo dvě minuty. Pro komunikaci se serverem zařízení využívá mobilní signál. V případě nedostupnosti signálu jsou informace o poloze ukládány do paměti zařízení a odeslány při prvním připojení do mobilní sítě. Výdrž zařízení při intervalu odesílání dat dvě minuty se udává kolem 50 hodin. K zařízení je v případě potřeby možné připojit přídavnou baterii a prodloužit tak jeho výdrž. Jak zařízení vypadá zobrazuje obrázek 3.

Uživatel má možnost sledovat pohyb svého zařízení na webové aplikaci poskytovatele. Aplikace umožňuje sledovat pohyb zařízení téměř v reálném čase. Je možné si zde definovat trasu včetně alternativních cest pro případ nečekané situace (např. špatného počasí). Na trase je možné definovat virtuální kontrolní body pro které aplikace vygeneruje informace o čase. Aplikace dokáže zobrazovat několik zařízení najednou. Jednotlivá sledovací zřízení je možné rozdělit do kategorií a přehledně tak seskupit jejich výsledky.

Aplikace působí velmi moderně a uživatelsky přívětivě. Veškeré ovládací prvky jsou přehledně umístěny a při prvním spuštění aplikace je k dispozici krátký návod, který vysvětlí základní funkce aplikace. Velmi přehledně je zpracovaná také tabulka výsledků, která obsahuje časy v kontrolních bodech pro jednotlivé závodníky. Aplikaci je možné vytknout, že není možné zpětně přehrávat záznam pohybu zařízení. Také tu úplně chybí veškeré záznamy o rychlosti a nadmořské výšce.

Výhody:

- aplikace v českém jazyce,
- moderní provedení aplikace,
- přehledný výpis časů pro kontrolní body závodu,
- možnost definovat náhradní trasu pro nepředvídatelné situace.

Nevýhody:

- aplikace postrádá zobrazení dat o rychlosti a nadmořské výšce,
- nelze zpětně zobrazit záznam události. [9]

Obrázek 3 - Sledovací zařízení projektu FollowMe, zdroj [9]

1.2.2 Dotvision Motion

Dotvision Motion je projekt, který poskytuje službu sledování osob při sportovních událostech, ale také služby jako je například sledování zásilky. Pro sledování polohy projekt využívá vlastních sledovacích zařízení. Uživatel může vybírat ze tří různých sledovacích zařízení, které platforma aktuálně podporuje. Zařízení se od sebe liší v mnoha parametrech, ale především účelem jejich použití.

Sigfox tracker: Určen především pro sledování zásilek. Komunikuje na stejnojmenné síti. Umožnuje posílat data o poloze jednou za 10 minut. Výdrž zařízení se pohybuje kolem jednoho měsíce.

GSM tracker: Zařízení určené pro sledování běžných sportovních aktivit. Pro určení polohy je využíváno GPS souřadnic, které jsou posílány po GSM síti. Je proto nutné, aby se zařízení pohybovalo v dosahu mobilního signálu. Shromážděná data o poloze jsou následně posílána na servery platformy pomocí sítě GPRS. Intervaly měření polohy je možné nastavit od 1 s do 1 h. Od tohoto nastavení se odvíjí výdrž baterie. Ta se pohybuje od 20 do 100 h.

Satelite tracker: Zařízení Satelite tracker také určuje polohu za pomoci GPS souřadnic. Na rozdíl od zařízení GSM tracker zde není využito GSM sítě pro přeposílání GPS souřadnic. Zařízení souřadnice získává přímo ze satelitů systému GPS. Pro posílání dat o poloze na server se využívá sítě pro satelitní připojení GlobalStar. Díky nezávislosti zařízení na mobilním signálu je možné jeho využití i v odlehlých oblastech jako jsou například pouště, nebo otevřené moře. Interval měření polohy je možné nastavit od 2,5 do 10 minut a výdrž baterie se pohybuje od 2 do 5 dní. Zařízení neumožnuje uchovávání dat při výpadku připojení a z důvodu přímé komunikace se satelity GPS je nutné, aby zařízení mělo nekrytý výhled na

oblohu. Ze všech poskytovaných zařízení je Satelite tracker finančně nejnáročnější na provoz. To je způsobeno především využíváním satelitního připojení.

Webová aplikace Dotvision Motion na první pohled působí zastarale. Je to způsobeno především rozmazanými ikonami, a ne příliš povedeným designem ovládacích prvků. Na druhou stranu aplikace poskytuje o mnoho více dat než třeba konkurenční FolowMe. Kromě základních funkcí jako je sledování pohybu závodníků téměř v reálném čase, nebo možnost sledování oblíbeného závodníka. Dokáže aplikace přehrávat záznam události zpětně. Nechybí ani možnost záznam libovolně posovat, nebo měnit jeho rychlost. Seznam závodníků je možné řadit podle aktuální vzdálenosti, aktuální rychlosti, nebo stavu. Nad časovou osou je možné zobrazit graf nadmořské výšky pro celou trasu. Na tomto grafu se zobrazuje pozice všech závodníků. Užitečná je také možnost výběru ze tří mapových podkladů. K dispozici jsou mapové podklady od Google map, Bing map a Open map. Velice pěkným způsobem je řešena možnost rozdělení trasy na jednotlivé úseky, kdy je rozdělení barevně odlišeno nejen na mapě, ale také na grafu nadmořské výšky. U jednotlivých závodníků je možné měnit barevné zobrazení. Co zde naprosto chybí je možnost definovat kontrolní body, nebo jiná možnost zobrazení souhrnných statistik o průběhu závodu. Veškeré informace jsou dostupné pouze v tabulce závodníků, a to pouze pro aktuální čas simulace. Zajímavou ale z mého pohledu poměrně zbytečnou funkcí je 3D pohled na trasu závodu.

Výhody:

- možnost výběru ze tří zařízení,
- v aplikaci lze přepínat mezi mapovými podklady,
- graf nadmořské výšky pro celou trasu závodu,
- možnost přehrávat záznam sportovní události.

Nevýhody:

- Uživatelsky nepřívětivý vzhled aplikace,
- veškeré údaje jsou dostupné jen pro aktuální čas. [10]

1.2.3 Followmychallenge.com

Projekt Folow my challenge umožnuje sledovat jednotlivce, nebo celé týmy při plnění osobních výzev. Takovou výzvou může být například cesta pěšky přes celou Kanadu, nebo plavba malou lodí kolem světa. Platforma nemá žádné své sledovací zařízení, které by klientům mohla nabídnout, ale dokáže se propojit s mnoha běžně prodávanými sledovacími zařízeními. Volba zařízení záleží tedy především na preferencích a finančních možnostech klienta. Po propojení s platformou je možné odesílaná data zobrazit na webové aplikaci.

Aplikace je velice jednoduchá, většinou zobrazuje plánovanou trasu výzvy a také dosavadní pohyb zařízení. Na mapě se mohou zobrazovat fotky pořízené klientem během cesty.

K dispozici jsou také podrobná data o teplotě vzduchu, rychlosti větru, rychlosti pohybu a nadmořské výšce. V aplikaci nelze zpětně přehrávat záznam pohybu.

Výhody:

- možnost využití vlastního sledovacího zařízení,
- přehledy v aplikaci obsahují také informace o aktuální teplotě, nebo rychlosti větru.

Nevýhody:

- aplikace neumožnuje přehrát záznam. [11]

1.2.4 Tractrac.com

TracTrac je služba určená především pro lodní závody, nebo orientační běhy. Služba poskytuje účastníkům závodu vlastní sledovací zařízení. Jedná se o stejný typ zařízení jako poskytuje projekt FollowMe.

Dalo by se říct, že webová aplikace služby TracTrac si vzala to nejlepší z aplikace FollowMe a aplikace Dotvision Motion a ještě přidala několik vlastních funkcí. V aplikaci je možné sledovat průběh závodu v téměř reálném čase, ale také přehrávat záznam. Je zde rozdělení soutěžících do kategorií, možnost vybrat oblíbené závodníky a možnost řadit seznam závodníků podle jednotlivých parametrů. Uživatel si také může změnit velikost ovládacích prvků. Na výběr má ze čtyř velikostí. Aplikace obsahuje nastavení trasy jednotlivých závodníků. Je možné zobrazovat trasu od začátku závodu, od posledního kontrolního bodu, nebo pouze několik sekund zpětně. Jediné, co v aplikaci opravdu chybí je možnost vystředění mapy a případné sledování oblíbeného závodníka. Takto je nutné mapu ručně posouvat což je nepříjemné. Vzhled aplikace je moderní a přívětivý. Jediná věc, co by se dala grafickému návrhu vytknout je veliká bílá hlavička. Jelikož je celá aplikace navržena do tmavých barev, bílá hlavička nepříjemně svítí a ruší tak dojem ze zbytku aplikace.

Výhody:

- možnost přehrávat záznam zpětně,
- je možné nastavit zobrazení trasy závodníků,
- aplikace umožňuje měnit velikost rozhraní.

Nevýhody:

- aplikace neumožňuje vystředit mapu na oblíbeného závodníka,
- nevhodně zvolený styl hlavičky aplikace. [12]

2 Vývoj moderních webových aplikací

Při návrhu moderní webové aplikace je nutné nejprve vybrat správné technologie pro jednotlivé části aplikace. Právě výběr správné technologie může velice usnadnit celý vývoj. Jelikož SPA aplikace jsou velice oblíbené není divu, že pro jejich tvorbu existuje nepřeberné množství technologií. V následujících kapitolách budou představeny a porovnány nejznámější a také některé méně známe technologie pro vývoj SPA.

2.1 Mapové aplikační rozhraní

Jelikož je mapa stěžejní částí aplikace je nutné vybrat správný zdroj pro mapové podklady. I když existuje mnoho volně využitelných mapových podkladů ne všechny jsou dostatečně kvalitní. V další kapitole budou popsány a porovnány dva pro náš region nejzajímavější zdroje mapových podkladů.

2.1.1 Seznam

Česká společnost Seznam.cz, a.s. poskytuje veřejně dostupné API pro mapové podklady. Podklady jsou dostupné zdarma i pro komerční využití. Nejsilnější stránkou map od Seznamu je bezesporu přesnost a detailnost podkladů pro českou republiku.

Výhody:

- velmi používaná mapa v ČR,
- detailní mapové podklady pro území ČR,
- volné licenční podmínky.

Nevýhody:

- omezené funkce pro tvorbu reaktivní mapy,
- chybí integrace do moderních JS frameworků. [13]

2.1.2 Google

Další veřejné mapové podklady poskytuje společnost Google. Google mapy jsou celosvětově používané mapové podklady s velmi širokou škálou funkcí.

Výhody:

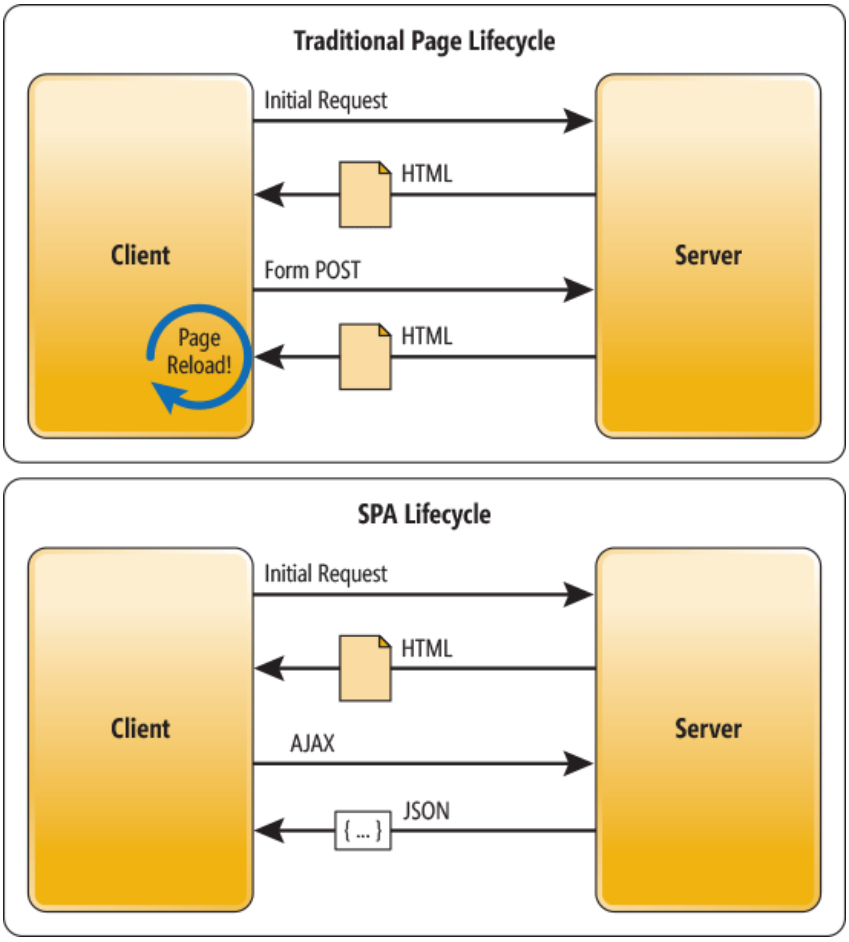
- celosvětový poskytovatel map,
- obsahují funkce pro vytvoření reaktivní mapy,
- integrace pro většinu moderních JS frameworků.

Nevýhody:

- pro oblast ČR méně detailní mapy, než mají Mapy.cz,
- pro napojení je nutné vytvořit Google API key,
- při větší návštěvnosti mohou vznikat zpoplatnění služby. [13]

2.2 SPA aplikace

Zkratka SPA je označení pro jednostránkové aplikace. Jednostránková aplikace se vyznačuje především jediným načtením dokumentu stránky. Pro další komunikaci se již využívá AJAX dotazů a nedochází k dalšímu načítání stránky. Odlišné přístupy pro komunikaci se serverem názorně popisuje obrázek 3. Veškeré změny aplikace se dějí v prohlížeči uživatele. Dokument, který je poslán ze serveru často neobsahuje nic kromě připojených JS souborů. O samotné vygenerování obsahu stránky a případné překreslení po aktualizaci dat se stará JS.



Obrázek 3 – Postup komunikace se serverem pro SPA a tradiční webovou stránku, zdroj [14]

Oproti klasickému přístupu, kdy je pokaždé načten celý dokument stránky a dochází tak v prohlížeči ke znovunačtení působí přístup SPA mnohem modernějším dojmem. Při získávání nových dat ze serveru se SPA zaměřuje pouze na data, která jsou aktuálně potřeba. Při práci se SPA není nutné při každé změně načítat veškeré statické zdroje jako jsou soubory CSS, JS, nebo obrázky. Díky tomuto přístupu dochází mnohdy k výrazné úspoře v objemu sítové komunikace.

Základ SPA tvoří View vrstva, která definuje strukturu dat prezentovaných uživateli. Každá View vrstva je propojena s Model View vrstovu. Model View vrstovu představuje objekt, který definuje data pro View vrstvu. Při aktualizaci modelu musí dojít také k překreslení View vrstvy s novými daty. Změna modelu může pocházet přímo od uživatele například uživatel odeslal formulář, nebo jako výsledek komunikace se serverem. Celému procesu se říká data binding a bývá velice složitý pro implementaci. Naštěstí existuje veliké množství open source frameworků, které tuto problematiku řeší.

Způsob, jakým SPA působí na uživatele je možné přirovnat spíše k desktopové aplikaci než ke klasické webové aplikaci. Výhodou SPA oproti desktopovým aplikacím je nezávislost na platformě. Funkce SPA závisí především na podpoře prohlížeče.

Hlavní nevýhodou SPA je optimalizace pro SEO. Jelikož je obsah celé stránky generovaný JS až v prohlížeči klienta, vyhledávače vidí pouze prázdnou stránku. Tento problém se řeší především prvním vygenerováním aplikace na straně serveru. Další nevýhodou oproti klasické webové aplikaci je menší odolnost proti útokům typu XSS. [14], [15]

2.2.1 Angular 2.0

Angular je Javascriptový framework pro psaní progresivních webových aplikací od společnosti Google. Jeho výjimečnost spočívá především v nutnosti psát JS kód v Typescriptu. Na rozdíl od Javascriptu je Typescript staticky typovaný což umožňuje při kompilaci kontrolovat datové typy operátorů při jednotlivých operacích a vyhnout se tak nesnadno odhalitelným chybám nechvalně známých z dynamicky typovaných jazyků. Angular je velice robustním řešením, které postačuje i pro velmi rozsáhlé aplikace. Pro projekty menšího rozsahu může být Angular až příliš univerzální a vynucený Typescript je pro některé vývojáře spíše na škodu. Právě kvůli univerzálnosti Angularu je jeho plné zvládnutí považováno za mnohem složitější, než je tomu u konkurenčních frameworků.

Pro rozdělení aplikace Angular používá vlastní modulární systém. Aplikace se pomocí tohoto systému dělí na jednotlivé NgModuly, které následně obsahují jednotlivé funkce. Každá aplikace musí mít právě jeden hlavní modul a jednu kořenovou komponentu.

Komponenty: Obsahují obslužné funkce pro komunikaci se šablonou. Vytváří obsah a následně reagují na vstup od uživatele. Každá komponenta je propojena se šablonou.

Šablony: Soubor obsahující HTML kód a Angular direktivy. Slouží pro vykreslení obsahu získaného z komponenty.

Služby: Obsahují obecnou funkcionalitu nezávislou na šablonách. Jednotlivé služby lze importovat napříč celou aplikací a dosáhnout tak přehledného a znovupoužitelného kódu.

Dekorátory: Jsou anotace definující typ jednotlivé komponenty, nebo služby. Pomocí dekorátoru jsou vytvořena metadata, která následně umožnují funkce jako je spárování šablony a komponenty, nebo možnosti dependency injection. [16], [17], [18]

Výhody:

- pevná struktura aplikace vhodná pro větší projekty,
- dependency injection,
- jednoduchá implementace pro Google služby,
- možnost psát nativní aplikace s podporou NativeScript, Ionic, nebo Cordova,
- serverside vykreslování pro mnoho serverových jazyků,
- využití TypeScriptu,
- plnohodnotný MVC framework.

Nevýhody:

- dlouhá doba potřebná k plnému zvládnutí,
- závislost na TypeScriptu,
- pro malé aplikace může být příliš robusní,
- pevně daná struktura aplikace může být omezující.

2.2.2 React

Stejně jako Angular je i React Javascriptový framework pro tvorbu webových aplikací. Za vznikem Reactu stojí společnost Facebook, která React interně používala a vyvíjela již několik let před jeho prvním vydáním. Při vydání se musel React potýkat s nepřátelským přijetím. Jeho návrh totiž připomínal spíše krok zpět, něž moderní a inovativní přístup k psaní webových aplikací. Bylo to zapříčiněno především tím, že React používá jazyk JSX pro psaní reaktivních šablon. Syntaxe JSX velice připomíná zápis jazyka HTML i když se jedná o syntax sugar pro psaní JS objektů. Na první pohled tak React komponenta vypadala jako směs Javascriptového kódu a HTML. Po nějakém čase si vývojáři na zápis JSX zvykli a React dnes patří mezi nejpoužívanější frameworky.

Velikou předností Reactu je, že vede programátora k psaní čistých funkcí bez postranních efektů. Právě tyto takzvané "Pure functions" jsou základem pro funkcionální programování. I když tato metoda někdy přináší komplikace, výsledný kód je vždy jasný a přehledný. Mezi další výhody Reactu je snaha o imutabilitu. Imutabilita volně přeloženo znamená neměnnost. V programovací praxi si to můžeme představit jako neměnné struktury či objekty. V případě že je potřeba změnit stav aplikace nedochází k úpravě existujícího stavu, ale vytvoří se stav nový s již změněnou hodnotou. Takovýto přístup umožňuje udržet si přehled o fungování i velmi rozsáhlé aplikace.

Základním kamenem každé React aplikace jsou komponenty. Existuje několik způsobů, jakými lze komponenty zapisovat. Nejčastějším a nejpřehlednějším zápisem je zápis pomocí ES6 třídy, která dědí základní implementaci od třídy Component, kterou je možné nalézt v balíčku React. Třída musí obsahovat konstruktor ve kterém je nutné zavolat konstruktor předka a funkci render. Návratovou hodnotou funkce render je šablona komponenty definována syntaxí JSX, nebo pomocí JS objektů. Komponenty můžeme rozdělit na komponenty bez stavové a na komponenty s vlastním stavem.

Výstup bez stavové komponenty je řízen pouze pomocí imutabilního objektu props, který je posílán jako argument z rodičovské komponenty. Objekt props nelze uvnitř komponenty měnit a s každou změnou properties v rodičovské komponentě dochází k překreslení celé komponenty. Bez stavové komponenty mají implementovaný implicitní konstruktor a ten se nemusí explicitně uvádět. Stavové komponenty na rozdíl od nestavových komponent udržují vlastní stav, který je tvořen na základě objektu props a může se měnit v reakci na události od uživatele. [19]

Výhody:

- možnost využít funkcionální programování,
- čisté funkce bez postranních efektů,
- možnost využívat JSX pro definici šablon,
- silná komunita se spoustou návodů a již hotových řešení.

Nevýhody:

- zásady funkcionálního programování nemusí vyhovovat každému,
- zápis šablony v JSX není ze začátku příliš intuitivní a člověk si na něj musí zvyknout.

2.2.3 Vuejs

Vuejs je poměrně nový framework jeho autorem je bývalý vývojář z Angular týmu, který využil svých zkušeností a navrhl framework, který bere jen to nejlepší z Angularu 1 a zároveň přidává mnoho vlastních funkcí. Největší výhodou Vuejs je určitě vysoce rychlá přímka učení. Pokud člověk má znalosti jazyka HTML a JS ES5 (běžně používaný standard Javascryptu) je schopný již po jednom dni psát netriviální aplikace.

Základní prvkem VueJs aplikace je komponenta. Komponenty jsou zapisovány do souboru s koncovkou vue. Soubor můžeme rozdělit na tři části.

Šablona: Část definující vzhled komponenty. Kód pro šablony se zapisuje mezi tagy template a jeho zápis svou syntaxi odpovídá zápisu značkovacího jazyka HTML. Každá Vue šablona musí obsahovat pouze jeden root element. V šablonách je možné kromě běžných HTML elementů zapisovat také tagy reprezentující jiné VueJs komponenty. Takto je možné vytvářet stromovou strukturu aplikace. Vložené komponentě je možné předat data pomocí atributů s prefixem "v-bind:nazev-atributu" názvy atributů musí být ve vkládané komponentě definovány pod objektem properties, nebo se musí jednat o standardní HTML5 atributy (např. title, nebo alt). V šabloně je možné využívat šablonovací atributy začínající vždy předponou "v-" mezi nejdůležitější patří například "v-if" pro podmíněný rendering obsahu, nebo třeba "v-for" pro využití cyklu.

Model: Část definující data pro šablonu musí být zapsána mezi tagy script a její syntaxe odpovídá JS objektu. Nejdůležitější objektu je metoda data, která vrací objekt představující stav komponenty. Jednotlivé položky tohoto objektu je možné tisknout v šabloně. Jakákoliv změna tohoto objektu automaticky promítne změnu i do šablony. Objekt může obsahovat

mnoho dalších atributů mezi které patří například atribut methods, properties. nebo computed.

Styly: Poslední částí komponenty jsou CSS styly zapisované mezi tagy style. Styly mohou být globální, nebo se mohou vztahovat pouze k dané komponentě. [20], [21], [22]

Výhody:

- možnost používat pro vykreslovací funkci šablony, které mají téměř stejnou syntaxi jako HTML,
- rychlostí srovnatelný s konkurenčními frameworky,
- úplná svoboda v návrhu struktury aplikace,
- přehledná dokumentace a rychle rostoucí komunita.

Nevýhody:

- u rozsáhlejších aplikací se může struktura stát nepřehlednou.

2.2.4 jQuery

jQuery je stále nejpoužívanější Javascriptovou knihovnou i když se o její budoucnosti jsou pochybnosti. Oproti ostatním frameworkům popsaných výše totiž používá přímou manipulaci s DOM, který je oproti virtuálnímu DOMu velice pomalý a každá operace je velice náročná na čas i na paměť. jQuery navíc neumožnuje téměř žádné strukturování aplikace, a tak u rozsáhlejších aplikací dochází k veliké nepřehlednosti kódu.

Funkce jQuery je postavena na query selektorech. Pomocí selektoru je možné vybrat konkrétní element, nebo skupinu elementů z DOM. Výběr elementu je specifikován CSS selektory. Nad elementem je možné provádět mnoho operaci jako je například změna obsahu, skrytí, nebo zkopírování elementu. Je také možné definovat metody reagující na změny od uživatele.

Největší výhodou jQuery je možnost manipulace s celým DOM. Na druhou stranu jQuery neobsahuje žádný reaktivní stav, který by automaticky reagoval na změnu a provedl aktualizaci šablony. O veškeré aktualizace dat se musí postarat programátor. [23]

Výhody:

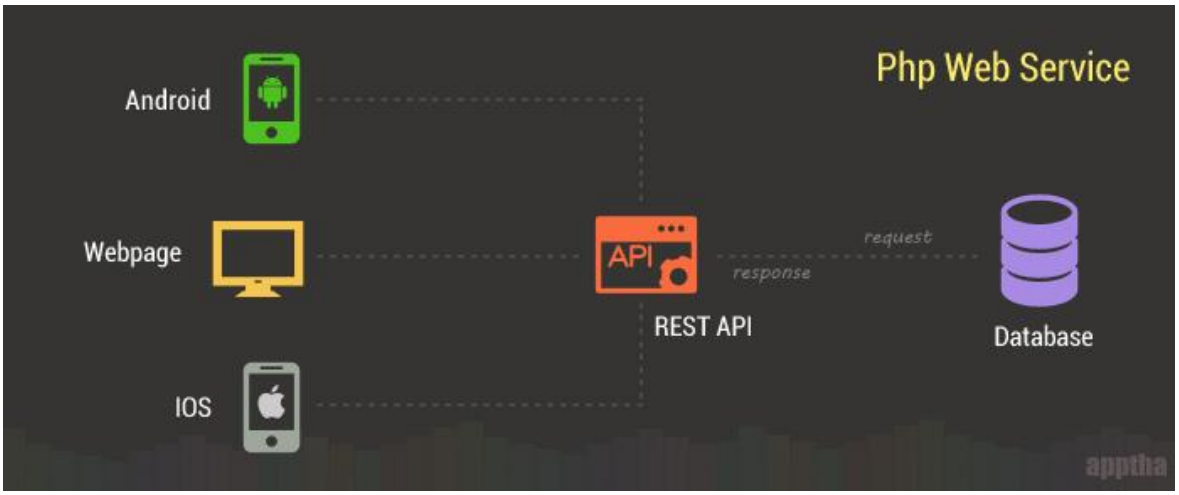
- velice jednoduché pro zvládnutí základních funkcí,
- široký výběr funkcí a rozšíření,
- možnost manipulace s celým DOM,
- možnost psát vlastní moduly a rozšíření.

Nevýhody:

- velmi obtížná organizace kódu,
- využívá přímou manipulaci s DOM,
- chybí reaktivní stav.

2.3 Datové aplikační rozhraní

Pro každou webovou aplikaci je nutné připravit rychlé přístupové API, které bude zprostředkovávat komunikaci aplikace s databází. Na obrázku 4 je možné vidět strukturu sítě, kde mají různé zařízení přístup k databázi skrze sdílené aplikační rozhraní. Největší výhodou sdíleného API je možnost využít jednotného rozhraní pro práci s daty skrze všechny platformy a zařízení, které dokáží s API komunikovat. Hlavním úkolem API je přijímat dotazy od aplikace, zpracovat je a vrátit správnou odpověď. Právě v této části návrhu je nutné myslet především na bezpečnost a vysoký výkon při zpracování výsledků.



Obrázek 4 - Struktura sítové komunikace při využití sdíleného aplikačního rozhraní, zdroj [24]

2.3.1 PHP

PHP je objektově orientovaný dynamicky typovaný skriptovací jazyk. PHP script je vložený v obsahu stránky a prováděný při dotazu na server. O provední PHP scriptu se stará engine Zend, který překládá PHP kód do Zend opcodes, které následně spouští. Výsledkem spuštění Zend opcodes je HTML. Jedná se tedy o jazyk prováděný na straně serveru používaný především pro psaní webových aplikací. V roce 2018 se jazyk PHP vyskytuje na pozadí téměř 60% všech internetových stránek.

PHP 7: Nová verze PHP. Oproti verzi 5.6 přináší až o polovinu větší výkon a nižší nároky na paměť. Toto zrychlení je výsledkem výrazného refaktoringu. [25]

2.3.2 Lumen

Lumen je PHP framework, který vznikl jako specializace pro API z frameworku Laravel. Lumen patří mezi frameworky s nejrychlejší odezvou. Jeho velikou výhodou je i možnost provést kdykoliv upgrade na plnohodnotný Laravel bez nutnosti jakéhokoliv přepisování kódu. Pro práci s databází Lumen využívá query builder z Laravelu. [26]

2.3.3 Silex

Silex je micro-framework napsaný v jazyce php využívající knihoven z frameworku Symfony. Tyto knihovny vytvářejí abstrakci nad HTTP dotazy a umožnují tak jednoduché psaní API. Pro jednoduchou komunikaci s databází Silex integruje Doctrine DBAL. Silex je dostupný ve dvou verzích. Verze "fat" obsahuje šablonovací systém, Symfony komponenty a mnohem více částí, které pro tvorbu API nejsou potřeba. Verze "slim" obsahuje pouze základní routing systém. [26]

2.3.4 Limonade

Dalším PHP micro-frameworkem pro psaní rychlého API je Limonade. Na rozdíl od frameworků představených výše je Limonade nezávislý framework s vlastní implementací http komunikace. Největší výhodou frameworku Limonade je jeho důraz na minimalizmus a jednoduchost použití. [26]

3 Technologie použité pro vývoj aplikace

3.1 Google map API

Hlavním důvodem, proč jsem si vybral Google mapy místo map od Seznamu bylo propracovanější API pro tvorbu interaktivních map. Také se mi povedlo nalézt modul pro integraci Google map do frameworku VueJS.

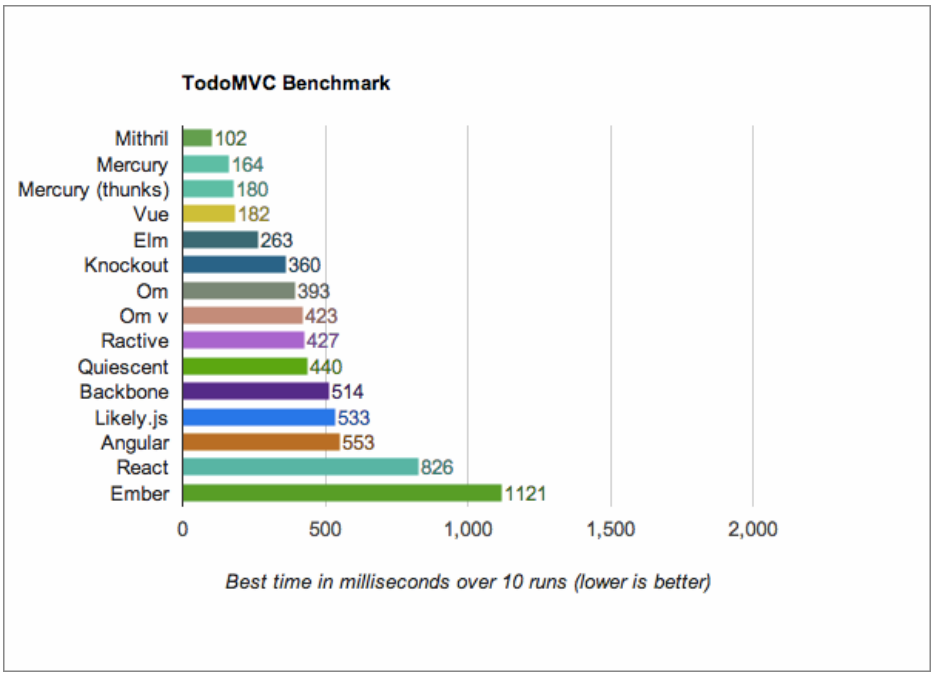
3.2 ECMA Script 2015

ECMA Script 2015 nebo také ES6 je verze jazyku Javascript, která přináší mnoho nového oproti starším verzím a posouvá tak Javascript na úplně novou úroveň. S ES6 přicházejí možnosti objektového programování založeného na třídách, Promises, anonymní funkce, destruktory a mnoho dalšího.

ES6 není bohužel plně podporován ze strany prohlížečů, a tak je nutné jej pomocí prekompilerů převádět na ES5 normu, která je dnes běžně využívána. Nejpoužívanějším nástrojem pro toto využití je v současnosti Babel. Babel bude podrobněji popsán v pozdější kapitole. [27]

3.3 JavaScriptový framework Vue.js

Framework VueJs byl zvolen především kvůli jeho rychlosti. Jak ukazuje graf na obrázku 5 Vuejs patří mezi nejrychlejší Javascriptové frameworky určené pro psaní SPA.

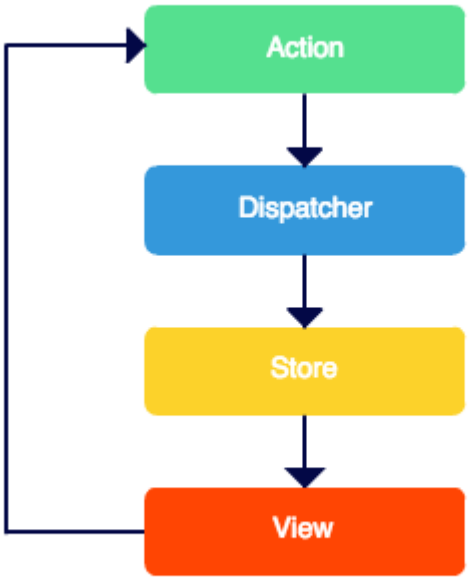


Obrázek 5 - Výsledky rychlostního testu, zdroj [28]

3.3.1 Vuex

Vuex je knihovna, která přináší do Vuejs možnost centralizovaného úložiště stavu aplikace. Jako předloha pro Vuex posloužil Redux, který je pro stejný účel používán v Reactu. Funkce Vuex je založena na čtyřech částech a svým návrhem odpovídá návrhovému vzoru Flux.

Flux: Je návrhový vzor od společnosti Facebook, který definuje proces jednosměrného toku dat. Tento proces je znázorněn na obrázku 6. [29], [30]



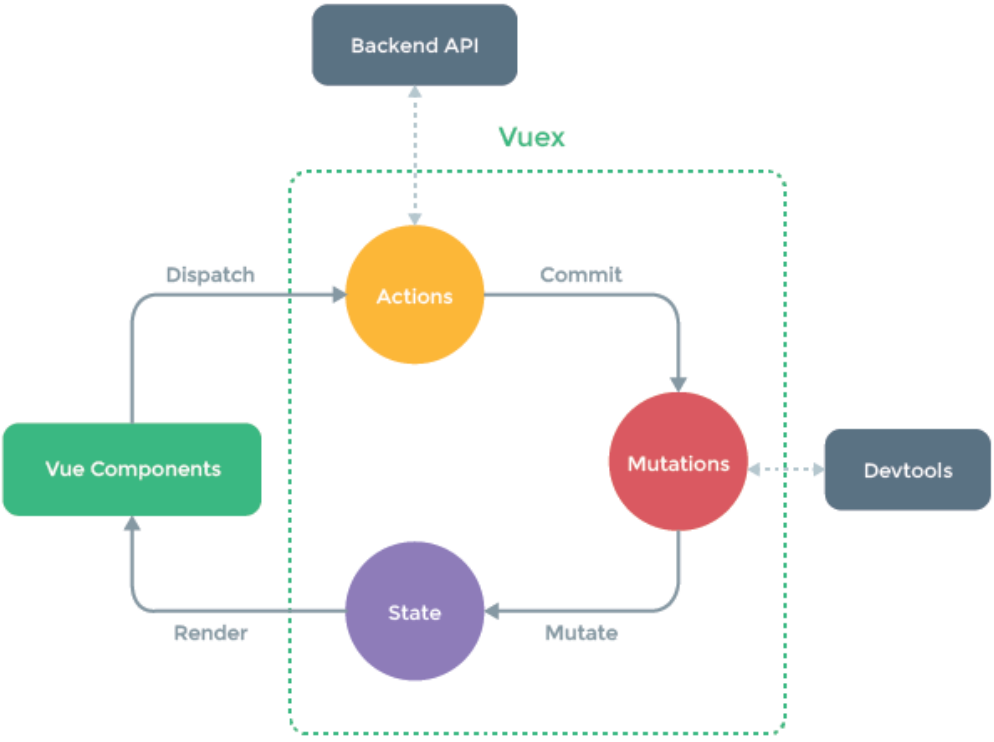
Obrázek 6 - Postup zpracování informací při jednosměrném toku dat, zdroj [29]

State: Je reaktivní úložiště sdílené skrze celou aplikaci, nebo její části. Největší výhodou sdíleného stavu je především to, že všechny komponenty mají přístup ke stejnému úložišti a je tak zajištěno snadné sdílení informací mezi komponentami a také zůstává zachována integrita aplikace.

Mutations: Jsou metody, které by jako jediné měly měnit state. Mimo mutace je možné měnit stav přímým přístupem. Tento postup ale nezajistí správnou aktualizaci stavu v celé aplikaci. Některé komponenty tak mohou mít přístup k již nepravdivým hodnotám. Tento přístup lze v konfiguraci Vuex zakázat, ale ve výchozí konfiguraci je povolený. Hlavní podmínkou pro mutace je, že musí být synchronní, nesmí obsahovat žádnou práci se vzdálenými zdroji ani jiné asynchronní operace.

Getters: Jsou metody pro získávání jednotlivých položek stavu. Výsledky těchto metod jsou uložené v mezipaměti a jejich obnovení závisí na změně stavu. Základní typy těchto metod jsou bezparametrické a nelze tak získat ze stavu například položku s určitým id. Tento problém lze řešit tím, že getter vrátí funkci s požadovanými parametry. Taková funkce bude mít stále přístup ke stavu a zároveň jí bude možné předat parametr jako třeba id. Gettry využívající tuto techniku není možné ukládat do mezipaměti a jejich výsledek je vždy počítán znovu. Při složitějších operacích, nebo velmi častých dotazech na hodnoty gettru to může znamenat výkonnostní problém.

Actions: jsou funkce používané pro komunikaci s vnějšími zdroji. Akce jsou asynchronní a mohou proto obsahovat asynchronní volání funkcí. Typické využití akce může představovat dotaz do API a následné zpracování odpovědi. V těle akce je možné volat jednotlivé Mutace, které zajistí změnu stavu. Grafické zobrazení změny stavu ve Vuex je možné vidět na obrázku 7. [30]



Obrázek 7 - Diagram změny stavu ve Vuex, zdroj [30]

3.3.2 Axios

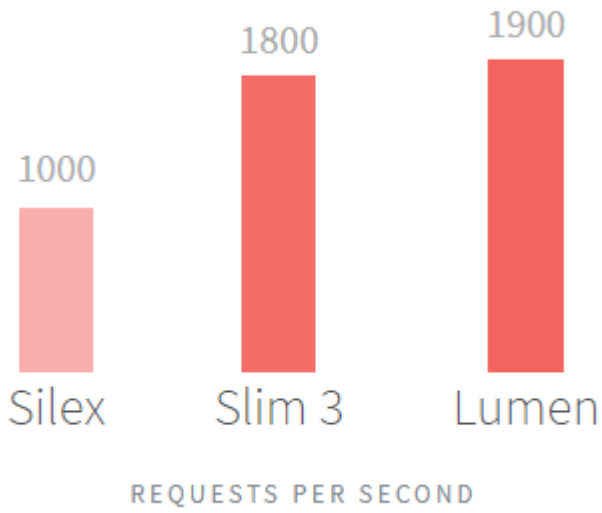
Axios je Javascriptová knihovna pro HTTP komunikaci. Umožňuje jednoduchý způsob práce s API. Celá funkce je založena na asynchronních dotazech a následném zpracování pomocí Promises. [31]

3.3.3 Element

Element je modul pro Vuejs definující užitečné HTML komponenty. Element funguje na velice podobném principu jako HTML framework Bootstrap. [32]

3.4 PHP framework Lumen

Z mnoha dostupných frameworků jsem si nakonec vybral Lumen. Bylo to především kvůli jeho rychlosti, jak ukazuje graf na obrázku 8. Jako další velikou výhodu považuji možnost využívat většinu funkcí frameworku Laravel, který dnes patří mezi nejoblíbenější PHP frameworky.



Obrázek 8 – Srovnání rychlosti vybraných PHP frameworků, zdroj [33]

Rozdělení frameworku:

Routing: Obsahuje mapování jednotlivých přístupových URL. U jednodušších dotazů může mapovací funkce obsahovat i samotnou logiku zpracování dotazu. Běžně se ale zpracování předává Kontroleru. Mapovací funkce může vracet mnoho formátů. Běžně používanými formáty jsou XML a JSON.

Middleware: Zde je možné provádět úpravy před, nebo naopak po zpracování dotazu. BeforeMiddleware je nejlepší místo pro autentizaci uživatele případně pro přesměrování na

jinou cestu. Middlewary je možné definovat globálně pro celou aplikaci, nebo je přidělit ke konkrétní cestě.

Controllers: Jsou třídy, jichž metody představují zpracování pro jednotlivé cesty. Kontrolery se specifikovanou cestou jsou poté volány jako routing callbacky.

Request: Je objekt představující HTTP request.

Response: Je objekt představující HTTP response.

3.5 REST aplikační rozhraní

Rest je architektura navržená pro přístup k datům. Autorem Rest architektury je Roy Fielding, který ji poprvé představil ve své disertační práci v roce 2002. Zatím co konkurenční technologie jako SOAP, nebo XML-RPC jsou orientované procedurálně Rest je navržen především pro práci s daty. Rest definuje několik metod pro přístup k datům. Tyto metody se velice podobají metodám protokolu HTTP je tomu tak nejspíše protože HTTP a Rest pocházejí od stejného autora. Při vytváření REST API jsou data rozdělena na jednotlivé zdroje a ke každému zdroji je přiřazena právě jedna URI. Takto vzniknou jasně definované přístupové body k jednotlivým zdrojům. Pomocí jednotlivých metod Rest následně určujeme, co se má s cílovými zdroji stát. [34]

Metody REST:

- GET – pro získání zdroje,
- POST – pro vytvoření zdroje,
- DELETE – pro smazání zdroje,
- PUT – pro aktualizaci existujícího zdroje. [34]

3.6 MySQL

MySQL je nejrozšířenější open source systém pro správu relačních databází na světě. Operace s daty je prováděna pomocí jazyka SQL.

SQL: Jazyk používaný pro komunikaci s relačními databázemi. Příkazy jazyka můžeme rozdělit do tří skupin na DML, DDL, DCL. [35]

3.7 Vývojové a sestavovací nástroje

K modernímu vývoji aplikací neodmyslitelně patří silné vývojové nástroje. Těchto nástrojů existuje veliké množství. V tomto projektu je využíván nástroj Webpack, který zajišťuje flexibilní virtuální server s možností automatické aktualizace běžící aplikace. Kromě vývojového serveru představuje Webpack silný nástroj pro sestavování dynamických zdrojů aplikace. Funkce Webpacku bude více vysvětlena v kapitole 3.7.3. Pro každý rozsáhlejší projekt je nutné využívat správce knihoven. V Javascriptu jsou jednotlivé knihovny rozděleny do balíčků. Pro správu balíčků existují balíčkovací systémy, mezi které patří například NPM, Bower, Yarn, nebo JSPM. Jako balíčkovací systém je v pro tento projekt

použit NPM. Balíčkovací systém NPM i sestavovací nástroj Webpack jsou postaveny na platformě Node.js.

3.7.1 Node.js

Node.js je Javascriptový framework, který umožňuje Javascriptu běžet na straně serveru. Jádro technologie Node.js tvoří otevřený Javascriptový compiler V8 od společnosti Google. [36]

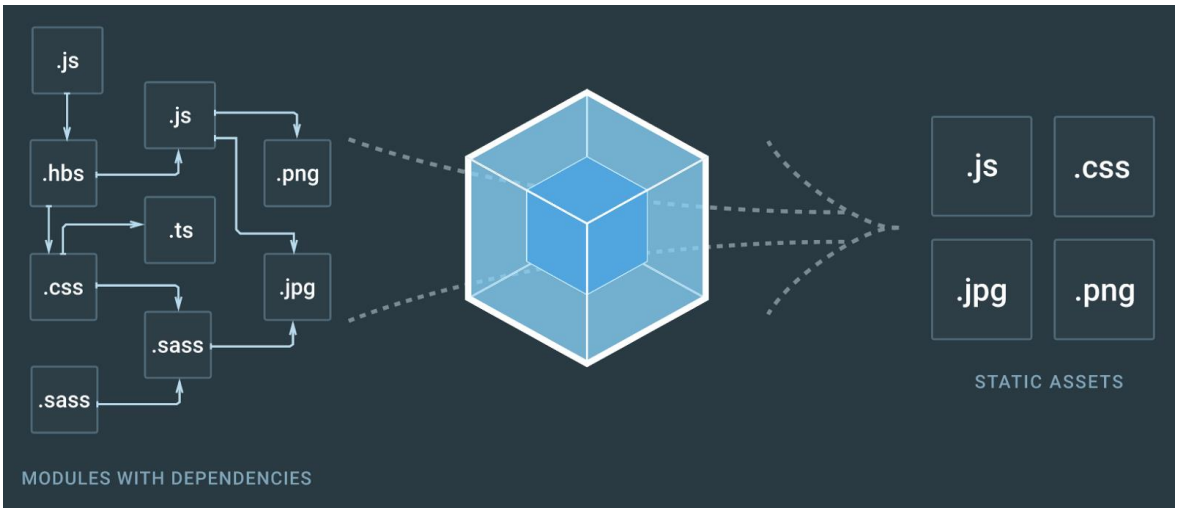
3.7.2 Node package manager

NPM je balíčkovací systém postavený na platformě Node.js. Obsahuje neuvěřitelné množství balíčků, které stále roste. Umožňuje jednoduchou správu závislostí. Seznam nainstalovaných balíčků je uložen v konfiguračním souboru package.json. V tomto souboru lze jednoduše spravovat verze jednotlivých balíčků. Užitečnou funkcí je možnost dělit balíčky na vývojové a produkční. [37]

3.7.3 Webpack

Webpack je buildovací prostředí postavené na platformě Node.js. Primárním úkolem nástroje Webpack je sestavení aplikace do spustitelné formy. Pomocí poměrně složité konfigurace je nutné nadefinovat, jak má výsledná aplikace vypadat.

Je zde možné nastavit automatickou kompilaci LESS, nebo SASS stylů, překlad z Typescriptu na JS a mnoho dalšího. Samozřejmostí je i možnost využití knihoven pro zajištění zpětné komptability ve starších prohlížečích a integrace nástrojů pro agregaci a komprimaci zdrojů. Základní funkce Webpacku je graficky znázorněna na obrázku 10. [37]



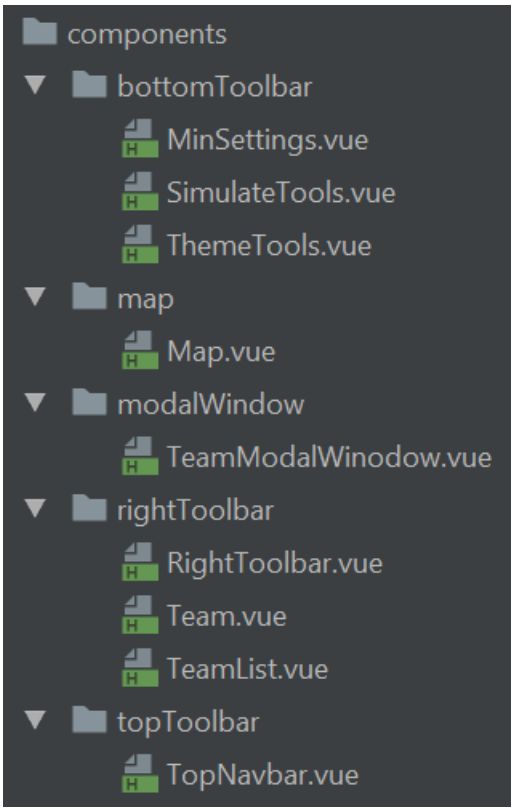
Obrázek 9 – Znázornění funkce Webpacku, zdroj [38]

4 Návrh aplikace - Front-end

4.1 Struktura aplikace

4.1.1 Komponenty

Komponenty tvoří hlavní část aplikace. Každá komponenta reprezentuje jednu část aplikace například komponenta Map.vue se stará o zobrazení mapy. Komponenty jsou rozděleny do souborů podle jejich umístění ve struktuře aplikace. Rozdělení komponent je možné vidět na obrázku 10.



Obrázek 10 - Rozdělení jednotlivých komponent do souborů, zdroj [vlastní]

Map.vue: Jak již bylo řečeno komponenta Map.vue se stará o zobrazení mapy. Pro zobrazení mapy je zde použit modul VueGoogleMaps.

SimulateTools.vue: Komponenta s názvem SimulateTools obsahuje ovládací prvky aplikace. Je zde také využita komponenta pro časovou osu s názvem VueSlider a komponenta pro nastavení vzhledu aplikace ThemeTools.

ThemeTools.vue: Je drobná komponenta pro zobrazení ovládacích prvků pro změnu vzhledu aplikace.

TeamModalWindow.vue: Obsahuje zobrazení detailu jednotlivých týmů.

RightToolbar.vue: Je obalovací komponenta pro výpis jednotlivých týmů.

TeamList.vue: Komponenta obsahující výpis týmů.

Team.vue: Představuje jednotlivé týmy a je tisknuta z mateřské komponenty TeamList.

TopNavbar.vue: Obsahuje prvky pro výběr závodu a konkrétní trasy.

4.1.2 Služby

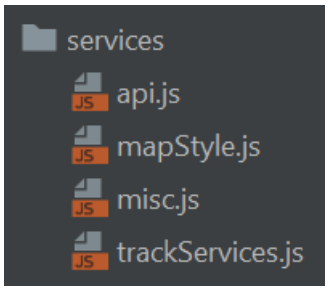
Služby jsou funkce nezávislé na jednotlivých komponentách. Každá služba je psána tak, aby jí bylo možné zavolat odkudkoliv z aplikace. Služby s podobným zaměřením jsou seskupovány do souborů. Rozdělení služeb do jednotlivých souborů je možné vidět na obrázku 11. Například soubor api.js obsahuje služby pro práci s API.

api.js: Jak již bylo zmíněno výše tento soubor obsahuje služby pro práci s API. Veškeré služby v tomto souboru pracují s instancí služby Axios.

mapStyle.js: Obsahuje jedinou službu, která má za úkol vrátit definovaný styl pro Google mapu.

misc.js: Tento soubor obsahuje obecné služby. Jednou z těchto služeb je například vygenerování náhodné barvy.

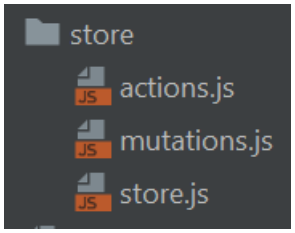
trackServices.js: Soubor obsahuje služby pro práci se simulačními daty.



Obrázek 11 - Rozdělení jednotlivých služeb do samostatných souborů, zdroj [vlastní]

4.1.3 Úložiště

Úložiště představuje objekt, který udržuje stav aplikace a sdílí jej mezi jednotlivé komponenty. Zde je tato funkce řešena pomocí knihovny Vuex. Jednotlivé části Vuex jsou rozděleny do souborů. Rozdělení zobrazuje obrázek 12.



Obrázek 12 - Rozdělení jednotlivých částí Vuex do souborů, zdroj [vlastní]

action.js: Obsahuje veškeré asynchronní operace, které lze pouštět nad stavem aplikace. V tomto případě to jsou initRace pro načtení veškerých dat o závodě a refreshRace pro načtení aktualizovaných dat o závodě.

mutation.js: Obsahuje veškeré synchronní operace, kterými je možné měnit stav aplikace. Na obrázku 13 je zobrazena definice mutace reset, která slouží pro nastavení výchozích hodnot aplikace.

```
reset(state, posadky) {
    state.posadky = state.posadky.slice(0, 0);
    state.simulation.startPositionIndex = 0;
    state.simulation.actualPositionIndex = 0;
    state.simulation.endPositionIndex = 0;
    state.raceTrack = state.raceTrack.slice(0, 0);
    state.selectedRace = null;
},
```

Obrázek 13 – Definice mutace reset, zdroj [vlastní]

Mutace je možné na stav aplikovat pomocí metody commit, která má jako první parametr název mutace. Druhý parametr metody je nepovinný a představuje argument mutace. Aplikování mutace reset je zobrazena na obrázku 14.

```
if (!idRoad) {
    this.$store.commit('reset');
    return;
}
```

Obrázek 14 - Volání mutace reset nad Vuex úložištěm, zdroj [vlastní]

4.1.4 Komunikace s webovým serverem

Pro komunikaci s webovým serverem jsou určené služby uložené v souboru api.js a knihovna Axios. Každá funkce z tohoto souboru vrací JS objekt takzvaný Promise. Promise umožnuje reagovat na výsledek asynchronního volání. Přesně takovým typem volání je dotaz do vzdáleného API. Ve chvíli kdy dostane Promise odpověď, zavolá nad získanými daty námi předem definovanou funkci, které předá získaná data asynchronního volání jako parametr. V případě dotazu přes knihovnu Axios dostáváme objekt Response, který kromě dat ve formátu JSON obsahuje také stavový kód odpovědi, informace o zasílaných http hlavičkách a mnoho jiných informací ze získané HTTP odpovědi.

getZavody: Získá ze serveru seznam závodů.

getTrasy: Získá seznam tras. Jako parametr je nutné uvést id závodu.

zavodPosadky: Získá veškerá data potřebná pro simulaci. Jako parameter je nutné uvést id závodu.

4.2 Výběr závodu a trasy

První akcí uživatele v aplikaci je vždy výběr jednoho závodu a poté jedné trasy. Možnosti pro výběr závodu a trasy jsou dynamicky generovány. Jako první se načte seznam všech závodů, které mají vyplněné alespoň jednu trasu. Seznam tras zůstává skrytý a zobrazí se teprve až uživatel zvolí některý závod. Podle zvoleného závodu aplikace načte odpovídající seznam tras a zároveň vybere první trasu jako aktivní a načte data pro simulaci. Obrázek 15 zobrazuje dynamicky generované selecty za použití komponent el-select a el-option.

```
<el-select v-model="selectedRace" clearable placeholder="Vyberte závod">
    <el-option
            v-for="item in zavodyOptions"
            :key="item.value"
            :label="item.label"
            :value="item.value">
    </el-option>
</el-select>

<el-select v-if="trasy.length" v-model="selectedTrace" clearable placeholder="Vyberte trať">
    <el-option
            v-for="item in trasyOptions"
            :key="item.value"
            :label="item.label"
            :value="item.value">
    </el-option>
</el-select>
```

Obrázek 15 – Ukázka šablony obsahující dynamické selecty, zdroj [vlastní]

4.3 Modelování dat

Modelování dat je velmi důležitou částí aplikace. Modelování dat má na starosti funkce raceAnimationPoints jejími argumenty jsou naměřená data ze sledovacího zařízení, trasa závodu a míra zjemnění. Jako první funkce zjemní trasu závodu. Následně jsou procházeny jednotlivé naměřené body. Nad každým párem bodů jsou prováděny potřebné operace, které budou popsány v následujících kapitolách. Výsledkem jsou data, které již není během simulace nutné měnit. To velice šetří výkon počítače a také umožnuje běh celé simulace bez připojení k síti. Připojení je vyžadováno v případě změny sledovaného závodu, nebo při automatické aktualizaci dat.

4.3.1 Zjemňování dat

Jelikož zařízení zasílá informace o poloze každých 10 s nemohou být data bez náležitého zjemnění použita pro plynulou simulaci. Je nutné podotknout, že výší zjemnění znamená vyšší náročnost při generování dat. Výsledná data mohou být výrazně větší a zabírat tak více místa v paměti. Pro slabší počítač proto není vhodné volit zjemnění menší než 2 souřadnice za 1 s což znamená že dopočítáme souřadnice po intervalu 500 ms.

Pro výpočet zjemnění mezi dvěma body je nutné nejprve zjistit rozdíl mezi jejich časy. Rozdíl je následně vydělen zjemněním a zaokrouhlen na celé číslo. Výsledné číslo představuje počet bodů, které je nutné vytvořit, aby došlo k požadovanému zjemnění.

Následuje výpočet rozdílu pro souřadnice Lat a Lng mezi jednotlivými body. Získané rozdíly souřadnic je nutné vydělit mírou zjemnění tím vznikne velikost přírůstku pro jeden přidaný bod. Posledním krokem je postupné generování zjemňujících bodů. Jejich počet bude odpovídat dříve vypočítanému číslu. Každý bod bude označován pořadím začínajícím od 0. Pro jednotlivé body lze definovat čas a pozice Lat a Lng jednoduchým výpočtem. Pro výpočet času je nutné k času prvního bodu přičíst vypočtený časový přírůstek vynásobený pořadovým číslem bodu. Pro souřadnice je postup téměř stejný jen je potřeba místo údaje pro čas použít údaje pro příslušnou souřadnici. Na obrázku 16 je zobrazena funkce pro modelování dat.

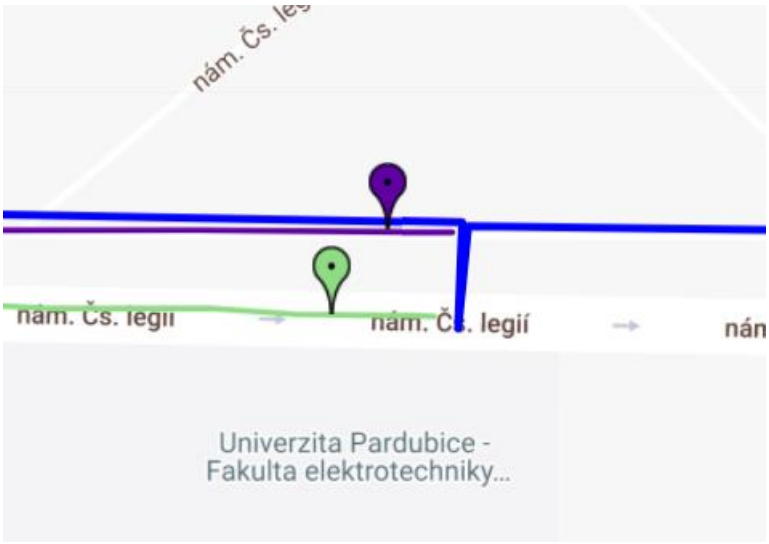
```
const poz1 = raceTrack[i], poz2 = raceTrack[i + 1];

const time = Math.round(moment(poz1.timestamp).valueOf() / ZJEMNENI_NA_SEC) * ZJEMNENI_NA_SEC;
const endTime = Math.round(moment(poz2.timestamp).valueOf() / ZJEMNENI_NA_SEC) * ZJEMNENI_NA_SEC;

const timeDiff = endTime - time;
const pocetKrokuPoStoMs = mathjs.round(timeDiff / ZJEMNENI_NA_SEC);

const rozdilLat = (poz2.lat - poz1.lat) / pocetKrokuPoStoMs;
const rozdilLng = (poz2.lng - poz1.lng) / pocetKrokuPoStoMs;

const speed = getSpeed(poz1, poz2);

for (let i = 0; i < pocetKrokuPoStoMs; i++) {
    const poz1Lat = poz1.lat + i * rozdilLat;
    const poz1Lng = poz1.lng + i * rozdilLng;


    newPositions[time + ZJEMNENI_NA_SEC * i] = {
        lat: poz1Lat,
        lng: poz1Lng,
        speed: speed
    };

    let minTrack = R.last(newRoad.sort((item, item2) => {
        const a = Math.abs(item.lat - poz1Lat) + Math.abs(item.lng - poz1Lng);
        const b = Math.abs(item2.lat - poz1Lat) + Math.abs(item2.lng - poz1Lng);

        return b - a;
    }));

    newPositions[time + ZJEMNENI_NA_SEC * i].minTrack = minTrack;
}
```

Obrázek 16 - Ukázka kódu sloužícího pro zjemnění GPS dat, zdroj [vlastní]

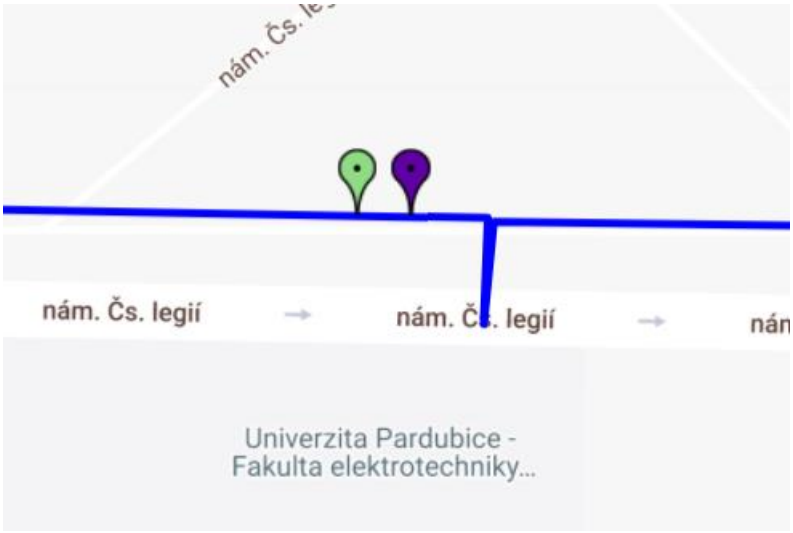
4.3.2 Zobrazení závodníků na trase

Originální data, které zařízení zasílá nejsou položena na souřadnicích trasy závodu. Pro zobrazení pozice závodníka na trase závodu je nutné jeho souřadnice aproximovat na souřadnice trasy závodu. Základní princip výpočtu spočívá v přiřazování ke každému bodu

trasy závodníka nejbližší bod trasy závodu. Tento bod je poté uložen jako pozice závodníka na trati závodu. Uložení se uskutečňuje při prvním zpracování dat a je tak možné kdykoliv během závodu přepínat mezi naměřenými souřadnicemi a souřadnicemi aproximovanými k trase závodu. Na obrázku 17 je jsou zobrazené originální polohy závodníků. Na obrázku 18 jsou polohy závodníků upravené na trasu závodu.



Obrázek 17 – Zobrazení originální polohy závodníků, zdroj [vlastní]



Obrázek 18 - Originální poloha závodníků je upravena pro trasu závodu, zdroj [vlastní]

4.3.3 Výpočet rychlosti mezi body

Pro každý bod je nutné vypočítat rychlost. K výpočtu rychlosti jsou potřeba vždy dva body. Nejprve je vypočtena vzdálenost mezi body a převedena na km, následně je vydělena časem v hodinách. Výsledná hodnota je rychlost v km / h

4.3.4 Výpočet celkové průměrné rychlosti

Pro každý tým je vypočítána celková průměrná rychlost. Při výpočtu se prochází všechny naměřené body a sčítá se jejich rychlost nakonec se celková rychlost vydělí počtem naměřených bodů.

4.3.5 Výpočet průměrné nadmořské výšky

Stejně jako je tomu u rychlosti se vypočítává celková průměrná nadmořská výška.

4.4 Vykreslování mapy

Vykreslování mapy je hlavní funkce této aplikace. Pro tento náročný úkol bylo vybráno již dříve popsané Google map API a jeho integrace do VueJs.

Modul vue-google-maps je hlavní integrací Google map do Vuejs. Celý modul funguje na principu obalování objektů z Google map API Vuejs komponentami.

Gmap-map: Je hlavní komponenta pro Google mapu. Jejím úkolem je vykreslit mapu do předem definovaného elementu. Element musí mít nenulovou výšku. Pro komponentu mapy leze zadat několik parametrů jako je například střed mapy, typ mapy, přiblížení, nebo styl mapy.

Gmap-marker: Je komponenta představující jednotlivé Týmy. Pro každou komponentu tohoto typu je nutné zadávat aktuální pozici na mapě. Jako další parametry je dobré nastavit možné operace jako je kliknutí, nebo táhnutí ikony.

GmapPolyLine: Je komponenta pro vykreslení trasy. Povinným parametrem je pole souřadnic, které určuje, kde se má trasa vykreslit. Dále je možné určit například šířku nebo barvu trasy.

4.5 Přehrávání závodu

Pro možnost simulace je nutné udržovat počáteční čas a konečný čas simulace v milisekundách. Obě tyto informace lze spočítat na základě naměřených dat. Pro určení počátečního času simulace je nutné projít naměřené souřadnice závodníků a nalézt souřadnici s nejmenším časem. Pro určení konečného času je také nutné projít souřadnice závodníků a nalézt souřadnici s největším časem. Pokud známe konečný a počáteční čas je potřeba ještě určit aktuální čas simulace. Ten se bude měnit na základě průběhu simulace, ale na začátku je stejný jako počáteční čas simulace.

Průběh simulace je založen na pravidelném zvyšování aktuálního času simulace. Interval mezi jednotlivými navýšeními je 100 ms a velikost navýšení závisí na rychlosti simulace. Proč ale právě 100 ms. Vyšší interval by již nezajišťoval dostatečnou plynulost simulace a

nižší interval by mohl mít u slabších počítačů negativní vliv na výkon. Je nutné si uvědomit, že ne pro každou hodnotu aktuálního času jsou k dispozici pozice závodníků. Například pokud by byla data zobrazena bez jakéhokoliv zjemnění. Docházelo by ke změně polohy týmu zhruba jednou za 10 s. Pokud pro aktuální simulační čas nejsou nalezeny souřadnice některého z týmů, zůstává tým na posledních známých souřadnicích.

4.5.1 Základní ovládací prvky

Pro ovládání simulace nabízí aplikace několik základních ovládacích prvků.

- pozastavit aplikaci – umožní pozastavit přehrávání aplikace,
- zastavit přehrávání – zastaví přehrávání aplikace a nastaví aktuální čas simulace na počáteční čas simulace,
- spustit / pokračovat přehrávání – spustí přehrávání simulace, pokud je simulace pozastavena bude pokračovat v přehrávaní od bodu, kde byla pozastavena,
- opakované přehrávání – pokud je možnost aktivní poběží simulace v nekonečném cyklu,
- změna rychlosti přehrávání – umožňuje měnit rychlost přehrávání na předem definované rychlosti.

4.5.2 Automatická aktualizace dat

Pokud dosáhne aktuální čas simulace konečného času simulace a je vypnuta možnost opakovaného přehrávání. Pokusí se aplikace každých 5 s získat nové souřadnice závodníků. Při určení intervalu bylo nutné se zamyslet nad problémem častého dotazování na server. Pokud by interval byl například 0,5 s mohlo by to znamenat značné vytížení serveru. Jelikož zařízení posílá nová data jednou za 10 s mohl by být interval na straně aktualizace stejný. V potaz je nutno vzít, že zařízení nebude pouze jedno, a tak se interval pro aktualizaci dat nebude pohybovat vždy kolem 10 s. Nastavení intervalu na 5 s se zdá být kompromisním řešením.

4.6 Nastavení týmů

Aplikace kromě ovládacích prvků pro simulaci obsahuje několik nastavení pro posádky, které se účastní závodu. Jednotlivá nastavení mají za úkol zpříjemnit a zjednodušit uživateli práci s aplikací.

4.6.1 Oblíbený tým

Uživatel si může jeden tým označit jako oblíbený. Může tak učinit kliknutím na ikonu hvězdy ve výpise týmů, nebo přímo na ikonu týmu na mapě. Takový tým je ve výpisu zvýrazněn a mapa je automaticky centrována na jeho polohu.

4.6.2 Zobrazit vybrané týmy

Aplikace umožňuje zobrazit pouze uživatelem vybrané týmy. Funkce je napsána na principu filtrovaného pole. Jednotlivé týmy jsou uloženy v poli v globálním stavu aplikace. Data pro výpis jednotlivých týmů na mapu jsou filtrována podle vybraných týmů. Nechybí ani možnost hromadného nastavení pro zobrazení případně skrytí všech týmů najednou.

4.6.3 Detail týmu

Pro každý tým je možné zobrazit modální okno s detailními informacemi o týmu. Jsou zde k dispozici přehledné grafy zobrazující nadmořskou výšku a rychlost týmu v čase závodu.

Pro zobrazení grafu se využívá modulu VueChartKick. Jedná se o integraci knihoven Google chart, Chartjs a Highcharts do VueJs.

5 Návrh aplikace - Back-end

Následující část bude věnována popisu serverové části aplikace, která poskytuje data, pro již popsanou klientskou část aplikace.

5.1 Struktura aplikace

Serverová část aplikace se skládá ze dvou částí. První část je tvořena aplikačním rozhraním a druhá část představuje databázi ve které jsou uložena veškerá data se kterými aplikace pracuje.

5.1.1 Struktura aplikačního rozhraní

Základem celého aplikačního rozhraní jsou URL, které definují přístupové body pro jednotlivé zdroje dat. Aplikace definuje tři zdroje dat a ke každému přiřazuje právě jednu URL. Pro komunikaci s API je možné využívat pouze HTTP metody GET pro získání dat a metody HEAD. Jelikož klientská aplikace neumožnuje zápis dat jsou ostatní metody http protokolu zakázány. Veškeré dotazy na zdroje aplikačního rozhraní se špatnou http metodou mají návratový status kód 405.

Závody: Prvním zdrojem jsou závody. Lze získat seznam všech závodů, nebo pouze jeden závod.

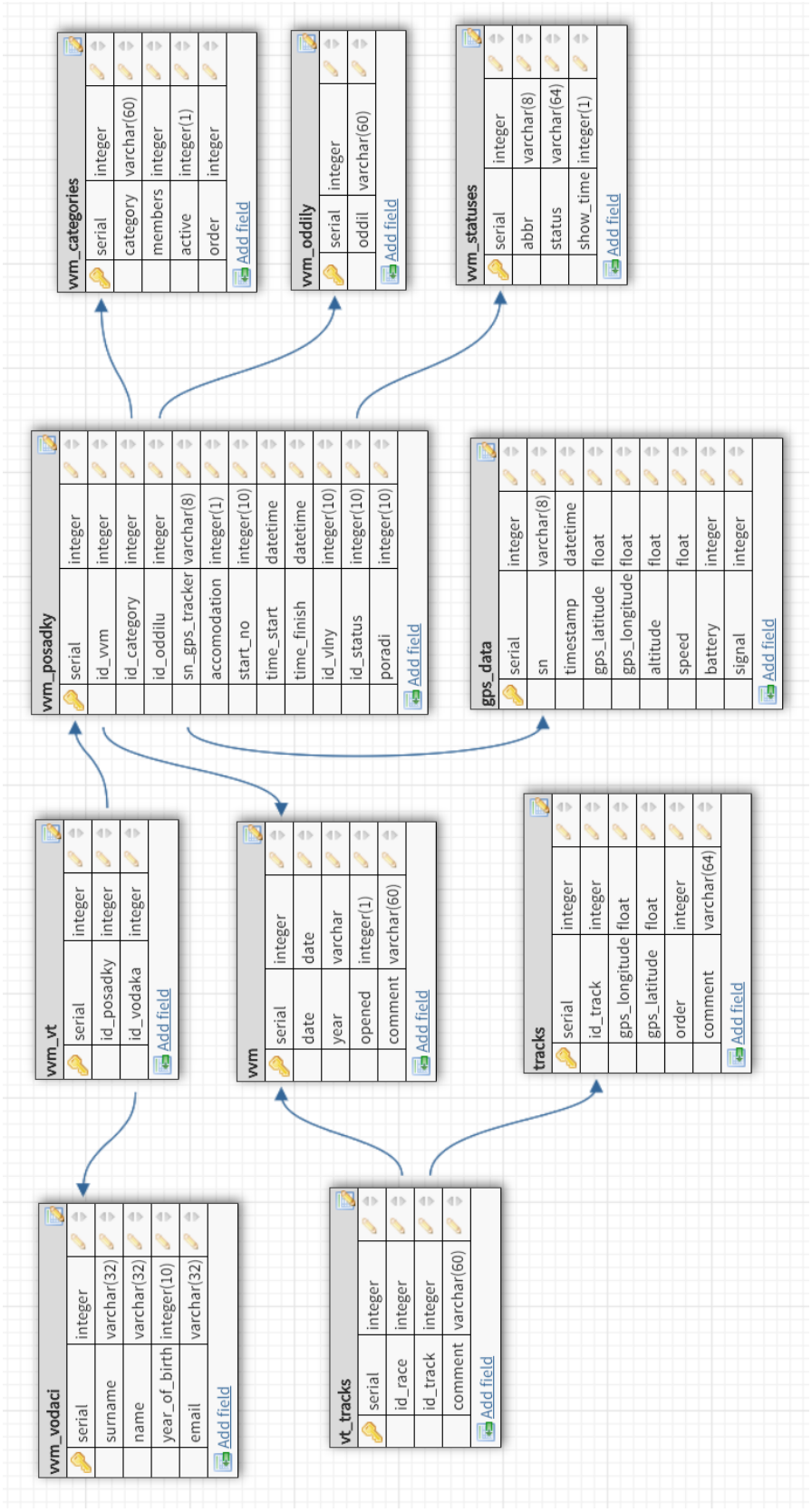
Trasy: Druhý zdroj jsou trasy jednotlivých závodů. Opět je možné získat seznam všech tras, nebo pouze jednu trasu. Také je tu možnost získat trasy pouze od jednoho závodu.

Data pro simulaci: Posledním zdrojem jsou data potřebná k simulaci. Získaná data jsou rozdělena podle týmů. Pro každý tým jsou získány GPS souřadnice pro aktuálně vybranou trasu závodu.

AppController: Je soubor obsahující funkce, které mají za úkol získat data pro jednotlivé zdroje a vrátit je ve formátu JSON. Tyto funkce jsou volány po přístupu na jednu z adres aplikačního rozhraní.

5.2 Struktura zdrojové databáze

Databáze, která slouží jako zdroj dat pro aplikaci obsahuje celkem deset tabulek. Strukturu jednotlivých tabulek a vztahy mezi nimi zobrazuje diagram na obrázku 19.



Obrázek 19 – Diagram zobrazující jednotlivé tabulky zdrojové databáze a vztahy mezi nimi, zdroj [vlastní]

5.3 Nastavení mezipaměti

Nastavení mezipaměti je pro API velice důležité a jeho zanedbání se může nepříjemně projevit na vytížení webového serveru, ale i databáze. Prvním krokem je ukládání vygenerovaných výsledků do tabulky cache, která je určena pro statickou cache. Ukládání výsledků probíhá na základě dotazované URL. Pro URL, která slouží pro získání seznamu závodů a seznamu tras je možné nastavit expirace mezipaměti na 1 hodinu, jelikož k aktualizaci těchto dat bude docházet pouze velmi zřídka. U dotazů na url, která má za úkol generovat data pro simulaci je nutné nastavit maximální čas expirace maximálně na 5 s což odpovídá intervalu dotazování z aplikace.

5.4 Zabezpečení

U běžných SPA aplikací by bylo nutné řešit autentizaci uživatele například knihovnou OAuth2. V tomto konkrétním případě není nutné autentizaci uživatele provádět, jelikož se jedná o aplikaci, která data pouze získává a nemá oprávnění pro zápis dat. I když aplikace nemá oprávnění pro zápis dat je tu stále nebezpečí útoku typu DDoS na URL API. Těmto útoků lze předcházet především správným nastavením Firewallu, ale také využitím mezipaměti na straně serveru, která dokáže ukládat vygenerovaný výstup pro jednotlivé URL adresy a minimalizovat tak režii serveru na pouhé odesílání již vygenerovaných dokumentů. Takovéto akce pro server neznamenají oproti procesům, kdy je nutné zajistit vygenerování výstupu pomocí PHP téměř žádnou náročnost. Nastavení mezipaměti na straně serveru sice DDoS útoku nezabrání, ale může zmírnit jeho následky.

Závěr

Úvod práce poukázal na existenci velikého množství vzájemně nezávislých systémů pro satelitní navigaci. Lidé tak nejsou odkázáni pouze na pokrytí jednoho systému, jak tomu bylo dříve. I když jsou technologie pro geolokaci dostupné pro každého, není jejich využití pro sportovní události příliš rozvinuto. Projektů pro toto využití neexistuje takové množství, jaké by se dalo očekávat. Například pro Českou republiku byl nalezen pouze jeden systém, který umožňuje sledovat polohu závodníka při závodě.

Vytvořená aplikace je plně funkční a svými funkcemi se vyrovná existujícím projektům. Nabízí funkce pro sledování závodníků v téměř reálném čase i možnost přehrávat pohyb zpětně. Obsahuje také možnosti pro kontrolu sledovaných závodníků, nebo možnost zvolit si oblíbeného závodníka. Technologie vybrané pro realizaci aplikace se ukázaly jako správná volba. Jejich využití se obešlo bez větších komplikací. Největším přínosem byl framework Vuejs, který velmi usnadnil vývoj aplikace. Realizaci práce velmi napomohla existence Vuejs implementace pro Google mapy, která urychlila celý proces vývoje. Nejobtížnější částí vývoje byly funkce pro zpracování GPS dat, které se nakonec podařilo úspěšně naimplementovat. Unikátní je funkce pro aproximaci GPS souřadnic závodníků na trasu závodu.

Aplikace je navržena s možností dalšího rozvoje. Do budoucna je v plánu příprava administrace pro jednotlivé závody a trasy. Nyní neexistuje žádná administrace a je nutné pracovat s daty přímo v databázi (např. přes rozhraní programu phpMyAdmin). Tato funkce by celý proces práce s daty zpříjemnila a zpřístupnila i pro méně technicky zaměřené uživatele. Dalším uvažovaným rozvojem je umožnit aplikaci využívat na mobilu jako nativní aplikaci za použití knihovny Nativescript a její integrace do Vuejs.

Práce ukázala některé zajímavé technologie pro psaní moderních SPA aplikací, které se stávají ve světě vývoje webových stránek stále oblíbenější. Autor během zpracování podkladů pro tuto práci získal větší přehled o Javascriptových frameworcích, než měl doposud. Největším přínosem pro autora byl postup při samostatném návrhu celé aplikace. Tato zkušenost je velice cenná pro budoucí projekty.

Literatura

[1] AGRAWAL, Subham. Glonass vs GPS vs Beidou: Complete Guide to Navigation Systems. *Agatton* [online]. Agatton, 2016 [cit. 2018-05-06]. Dostupné z: https://agatton.com/glonass-vs-gps-vs-beidou-complete-guide-navigation-systems

[2] GPS. *GPS - Stránka o satelitní navigaci* [online]. 2014 [cit. 2018-05-06]. Dostupné z: http://gps.slansko.cz

[3] Space Segment. *GPS.gov* [online]. USA: National Coordination Office for Space-Based Positioning, Navigation, and Timing, 2018 [cit. 2018-05-06]. Dostupné z: https://www.gps.gov/systems/gps/space

[4] Control Segment. *GPS.gov* [online]. USA: National Coordination Office for Space-Based Positioning, Navigation, and Timing, 2018 [cit. 2018-05-06]. Dostupné z: https://www.gps.gov/systems/gps/control

[5] Americký družicový navigační systém NAVSTAR GPS. *Český kosmický portál* [online]. Praha: Odbor ITS, kosmických aktivit a VaVaI, 2017 [cit. 2018-05-06]. Dostupné z: http://www.czechspaceportal.cz/3-sekce/gnss-systemy/gnss-mimo-evropu/americky-navstar-gps

[6] Ruský globální družicový navigační systém GLONASS. *Český kosmický portál* [online]. Praha: Odbor ITS, kosmických aktivit a VaVaI, 2017 [cit. 2018-05-06]. Dostupné z: http://www.czechspaceportal.cz/3-sekce/gnss-systemy/gnss-mimo-evropu/rusky-glonass

[7] GALILEO - Evropský globální navigační družicový systém. *Český kosmický portál* [online]. Praha: Odbor ITS, kosmických aktivit a VaVaI, 2017 [cit. 2018-05-06]. Dostupné z: http://www.czechspaceportal.cz/3-sekce/gnss-systemy/galileo

[8] Čínský navigační systém Beidou / Compass. *Český kosmický portál* [online]. Praha: Odbor ITS, kosmických aktivit a VaVaI, 2017 [cit. 2018-05-06]. Dostupné z: http://www.czechspaceportal.cz/3-sekce/gnss-systemy/gnss-mimo-evropu/cinsky-beidou---compass

[9] Follow Me. *Follow Me* [online]. Follow Me, 2018 [cit. 2018-05-06]. Dostupné z: http://cs.follow.me.cz/funkce

[10] Tracking - DotVision Motion. *DotVision Motion* [online]. Moissy-Cramayel (France): DotVision Motion, 2018 [cit. 2018-05-06]. Dostupné z: https://motion.dotvision.com/Solutions/Tracking

[11] Live GPS Tracking. *Follow My Challenge* [online]. VG Voorschoten (The Netherlands): Follow My Challenge, 2015 [cit. 2018-05-06]. Dostupné z: https://www.followmychallenge.com/features

[12] Get event - Sailing. *TracTrac* [online]. Lyngby (Denmark): TracTrac ApS, 2017 [cit. 2018-05-06]. Dostupné z: http://www.tractrac.com/web/get-event

[13] VYLEŤAL, Martin. Souboj mapových titánů: vyrovnají se nové Mapy.cz mapám Googlu?. *Lupa.cz* [online]. Praha: Internet Info, 2014 [cit. 2018-05-06]. Dostupné z: https://www.lupa.cz/clanky/souboj-mapovych-titanu-vyrovnaji-se-nove-mapy-cz-mapam-googlu

[14] WASSON, Mike. ASP.NET - Single-Page Applications: Build Modern, Responsive Web Apps with ASP.NET. *MSDN Magazine*[online]. Redmond (Washington): Microsoft, 2013 [cit. 2018-05-06]. Dostupné z: https://msdn.microsoft.com/en-us/magazine/dn463786.aspx

[15] SKÓLSKI, Paweł. Single-page application vs. multiple-page application. *Neoteric* [online]. Gdańsk (POLAND): NEOTERIC SP Z O O, 2017 [cit. 2018-05-06]. Dostupné z: https://neoteric.eu/single-page-application-vs-multiple-page-application

[16] What exactly is Angular 2?. *Quora* [online]. Mountain View (California): Quora, 2017 [cit. 2018-05-06]. Dostupné z: https://www.quora.com/What-exactly-is-Angular-2

[17] Angular 2 pros and cons. *Andrea Fiori* [online]. Florence (Italy): ANDREAFIORI.NET, 2017 [cit. 2018-05-06]. Dostupné z: https://www.andreafiori.net/posts/angular-2-pros-and-cons

[18] MURRAY, Nate, Felipe COURY, Ari LERNER a Carlos TABORDA. *Ng-book: The Complete Guide to Angular 4*. 4 edition. San Francisco (California): CreateSpace Independent Publishing Platform, 2017. ISBN 978-1546376231.

[19] DOMES, Scott. Everything You Should Know About React: The Basics You Need to Start Building. *Medium* [online]. Medium Corporation, 2017 [cit. 2018-05-06]. Dostupné z: https://medium.freecodecamp.org/everything-you-need-to-know-about-react-eaedf53238c4

[20] Introduction. *Vue.js* [online]. New York (USA): Evan You, 2018 [cit. 2018-05-06]. Dostupné z: https://vuejs.org/v2/guide

[21] MALOS, Pavel. Impressions about VueJs & ReactJS. What people think…. *BootstrapBay* [online]. Buzau (Romania): BootstrapBay, 2017, May 19, 2017 [cit. 2018-05-06]. Dostupné z: https://bootstrapbay.com/blog/vuejs-vs-reactjs

[22] FILIPOVA, Olga. *Learning Vue.js 2: Learn how to build amazing and complex reactive web applications easily with Vue.js*. December 13, 2016. Birmingham: Packt Publishing, 2016. ISBN 978-1-78646-994-6.

[23] DOYLE, Matt. What Is jQuery?. *Elated* [online]. Elated Communications, 2010, 4 February 2010 [cit. 2018-05-06]. Dostupné z: https://www.elated.com/articles/what-is-jquery

[24] BILAL SHAREEF, Mohammed. How to Build REST API Using PHP. *Apptha* [online]. Čennai (India): apptha.com, 2015, November 27, 2015 [cit. 2018-05-06]. Dostupné z: https://www.apptha.com/blog/how-to-build-a-rest-api-using-php

[25] MONUS, Anna. PHP 7: 10 Things You Need to Know. *Hongkiat* [online]. Hongkiat.com, 2018 [cit. 2018-05-06]. Dostupné z: https://www.hongkiat.com/blog/php7

[26] SANDOVAL, Kristopher. 5 Lightweight PHP Frameworks to Build REST APIs. *Nordic APIs* [online]. Stockholm (Sweden): Nordic APIs AB, 2017, June 6th, 2017 [cit. 2018-05-06]. Dostupné z: https://nordicapis.com/5-lightweight-php-frameworks-build-rest-apis

[27] MARDAN, Azat. Top 10 ES6 Features Every Busy JavaScript Developer Must Know. *Webapplog: programming weblog* [online]. Azat Mardan, 2015 [cit. 2018-05-06]. Dostupné z: https://webapplog.com/es6

[28] HORIE, Leo. *Mithril.js* [online]. Leo Horie, 2018 [cit. 2018-05-06]. Dostupné z: http://lhorie.github.io/mithril-presentation-duolingo

[29] ROUBÍČEK, Aleš. Potřebujeme Flux?. *Zdroják* [online]. Praha: Devel.cz Lab, 2018, 16.3.2015 [cit. 2018-05-06]. Dostupné z: https://www.zdrojak.cz/clanky/potrebujeme-flux

[30] What is Vuex?. *Vuex* [online]. 2018, 16.3.2015 [cit. 2018-05-06]. Dostupné z: https://vuex.vuejs.org/en/intro.html

[31] ESCHWEILER, Sebastian. Getting Started With Axios: Accessing REST Web Services / HTTP APIs in JavaScript. *Medium* [online]. New York: Medium Corporation, 2018, Mar 7, 2017 [cit. 2018-05-06]. Dostupné z: https://medium.com/codingthesmartway-com-blog/getting-started-with-axios-166cb0035237

[32] Element. *Element* [online]. Element, 2018 [cit. 2018-05-06]. Dostupné z: http://element.eleme.io

[33] Lumen - PHP Micro-Framework By Laravel. *Lumen* [online]. Taylor Otwell, 2018 [cit. 2018-05-06]. Dostupné z: https://lumen.laravel.com

[34] MALÝ, Martin. REST: architektura pro webové API. *Zdroják* [online]. Praha: Devel.cz Lab, 2009, 3.8.2009 [cit. 2018-05-06]. Dostupné z: https://www.zdrojak.cz/clanky/rest-architektura-pro-webove-api

[35] About MySQL. *MySQL* [online]. Redwood City (California): Oracle Corporation, ©2018 [cit. 2018-05-06]. Dostupné z: https://www.mysql.com/about

[36] PATEL, Priyesh. What exactly is Node.js?. *Medium* [online]. New York: Medium Corporation, 2018, Apr 18, 2018 [cit. 2018-05-06]. Dostupné z: https://medium.freecodecamp.org/what-exactly-is-node-js-ae36e97449f5

[37] GASIMZADA, Gasim. What are NPM, Yarn, Babel, and Webpack; and how to properly use them?. *Medium* [online]. New York: Medium Corporation, 2017, May 10, 2017 [cit. 2018-05-06]. Dostupné z: https://medium.com/front-end-hacking/what-are-npm-yarn-babel-and-webpack-and-how-to-properly-use-them-d835a758f987

[38] METNEW, Vladimir. Webpack loaders and plugins for your new Progressive Web App. *HACKERNOON* [online]. New York: HACKERNOON, 2014 [cit. 2018-05-06]. Dostupné z: https://hackernoon.com/webpack-loaders-and-plugins-for-your-new-progressive-web-app-378e09f469

[39] ORLICH, M. Základní lokalizační metody v GSM. *Access Server* [online]. České vysoké učení technické v Praze. Katedra telekomunikační techniky, 2006, 28. 02. 2006, **16** [cit. 2018-05-07]. ISSN 1214-9675. Dostupné z: http://access.feld.cvut.cz/view.php?cisloclanku=2006022801