Univerzita Pardubice

Fakulta elektrotechniky a informatiky

Aplikační využití geodat z API geosociálních sítí a z API o COVID-19 s doporučovacím systémem při overtourismu

Filip Odvárka

Bakalářská práce

2021

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: 2020/2021

ZADÁNÍ BAKALÁŘSKÉ PRÁCE
(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Filip Odvárka**
Osobní číslo:	**I18168**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Téma práce:	**Aplikační využití geodat z API geosociálních sítí a z API o COVID-19 s doporučovacím systémem při overtourismu**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování

Cílem bakalářské práce bude v teoretické části analyzovat využitelnost dostupných API geosociálních sítí, poskytujících data o turistických bodech zájmu a také API poskytujících data o nakažených virem COVID-19. Součástí práce bude i přehled vybraných API, identifikace jejich výhod a nevýhod.
Druhým cílem v praktické části bakalářské práce bude navrhnout a vyvinout webovou aplikaci využívající data z vybraných API, inovativním způsobem poskytující turistům informace o návštěvnosti daného bodu zájmu s doporučovacím systémem, zohledňujícím zadaná kritéria (zejména minimalizaci overtourismu daného turistického bodu zájmu a podmínek pro šíření infekčních nemocí, počet nakažených v oblasti, opatření v dané oblasti).

Rozsah pracovní zprávy: **35**
Rozsah grafických prací:
Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

- LI, Yanhua, et al. *Region sampling and estimation of geosocial data with dynamic range calibration*. In: 2014 IEEE 30th International Conference on Data Engineering. IEEE, 2014. p. 1096-1107. ISBN 9781479925544.
- HOROZOV, Tzvetan; NARASIMHAN, Nitya; VASUDEVAN, Venu. *Using location for personalized POI recommendations in mobile environments*. In: International Symposium on Applications and the Internet (SAINT'06). IEEE, 2006. p. 6 pp.-129. ISBN 0-7695-2508-3.
- KYSELA, Jiří. *A Comparison of Text String Similarity Algorithms for POI Name Harmonisation*. In: International Conference on Articulated Motion and Deformable Objects. Springer, Cham, 2018. p. 121-130. ISBN 978-3-319-94544-6.
- WANG, Yan, et al. *Discovery of accessible locations using region-based geo-social data*. World Wide Web, 2019, 22.3: 929-944.
- CONOLLY, Thomas, Carolyn E. BEGG a Richard HOLOWCZAK. *Mistrovství – databáze: profesionální průvodce tvorbou efektivních databází*. Brno: Computer Press, 2009. ISBN 978-80-251-2328-7.
- PASSAGLIA, Andrea. *Vue.js 2 Cookbook*. Packt Publishing Ltd, 2017. ISBN 9781786465061.
- TATROE, Kevin; MACINTYRE, Peter. *Programming PHP: Creating Dynamic Web Pages*. O'Reilly Media, 2020. ISBN 9781449365837.

Vedoucí bakalářské práce: **Ing. Jiří Kysela, Ph.D.**
Katedra informačních technologií

Datum zadání bakalářské práce: **31. října 2020**
Termín odevzdání bakalářské práce: **14. května 2021**

L.S.

Ing. Zdeněk Němec, Ph.D. v.r.
děkan

Ing. Jan Panuš, Ph.D. v.r.
vedoucí katedry

V Pardubicích dne 26. února 2021

Prohlašuji:

Tuto práci jsem 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.

Beru na vědomí, že v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, a směrnicí Univerzity Pardubice č. 9/2012, bude práce zveřejněna v Univerzitní knihovně a prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne 10. 5. 2021

Filip Odvárka

PODĚKOVÁNÍ

Tímto bych chtěl poděkovat vedoucímu mojí bakalářské práce Ing. Jiřímu Kyselovi, Ph.D. za pomoc při zpracování bakalářské práce. V neposlední řadě bych chtěl také poděkovat své přítelkyni, která mi byla v době zpracování práce velkou oporou. Velký dík patří i mojí rodině, od které jsem vždy dostal podporu, kterou jsem potřeboval.

ANOTACE

Bakalářská práce s názvem „Aplikační využití geodat z API geosociálních sítí a z API o COVID-19 s doporučovacím systémem při overtourismu“ se zabývá analýzou dat veřejně dostupných API a jejich následným zpracováním za účelem předcházení nadměrného vytížení bodů zájmů. V teoretické části práce jsou popsána veřejná aplikační rozhraní geosociálních sítí včetně jejich finálního srovnání. Práce obsahuje i praktickou část ve formě webové aplikace využívající data z aplikačního rozhraní vybrané geosociální sítě pro doporučování vhodných bodů zájmů ve vyhledávané lokalitě či blízko polohy uživatele. Data bodů zájmu jsou obohacena o počet nakažených virem COVID-19 v dané oblasti.

KLÍČOVÁ SLOVA

API, geosociální síť, overtourismus, využití dat, COVID-19, Symfony, Vue.js

TITLE

Application Use of Geodata From API of Geosocial Networks and API About COVID-19 With Recommendation System in Case of Overtourism

ANNOTATION

Bachelor thesis with name of „Application Use of Geodata From API of Geosocial Networks and API About COVID-19 With Recommendation System in Case of Overtourism“ analyzes data from public API and performs operations of processing for the purpose of point of interest overloading prevention. The public aplication interface of geosocial networks are described and compared in the theoretical section. Thesis also includes a practical part in form of web application which uses the processed data in order to appropriately recommend points of interests to visit in selected area or in the area near the user. The data of each point of interest is enriched by the number of COVID-19 infected people in the given area.

KEYWORDS

API, geosocial network, overtourism, data usage, COVID-19, Symfony, Vue.js

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM ZKRATEK

API	Application Programming Interface
CRUD	Create Read Update Delete
CSS	Cascading Style Sheets
CSV	Comma-Separated Values
DOM	Document Object Model
GPS	Global Positioning System
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure
HTML	HyperText Markup Language
IP	Internet Protocol
JSON	JavaScript Object Notation
MVVM	Model ViewModel View
MZČR	Ministerstvo zdravotnictví České republiky
NAVSTAR	Navigation Signal Timing and Ranging
NPM	Node Package Manager
ORM	Object-Realational Mapping
PHP	Hypertext Preprocessor
PNG	Portable Network Graphics
REST	REpresentational State Transfer
RSS	Rich Site Summary
SLWS	Symfony Local Web Server
SQL	Structured Query Language
SSL	Secure Sockets Layer
TCP	Transmission Control Protocol
TLS	Transport Layer Security
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
Wi-Fi	Wireless Fidelity
ZSJ	Základní Sídelní Jednotka
ZÚJ	Základní Územní Jednotka

ÚVOD

Turismus se v dnešní době vyvinul do takové podoby, kdy velmi často k nadměrnému vytěžování turistických bodů zájmu, aplikace vytvořená v praktické části práce má za cíl tuto skutečnost pomáhat řešit. Bakalářská práce je rozdělena na dvě části, teoretickou a praktickou, kdy praktická část je obsažena v poslední kapitole práce. Cílem této práce je v teoretické části analyzovat využitelnost dostupných API geosociálních sítí, které poskytují data o turistických bodech zájmu, včetně jejich srovnání a API poskytujících data o nakažených virem COVID-19. V praktické části je cílem navrhnout a vyvinout webovou aplikaci založenou na libovolné technologii, která využívá data získaná z vybraných aplikačních rozhraní a inovativním způsobem je prezentuje uživateli s cílem předcházení nadměrného využití turistických bodů zájmu.

Autor teoretickou část doplňuje o kapitoly týkající se komunikace na internetu, vzhledem k tomu, jak je tato problematika úzce spjatá s využitím aplikačních rozhraní vybraných geosociálních sítí. V neposlední řadě je doplněna kapitolu zabývající se overtourismem a návrhem jeho přecházení. Samostatná kapitola je věnována i aplikačnímu využití otevřených dat.

Praktická část popisuje návrh vývoje aplikace, je znázorněna architektura aplikace, popsáný technologie použité při vývoji, její běhové prostředí a entity poskytující vrstvu abstrakce v aplikaci. Použité technologie byly zvoleny vzhledem k charakteru aplikace, tzn. umožnění uživatelsky příjemného ovládání, responzivity a přirozené multiplatformnosti webových aplikací. Dalším důležitým faktorem byla autorova obliba zvolených technologií a jeho zkušenost vývoje s jejich využitím.

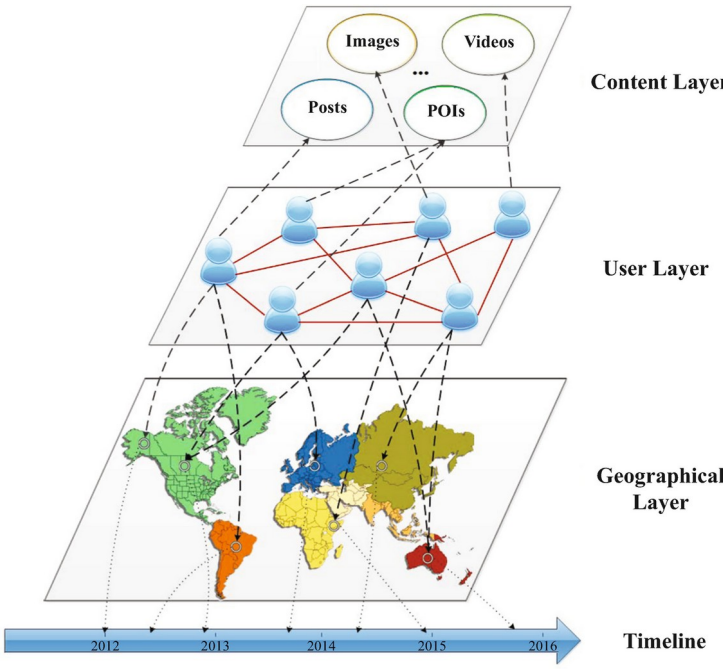
1 GEOSOCIÁLNÍ SÍŤ [1]

Geosociální síť je sociální síť obohacená o geografické informace. S pomocí aplikace založené na takové síti, zpravidla na mobilním zařízení, může uživatel najít dostupné formy zábavy, restaurace nebo ostatní uživatele. Geosociální síť může být sociální síť (Facebook, Twitter apod.), která byla rozšířena o nějakou funkcionalitu v podobě oznámení polohy na nějakém bodě zájmu nebo čistě geosociální síť (Foursquare, Yelp) zaměřená vyloženě na „location-based“ služby (podle polohy).

Geosociální síť může být reprezentována grafem, kde graf je množina vrcholů a hran. V tomto grafu každý uzel představuje uživatele, hrany představují vztahy mezi uživateli. Ke každému uzlu pak existuje seznam souřadnic či lokací navštívených uživatelem. Hrany mohou být orientované i ohodnocené – takové hodnocení poté znamená sílu vtahu mezi uživateli.

Vztah mezi uživateli a jejich geografická poloha má velký vliv i na vztahy mezi uživateli. Podle [2] závisí až 66% sociálních vztahů mezi jedinci na jejich lokaci.

Uživatelé mohou kromě oznamování své polohy přidávat geografická metadat i k jejich obsahu. To mohou být příspěvky, obrázky, videa apod.



Obrázek 1: Nákres architektury geosociální sítě

Zdroj: [3]

2 ANALÝZA GEOLOKAČNÍCH SLUŽEB [4]

2.1 Úvod do geolokačních služeb

Geolokace je proces určování geografické polohy objektu. Existuje několik způsobů či metod určování této polohy. Metody mají různé úrovně přesnosti a požadavky na zařízení určující polohu. Mezi zařízení nejčastěji využívaná ke geolokace patří mobilní telefony, tablety, notebooky, PC či chytré hodinky. Pokud zařízení disponuje například čipem GPS je možné určit jeho polohu s poměrně vysokou přesností v řádu jednotek metrů. [5] Ostatní způsoby bývají méně přesné, avšak některé lze použít i bez vědomí uživatele. Způsoby budou blíže popsány a analyzovány v další kapitole.

2.2 Způsoby geolokace

2.2.1 GPS

Systém GPS byl pod původním názvem NAVSTAR GPS (Navigation Signal Timing and Ranging Global Positioning System) vyvíjen od roku 1973 s cílem návaznosti a nahrazení předchozího systému Transit. Po pěti letech od počátku vývoje byl na oběžnou dráhu vypuštěn první satelit GPS. Nicméně trvalo dalších 16 let než systém dosáhl plné operační přesnosti díky více než 20 satelitům na šesti různých oběžných drahách Země. [6]

Obrázek 2: Ilustrace trajektorií satelitů GPS na oběžné dráze Země

Zdroj: [11]

Jak již bylo zmíněno v úvodu, GPS je velmi přesný způsob geolokace. Podmínkou je, že zařízení musí mít GPS lokátor. GPS poskytuje mimo získávání polohy i služby jako navigace a časování. [6]

2.2.2 Celulární síť

Dalším možným způsobem určování polohy je použití celulární mobilní sítě. S využitím této sítě pracují, mimo jiné, i tyto metody geolokace [7]:

- Cell of Origin – Využívá buňkové architektury sítě, polohu určuje na základě toho, na které vysílače signálu je zařízení připojeno. Přesnost se pohybuje od 150 metrů v hustě obydlených oblastech do 35 kilometrů (nejvyšší možný poloměr GSM buňky).
- Timing Advance – K fungování potřebuje mít určenou vzdálenost minimálně tří vysílačů. Samotné zařízení pak určí sílu rádiového signálu, resp. vzdálenosti vysílačů. Přesnost závisí na hustotě GSM sítě a dosahuje v nejlepším scénáři možnost určení polohy s přesností 125 metrů.
- Time of Arrival – Podobně jako metoda Timing Advance, vyžaduje tato metoda znalost vzdálenosti ze tří vysílačů (základnových stanic). Oproti předchozí metodě jsou ale určováním vzdálenosti pověřeny základnové stanice.

2.2.3 Wi-Fi

Určování polohy s pomocí Wi-Fi je velmi účinná metoda, zejména pokud se v okolí zařízení vyskytuje větší počet přístupových bodů. Čím více, tím lépe. V principu jde o zjištění unikátních ID přístupových bodů Wi-Fi v okolí a síly jejich signálu. Tyto hodnoty se použijí k dotazování na databázi známých poloh přístupových bodů. Důležité je zmínit, že není nutné, aby se zařízení na Wi-Fi připojilo. [8]

2.2.4 IP adresa

Technika geolokace podmíněná připojením zařízení do sítě internet. Každé zařízení v síti internetu má přiřazenou vlastní unikátní IP adresu, kterou komunikuje se zbytkem sítě. Existuje řada databází IP adres a jejich využití je zpravidla placené.

IP adresa	77.236.200.110
Země	Czechia
Oblast	Pardubicky kraj
Město	Pardubice
PSČ / poštovní směrovací číslo	530 09
Zeměpisná šířka	50.04075
Zeměpisná délka	15.77659
ISP	Edera Group a.s.

Obrázek 3: Geolokace pomocí IP adresy

Zdroj: vlastní

2.3 Možnosti využití geolokace

2.3.1 Odhalování podvodů

Finanční instituce, jako například banky, mohou použít geolokační služby k odhalování podvodů s platebními kartami. Na základě znalosti polohy uživatelova zařízení ve kterém je přihlášen k bankovnictví je možné tuto polohu porovnat například s polohou použití platební karty a pokud se zásadně liší, tak zamítnout platbu.

2.3.2 Geomarketing

Jak je možné odhadnout z názvu, geomarketing se zabývá reklamou na základě polohy uživatele. Data získána od uživatelů totiž velmi rostou na hodnotě, jsou-li opatřená i polohou a mohou sloužit jako stěžejní bod plánování a provedení marketingového postupu. Geomarketing má přímý vliv na vývoj moderního obchodu.

2.3.3 Ostatní

Geolokace může být využita i k následujícím účelům:

- Geotagging – Proces přidávání metadat určujících geografické umístění k médiím jako například fotografie, videa, webové stránky nebo RSS. Tato metadata zpravidla představují zeměpisné souřadnice, ale mohou obsahovat mnohem více: nadmořskou výšku, data o přesnosti, název místa apod. [9]

- Geographic Information System – Je nástroj pro získávání, spravování a analýzu geografických dat. Mimo jiné analyzuje geografická data v prostoru a nabízí možnost její vizualizace v porovnání s entitami nacházejícím se v blízkosti. Tímto dává možnost lepšího pochopení spojitostí a náhledu nad vztahy objektů v oblasti. [10]

2.4 Ochrana soukromí

Užívání geolokace v geosociálních sítích by se dalo označit za dvousečný meč. Při pohybu na internetu je třeba se mít na pozoru a nesdělovat o sobě zbytečně moc informací. Nevíme totiž, kdo může naše příspěvky sledovat a jaké má úmysly. Umožnění interakce s blízko se nacházejícími uživateli může být zajímavé a člověk je ze své podstaty tvor společenský, ale neměl by se nechat unést a vždy používat zdravý rozum.

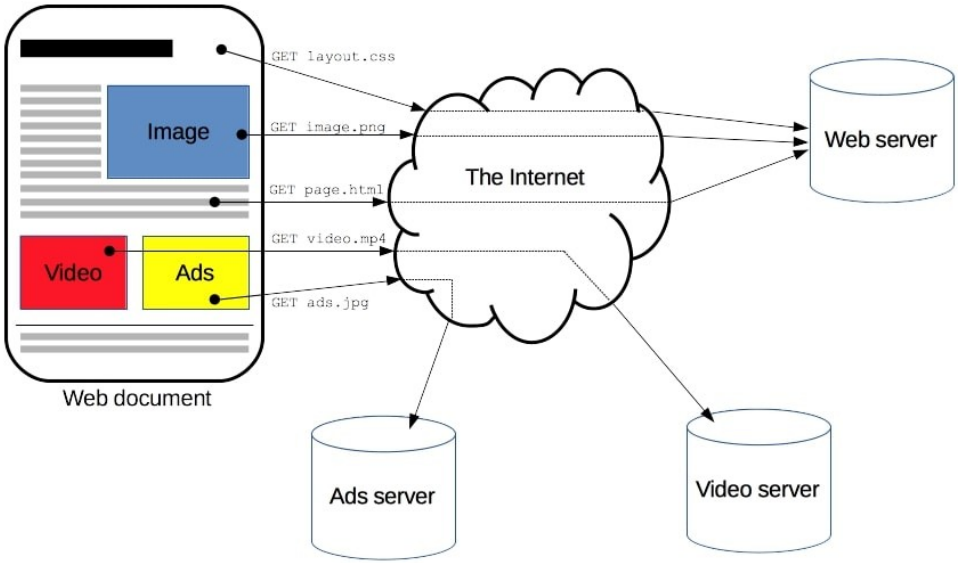
2.5 Vliv Big Data na geolokaci

V poslední době došlo díky Big Data k velkému rozvoji v oblasti geolokace. Umožňují totiž získat z dat nasbíraných od uživatelů výstup, který má smysl, je strukturovaný a ucelený. Spojení Big Data a geolokace, také velmi ovlivňuje společnost a jedním z nástrojů je Geofencing. Na základě geolokačních dat mohou být například podniky v oblasti upozorněny na vstup uživatele do ní a v reakci na tuto skutečnost nabízet zákazníkovi produkty se slevou. V návaznosti na předešlé akce pak lze sestavovat dokonce vzorce chování zákazníků a poskytnout tak ucelenější pohled na trh a jejich průměrného zákazníka.

3 KOMUNIKACE NA WEBU

3.1 Prokolol HTTP

HTTP je nestavový protokol aplikační vrstvy naprosto fundamentální pro komunikaci a výměnu dat v prostředí webu. Umožňuje získávání zdrojů, mezi které můžou patřit HTML dokumenty, CSS stylopisy, soubory JavaScriptu, obrázky, videa apod. Využívá architektury klient-server a požadavky na server jsou zpravidla iniciovány webový prohlížečem. Kompletní podoba stránky je pak složena ze získaných sub-dokumentů. [12]



Obrázek 4: Získávání zdrojů pro zobrazení webové stránky

Zdroj: [12]

3.1.1 Požadavky a odpovědi

Klient se serverem komunikuje za pomocí výměny zpráv protokolu HTTP. Klient zasílá na server *požadavek (response)* a server mu zasílá *odpověď (response).* HTTP požadavek obsahuje následující:

- HTTP metodu – indikuje jaká akce se má provést na základě požadavku [13],
- URL cestu – cesta požadovaného zdroje,
- verzi protokolu,

- hlavičky – umožňují předávání dalších informací.

```
GET / HTTP/1.1
Host: upce.cz
Accept-Language: cs
Cache-Control: max-age=0
Cookie: has_js=1
```

Obrázek 5: Příklad HTTP požadavku

Zdroj: vlastní

Na požadavek server zasílá odpověď. Odpověď obsahuje:

- verzi protokolu,
- stavový kód – určuje zda byl požadavek úspěšně proveden,
- stavovou zprávu,
- hlavičky,
- případně může také obsahovat tělo s žádaným zdrojem, například HTML dokument, XML apod.

```
HTTP/1.1 200 OK
Date: Sat, 09 Apr 2021 14:20:08
Server: Apache/2.4.38 (Debian)
Content-Language: cs
Content-Type: text/html; charset=utf-8
Cache-Control: no-cache, must-revalidate
```

Obrázek 6: Příklad HTTP odpovědi

Zdroj: vlastní

Metody [13]

Metody požadavků indikují akce, které se mají provést pro zdroj definovaný pomocí URI požadavku. Každá z metod má vlastní sémantiku, nicméně mají nějaké společné vlastnosti. Jejich chování lze rozdělit do tří skupin, popis metod a skupiny, do kterých nich patří jsou k nalezení v následující tabulce. Jednotlivé skupiny obsahují metody které jsou:

- Bezpečné (Safe) – Bezpečné metody žádným způsobem neovlivňují stav serveru. [14]

- Idempotentní (Idempotent) – Zařazení do této skupiny naznačuje to, že opakované provedení požadavku nebude mít žádný vliv. Stav tedy bude stejný po prvním i po pátém stejném požadavku. Všechny bezpečné metody jsou idempotentní. [15]
- Cachable – Takové metody mohou být cachovány – jejich obsah je dočasně uložen do mezipaměti a použit bez nutnosti opětovného požadavku na server. [16]

Tabulka 1: HTTP metody [13]

Metoda	Popis	Skupiny		
		Safe	Idempotent	Cachable
GET	Žádá reprezentaci zdroje. Metoda by mělo pouze získávat data.	X	X	X
HEAD	Žádá o stejnou odpověď jako GET, ale bez těla.	X	X	X
POST	Indikuje zaslání nějakých dat. Může způsobit změnu stavu nebo mít vliv na server.			X*
PUT	Nahradí momentální stav zdroje zasílaným.		X	
DELETE	Vymaže zdroj.		X	
CONNECT	Vytvoří tunel k server na kterém se zdroj nachází.			
OPTIONS	Žádá o popis možných metod komunikace.	X	X	
TRACE	Provede test zdroje na uvedené cestě. Vhodné pro ladění.	X	X	
PATCH	Aktualizuje stávající zdroj.			X*
* Pouze pokud je použita hlavička *Content-Location* a je definována *„freshness“*.				

Stavové kódy

Slouží k upřesnění, jakým způsobem byl požadavek na serveru zpracován. Stavové kódy mají číselnou hodnotu a podle jejich charakteristiky jsou zařazeny do několika tříd:

- informační (100-199),
- oznamující úspěch (200-299),
- směrovací (300-399),
- oznamující chybu na straně klienta (400-499),
- oznamující chybu na straně serveru (500-599).

Při pokročilejší práci s protokolem HTTP je důležité kódy znát a správně je používat. To platí hlavně pro vývojáře API, kde může správný stavový kód pomoci jeho uživatelům při hledání chyby v požadavku. Stavových kódů jsou desítky, avšak běžně se setkáváme s několika základními:

- 200 OK – Požadavek byl úspěšný.

- 301 Moved Permanently – Zdroj byl permanentně přesunut na jinou lokaci. Odpověď obsahuje i novou lokaci.
- 403 Forbidden – Selhání autorizace.
- 404 Not Found – Požadovaný zdroj nebyl nalezen.
- 500 Internal Server Error – Požadavek způsobil chybu na straně serveru a nemohl být zpracován.

3.1.2 Spojení

Jak již bylo zmíněno, jedná se o protokol aplikační vrstvy a pro svůj přenos na transportní vrstvě vyžaduje spolehlivý protokol. Nejčastěji je tedy použit TCP. Transmission Control Protocol je spojovaný a spolehlivý, což znamená, že se před navázáním spojení ověří jeho dostupnost a teprve poté se naváže. Navázání spojení se realizuje pomocí třícestného potřesení rukou. Aby byl spolehlivý (mj. znamená spolehlivost: doručení informace o selhání doručení a správné pořadí segmentů) používá sekvenční číslování a příznaky (ACK, SYN, FIN a další). [18]

I přes navázání a udržování spojení pomocí TCP je protokol bezstavový. To znamená, že mezi dvěma požadavky zaslanými za použití stejného spojení není žádné spojení. Když si představíme, jak fungují webové stránky a k čemu se používají, tak je jasné, že nějaký způsob pro udržení stavu musí existovat. Stav je možné ukládat do tzv. Cookies – to je nějaká struktura dat uložená u klienta.

3.1.3 Zabezpečení

Komunikace samotného protokolu HTTP zabezpečená není, lze však využít šifrovaných protokolů relační vrstvy: SSL (Secure Sockets Layer) nebo TLS (Transport Layer Security). Pokud je přenos takto zabezpečen, jedná se o protokol HTTPS (Hypertext Transfer Protocol Secure).

3.1.4 Proxy

Mezi prohlížečem a serverem se většinou nachází více zařízení – spojení není přímé. Většina z nich ale operuje na nižších vrstvách TCP/IP modelu. Zařízení pracující na aplikační vrstvě se nazývají proxy servery. Mohou být transparentní (nemění obsah požadavku) nebo netransparentní (mění nějakým způsobem obsah požadavku), jejich typ tedy určuje způsob interakce s požadavkem než ho předají webovému serveru. Mohou provádět operace jako například: vyvažování zátěže, autentizaci, logování, ukládání do mezipaměti apod.

3.2 REST [20]

3.2.1 Základní informace

RESTful API je aplikační programové rozhraní, které splňuje omezení a konvence REST architektury. Web je místo, kde je každý zdroj identifikován pomocí URI. Každý takový zdroje je komunikován pomocí jeho reprezentace v obecně známých formátech (XML, HTML, CSS, PNG) za použití protokolů. Jedna z více omezujících a specifických architektur je právě REST, navržený Royem Fieldingem v jeho disertační práci v roce 2000. REST je takovou podmnožinou webu založenou na HTTP, s jejíž pomocí je umožněno využitím definované jednotné sémantiky provádět CRUD operace nad zdroji místo implementace aplikačních rozhraní pro každou aplikaci zvlášť. Manipulace se zdroji tak probíhá výměnou jejich reprezentací. [19]

3.2.2 Architektura

Jak již bylo zmíněno, REST je vlastně derivátem nebo je založena na webu. K definování architektury REST její autor Roy Fielding používá, mimo jiné, i několik omezení, které musí systém splňovat. Tato omezení implikují existence řady vlastností REST, kterými se odlišuje do jiných architektur. Mezi tyto vlastnosti patří: výkonnost, škálovatelnost, jednoduchost rozhraní a další.

Klient-server

Architektura klient-server je architektura počítačové sítě v níž více klientů zasílá požadavky a používá služby jednoho centralizovaného serveru. Oddělením problematiky uživatelského rozhraní od požadavků na logickou část systému dosáhneme lepší škálovatelnosti systému a jeho přenositelnosti.

Bezstavovost

Dalším omezením architektury je bezstavovost. Takové omezení udává, že každý požadavek od klienta na server musí obsahovat všechny potřebné informace k porozumění a zpracování požadavku a na serveru nesmí být uložen žádný kontext komunikace.

Toto omezení bylo zavedeno z důvodu lepší orientace a rozšiřitelnosti. Při zpracování požadavků se tak nemusí hledat, jestli daný požadavek souvisí s předchozím a nemusí žádným způsobem ukládat kontext těchto požadavků. Nevýhodou tohoto omezení je zvýšení síťového provozu, protože je nutné poskytnout všechny informace v každém požadavku.

Využití mezipaměti

Omezení vyžadující využití mezipaměti je definováno za účelem zlepšení výkonnosti sítě, ukládá totiž, že data komunikovaná jako odpověď na požadavek mohou být označena jako určená pro ukládání do mezipaměti („*cachable*“) nebo naopak („*non-cacheable*“). Pokud je odpověď označena jako „*cacheable*“, dává to klientovi právo jejího znovupoužití v případě provedení stejného požadavku.

I přes výrazné zlepšení výkonu a odezvy aplikace může dojít k situaci, kdy je zdroj uložený v mezipaměti klienta výrazně odlišný oproti tomu, který by získal opětovným požadavkem.

Vrstvený systém

Za účelem zlepšení využití architektury v prostředí internetu byly zavedeny omezení vrstev. Takový systém je pak složen z vrstev, kdy funkcionalita jedné vrstvy je ukryta těm ostatním. Tímto jsou určeny nějaké meze zodpovědnosti součástí systému a jejich nezávislost. Mohou být využity k oddělení služeb mezi systémy zabývající se různou problematikou nebo k vkládání prostředníku mezi klienta a server, takový prostředník pak může být zodpovědný například za vyvažování zátěže, kompresi dat či šifrování.

Jednotné rozhraní

Hlavní vlastností rozlišující REST od ostatních architektur síťové komunikace je důraz na jednotné rozhraní. Toho dociluje použitím obecnosti v definici rozhraní a zjednodušuje tím komunikaci a pomáhá čitelnosti interakcí.

Pro dosažení tohoto omezení je nutné definovat několik dalších omezení. Rozhraní REST je proto definováno čtyřmi vlastnostmi:

- Identifikace zdrojů – Zdroj je klíčová forma abstrakce pro REST. Mezi zdroje patří dokument, soubor, obrázek apod.
- Manipulace se zdroji pomocí jejich reprezentace – Komponenty REST provádějí operace nad zdroji za použití jejich aktuální či vyžadované reprezentace.
- Sebepopisující zprávy – Vzhledem k bezstavovosti je nutné, aby zprávy byly sebepopisující, tudíž si k nim komponenta REST nemusí dohledávat kontext.
- Hypermédia jako řídící mechanismus stavu aplikace – Umožňuje komunikaci klienta se serverem skrze médium, které je poskytováno dynamicky a díky tomu není nutná žádná předchozí znalost možných způsobů komunikace se serverem a postačí porozumění danému médiu. [21]

Kód na vyžádání

Poslední z omezení umožňuje rozšířit funkcionalitu na klientovi pomocí stahování a vykonávání kódu za pomoci tzv. appletů nebo skriptů. To má za následek snížení komplexity na straně klienta, zároveň také odpadá nutnost implementace předem. Možnost stažení funkcionality později tak zlepšuje škálovatelnost systému. Nicméně, vzhledem ke skutečnosti, že tato vlastnost kromě svých výhod snižuje viditelnost, je nepovinná.

3.2.3 Příklad

Mějme REST rozhraní implementované na straně systému pro správu událostí a definujme si operace, které nad nimi chceme pomocí něj provádět. Budeme chtít získávat množiny událostí, provádět operace nad konkrétní událostí a získat vystupující na dané události. URI pro získání zdrojů by pak mohly vypadat takto:

Tabulka 2: Příklad REST API

URI	Použitá HTTP metoda	Popis
/event	GET	Vrátí kolekci událostí. Mohlo by podporovat i filtraci podle data, místa konání apod. za použití GET parametrů.
	POST	Vytvoří novou událost na základě zaslané reprezentace.
/event/EVENT_ID	GET	Vrátí reprezentaci konkrétní události.
	PUT	Nahradí aktuální reprezentaci události na serveru tou zaslanou v požadavku.
	DELETE	Vymaže událost.
/event/EVENT_ID/performer	GET	Vrátí kolekci vystupujících na události.
	POST	Přidá vystupujícího k události.

Jak můžeme vidět, tak struktura rozhraní je velmi dobře čitelná a ve spojení se znalostí metod HTTP by se dala označit i jako sebepopisující. Přidání další typy zdrojů (uvedeny jsou *event* a *performer*) lze pak API horizontálně škálovat bez ovlivnění stávajících přístupových bodů rozhraní, ale při zachování stejné sémantiky.

4 ANALÝZA VEŘEJNÝCH API GEOSOCIÁLNÍCH SÍTÍ

V následujících kapitolách bude popsáno možné využití geosociálních sítí z pohledu vývojáře a to s pomocí jejich veřejných API, která data v různé míře skrze tuto konstrukci zpřístupňují. Ve většině případů tyto API využívají konstrukci zvanou REST. Budou popsány způsoby získávání dat z rozhraní a jejich následné zpracování zahrnující například mapování zdrojových dat na entity používané v aplikaci.

Každá z probíraných API implementuje nějaké omezení z pohledu množství získávaných dat. Povětšinou se tato omezení kladou na maximální počet HTTP požadavků na rozhraní za nějaký daný časový úsek. Míra omezení se odvíjí od zvoleného tarifu, přičemž jsou nabízeny i tarify zpřístupňující určité množství dat zdarma.

4.1 Foursquare [22]

Foursquare je společnost poskytující mj. služby týkající se zefektivnění reklamy na základě umístění uživatele. To umožňuje zobrazovat velmi přesnou reklamu, díky analýze chování uživatelů. Tohle by nebylo možné bez velkého množství dat, která společnost vlastní. Tato data jsou velmi vhodná pro geosociální aplikaci, kterou v praktické části práce budujeme. Foursquare je totiž zpřístupňuje skrze řadu služeb, jakou jsou: *Places API*, *Places Database* nebo *Pilgrim SDK*. Zejména službu Places API budeme v aplikaci využívat.

Data společnost získává přímo od uživatelů. V databázi má přes 105 miliónů bodů zájmu a sbírá data z více než 500 miliónů zařízení. Jejich proprietární nástroj *Pilgrim* využívá informace o bodech zájmu a data ze zařízení k přesnému a spolehlivému určení navštívení místa ve světě, zároveň se z těchto dat učí o lidském pohybu a analyzuje ho. Jako jedna z mála metod je metoda, kterou používá *Pilgrim*, akreditovaná autoritou *Media Ratings Council*. Úkolem této autority je mj. správa akreditací pro výzkum a jejich hodnocení. [23]

4.1.1 Places API [24]

Poskytuje přístup k datům bodů zájmu a s nimi spojeným obsahem z více než 100 000 ověřených zdrojů. Data jsou aktualizována v reálném čase na základě akcí miliónů uživatelů. Primárně s využitím této služby můžou aplikace získávat data pro zobrazování a nabízení bodů zájmu.

4.1.2 Získávání dat [24]

Pro získání dat existuje řada REST API koncových bodů, s nimi lze komunikovat pomocí HTTP protokolu. Pro začínající uživatele je nejsnazší způsob vyzkoušení *Places API* pomocí kolekce do programu *Postman*, ve kterém ji lze pohodlně vyzkoušet.

Nutná příprava

První podmínkou je mít založený účet u Foursquare. Dalším nutným krokem je vytvoření aplikace na straně tohoto účtu – nastavení názvu aplikace a URL, kde je tato aplikace provozována. Po vytvoření Foursquare aplikace je poskytnut pár klíčů (Client ID, Client Secret) sloužících k autentizaci požadavků prováděných na API. Tento pár klíčů bude nutné zasílat s každým požadavkem jako parametr *GET* v URL adrese. Zároveň se musí v každém požadavku uvést parametr verze API, díky kterému je umožněno aplikaci využívající API verzovat nezávisle. Verze je reprezentována datem ve formátu *RRRRMMDD*, například *20200606*. Základní URL je *https://api.foursquare.com/v2*, všechny následující požadavky budou vznikat připojením jejich parametrů a URI k ní.

Vyhledávání bodů zájmu

Vyhledávání je základní operací, kterou lze provádět. Pro provedení této operace je nutné použít koncový bod */venues/search*, který vrací seznam bodů zájmu v blízkosti polohy, případně vezme při vyhledávání v potaz i klíčové slovo. V následující tabulce jsou uvedeny parametry, které je možné zasílat pro vyhledávání. Jedná se o požadavek s cílem najít restaurace v Pardubicích. Povinné parametry mají jméno parametru napsané tučně.

Tabulka 3: Fousquare: Parametry požadavku vyhledávání bodů zájmu

Jméno parametru	Příklad hodnoty	Popis
ll	50.03401129693814, 15.766541206973837	Zeměpisná šířka a délka polohy uživatele.
near	Pardubice	Řetězec definující místo.
radius	2000	Maximální vzdálenost místa. Bere se v potaz pouze v případě uvedení parametru categoryId.
query	pizza	Klíčové slovo, které by měl obsahovat název místa.
limit	10	Maximální počet vrácených výsledků.
caterogoryId	4d4b7105d754a06374d81259	Seznam identifikátorů kategorií oddělených čárkou. Pokud se jedná o rodičovskou kategorii, jsou do výsledku zahrnuty i její subkategorie.

Po zaslání požadavku dle specifikace v tabulce dostaneme odpověď obsahující seznam míst odpovídajících dotazu, jedno z nich je znázorněné v následující tabulce, nicméně strukturou jsou všechna stejná.

Tabulka 4: Fousquare: Pole odpovědi vyhledávání bodů zájmu

Jméno parametru	Příklad hodnoty	Popis
id	5752f4f0cd1087abada63fe0	Identifikátor bodu zájmu.
name	Galerie Cafe Park Restaurant	Řetězec definující místo.
location	{address, crossStreet, city, state, postalCode, country, lat, lng, distance}	Objekt obsahující žádná, některá nebo všechna pole ze sloupce příkladu hodnoty.
categories	[{categoryId, name, pluralName, shortName, icon, primary}]	Pole objektů kategorií s atributy kategorie.

Získání detailu bodu zájmu

Ačkoliv zmíněné vyhledávání vrací výsledky velice dobře vztažené k dotazu, tak aby byla API využitelná v aplikaci, je nutné o bodech zájmu získat více informací. K tomu nejlépe slouží koncový bod */venues/VENUE_ID*. Na tomto koncovém bodě lze získat další informace o místu, v URI požadavku se musí uvést povinný parametr *VENUE_ID,* který jsme mohli získat ve vyhledávání. Bohužel je tento přístupový bod rozhraní označen jako prémiový, což má za následek daleko výraznější omezení jeho využití. Konkrétně se jedná o omezení na pouze 500 požadavků denně oproti 99 500 pro takto neoznačené přístupové body. Vybraná pole zasílaná jako odpověď na tento požadavek byla zpracována do následující tabulky. Požadavek ob-

sahuje i pole zmíněná ve vyhledávání, která v této tabulce již z důvodu duplicity nebudou uvedena.

Vzhledem k daleko větší obsáhlosti dat z tohoto bodu je z nich daleko snazší sestavit obsahově zajímavější a pestrý výstup uživateli. Zobrazení dalších metadat Nemusíme se ani omezovat na zobrazení dat pouze v textové podobě, ale můžeme k místu ukázat například i jeho fotografii.

Tabulka 5: Fousquare: Pole odpovědi detailu bodu zájmu

Jméno parametru	Příklad hodnoty	Popis
contact	{twitter, facebook, instagram, phone, formattedPhone}	Objekt obsahující žádná, některá nebo všechna pole ze sloupce příkladu hodnoty. Některé z nich mohou být i identifikátory tohoto místa v jiných sociálních sítích.
stats	{checkinsCount, usersCount, tipCount}	Objekt obsahující počet odbavení, počet uživatelů zde odbavených a počet doporučení.
hours	{hours}	Objekt s informacemi ohledně otevírací doby ve formátu čitelném pro člověka.
rating	10	Hodnocení místa.
hereNow	6	Počet uživatelů na místě.
description	„Popis“	Popis místa.
url	„https://www.galeriecafepardubice.cz/“	URL webové stránky místa.
photos		Objekt popisující skupinu fotografií tohoto místa spolu s URL fotografií.

Další užitečné přístupové body

Zmíněné body pro hledání míst a získání jejich detailů jsou dle autora ty nejpodstatnější pro aplikační využití. Nicméně *Places API* nabízí řadu dalších přístupových bodů, jejichž výsledky by bylo možné aplikaci ještě dále obohatit.

Mezi takové patří například tyto:

- */venues/trending* – vrátí seznam bodů zájmu v okolí, kde se momentálně nachází největší počet uživatelů,
- */venues/VENUE_ID/similar* – vrátí seznam podobných bodů zájmu,
- */venues/VENUE_ID/events* – vrátí informace o probíhajících událostech na místě.

4.2 Yelp

Yelp patří mezi další známé geosociální sítě. Byl založen v roce 2004 v San Franciscu (USA). Oproti Foursquare není Yelp tolik zaměřený na zobrazování míst pro volno-časové aktivity a stravovací zařízení, ale nabízí širší škálu bodů zájmu a dokonce i služeb. Yelp propojuje uživatele a místní podniky s pomocí velkého množství informací, mezi něž se řadí například fotky a recenze. Poskytuje platformu pro spotřebitele k objevování, propojování a interakci s místními podniky všech velikostí. To se mu daří i díky zjednodušení interakce mezi zúčastněnými subjekty, tedy zákazníkem a podnikem. Zákazník se může s použitím Yelpu dotázat na cenovou nabídku, zařadit se na čekací listinu, vytvořit rezervaci apod. [40]

Kromě služeb pro typického uživatele, nabízí Yelp řadu služeb pro vývojáře či provozovatele podniků. Vývojáři ocení zejména Yelp Fusion (zpřístupňuje obsah skrze REST API), Yelp GraphQL (zpřístupňuje obsah pomocí GraphQL) a pro nekomerční použití i Yelp Open Dataset (umožňuje stažení dat o podnicích, hodnoceních a uživatelích). Provozovatelé pak ocení například Yelp Knowledge (mimo hodnocení a recenzí uživatelů poskytuje metriky zájmu uživatelů a další analytická data pro komerci) nebo Yelp Advertising (služba zaměřená na reklamu). [41]

4.2.1 Yelp Fusion

Yelp Fusion API umožňuje získávat obsah vztažený k poloze i uživatelská hodnocení pro milióny podniků z 32 zemí. [42] Ke zpřístupnění obsahu používá architekturu REST API. Ke komunikaci se používá, jak je pro tuto architekturu typické, protokol HTTP. Kmenová URL, pod níž se všechny přístupové body rozhraní nacházejí, je *https://api.yelp.com/v3*.

Komunikační prerekvizity

Pro umožnění komunikace je nutné si založit účet u Yelpu a v tomto účtu si vytvořit aplikaci. Po vyplnění jména aplikace, sekce průmyslu, do které aplikace patří a kontaktního emailu se v nastavení aplikace objeví vygenerovaný pár přístupových údajů (Client ID a API Key). Při komunikaci je nutné posílat API Key v hlavičce *Authorization*, aby byl každý přístup na autorizován.

Vyhledávání míst

Jedním z přístupových bodů nabízených *Fusion API* je *Business Search*. Je tím základním bodem, na jehož datech můžeme stavět další komunikaci a nachází se na adrese */businesses/search*. Umožňuje vyhledávání míst dle klíčových slov, kategorií, polohy atd. V následující tabulce jsou uvedeny vybrané parametry požadavku na tento přístupový bod.

Tabulka 6: Yelp: Parametry požadavku vyhledávání bodů zájmu

Jméno parametru	Příklad hodnoty	Popis
term	„food“, „restaurant“, „Starbucks“	Vyhledávané klíčové slovo, může obsahovat i názvy podniků.
location	„Pardubice“	Řetězec definující místo. Povinný parametr pokud není uveden parametr coordinates.
coordinates	50.03357024001322, 15.767142021737632	Souřadnice polohy uživatele. Povinný parametr pokud není uveden parametr location.
radius	20000	Maximální vzdálenost od polohy.
caterogories	„bar“	Jeden či více identifikátorů kategorie oddělených čárkou.
limit	20	Maximální počet výsledků.
price	„1“, „2“, „1,2,3“	Cenové hladiny do kterých může místo spadat. 1 – nejlevnější, 4 – nejdražší.
sort_by	„distance“, „best_match“, „rating“, „review_count“	Požadovaný atribut řazení. Uvedené řazení nemusí být ve výsledku použito.

Kromě parametrů uvedených v tabulce je možné zasílat i parametry další. Mezi ně se řadí například *locale*, který umožňuje lokalizaci výsledku nebo *attributes*, který přijímá hodnoty příznaků (*hot_and_new* pro nové podniky, *reservation* pro podniky umožňující rezervaci pomocí Yelpu apod.).

API Yelpu vrací na požadavek vyhledání míst nejen samotná místa, ale i další užitečná data využitelná v aplikaci. Kromě polí uvedených v tabulce níže, což jsou pole každého místa, které je vráceno v seznamu míst, se v odpovědi nachází pole s celkovým počtem vyhledaných míst a souřadnice geografického středu těchto míst, které se v klientské aplikaci dají využívat k vycentrování bez nutnosti jejich výpočtu. Kořenový prvek odpovědi tedy obsahuje tři pole – seznam míst, geografický střed a počet míst.

Tabulka 7: Yelp: Vybraná pole míst vrácených vyhledáváním

Jméno pole	Příklad hodnoty	Popis
name	„Potrefená Husa Pardubice“	Textový řetězec obsahující jméno místa.
coordinates	{longitude, latitude}	Objekt se souřadnicemi místa.
distance	1550.5	Vzdálenost od polohy uživatele k danému místu.
id	„4kMBvIEWPxWkWKFN__8SxQ“	Jednoznačný identifikátor místa.
image_url	https://s3-media2.fl.yelpcdn.com/bphoto/UUMdjzkbbB-whooAfTQJ-Fg/o.jpg	URL hlavního obrázku místa.
location	{address1, address2, address3, city, country, ...}	Údaje o lokaci obsahující adresu místa.
rating	2.5	Hodnocení v interval od 1 do 5, kde 5 znamená hodnocení nejvyšší.
review_count	10	Počet hodnocení.
url	https://www.yelp.com/biz/potrefen%C3%A1-husa-pardubice	URL místa na straně Yelpu.
price	Jedno z „$“, „$$“, „$$$“, „$$$$“	Cenová hladina místa.

Získávání detailu místa

Stejně jako v případě ostatních aplikačních rozhraní geosociálních sítí existuje více úrovní rozlišení či detailu, ve kterých lze místo získat. Ačkoli již samotné vyhledávání nabízí poměrně široký záběr informací o místě včetně jedné jeho fotografie, hodnocení a dokonce i telefonního čísla, tak pro získání dalších informací je možné využít právě rozhraní pro detail místa. V detailu jsou navrácena všechna pole obsažena ve vyhledávání vyjma toho se vzdáleností od aktuální polohy, jelikož se v požadavku na tento koncový aktuální poloha neuvádí.

Tabulka 8: Yelp: Vybraná pole odpovědi detailu místa

Jméno pole	Popis
photos	Pole obsahující až tři URL fotografií místa.
hours	Objekt s informacemi ohledně otevírací doby.
categories	Pole s objekty kategorií míst.

4.3 Srovnání Foursquare a Yelp API

Přístup na obě API lze zřídit během pár minut a u obou je nabízen bezplatný tarif s patřičnými omezením denní kvóty požadavků. Dokumentační úroveň je pro obě API na velmi dobré úrovni, ale v této disciplíně pro autora vítězí spíše Foursquare z důvodu o trochu lepší přehlednost.

Obě rozhraní také nabízejí kolekci vzorových požadavků do programu Postman, ve kterém si lze požadavky vyzkoušet. Když přijde na řadu implementace komunikace na jednu či druhou API, tak nás jako vývojáře kryjí knihovny řešící právě tuto problematiku – v obou případech přímo existuje seznam doporučených knihoven přímo od poskytovatele API.

V disciplíně počtu přístupových bodů jasně vítězí Foursquare, které nabízí body rozhraní nejen pro hledání a detail, ale i třeba podobná místa k dotazovanému místu apod. Je tedy možnost místa nejen prohlížet, ale je i hodnotit, přidat k nim fotku, získat k nim statistiku vztaženou k přihlášenému uživateli atd.

Z pohledu aplikačního využití je zásadní několik vlastností poskytovaných dat: kvalita, dostupnost a cena. Obecně se data z Foursquare dají považovat za kvalitnější, ale kvalita něco stojí, hlavní vliv na to má nutnost více požadavků na rozhraní pro získání obsahově bohatých dat. V následující tabulce jsou shrnuty, podle autora základní, atributy bodu zájmu.

Tabulka 9: Srovnání dostupnosti dat z API

Atribut	Foursquare	Yelp	
Jméno			
Popis			
Kategorie			
Souřadnice			
Otevírací doba			
Fotografie		1 fotografie	Až 3 fotografie
Adresa			
Telefonní číslo			
Hodnocení a počet recenzí			
Recenze			
Vzdálenost místa od polohy			
Statistiky (počet návštěvníků)			
Cenová kategorie			
Pokud je buňka podbarvena zelenou barvou, znamená to, že se údaj nachází datech vrácených vyhledáváním, oranžová barva označuje výskyt v detailu nebo pomocí jiného bodu rozhraní a červená úplnou absenci takového údaje.			

V tabulce můžeme jasně vidět, že API Yelpu obsahuje většinu zásadních údajů již při vyhledávání. Pro získání některých z těchto údajů se ale v případě Foursquare budeme muset uchýlit k dalším požadavkům na API, přičemž Foursquare má většinu přístupových bodů, které slouží pro získání bohatších dat, uvedenou s příznakem „Premium“. Kvóta počtu možných požadavků na takové přístupové body je výrazně nižší.

5 OVERTOURISMUS A NÁVRH JEHO PŘEDCHÁZENÍ

5.1 Úvod

Overtourismus je „Situace, kdy nějaké místo navštíví příliš velké množství lidí (turistů) a tím způsobí jeho přeplnění s následkem degradace turistického zážitku a kvality života místních obyvatel.“. [25]

K tomuto problému dochází zejména v turisty oblíbených městech Evropy. Konkrétně Barcelona (Španělsko) vnímá situaci jako velký problém hlavně v turistické špičce. V roce 2018 dle odhadu [26] v této španělské metropoli přebývalo přes noc na 30 miliónů turistů, přičemž stálých obyvatel města bylo dle průzkumu [27] pouze 1 625 137.

Obrázek 7: Overtourismus v Barceloně

Zdroj: vlastní

5.2 Předcházení overtourismu

Při analýze a návrhu aplikace se kromě zdrojů, ze kterých se následně získávají body zájmu, měl najít způsob, kterým se bude předcházet overtourismu/overcrowdingu těchto bodů zájmu. Jako jeden z hlavních a nejspolehlivějších poskytovatelů informací, které se dají takovým způsobem využít jsou mobilní operátoři.

5.2.1 O2 Liberty API [30]

Společnost O2 nabízela službu Liberty API, která byla schopná takové informace zprostředkovat skrze REST API. Tato služba se v době rešerše na bakalářskou práci zdála býti po-

užitelná a aktivní, jenže v průběhu tvorby práce vyšla na povrch skutečnost, že byl ukončen provoz této služby. Následující kapitola bude tedy pojednávat o možném využití služby v případě jejího opětovného uvedení do provozu či vzniku služby podobné.

Mobility API

První ze dvou nabízených koncových bodů Liberty API je Mobility API. Tato API poskytuje agregovaný počet osob přesunutých z lokality A do lokality B za uplynulý den. Je nutné zmínit i fakt, že tento pohyb je jednosměrný, neagreguje tedy počty přesunutých z lokality A do lokality B a z lokality B do lokality A.

Socio-demo API

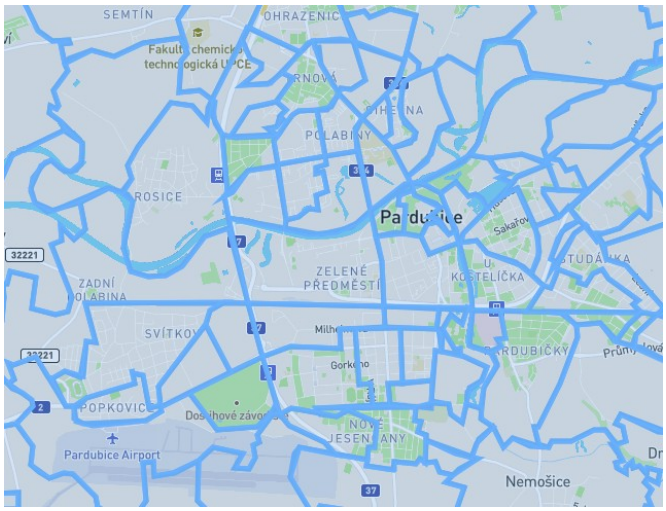
Tato API vrací agregovaný počet osob z dané sociodemografických skupiny, které se v dané lokalitě nacházeli v danou hodinu. Pracuje na základě parametrů: lokalita, hodina a pohlaví nebo věková skupina.

Socio-demo API zaštiťuje dva přístupové body, jeden pro zjišťování počtu na základě věku a druhý na základě pohlaví. První případ vyžaduje jako parametr identifikátor určující věkovou skupinu (1 – 8-19 let; 2 – 19-25 let; 3 – 26-35 let; 4 – 36-55 let; 5 – 55 let a více). Druhý přístupový bod pak identifikátor pohlaví (1 – muž, 2 – žena).

Zjištění lokality

Pro obě API spadající pod Liberty API platí, že jsou jejich lokality určené a vymezené základní sídelní jednotkou (ZSJ) nebo základní uzemní jednotkou (ZÚJ). Pro jejich vyhledávání lze využít databázi IRSO poskytovanou Českým statistickým úřadem, nicméně to by pro náš případ použití nebylo vhodné, vzhledem k absenci veřejného rozhraní systému pro aplikační využití. Společnost O2 naštěstí poskytla na webu s dokumentací API vymezení polohy jednotek ve formátu GeoJSON.

GeoJSON je formát umožňující uložení řady geografických datových struktur. Mezi tyto struktury patří například *Point*, jehož metadata obsahují souřadnice jeho polohy či *Polygon*, který je určený množinou souřadnic tvořících mnohoúhelník. [31] S tímto formátem je pak daleko jednodušší pracovat, než s webovou aplikací ČSÚ. Například aplikace GNOME Maps pro desktopové prostředí GNOME (GNU/Linux, FreeBSD, Solaris) je schopná takto určené oblasti přímo zobrazit v mapě.



Obrázek 8: Ukázka vymezení sídelních jednotek obce Pardubice na mapě

Zdroj: vlastní

Jak můžeme vidět na obrázku 8, tak jednotlivé ZSJ mohou představovat třeba části nebo čtvrti města. Způsob zobrazení, který je vidět na mapě je sice vhodný pro člověka, aby si našel adresu na mapě a k ní odpovídající ZSJ, ale opět nám neumožňuje ji určit ze souřadnic, což potřebujeme pro nejefektivnější určení míry „overcrowdingu“ pro nalezené body zájmu, jejichž poloha je vždy určena právě souřadnicemi. V další si tedy popíšeme, co vše muselo být provedeno, aby bylo možné zjistit ZSJ na základě souřadnic.

Formát GeoJSON lze importovat do řady databází podporujících prostorové datové typy, ať už se jedná o relační či grafové databáze. Po rešerši na zjištění nejvhodnější databáze jsem se rozhodl pro ukládání a práci s daty použít takovou, se kterou mám největší zkušenost, tedy MySQL. MySQL podporuje prostorové datové typy, jejich supertypem je typ *geometry*. Konkrétními zástupci množiny těchto typů jsou například *Point, Line, Polygon,* atd. Co je pro nás důležité je podpora prostorových funkcí, zejména pak funkce určující, zda se daný bod (souřadnice) nachází ve vymezeném prostoru. [32]

Soubor GeoJSON byl tedy naimportován do databázové tabulky obsahující sloupce: kód sídelní jednotky (varchar) a tvar určující sídelní jednotku (geometry). Import byl proveden pomocí jednoduchého skriptu v jazyce PHP. Na databázi se následně bylo možné dotazovat a zjistit kód sídelní jednotky na základě souřadnic. Takový způsob dotazování byl však neefektivní a v aplikaci nepoužitelný, kvůli dlouhé době provádění – určit kód jednoho páru souřadnic zabralo až 3 sekundy, přičemž v aplikaci bude nutné zjišťovat až desítky kódů najednou a bude ji vyžívat více uživatelů. Pro optimalizaci se provedlo následující: při impor-

tu se pro každou sídelní jednotku zjistily souřadnice nejmenšího možného čtyřúhelníku do kterého se její oblast vejde. Při určování kódu sídelní jednotky se nejprve výrazně omezilo množství záznamů, nad kterými se musela provádět výpočetně náročná operace určující zda bod patří do oblasti určené množinou bodů – mnohoúhelníku. Optimalizace zkrátila čas provádění na jednotky až desítky milisekund oproti původním jednotkám sekund.

Určení přelidnění oblasti

K určení míry přelidnění oblasti autor považuje za nejvhodnější volbu využití Socio-demo API za použití přístupového bodu vracejícího počet osob na základě pohlaví. Autor doporučuje tuto možnost z toho důvodu, že je z navrácené hodnoty jednodušší odhadovat celkový počet osob v oblasti. Reálná implementace by tedy zahrnovala zjištění počtu osob v lokalitě pro jedno z pohlaví a na základě podílu daného pohlaví v populaci spočítala přibližně celkový počet osob v lokalitě.

Proces určení by ale nemohl být takhle přímočarý. Jak můžeme pozorovat na obrázku 8, každá oblast má jinou velikost. Z tohoto důvodu by se tedy musela vyjádřit hustota osob v oblast, jako jedna z možností se nabízí: počet osob na km^2. Tuto hodnotu autor zamýšlel použít jako ukazatel přelidněnosti oblasti a množinu bodů zájmu nabízených uživateli dle této hodnoty patřičně upravit. První by se uživateli ukazovaly body zájmu, které leží v lokalitě s nejnižším počtem osob na km^2.

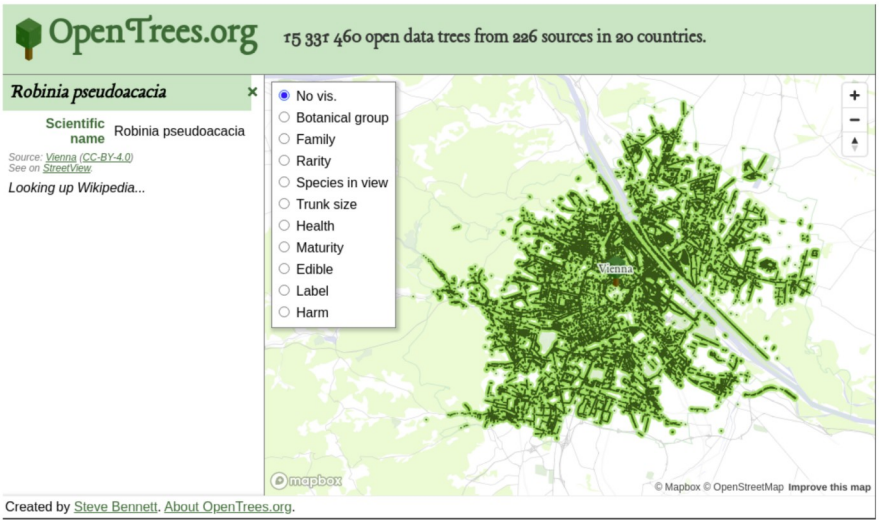
6 APLIKAČNÍ VYUŽITÍ OTEVŘENÝCH DATOVÝCH SAD [33]

Otevřená data jsou dle jejich definice v § 3 odst. 11 zákona č. 106/1999 Sb. O svobodném přístupu k informacím „… informace zveřejňované způsobem umožňujícím dálkový přístup v otevřeném a strojově čitelném formátu, jejichž způsob ani účel následného využití není omezen a které jsou evidovány v národním katalogu otevřených dat.“. [34] Jsou efektivním způsobem poskytování informací veřejného sektoru.

6.1 Příklady použití otevřených datových sad [35]

6.1.1 OpenTrees.org [36]

„OpenTrees.org“ je projekt se zaměřením na životní prostředí realizovaný městem Vídeň (Rakousko). Je založen na otevřených datech z více než 150 světových měst a díky tomu poskytuje největší takovou databázi na světě. Databáze obsahuje data o stromech nacházejících se v ulicích a parcích měst. Ke stromům jsou uvedeny komplexní data, mezi které patří: čeleď, rod, latinský název, ale i data o konkrétním stromu jako jeho velikost kmenu, kondice, míra vyspělosti nebo jestli na něm rostu jedlé plody. Tato data zobrazuje na mapě OpenStreetMap.



Obrázek 9: Město Vídeň na OpenTrees.org

Zdroj: vlastní

6.1.2 Betterplace [37]

„Betterplace“ je španělská společnost, která založila své fungování na otevřených datech místní vlády a státních institucí. Jejich platforma je geolokační aplikací, která vyhodnocuje vztahy mezi sadami dat a na základě této analýzy vytváří informace využitelné pro podniky v oblasti. Tím poskytuje informace na základě kterých může podnik optimalizovat využití zdrojů, času apod.

6.2 Využití otevřených dat v aplikaci

Při rešerši na téma bakalářské práce autor našel řadu API nabízejících data zaměřená na problematiku viru COVID-19. Nicméně, stejně jako stav pandemické situace a představitel ministra zdravotnictví České republiky se i tyto API často mění a validita jejich dat je nejasná. Proto se autor rozhodl pro získávání dat ohledně viru využít otevřené datové sady Ministerstva zdravotnictví České republiky, které jsou aktualizovány v některých případech i několikrát denně a poskytují oficiální čísla. Data jsou poskytovány ve formátech JSON a CSV.

Konkrétní využití dat z otevřených datových sad v aplikaci spočívá v periodickém stahování datové sady obsahující epidemiologickou charakteristiku obcí České republiky. Na základě dat z datové sady se aktualizují počty nakažených vztažené k bodům zájmu v aplikaci a tato čísla jsou následně prezentována uživatelům pro informaci.

6.2.1 Epidemiologická charakteristika obcí

Popis datové sady

Jak již bylo zmíněno, využitá datová sada zobrazuje epidemiologická data k obcím. Musel být tedy nalezen způsob, jak data obcí přiřazovat bodům zájmu – hlavní zobrazované entitě v aplikaci. Ve věci této problematiky byl využit LAU2 kód obce, který je obsažen v této datové sadě a každá obec ho má, v datové sadě ho definuje atribut *obec_kod*. Pro lepší představu možnosti využití bylo vybráno několik nejdůležitějších atributů sady do následující tabulky.

Tabulka 10: Atributy datové sady epidemiologické charakteristiky obcí

Atribut	Datový typ	Poznámka
den	string	Den v týdnu
datum	date	Datum ve formátu „RRRR-MM-DD“
obec_kod	integer	Kód obce
nove_pripady	integer	Počet nových případů od poslední aktualizace.
aktivni_pripady	integer	Aktivní případy.

nove_pripady_7_dni	integer	Počet případů za posledních 7 dní.
nove_pripady_14_dni	integer	Počet případů za posledních 14 dní.

Nutné operace k použití datové sady v aplikaci

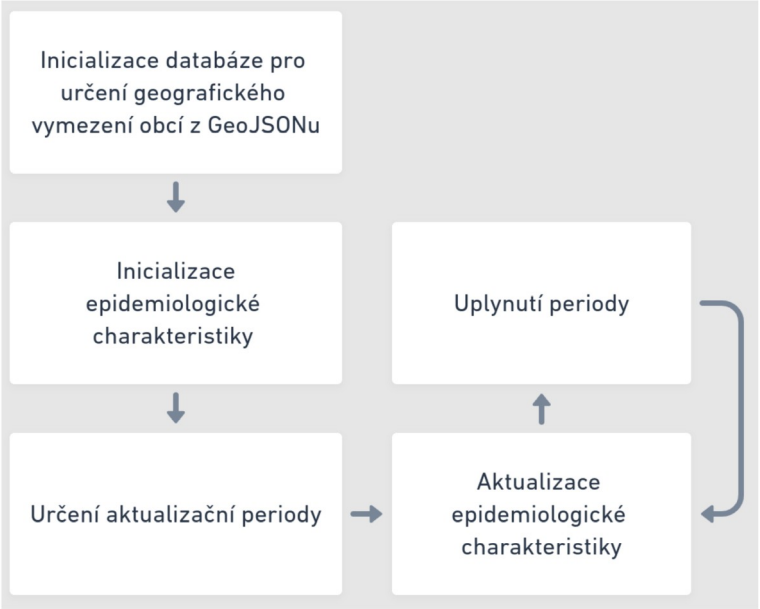
Pro párování dat z datové sady se tedy použil LAU2 kód obce, který každou obec jednoznačně identifikuje. Tento údaj je však velice specifický pro data státní správy a není běžným parametrem adres. I přesto je takové operace v aplikaci docíleno s pomocí údajů o určení geografického vymezení obcí, k čemuž byl využit již popsaný formát GeoJSON. Určení obcí v tomto formátu autor nenalezl, proto musel najít řešení v použití jiného formátu a následného převodu do toho požadovaného.

Na webu Českého úřadu zeměměřičského a katastrálního je k dispozici geografické určení oblastí obcí ve formátu DBF. Za použití webové ho nástroje na adrese *https://mapshaper.org/* bylo možné data převést do formátu GeoJSON. Pro snížení objemu dat se zachováním co největší věrnosti originálu nabízí nástroj zjednodušení mnohoúhelníků určujících oblast obce za použití několika metod. Byla zvolena metoda „Visvalingram / weighted area“ při zachování 25% původní velikosti.

Obrázek 10: Mnohoúhelníky zobrazené na mapě Pardubic a okolí

Zdroj: vlastní

Do databáze aplikace byly tyto mnohoúhelníky vloženy s použitím prostorových databázových polí. Ke každému záznamu byl uveden i kód obce, který slouží k pozdějšímu přiřazení epidemické charakteristice k obci.



Obrázek 11: Diagram aktualizace epidemiologické charakteristiky

Zdroj: vlastní

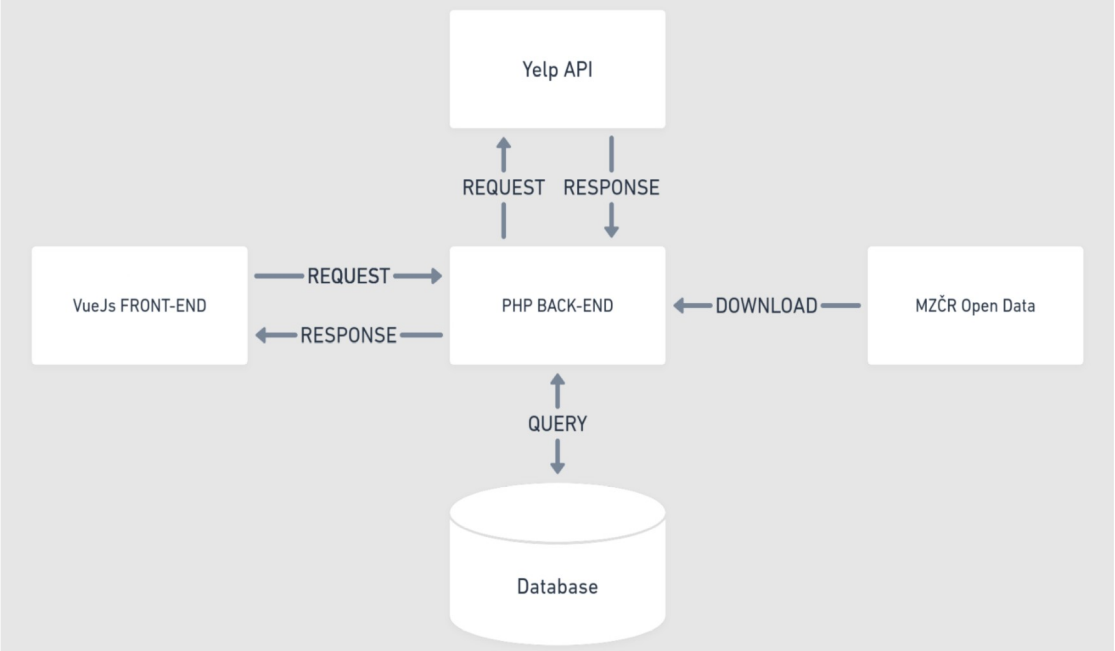
7 WEBOVÁ APLIKACE A POUŽITÉ TECHNOLOGIE

Webová aplikace slouží jako praktická ukázka výstupu analýzy API geosociálních sítí a možnosti jejich využití v praxi. Využívá technologie, jež byly pro jejího autora nejznámější a s kterými má prozatím největší zkušenosti v jednotlivých vrstvách architektury.

7.1 Architektura aplikace

U této aplikace a jí podobných je důležité oddělení její logické a prezentační části. To umožňuje snadnější a efektivnější správu a rozšiřitelnost. Díky tomu je možné využívání její logické části i napříč více klienty, to by se mohlo hodit v budoucnu v případě vývoje například mobilní aplikace. Mohla by využívat stejnou logickou část a prezentovat obsah svým způsobem.

Základem na straně serveru je aplikace zodpovědná za poskytnutí dat aplikaci běžící na straně klienta. Serverová aplikace je napsaná v jazyce PHP 8.0 s využitím frameworku Symfony 5 a je připojena na databázi MySQL, zatímco klientská aplikace běží na frameworku Vue.js v jazyce JavaScript.



Obrázek 12: Digram architektury aplikace

Zdroj: vlastní

Serverová aplikace zprostředkovává data získaná z API geosociální sítě aplikaci klientské. Back-end aplikace spolu s databází tak tvoří jakýsi mezičlánek architektury. Díky tomu je

umožněna větší kontrola nad přenášenými daty a je umožněno nad nimi provádět nějaké operace, ukládat je do mezipaměti nebo je obohacovat o související data z jiného zdroje.

7.2 PHP [43], [44]

Hypertext Preprocessor (PHP) je široce používaný, „open-source“, typově dynamický skriptovací jazyk zejména vhodný na tvorbu webových stránek. Jeho kód je v daném případě použití vykonáván na straně serveru a poté zaslán klientovi webovým serverem. Cílem PHP je být přístupné pro nového uživatele a umožňovat rychlý a jednoduchý začátek programování v něm. Mezi jeho výhody patří:

- Má syntaxi podobnou jazyku C. Díky tomu je většině vývojářů blízký.
- Je open-source s rozsáhlou komunitou.
- Jednoduše lze implementovat na webový server Apache.
- Snadno komunikuje s databází, ať už se jedná MySQL, PostgreSQL.
- Je multiplatformní.
- Velké množství funkcí v jádře.

Kromě výhod samozřejmě existuje řada nevýhod. Mezi hlavní patří, dle autora, slabší podpora objektového programování. PHP sice podporuje základy objektového programování, jako tvorbu tříd a rozhraní, dědění tříd a implementace rozhraní, ale vzhledem k dynamickému typování není možné například přetěžování metod. Jeho poslední verze 8.0 také řeší, mimo jiné, často kritizované nepředvídatelné porovnávání řetězců s čísly. [45]

V PHP lze vytvořit webové aplikace od internetových obchodů, přes diskuzní fóra po redakční systémy. I díky tomu 79,2% webových stránek používá PHP. [46]

7.2.1 Symfony

Symfony je množina komponent pro vývoj webových aplikací v jazyce PHP. Tyto komponenty společně tvoří Symfony Framework. Na jeho vývoji se podílí na 3000 vývojářů z celého světa, je open-source. [47]

Příklad použití Symfony Route anotace pro směrování na základě URL a omezení HTTP metody:

```php
<?php

namespace App\Controller;

/**
 * @Route ("/api/venue", name="api_venue")
 */
class VenueController extends ApiController
{
    /**
     * @Route("/", name="_all", methods={"GET"})
     */
    public function getVenues(Request $request): JsonResponse {...}

    /**
     * @Route("/{id}", name="_detail", methods={"GET"})
     */
    public function getVenueDetail(Venue $venue): JsonResponse {...}
}
```

7.3 HTML [48]

HyperText Markup Language je značkovací jazyk založený na XML a je základním stavebním kamenem vývoje webových stránek. Není to programovací jazyk, takže neumožňuje vytvoření dynamické funkcionality. Za jeho použití je umožněno tvořit strukturu webových stránek a aplikací pomocí sekcí, odstavců, odkazů a dalších prvků. Zápis takových struktur probíhá pomocí HTML „tagů“ a jejich atributů, které definují určení a význam prvku na stránce. Soubor obsahující definici HTML stránky lze editovat v jakémkoliv textovém editoru a následně uložit s příponou *html* nebo *htm*. Je ideální použít editor uzpůsobený k editaci nebo s podporou jazyka HTML, aby mohl být kodér upozorněn na případné chyby v jeho kódu.

První verze HTML byla publikována v roce 1991 a vyvinul ji Tim Berners-Lee. Tehdy HTML obsahovalo definici pouze 18 tagů, ale od té doby v průběhu vývoje další přibývají a současná verze 5, vydaná v roce 2014, jich obsahuje přes 100. Díky rychlému nárůstu popularity se stalo později HTML webovým standardem a je spravováno World Wide Web Consortium (W3C).

Kořenovým tagem je tag *html*, který obaluje veškerý obsah stránky. Každá taková stránka musí obsahovat také tagy *head* a *body*, sloužící k definici meta-informací o stránce (název stránky, znaková sada apod.), respektive k obalení obsahu určeného k vykreslení. Používané tagy mohou být párové (*<h1>Nadpis</h1>*) či nepárové (*
). V případě párových tagů je zakázáno jejich křížení (<h1>Fakulty<h2>Fakulta elektrotechniky a informatiky</h1></h2>*).

Příklad zápisu HTML:

```
<html>
  <head>
    <title>Název stránky</title>
    <meta charset="UTF-8">
  </head>
  <body>
    <h1>Nadpis první úrovně</h1>
    <br />
    <a href="/dalsi-stranka.html">Další stránka</a>
  </body>
</html>
```

7.4 CSS

7.4.1 Základní informace

CSS neboli Cascading Style Sheets, kaskádové styly, je jazyk pro definici prezentace webových stránek ve smyslu barev, rozložení či písma a je nezávislý na HTML, nicméně se v kombinaci s ním používá nejčastěji. Za použití tohoto jazyka je umožněno prezentovat například jednu webovou stránku několika rozdílným zařízením bez nutnosti změny stránky samotné. Mezi zařízení, dle kterých jsou CSS schopné rozlišit prezentaci, mohou patřit zařízení s malými či velkými obrazovkami, ale například i tiskárny. [49]

Práce s CSS spočívá v definici tzv. vlastností elementů. S jeho je možné definovat velmi široké spektrum stylů, pomocí kterých je umožněno formátování textu (velikost písma, prokládání, kapitálky), změnit vnitřní nebo vnější odsazení jednotlivých prvků, reagovat na přejetí myší po prvku, nastavit okraje a mnoho dalšího. [50]

Styly je možno deklarovat třemi způsoby:

1. Přímo u konkrétního HTML elementu v jeho atributu *style*. Takový styl se nazývá přímý.
2. V hlavičce stránky uvést stylopis uvnitř tagu *style*.
3. Pomocí metatagu *link* v hlavičce webu uvést odkaz na externí stylopis. To má výhodu například v možnosti uvedení stejného stylopisu na několika stránkách pro docílení stejného vzhledu. [50]

Pokud bychom chtěli docílit například změny barvy a dekorace nadpisu první úrovně stránky, je možné toho docílit za pomocí deklarace zmíněnými třemi způsoby, ty budou uvedeny v další kapitole. První způsob deklarující styl přímo v atributu elementu se hodí pro formátování konkrétního prvku, má totiž největší váhu a vlastnosti v něm definované přepíší ty definované ve stylopisu. Tato vlastnost platí i pro stylopisy, takže poslední importovaný stylopis přepisuje vlastnosti těch předchozích. [50]

V externích stylopisech je nutné uvádění tzv. selektoru, který slouží k definici množiny elementů, na které se daná množina stylů aplikuje. Selektorů máme několik druhů a liší se ve

způsobu deklarace. Selektory mohou provádět výběr množiny prvků na základě jejich: tagu, identifikátoru, třídě nebo pseudo-třídě a tyto způsoby se dají kombinovat a řetězit. Taková konstrukce umožňuje uživateli zacílit daný prvek s jistou toho, že změna stylopisu neovlivní vzhled stránky jinde. [51]

7.4.2 Příklady deklarací

Deklarace pomocí přímého stylu

```
<h1 style="color: blue; text-decoration: underline">Nadpis</h1>
```

Deklarace stylopisem v hlavičce stránky

```
<style>
h1 {
  color: blue;
  text-decoration: underline;
}
</style>
```

Deklarace v externím stylopisu

Toto řešení vyžaduje uvedení odkazu na stylopis v hlavičce stránky:

```
<link rel="stylesheet" type="text/css" href="styly.css">
```

Obsah souboru *styly.css*:

```
h1 {
  color: blue;
  text-decoration: underline;
}
```

7.5 JavaScript [52]

JavaScript je interpretovaný programovací jazyk, který je hojně využíván na webových stránkách. Lze ho zapisovat přímo do HTML kódu v tagu *script* a nebo ho importovat pomocí metatagu *link*, podobně jako stylopisy. Skripty se typicky provádí na straně klienta, avšak v dnešní není výjimkou i jeho provoz na straně serveru. Nicméně v této kapitole bude popsáno jeho využití u klienta. Jeho syntaxe je založena či inspirována mj. jazykem Java, přičemž s ním má společnou i skutečnost, že je objektově orientovaný.

JavaScript spouští na straně klienta webový prohlížeč, proto může docházet k různé interpretaci jazyka různými prohlížeči, což je skutečnost na kterou si jeho vývojář musí dát pozor. K jeho omezením patří mimo jiné, že interpretace může být zakázána klientem a také nenabízí možnost přímé práce se soubory nebo interakce s operačním systémem. Ukládání dat je umožněno pomocí Cookies.

7.5.1 Vue.js [53]

Vue.js je framework pro jazyk JavaScript sloužící k vytváření uživatelské rozhraní. Byl vyvinut a navrhnut s cílem vytvoření ekosystému, ve kterém je možné vytvářet rozsáhlé Single-Page aplikace i knihovny s menšímu rozsahem.

Vue.js lze integrovat do projektu pomocí importu jejich vývojové nebo produkční verze skriptu. To v čem spočívá síla Vue, je reakivita, které je docíleno svázáním datových složek komponent s DOM (Document Object Model) stránky.

Vue.js aplikaci lze sestavovat z tzv. komponent. Při vývoji aplikace je velmi příhodné její strukturu rozdělovat mezi komponenty, což umožňuje lepší škálovatelnost aplikace, zvýší přehlednost kódu a takto definované komponenty lze znovu použít na jiném místě v aplikaci.

Po vytvoření instance celé Vue.js aplikace se musí tyto komponenty k instanci zaregistrovat, aby bylo umožněno jejich použití.

Registrace komponenty v souboru JavaScriptu:

```
Vue.component('button-counter', {
  data: function () {
    return {
      count: 0
    }
  },
  template: '<button @click="count++">Počet kliknutí: {{ count
}}</button>'
})

new Vue({ el: '#app' })
```

Použití takové komponenty v souboru HTML:

```
<div id="app">
  <button-counter></button-counter>
</div>
```

Tento způsob registrace komponent však není vhodný pro rozsáhlejší aplikace. V případě větších aplikací se komponenty rozdělují do vlastních souborů, kde je jejich definice přehlednější. Pokud je výše zmíněná komponenta definována ve vlastním souboru, jeho obsah vypadá takto.

```
<template>
  <button @click="count++">Počet kliknutí: {{ count }}</button>
</template>

<script>
export default {
  name: "button-counter",
  data: function () {
    return {
      count: 0
    }
  }
}
</script>
```

Zdroj: vlastní

7.5.2 MVVM [54]

Framework Vue.js využívá ke svému fungování návrhový vzor MVVM (Model ViewModel View), který má za cíl oddělení logiky aplikace od uživatelského rozhraní a umožnění snazšího spravování a škálování aplikace.

Model představuje vrstvu s daty aplikace a definuje její rozhraní. V případě implementované aplikace do této vrstvy patří *Vuex* a služby pro komunikaci s API.

Vrstva **ViewModel** komunikuje s modelem a poskytuje vrstvě View hodnoty, které má tato vrstva pozorovat a zobrazovat jejich hodnoty. Ve Vue to zastupují tuto funkcionalitu „computed“ proměnné a reaktivní proměnné.

Vrstva **View** má za úkol zobrazování hodnot poskytnutých ViewModelem v uživatelském rozhraní a interakci s uživatelem. Pro Vue je tato vrstva představena Document Object Modelem stránky, který komunikuje s ViewModelem ve formě událostí.

7.5.3 Použité knihovny

V klientské aplikaci je použita řada knihoven nejen pro docílení určitých funkcionalit, ale i ulehčení práce a úspoře času autora. Jsou použity naskrz celou aplikací, od knihoven poskytujících komponenty pro tvorbu šablon ve stylu „Material designu“ po ty, které usnadňující provádění HTTP požadavků a tedy komunikaci s API. Pro správu knihoven byl využit nástroj *npm*.

npm

Je to nástroj, který mimo jiné skrze rozhraní příkazové řádky umožňuje instalaci balíčků včetně kontroly jejich závislostí, spouštění lokálního vývojového serveru nebo vytvoření sestavení celé aplikace optimalizovaného pro produkci. Soubor *package.json* pak popisuje závislosti (určené pro sestavení nebo pouze pro vývoj) a další informace o projektu.

Axios

Pro volání API byla využita knihovna *Axios*, což je HTTP klient, který skrze jeho rozhraní zpřístupňuje a usnadňuje používání HTTP požadavků. Jeho důležitou vlastností je návrat hodnot typu *Promise*. *Promise* je objekt představující eventuální dokončení či selhání asynchronní a jejich hodnot. [55] Příklad typického použití je možné vidět níže. Probíhá v něm zasílání HTTP GET požadavku na REST API za účelem získání komentářů uživatele. Po dokončení požadavku, respektive obdržení odpovědi se provede kód metody *then*, kde lze zpracovat obsah odpovědi na požadavek. Metoda *catch* je provedena v případě chyby a metoda *then* na ni navazující v případě úspěchu i chyby. Tato konstrukce tedy může do jisté míry připomínat *try-catch-finally* konstrukci známou z jiných jazyků.

```
axios.get('/user/1/comments')
  .then((response) => {
    // handle success
    console.log(response);
  })
  .catch((error) => {
    // handle error
    console.log(error);
  })
  .then(() => {
    // always executed
  });
```

Zdroj: vlastní

Material Design

Knihovna Material Design poskytuje jejímu uživateli prvky určující rozložení stránky či formulářové prvky, jejichž vzhled byl upraven v souladu s designovým systémem Material Design od společnosti Google. Tento systém byl vyvinut za účelem vytvoření konvence, které mohou pomáhá k vytváření přehledných a funkčních uživatelských rozhraní.

V praxi se tak například místo použití prvku *button* použije prvek *md-button*, který má implicitně nastavený vzhled, respektive jsou do jeho atributu třídy vloženy třídy, které definují množinu vlastností pro docílení vzhledu dle Material Designu.

Google Maps

Jednou ze stěžejních funkcí aplikace je zobrazování bodů zájmu na mapě. Google poskytuje JavaScript rozhraní a možnost si mapu na stránku vložit, knihovna na této skutečnosti staví a integruje mapu do ekosystému Vue, čímž s ní výrazně ulehčuje práci. Do šablon Vue tak stačí vložit element *GMap* a do něj například elementy *GmapMarker* pro zobrazení mapy a „špendlíků“ v ní.

7.5.4 Komunikace s API

Pro komunikaci s API byl použit klient z knihovny *Axios*. Aby byl skrze komponenty aplikace definován jednotný způsob komunikace, byla vytvořena třída *ApiService*. Tato třída využívá vytvořenou instanci klienta *Axios* a definuje své rozhraní několika metodami, přičemž každá metoda představuje HTTP metodu.

```
const api = axios.create({...})

class ApiService {
  post(endpoint, data) {
    return api
      .post(endpoint, data)
      .then(
        response => response.data,
        error => error.message
      );
  }

  get(endpoint) {
    return api
      .get(endpoint)
      .then(response => response.data);
  }

  ...
}
```

Zdroj: vlastní

Třídu *ApiService* využívají další třídy pro konkrétní implementaci komunikace jednotlivých entit. V následujícím fragmentu kódu třídy *VenueService* můžeme vidět metodu *getFiltered(filters)*, která slouží pro získání množiny bodů zájmu z API na základě filtrů, její implementace je v ukázce skryta. Další metoda v ukázce, tedy metoda *getDetail(venueId)*, slouží k získání jednoho JSON objektu bodu zájmu na základě jeho identifikátoru. V těle této metody lze vidět způsob použití zmiňované třídy *ApiService*, konkrétně její metody *get*.

```
class VenueService {
      getFiltered(filters) {...};

      getDetail(venueId) {
          return ApiService
            .get('/venue/' + venueId + '/detail')
            .then(data => data);
      };

      ...
}
```

Zdroj: vlastní

7.5.5 Správa stavu aplikace

Pro centralizované ukládání stavu dat a rozhraní pro manipulaci s těmito daty, bylo využito *Vuex*. Nástroj *Vuex* to umožňuje svou konstrukcí, která obsahuje data aplikace (vyhledané bod zájmu, oblíbené body zájmu, stav načítání apod.) a definici metod pro jejich manipulaci. V příkladu je vidět část *Vuex* zodpovědná za správu stavu načítání, komponenty v aplikaci dle

tohoto stavu přizpůsobují své chování či vzhled. Při načítání například nejde spustit vyhledávání a v aplikaci je zobrazena animace načítání.

```
{
  state: {
    loading: true
  },
  getters: {
    isLoading: (state) => state.loading,
  },
  mutations = {
    setLoading(state, loading) {
      state.loading = Boolean(loading)
    }
  }
}
```

Zdroj: vlastní

7.6 Databáze

7.6.1 Úvod

Databáze je organizovaná množina strukturovaných informací či dat na paměťovém médiu a je spravována SŘBD (Systémem Řízení Báze Dat), v angličtině DBMS (Database Managment System). Databáze a její řídící systém tvoří databázový systém nebo zkráceně databázi. V případě relačních databází se software, který ji spravuje nazývá RDBMS (Relational Database Managment System). [38]

V relačních databázích jsou data strukturována do tabulek, které obsahují sloupce a řádky. [38] Databáze je pak tvořena množinou tabulek, které mezi sebou mohou mít vztahy realizované pomocí cizích klíčů – odkazů na konkrétní záznam v jiné tabulce. Takovým způsobem je možné vytvořit určitou abstrakci nad daty a vytvářet mezi nimi hierarchii.

7.6.2 Dotazovací jazyk SQL

Structured Query Language (SQL) je programovací jazyk, který lze použít pro provádění operací nad většinou relačních databází. Mezi tyto operace patří: dotazování, manipulace a definice dat nebo řízení přístupu. [38]

Základními příkazy nebo klauzule jazyka SQL pro manipulaci s daty jsou:

- Pro získávání dat z databáze se využívá příkaz *SELECT*. V uvedeném příkladu použití jsou specifikovány 3 sloupce dané tabulky. Výsledkem dotazu bude množina řádků tabulky s uvedenými sloupci.

  ```
  SELECT jmeno, prijmeni, datum_narozeni
  FROM uzivatele;
  ```

- Ke vkládání dat do databáze lze využít příkaz *INSERT INTO*. V příkaze je nutné specifikovat všechny sloupce tabulky, které mají příznak *NOT NULL*.

```
INSERT INTO uzivatele (jmeno, prijmeni, datum_narozeni)
VALUES ('Filip', 'Odvárka', '1999-06-06');
```

- Klauzule *WHERE* se používá k filtrování výsledků dotazu či omezení množiny řádků, nad kterou bude dotaz proveden. Toho je docíleno použitím porovnávacích operátorů. V příkladu je ilustrováno použití klauzule pro získání uživatelů narozených v tomto tisíciletí.

```
SELECT *
FROM uzivatele
WHERE datum_narozeni >= '2000-01-01';
```

- Další možnou operací nad daty v databázi je jejich aktualizace. Pro změnu hodnot databázových sloupců se používá příkaz *UPDATE*. Příklad znázorňuje změnu hodnoty sloupce *jmeno* pro záznam definovaný hodnotou *id*.

```
UPDATE uzivatele
SET jmeno = 'Tomáš'
WHERE id = 1;
```

- Mazání záznamů může být docíleno použitím příkazu *DELETE*. V případě vynechání klauzule *WHERE* dojde k odstranění všech záznamů tabulky.

```
DELETE FROM uzivatele;
```

7.6.3 Databáze v aplikaci

Jako konkrétní implementace databáze pro webovou aplikaci byla vybrána relační databáze MySQL od společnosti Oracle. Byla zvolena z důvodu její spolehlivosti, kvalitní dokumentace a v neposlední řadě také jednoduché integrace do serverové aplikace. Databáze MySQL byla použita ve verzi 8.0.

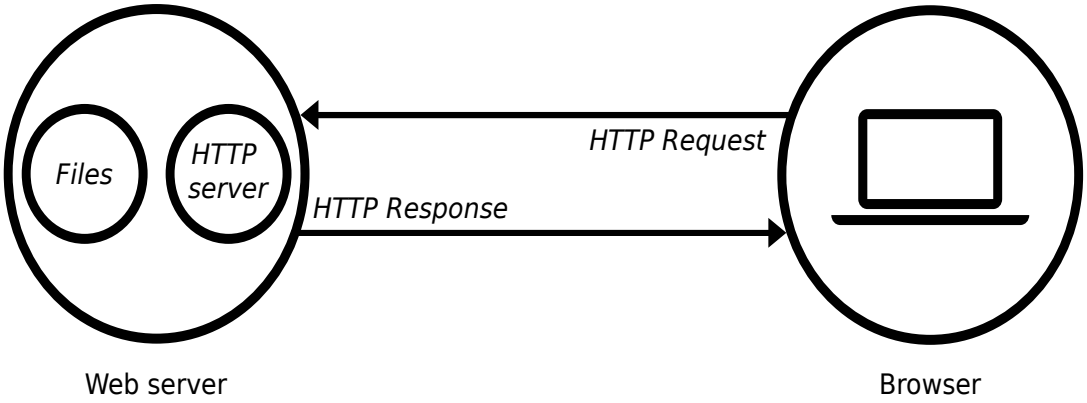
Pro správu databáze se používaly nástroje Adminer 4.8, který běží na webovém serveru s podporou jazyka PHP, který ke komunikaci s databází používá rozšíření MySQLi a DataGrip, což je databázový klient od společnosti JetBrains.

7.7 Webový server [39]

Z pohledu hardwaru lze webový server považovat za počítač na kterém běží software webového serveru a jsou uložené soubory webových komponent. Mezi ty můžou patřit: HTML dokumenty, obrázky, CSS stylopisy nebo i soubory JavaScriptu. Důležité je uvést, že tento počítač je připojen do sítě internetu a tímto kanálem je schopen komunikovat s ostatními zařízeními k internetu připojenými.

Když se na webový server podíváme ze strany software, je to nástroj obsahující několik částí určujících, jakým způsobem může klient přistupovat k souborům na webovém serveru. Minimální potřebná funkcionalita je server HTTP, což je software podporující webové adresy URL a je schopný zpracovávat požadavky protokolu HTTP. V návaznosti na takové poža-

davky poté server odpovídá obsahem, může být statický nebo dynamický. Statický odesílá klientovi obsah v takové podobě, v jaké je na serveru uložen. Oproti tomu dynamický disponuje navíc dalším softwarem, nejčastěji aplikačním serverem a databází. Ten potom generuje obsah, který je klientovi zaslán jako odpověď na jeho požadavek. V případě aplikace popisované v této práci je použit server dynamický. Webový server funguje na architektuře klient-server.



Obrázek 13: Znázornění komunikace mez klientem a serverem

Zdroj: [39]

Jako konkrétní software webového serveru byl zvolen Apache ve verzi 2.4. Je široce používaný a podporuje řadu operačních systému, včetně unixových systémů i Windows a především nabízí podporu jazyka PHP, ve kterém je back-end aplikace naprogramován.

7.8 Datové struktury, entity a jejich vztahy

Při návrhu jakékoliv aplikace je velmi důležité správně navrhnout datové struktury a entity, které se budou využívat pro strukturované uchování dat v kódu aplikací. Entita, která je poté v programu reprezentována třídu či jinou strukturou, nám dává možnost abstrakce nad jinak na první pohled nesouvisejícími množinami dat. Nad takto reprezentovanou entitou lze pak provádět operace, zobrazovat ji apod.

Framework použitý pro vývoj serverové aplikace velmi usnadňuje práci s takovými entitami. S pomocí jeho rozhraním v příkazové řádce lze do projektu přidávat entity velice jednoduše, pro vytvoření entity se spustí průvodce s jehož pomocí lze konfigurovat její vlastnosti. Vlastnosti se pojmenují, zvolí se jejich datový typ, k datovému typu se následně váže procedura jeho nastavení, která se pro různé typy liší – číselný datový typ má jiné možnosti a úroveň nastavení než například datum. Na základě poskytnutých informací nástroj vygene-

ruje soubor s třídou reprezentující entitu. Ta obsahuje všechny vlastnosti ve formě atributů a metody k jejich získávání či nastavování z vně třídy. Takto vygenerovaná třída dodržuje techniku zapouzdření. Nástroj nezůstane jen u toho a vygeneruje entitě i třídu sloužící k jejímu získávání z databáze.

Symfony používá Object-Realational Mapping (ORM) pro konverzi dat mezi jejich databázovou a objektovou reprezentací. To nám velice usnadňuje načítání entit z databáze, protože je nemusíme načítat do struktury z dotazu, ale je to řešeno implicitně na úrovni vrstvy ORM. Třídu pro takovéto mapování vytvoří nástroj *symfony console* společně s entitou a obsahuje základní metody pro získávání dat – načtení entity dle jejího id, hodnoty vlastnosti či načtení všech entit daného typu.

7.8.1 Venue

Entita *Venue* (místo/bod zájmu) je tou nejdůležitější a nejzákladnější v aplikaci, všechny ostatní jsou na ní nějakým způsobem závislé, spojuje je do celku jež je prezentován uživateli aplikace.

Její vlastnosti jsou:

- Hodnotové vlastnosti, které neuchovávají odkaz na jinou entitu jsou: *name*, *latitude*, *longitude*, *price_tier* a *yelpId*. Poslední ze zmíněných vlastností je referenční identifikátor ze systému Yelp. Obsahuje také hodnotu držící datum a čas posledního aktualizování detailu místa.
- Další vlastnosti slouží k referenci na entity jiného typu. Mezi referencované entity patří: Address, ContactInfo, Rating a CovidData.

7.8.2 Address

Entita *Address* slouží k uchování dat ohledně adresy daného bodu zájmu a kardinalita vztahu k *Venue* je 1:1. Vznikla primárně za účelem logického rozdělení atributů entity *Venue*. Vlastnosti entity *Address* jsou:

- *address* – Obsahuje název ulice a číslo popisné.
- *postal_code* – Obsahuje poštovní směrovací číslo adresy.
- *city* – Obsahuje název města adresy.
- *country* - Obsahuje dvouznakový název země adresy.

7.8.3 ContactInfo

Pro uložení kontaktních údajů byla vytvořena entita *ContactInfo*. Kardinalita jejího vztahu k *Ven*ue je 1:1 a její vlastnosti jsou:

- *phone* – Neformátovaný řetězec obsahující telefon daného bodu zájmu.
- *formatted_phone* – Formátovaný řetězec obsahující telefon daného bodu zájmu.
- *facebook* – Obsahuje ID bodu zájmu na straně Facebooku. Lze využít k vytvoření odkazu na stránku tohoto bodu zájmu v sociální síti Facebook.
- *facebook_name* – Obsahuje název bodu zájmu na straně Facebooku.
- *instagram* – Obsahuje název účtu bodu zájmu na sociální síti Instagram a nabízí možnost konstrukce odkazu do této sítě.
- *twitter* – Obsahuje název účtu bodu zájmu na sociální síti Twitter a nabízí možnost konstrukce odkazu do této sítě.

7.8.4 CovidData

Entita sloužící k udržování informací týkajících se počtu nakažených v dané oblasti a dalších hodnot. Počet entit tohoto typu je po celou dobu provozu aplikace stejný, jsou inicializovány ze souboru obsahující geografické určení obcí v České republice, jak je to popsáno v kapitole 6.2.1. Struktura entity je následující:

- *lau2_code* – Kód obce, který slouží k párování entity k záznamu o počtu nakažených z otevřené datové sady MZČR.
- *number_of_infected* - Počet nakažených v oblasti.
- *area* – Geografická oblast obce určená prostorovým datovým typem.
- *minimal_latitude, maximal_latitude, minimal_longitude, maximal_longitude* – Množina polí určujících nejmenší možný geografický čtyřúhelník, který obsáhne danou oblast obce.

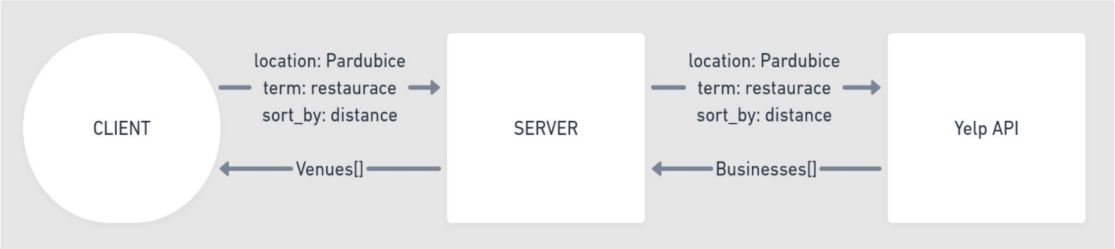
7.8.5 Rating

Rating je další entitou sloužící k uchování dodatečných informací k bodům zájmu, v tomto případě jde o informace týkající se hodnoty hodnocení a počtu hlasů. Obsahuje tedy atributy:

- *rating* – Desetinná hodnota od 1 do 5, včetně krajních hodnot. Vyjadřuje hodnocení místa.
- *rating_signals* – Celočíselná hodnota udávající počet hlasů, které vedly k určení daného hodnocení.

7.9 Synchronizace dat z Yelp API do aplikace

Synchronizace dat probíhá průběžně tím, jak ji uživatelé používají. Ve chvíli, kdy uživatel vyhledá body zájmu na základě nějakého filtru v klientské aplikaci tato klientská aplikace zašle požadavek na API serverové aplikace. Ta parametry tohoto požadavku patřičně upraví na formát, který požaduje Yelp API a na tuto API provede požadavek na vyhledávací koncový bod (detailní popis v kapitole 4.2.1). Po obdržení odpovědi, která v případě úspěchu požadavku obsahuje seznam bodů zájmu (*Businesses[]*). Seznam je zpracován a z každého bod zájmu je vytvořena entita *Venue*. Párování bodů zájmu s již existujícími entitami *Venue* probíhá na základě vlastnosti *yelpId*. Pokud v systému neexistuje *Venue* s obdrženým *yelpId*, je taková entita vytvořena. V podmínkách použití Yelp API stojí, že hodnoty z rozhraní získané by neměly být prezentovány uživatelům pokud jsou starší než 24 hodin. Aplikace tuto podmínku použití řeší vlastností *updatedAt* entity *Venue*, kde vlastnost obsahuje datum a čas poslední aktualizace dat. Pokud jsou starší 24 hodin, provede se aktualizace entity na základě dat získaných z koncového bodu Yelp API poskytujícího detail pro konkrétní bod zájmu.



Obrázek 14: Diagram komunikace s Yelp API

Zdroj: vlastní

7.10 Vývoj aplikace

7.10.1 Vývojové prostředí

Jako vývojové prostředí byl použit nástroj od společnosti JetBrains pro vývoj PHP. Autor má s nástroji od této společnosti velmi dobré zkušenosti a největší znalosti týkající se používání, právě proto byl PhpStorm jasná volba.

7.10.2 Lokální webový server

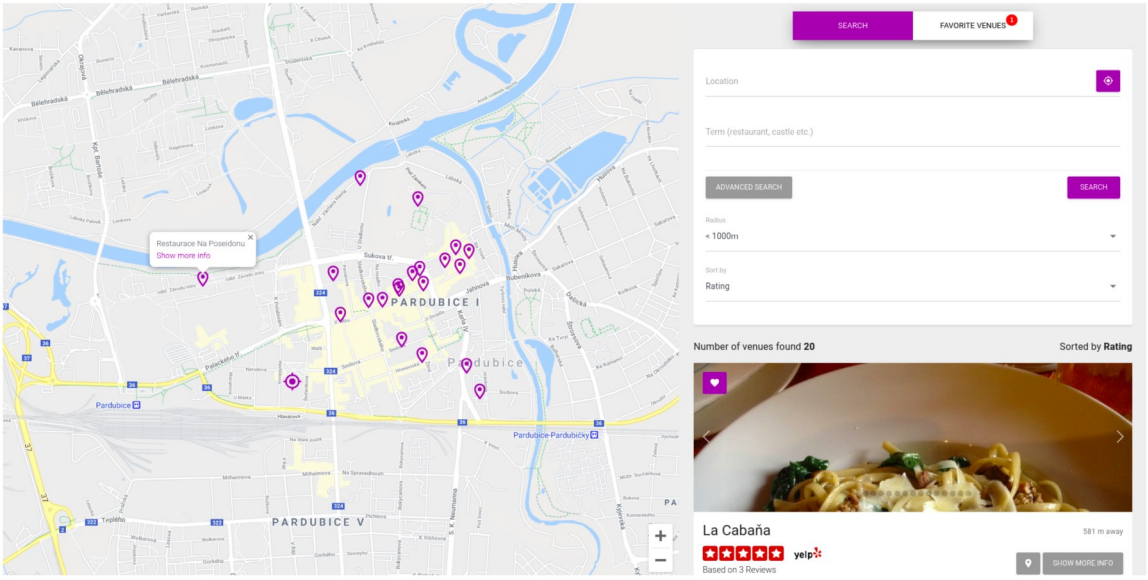
Během vývoje aplikace běžela lokálně na serveru SLWS (Symfony Local Web Server). Aplikace využívající framework Symfony samozřejmě mohou běžet na jakémkoliv serveru

s podporou PHP, nicméně SLWS je ideálním nástrojem pro práci na aplikacích založených na zmíněném frameworku z důvodu uzpůsobení přesné problematice.

SLWS je obsažen v balíčku *symfony*, který je dostupný pro platformy Linux, macOS a Windows. Tento balíček obsahuje, kromě již zmíněného webového serveru, řadu velmi užitečných nástrojů týkajících se správy aplikace (správa cache, logování, prostředí běhu), vytváření zdrojového kódu na základě šablon, správa databáze (cache, schéma, vytváření migrací) a dalších za pomocí příkazového řádku. [57]

7.11 Uživatelské rozhraní

Aplikace nabízí minimalistické uživatelské rozhraní s rozložením typickým pro aplikace pracující s vyhledáváním a zobrazováním entit na mapě. Aplikace je rozdělena na dva sloupce, přičemž v pravém zaujímá místo vyhledávací formulář s možností filtrace výsledků a případně samotné výsledky, levou stranu tvoří mapa zobrazující výsledky hledaní v prostoru.

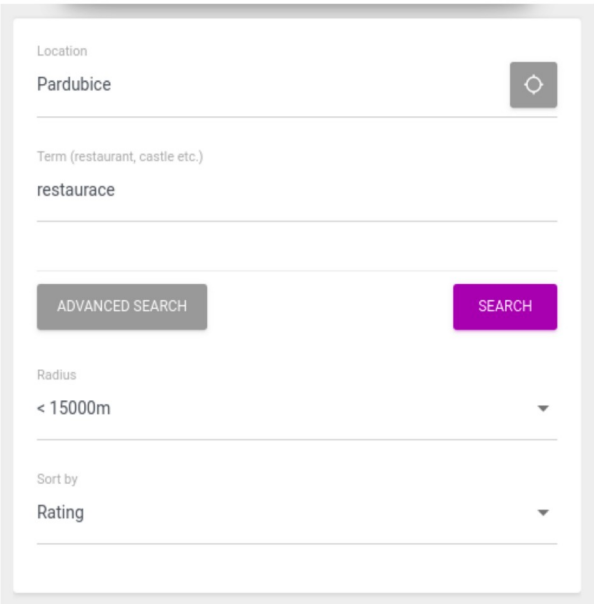


Obrázek 15: Uživatelské rozhraní aplikace

Zdroj: vlastní

7.11.1 Vyhledávání

Pro vyhledávání je nutné poskytnout lokaci hledání, které lze specifikovat názvem místa, například „Pardubice“, anebo na základě aktuální polohy. Vyhledávací formulář také poskytuje filtraci bodů zájmu na základě klíčového slova a maximální vzdálenosti od poskytnuté polohy. Kromě toho je možné výsledky seřadit dle několika základních metrik: nejlepší shody (v případě uvedení klíčového slova), hodnocení, počet recenzí a vzdálenost.

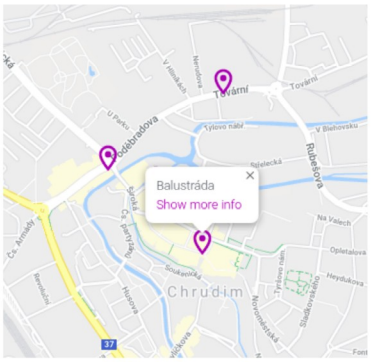


Obrázek 16: Vyhledávací formulář

Zdroj: vlastní

7.11.2 Zobrazení bodů zájmu na mapě

Zobrazení umístění bodů na mapě je realizováno „špendlíky“. Špendlíky jsou také jistým prostředkem interakce, po kliknutí na špendlík v mapě je zobrazeno informační blok se jménem daného místa a odkaz, který konkrétní místo zobrazí v seznamu v pravé části aplikace.

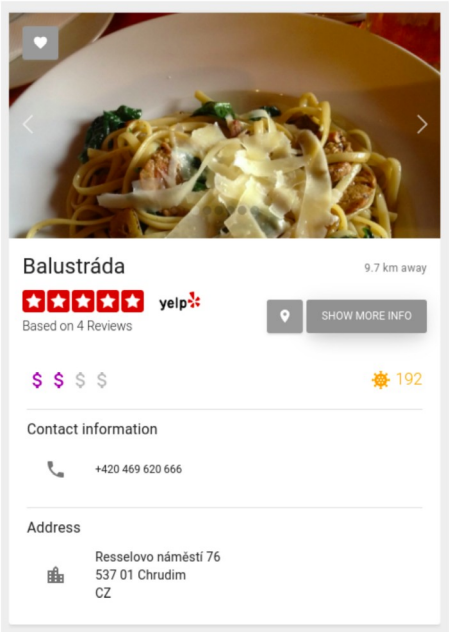


Obrázek 17: Body zájmu v mapě

Zdroj: vlastní

7.11.3 Kartička bodu zájmu

Po vyhledání se kromě zobrazení bodů na mapě načte seznam bodů zájmu do pravého sloupce aplikace. Každá položka seznamu je karta, kterou lze rozbalit tlačítkem „Show more info“ a tím si zobrazit více informací o daném bodu zájmu.



Obrázek 18: Karta bodu zájmu

Zdroj: vlastní

V horní části kartičky můžeme vidět fotogalerii místa, přičemž v úplném levém horním rohu tlačítko, sloužící k označení bodu zájmu jako oblíbeného. Tlačítko změní svou barvu v případě, že je bod zájmu mezi oblíbenými a začne sloužit k opačnému účelu, tedy odebrání místa z oblíbených.

Ve středu kartičky je název místa, vzdálenost, blok prvků pro zobrazení hodnocení Yelpu s informací o počtu hodnocení a loga Yelpu, které odkazuje na dané místo na webu Yelpu. Poslední tlačítko ve středu, mimo již zmiňovaného tlačítka na expanzi karty je tlačítko pro vystředění bodu zájmu, respektive jeho „markeru“ na mapě.

Dále je v kartičce vyznačena cenová kategorie a počet nakažených v obci, kde se bod zájmu nachází. Další informace se týkají kontaktu a adresy místa.

7.11.4 Oblíbené body zájmy

Jak bylo zmíněno, pomocí tlačítka v kartě lze bod zájmu přidat/odebrat do/z oblíbených. V horní část pravého sloupce aplikace je přepínač, který slouží k přepnutí pohledu. Existují

dva pohledy, vyhledávací („Search“) a zobrazující oblíbené body zájmu („Favorite venues“). U části přepínače na pravé straně je zobrazen počet oblíbených. Pohled „Favorite venues“ se mimo jiného obsahu jeho seznamu bodů zájmu liší oproti druhému pohledu absencí vyhledávacího formuláře.

Obrázek 19: Přepínač pohledu

Zdroj: vlastní

7.12 Nasazení aplikace

Na prostředí běhu aplikace je kladeno několik zásadních požadavků. Je nutné, aby takové prostředí splňovalo následující:

- Prostředí musí umožňovat instalaci a provozování webového serveru s podporou PHP.
- Je nutné mít možnost připojení na databázový server s podporou MySQL.
- Musí být zřízen bezpečný šifrovaný přístup k webové aplikaci s využitím TLS nebo SSL.

Veškeré zdrojové kódy aplikace jsou uloženy v soukromém repozitáři verzovacího systému git u poskytovatele GitHub. Pro běh aplikace je nutné tyto zdrojové kódy stáhnout do prostředí běhu. Průběh nasazení by mohl být realizován takto:

- Naklonování vzdáleného repozitáře.

```
$ git clone https://github.com/filipodvarka/bakalarska-prace.git
```

- Přechod do adresáře repozitáře.

```
$ cd bakalarska-prace
```

- Instalace knihoven a závislostí webové aplikace.

```
$ composer install
```

- Instalace knihoven a závislostí klientské aplikace a vytvoření jejího produkčního skriptu.

```
$ npm install && npm run build
```

- Následně se musí vytvořit konfigurační soubor *.env.local* a v něm definovat hodnoty třech proměnných nezbytných pro správný běh aplikace.

```
APP_ENV=prod
APP_SECRET=4836be2645be9bfc39c29c6bae0ec621
DATABASE_URL=mysql://user:pass@127.0.0.1:3306/db?serverVersion=8.0
```

- Pro vytvoření schématu databáze v databázi připojené pomocí URL v souboru *.env.local*. Schéma je vytvořeno na základě anotací atributů tříd entit, které se nacházejí v adresáři *src/entity*.

```
$ symfony console doctrine:schema:create
```

- Aby správně fungovalo zobrazování počtu nakažených je nutné provést dva příkazy. Nejprve inicializační příkaz, který vytvoří entity *CovidData*.

```
$ symfony console app:init-covid
```

- Spustit příkaz pro synchronizaci počtu nakažených, který do entit typu *CovidData* vyplní počty nakažených. Tento příkaz je doporučeno provádět každý den pro udržení aktuálnosti zobrazovaných dat.

```
$ symfony console app:sync-covid
```

ZÁVĚR

Cílem práce bylo provést analýzu dostupných API geosociálních sítí, které poskytují informace o turistických bodech zájmu, včetně jejich srovnání na základě několika metrik a následného praktického aplikačního využití. V neposlední řadě bylo úkolem inovativním způsobem poskytnou uživatelům informace týkající se informací o návštěvnosti turistických bodů zájmu. Tato problematika byla popsána a navrhnuta pouze v teoretické části bakalářské práce z důvodu vyřazení neočekávaného vyřazení zdroje informací z provozu, což znemožnilo její použití v aplikaci.

V práci byl zpracován úvod čtenáře do problematiky geosociálních sítí ve smyslu jejich účelu a důvodu zrodu. Následovala kapitola zabývající se analýzou geolokačních služeb a možností určování polohy s jejich pomocí. Bylo zpracován jednoduchý úvod k věci komunikace na internetu vztažený primárně na protokol HTTP a jeho využití v kombinaci s REST rozhraním. V kapitole zabývající se analýzou veřejných API geosociálních sítí byla vybrána dvě rozhraní pro hlubší analýzu týkající se jejich dostupnosti, možnostech použití a jejich srovnání. Následující kapitola byla věnována overtourismu včetně zpracování návrhu na jeho předcházení založeném na využití O2 Liberty API, tato služba byla bohužel vyřazena z provozu a nebyla tudíž implementován v aplikaci. Pro získání informací ohledně nakažených virem COVID-19 bylo využito datových sad poskytnutých Ministerstvem zdravotnictví České republiky, jež byly popsány včetně nutných kroků k umožnění jejich využití v aplikaci. Byly zmíněny i případové studie datových sad jiných poskytovatelů.

V druhé části práce byla popsána webová aplikace. Popsán byl její vývoj od technologií po použité nástroje či běhové prostředí. Aplikace využívá v její serverové části skriptovací jazyk PHP a na něm postavený framework Symfony, kde tyto technologie spolu s databází MySQL tvoří takzvaný „back-end“ aplikace. Na části „front-endu“ běží aplikace založená na reaktivním frameworku Vue.js, který, jak již název napovídá, využívá jazyk JavaScript.

Vyvinutá aplikace využívá data z geosociální sítě Yelp, data jsou obohacena o informace týkající se nakaženosti obyvatel virem COVID-19 v oblasti získané z otevřených datových sad Ministerstva zdravotnictví České republiky. V aplikaci je umožněno vyhledávat body zájmu se zohledněním polohy a případně i klíčového slova. Výsledky v podobě bodů zájmu lze řadit či označit jako oblíbené. Aplikace tyto výsledky prezentuje jejímu uživateli na mapě a ve formě seznamu.

POUŽITÁ LITERATURA

[1] ARMENATZOGLOU, Nikos a Dimitris PAPADIAS. Geo-Social Networks. In: Encyclopedia of Database Systems [online]. B.m.: Springer New York, 2018, s. 1620–1623. Dostupné z: doi:10.1007/978-1-4614-8265-9_80714

[2] LIBEN-NOWELL, D., J. NOVAK, R. KUMAR, P. RAGHAVAN a A. TOMKINS. Geographic routing in social networks. Proceedings of the National Academy of Sciences [online]. 2005, 102(33), 11623–11628. Dostupné z: doi:10.1073/pnas.0503018102

[3] YIN, Hongzhi, Bin CUI a Xiaofang ZHOU. Spatiotemporal Recommendation in Geo-Social Networks. In: Encyclopedia of Social Network Analysis and Mining [online]. B.m.: Springer New York, 2018, s. 2930–2948. Dostupné z: doi:10.1007/978-1-4939-7131-2_110177

[4] FRANKENFIELD, JAKE. Geolocation. *Investopedia* [online]. 25. 1. 2021, [cit. 2021-04-09]. Dostupné z: https://www.investopedia.com/terms/g/geolocation.asp

[5] GPS Accuracy. *GPS.gov* [online]. National Coordination Office for Space-Based Positioning, Navigation, and Timing, 22. 4. 2020 [cit. 2021-04-09]. Dostupné z: https://www.gps.gov/systems/gps/performance/accuracy/

[6] What is GPS?. *GPS.gov* [online]. National Coordination Office for Space-Based Positioning, Navigation, and Timing, 22. 2. 2021 [cit. 2021-04-09]. Dostupné z: https://www.gps.gov/systems/gps/

[7] GSM - metody určování polohy. *WIKI CR: informační databáze* [online]. UHK, 28. 1. 2016 [cit. 2021-04-09]. Dostupné z: https://fim2.uhk.cz/wikicr/web/index.php/home/9-ict/244-2016-01-28-10-16-20

[8] LESCONNEC, NICOLAS. The power of Wifi geolocation. *Nicolas Lesconnec* [online]. Medium, 15. 4. 2019 [cit. 2021-04-09]. Dostupné z: https://medium.com/@nlesconnec/the-power-of-wifi-geolocation-70d2494b066d

[9] What is Geotagging?. *Samsung* [online]. Samsung, 21. 3. 2018 [cit. 2021-04-09]. Dostupné z: https://www.samsung.com/in/support/mobile-devices/what-is-geotagging/

[10] What is GIS?. *Esri* [online]. Arcdata Praha, 21. 7. 2018 [cit. 2021-04-09]. Dostupné z: https://www.esri.com/en-us/what-is-gis/overview

[11] BEZPALEC, PAVEL. *Nové trendy v elektronických komunikacích: Lokalizace a navigace*. [online]. Publi, 5. 3. 2020 [cit. 2021-04-09]. Dostupné z: https://publi.cz/books/231/03.html

[12] An overview of HTTP. *MDN Web Docs*. [online]. Mozilla, 2021 [cit. 2021-04-09]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview

[13] HTTP request methods. *MDN Web Docs*. [online]. Mozilla, 2021 [cit. 2021-04-09]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods

[14] Safe. *MDN Web Docs*. [online]. Mozilla, 2021 [cit. 2021-04-09]. Dostupné z: https://developer.mozilla.org/en-US/docs/Glossary/Safe

[15] Idempotent. *MDN Web Docs*. [online]. Mozilla, 2021 [cit. 2021-04-09]. Dostupné z: https://developer.mozilla.org/en-US/docs/Glossary/Idempotent

[16] Cacheable. *MDN Web Docs*. [online]. Mozilla, 2021 [cit. 2021-04-09]. Dostupné z: https://developer.mozilla.org/en-US/docs/Glossary/Cacheable

[17] Headers. *MDN Web Docs*. [online]. Mozilla, 2021 [cit. 2021-04-09]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers

[18] Transmission Control Protocol. *IETF Tools*. [online]. IETF, 1981 [cit. 2021-04-09]. Dostupné z: https://tools.ietf.org/html/rfc793

[19] Web Services Architecture. *W3C Working Group Note*. [online]. W3C, 11. 2. 2004 [cit. 2021-04-09]. Dostupné z: https://www.w3.org/TR/2004/NOTE-ws-arch-20040211/#relwwwrest

[20] FIELDING, Roy T. Architectural styles and the design of network-based software architectures. Irvine: University of California, Irvine, 2000.

[21] REST APIs must be hypertext-driven. *Untangled: musings of Roy T. Fielding*. [online]. Roy T. Fielding, 20. 10. 2021 [cit. 2021-04-11]. Dostupné z: https://roy.gbiv.com/untangled/2008/rest-apis-must-be-hypertext-driven

[22] Foursquare. [online]. Foursquare: ©2021 [cit. 2021-04-07]. Dostupné z: https://foursquare.com/

[23] Media Rating Council. [online]. Media Rating Council [cit. 2021-04-07]. Dostupné z: http://mediaratingcouncil.org/

[24] Venue Search. *Foursquare: /developers*. [online]. Foursquare: ©2021 [cit. 2021-04-07]. Dostupné z: https://developer.foursquare.com/docs/api-reference/venues/search/

[25] Meaning of overtourism in English. *Cambridge Dictionary*. [online]. Cambridge University Press: ©2021 [cit. 2021-04-07]. Dostupné z: https://dictionary.cambridge.org/dictionary/english/overtourism

[26] Barcelona tourism activity report. [online]. Observatori del Turisme a Barcelona: 2017 [cit. 2021-04-07]. Dostupné z: https://web.archive.org/web/20190424013436/https://ajuntament.barcelona.cat/turisme/sites/default/files/1_turisme_estadistiques_2017_caps1_0.pdf

[27] Estadística de la Població de Barcelona: Resum de resultats. [online]. Ajuntament de Barcelona: 1. 1. 2017 [cit. 2021-04-07]. Dostupné z: https://ajuntament.barcelona.cat/premsa/wp-content/uploads/2017/07/Resum-de-resultats_Poblaci%C3%B32017.pdf

[28] Barcelona, a City at the frontlines of Overtourism. *THE HEARTLANDER OVERSEAS: Bumbling Around the World* [online]. Wordpress: 2. 11. 2018 [cit. 2021-04-07]. Dostupné z: https://heartlanderoverseas.wordpress.com/2018/11/02/barcelona-a-city-at-the-frontlines-of-overtourism/

[29] Overtourism: a growing global problem. *The Conversation: Academic rigour, journalistic flair* [online]. The Conversation: 18. 7. 2018 [cit. 2021-04-07]. Dostupné z: https://theconversation.com/overtourism-a-growing-global-problem-100029

[30] O_2 Developer Portal. [online]. O2 Czech Republic [cit. 2021-04-01]. Dostupné z: https://developer.o2.cz/

[31] GeoJSON. [online]. GeoJSON [cit. 2021-04-01]. Dostupné z: https://geojson.org/

[32] Spatial Data Types. *MySQL* [online]. Oracle Corporation: ©2021 [cit. 2021-04-07]. Dostupné z: https://dev.mysql.com/doc/refman/8.0/en/spatial-type-overview.html

[33] Co jsou otevřená data?. *Otevřená data* [online]. Ministerstvo vnitra: 3. 6. 2020 [cit. 2021-04-11]. Dostupné z: https://opendata.gov.cz/informace:start

[34] ČESKO. § 3 odst. 11 zákona č. 106/1999 Sb., o svobodném přístupu k informacím. In: *Zákony pro lidi.cz* [online]. © AION CS 2010-2021 [cit. 11. 4. 2021]. Dostupné z: https://www.zakonyprolidi.cz/cs/1999-106#p3-11

[35] Use Cases. *European Data Portal* [online]. European Data Portal: 16. 4. 2021 [cit. 2021-04-12]. Dostupné z: https://www.europeandataportal.eu/en/impact-studies/use-cases

[36] OpenTrees.org. *European Data Portal* [online]. European Data Portal [cit. 2021-04-12]. Dostupné z: https://www.europeandataportal.eu/sites/default/files/use-cases/global_-_opentrees.pdf

[37] Betterplace. *European Data Portal* [online]. European Data Portal [cit. 2021-04-12]. Dostupné z: https://www.europeandataportal.eu/sites/default/files/use-cases/spain_-_betterplace.pdf

[38] What Is a Database?. *Oracle* [online]. Oracle: ©2021 [cit. 2021-04-17]. Dostupné z: https://www.oracle.com/database/what-is-database/

[39] What is a web server?. *MDN Web Docs*. [online]. Mozilla, 2021 [cit. 2021-04-06]. Dostupné z: https://developer.mozilla.org/en-US/docs/Learn/Common_questions/What_is_a_web_server

[40] Fast Facts. *Yelp Newsroom*. [online]. Yelp ©2021 [cit. 2021-04-06]. Dostupné z: https://www.yelp-press.com/company/fast-facts/default.aspx 7. 4. 2021

[41] Yelp developers. *Yelp Developers*. [online]. Yelp ©2021 [cit. 2021-04-07]. Dostupné z: https://www.yelp.cz/developers?country=US

[42] Endpoint documentation. *Yelp Fusion*. [online]. Yelp ©2021 [cit. 2021-04-07]. Dostupné z: https://www.yelp.cz/developers/documentation/v3

[43] What is PHP?. *PHP*. [online]. The PHP Group ©2021 [cit. 2021-04-18]. Dostupné z: https://www.php.net/manual/en/intro-whatis.php

[44] KYSILKA, Pavel. Linuxsoft.cz [online]. Kysilka 2003-2017 [cit. 2021-04-18]. PHP (1) - Historie a budoucnost. Dostupné z: http://www.linuxsoft.cz/article.php?id_article=171. ISSN 1801-3805.

[45] PHP RFC: Saner string to number comparisons. *PHP*. [online]. The PHP Group ©2021, 26. 2. 2019 [cit. 2021-04-18]. Dostupné z: https://wiki.php.net/rfc/string_to_number_comparison

[46] Usage statistics of server-side programming languages for websites. W3Techs [online]. Q-Success ©2021 [cit. 2021-04-18]. Dostupné z: https://w3techs.com/technologies/overview/programming_language

[47] What is Symfony. *Symfony* [online]. Symfony [cit. 2021-04-18]. Dostupné z: https://symfony.com/what-is-symfony

[48] G., Damantas. What is HTML? The Basics of Hypertext Markup Language Explained. *Hostinger turorials* [online]. Hostinger.com ©2021, 25. 11. 2019 [cit. 2021-04-14]. Dostupné z: https://www.hostinger.com/tutorials/what-is-html

[49] HTML & CSS. *World Wide Web Consortium (W3C)* [online]. W3C, 2005 [cit. 2021-04-14]. Dostupné z: https://www.w3.org/standards/webdesign/htmlcss

[50] JANOVSKÝ, Dušan. CSS styly - úvod. Jak psát web [online]. Janovský, 2005 [cit. 2021-04-18]. ISSN 1801-0458. Dostupné z: https://www.jakpsatweb.cz/css/css-uvod.html

[51] JANOVSKÝ, Dušan. Přehled vlastností CSS. Jak psát web [online]. Janovský, 2005 [cit. 2021-04-18]. ISSN 1801-0458. Dostupné z: https://www.jakpsatweb.cz/css/css-vlastnosti-hodnoty-prehled.html

[52] JANOVSKÝ, Dušan. Úvod do JavaScriptu. Jak psát web [online]. Janovský, 2005 [cit. 2021-04-18]. ISSN 1801-0458. Dostupné z: https://www.jakpsatweb.cz/javascript/javascript-uvod.html

[53] Introduction. *Vue.js* [online]. Even You ©2021 [cit. 2021-04-14]. Dostupné z: https://vuejs.org/v2/guide/

[54] SALEH, Hazem. MVVM architecture, ViewModel and LiveData (Part 1) [online]. 31. 5. 2017 [cit. 2021-04-14]. Dostupné z: https://proandroiddev.com/mvvm-architecture-viewmodel-and-livedata-part-1-604f50cda1

[55] Promise. *MDN Web Docs*. [online]. Mozilla, 2021 [cit. 2021-04-15]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Promise

[56] Introduction. *Material design*. [online]. Google [cit. 2021-04-15]. Dostupné z: https://material.io/design/introduction

[57] Symfony Documentation. *Symfony* [online]. Symfony [cit. 2021-04-07]. Dostupné z: https://symfony.com/doc/current/index.html

[58] Client-server architecture. *Britannica, The Editors of Encyclopaedia* [online]. Encyclopedia Britannica, 23. 11. 2015 [cit. 2021-04-06]. Dostupné z: https://www.britannica.com/technology/client-server-architecture.

PŘÍLOHY

[1] Kompletní textová část bakalářské práce ve formátu PDF

[2] Archiv se zdrojovým kódem webové aplikace