Katedra informatiky
Přírodovědecká fakulta
Univerzita Palackého v Olomouci

BAKALÁŘSKÁ PRÁCE

Otevřená databáze bodů zájmu

2021

Jakub Večeřa

Vedoucí práce: Mgr. Petr Krajča, Ph.D.

Studijní obor: Aplikovaná informatika, prezenční forma

Bibliografické údaje

Autor:	Jakub Večeřa
Název práce:	Otevřená databáze bodů zájmu
Typ práce:	bakalářská práce
Pracoviště:	Katedra informatiky, Přírodovědecká fakulta, Univerzita Palackého v Olomouci
Rok obhajoby:	2021
Studijní obor:	Aplikovaná informatika, prezenční forma
Vedoucí práce:	Mgr. Petr Krajča, Ph.D.
Počet stran:	29
Přílohy:	1 CD/DVD
Jazyk práce:	český

Bibliograhic info

Author:	Jakub Večeřa
Title:	Open Database of Points of Interest
Thesis type:	bachelor thesis
Department:	Department of Computer Science, Faculty of Science, Palacký University Olomouc
Year of defense:	2021
Study field:	Applied Computer Science, full-time form
Supervisor:	Mgr. Petr Krajča, Ph.D.
Page count:	29
Supplements:	1 CD/DVD
Thesis language:	Czech

Anotace

Cílem bakalářské práce bylo vyvinout otevřenou databázi bodů zájmu (POI) pomocí webových technologií s plnohodnotným využitím pro mobilní zařízení. Aplikace umožňuje vytvářet vrstvy a mapy, ve kterých lze evidovat body zájmu a prezentovat je vhodnou vizuální formou. U jednotlivých bodů zájmu je možné evidovat doplňující informace, se kterými lze dále pracovat.

Synopsis

The aim of the bachelor thesis was to develop an open database of points of interest (POI) using web technologies with full use for mobile devices. The application allows creation of individual layers and maps in which you can record points of interest and present them in a suitable visual form. For individual points of interest, it is possible to store additional information and allow further operations.

Klíčová slova: mapová aplikace; webová aplikace; POI; PostGIS; Laravel; API; Leaflet; Vue.js; Nuxt.js; PWA;

Keywords: map application; web application; POI; PostGIS; Laravel; API; Leaflet; Vue.js; Nuxt.js; PWA;

Chtěl bych poděkovat Mgr. Petru Krajčovi Ph.D, rodině a blízkým přátelům.

Místopřísežně prohlašuji, že jsem celou práci včetně příloh vypracoval/a samostatně a za použití pouze zdrojů citovaných v textu práce a uvedených v seznamu literatury.

datum odevzdání práce podpis autora

Obsah

Seznam obrázků

Seznam tabulek

1 Úvod

Jako téma své bakalářské práce jsem zvolil otevřenou databázi bodů zájmu (POI) s možností vytváření vlastních tematických map. Tuto práci jsem vypracoval pod vedením Mgr. Petra Krajči Ph.D.

Vytvořená aplikace umožňuje spolupracovat na vzniku nových map, do kterých lze zaznačit libovolné body zájmu. Body zájmu mohou reprezentovat například navštívená místa, výskyt živočichů či rostlin nebo společenské a sportovní aktivity. Další z vlastností aplikace je archiv změn, ve kterém jsou zaznamenány provedené úpravy. Aplikace je postavena pomocí webových technologií a zároveň umožňuje plnohodnotné využití na mobilních zařízeních pomocí technologie PWA (více popsáno v kapitole 3.4).

Hlavním informačním zdrojem pro udržení prostorových struktur je *PostGIS in Action*[1], který podrobně popisuje funkcionalitu a práci s prostorovými objekty v prostorovém rozšíření databáze PostgreSQL (více popsáno v kapitole 3.1). Pro návrh backendové aplikace a uchopení vývojového frameworku[1] Laravel (viz kapitola 3.3) je vycházeno z jeho dokumentace[2]. Náhled na výstavbu API[3] poskytnul zdroj *Designing Web APIs[3]* a pro práci s JavaScriptovým frameworkem *Vue.js* (viz kapitola 3.5) slouží zdroj *Vue.js: Up and Running*[4], který se hlavně zabývá jeho využitím v praxi.

[1]Framework je softwarová struktura, která slouží jako podpora při programování, vývoji a organizaci jiných softwarových projektů. Může obsahovat podpůrné programy, knihovny API, podporu pro návrhové vzory nebo doporučené postupy při vývoji.[2]

[2]Dokumentace Laravel je přístupná na adrese https://laravel.com/docs/7.x

[3]Application programming interface (API) je rozhraní, kterým se program prezentuje. [3]

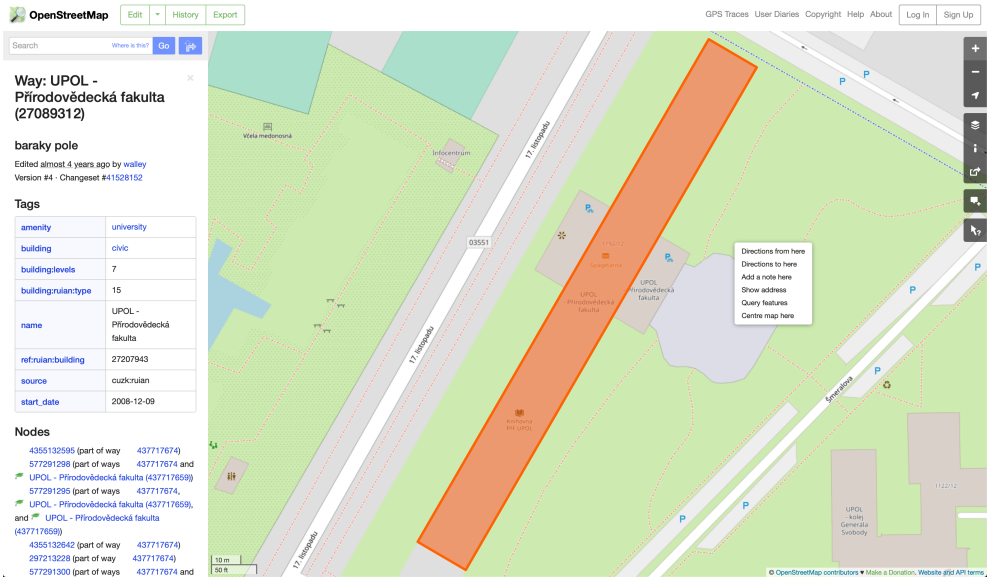
2 Mapové aplikace

U mapových aplikací se jednotlivé části map skládají z obrázků ve formě dlaždicových vrstev. Tato aplikace využívá JavaScriptovou knihovnu Leaflet.js (více v kapitole 3.7), která se o uspořádání obrázků stará automaticky, stačí pouze poskytnout adresu a z té se obrázky načítají.

2.1 OpenStreetMap

OpenStreetMap (OSM) je open-source editovatelná mapa světa s hlavním zaměřením na geografická data. Umožňuje uživateli přidávat různé geometrické útvary do mapy a spravovat jejich vlastnosti pomocí značek.

Základními prvky v OpenStreetMap jsou uzly, cesty a relace. Uzly mohou být spojeny a tvořit cesty, které se dělí na otevřené a uzavřené. Relace se skládají z jednotlivých uzlů, cest a dalších relací. S každým prvkem mohou být asociovány značky tvořené dvojicí klíč a hodnota a ty pak slouží pro ukládání vlastností. Klíče lze chápat jako kategorie, které seskupují několik značek s odlišnými hodnotami k sobě. Po vybrání prvku se zobrazí tabulka s informacemi, jak je znázorněno v obrázku 1.

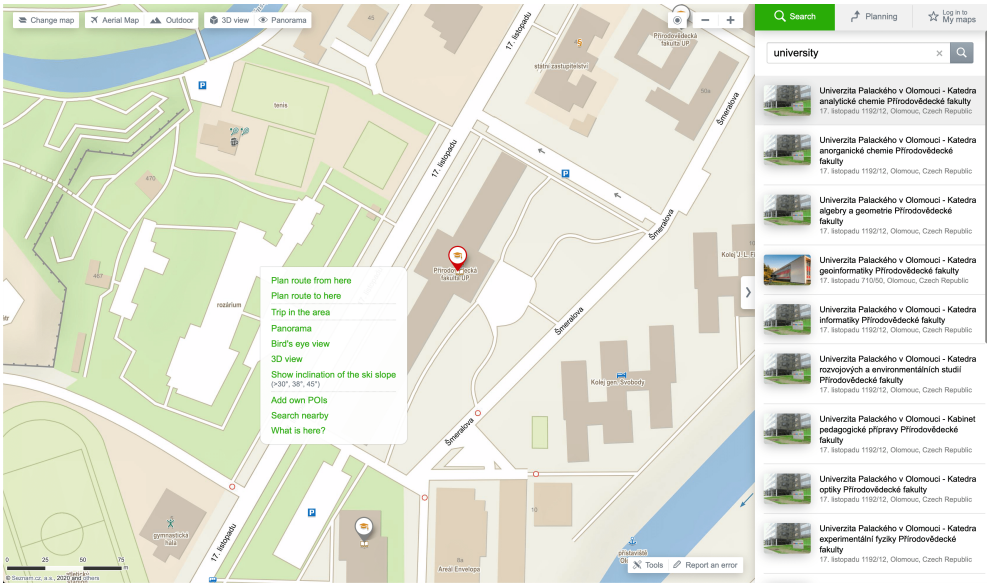


Obrázek 1: Ukázka z aplikace OpenStreetMap[5]

2.2 Mapy.cz

Mapy.cz je mapová aplikace pro webové stránky, iOS a Android od společnosti Seznam.cz. V zahraničí jsou Mapy.cz také známé jako Windy Maps. Mapy.cz dále nabízejí své API pro geokódovací služby a tvorbu mapových aplikací. Náhledu na aplikaci koresponduje obrázek 2.

Aplikace umožňuje uživateli přidávat body zájmu a jejich správu v nabídce „Moje mapy“ a zároveň nabízí možnost vyhledávání pomocí klíčových slov.



Obrázek 2: Ukázka z aplikace Mapy.cz[6]

3 Použité technologie

3.1 PostGIS

PostGIS je open-source rozšíření PostgreSQL, které umožňuje práci s prostorovou databází. Ta definuje speciální datové typy pro geometrické objekty, používá se jako úložný kontejner pro prostorová data a umožňuje provádět výpočty vzdáleností[1]. Díky tomu lze spravovat prostorová data jako je bod, lomená čára a polygon.

Při instalaci PostGIS se automaticky vytvoří tabulka *geometry_columns* pro uložení metadat. Sloupec *coord_dimension* reprezentuje rozměr s přípustnými hodnotami 2, 3 a 4. Sloupec *SRID*[4] odkazuje do tabulky *spatial_ref_sys* a ta je rovněž vytvořena automaticky.[1]

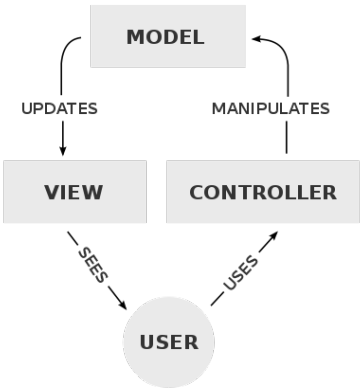
3.2 Architektura MVC

Model-View-Controller (MVC) je architektura pro vývoj aplikací, které odpovídá diagram znázorněný v obrázku 3. Tato architektura rozděluje aplikace na:

- model obsahující data a logiku se kterými se pracuje

[4]Spatial reference identifier (SRID) určuje prostorový referenční systém pro souřadnice[1]

- pohled, který se stará o převod dat z modelu do vhodné podoby pro uživatele
- řadič starající se o aktualizaci modelu při nějaké události



Obrázek 3: Diagram architektury MVC[7]

3.3 Laravel framework

Laravel je open-source PHP framework pro vývoj webových aplikací s MVC architekturou (viz kapitola 3.2). Laravel je postavený na frameworku *Symphony* a skládá se z několika částí jako je například *Eloquent ORM* a *Composer*, který má na starost správu závislostí.

Eloquent ORM je základní řešení pro objektově relační zobrazení[5]. Tabulkám odpovídají dané modely a řádky tabulek jsou instance třídy s jednotlivými sloupci jako jejich vlastnostmi.

Pro vytváření jednotlivých tabulek se používají *migrace*, které se automaticky verzují pro snadné řešení změn. O naplňování tabulek se starají třídy nazývané *seedery*, ty jsou užitečné pro nastavení výchozího stavu databáze.

Pro převádění dat z modelů na JSON[6] poskytuje Laravel formátovací třídy nazývané *Resources*, ve kterých lze definovat strukturu dat a lze je navíc doplnit různými metadaty nebo stránkováním. Snadnější práci s daty v podobě objektů umožňují kolekce (collections) pomocí předem definovaných metod.

3.4 PWA

Progresivní webová aplikace (Progressive Web Application) je webová aplikace, která vypadá a načítá se jako běžná webová stránka. Navíc ale nabízí funkce

[5]Objektově relační zobrazení (ORM) je programovací technika, která zajišťuje automatickou konverzi dat mezi relační databází a objekty.[8]

[6]JavaScriptový objektový zápis (JavaScript Object Notation) je způsob zápisu dat (datový formát) nezávislý na počítačové platformě určený pro přenos dat.[9]

běžně dostupné pouze nativním aplikacím, např. práce nezávislá na připojení, push notifikace nebo přístup k hardwaru zařízení.[10]

3.5 Vue.js framework

Vue.js je open-source JavaScriptový framework sloužící k vytváření uživatelského rozhraní a SPA[7]. Umožňuje kód rozdělovat na komponenty, které se chovají jako nové HTML elementy a ty lze doplnit o vlastní logiku. Pomocí správce balíčku můžeme Vue doplnit řadou funkcionality.

Vue.js je reaktivní, což má za následek automatickou aktualizaci dat na všech místech, kde se používají. Data lze předávat z předka na potomka pomocí *props*. Pro předání dat z potomka na předka se musí poslat zpráva, na kterou se u předka naslouchá. Další možností je použít balíček *Vuex*, který se stará o sdílení dat napříč celou aplikací. Pravidla *Vuex* zaručí, že se sdílená data budou měnit pouze prostřednictvím předem definovaných funkcí.

3.6 Nuxt.js framework

Nuxt.js je open-source JavaScriptový framework rozšiřující Vue.js a využívá řadu balíčků, které jsou připraveny přímo k použití, např. *Vue Router* zodpovídající za směrování[8] nebo zmíněný *Vuex*.

Framework možňuje skriptování na straně serveru (server side rendering) neboli SSR. Jinými slovy je stránka vykreslována na straně serveru a až následně poslána klientovi. To znamená, že se klientovi pošle DOM[9], ve kterém je již vykreslená aplikace včetně komponent s vyplněnými daty, která jsou pro webové vyhledávače viditelná. Tahle možnost lze vypnout jak pro jednotlivé části tak i pro celou aplikaci. Jedna z dalších možností je generování statických stránek, které na serveru nemusí běžet.

3.7 Leaflet

Leaflet je open-source JavaScriptová knihovna pro tvorbu mapových aplikací. Umožňuje zobrazovat rastrové a vektorové objekty. Rastry také tvoří jednotlivé dlaždicové vrstvy mapy, které jsou automaticky skládány a aktualizovány podle přiblížení. Na ty lze nanášet jednotlivé objekty. Dále existuje velká řada balíčků pro rozšíření funkcionality. Jedním z rozšiřujících balíčků je například *Leaflet Draw*, který umožňuje kreslení vektorů.

[7]SPA neboli jednostránková aplikace (Single-Page Application) je aplikace s více stránkami, ke kterým lze přistupovat bez vytvoření nového dotazu.[4]

[8]Směrování (též routování) je akt, při kterém se vezme cesta např. */users/12345/posts/* a rozhodnutí, co by se mělo na stránce zobrazit. [4]

[9]Document Object Model reprezentuje HTML jako strom a každou jeho část jako uzel.

4 Programátorská dokumentace

4.1 Backend aplikace

Backend aplikace, dále označovaná také jako backend, je napsána ve frameworku Laravel a její hlavní funkcí je správa dat. Data jsou uložena v databázi, která obsahuje jednotlivé tabulky a ty je možné vidět v seznamu tabulek 1.

Tabulka 1: Seznam tabulek

Tabulka	Popis
`tile_layers`	tabulka pro dlaždicové vrstvy, obsahuje sloupce pro název a zdroj
`maps`	tabulka pro mapy, obsahuje sloupce pro název a cizí klíč na tabulku *tile_layers*
`layers`	tabulka pro vrstvy, obsahuje sloupce pro název, barvu a geometrický typ prostorového objektu
`features`	tabulka pro prostorové objekty, obsahuje prostorový sloupec pro reprezentaci souřadnic a sloupec pro cizí klíč na tabulku *layers*
`fields`	tabulka políček, která jsou provázaná s vrstvou pomocí cizího klíče na tabulku *layers*, dále obsahuje sloupce pro název, typ vstupu, výchozí hodnotu, komentář a jiné
`properties`	tabulka hodnot políček pro daný prostorový objekt, která obsahuje sloupce pro cizí klíče na tabulky *feature* a *fields* a sloupce pro reprezentaci hodnoty daného políčka
`layer_map`	je určena pro provázání map a vrstev, obsahuje sloupce pro cizí klíče na tabulky *maps* a *layers* a sloupce pro barvu a cizí klíč na tabulku *fields* pro určení výchozího zobrazovacího políčka
`tags`	tabulka pro značky obsahující sloupce pro klíč a hodnotu
`taggables`	tabulka obsahující sloupec pro cizí klíč na tabulku *tags* a sloupce *taggable_type* a *taggable_id* pro provázání značky pomocí určení modelu a jeho id
`media`	tabulka pro obrázky k ikonkám
`activity_log`	tabulka pro zachování změn prostorových objektů, kterých se využívá v archívu

Aplikace pracuje s daty z databáze pomocí modelů (viz kapitola 3.2) a jak bylo zmíněno v kapitole 3.3, ke každému modelu koresponduje příslušná tabulka. Eloquent se dále postará o automatickou konverzi dat. Přehled modelů uvádí tabulka 2. Model Feature navíc implementuje funkcionalitu *soft delete*, která

zajišťuje, že se při mazání prvek neodebere z databáze, ale nastaví se jeho sloupec *deleted_at* na dané datum.

Tabulka 2: Seznam modelů

Model	Popis
`Map`	reprezentuje mapu, ke které lze navázat libovolné vrstvy
`Layer`	reprezentuje vrstvu v mapě, u které je nutné vybrat, jaký typ prostorového objektu bude obsahovat
`Field`	reprezentuje políčko ve vrstvě, u kterého je možné zvolit jeho název, typ vstupu, zda je povinný, jeho výchozí hodnotu a komentář, který se zobrazí jako nápověda
`Feature`	reprezentuje prostorový objekt ve vrstvě s geometrickým útvarem typu bod, polygon nebo lomená čára
`Property`	informace o prostorovém objektu
`Tag`	informace ve formě dvojice klíč a hodnota
`TileLayer`	reprezentuje zdroj pro dlaždicovou vrstvu mapy

O spojování jednotlivých tabulek se stará Eloquent, v modelu je však nutné definovat o jaký typ vztahu se bude jednat. Příklad řešení vztahů pro model *Map* je uveden ve zdrojovém kódu 1. Pokud by funkcionalita Eloquentu nedostačovala, lze zvolit jiný balíček, popřípadě je možné všechno spojování naprogramovat.

```
1  public function layers()
2  {
3     return $this->belongsToMany(Layer::class)
4        ->withPivot('color', 'display_field_id');
5  }
6
7  public function tileLayer()
8  {
9     return $this->belongsTo(TileLayer::class, 'tile_layer_id');
10 }
```

Zdrojový kód 1: Vztahy modelu Map řešené pomocí Eloquent

V modelu se nastavuje, jaké sloupce lze vyplnit pomocí proměnné *fillable*. Dále lze do modelu doplnit metody pro přeměnu získaných dat (accessors) a pro přeměnu ukládaných dat (mutators). Tím lze také vytvářet de facto nové proměnné, které nejsou v databázi a mohou sloupce kombinovat. Příklad pro přeměnu ukládaných dat je možné vidět ve zdrojovém kódu 2.

```
public function setNameAttribute($value)
{
    $this->attributes['name'] = $value;
    $this->attributes['slug'] = Str::slug($value);
}
```

Zdrojový kód 2: Přeměna ukládaných dat u modelu Field

4.1.1 Koncové body

Pro komunikaci s webovou aplikací obsahuje backend několik koncových bodů[10]. Při přístupu na koncový bod se volají metody, které jsou obsažené v *řadičích* (viz kapitola 3.2). Koncové body lze seskupovat a dodávat jim například prefix. Způsob jejich tvoření je možné vidět ve zdrojovém kódu 3.

```
Route::group(['prefix' => 'features'], function () {
    Route::get('/', 'FeatureController@index');
    Route::post('/', 'FeatureController@store');
    Route::get('/{feature}', 'FeatureController@show');
    Route::patch('/{feature}', 'FeatureController@update');
    Route::delete('/{feature}', 'FeatureController@destroy');
    Route::post('/{feature}/revert', 'FeatureController@revert');
});
```

Zdrojový kód 3: Definice koncových bodů pro prostorový objekt

Koncové body lze označit jako RESTful, to znamená, že zpřístupňují data jako zdroje a pomocí standardních metod HTTP reprezentují CRUD (Create, Read, Update a Delete) transakce[3], čímž umožňují spravovat příslušné tabulky.

4.2 Webová aplikace

Webová aplikace je napsána ve frameworku Nuxt.js využívající technologii SPA. Stránky jsou automaticky tvořeny ze souborů v adresáři „pages“. Při použití symbolu „_“ před názvem souboru se stránka nastaví jako dynamická, neboli stránka, která může měnit svůj obsah na základě různých vlastností. Framework dále umožňuje tvorbu *pluginů* pro spuštění části kódu před inicializací Vue instance. Zároveň jsou *pluginy* používány pro sdružení nějakého logického celku s možností jeho využití napříč celou aplikací.

Pro interakci s backendem prostřednictvím API se používají *async* funkce, které jsou funkce vracející příslib (promise). Příslib reprezentuje zatím nevyhodnocenou hodnotu, u které víme, že se nějak vyhodnotí. Při použití výrazu *await* v *async* funkci můžeme přerušit proces, dokud není příslib splněn.

[10]Koncový bod je adresa a část API, přes kterou lze s aplikací interagovat. Koncový bod je často označován jako endpoint.

4.2.1 Mapa

Interaktivní mapa je zprostředkována pomocí knihovny Leaflet.js, která je popsána v kapitole 3.7. Knihovna Leaflet zároveň s knihovnout Leaflet Draw jsou před inicializací aplikace načteny. Při načtení knihovny na straně klienta se do instance Vue připíší proměnné reprezentující mapu a jednotlivé prostorové objekty. Ukázku načtení knihovny Leaflet.js je možné vidět v kódu 4.

```
1 import * as L from 'leaflet'
2 import 'leaflet-draw'
3
4 export default (context, inject) => {
5    L.Map.addInitHook(function() {
6       inject('map', this)
7       inject('drawn', new L.FeatureGroup())
8    })
9 }
```

Zdrojový kód 4: Kód pro načtení knihovny Leaflet.js

Při přístupu na mapu z hlavní nabídky se nejdříve získají data o mapě z koncového bodu *maps/get*. Z dat se určí název mapy, nanesou se na mapu jednotlivé prostorové objekty, nastaví se výchozí dlaždicová vrstva a vycentruje se mapa na danou pozici. Ve zkráceném zdrojovém kódu 5 je možné vidět postup při načtení dat. Data jsou uložena ve Vuex a tak je k nim možné přistupovat z libovolné komponenty. Část kódu, který se používá na více místech a souvisí s knihovnou Leaflet.js je umístěn v pluginu leaflet.js.

```
1  async asyncData({ params, $axios }) {
2     const data = await $axios.$get(`maps/get/${params.slug}`)
3     return { data }
4  },
5
6  mounted() {
7     this.$store.commit('map/setData', this.data)
8     this.selectTileLayer(this.data.tileLayer)
9     this.addFeatures()
10    this.setMapPosition()
11 }
```

Zdrojový kód 5: Zkrácený zdrojový kód při načtení dat mapy

Při nanášení jednotlivých prostorových objektů se u každé nastaví její ikonka. Pokud má vrstva nastavený obrázek pro ikonku, je v ikonce zobrazen. Dále se v ikonce vyplní hodnota současného zobrazovacího políčka a nastaví se jeho barva. Veškerá práce s ikonkami je ucelená v pluginu icons.js.

4.2.2 Přepínání mezi prostorovými objekty

Přepínání mezi prostorovými objekty se provádí pomocí změny *id* daného objektu ve Vuex. Tato změna je provedena po kliknutí na vyznačený prostorový objekt na mapě, nebo pomocí odposlechu reagujícího na přidání prostorového objektu prostřednictvím balíčku Leaflet Draw. Zmíněné funkcionalitě odpovídá zdrojový kód 6, který ve skutečnosti obsahuje kroky navíc. Prostorový objekt se také automaticky přepíná při vyhledávání pomocí názvu nebo adresy. Po změně id se zobrazí komponenta *Sidebar*. Po přepnutí se pomocí id a metody *getLayer* z proměnné *drawn* získá výsledný prostorový objekt, který je reprezentován na mapě pomocí *layer*[11].

```
1  mounted() {
2     this.map.on('draw:created', this.handleDrawCreated)
3     this.drawn.on('click', this.handleItemsClick)
4  },
5
6  methods: {
7     handleDrawCreated(e) {
8        this.drawn.addLayer(e.layer)
9        this.$store.commit('setId', e.layer._leaflet_id)
10    },
11    handleItemsClick(e) {
12       this.$store.commit('setId', e.layer._leaflet_id)
13    }
14 }
```

Zdrojový kód 6: Zkrácená verze funkcionality pro změnu prostorového objektu

4.2.3 Komponenty

Webová aplikace se skládá z řady komponent, které jsou uvedeny v tabulce 3. Komponenty využívané aplikací se nacházejí v adresáři *components*, dají se využívat napříč celou aplikací a lze je různě zanořovat. Pomocí komponent lze vytvářet de facto nové elementy.

Kromě komponent vypsaných v tabulce je dále řada komponent v podsložce *Forms*, které představují jednotlivé formuláře. Ve formulářích se navíc pracuje s řadou komponent, které jsou v podsložce *FormWrapper*. Tyto komponenty tvoří abstrakci nad jednotlivými prvky formuláře jako je například *Input*, *Select* a zároveň přidávají prvky nové jako jsou *FormColor* nebo *FormOpeningHours*.

[11] Leaflet označuje „layer“ jako jakýkoliv objekt, který se pohybuje současně s mapou.[11]

Tabulka 3: Seznam komponent

Komponenta	Popis
`Activity`	obsahuje záznamy archívu a jejich funkcionalitu pro návrat
`LayerMenu`	seznam aktuálních vrstev, ve kterých je počet prostorových objektů dané vrstvy a tlačítko pro vytvoření nového prostorového objektu
`TileLayerMenu`	funkcionalita pro přepínání dlaždicových vrstev
`Toolbar`	panel nástrojů obsahující základní navigaci, přepínání dlaždicové vrstvy a seznam aktuálních vrstev
`Dropdown`	obaluje obsah, který bude zobrazen ve formě rozbalovací nabídky
`Modal`	obaluje obsah, který bude zobrazen ve formě modálu
`Searchbar`	obsahuje jednotlivé prvky vyhledávacího řádku a filtrace
`Sidebar`	obsahuje informace o vybraném prostorovém objektu

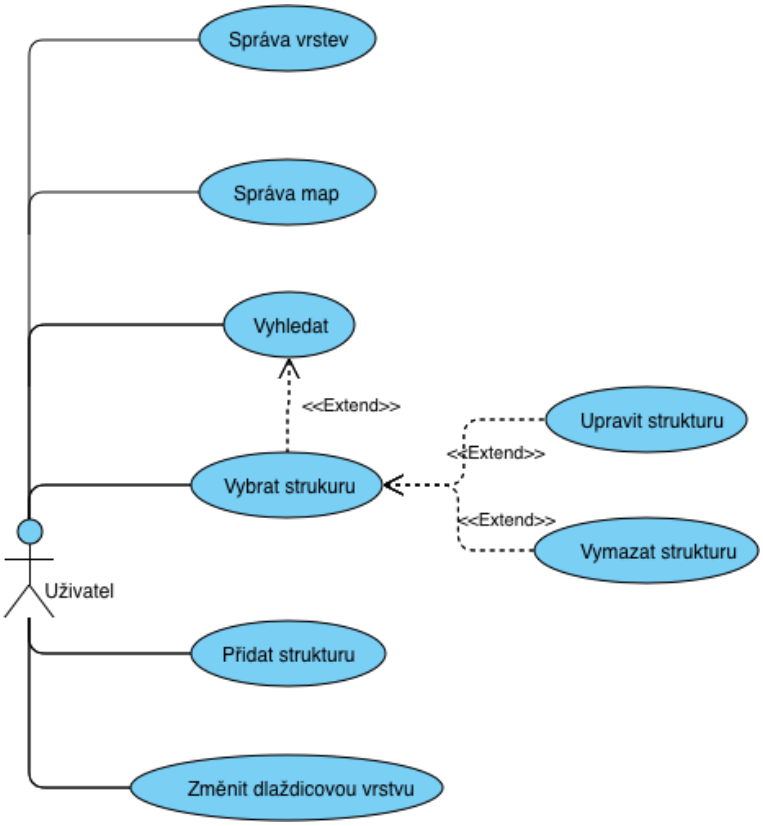
4.3 Mobilní aplikace

Mobilní aplikace je vytvořena pomocí technologií PWA (viz kapitola 3.4) a *PWA Builder*[12], které automaticky vygenerují aplikace pro jednotlivé platformy. Jednou z nevýhod tohoto řešení může být, že je nekonzistentní vzhledem k dané platformě.

[12]PWA Builder je open-source služba poskytovaná společností Microsoft, která umožňuje vytvářet PWA napříč různými platformami.

5 Uživatelská dokumentace

Uživatelská dokumentace slouží jako návod pro uživatele, jak s aplikací pracovat. Soustředí se na webovou verzi, ale zároveň má mnoho podobných prvků s verzí mobilní. Některé z těchto prvků jsou však umístěny na jiných částech obrazovky. Obrázek 4 pojednává o různých možnostech, které může uživatel v aplikaci využít.



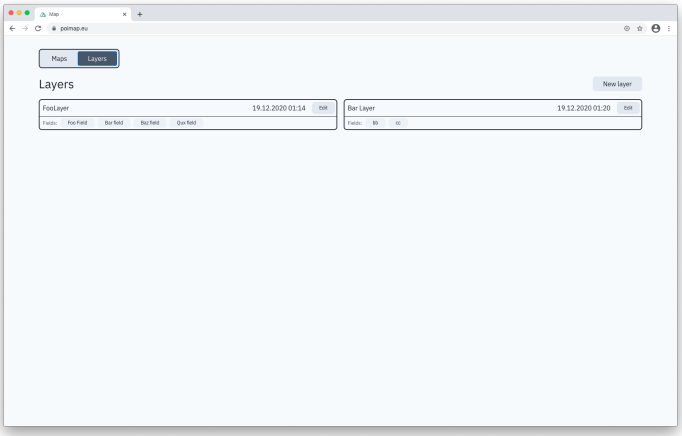
Obrázek 4: Případy užití pro aplikaci

5.1 Správa vrstev

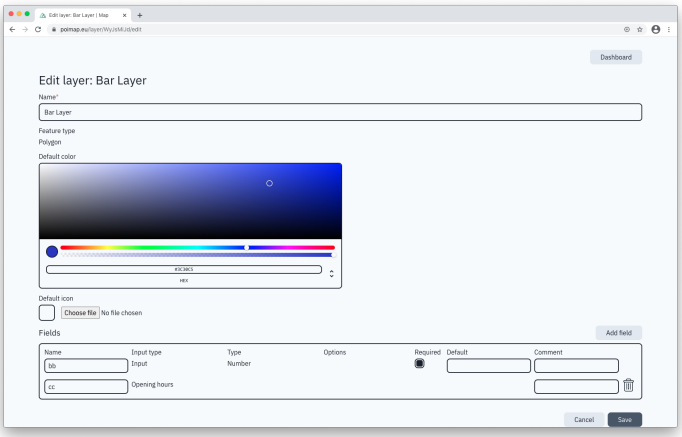
Vrstvy představují úrovně nanesené na mapu a obsahují dané prostorové objekty. Každá vrstva má svůj název, barvu, volitelnou ikonku a informaci o tom, jaký typ prostorového objektu obsahuje. Dále obsahuje seznam volitelných políček s různými vlastnostmi, které lze u každého prostorového objektu vyplnit.

Pro vytvoření vrstvy slouží tlačítko „Create layer“ a pro úpravu vrstvy tlačítko „Edit“. Při kliknutí na jedno z tlačítek se zobrazí formulář, ale pro úpravu vrstvy budou předem vyplněná políčka. Políčka reprezentují zmíněné vlastnosti vrstvy. Správa seznamu volitelných políček vrstvy je skrz tabulku „Fields“. U každého políčka vrstvy je možné určit jeho název, druh vstupu a jeho typ, popřípadě

jeho možnosti, pokud se jedná o druh vstupu typu „select“. Dále je možné určit, zda je políčko povinné, jeho výchozí hodnotu a komentář, který se zobrazí formou nápovědy (tooltip) u každého prostorového objektu.



Obrázek 5: Seznam vrstev

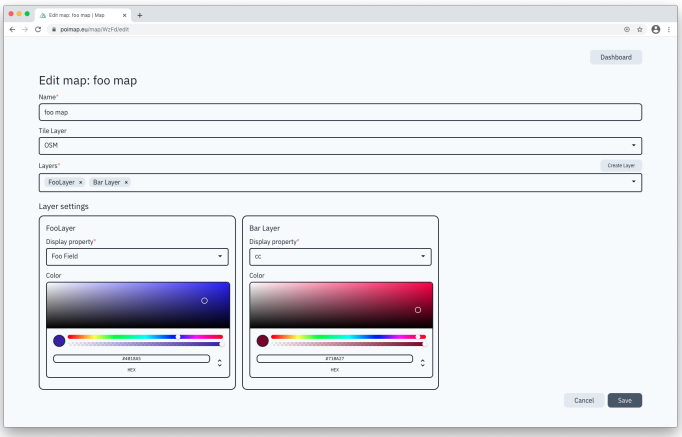


Obrázek 6: Úprava vrstvy

5.2 Správa map

Mapy mohou obsahovat libovolné množství vrstev, ve kterých jsou jednotlivé prostorové objekty uloženy. Každá mapa má svůj název, výchozí dlaždicovou vrstvu a seznam vrstev.

Novou mapu je možné vytvořit při kliknutí na tlačítko „Create map“ a lze upravit kliknutím na tlačítko „Edit“. Ve formuláři pro vytváření a úpravu mapy je mimo zmíněné vlastnosti mapy dále možné specifikovat ke každé vybrané vrstvě její barvu a výchozí zobrazovací políčko. Při zvolení výchozího zobrazovacího políčka se u dané vrstvy na mapě implicitně ukáže jeho hodnota.



Obrázek 7: Úprava mapy

5.3 Interakce s mapou

Při výběru ze seznamu map se zobrazí mapa obsahující dané vrstvy. Je možnost zobrazit také pouze samostatnou vrstvu. Obsahuje-li vrstva nějaké prostorové objekty, jsou naneseny na mapu.

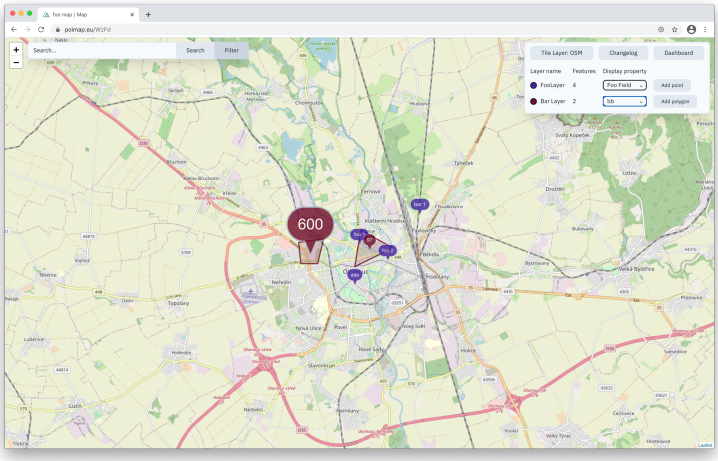
V levém rohu jsou tlačítka obsahující symboly „+“ a „-“, které slouží pro přibližování a oddalování. K přibližování a oddalování také slouží kolečko myši. Posouvání v mapě umožňuje levé kliknutí a posunutí myší.

Vedle tlačítek pro přibližování a oddalování se nachází vyhledávací řádek a tlačítko pro filtraci, které budou popsány více v kapitole 5.9.

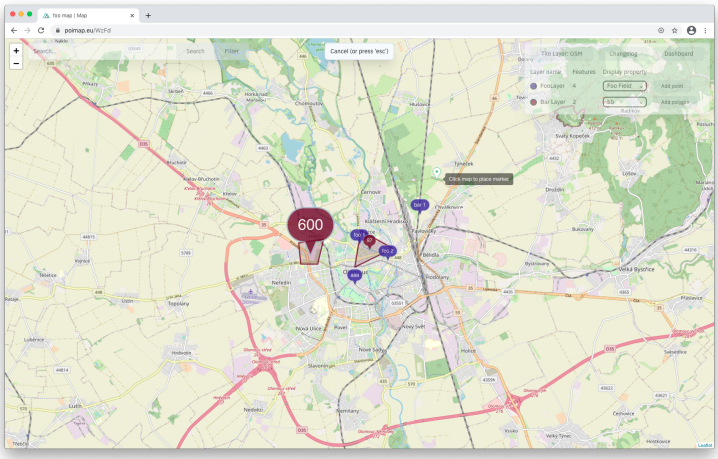
V pravém rohu se nachází menu obsahující tlačítka pro změnu dlaždicové vrstvy, otevření archívu mapy (viz kapitola 5.8) a návrat na domovskou stránku. Dále se v menu nachází seznam všech vrstev. U každé vrstvy je vypsaný název a počet prostorových objektů a je možné přepínat zobrazení políček. Při kliknutí na tlačítko „Add“ lze přidat prostorový objekt, více v kapitole 5.4.

5.4 Přidání prostorového objektu

Přidání nového prostorového objektu je možné prostřednictvím tlačítka umístěném v menu všech vrstev. Na základě typu prostorového objektu vrstvy se při kliknutí na tlačítko spustí režim editování mapy a automaticky se vybere daný geometrický útvar. Pro kreslení lomené čáry nebo polygonu je třeba postupně nanášet uzly pomocí levého tlačítka. Pro dokončení je nutné kliknout dvakrát na mapu a u polygonu lze také kliknout na počáteční uzel. U bodu stačí pouze kliknout na dané místo na mapě. Při kreslení lze také geometrický útvar vymazat pomocí tlačítka „Cancel“.



Obrázek 8: Úprava vrstvy



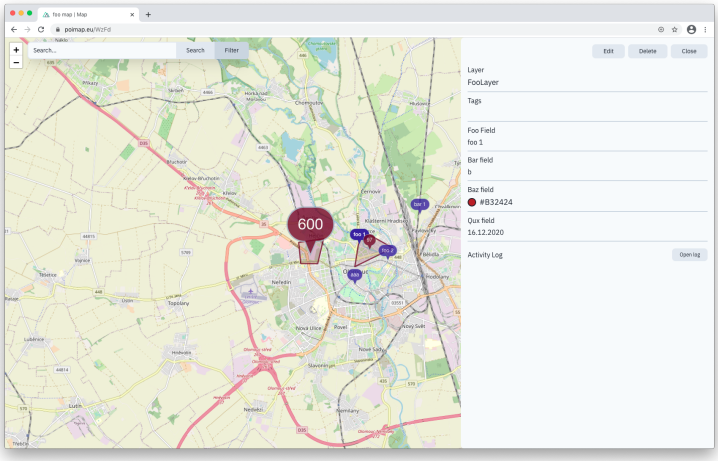
Obrázek 9: Přidání prostorového objektu

5.5 Ukládání prostorového objektu

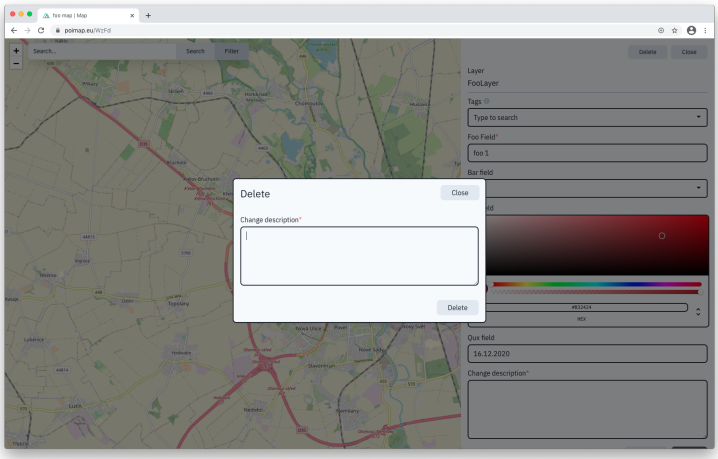
Po přidání nového prostorového objektu se zobrazí oblast pro doplnění tagů a vlastností. V prvním políčku lze vyhledávat jednotlivé tagy uložené v databázi popřípadě přidávat vlastní tagy pomocí klávesy „Enter“. Pro odeslání formuláře a uložení prostorového objektu slouží tlačítko „Save“.

5.6 Přehled prostorového objektu

Po kliknutí na prostorový objekt se otevře jeho přehled. Pro uzavření přehledu slouží tlačítko „Close“, pro úpravu prostorového objektu slouží tlačítko „Edit“ a pro vymazání tlačítko „Delete“. Při mazání se otevře modální okno s políčkem pro zápis důvodu mazání.



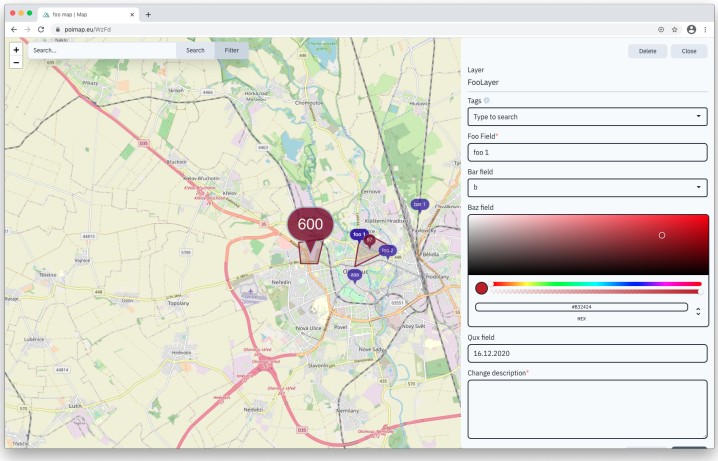
Obrázek 10: Přehled prostorového objektu



Obrázek 11: Mazání prostorového objektu

5.7 Úprava prostorového objektu

Po stisknutí tlačítka „Edit“ v přehledu prostorového objektu se přepne na režim úpravy. Ten vypadá stejně jako při ukládání s tím rozdílem, že jsou některá políčka předem vyplněná. Jednotlivá políčka lze upravit. Pro smazání tagu slouží tlačítko vedle jeho názvu. Pro uložení změn je nutné kliknout na tlačítko „Save“. K úpravám je nutné uvést komentář jako zdůvodnění změny.

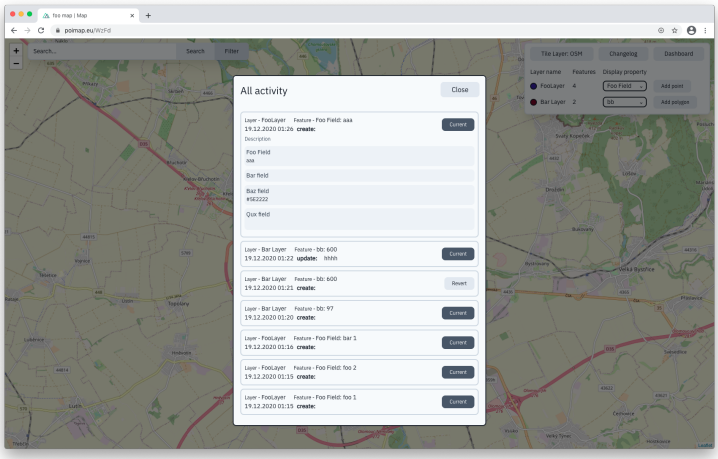


Obrázek 12: Úprava prostorového objektu

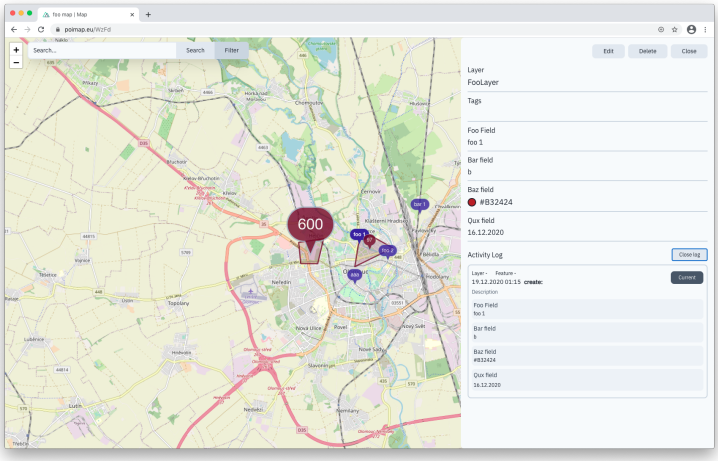
5.8 Archív

V horní pravé nabídce je obsaženo tlačítko „Archive“. Po kliknutí se otevře modální okno se změnami všech prostorových objektů dané mapy. Dále každý prostorový objekt obsahuje archív jeho změn. Ten lze zpřístupnit skrze tlačítko „Activity Log“.

U každé položky archívu lze kliknout na tlačítko „Revert“ a uvést komentář ke změně, pokud se nejedná o poslední změnu.



Obrázek 13: Archív

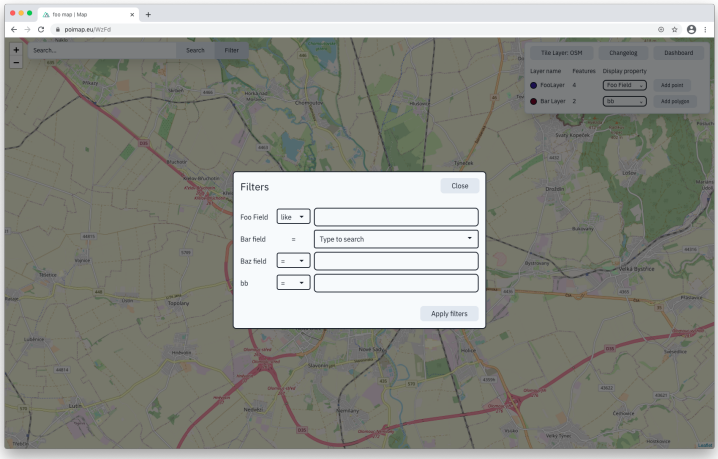


Obrázek 14: Archív prostorového objektu

5.9 Vyhledávání a filtrování

Při zadání vstupu do vyhledávacího řádku a stisknutí tlačítka „Search“ nebo klávesnice „Enter“ se v mapě vyhledají všechny prostorové objekty obsahující daný vstup v jakémkoliv z jeho políček.

Je možné také filtrovat z vybraných políček, které mapa obsahuje. Ty jsou získány z jejich vrstev. Filtr je možné otevřít prostřednictvím tlačítka „Filter“. Na základě typu vstupu políček je možné vybírat z porovnávacích operací. Například u vstupu typu číslo je možné vybrat operace „větší než“ a „menší než“ a u vstupu typu text operaci „like“, která porovnává, zda je hodnota podobná.



Obrázek 15: Filtrování

6 Další rozvoj aplikace

Aplikaci je možné dále rozvinout. Momentálně se nabízí vytváření vrstev a map, které jsou přístupné všem ostatním uživatelům. Je možné poskytnout případným uživatelům registraci a vytvoření jejich soukromého účtu. Při vytváření mapy a prostorového objektu by si mohli zvolit, zda bude daná mapa nebo vrstva zobrazena i ostatním uživatelům. Jednotliví uživatelé by mohli mít svůj vlastní seznam map a vrstev.

Pro každou vrstvu lze momentálně zvolit pouze jedno zobrazovací políčko. Do budoucna bych chtěl umožnit výběr více políček, které by se u prostorových objektů v mapě zobrazovaly pod sebou.

Při vytváření políček je momentálně omezené množství typů, které lze zvolit. Do budoucna plánuji přidat například typ pro obrázek.

Počet dlaždicových vrstev mapy je v aktuální verzi předem daný. Pokud by byl uživatel přihlášen, mohl by si seznam dlaždicových vrstev přizpůsobit a popřípadě vložit vlastní.

Mobilní aplikace momentálně neumožňuje kompletní offline verzi. Vyhledávání je možné pouze online a dlaždicové vrstvy se načítají ze serveru. Řešením by tedy bylo ukládání jednotlivých oblastí přímo do paměti zařízení.

Závěr

Cílem této práce bylo vyvinout otevřenou databázi bodů zájmu pomocí webových technologií s plnohodnotným využitím pro mobilní zařízení. Aplikace umožňuje vytváření tematických map, zaznamenávat body zájmu a prezentovat je vhodnou vizuální formou. U jednotlivých bodů zájmu je možné evidovat doplňující informace, se kterými lze dále pracovat.

Vývoj této aplikace mi rozšířil znalosti o práci s webovými technologiemi a možnosti využití mapových aplikací.

Jednotlivé návrhy pro rozšíření aplikace jsem zmínil výše. Jsem si vědom, že jsem nemohl téma zcela vyčerpat, ale chtěl bych se této problematice dál věnovat a případně problematiku zpracovat v diplomové práci.

Conclusions

The aim of this bachelor thesis was to develop an open database of points of interest using web technologies with full use for mobile devices. The application allows you to create thematic maps and record points of interest and present them in a suitable visual form. For individual points of interest, it is possible to record additional information, which can be further worked with.

The development of this application expanded my knowledge of working with web technologies and possibilities of using map applications.

I mentioned suggestions for future changes of the application above. I am aware that I could not completely exhaust the topic, but I would like to continue to address this issue in the future.

A Obsah přiloženého CD/DVD

Na samotném konci textu práce je uveden stručný popis obsahu přiloženého CD/DVD, tj. jeho závazné adresářové struktury, důležitých souborů apod.

bin/
: Kompletní adresářová struktura webové aplikace (v ZIP archívu) pro zkopírování na webový server. Adresář obsahuje i všechny runtime knihovny a další soubory potřebné pro pro bezproblémový provoz webové aplikace na webovém serveru.

doc/
: Text práce ve formátu PDF, vytvořený s použitím závazného stylu KI PřF UP v Olomouci pro závěrečné práce, včetně všech příloh, a všechny soubory potřebné pro bezproblémové vygenerování PDF dokumentu textu (v ZIP archívu), tj. zdrojový text textu, vložené obrázky, apod.

src/
: Kompletní zdrojové kódy webové aplikace se všemi potřebnými knihovnami a dalšími soubory potřebnými pro bezproblémové vytvoření spustitelných verzí adresářové struktury pro zkopírování na webový server.

readme.txt
: Instrukce pro nasazení webové aplikace na webový server, včetně všech požadavků pro její bezproblémový provoz, a webová adresa, na které je aplikace nasazena pro účel testování při tvorbě posudků práce a pro účel obhajoby práce.

U veškerých cizích převzatých materiálů obsažených na CD/DVD jejich zahrnutí dovolují podmínky pro jejich šíření nebo přiložený souhlas držitele copyrightu. Pro všechny použité (a citované) materiály, u kterých toto není splněno a nejsou tak obsaženy na CD/DVD, je uveden jejich zdroj (např. webová adresa) v bibliografii nebo textu práce nebo v souboru `readme.txt`.

Literatura

[1] REGINA O. OBE, Leo S. Hsu. *PostGIS in Action*. 2011.

[2] WIKIPEDIE. *Framework*. 2019. Dostupný také z: ⟨https://w.wiki/uhD⟩.

[3] BRENDA JIN Saurabh Sahni, Amir Shevat. *Designing Web APIs*. První vyd. 2018.

[4] MACRAE, Callum. *Vue.js: Up and Running*. První vyd. 2018.

[5] *OpenStreetMap*. Dostupný z: ⟨https://www.openstreetmap.org/way/27089312⟩.

[6] *Mapy.cz*. Dostupný z: ⟨https://mapy.cz/?x=17.262&y=49.592&z=18&q=university⟩.

[7] WIKIMEDIA. *File:MVC-Process.svg*. 2020. Dostupný také z: ⟨https://w.wiki/uhC⟩.

[8] WIKIPEDIE. *Objektově relační mapování*. Dostupný také z: ⟨https://w.wiki/uhB⟩.

[9] WIKIPEDIE. *JavaScript Object Notation*. Dostupný také z: ⟨https://w.wiki/uhA⟩.

[10] WIKIPEDIE. *Progresivní webové aplikace*. Dostupný také z: ⟨https://w.wiki/uh9⟩.

[11] *Leaflet layer*. Dostupný z: ⟨https://leafletjs.com/examples/extending/extending-2-layers.html⟩.