Systém pro správu panoramat v rámci webové prezentace

Martin Juřík

Bakalářská práce
2020

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav automatizace a řídicí techniky

Akademický rok: 2019/2020

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

Jméno a příjmení:	**Martin Juřík**
Osobní číslo:	**A16684**
Studijní program:	**B3902 Inženýrská informatika**
Studijní obor:	**Informační a řídicí technologie**
Forma studia:	**Kombinovaná**
Téma práce:	**Systém pro správu panoramat v rámci webové prezentace**
Téma práce anglicky:	**A Panorama Management System for Web Presentations**

Zásady pro vypracování

1. Seznamte se s aplikacemi pro prohlížení 360° panoramat prezentovaných v rámci internetu.
2. Nastudujte možnosti nastavení a úpravy panoramat a zobrazovaných klíčových bodů.
3. Vhodným způsobem definujte požadavky na aplikaci a vyberte vhodné technologie pro implementaci.
4. Navrhněte strukturu vyvíjené aplikace a rozdělení jednotlivých dílčích částí.
5. Implementujte aplikaci s využitím zvolených technologií a proveďte testování.

Rozsah bakalářské práce:
Rozsah příloh:
Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

1. GASSTON, Peter. Moderní web. Přeložil Ondřej BAŠE. Brno: Computer Press, 2015. ISBN 978-80-251-4345-2.
2. WIDIYANINGTYAS, Triyanna, Didik Dwi PRASETYA a Aji P WIBAWA. Web-based Campus Virtual Tour Application using ORB Image Stitching. In: *2018 5th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI): 2018 5th* s. 46?49. ISSN null. Dostupné z: doi:10.1109/EECSI.2018.8752709.
3. SKLAR, David. *PHP 7: praktický průvodce nejrozšířenějším skriptovacím jazykem pro web*. Přeložil Jan POKORNÝ. Brno: Zoner Press, 2018. Encyklopedie Zoner Press. ISBN 978-80-741-3363-3.
4. WELLING, Luke a Laura THOMSON. *Mistrovství PHP a MySQL*. Přeložil Ondřej BAŠE. Brno: Computer Press, 2017. ISBN 978-80-251-4892-1.
5. LAURENČÍK, Marek. *Tvorba www stránek v HTML a CSS*. Praha: Grada Publishing, 2019. Průvodce (Grada). ISBN 978-80-271-2241-7.

Vedoucí bakalářské práce: **Ing. Peter Janků**
Ústav informatiky a umělé inteligence

Datum zadání bakalářské práce: **20. prosince 2019**
Termín odevzdání bakalářské práce: **15. května 2020**

doc. Mgr. Milan Adámek, Ph.D.
děkan

prof. Ing. Vladimír Vašek, CSc.
ředitel ústavu

Ve Zlíně dne 20. prosince 2019

Prohlašuji, že

- beru na vědomí, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou/bakalářskou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne …….

podpis diplomanta

ABSTRAKT

Protože jsou v dnešní době panoramata čím dál více populárnější, objevují se nové požadavky, jak s nimi pracovat. Tato bakalářská práce se zabývá tvořením webové aplikace, která umožňuje spravovat panoramata. Součástí práce je přehled aplikací, které dovedou tvořit panoramatické snímky a také popis webových technologií včetně využití pro danou problematiku. Cílem této práce je vytvořit webovou aplikaci, která může dané panoramata ukládat na serveru a pracovat s nimi skrze webový prohlížeč.

Klíčová slova:

panorama, marzipano, správa, systém, php, html, css, javascript, bootstrap, knockout

ABSTRACT

In these days panoramas are getting more popular than ever and thank to this, new requirements are needed to work with them. This Bachelor‘s thesis gives a brief overview of creating a web application which allows managing panoramas. This work also contains a summary of applications which are used for creating panoramas and also the description of web technologies including their use for the aforesaid problem. The aim of this thesis is to create a web application that would allow storing panoramas on the server and working with them using a web browser.

Keywords:

panorama, marzipano, management, system, php, html, css, javascript, bootstrap, knockout

Poděkování

Tímto bych chtěl poděkovat vedoucímu práce, kterým byl pan Ing. Peter Janků, Ph.D. Za ochotu, pomoc a odborné rady v této nelehké době, které mi při zpracování této práce poskytl.

OBSAH

ÚVOD

Využívání, tvoření a upravování 360° fotografií – neboli panoramat v poslední době získává velmi na popularitě. Vezmeme si například takový seznam.cz nebo google.com a jejich mapy. Prohlížení ulic není nic jiného než sada panoramatických snímků, mezi kterými se lze přesouvat. Co když ale chceme mít na jednom místě několik takovýchto panoramat a upravovat je přímo v prohlížeči nebo si je prohlížet nezávisle na sobě?

Tato bakalářská práce se zabývá tvorbou webové aplikace, která dokáže evidovat a také upravovat panoramatické snímky. Práce je rozdělena na teoretickou a praktickou část přičemž teoretická část obsahuje části tři a praktická zbylé dvě.

V první části této bakalářské práce je rozbor aplikací, které vytváří panoramatické snímky a jaké nástroje pro to využívají. Také jsou zde napsány důvody, proč má práce pracuje primárně s konkrétním softwarem.

V další části se práce soustředí na to, co jsou to klíčové body v panoramatech a jak je aplikace vytváří a jak je lze modifikovat.

Ve třetí části se definují požadavky na aplikaci včetně popisu technologií, které se při tvorbě aplikace využívají.

Následuje čtvrtá část resp. první praktická část, kde se řeší samotný návrh aplikace a rozdělení na dílčí části, které jsou stěžejní pro výslednou aplikaci.

V poslední části jsou popsány konkrétní postupy a řešení dané problematiky, včetně popisu jednotlivých serverových metod, které aplikace využívá pro svůj chod. Mimo jiné je zde popsáno, co vytvořená aplikace umí a jaká je její výsledná podoba.

I. TEORETICKÁ ČÁST

1 APLIKACE PRO PROHLÍŽENÍ PANORAMATICKÝCH SNÍMKŮ

Existuje několik různých aplikací pro prohlížení panoramatických snímků. Mohou být:

- Placené/zdarma.
- Desktopové.
- Mobilní.
- Webové.

V následující kapitole je napsán přehled aplikací prezentovaných v rámci internetu.

1.1 Krpano

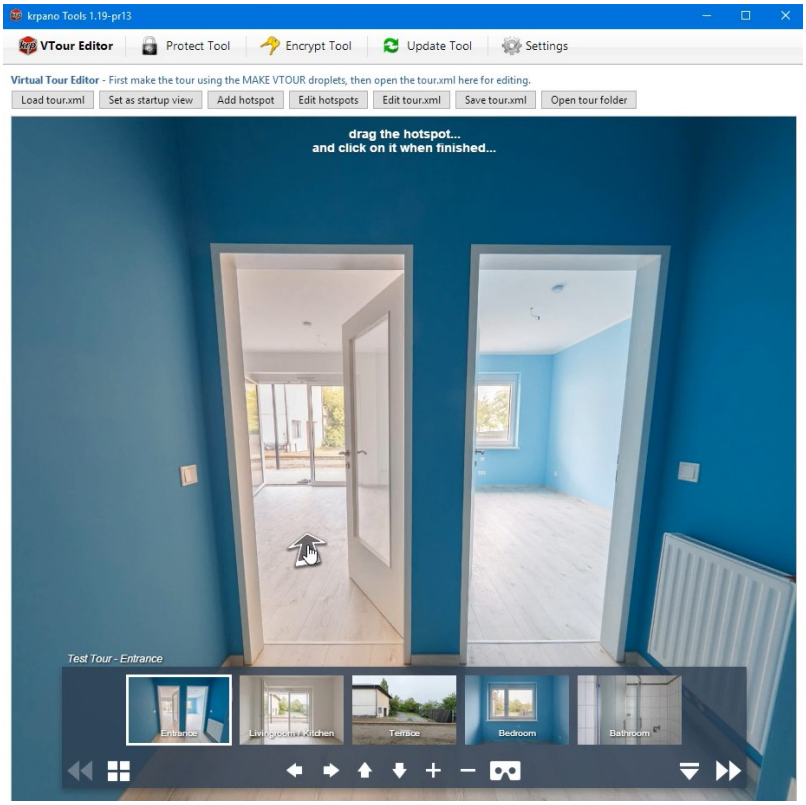
Jednoduchý a velmi výkonný software, který dokáže zpracovávat obrázky s vysokým rozlišením a vytvářet různé virtuální túry, ke kterým lze přidávat například i hudbu nebo si upravovat interface podle vlastního přání. Jedná se o placený software, kde je kladen důraz na jednoduchost, výkon a multiplatformost [6].

1.1.1 Jak funguje?

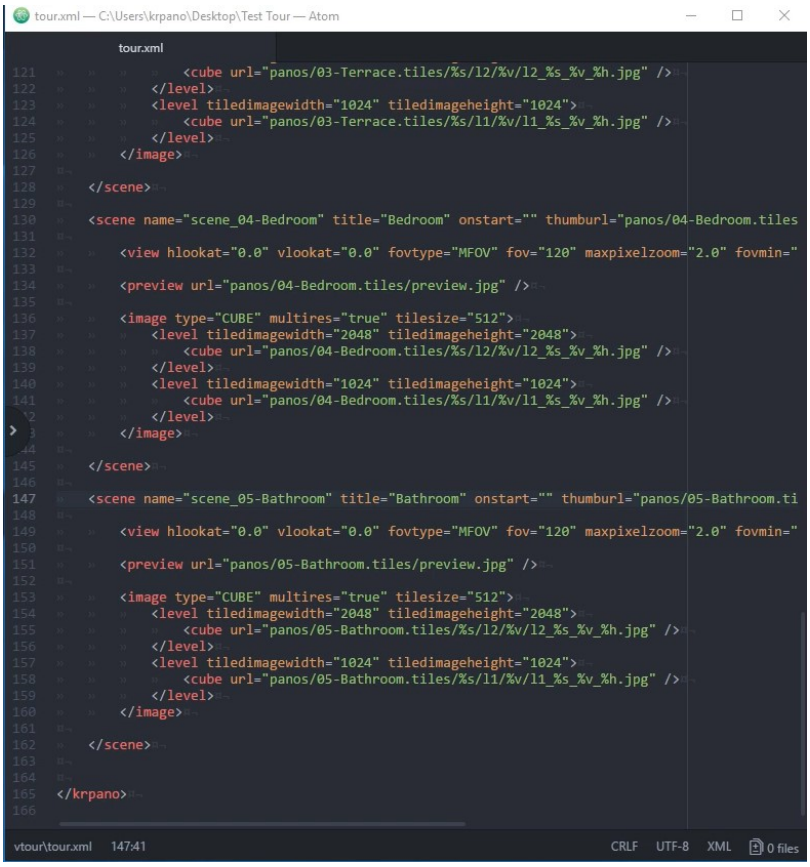
Krpano využívá XML soubory, které v sobě uchovávají nastavení programu. Tyto soubory mohou být jak napsány, tak editovány v jakémkoliv textovém editoru. Nicméně u XML souborů je důležité zachovat syntaxi, jinak program nebude fungovat správně.

Logiku aplikace lze ovlivňovat funkcemi. Krpano má předem definované funkce, které se nazývají akce. Je možné využít jak existující akce, tak si vytvořit vlastní a tím upravovat program dle svého přání. Kromě těchto vlastností lze používat spousty různých pluginů pro úpravu výsledné aplikace.

Krpano lze vyexportovat do HTML pro webové prohlížení. Samotná modifikace panoramat se provádí pouze přes aplikaci, případně přímou modifikací XML souboru, kde je nutná znalost aplikace [6].



Obrázek 1 – Ukázka aplikace krpano [6]



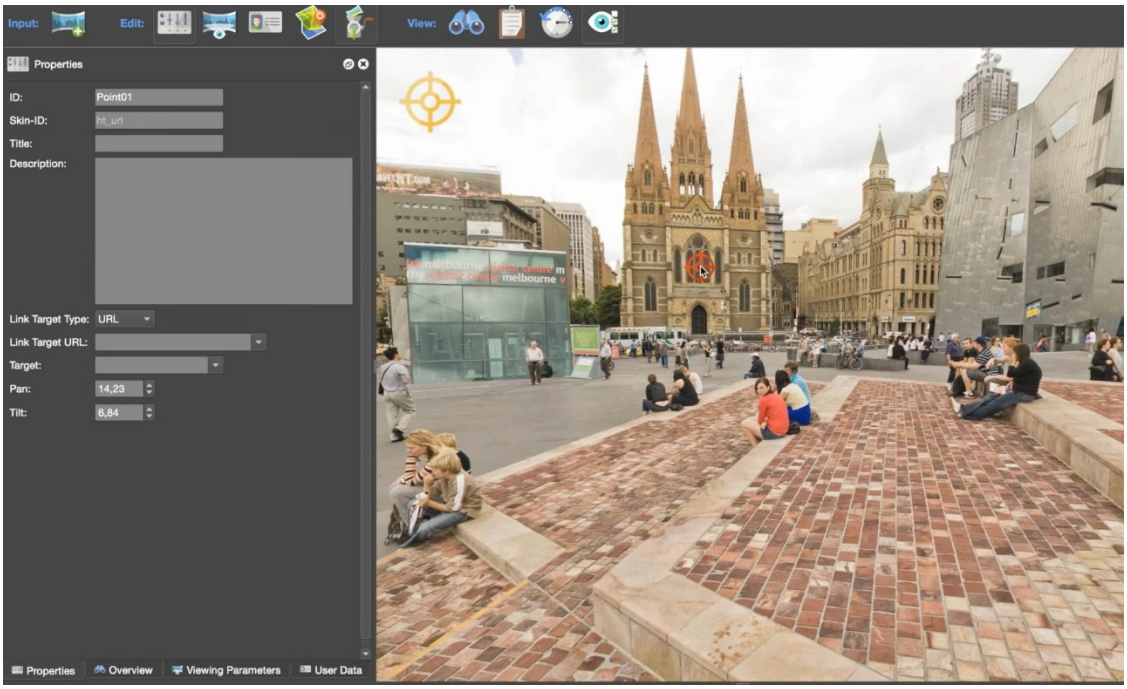
Obrázek 2 – Ukázka nastavení v XML [6]

1.2 Pano2vr

Stejně jako Krpano je i Pano2vr placený software. Jedná se o mnohem sofistikovanější program, než je Krpano. Pano2vr je profesionální software, který už se velmi podobá aplikaci Adobe Photoshop a jiným velkým programům na úpravu fotografií a videí. Je mnohem více uživatelsky přívětivější než Krpano. Z vašich panoramatických snímků lze například umazávat objekty, které se vám nehodí. Přidávat různé grafické prvky, tlačítka, zvuky nebo efekty. Nechybí ani důležité přidávání hotspotů [7].

1.2.1 Jak funguje?

Využívá XML, kde má uchované informace o panoramatických snímcích. Jelikož se nejedná o open source, je problém zjistit více informací, jak přesně aplikace funguje. Autoři na vše využívají vlastní knihovny včetně té JavaScriptové. Ta slouží na zobrazování panoramat, které jsou vyexportovány do HTML. Díky tomu si lze prohlížet panoramata i z webu, nicméně nutno říct, že samotná editace panoramat je možná pouze ze zmiňované aplikace [7].



Obrázek 3 – Ukázka aplikace Pano2vr [7]

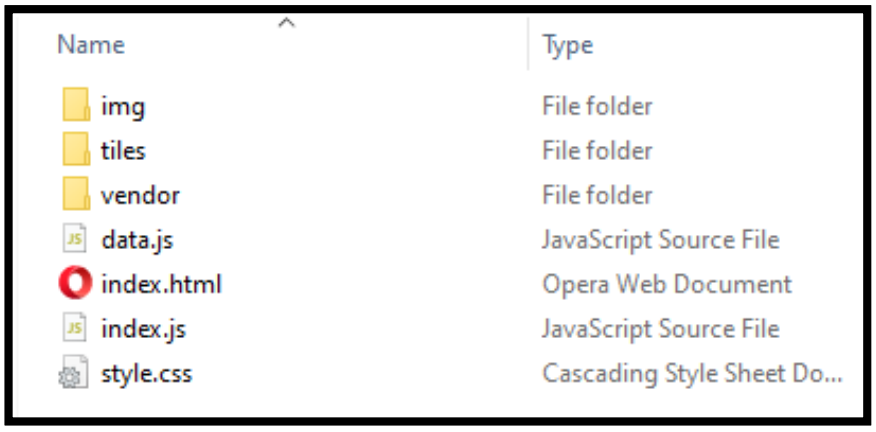
1.3 Marzipano

Aplikace, která je zdarma a je open source. Tedy nejlepší možná varianta pro tuto práci. Jedná se o velmi jednoduchou a přehlednou aplikaci, která funguje jak na všech různých mobilních zařízení, tak i napříč všemi novějšími prohlížeči. Většina z nich má dnes stejné jádro, takže s jinými prohlížeči by Marzipano nemělo mít problém [8].

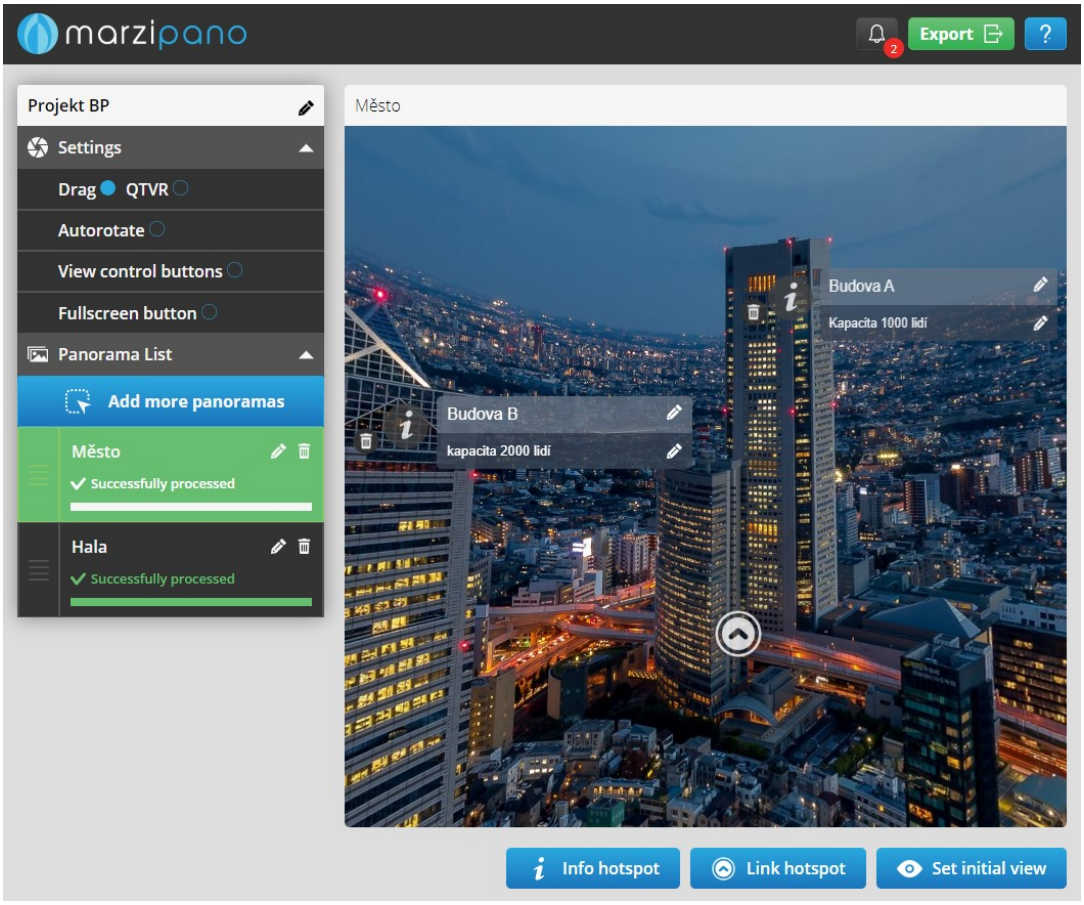
1.3.1 Jak funguje?

Existují dvě varianty, jak s Marzipanem pracovat. Buď si jej lze nainstalovat na své zařízení podle návodu na stránkách aplikace nebo s ním jde pracovat přímo na webu jako s aplikací. Ta funguje tak, že se do aplikace nahraje panoramatický snímek a pak jej přes interface lze editovat. Jakmile je práce na panoramatickém snímku hotova, stačí zadat export a začne se stahovat projekt, který je zabalený v zipu. Tento zip obsahuje složku *app-files*, která zahrnuje celý projekt jako webovou stránku. Tedy všechny potřebné knihovny, HTML a kaskádové styly, které jsou potřeba pro chod webového panoramata. Hlavní a stěžejní pozitivum Marzipana je fakt, že se jedná o open source. To znamená, že jakmile se projekt vyexportuje, obsahuje veškerý kód, který si zkušenější programátor může upravit dle sebe a přidávat tam například i další funkce. Na to slouží soubor *index.js*, což je mozek Marzipana. Do tohoto souboru si lze přidávat vlastní funkce, případně upravovat ty stávající [8].

Také Marzipano si ukládá nastavení panoramat do souborů, ale na rozdíl od svých konkurentů si ukládá nastavení do JavaScriptového souboru, který je zde v podstatě JSON (viz kapitola 3.5).



Obrázek 4 – Ukázka rozbaleného Marzipano projektu



Obrázek 5 – Ukázka Marzipano aplikace [8]

```
"initialViewParameters": {
  "pitch": 0,
  "yaw": 0,
  "fov": 1.5707963267948966
},
"linkHotspots": [
  {
    "yaw": -0.27129948575454854,
    "pitch": 0.11282182684876219,
    "rotation": 0,
    "target": "1-ukzka-haly"
  }
],
"infoHotspots": [
  {
    "yaw": -1.2591838431237203,
    "pitch": -0.14599441606349828,
    "title": "Budova 2",
    "text": "volné kanceláře"
  },
  {
    "yaw": -0.11430619399856212,
    "pitch": -0.5986462726622648,
    "title": "Budova 1",
    "text": "kapacita až 1000 lidi"
  }
```

Obrázek 6 – Ukázka nastavení v JSON struktuře

2 NASTAVENÍ PANORAMAT A KLÍČOVÝCH BODŮ

V panoramatických snímcích jde upravovat ledacos. Například Pano2vr umí mazat i objekty z panoramat a nahrazovat je jinými objekty. Jedná se o již velmi pokročilé funkce. Tato práce se však bude věnovat hlavně Marzipanu a jeho klíčovým bodům neboli hotspotům. Je to vlastnost, která je společná pro všechny aplikace zmíněné v této práci. Jedná se o jakési body, které mohou nést informace o daném bodu uvnitř panoramatu. Díky těmto bodům lze popisovat různé budovy, místa nebo dokonce odkazovat na jiná panoramata.

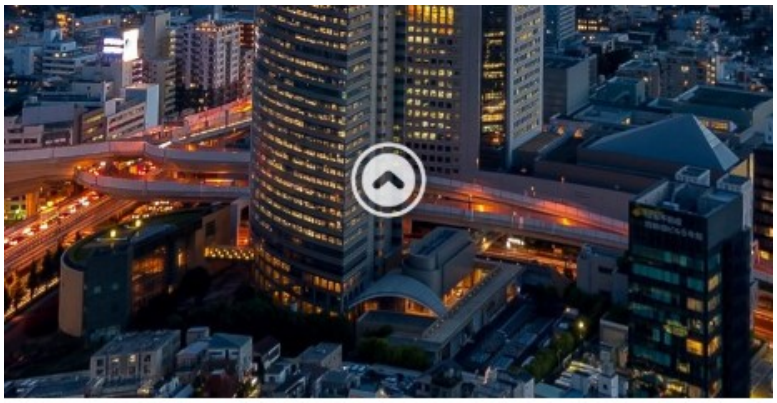
Nejdůležitější informací hotspotu je souřadnice. Každé panorama má soustavu x/y, se kterou pracuje a Marzipano není výjimka. Jedná se o atributy *yaw* a *pitch* (viz Obrázek 7). Pomocí těchto souřadnic lze určit polohu konkrétních klíčových bodů.

2.1 Typy hotspotů a initial view

- **infoHotspot** – tzv. informační, fungují tak, že při kliknutí na daný bod se zobrazí informace o daném místě. V tomto případě Marzipano zobrazuje jednoduchý text.

Obrázek 7 – Ukázka infoHotspotu a jeho nastavení

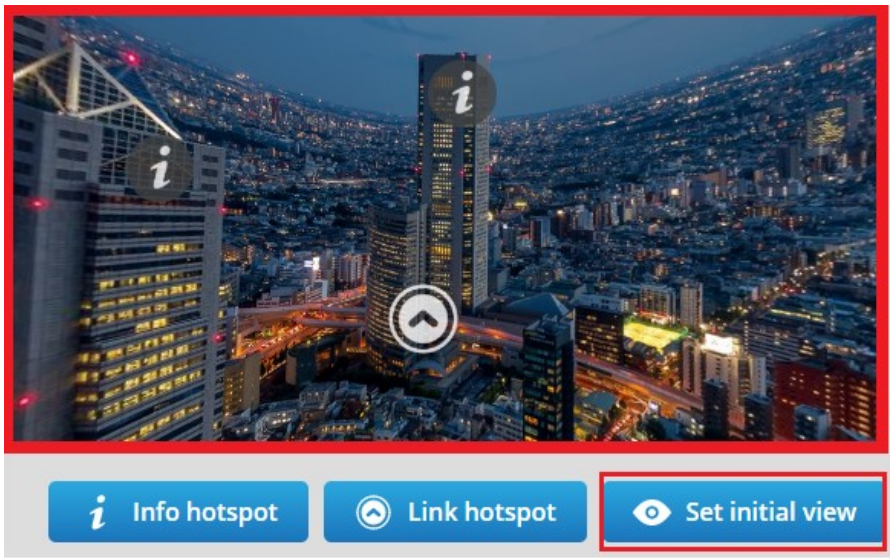
- **linkHotspot** – tzv. odkazové, fungují tak, že při kliknutí odkáží na jiné panorama. Je to již zmiňovaný princip prohlížení ulic v google nebo seznam mapách.



```
"linkHotspots": [
  {
    "yaw": -0.2630715009672553,
    "pitch": 0.2623884512111996,
    "rotation": 0,
    "target": "1-hala"
  }
],
```

Obrázek 8 – Ukázka linkHotspotu a jeho nastavení

- **initialView** – neboli pohled, který má uživatel vidět jako první, když se otevře panorama.




```
"initialViewParameters": {
  "pitch": 0,
  "yaw": 0,
  "fov": 1.5707963267948966
},
```

Obrázek 9 – Ukázka initialView a jeho nastavení

3 POŽADAVKY A WEBOVÉ TECHNOLOGIE

Následující kapitola stručně vysvětluje, co jsou to požadavky na vyvíjenou aplikaci a jak se získávají. Dále obsahuje souhrn všech využitých technologií a potenciální zabezpečení.

3.1 Požadavky na vyvíjenou aplikaci

Jedná se o body, které vyvíjená aplikace musí mít a jaké technologie má využívat. Tyto informace jsou zpravidla definovány zadavatelem (např. zákazníkem) [19].

3.1.1 Nefunkční požadavky

Nefunkční požadavky jsou ty, které se neobjevují v případech užití (use case). Primárně se řeší, jakým způsobem aplikace poskytne potřebnou funkcionalitu než to, jaké funkce bude aplikace obsahovat. [19].

Existují 3 různé oblasti nefunkčních požadavků[19]:

- **Technické omezení –** jedná se o specifikování technologií, které aplikace musí využívat. Příklad: „Naše firma má pouze Java vývojáře, takže musíme vyvíjet v jazyku Java.“
- **Business omezení –** jedná se o požadavek čistě obchodní. Příklad: „Aby naše firma rozšířila svou uživatelskou základnu, musíme využívat nástroj XYZ.“
- **Atributy kvality –** definují požadavky aplikace ve smyslu využitelnost, dostupnost, přenositelnost, výkonnost atd.

3.1.2 Funkční požadavky

Měly by definovat základní akce, kde se jasně určí co je vstupem a co je výstupem. Zpravidla jsou tyto požadavky psány jako seznam „systém musí…“ [20].

Požadavky zahrnují [20]:

- Definice posloupnosti operací.
- Ošetření nekorektního chování (např. zvládnutí chybových stavů a případná obnova).
- Definice parametrů.
- Co je vstup a co je výstup.

3.2 HTML

Webová stránka je zapsána v kódu HTML a jeho struktura se skládá ze dvou částí [5]:

- Texty, které budou na stránce zobrazeny.
- Řídící příkazy jazyka HTML, které se označují jako elementy (někdy také tagy).

Když se vytváří webové stránky, tak všechny texty, které se do nich zapíší, slouží pouze jako informace pro webový prohlížeč. To, jak prohlížeč stránky zobrazí, lze ovlivnit použitím elementů jazyka HTML [5].

Pravidla pro zobrazování textu [5]:

- Prohlížeč ignoruje přechody na nový řádek.
- Prohlížeč nezobrazuje prázdné řádky.
- Mezera je brána jako oddělovač mezi slovy.
- Tabulátory se berou jako jedna mezera.

Tato pravidla jsou jasná, protože prohlížeč musí zobrazovaný text přizpůsobit počítači, na kterém se webové stránky spouští.

3.2.1 Chyby v HTML

Většina programových jazyků včetně jazyka PHP odhalí chybu a ohlásí ji. Nedovolí dál pokračovat. Webový prohlížeč zpravidla žádnou chybu neohlásí a stránku se pokouší zobrazit. Výsledek pak není dle našich představ [5].

Webové prohlížeče se dokáží vyrovnat i s nekorektně zapsaným kódem. Není však dobré chyby ignorovat. Vývoj prohlížečů směřuje ke stále širším možnostem, ale měly by se také zvyšovat požadavky na správnost zapsaného kódu [5].

3.2.2 Odkaz v rámečku

Je to funkce HTML, která dokáže vložit webovou stránku do webové stránky. Jde o tzv. vložený rámeček, který vytváří sekundární okno, v němž lze zobrazit soubor nebo jiný odkaz [5].

Vložený rámeček se vytváří párovým elementem <*iframe*> [1]. Do těla elementu lze zapsat text, který se zobrazuje v případě, když rámečky nejsou podporovány. To už se ale dnes stává velmi ojediněle, protože prakticky všechny prohlížeče iframe podporují [5].

Důležité atributy elementu <*iframe*> [1, 5]:

- *Src* je odkaz na vložený obsah.

- *Width* a *height* neboli šířka a výška rámečku uváděná v pixelech nebo procentech okna prohlížeče.
- *Align* je zarovnání rámečku. Uplatňuje se, pokud je rámeček umístěn přímo v textu.

3.3 CSS - kaskádové styly

Jsou to pravidla, která obsahují informace o formátování webové stránky. Pomocí těchto stylů lze nastavovat vzhled celé webové stránce nebo jejím dílčím částem. Při změně daného stylu se automaticky mění vzhled všech tagů, které jsou pomocí tohoto stylu nastaveny [5].

Výhody využívání kaskádových stylů [5]:

- Poskytuje řadu možností při formátování, kterých použitím HTML nejde docílit.
- Umožňuje nastavit jednotný vzhled u celého webu, který lze snadno měnit.
- Obsah a vzhled stránek jsou od sebe jasně odděleny.

Definice stylu obsahuje jedno nebo více pravidel, které určují, co a jak se má formátovat. Tato pravidla jsou složena z názvu atributu, dvojtečky a hodnoty. Každé pravidlo je ukončeno středníkem [5].

Příklad pravidla:

- *color:red;*

Tento zápis říká, že pro daný úsek bude použito červené písmo. K definici stylu lze využít také atribut *style*, který se dá použít u většiny elementů HTML [5].

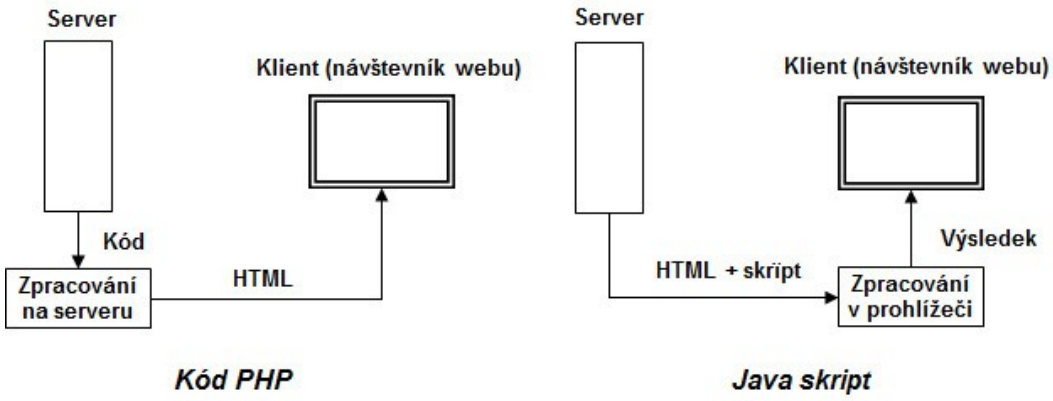
Příklad atributu *style*:

- *<li style="color:red;">*

Znamená to, že v jednom bodu seznamu bude použito červené písmo. Tento zápis se využívá zejména pro konkrétní případy pro formátování určitých úseků, které se jinde nevyskytují [5].

3.4 JavaScript

Díky JavaScriptu lze výrazně zvýšit možnosti webových stránek. Využívají se zde tzv. klientské skripty, což znamená, že program je součástí kódu HTML a provádí vše na straně klienta. Opačný přístup využívá jazyk PHP, který kód provede na straně serveru a k uživateli tak dorazí už hotový kód [5].



Obrázek 10 – Práce kódu PHP a JavaScriptu [5]

Hlavní výhodou JavaScriptu je fakt, že provádění kódu nevyžaduje další komunikaci se serverem. Využívání JavaScriptu sebou nese ale i určité nevýhody [5]:

- Samotný JavaScript má méně možností než PHP.
- Způsob, jakým je kód prováděn, může být závislý na použitém prohlížeči.

Kód zapsaný v JavaScriptu se zpravidla provádí ihned po načtení stránky [5].

Způsoby vložení JavaScriptu do webové stránky [5]:

- Skript je zapsán přímo v kódu HTML.
- Skript je načten ze samostatného souboru.
- Příkazy jsou zapsány jako hodnota atributu v elementu HTML.

3.4.1 XMLHttpRequest

Funkce, která umožnuje vytvořit HTTP požadavky v JavaScriptu. I když má ve svém názvu slovo XML, tak umí pracovat s jakýmkoliv typem souboru. Díky této funkci lze nahrávat/stahovat soubory, zjišťovat průběh operace a mnohem víc [12].

Velmi užitečný je handler *onreadystatechange*. Pokud je použit, tak lze definovat funkci, která bude čekat na odpověď serveru [12].

XMLHttpRequest obsahuje následující parametry [12]:

- **Metoda** – HTTP metoda. Tradiční GET a POST.
- **URL** – adresa požadavku.
- **Async** – pokud je nastaven na *false*, tak je požadavek synchronní.
- **Uživatel a heslo** – pro základní http autorizaci – pokud je potřebná.

Metody GET a POST [5]**:**

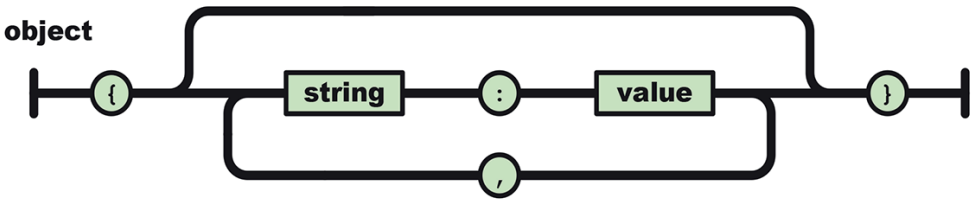
- GET předává data přes adresní řádek prohlížeče. Délka textového řetězce je omezena na cca 3000 znaků. Nehodí se pro odesílání citlivých dat, protože odesílané údaje jdou vidět na adrese.
- POST odesílá data jako samostatný HTTP objekt. Délka není nijak omezená a odeslané údaje jsou skryté. Na rozdíl od GET lze odesílat i soubory.

3.5 JavaScript Object Notation

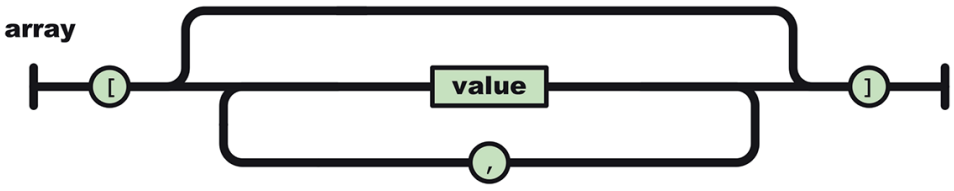
Známý především jako JSON. Jedná se o jazykově nezávislý zápis postavený na bázi psaného textu. Je dobře čitelný a jednoduše generovatelný. Byl vytvořen na základě programovacího jazyku ECMAScript [4].

Hodnoty, které může JSON nabývat jsou [4]:

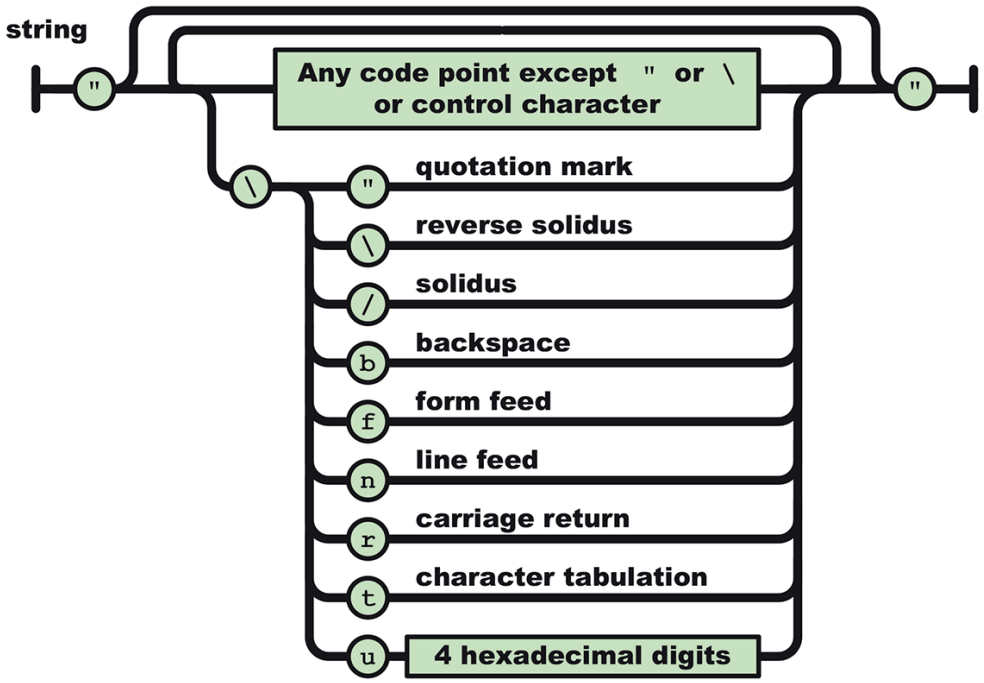
- objekt,
- pole,
- číslo,
- textový řetězec,
- boolean true/false,
- null.



Obrázek 11 – Zápis objektu [4]



Obrázek 12 – Zápis pole [4]



Obrázek 13 – Zápis textového řetězce [4]

3.6 PHP

Opak jazyku JavaScript, který provádí kód až po načtení stránky na straně klienta [5]. PHP je skriptovací jazyk, který funguje na straně serveru. Znamená to, že kód v PHP se provádí dříve, než se odesílá obsah webové stránky ke klientovi [3]. PHP tedy vytvoří část nebo celou webovou stránku a výsledek poté zobrazuje uživateli v prohlížeči. Díky tomu není potřeba řešit typ prohlížeče [5].

PHP nedokáže zareagovat na události, které se dějí na webové stránce. Na druhou stranu má oproti JavaScriptu jisté výhody [3, 5], např.:

- Práce se soubory – stahování, extrahovaní, tvorba nových souborů, exportování dat, do různých typů souboru a podobně.
- Zpracování údajů.
- Odesílání e-mailů.

3.7 Jak zabezpečit web v PHP

Většina slabin webu je vytvořena špatnými programovacími návyky nebo také nedostatečným povědomím o potenciálních problémech. Je to způsobeno i tím, že uživatelův vstup je brán jako důvěryhodný [15].

Při tvoření webu je nutné se ujistit, že to co je právě prováděno je validní (tedy že to, co se děje je v pořádku) a splňuje všechny podmínky. Také je nutné se ujistit, že výstup do HTML je řádně ošetřen, tedy aby se neprovedl škodlivý kód v případě, že by se útočník pokusil injektovat do obsahu. Pokud jsou dodrženy jednoduché a základní procedury pro každý web, významně se tak minimalizují potenciální bezpečnostní rizika [15].

3.7.1 SQL Injection

SQL injekce je jedna z nejvíce nebezpečných slabin webu. Pokud se ve skriptu dostává do SQL query parametr z PHP funkce $_GET. Může tím vzniknout riziko, že se do parametrů dostane škodlivý kód, který se odešle do databáze. Úspěšný útok, může zapříčinit získání dat z DB, včetně hesel, e-mailů a dalších informací [15].

3.7.2 Code Injection a Directory Traversal

Útok je opět postaven na používání $_GET. Útočník využije výhody, že skript obsahuje systémové funkce a může tak číst nebo spustit škodlivý kód na serveru. Například pokud skript využívá funkci exec() pro vykonávání operací s kombinací $_GET, útočník toho může využít. To samé platí pro využívání funkce include(). Útočník si může do parametru v adresním řádku zavolat funkce pro zjištění souborové struktury na serveru a dostat se tak ke spuštění skriptů, které by uživatel neměl vidět [15].

3.7.3 Cross-site Scripting (XSS útoky)

Spočívá v tom, že pokud PHP vkládá přímo do HTML struktury pomocí echo(), tak útočník může přes $_GET předat kus kódu, který se zapíše do webových stránek. Díky tomu může přidat např. přesměrování na cizí webové stránky, které mohou obsahovat kód pro ukradnutí cookies [15].

Web lze zabezpečit pomocí PHP funkcí htmlspecialchars() a htmlentities(), které konvertují speciální znaky [15].

3.7.4 Ukládání hesel

Práce s hesly je velmi důležitý aspekt v bezpečnosti webových technologií. Je to bezpečnostní vrstva navíc, která pomáhá vyhnout se potenciálním útokům. Pokud se podaří útočníkovi SQL injection, případně se mu podaří vyčíst data přímo ze serveru, ze kterých získá hesla, tak v případě, že jsou silně zahashované, to může být pro útočníka velký problém. Prolomit šifrování je totiž vždy složité. Dva nejdůležitější aspekty u ukládání hesel

jsou využití hash algoritmů a tzv. salt. Salt je textový řetězec, který je přidán k heslu ještě před tím, než je zašifrován. Pokud je salt dlouhý a nahodilý, tak to dokáže eliminovat některé metody prolomení [15].

Rady pro práci s hesly [15]:

- Nepoužívat slabé hash algoritmy jako je MD5 nebo SHA1.
- Nepoužívat stejné nebo slabé salt (nechat aby je nastavila funkce password_hash()).
- Netvořit vlastní šifrovací funkce.
- Zvýšením pracovního faktoru ve funkci password_hash() [14], lze docílit mnohem silnějších hashů, ale nastavením příliš vysokých hodnot, mohou ovlivnit výkon serveru.

3.7.5 Session fixation

Útočník nepotřebuje krást identifikátor relace. Cílem je získat validní ID relaci z webového serveru a poté donutit prohlížeč své oběti, aby získanou relaci ověřil ve webové aplikaci. Jakmile je ID relace ověřena, útočník získává úplný přístup do aplikace, stejně tak jako uživatel [16].

Možné řešení [17] je, že při přihlášení do aplikace se bude generovat nové ID relace. To se provádí v PHP pomocí funkce session_regenerate_id() [14]. Zapříčiní to, že aplikace vygeneruje novou relaci a tím pádem nastrčené ID od útočníka již nebude platné.

3.7.6 Session steal nebo také hijacking

Na rozdíl od Session fixation, který je brán jako bug aplikace a může jej ošetřit programátor, tak krádež relace není tak úplně problém aplikace, jelikož útočník může provést následující [16]:

- **Cross-side scripting** – využívá JavaScript, útočník odešle odkaz své oběti. Při načítání stránky, kód JavaScriptu přečte session ID z počítače své oběti a odešle jej útočníkovi.
- **Network Sniffing** – útočník si odchytne komunikaci mezi webovým serverem a uživatelem, aby získal ověřené session ID a současně tak získá přístup do aplikace, stejně tak jako jeho oběť.

Ošetření v aplikaci může být trochu obtížné. Jednou z variant je ta, aby aplikace umožnila aktivní pouze 1 relaci. To by znamenalo, že žádnou další relaci aplikace neumožní a tedy se

nemůže nikdo další do aplikace přihlásit [16]. Jinou variantou může být uložit do relace IP adresu, případně další informace o uživateli (např. jaký používá prohlížeč) a poté tyto informace při načítání stránek kontrolovat. Pokud uživatel, který stránku obsluhuje, je stejný jako ten co založil danou relaci, tak je vše v pořádku a aplikace je přístupná [18].

3.8 Bootstrap

Existuje spousty různých pomocníku při tvorbě webu. Bylo potřeba najít řešení, které lze reálně využít a usnadnit si práci. Jedna z nejlepších možností byla využít Bootstrap a to hlavně proto, aby se aplikace nemusela psát úplně od základu.

Jedná se o jeden z nejznámějších a nejpopulárnější open source nástrojů pro vytváření webu, včetně responsivního vzhledu, JavaScript pluginů a webových komponent [9].

Kromě toho, že je Bootstrap open source, tak jejich webové stránky obsahují i nespočet různých příkladů, které si lze stáhnout a na nich stavět. Za zmínku také stojí jeho rozsáhlé třídy v kaskádových stylech, které je dobré využít [9].

3.9 Knockout

Jedná se o open source knihovnu, která pomáhá tvořit responsivní zobrazení a uživatelsky přívětivé prostředí pro editaci souborů na základě definovaného modelu. Vždy když se data změní, Knockout na tuto změnu reaguje a mění tak obsah webových stránek [10].

Hlavní výhody [10]:

- **Elegantní hledání závislostí** – vždy když se změní vámi definovaný model, Knockout automaticky upravuje pouze tu část, která se změnila.
- **Deklarace vazeb** – jednoduchá a přímá cesta, jak spojit jednotlivé části user interface s data modelem.
- **Jednoduchá rozšiřitelnost** – implementace vlastního chování pomocí deklarace vazeb.

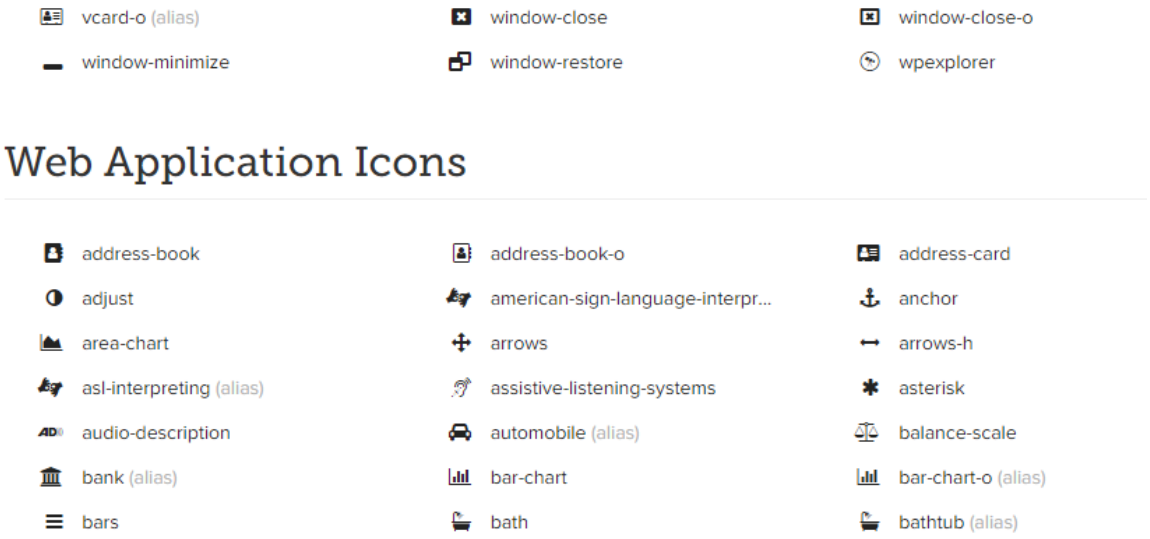
Další výhody [10]:

- **Knihovna napsána pomocí čistého JavaScriptu** – funguje jak na straně serveru, tak na straně klienta.
- **Lze ji přidat do již existujících webových aplikací** bez nutnosti jakékoliv změny v architektuře dané aplikace.

- **Kompaktní** – zhruba 13kb po gzippingu.
- **Funguje na všech prohlížečích** (IE 6+, Firefox 2+, Chrome, Safari, Edge a další).

3.10 Font Awesome

Jedná se o open source knihovnu obsahující vektorové ikony, které mohou být jakkoliv modifikovány pomocí CSS. Obsahuje až 675 ikon, bez nutnosti JavaScriptu. Ikonám je možné měnit vzhled, barvu, stíny a v podstatě všechno co je v CSS vůbec možné [13].



Obrázek 14 – Ukázka ikon z Font Awesome [13]

II. PRAKTICKÁ ČÁST

4 NÁVRH A DÍLČÍ ČÁSTI APLIKACE

Než začne práce na samotné aplikaci, je nutné definovat požadavky (viz Kapitola 3.1) včetně návrhu na vyvíjenou aplikaci a z těchto závěrů vyvodit potenciální dílčí části.

Požadavky jsou tvořeny primárně zákazníkem, ale v případě této práce byly požadavky tvořeny vedoucím a autorem.

4.1 Funkční požadavky

Aplikace musí:

- Umět nahrát zip s panoramatem na server a uložit ho do předem dané struktury.
    - Je nutné vymyslet strukturu aplikace.
- Pracovat s definovanou strukturou.
    - Mazat, přepisovat, kopírovat soubory.
- Umět vytvořit editační formulář pro soubor *data.js*, ve kterém je uloženo nastavení infoHotspotů a následně je modifikovat, včetně obrázků.
- Umožnit nahrát na server vlastní obrázky pro editační formulář.
- Zobrazit všechna panoramata na serveru a umožnit si je otevřít.
- Mít alespoň základní přihlášení do aplikace, aby nedocházelo k neoprávněným úpravám.
- Zachovat stejnou souborovou strukturu i pro URL.
- Kopírovat *index.js* z definované složky do projektu Marzipana.

Poslední bod je potřeba více rozvést, a tedy uvést důvod, proč kopírovat soubor *index.js* do projektu Marzipana. Jak již bylo vysvětleno, zmiňovaný soubor je mozek Marzipana a tvořená práce má tento soubor již předem připravený a modifikovaný. Znamená to, že aplikace bude vkládat vlastní a upravené funkce do všech projektů, které právě nahrává na server.

4.2 Nefunkční požadavky

Technické omezení

Marzipano projekt je webová stránka a je zpravidla uložen na serveru. Znamená to, že požadavkem je vytvoření webové aplikace, která bude pracovat s touto webovou stránkou a nejlépe přímo na serveru. Z toho vyplývá využít jazyk PHP, který má dostatek funkcí pro

správu serveru (viz Kapitola 3.6). Uživatel potřebuje rozhraní, kterým bude posílat požadavky na tento server. Využitím HTML (viz Kapitola 3.2) a JavaScriptu (viz Kapitola 3.4) lze připravit jak editační formuláře, tak zobrazení dat na serveru.

Business omezení

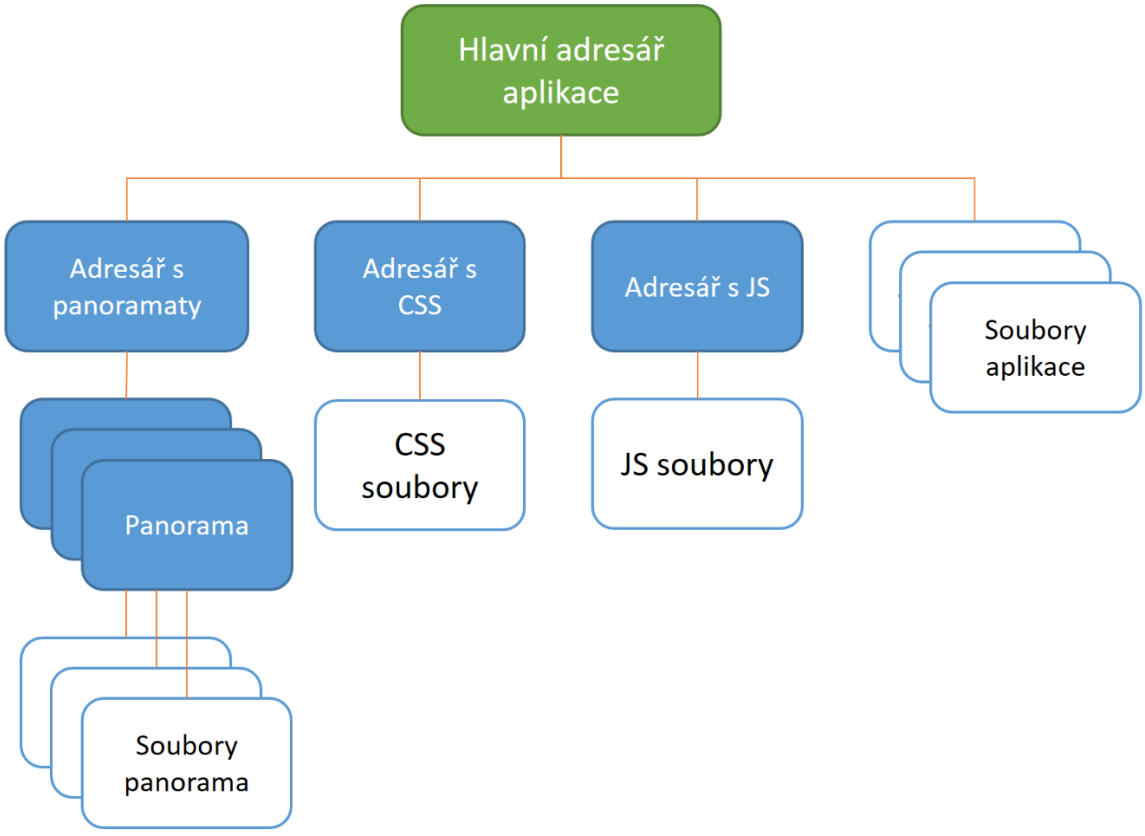
Tvořená aplikace musí být open source.

Atributy kvality

U tohoto bodu nevzešel žádný požadavek na přenositelnost, výkonnost apod.

4.3 Návrh souborové struktury

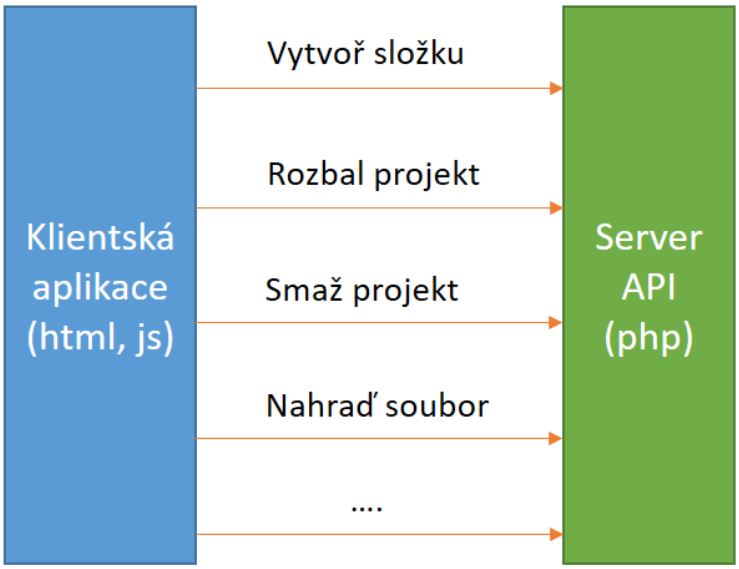
Základní serverová struktura musí obsahovat složku, ve které budou uložena panoramata Marzipana a dále složky, které obsahují důležité knihovny a soubory pro chod aplikace.



Obrázek 15 – Návrh souborové struktury

4.4 Návrh komunikace

Webové aplikace zpravidla fungují tak, že klientská aplikace odesílá požadavky na server a ten je vyřizuje. Většinou se jedná o asynchronní operace. Aplikace tvořená v této práci nebude výjimka. Bude rozdělená klasicky na klientskou část a serverovou část. K vyřizování požadavků bude sloužit serverové API (Application Programming Interface).



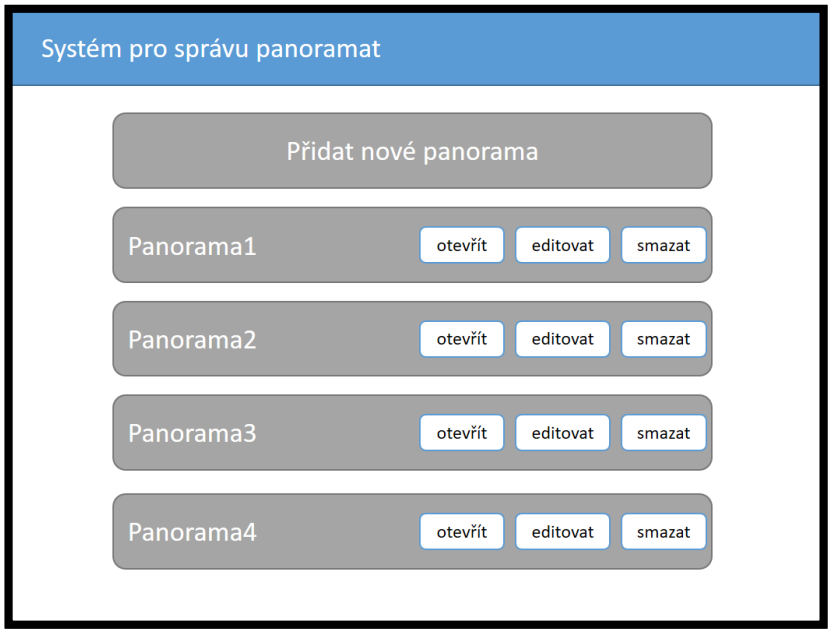
Obrázek 16 – Návrh komunikace

4.5 Návrh klientské aplikace

Vizuální část bude zahrnovat možnosti prohlížení panoramat a editování klíčových bodů.

4.5.1 Prohlížení panoramat

Aplikace bude zobrazovat všechna panoramata a mít tlačítka otevřít panorama, editovat klíčové body a smazání celého projektu. Základní návrh (viz Obrázek 17) toto splňuje.

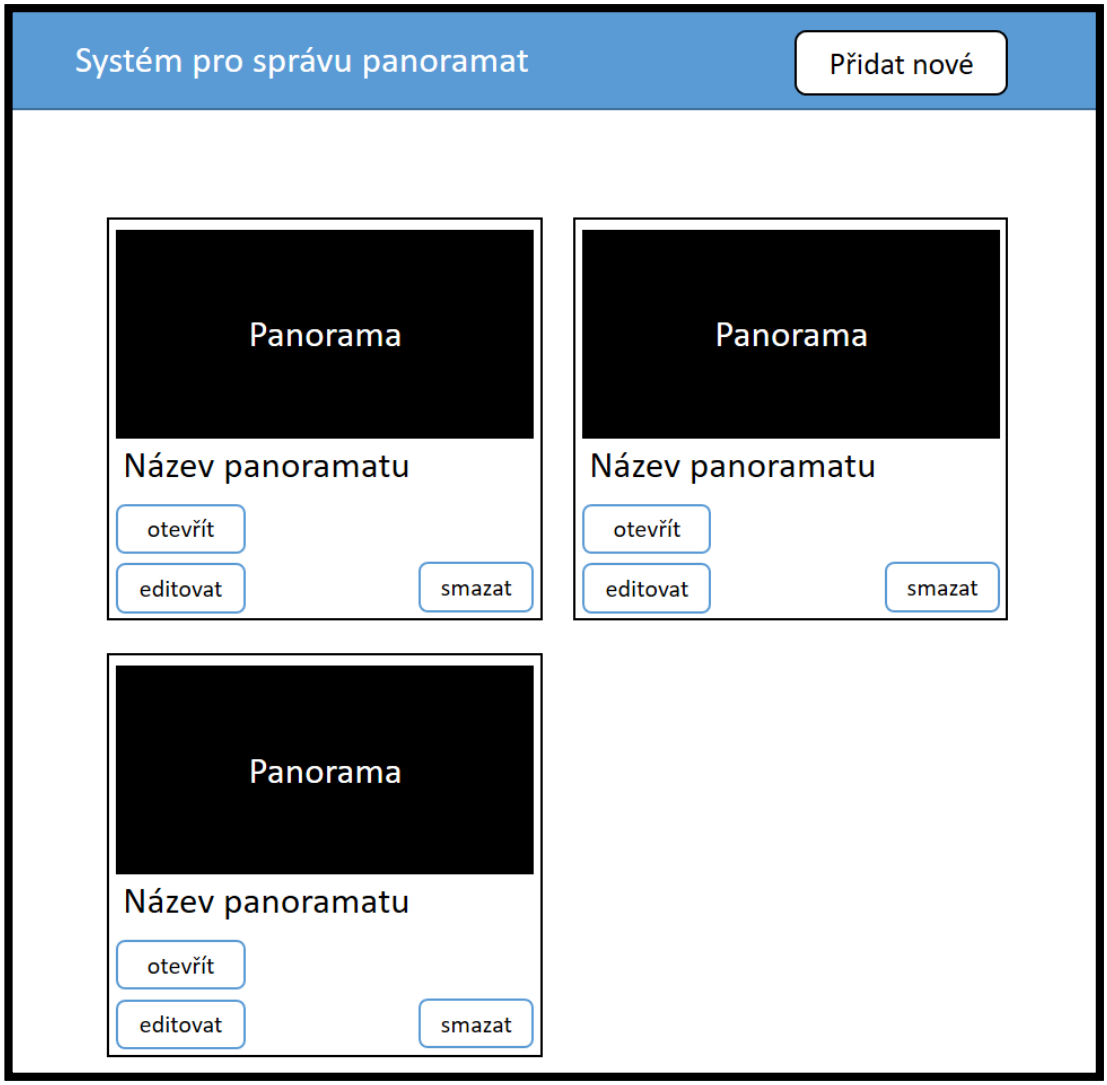


Obrázek 17 – Původní návrh hlavní strany aplikace

Ve studii o vytváření systému pro virtuální prohlídku univerzity [2] řeší mimo jiné i algoritmus pro tvoření samotného panoramatu. To však není cílem této práce. Nicméně jsou v ní hodnotné informace, jak byl vytvářen informační systém pro nahlížení jednotlivých panoramat. Jejich systém (viz Obrázek 18) zobrazuje panoramatické snímky jako galerii fotografií, přičemž po kliknutí z galerie se zobrazí daná fotografie.

Obrázek 18 – Ukázka aplikace pro nahlížení panoramat [2]

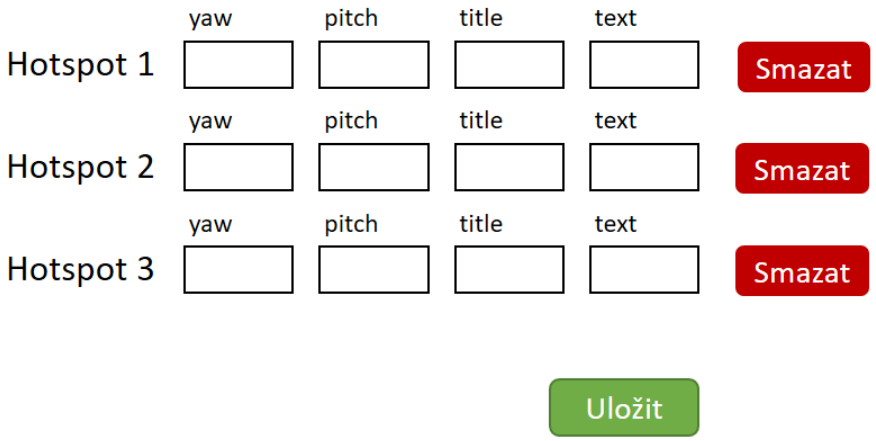
Původní návrh (viz Obrázek 17) byl na základě poznatků ze zmiňované studie předělán, aby se více podobal galerii (viz Obrázek 19).



Obrázek 19 – Nový návrh pro hlavní stranu

4.5.2 Úprava klíčových bodů

Po kliknutí na tlačítko upravit se otevře editační formulář, ve kterém lze modifikovat jednotlivé hotspoty. Návrh pro editaci je velmi jednoduchý (viz Obrázek 20).



Obrázek 20 – Ukázka návrhu editačního formuláře

Po kliknutí na tlačítko Uložit server zareaguje a uloží nové údaje do souboru *data.js*. Stejně tak zareaguje server na tlačítko Smazat a vymaže příslušné hotspoty.

4.6 Návrh serverové služby

Server musí vyřizovat požadavky a musí to být skript, který obsahuje všechny potřebné funkce pro správu. Toho lze docílit vytvořením jednotného API v PHP.

4.6.1 Práce se zipem a jeho soubory

Po odeslání na server, musí API na serveru provést následující body:

1. Načíst zip.
2. Vytvořit složku pro panorama dle struktury.
3. Rozbalit ho do příslušné složky.
4. Importovat upravený *index.js* do Marzipano projektu.

4.6.2 Správa na serveru

Server musí reagovat i na další požadavky jako je:

- Smazat složky.
- Uložit jakékoliv soubory.
- Tvořit soubory (např. JSON).

4.7 Dílčí části

Z předešlých kapitol vyplývá následující rozdělení.

Klientská aplikace:

- Zobrazení panoramat.
- Zobrazení editačního formuláře.
- Rozhraní pro nahrávání zipu.
- Přihlášení do aplikace.

Serverové API:

- Reagovat a provést požadavky vytvořit, rozbalit, vymazat a kopírovat.

5 IMPLEMENTACE

Jelikož je snahou vytvořit aplikaci co nejvíce soběstačnou, nebude se v ní vyžívat databáze, kde by se musely řešit přístupy, hesla a připojení. Tím také odpadá potenciální riziko útoku SQL Injection (viz Kapitola 3.7.1). Bylo však potřebné počítat s dalšími riziky (viz Kapitola 3.7) a psát kód tak, aby nebyl napadnutelný, což se do jisté míry podařilo, protože v aplikaci se $_GET využívá velmi zřídka s parametry a bylo postupováno dle rad diskutovaných ve zmiňované kapitole.

Všechny důležité informace jsou ukládány do JSON struktur, odkud si je aplikace vyčítá. Nutno však říct, že není ošetřena proti neoprávněnému čtení těchto soborů. Toto by si měl pohlídat administrátor, který si pořídí/vytvoří server. Měl by využít například Apache a přes nastavení souboru *.htaccess* nastavit, které složky a soubory by neměly být veřejné [21]. Pokud toto bude provedeno, existuje velmi malá šance, že by se útočník k datům dostal. Jedině v případě napadnutí celého serveru.

Následující kapitola popisuje stěžejní části při vytváření výsledné aplikace. Jedná se o chronologický popis řešení jednotlivých částí.

5.1 Tvoření editačního formuláře

Nejprve bylo nutné vyřešit úpravu hotspotů a to z důvodu, že se jednalo o důležitou funkční část aplikace. Zde se naskytla příležitost pracovat s knihovnou JavaScriptu, díky které lze velmi dobře pracovat s JS/JSON soubory a mnohem víc. Řeč je o knihovně Knockout (viz Kapitola 3.9).

Načítání editačního formuláře je jediná stránka aplikace, která využívá metody GET pro parametry. Je to z důvodu čitelnosti URL, kde uživatel může měnit přímo adresní řádek a přeskočit na editaci jiného projektu. Parametry formuláře jsou názvy složek pro kategorii a projekt. Ošetření proti injektování obsahu (viz Kapitola 3.7.2) je zde vyřešeno tak, že si aplikace po načtení těchto hodnot převádí na formát cesty k souboru a přes funkci file_exists() hledá pomocí těchto parametrů soubor *data.js*. Pokud cesta neexistuje (je v parametru něco co tam být nemá), tak skript končí provádění.

5.1.1 Načítání hotspotů

Z textu o Knockoutu vyplývá, že stačí vytvořit model a knihovna by se měla o dynamickou stránku, včetně vytvoření výsledných dat, postarat sáma.

Jelikož si Marzipano ukládá informace o hotspotech do JS resp. JSON souboru, bylo nutné namapovat potřebné atributy jak do HTML tak do JavaScriptu. Tvořená aplikace pracuje pouze s infoHotspoty, tedy důležité je namapovat hlavně tyto atributy na editační formulář.

```
"infoHotspots": [{
    "yaw": -0.10602694850742722,
    "pitch": -0.5591886750595538,
    "title": "Budova A",
    "extend": true,
    "foto": "office.jpg",
    "targetLink": "www.kancelare.com",
    "targetLinkText": "Ukazat nabidky kancelari",
    "text": ["Kapacita 1000 lidi", "200 kancelari", "Jidelna na kazdem patre"]
}]
```

Obrázek 21 – Ukázka JSON struktury pro infoHotspoty

K mapování v HTML slouží element atribut *data-bind*, který lze vidět v ukázce níže (viz Obrázek 22). Jakmile je element nabindován pomocí své proměnné, stává se hlídanou částí Knockoutu. Je však nutné tento atribut zahrnout v modelu, který se definuje v JavaScriptu jinak knihovna nebude fungovat.

Struktura klíčových bodů (viz Obrázek 21) se mírně liší, než je ukázáno v kapitole o Marzipanu. Objekty disponují více atributy. To je zapříčiněno tím, že se pracuje již s modifikovanou verzí *index.js* Marzipana. Stěžejní je atribut *extend*, který rozšiřuje klíčový bod o další atributy jako je např. *targetLink, foto* a *text*, který v tomto případě není pouze textový řetězec, ale jedná se o pole textových řetězců. Všechny tyto body bylo nutné zahrnout při tvoření modelu pro Knockout.

```html
<div data-bind="foreach: tabs" class="tab-content" id="v-pills-tabContent">
    <div data-bind="attr: { id: contentID, 'aria-labelledby': contentAria }" class="tab-pane fade"
        role="tabpanel">
        <div class="row">
            <div class="form-group col-md-4">
                <input data-bind='textInput:yaw' id="yawInput" class="form-control"
                    placeholder="Yaw" />
            </div>
            <div class="form-group col-md-4">
                <input data-bind='textInput:pitch' id="pitchInput" class="form-control"
                    placeholder="Pitch" />
            </div>
            <div class="form-group col-md-4" id="showPanoButton">
                <a class="btn btn-success" onclick="showPano();" href="#">Změnit souřadnici
                    hotspotu</a>
            </div>
        </div>
        <div class="form-group row col">
            <label for="titleInput">Titulek</label>
            <input data-bind='value: title' id="titleInput" class="form-control" />
        </div>
        <div class="form-group row col" data-bind='ifnot: extend'>
            <label for="pureText">Jednoduchý popisek</label>
            <input data-bind='value: pureText' id="pureText" class="form-control" />
        </div>
```

HTML

JS

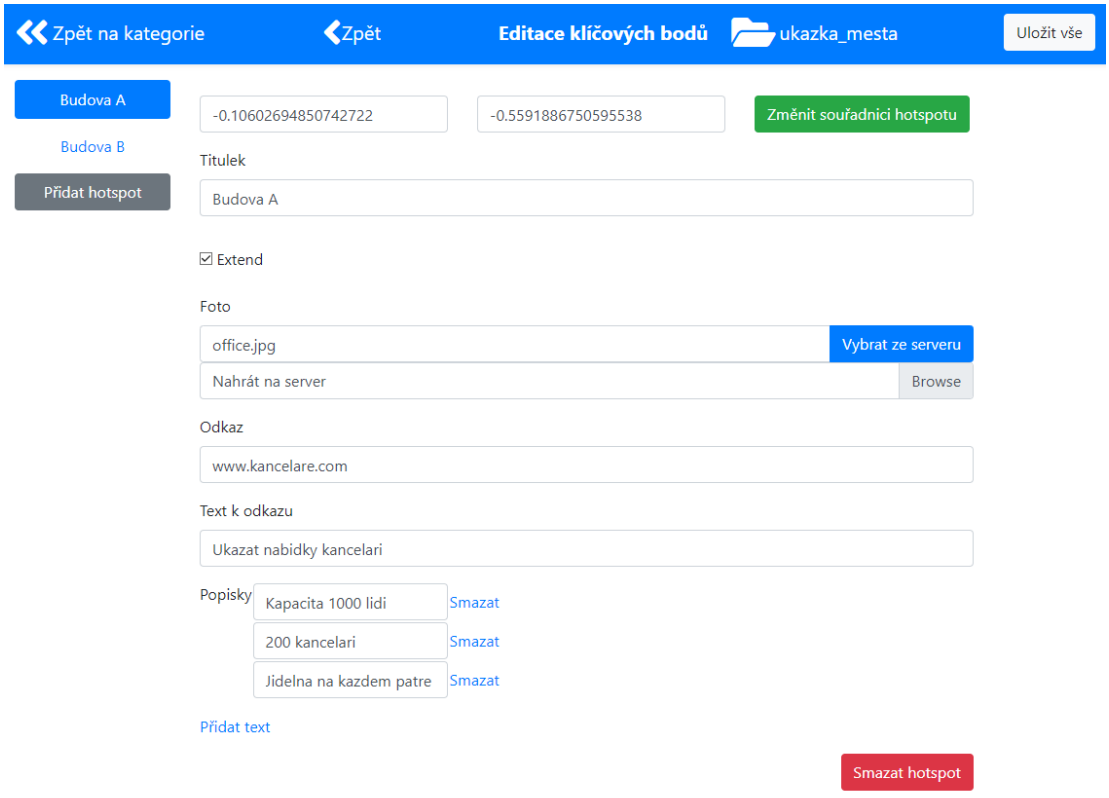
```js
var myTabs = function(tabs) {
    var self = this;

    self.tabs = ko.observableArray(ko.utils.arrayMap(tabs, function(tab) {

    if (!Array.isArray(tab.text)) {
        var text = [];
        text.push(tab.text);
        tab.text = text;
    }

    var observableText = ko.observableArray($.map(tab.text, function(array) {
        return {
            val: ko.observable(array)
        };
    }));

    //there is no need to have specific number for menu, but for generating it must be something
    var generatedNumberForMenu = Math.floor(Math.random() * 10001);

    // use my own name for TAB with generated number
    var generatedText= "generated-" + String(generatedNumberForMenu);

    return {
        //tab generating
        taba: generatedText + "-tab",
        hrefTab: "#" + generatedText,
        ariaControls: generatedText,
        hotspotName: tab.title || "Nový hotspot",
        contentID: generatedText,
        contentAria: generatedText + "-tab",
        //form
        yaw: tab.yaw,
        pitch: tab.pitch,
        title: tab.title,
```

Obrázek 22 – Ukázka knockoutu HTML vs. JS

Nejenže Knockout dokáže načíst z JSON souboru data a vytvořit pro ně webovou stránku, ale umí měnit strukturu webové stránky při běhu. Při tvoření modelu lze vytvořit vazby i pro jednotlivé části webu. To znamená, že lze například přidat dynamicky novou položku menu.

Struktura hotspotů (viz Obrázek 21) je pole objektů, kde má každý objekt své atributy. Původně editační formulář vypadal, jak je popsáno v kapitole 4.5.2. Ukázalo se však, že toto zobrazení není příliš přehledné v případě, že je pro úpravu nadefinováno více atributů. Když se k tomu přidal fakt, že panorama může mít i 10 klíčových bodů, tak formulář působil nepřehledně.

Bylo vytvořeno nové řešení, kde každý objekt v poli je převeden jako položka v menu. Na vzhled a strukturu záložek posloužil Bootstrap a využití jejich tabů a kaskádových stylů. Stačilo pouze dát pozor na strukturu záložek a atributů v jejich elementech. Tedy správně tyto elementy nabindovat a poté je zahrnout do modelu. To lze vidět výše (Obrázek 22), kde je ukázka bindování částí pro menu (jedná se o proměnné taba, hrefTab, contentID atd.).



Obrázek 23 – Ukázka editačního formuláře pro klíčový bod

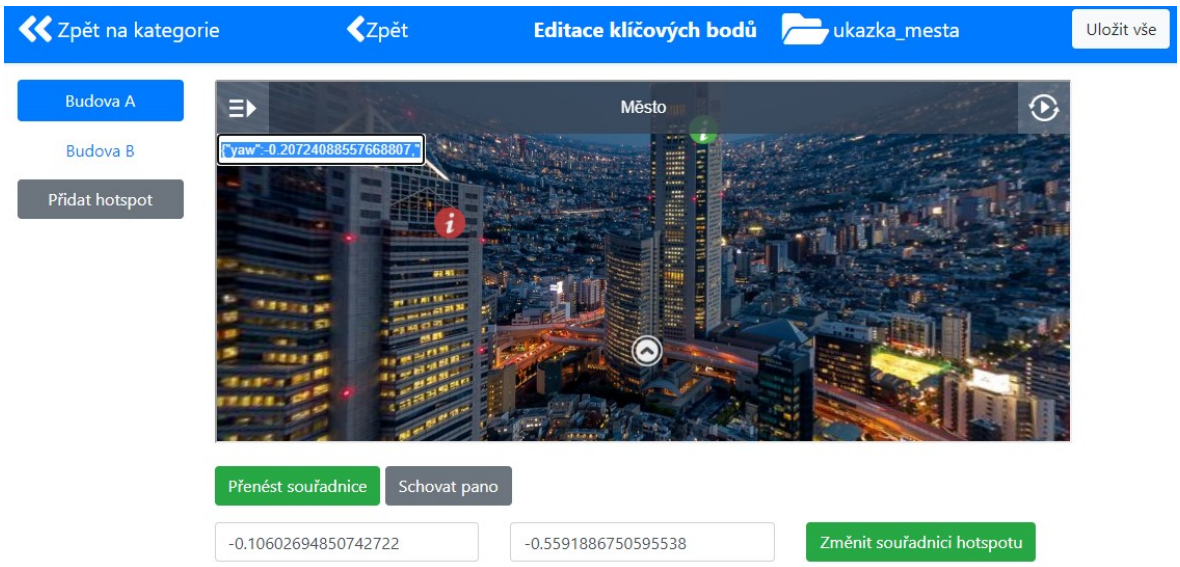
5.1.2 Editace souřadnic

Aby uživatel mohl vybrat svůj vlastní bod, musí dané panorama vidět. Zde je využit rámeček *iframe* (viz Kapitola 3.2.2), který je nasměrován přes atribut *src* na dané panorama. Zbývalo vyřešit přenášení souřadnic z jednoho webu na druhý.

Jelikož je využíván již modifikovaný *index.js* Marzipano projektu, existuje v něm možnost otevřít panorama v tzv. debug módu.

Debugovací funkce funguje tak, že při kliknutí do panoramatu, se odešlou informace o daném místě jak do konzole, tak i do inputu. Tento input je vytvořený JavaScriptem, jehož třída je zadefinována jako *log-div*. Tvořená aplikace s touto třídou počítá.

Uživatel otevře okno s panoramatem a může si vybrat libovolný bod. Ve chvíli, kdy je uživatel rozhodnutý, stačí kliknout na tlačítko přenést souřadnice. Aplikace se podívá do svého rámečku a přes querySelector na *log-div* si najde výstup se souřadnicemi a ty si poté rozděluje do jednotlivých inputů ve formuláři. To, na který objekt přenést tyto souřadnice, je rozlišováno pomocí třídy *active*, která říká, jaká položka menu je zrovna otevřena (tedy ta, která se modifikuje).

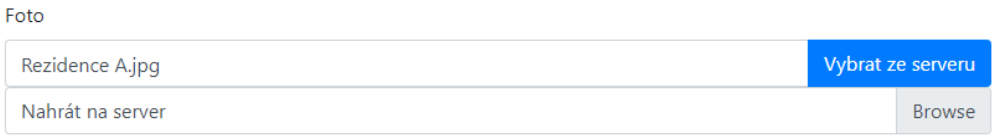


Obrázek 24 – Ukázka editace souřadnic

5.1.3 Editace obrázku

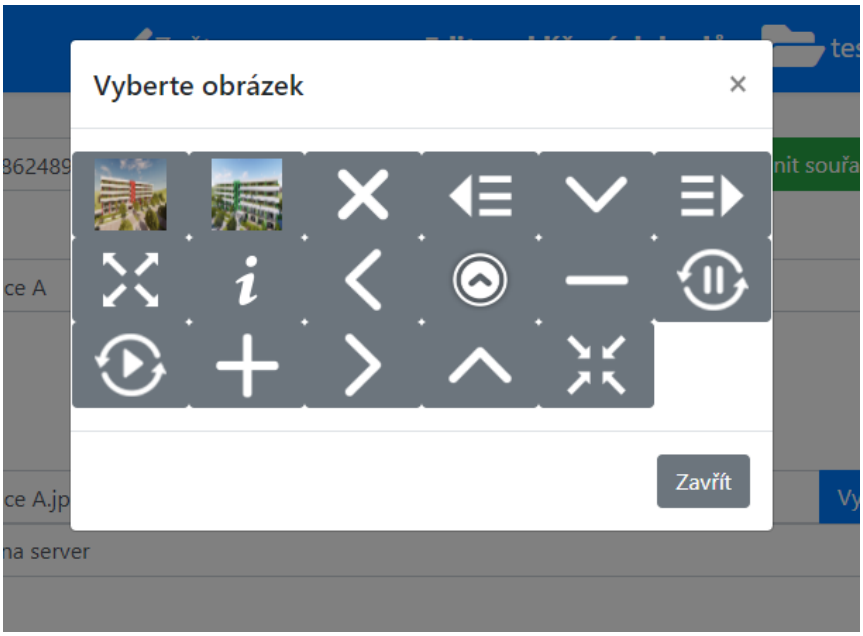
Dalším důležitým bodem v editaci bylo umožnit výběr obrázku pro klíčový bod. Na základě funkčních požadavků bylo nutné vytvořit dvě varianty editace. Buďto vybrat si obrázek

přímo z adresáře *img*, který je na serveru v Marzipano projektu anebo umožnit vložit vlastní obrázek.



Obrázek 25 – Ukázka políčka Foto

Vybrat obrázek, funguje tak, že se při kliknutí na tlačítko volá serverové API (viz Kapitola 5.5.5), kde je metoda, která vrací obsah adresáře *img*, resp. cesty k těmto obrázkům. Otevře se modální okno (viz Obrázek 26), kde je seznam miniatur obrázků. Po kliknutí na miniaturu obrázku se okno zavře a vyplní se příslušný input. Toho je docíleno atributem *onclick* v elementu, který volá vytvořenou funkci selectFoto(). To vezme název fotografie z cesty a vyplní jej do políčka *Foto* ve formuláři.



Obrázek 26 – Výběr obrázku z adresáře img na serveru

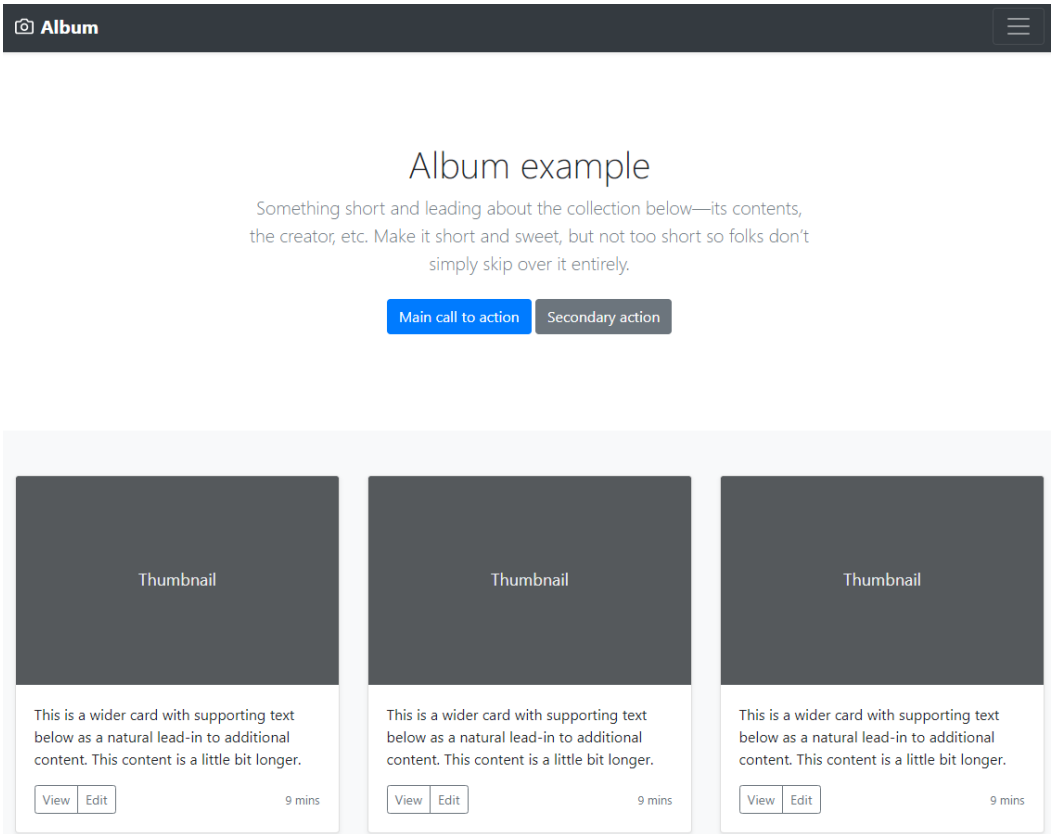
Druhou možností je nahrát obrázek přímo. Ve chvíli, kdy uživatel provádí tuto variantu a vybere si obrázek ze svého počítače, aplikace okamžitě nahraje obrázek na server a plní input názvem zadaného obrázku. Je to z důvodu, že ve chvíli, kdy se ukládá samotný JSON s nastavením, tak obrázek už v něm musí být nadefinovaný a existovat na serveru.

5.1.4 Uložení formuláře

Samotné ukládání funguje tak, že ve chvíli, kdy se změní jakýkoliv input, tak Knockout o tom ví a tím pádem je připraven vytvořit nový JSON objekt. To znamená, že když uživatel klikne na uložení editačního formuláře, tak knihovna vygeneruje již upravenou JSON strukturu. S touto strukturou pak klientská aplikace posílá požadavek na API (viz Kapitola 5.5.3), která přepíše soubor *data.js*.

5.2 Prohlížení panoramat a práce s nimi

Bylo využito šablony z Bootstrapu, která přesně odpovídá návrhu. Jedná se o příklad galerie resp. alba fotografií na webu. Na všechny ikony, které jdou vidět v aplikaci, bylo využito Font Awesome (viz Kapitola 3.10).



Obrázek 27 – Ukázka příkladu ALBUM v bootstrapu

Využívaná šablona se jmenuje Album (viz Obrázek 27) a posloužila jako základ pro zobrazení panoramat. Tuto šablonu stačí stáhnout na lokální počítač a změnit cesty k Bootstrap knihovnám, které jsou potřeba také stáhnout.

5.2.1 Zobrazení panoramat

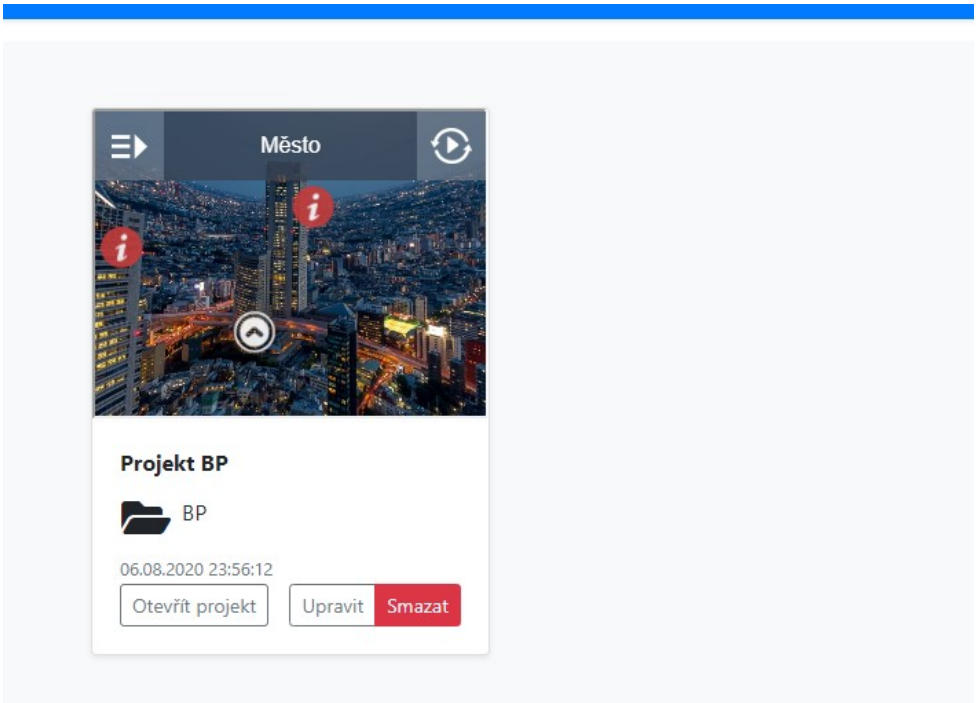
Pro zobrazení panoramat jsou využity získané informace o Knockoutu a jeho práci s JSON soubory.

Album z Bootstrapu využívá tzv. karty (viz Obrázek 27). Stačí si uložit potřebné informace ke kartě do JSON souboru, nabindovat jednotlivé elementy v HTML pomocí atributu *data-bind* a knihovna by měla na základě vytvořeného modelu vykreslit všechny karty podle zadaného souboru.

Celý proces vypadá tak, že v klientské aplikaci je vytvořený input, do kterého uživatel nahraje zip s panoramatem a po kliknutí na uložit se odesílá požadavek na serverové API (viz Kapitola 5.5.2). Jakmile je operace dokončena, Knockout na to reaguje a vykresluje ze svého JSON souboru kartu na hlavní straně.

Dalším bodem bylo vyřešit ukázku panoramatu v kartě. Původní myšlenka udělat pouze náhled ve formě obrázku, je na konec vyřešen přes rámeček *iframe* (viz Kapitola 3.2.2).

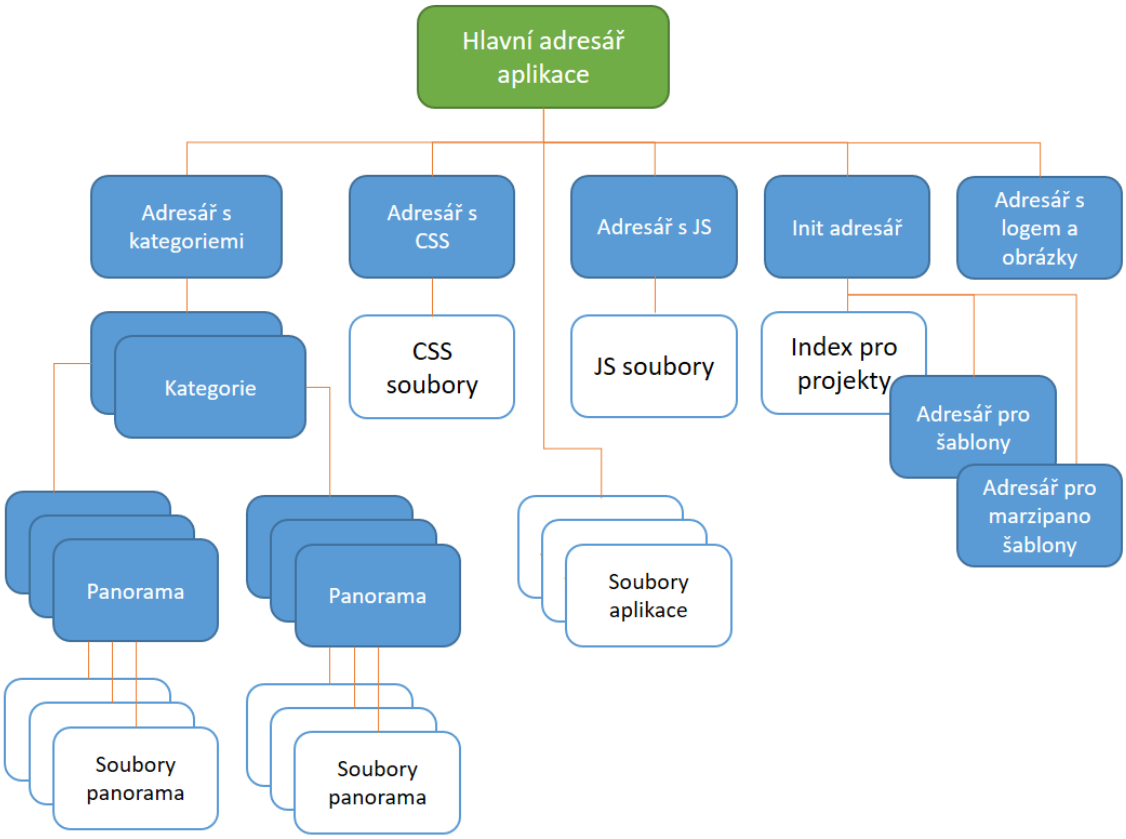
Rámečky by mohly být trochu náročnější pro prohlížeč, protože v tomto případě musí načíst více webových stránek resp. panoramat najednou. Nese to sebou výhodu, že uživatel si může prohlédnout panorama okamžitě bez nutnosti otevření (i když v malém rozlišení). Nevýhodou je zmiňovaná náročnost, ale pro potřeby této práce je řešení dostačující. Lze však *iframe* v budoucnu předělat na obyčejnou fotografii panoramatu.



Obrázek 28 – Ukázka nahraného panoramatu v aplikaci

5.3 Tvoření kategorií

Původní návrh struktury aplikace z kapitoly 4.3 bylo nutné pozměnit, protože v průběhu práce vznikl nový požadavek a to mít seskupení panoramat, které mají něco společného (např. panoramata památek v jednom konkrétním městě). Navíc na základě testování se ukázalo, že příliš mnoho panoramat, které mezi sebou nemají žádný vztah, nedávalo smysl. Výsledkem bylo posunutí hlavní strany s panoramaty o úroveň níže a nad nimi tvořit složky (viz Obrázek 29).



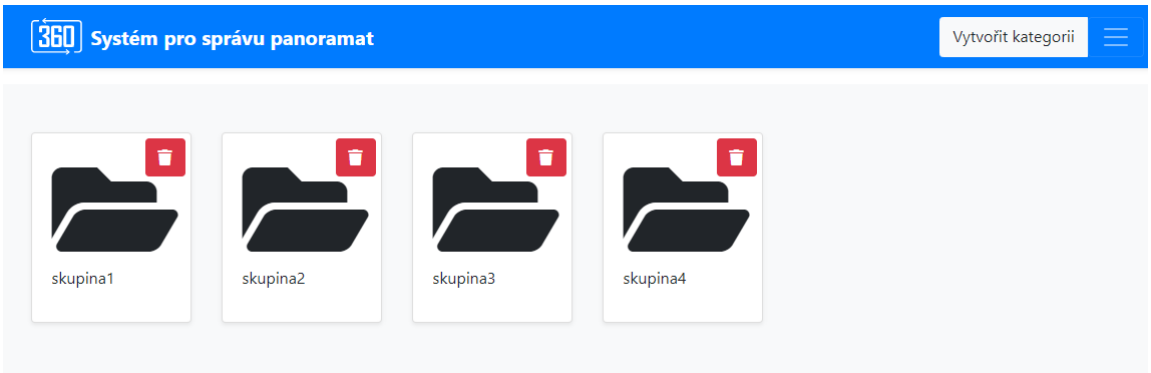
Obrázek 29 – Výsledná struktura aplikace

Kategorie využívají také Knockout a princip je velmi obdobný jako u zobrazování panoramat, resp. karet (viz Kapitola 5.2.1). Kategorie je nyní hlavní strana, kterou uživatel uvidí po přihlášení. Index je naplněn kartami, které využívají ikonu složky *fa-folder-open* z Font Awesome (viz Kapitola 3.10). Jednotlivé karty využívají atribut *href* s odkazem na hlavní stranu kategorie a třídu *stretched-link*, díky které se karta stává odkazem. Kamkoliv uživatel na kartě klikne, je přesměrován na příslušné místo.

Pro založení kategorie stačí odeslat požadavek na serverové API (viz Kapitola 5.5.6) a to danou složku vytvoří a do svého JSON souboru (viz Obrázek 30) si poznačí, že vytvořilo novou kategorii a také cestu k této kategorii (právě pro otevírání přes element atribut *href*).


```
[
    {
        "categoryName":"skupina1",
        "pathToCategory":"categories\/skupina1\/"
    },
    {
        "categoryName":"skupina2",
        "pathToCategory":"categories\/skupina2\/"
    },
    {
        "categoryName":"skupina3",
        "pathToCategory":"categories\/skupina3"
    },
    {
        "categoryName":"skupina4",
        "pathToCategory":"categories\/skupina4"
    }
]
```


Obrázek 30 – Ukázka značení kategorií v aplikaci



Obrázek 31 – Ukázka zobrazení kategorií v aplikaci

Každá kategorie v tomto případě potřebuje svoji hlavní stranu. To znamená, že je potřeba nadefinovat index pro každou kategorii. To bylo vyřešeno složkou *init*, která obsahuje soubor s kódem pro zobrazování panoramat a každá kategorie se na tento soubor odkazuje. Všechny hlavní stránky kategorie jsou tedy totožné, liší se pouze jejich obsah a ten je vyčítán z JSON souborů. Každá kategorie (složka) si uchovává informace o svých panoramatech a stejně tak si aplikace uchovává informace o svých kategoriích.

```
    <script>
        $(function() {
            $('#index').load("../../init/allProjectPage.php");
        });
    </script>
</head>


<div id="index">
</div>


</html>
```

Obrázek 32 – Ukázka indexu pro kategorii

5.4 Přihlášení do aplikace

Přihlašování nemá svůj návrh. Jedná se o klasické přihlášení, jak jej každý dnes zná. Uživatel zadá jméno s heslem a přihlásí se. Jelikož ale není využívána databáze, bylo potřeba vytvořit strukturu, ve které budou uloženi uživatelé. I zde je využíván JSON soubor, který si uchovává informace o uživatelích.

Tento soubor obsahuje zašifrované heslo pomocí funkce password_hash() [14]. Jedná se o jednosměrné šifrování, tedy není možné toto heslo odšifrovat zpět. Při přihlášení se kontroluje pomocí password_verify() [14], zda se zadané heslo rovná zašifrovanému heslu v JSON souboru. Pokud ano, tak se do vytvářené relace nastaví název uživatele, informace o uživateli (pokud je vyplněná v hlavičce – zpravidla to jsou informace o prohlížeči a systému [14]) a také IP adresa. Tyto informace jsou zde vkládány jako možná prevence proti hijackingu (viz Kapitola 3.7.6). Tvořená relace je tedy omezena na daný počítač.

```
if($val->username == $_POST["username"] && password_verify($_POST["password"], $val->password))
    {
        //save user, his IP and his user agent
        $_SESSION['user_id'] = $_POST["username"];
        $_SESSION['ip_address'] = $_SERVER['REMOTE_ADDR'];
        $_SESSION['user_agent'] = (isset($_SERVER['HTTP_USER_AGENT'])) ? $_SERVER['HTTP_USER_AGENT'] : '';
        header("Location: index.php");
    }
```

Obrázek 33 – Ukázka nastavení relace

Jakmile se uživatel přihlásí, ID relace je přegenerováno pomocí funkce session_regenerate_id() [14]. To slouží jako ochrana proti útoku session fixation (viz Kapitola 3.7.5).

Pro správný chod bylo nutné upravit všechny soubory ze souborového typu *.html* na *.php*. Na začátku každého skriptu pak stačilo kontrolovat, zda je relace aktivní a s jakými parametry. Pokud jsou všechny parametry při načítání stránky stejné jako v již vytvořené relaci, tak jsou načteny webové stránky. V opačném případě je uživatel přesměrován zpět na stránku s loginem.

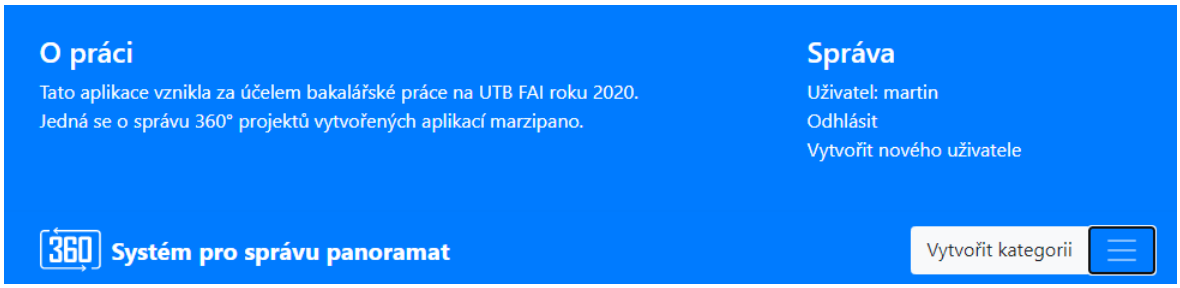
```
if(!isset($_SESSION['user_id']) ||
    $_SESSION['ip_address'] !== $_SERVER['REMOTE_ADDR'] ||
        ($_SESSION['user_agent'] !== $_SERVER['HTTP_USER_AGENT'])){
   session_destroy();
   header("location:login.php");
   die();
}
```

Obrázek 34 – Ukázka kontroly relace

Vzhled (viz Obrázek 35) byl vyřešen opět šablonou Bootstrapu, která se jmenuje sign-in. Jedná se o holou šablonu, kterou bylo potřeba mírně doladit. Obrázek na pozadí, který je zde využíván, je ze stránek unsplash [11], což je web, který poskytuje fotografie zdarma pro komerční využití. Logo je vytvořeno autorem této práce a uloženo v SVG formátu.

Obrázek 35 – Ukázka přihlášení

S loginem bylo nutné přidat do aplikace menu pro odhlášení s případnou informací, jaký uživatel je zrovna přihlášen.



Obrázek 36 – Ukázka informačního okna s možností odhlásit

Po kliknutí na ikonu (tři čárky vedle tlačítka Vytvořit kategorii) v hlavní liště se zobrazí okno, které obsahuje potřebné informace, včetně odhlášení (viz Obrázek 36). Zobrazení uživatele je napsáno pomocí funkce echo(), která vypíše jméno z relace. Pro jistotu je zde použita funkce htmlentities() z kapitoly o bezpečnosti webu (viz Kapitola 3.7.3). Odkaz Vytvořit nového uživatele využívá atribut *onclick*, který zavolá jednoduchou funkci a ta už odesílá pomocí POST požadavek na API (viz Kapitola 5.5.8). Stejně je řešeno i odhlášení (viz ukázka Obrázek 37).

```
<div class="col-sm-4 offset-md-1 py-4">
  <h4 class="text-white">Správa</h4>
  <ul class="list-unstyled">
    <?php
        echo "<li><a class=\"text-white\">Uživatel: ".htmlentities($_SESSION['user_id'])."</a></li>"
    ?>
    <li><a href="#" class="text-white" onclick="signOutUser();">Odhlásit</a></li>
    <li><a href="#" class="text-white" onclick="newUser();">Vytvořit nového uživatele</a></li>
  </ul>
```

Obrázek 37 – Ukázka HTML/PHP zápisu pro správu v aplikaci

```
[
    {
        "username":"martin",
        "password":"$2y$10$qrUwTcozxTPFsBwtIpPtW.FOtWrClhj\/rNAvFaO07dSXGF.tBIZY2"
    },
    {
        "username":"administrator",
        "password":"$2y$10$XnFpG4fEWEBlUvvp7WomvOliGorrN8smqUFQ1MN9j2aMIWpTdVp36"
    }
]
```

Obrázek 38 – Ukázka šifrování v JSON struktuře

5.5 API a jeho volání

API má za úkol vyřizovat všechny požadavky, které klientská aplikace v danou chvíli potřebuje. Například vytváření složky nebo ukládání souborů. V následující kapitole jsou popsány základní metody, které serverové API má a to jakým způsobem jsou požadavky volány.

5.5.1 Volání metody z klientské části

Klientská aplikace musí odeslat požadavek. Toho lze docílit JavaScript funkcí XMLHttpRequest (viz Kapitola 3.4.1). Po vyplnění všech potřebných parametrů se odesílá požadavek na server. V aplikaci jsou rozlišené dvoje volání a to jednou s handlerem *readyonstatechange* [12] a jednou bez něj. Je to callback, který dokáže zpětně říct, jestli při operaci na serveru nastala chyba nebo ne.

Při využití *readyonstatechange* (viz Obrázek 39), je vytvořena funkce, která čeká na odpověď od serveru. Pokud server vrátí kód od 200 do 400, tak celá operace proběhla v pořádku a není potřeba, aby klient zasahoval. Pro informaci se vypisuje do logu konzole prohlížeče, co API vrátilo za zprávu. V opačném případě nastal v průběhu provádění operací na serveru nějaký problém a pomocí funkce alert() danou zprávu od serveru vypíše. Volání bez *readystatechange* vypadá stejně. Rozdíl je v tom, že neobsahuje zmiňovaný handler. Místo toho je v tomto případě spoléháno na completeHandler, který funguje jako naslouchač (listener) a jen aplikaci ohlásí, že operace je u konce. Tedy nekontroluje průběh provádění.

```
var formdata = new FormData();
formdata.append("projectName", projectName);
formdata.append("zip_file", file);
formdata.append("category", getCategory());
var ajax = new XMLHttpRequest();
ajax.addEventListener("load", completeHandler, false);
ajax.open("POST", "../../api.php?method=storeZip");
ajax.onreadystatechange = function() {
    if (ajax.readyState === XMLHttpRequest.DONE) {
        var status = ajax.status;
        if (status === 0 || (status >= 200 && status < 400)) {
            // The request has been completed successfully
            console.log(ajax.responseText);
        } else {
            $(".overlay").addClass("d-none");
            //There has been an error with the request! Calling the alert!
            alert(ajax.responseText);
        }
    }
};

ajax.send(formdata);
```

Obrázek 39 – Ukázka volání API z klientské aplikace

5.5.2 storeZip

Tato metoda je volána z hlavní strany vybrané kategorie a má za úkol zpracovat daný zip. Nejprve si zjistí, v jaké kategorii se uživatel nachází a pro tuto kategorii projekt i vytvoří. Nejprve je nutné se posunout v rámci serveru na správnou souborovou strukturu. Tedy pomocí php funkce chdir() se změní cesta do správné kategorie a následně se vytváří nová instance třídy ZipArchive(), která obsahuje funkce open() a extractTo(). První zmiňovaná funkce daný zip otevře a druhá jej extrahuje. Po těchto funkcích se volá vytvořená funkce createProject(), která si přichystá data o novém projektu, načte si svůj JSON soubor a vkládá do něj nový záznam. Při extrahování bylo nutné pohlídat, že uživatel může nahrávat jak svůj

upravený Marzipano projekt, tak nijak neupravený. Jak bylo zmíněno v kapitole o Marzipanu (viz Kapitola 1.3.1), samotný projekt je uložen ve složce *app-files*. Metoda počítá s oběma variantami. Pokud zip obsahuje zmiňovanou složku, tak se extrahuje její obsah. V opačném případě se extrahuje celý zip. Na konec metoda kopíruje *index.js* do nově vloženého projektu spolu se *style.css*. Zde je mírná změna oproti požadavku a návrhu, kde bylo napsáno, že se bude kopírovat pouze *index.js*. V případě, kdy se upraví struktura pro hotspot, musí se také přenést i vzhled pro tyto úpravy.

5.5.3 saveJSON

Je volána z editačního formuláře. Metoda již obdrží modifikovaný JSON z editačního formuláře. Stačí pouze načíst soubor *data.js*, ve kterém je uloženo nastavení panoramatu a přepsat jeho obsah. Zde bylo nutné si pohlídat proměnnou v *data.js*, protože tento soubor není typu JSON, obsahuje pouze takovou notaci. V konečné fázi bylo nutné objekt, který přišel v metodě POST, upravit zpět do proměnné.

5.5.4 deleteProject

Metoda je volána z hlavní strany vybrané kategorie. Vymaže záznam z JSON souboru a poté vymaže samotnou složku a všechen její obsah pomocí funkce rrmdir().

5.5.5 getImages

Metoda je volána z editačního fomuláře. Má za úkol vrátit cesty ke všem obrázkům dle kategorie a projektu. Na to slouží php funkce scandir(), která vrací obsahy adresářů.

5.5.6 createCategory

Metoda je volána nad seznamem kategorií. Nejprve si zjistí, jestli zadaná složka neexistuje a poté vytvoří záznam do JSON souboru a současně vytvoří složku na serveru. Nakopíruje do ní šablonu indexu z *init* složky, která odkazuje na hlavní stránku s projekty. Je zde také volání vytvořené funkce, kde se provádí kontroly pomocí regulárního výrazu, že název složky neobsahuje speciální znaky, diakritiku ani mezery.

5.5.7 deleteCategory

Tato metoda je volána nad složkou konkrétní kategorie. Do metody dorazí informace, kterou složku je potřeba smazat. Funkce si složku najde a smaže pomocí funkce rrmdir(). Na závěr si odebere záznam o složce ze svého JSON souboru.

5.5.8 createNewUser

Požadavek na vytvoření nového uživatele. Odeslané heslo se zašifruje pomocí PHP funkce password_hash() a vytvoří záznam o uživateli.

5.5.9 signOut

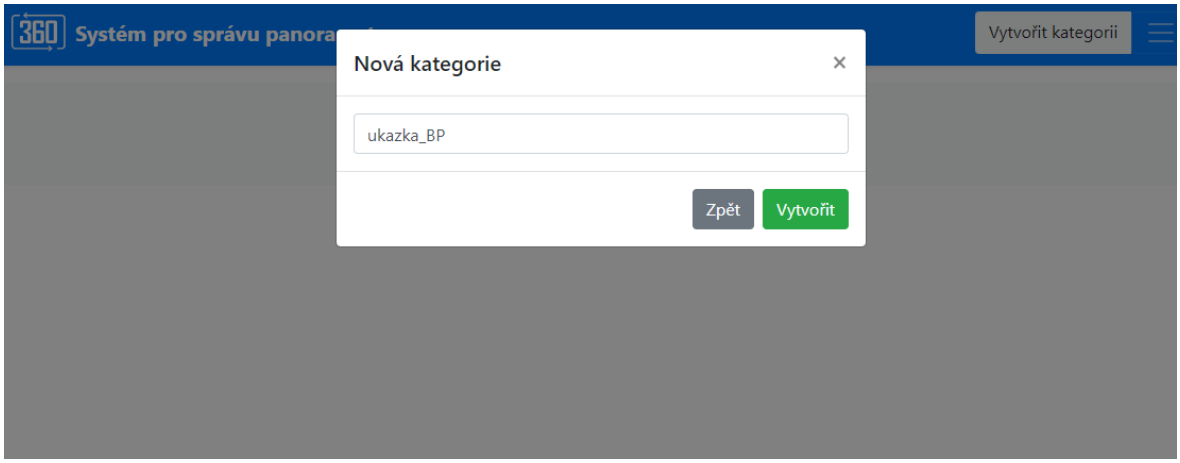
Požadavek na odhlášení zruší relaci pomocí PHP funkce session_destroy().

5.6 Výsledná aplikace

Následující část prezentuje hotovou aplikaci s popisem, jakou funkcionalitu aplikace obsahuje a jaké požadavky byly splněny.

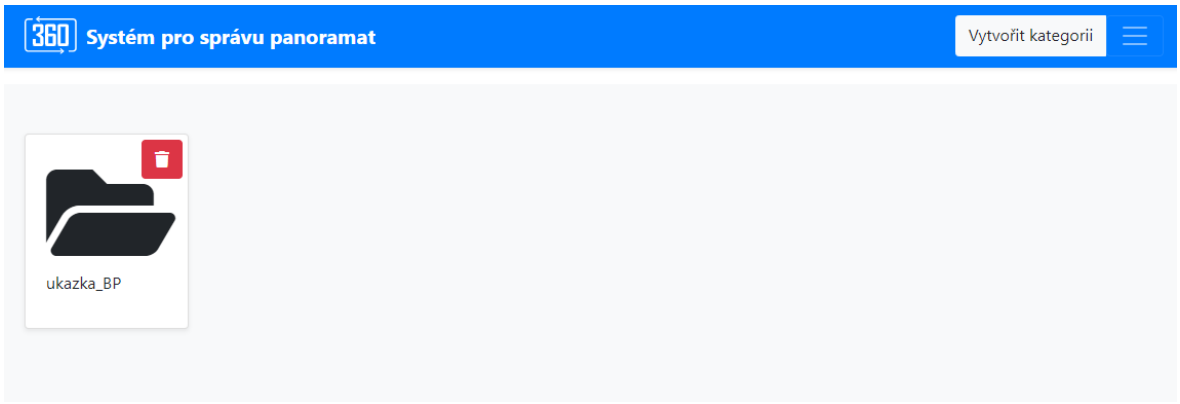
Po přihlášení se uživatel dostane na hlavní stranu, kde je seznam kategorií. Z hlavní lišty má možnost otevřít správu, kde může vytvořit nového uživatele, případně se odhlásit (viz Obrázek 36).

Po kliknutí na **Vytvořit kategorii** se otevře okno (viz Obrázek 40), ve kterém uživatel musí zadat název složky. Pokud složka s tímto názvem existuje, prohlížeč na to upozorní přes alert okno, kde je napsána zpráva, že jméno již existuje. Také jsou zde kontroly, které neumožnují uložit složku s diakritikou, speciálním znakem nebo mezerou. Je to z důvodu požadavku na možnost přepisovat adresní řádek. Očekává se, že uživatel zná svou strukturu a ví, kde má které panorama uložené.



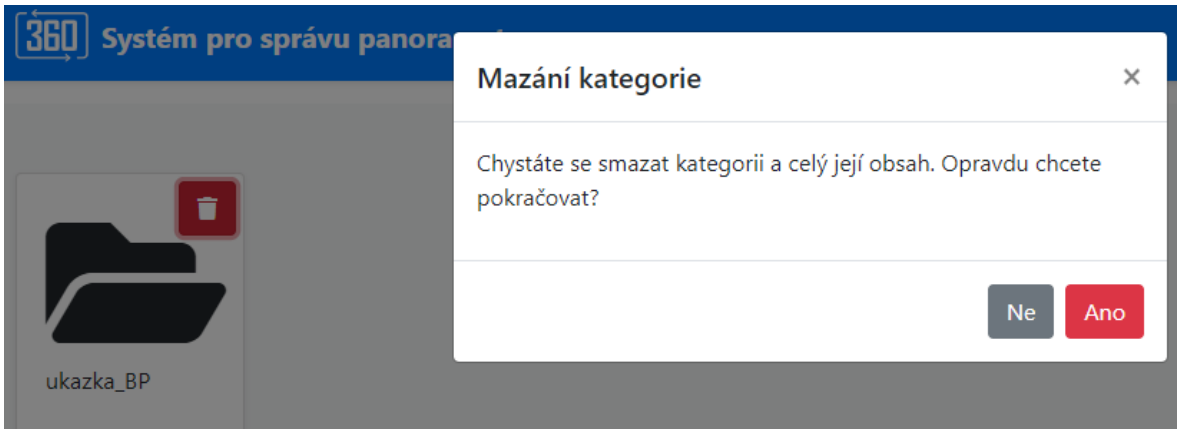
Obrázek 40 – Okno pro vytvoření nové kategorie

Po vytvoření složky, resp. kategorie, ji aplikace zobrazí (viz Obrázek 41).



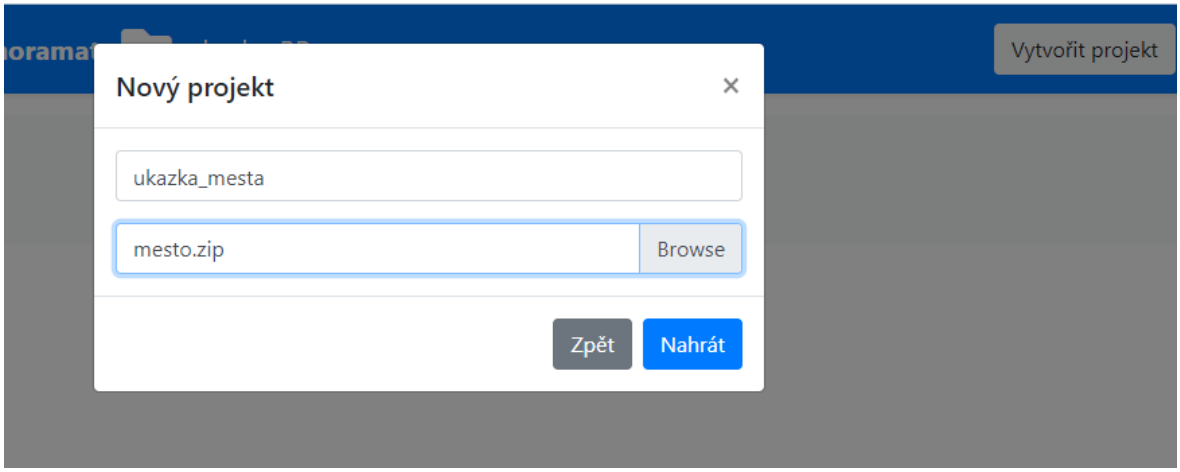
Obrázek 41 – Ukázka kategorií

Uživatel má možnost jakoukoliv složku v aplikaci smazat, pomocí červeného tlačítka. Každé mazání má potvrzovací okno (viz Obrázek 42).



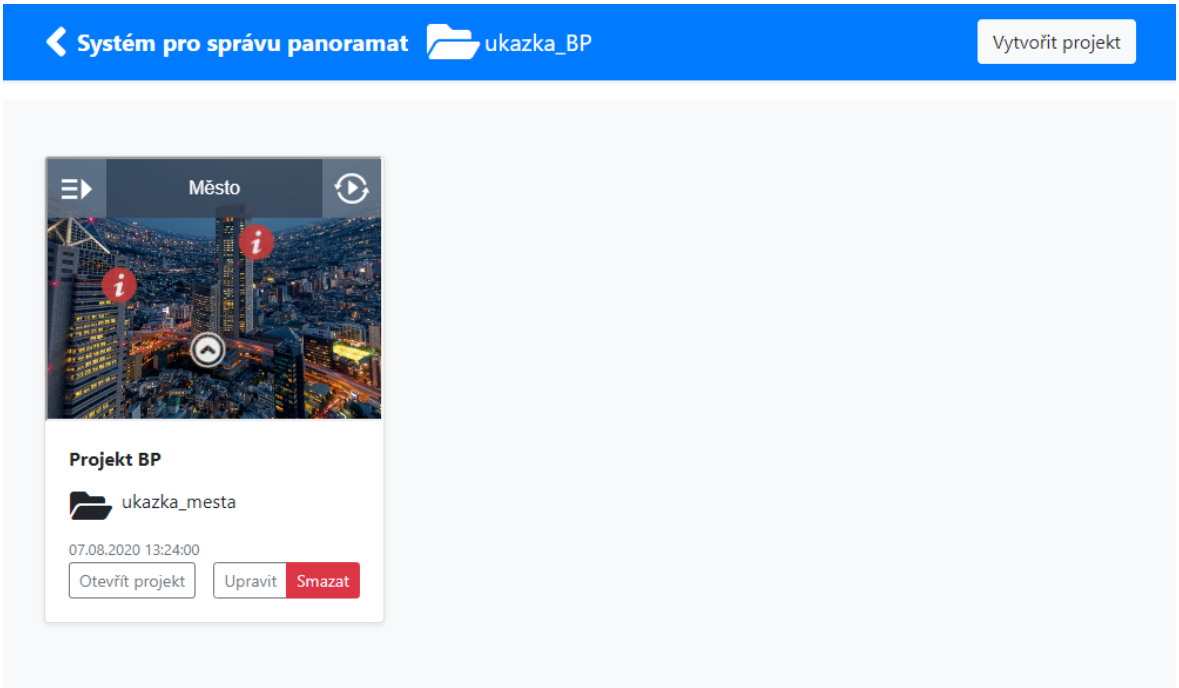
Obrázek 42 – Potvrzovací okno pro smazání

Po kliknutí na danou kategorii se otevírá strana projektů, kde je tlačítko **Vytvořit projekt**. To otevírá okno (viz Obrázek 43), kde uživatel opět vyplňuje název složky, ale tentokrát je tam políčko i pro nahrání zipu na server.



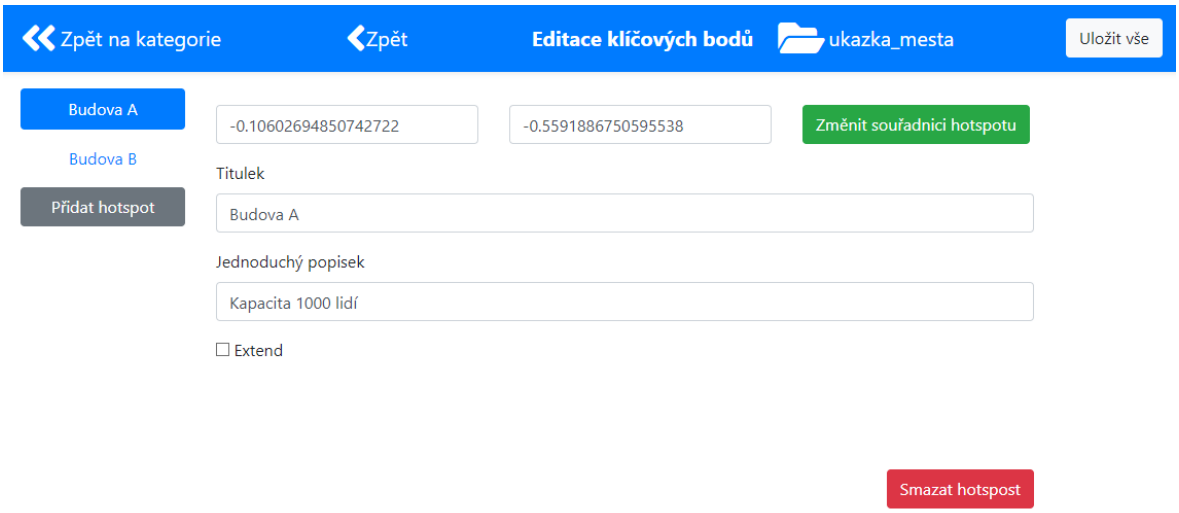
Obrázek 43 – Okno pro vytvoření projektu

Po kliknutí na **Nahrát** se aplikace zatmaví s loading ikonou a probíhají operace na serveru. Po načtení se na stránce objeví nahrávaný projekt (viz Obrázek 44). Uživatel na této stránce má možnost vrátit se zpět přes ikonu šipka vlevo. Webová stránka mu také poskytuje informaci o tom, v jaké složce se právě nachází – ikona složky v modré liště. Uživatel na této stránce může otevřít daný projekt – přesměrování v nové záložce na panorama. Upravit klíčové body nebo projekt smazat.



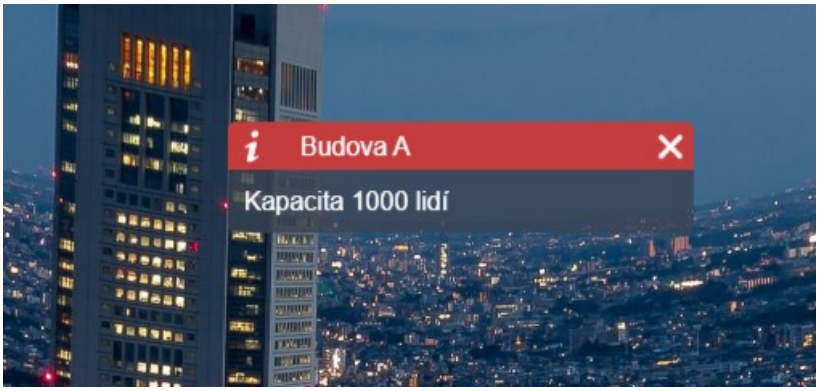
Obrázek 44 – Ukázka hlavní strany projektů

Při editaci klíčových bodů se otevře editační formulář (viz Obrázek 45), který obsahuje položky menu jako jednotlivé hotspoty. V hlavní liště je možné se pomocí tlačítek vrátit na kategorie anebo zpět na projekty. Nechybí ani ukazatel, který projekt se zrovna modifikuje. Formulář nekomunikuje ihned se serverem. Výsledná komunikace se serverem probíhá až po kliknutí na tlačítko **Uložit vše**. Toto je rozdíl oproti návrhu, kde se očekávalo, že na každou změnu (mazání či přidání hotspotu) se bude odesílat požadavek na serverové API. Jakékoliv změny uživatel při editaci udělá, jsou po znovunačtení stránky nenávratně ztraceny.



Obrázek 45 – Ukázka editace klíčového bodu Budova A

Uživatel má možnost rozšířit přes checkbox *extend* základní bod o další položky anebo vytvořit zcela nový klíčový bod tlačítkem **Přidat hotspot**. Při jediné editaci je možné jich přidat nespočet. To je výhoda JavaScriptu a Knockoutu (podrobnější ukázky editace včetně vysvětlení lze vidět v Kapitole 5.1.2 a Kapitole 5.1.3).



Obrázek 46 – Základní klíčový bod před editací

Obrázek 47 – Rozšířený klíčový bod po editaci

ZÁVĚR

Cílem této práce bylo vytvořit aplikaci, která by mohla pracovat s panoramatickými snímky a následné popsání průběhu zvoleného řešení. Na internetu bylo nalezeno spousty aplikací pro vytváření panoramatických snímků. Vytvořit aplikaci, která by byla natolik obecná a mohla pracovat se všemi tipy, nelze. Ukázalo se, že jedinou variantou, jak docílit řešení, bylo pracovat s open source aplikací Marzipano.

V rámci teoretické části byly popsány stávající systémy. Následně byly prostudovány technologie a bezpečnost včetně shrnutí, co jsou to požadavky a jak se tvoří. I když oficiální zadání neobsahovalo bezpečnost webu, v práci jsou stručně popsány možné útoky na web a jak se proti nim bránit. Tím byl sestaven teoretický rámec pro možný návrh aplikace a jejího vývoje.

Návrh obsahuje vydefinování všech požadavků, které jsou tvořeny autorem a vedoucím této práce. Všechny tyto požadavky byly ve výsledné aplikaci splněny a také přibyly i další funkčnosti, jako je tvoření kategorií.

Při vývoji byl nejprve vytvořen editační formulář pomocí knihovny Knockout a na základě těchto informací byla knihovna použita na celou aplikaci. Následovalo tvoření hlavní strany a rozhraní pro nahrávání zipu. Právě v tomto bodě přišla změna, že hlavní stránka nebudou projekty, ale kategorie. Současně s vývojem na klientské aplikaci se tvořilo serverové API, které vždy reagovalo na tvořenou část. Poslední věc, která byla na aplikaci vyvíjena, bylo přihlašování neboli login. Bezpečnost webu je velmi obsáhlé a složité téma, avšak i v této aplikaci byla snaha o bezpečnost. Tvořený login je dočasný a bude potřebovat více péče, nicméně věřím, že zabezpečení je dostatečné alespoň proti botům a základním útokům. Závěrečná část práce se věnuje výsledné aplikaci, tedy jak funguje a jaký má design.

SEZNAM POUŽITÉ LITERATURY

[1] GASSTON, Peter. Moderní web. Přeložil Ondřej BAŠE. Brno: Computer Press, 2015. ISBN 978-80-251-4345-2.

[2] WIDIYANINGTYAS, Triyanna, Didik Dwi PRASETYA a Aji P WIBAWA. Web-based Campus Virtual Tour Application using ORB Image Stitching. In: 2018 5th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI): 2018 5th s. 46?49. ISSN null. Dostupné z: doi:10.1109/EECSI.2018.8752709.

[3] SKLAR, David. PHP 7: praktický průvodce nejrozšířenějším skriptovacím jazykem pro web. Přeložil Jan POKORNÝ. Brno: Zoner Press, 2018. Encyklopedie Zoner Press. ISBN 978-80-741-3363-3.

[4] ISO/IEC 21778:2017. Information technology — The JSON data interchange syntax. Mezinárodní technická norma, 2017.

[5] LAURENČÍK, Marek. Tvorba www stránek v HTML a CSS. Praha: Grada Publishing, 2019. Průvodce (Grada). ISBN 978-80-271-2241-7.

[6] WWW stránky: Krpano Panorama Viewer [online]. URL https://krpano.com/docu/xml/

[7] WWW stránky: Pano2VR [online]. URL: https://ggnome.com/doc/index/

[8] WWW stránky: Marzipano [online]. URL: https://www.marzipano.net

[9] WWW stránky: Bootstrap [online]. URL: https://getbootstrap.com

[10] WWW stránky: Knockout [online]. URL: https://knockoutjs.com/index.html

[11] WWW stránky: Unsplash [online]. URL: https://unsplash.com

[12] WWW stránky: The Modern JavaScript Tutorial [online]. URL: https://javascript.info/xmlhttprequest

[13] WWW stránky: Font Awesome [online]. URL: https://fontawesome.com/v4.7.0/

[14] WWW stránky: php [online]. URL: https://www.php.net/docs.php

[15] ACUNETIX [online]. PHP Security 1-5.
URL: https://www.acunetix.com/websitesecurity/php-security-1/

[16] J. HASAN, A. M. ZEKI. Evaluation of web application session security, 2nd Smart Cities Symposium (SCS 2019). Bahrain. 2019. Dostupné z:doi: 10.1049/cp.2019.0178.

[17] KOLŠEK, Mitja. Session Fixation Vulnerability in Web-based Applications. 2007. Dostupné z: http://www.acrossecurity.com/papers/session_fixation.pdf

[18] WWW stránky: OWASP - Session Management Cheat Sheet [online]. URL: https://cheatsheetseries.owasp.org/cheatsheets/Session_Management_Cheat_Sheet.html

[19] GORTON, Ian. Essential Software Architecture. Berlin: Springer, 2016. ISBN 978-3-642-19175-6.

[20] IEEE Recommended Practice for Software Requirements Specifications," in IEEE Std 830-1998 , vol., no., pp.1-40, 20 Oct. 1998, Dostupné z:doi: 10.1109/IEEESTD.1998.88286.

[21] WWW stránky: APACHE [online].

URL: https://httpd.apache.org/docs/2.4/misc/security_tips.html

SEZNAM OBRÁZKŮ