FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání bakalářské práce

Název:	Starý Most - webový systém pro správu dat
Student:	Michal Pražák
Vedoucí:	Ing. Jiří Chludil
Studijní program:	Informatika
Obor / specializace:	Webové a softwarové inženýrství, zaměření Softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	do konce letního semestru 2024/2025

Pokyny pro vypracování

1. Analyzujte vybrané práce z projektu VMČK a navazujícího Starý Most.
2. Analyzujte nové funkční a nefunkční požadavky na administrační rozhraní a backend.
3. Prozkoumejte možnosti aktualizace použitých balíčků a technologií ve webové aplikaci a backendu.
4. Aktualizujte použité balíčky a technologie podle identifikovaných možností.
5. Navrhněte úpravy stávající webové aplikace a API, které odpovídají identifikovaným požadavkům.
6. Rozšiřte stávající webovou aplikaci a API o navrhované změny.
7. Výsledek podrobte testům (integrační, akceptační, uživatelské)

Elektronicky schválil/a Ing. Michal Valenta, Ph.D. dne 30. ledna 2024 v Praze.

Bakalářská práce

STARÝ MOST – WEBOVÝ SYSTÉM PRO SPRÁVU DAT

Michal Pražák

Fakulta informačních technologií
Katedra softwarového inženýrství
Vedoucí: Ing. Jiří Chludil
16. května 2024

Obsah

Seznam obrázků

Seznam tabulek

Seznam výpisů kódu

Chtěl bych poděkovat především vedoucímu práce Ing. Jiřímu Chludilovi za jeho rady a pomoc při tvorbě této práce. Dále bych chtěl poděkovat své rodině a kamarádům, kteří mě při psaní podporovali a motivovali.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.
Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů, zejména skutečnost, že České vysoké učení technické v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle §60 odst. 1 citovaného zákona.

V Praze dne 16. května 2024

Abstrakt

Tato bakalářská práce se zabývá návrhem a implementací webového systému pro správu dat projektu Starý Most. Tento systém umožňuje správu ulicí, struktur, fotografií a 3D objektů vytvořených a používaných v rámci tohoto projektu. Systém obsahuje uživatelské rozhraní implementované v technologii React pro snadnou manipulaci s daty.
Součástí práce je analýza, která rozebírá podobné práce z projektu Věnná města českých královen. Z této analýzy vyplynul potenciál navázat na Private API, které se zabývá správou dat, a na webový klient pro schvalování a publikaci 3D modelů. V Private API byly odstraněny identifikované chyby, aktualizovány použité balíčky a technologie, včetně modernizace kódu, a bylo rozšířeno o podporu fotografií. Díky aktualizaci a modernizaci bylo odstraněno množství zranitelností a tím byla zásadně zlepšena bezpečnost aplikace. Klient pro schvalování modelů byl rozšířen o správu ulic, struktur a 3D objektů. Systém implementovaný v této práci usnadňuje spolupráci historiků, grafiků a modelářů a umožňuje přístup k datům projektu Starý Most dalším aplikacím.

Klíčová slova správa dat, webová aplikace, uživatelské rozhraní, Starý Most, React, Express, Node.js, TypeScript

Abstract

This bachelor thesis focuses on the design and implementation of a web-based data management system for the Old Most project. This system allows the management of streets, structures, photographs and 3D objects created and used within this project. The system includes a user interface implemented in React for easy data manipulation.
This thesis includes an analysis that focuses similar works from the Dowry Towns of the Queens of Bohemia project. This analysis showed the potential to build on Private API, which deals with data management, and the web client for approving and publishing 3D models. Identified bugs in Private API were fixed, packages and technologies used were updated, including code upgrades, and it was extended to support photos. Thanks to the update, a number of vulnerabilities have been removed and therefore the security of the application has been significantly improved. The model approval client has been extended to manage streets, structures and 3D objects. The system implemented in this thesis facilitates collaboration between historians, graphic designers and modelers and allows other applications to access the Old Most project data.

Keywords data management, web application, user interface, Old Most, React, Express, Node.js, TypeScript

Seznam zkratek

API	Application programming interface – rozhraní pro programování aplikace
VMČK	Věnná města českých královen
3D	Trojrozměrný
AR	Augmented reality – rozšířená realita
VR	Virtual reality – virtuální realita
HTTP	Hypertext Transfer Protocol
SDK	Software development kit – sada pro vývoj softwaru
REST	Representational State Transfer – převod reprezentativního stavu
JS	JavaScript
TS	TypeScript

Úvod

Tato práce je součástí projektu Starý Most, což je projekt, na kterém spolupracuje Fakulta informačních technologií ČVUT v Praze, Filozofická fakulta Univerzity Jana Evangelisty Purkyně v Ústí nad Labem a Výzkumný ústav geodetický, topografický a kartografický, jehož cílem je vytvořit virtuální model dnes neexistujících, či změněných lokací v historické části města Most. Jako zdrojový materiál poslouží převážně historické kresby a fotografie, které budou opraveny, kolorovány a poslouží k vytvoření 3D modelů pro virtuální realitu dostupnou jak přes webové rozhraní, tak VR headsety. Cílem tohoto projektu je prohloubení povědomí o tomto historickém městě u mladší generace [1, 2].

Tato práce se zabývá analýzou, návrhem a implementací webového systémem pro správu dat pro projekt Starý Most, který by měl umožnil spolupráci historiků, modelářů a grafiků a dalších osob účastnících se na tomto projektu.

Práce je rozčleněna do několika částí. První část se zabývá analýzou prací vypracovaných v rámci projektu Věnná města českých královen, což je podobný projekt, v rámci kterého vzniklo několik závěrečných prací zabývající se podobnými problémy. Dále budou analyzovány uživatelské požadavky na webový systém pro správu dat. Druhá část práce se věnuje návrhu webové systému. Kapitola Realizace se zabývá implementací samotného webového systému. V této části jsou rozebrány postupy, které byly využity při implementaci systému. V kapitole testování je systém podroben testům. Na konci této práce je uvedena instalační příručka, ve které je popsán postup instalace systému pro správu dat, uživatelská příručka a programátorská příručka.

Kapitola 1
Cíle

Prvním cílem je analyzovat práce z podobného projektu – Věnná města českých královen a určit jejich využitelnost v projektu Starý Most.
Druhým cílem je získat informace o požadavcích a analyzovat funkční a nefunkční požadavky na webový systém pro správu dat.
Třetím cílem je navrhnout webovou aplikaci pro správu dat projektu Starý Most včetně úprav již existujících projektů, které budou využity v rámci tohoto systému.
Čtvrtým cílem je aktualizace balíčků a modulů starších prací z projektu VMČK a tím mimo jiné zvýšit jejich bezpečnost.
Pátým a hlavním cílem je implementace samotného webového systému, tedy backendu a administračního rozhraní, a změn nutných pro adaptaci existujících prací na požadavky projektu Starý Most.
Šestým cílem je provést integrační, akceptační a uživatelské testy webového systému.

Kapitola 2
Analýza

V této kapitole budou nejdříve analyzovány vybrané práce vytvořené v průběhu projektu Věnná města českých královen (VMČK). Nejdříve bude ale uveden popis projektu, jeho cíle, a proč je pro tuto práci a celý projekt Starý Most významný.
Nakonec budou analyzovány funkční a nefunkční požadavky na webový administrační systém pro projekt Starý Most.

2.1 Věnná města českých královen

Věnná města českých královen je projekt vyvíjený za spolupráce Fakulty informačních technologií ČVUT v Praze a Filozofické fakulty Univerzity Hradci Králové. Jeho hlavním cílem je vizualizace historických budov a místností prostřednictvím virtuální reality pomocí VR headsetů a rozšířené reality s využitím mobilních telefonů [3]. V rámci tohoto rozsáhlého projektu vzniklo několik týmových, bakalářských a závěrečných prací, které se zaměřují na klíčové aspekty, jako je správa dat a jejich dostupnost prostřednictvím aplikačního rozhraní (API), uživatelské rozhraní pro správu dat, procesy schvalování a vizualizace. Tyto práce mají za cíl umožnit uživatelům interakci s historickými daty a prostředími nejen prostřednictvím VR headsetů a mobilních telefonů s AR, ale také přes webové rozhraní. [2] Projekt Starý Most sdílí podobné cíle, a proto jsou práce provedené v rámci tohoto projektu zásadní při vývoji systému pro správu dat.

2.1.1 Private API

Private API představuje jednou z nejdůležitějších součástí projektu VMČK, protože slouží ke správě a poskytuje přístup k většině dat. Současný stav Private API je výsledkem několika závěrečných prací.
Toto API zahrnuje několik modulů a závislostí na externích systémech. Pro usnadnění nasazení a integrace s externími systémy využívá Private API kontejnerizační systém Docker. Hlavními součástmi jsou:

- Private API – NodeJS kontejner, ve kterém běží samotné API.
- PostgreSQL – SQL databáze obsahující data.
- MongoDB – NoSQL databáze sloužící primárně k ukládaní souborů, jako jsou 3D modely, assety a textury.

Vedle těchto externích závislostí obsahuje řetězec kontejnerů Private API také nástroje pro správu. Mezi ně patří:

- Mongo Client – slouží k přístupu a ke správě dat databáze MongoDB
- Adminer – umožňuje úpravy v databázi PostgreSQL
- Swagger – obsahuje dokumentaci k samotnému Private API

2.1.1.1 Využívané technologie

Docker je otevřená platforma pro vývoj a spouštění aplikací. Umožňuje spouštět aplikace v izolovaném prostředí nazývaném kontejner a tím usnadnit vývoj a nasazení aplikací [4].

Node.js je open source JavaScriptové běhové prostředí (také nazývané runtime), založené na enginu V8 vyvíjeném společností Google pro webový prohlížeč Google Chrome, využívající asynchronní I/O operace pro maximalizaci výkonu. Je určený pro vývoj škálovatelných serverů, webových aplikací, nástrojů příkazové řádky a skriptů. Node.js poskytuje standardní knihovnu, která umožňuje například přístup k souborovému systému a síťovým operacím [5].

Typescript je syntaktická nadstavba nad jazykem JavaScript vyvíjená společností Microsoft. TypeScript rozšiřuje JavaScript o statické typování, což ulehčuje orientaci v kódu a zlepšit detekci chyb. Kód v jazyce TypeScript se kompiluje do jazyka JavaScript [6].

Express.js je minimalistický framework pro vývoj serverových aplikací v jazyce JavaScript pro platformu Node.js. V současné době jde o nejpopulárnější framework pro tuto platformu. Některá média ho označují za de-facto standard pro vývoj serverových aplikací v Node.js, proto se také těší velké komunitní podpoře. Je zaměřen na vysoký výkon a nabízí routing, podporu middleware a view [7, 8].

TypeORM je knihovna, která poskytuje objektově relační mapování pro jazyky JavaScript a TypeScript. Podporuje migrace.

JWT (JSON Web Token) je otevřený standard pro bezpečnou výměnu informací mezi dvěma stranami. Tyto informace mohou být digitálně podepsány. Standard JWT podporuje také šifrování. Private API využívá JWT pro autentifikaci uživatelů [9, 10].

PostgreSQL je objektově-relační databázový systém s otevřeným zdrojovým kódem. Je aktivně vyvíjen více než 35 let a získal si dobrou pověst díky své spolehlivosti, robustnosti funkcí a výkonu. Private API využívá PostgreSQL pro ukládání většiny informací [11]. Podle průzkumů se těší velké popularitě a je jedním z nejvíce používaných databázových systému na světě [12].

MongoDB je dokumentově orientovaná NoSQL databáze vyvíjená společností MongoDB Inc. (původně 10gen). Místo tabulek využívá kolekce dokumentů. K dotazování se využívá jazyk MQL (MongoDB Query Language) [13].

Jest je testovací framework pro JavaScript zaměřený na jednoduchost původně vyvíjený společností Meta a od roku 2022 součástí OpenJS Foundation.

OAuth je otevřený standard pro autentifikaci. Private API využívá služby Google OAuth 2.0, což je jedna z nejpopulárnějších služeb na webovou autentifikaci.

V současné době je velkou nevýhodou Private API již poměrná zastaralost využitých technologií, ne po stránce samotné zastaralosti technického řešení, ale většina využitých balíčků využívá verze, které jsou často i více než 4 roky staré a již nepodporované. Toto se projevuje i mimo samotný zdrojový kód. JavaScript runtime Node.js je ve verzi 12 z roku 2019 a PostgreSQL ve verzi 11 z roku 2018. Hlavně v důsledku zastaralých balíčků hlásí bezpečnostní audit pomocí příkazu yarn audit 608 identifikovaných zranitelností v současně používaných balíčcích.

2.1.1.2 Entity

Area – jedná se o entitu, která sdružuje více struktur a reprezentuje konkrétní geografickou oblast s GPS souřadnicemi. Dále tato entita obsahuje informace jako je název, slovní popis, město a internetové odkazy. Oblasti také mohou obsahovat seznamy GPS souřadnic pojmenované jako AR Border a VR Border vymezující pracovní oblast virtuální a rozšířené reality [2].

Struktura – reprezentuje reálný objekt, který se skládá s 3D modelů. Slouží tedy pro obalení 3D modelů a sdružuje k nim relevantní informace jako například název, poloha a oblast, do které daná struktura patří [10, 14].

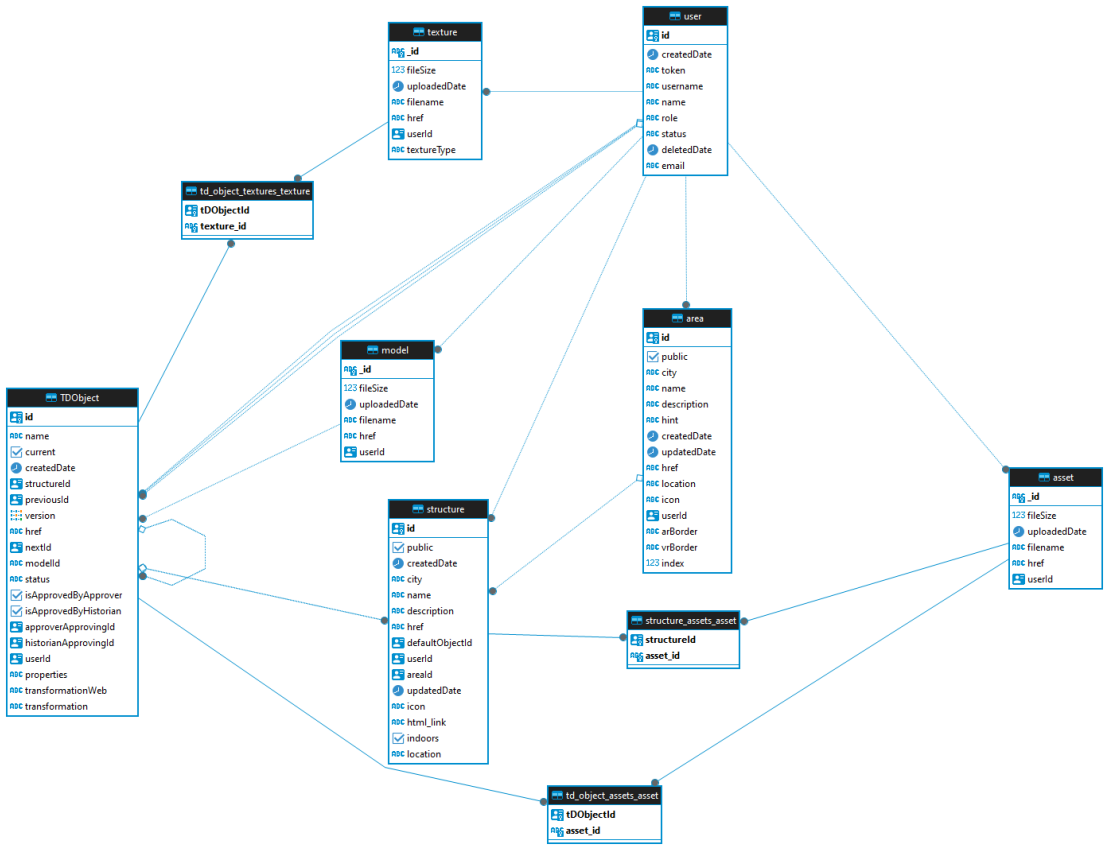
3D objekt – je entita, která obsahuje metadata k 3D modelům jako je jméno, verze, status a transformační matice. Dále odkazuje na samotný 3D model, jehož soubor je uložen v databázi MongoDB. Tento model propojuje s texturami a assety. API podporuje verzování 3D objektů. Takto verzované 3D objekty jsou ukládány pomocí spojového seznamu. Manipulovat a upravovat je možno pouze nejnovější verzi 3D objektu. Pro 3D modely jsou podporovány formáty TDS, BLEND, FBX, MAX, OBJ, SFB, GLB a GLTF [10, 2, 15].

Textura – je obrázek aplikovaný na 3D modely. API podporuje obrázky typu JPEG a PNG. Dále je podporováno několik typů textur jako je basecolor, specular, metallic, roughness, normal, height a ao.

Asset – je soubor obsahující dodatečné informace k 3D modelu jako jsou například materiály. Podporovány jsou formáty MTL a JSON [2].

Autentifikace – k přístupu k API je vyžadován přístupový JWT token. Private API k autentifikaci uživatelů využívá Google OAuth [10].

Schvalovací procesy – API poskytuje rozhraní pro schvalovací procesy. Tyto procesy slouží k ověření a schválení modelů a skládají se z několika fází. Ve schvalovacím procesu spolu spolupracují modeláři, grafici a historici. První fáze procesu se skládá z iterací, kde je model postupně vytvářen modelářem a následně hodnocen grafikem a historikem, kteří mohou k modelu přidávat komentáře. Druhá fáze procesu slouží v vytvoření variant modelu jako například model po aplikaci efektů počasí [10, 15].



Obrázek 2.1 ER diagram důležitých entit Private API. Vygenerováno autorem pomocí nástroje DBeaver [16]

2.1.1.3 Uživatelské role

Každý uživatel Private API musí mít přiřazenou roli, která mu dává práva k přístupu k jednotlivých funkcionalitám. Každá role reprezentuje jinou sestavu cílů, které uživatel v systému má. Roli uživatel získá registrací pomocí Google OAuth, avšak nově registrovaný uživatel není aktivní a pro využívání systému musí být jeho účet aktivován správcem. Výchozí rolí je role commonUser. Každý uživatel smí vytvářet nové objekty a dále smí upravovat a mazat objekty, které vytvořil. Administrátor smí upravovat a mazat všechny objekty v systému.

Common User – běžný uživatel, má nejnižší oprávnění a nesmí v systému upravovat ani mazat žádné objekty.

Modeller – modelář, role zamýšlená pro ověřeného běžného uživatele. Jde o uživatele, který do systému vytváří nové modely.

Historian – historik, role je určená pro uživatele, kteří poskytují zpětnou vazbu a schvalují modelářem vymodelovaný model. Tento uživatel posuzuje historickou přesnost modelu.

Approver – role zamýšlená pro grafika, tento uživatel také schvaluje modelářem vymodelovaný 3D model. Posuzuje jeho kvalitu z grafického hlediska.

Admin – administrátor, má neomezený přístup k funkcionalitám API. Může neomezeně vytvářet upravovat a mazat objekty.

Z analýzy minulých prací, a především zdrojového kódu Private API, vyplynulo, že role modeller, historian a approver se liší jen v pojmenování. Tyto role slouží k rozlišení uživatelů jen ze sémantického hlediska a nezakazují například manipulaci s určitými části API. Tito uživatelé jen nesmějí upravovat a mazat entity vytvořené jiným uživatelem.

2.1.1.4 Funkcionality a endpointy

V této části budou rozebrány jednotlivé endpointy Private API. Tímto budou zmapovány funkcionality, které API nabízí. Tyto informace jsou převzaty z [17].

Auth – slouží pro autentifikaci uživatelů. V současné době podporuje autentifikaci pomocí Google OAuth. Přijímá post requesty na endpointu /auth/googlesignin.

Area – skupina endpointů pro manipulaci s entitou Area (Oblast). Endpoint /areas podporuje operace POST pro vytvoření nové oblasti a GET pro získání seznamu všech oblastí. S jednotlivými oblastmi lze manipulovat pomocí jejich identifikátorů odesláním požadavků na adresu s jejich identifikátorem. Podporované metody jsou GET pro zobrazení informací o oblasti, PUT pro úpravu informací oblasti a DELETE pro odstranění oblasti.

Structure – nabízí správu entity struktura. Stejně jako oblasti nabízí endpointy pro vytvoření struktury a zobrazení seznamu všech struktur a také pro manipulaci se strukturou pomocí jejího identifikátoru. Navíc nabízí endpoint pro přidání 3D objektu ke struktuře, který přijímá název, soubor s 3D modelem, informaci o souborovém formátu a transformační matici.

3Dobjects – jsou endpointy využívané ke správě 3D objektů. Neumožňují přímo vytvořit 3D objekt, protože 3D objekty musí být přiřazeny ke strukturám. Umožňují ale vytvořit novou verzi 3D objektu, zobrazit informace o objektu, zobrazit seznam všech 3D objektů, upravit nebo odstranit verzi 3D objektu a nahrát textury a assety k 3D objektu.

Models – slouží především ke stahování modelů obsažených v 3D objektech. Dále umožňují tento soubor přepsat požadavkem PUT nebo odstranit požadavkem DELETE.

Textures – skupina endpointů určená pro správu textur. Stejně jako u modelů slouží především k získání souborů a umožňují texturu přepsat nebo odstranit.

Approvals – jedná se o velkou skupinu endpointů pro schvalovací procesy 3D modelů. Základní entitou je schvalovací proces, který má členy, jimiž jsou uživatelé, a iterace. Členové mohou ke schvalovacím procesům přidávat komentáře.

V dokumentaci k Private API jsou popsány také properties (vlastnosti), které jsou zamýšlené jako tagy 3D objektu, permissions, tedy oprávnění uživatelů, která by měla rozšířit oprávnění uživatelů nad rámec přiřazené role, efekty a pluginy, které by měly sloužit jako Blender transformace modelů a mapy a chunky (jednotlivé části mapy), pro ukládání například historických map [17]. Žádný z těchto endpointů však není implementovaný a pokud v kódu již jsou, jedná se většinou jen o kostru bez hlubší implementace.

2.1.1.5 CI/CD pipeline

Private API využívá gitlab CI/CD pipeline. Díky využití dockeru a přibalených testů je možné aplikaci sestavit, otestovat a nasadit. Pipeline se skládá ze stageů build, test a deploy. Stage test je dále rozdělen na joby – test, který testuje základní funkcionalitu API a test-seeding [10]. V současném stavu ale CI/CD pipeline nefunguje.

2.1.1.6 Použitelnost

Cílem Private API je uchovávat a poskytovat data o 3D modelech aplikacím, které tyto modely zobrazují v rozšířené realitě pomocí mobilních zařízení a ve virtuální realitě pomocí headsetů. Data jsou logicky strukturovány v oblastech, které obsahují jednotlivé struktury. O těchto objektech jsou uchovávány užitečné informace jako jsou popisy a GPS souřadnice. Vzhledem k podobným cílům jako má projekt Starý Most je použitelnost Private API v projektu webového systému pro správu dat velmi vysoká. Private API bude možné využít jako základ systému, bude však nutné implementovat nové funkcionality, které vyžaduje projekt Starý Most a API modernizovat.

2.1.2 Webové administrační rozhraní

Tato práce se zabývala vytvořením administračního rozhraní Private API pro projekt Věnná města českých královen. Aplikace je implementována v jazyce TypeScript a jako základ aplikace byl zvolen framework React. Pro 3D zobrazení využívá knihovnu Three.js [18].
Tato práce je ale pouze dobrým prototypem, jelikož v době psaní Private API postrádalo funkčnost nutnou pro funkci této aplikace, uchýlil se autor k simulování volání API [18]. Od této doby se API také značně rozvinulo. Aplikace proto neodpovídá současnému stavu Private API a je z praktického hlediska nevyužitelná. Práce ale může sloužit jako inspirace při návrhu dalších administračních nástrojů [2].

2.1.3 Administrační rozhraní SP1/2

V rámci předmětu BI-SP1 vznikla webového aplikace pro práci s daty uložených v Private API. Cílem této aplikace bylo sloužit jako administrační rozhraní. V navazujícím předmětu BI-SP2 byla rozšířena funkcionalita této aplikace na správu dat pro VR a AR aplikace. Aplikace je implementována v jazyku JavaScript s použitím frameworku Vue 3 a knihovny Bootstrap a podporuje základní CRUD operace nad entitami area, structure a 3D object. Dále umožňuje k 3D objektům nahrávat assety a textury. Kvůli tehdejším nesrovnalostem v API ale nejsou některé funkcionality plně funkční, jako například stahování 3D modelů [19].
Kvůli tomu, že toto rozhraní spíš než komponenty využívá pouze pohledy, dochází k opakování kódu a tyto pohledy nejsou modulární; další rozvoj a údržba takové aplikace je složitá. Tím, že je aplikace napsaná ve Vue a vedoucí této práce preferuje React je proto využitelnost nízká.

2.1.4 Webový klient pro schvalování a publikaci 3D modelů

Tato práce je nejnovější frontendová práce v rámci projektu VMCK. Jelikož pracovala s nejnovější verzí Private API je s ním stále kompatibilní. Je implementována v jazyce TypeScript a využívá framework React. Obsahuje komponenty pro správu 3D modelů, umožňuje spolupráci historiků, modelářů a grafiků nad těmito 3D modely a práci s nimi v rámci schvalovacích procesů [15].

Součástí tohoto klienta jsou kompletní komponenty pro nahrávání a správu 3D objektů, které umožňují:

- Vytváření, úpravy a mazání 3D objektů.
- Nahrávání modelů, assetů a textur.
- Zobrazení verzí 3D objektů včetně vytvoření nové verze.
- Stahování modelů, assetů a textur.

Dále jsou součástí nástroje a funkce pro manipulaci s daty a odesílání požadavků na backend, kterým je Private API.

2.1.4.1 Použitelnost

Tato práce je přínosná díky poměrně široké škále již implementovaných komponent a s tím spojené vrstvy, která komunikuje s Private API. Jelikož je vhodné pro nový systém pro správu dat využít Private API, bude možné využít komponenty vytvořené pro tento webový klient v novém uživatelském rozhraní nebo webový klient pro schvalování 3D modelů rozšířit o administrační modul pro správu dat.

2.1.5 Komponenta pro vizualizaci 3D modelů

Cílem této práce bylo vytvořit univerzální komponentu pro zobrazení 3D modelů na webu pro využití v aplikacích užívajících framework React. K vykreslování 3D využívá knihovnu Three.js. Komponenta umožňuje jak základní zobrazení 3D modelů na webu, tak porovnání 2 různých modelů. Toto je možné realizovat buď v jedné 3D scéně, kde jsou modely zobrazeny přes sebe, nebo ve dvou scénách, kde jsou tyto scény zobrazeny vedle sebe. Aplikace pro zobrazení porovnání scén využívá integrovaný synchronizér – interní komponentu pro synchronizaci pohybu kamery. Toto umožňuje synchronizovat pohyb kamery v jednom pohledu, což se automaticky promítne i do druhého pohledu. Tím je zajištěno, že je model zobrazen ze stejné pozice v obou pohledech [20].

Komponentu pro zobrazení náhled modelu je možno využít takto:

```
<ModelView style ={ style }
    model={ model }
    cameraOpt ion={ camera }
    controlsOption={ controls }
    requestHeaders={ requestHeaders }
    environmentParams={ environment }
/>
```

Výpis kódu 1 Zobrazení náhledu modelu. Převzato z [20].

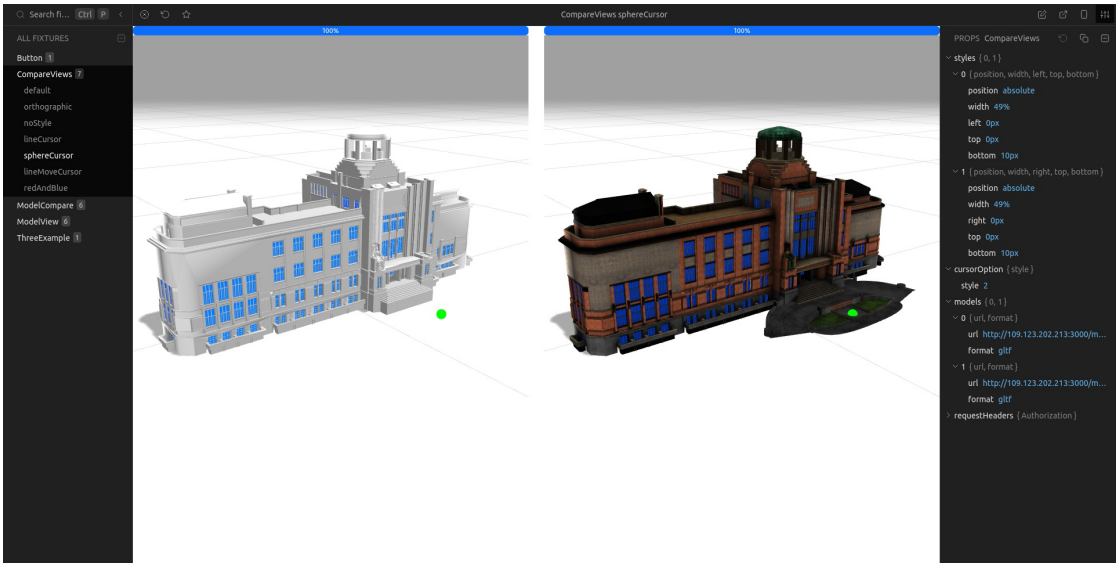
Kde:

- Model – objekt s URL a formátem 3D modelu.
- CameraOpt – přepnutí mezi ortografickou a perspektivní kamerou.
- ControlsOption – přepnutí mezi orbitálním a trackball ovládáním.
- RequestHeaders – hlavička HTTP request, který bude použit pro získání 3D modelu.
- EnvironmentParams – nastavení podlahy, mřížky a mlhy.
- CursorOption – přepínaní mezi line a sphere kurzorem. [20]

Komponentu pro zobrazení porovnání dvou modelů lze vyvolat takto:

```
<CompareViews styles={ styles }
    models={ models }
    cameraOption={ camera }
    cursorOption={{ style : cursorStyle, event : cursorEvent }}
    requestHeaders={ requestHeaders }
    environmentParams={ environment }
/>
```

Výpis kódu 2 Zobrazení porovnání modelů. Převzato z [20].



Obrázek 2.2 Komponenta pro porovnávání modelů. Získáno z [20].

2.2 Mapy

V této části bude provedena analýza jednotlivých mapových webových komponent poskytující možnost vlastního vykreslování objektů do mapy a interakcí s mapou Mnoho entit v projektu Starý Most závisí na geolokačních údajích. Protože součástí webového systému pro správu dat by mělo být i jednoduše použitelné administrační rozhraní, je třeba aby bylo možné zadávat geolokační údaje přetažením bodu na mapě.

Mapy.cz

Jedná se o populární mapovou službu vyvíjenou společností Seznam v České republice. Výhodou jsou proto velice dobré mapové podklady pro Českou republiku. Nevýhodou je však, že v roce 2025 bude ukončena podpora původního JS SDK a mapové komponenty založené na tomto SDK přestanou fungovat. Společnost Seznam poskytuje nové REST API jako náhradu. Pro využití této mapy je třeba svojí aplikaci zaregistrovat a získat API klíč. Od určitého počtu přístupů je API zpoplatněné. Jelikož společnost Seznam již neposkytuje k tomuto API knihovnu, doporučuje na svém webu knihovny pro zobrazení mapových dlaždic. Jmenovitě Leaflet, OpenLayers, MapLibre a CesiumJS [21].

Google Mapy

Google Mapy je webová aplikace vyvíjená společností Google. Poskytuje knihovnu a API pro vývoj aplikací. Pro přístup k tomuto API je opět nutné svojí aplikaci zaregistrovat a získat API klíč. Prvních 28 000 načtení mapy je zdarma, za další požadavky se však platí [22]. Pro zobrazení statické mapy je možno použít komponenty třetích stran jako je Leaflet, podle licenčních podmínek je ale pořád nutné využívat oficiální API [23].

Bing Mapy

Jedná se o mapovou službu od společnosti Microsoft. API Bing Map funguje na podobném principu jako REST API Seznam map. Vývojář potřebuje k přístupu API token a má k dispozici 125 000 API požadavků zdarma. Za další požadavky se platí. Společnost Microsoft podobně jako společnost Google poskytuje k těmto mapám vlastní JavaScript knihovnu pro usnadnění práce s mapou, nicméně lze mapy zobrazovat i pomocí komponent třetích stran jako je Leaflet [24].

OpenStreetMap

Jde o komunitního poskytovatele mapových podkladů. Tyto mapové podklady jsou nabízeny jako otevřená data pod licencí Open Data Commons Open Database License. Aplikace, které používaní OpenStreetMap mají povinnost uvést jako zdroj dat OpenStreetMap s informací o autorských právech a specifikovat, že se jedná o data licencovaná pod Open Database License například uvedením odkazu na stránku s autorskými právy [25].

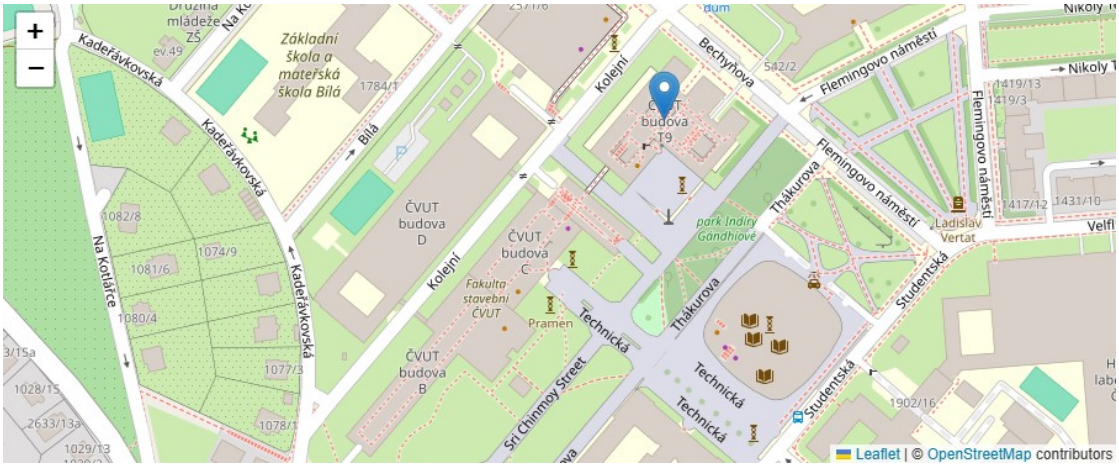
Leaflet

Leaflet je JavaScriptová knihovna pro zobrazování mapových dlaždic a tvorbu interaktivních map. Nabízí možnost zobrazovat na mapě body a přesun těchto bodů. Knihovna má podporu vrstev, překrytí a nabízí možnost kreslení vektorových obrazců definovaných pomocí skupiny GPS souřadnic do mapy. Poskytovatelem mapy může být jakýkoliv poskytovatel, který nabízí mapové

dlaždice ve formě obrázků. Existují komponenty pro populární framework jako je react-leaflet nebo Vue Leaflet [26].

```
<MapContainer center={position} zoom={13} scrollWheelZoom={false}>
    <TileLayer
      attribution='&copy;
      <a href="https://www.openstreetmap.org/copyright">OpenStreetMap</a> contributors'
      url="https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png"
    />
    <Marker position={position} />
</MapContainer>
```

Výpis kódu 3 Použití komponenty Leaflet s OpenStreetMaps podklady [27]



Obrázek 2.3 Komponenta Leaflet s podklady OpenStreetMap

OpenLayers

OpenLayers je knihovna pro tvoření dynamických map na webu. Stejně jako Leaflet podporuje vektorové kreslení [28]. Není ale zdaleka tak populární jako Leaflet a přesto, že existují implementace pro populární frameworky jako je React, nejsou tak kvalitně udržované jako Leaflet a ani se netěší moc velkému zájmu.

MapLibre

MapLibre je množina open-source komponent pro práci s mapami. Základní komponentou je stejně pojmenovaná komponenta MapLibre pro zobrazení map na webu. Podporuje GPU akcelerované vykreslování a vektorové kreslení [29]. Nicméně, stejně jako OpenLayers, trpí tím, že není příliš populární, což vede k nedostatku udržovaných implementací pro oblíbené frameworky.

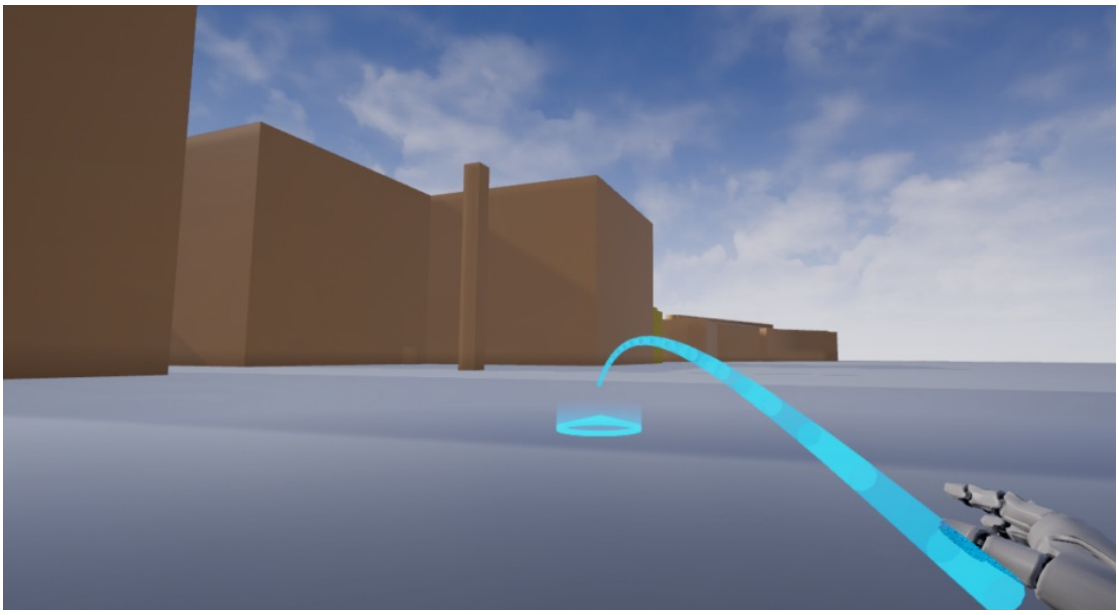
2.2.1 Použitelnost

Jako nejlepší volbou se zde jeví komponenta Leaflet s podklady OpenStreetMap a to hlavně proto, že Leaflet je z výše zmíněných komponent suverénně nejpopulárnější a díky tomu se těší velké komunitní podpoře. Ve prospěch OpenStreetMap pak hraje to, že jsou to jediné mapové podklady, které jsou dostupné skutečně zdarma a jejich licence je svobodná. Není potřeba žádných registrací a tokenů a jejich použití není nijak limitované.

2.3 Virtuální a rozšířená realita

Projekt Věnná města českých královen počítal s výstupními aplikacemi v podobě mobilního průvodce a virtuální reality. Pro distribuci modelů pro tyto aplikace vzniklo centralizované datové úložiště, ze kterého se časem stalo Private API. Myšlenkou tohoto bylo, že budou modely a další prostředky uloženy centrálně a výstupní aplikace k nim budou přistupovat přes webové API. Mobilní aplikace funguje velmi dobře a zobrazuje modely v prostoru pomocí rozšířené reality [2]

Virtuální realita se ukázala jako větší problém. Hlavní myšlenkou bylo implementovat aplikaci v herním engine jako je například Unreal Engine a Unity, ale protože tyto enginy využívají vlastní formát pro assety, ukázal se import v reálném čase jako velmi problematický. První prototypy vznikly v rámci týmových projektů a skládaly se z konfigurátoru, ve kterém bylo možné sestavit plánek budov a ze zobrazovací části, která . Následně vznikl v rámci bakalářské práce v Unreal Engine 4 prototyp klienta, který byl s pomocí pluginu Blender OSM schopen převádět OpenStreetMap do 3D struktur a načítat modely z Private API. Nakonec vznikl plugin na úrovni editoru, který vytvořil 3D scénu a zasadil do ní model stažený z API. Tato scéna se poté zkompilovala a vznikla tak aplikace na prohlížení daného modelu ve virtuální realitě. [2, 30]



Obrázek 2.4 Prototyp VR klienta v Unreal Engine 4. Převzato z [30]

V současné době probíhá analýza formátu OpenUSD pro využití v projektech VMČK a Starý Most. Pro herní engine Unity existuje plugin, který by se dal pro zobrazení scén v tomto formátu využít [2].

2.4 Funkční a nefunkční požadavky

2.4.1 Funkční požadavky

F1: Správa uživatelů

F1.1: Registrace

Aplikace umožní registraci nového účtu pomocí Google OAuth a přidělí nově vzniklému uživateli výchozí roli.

F1.2: Přihlášení

Aplikace umožní existujícímu uživateli přihlásit se do již existujícího účtu pomocí Google OAuth.

F2: Správa ulic

F2.1: Aplikace umožní uživateli vytvořit novou ulici v systému.

F2.2: Aplikace umožní existující ulici upravit nebo smazat. Také umožní zobrazit informace uložené v konkrétní ulici.

F3: Správa struktur ulice

F3.1: V rámci entity ulice existuje množina struktur, které se uživateli zobrazí při správě informací o konkrétní ulici.

F3.2: Aplikace umožní v této ulici vytvářet nové struktury a upravovat nebo mazat existující struktury. Stejně jako u ulic, bude možné zobrazit informace o struktuře.

F4: Správa 3D objektů

Každá struktura obsahuje 3D objekty. Každý 3D objekt obsahuje právě jeden 3D model, textury a assety.

F4.1: V ulicích bude možné vytvářet nové 3D objekty a upravovat stávající. Systém umožní nahrávání souborů s modelem, assety a texturami. 3D objekty budou obsahovat transformační matice.

F4.2: Systém bude 3D objekty verzovat. Při vytvoření nové verze 3D objektu bude stará verze zachována.

F5: Správa fotografií

Při tvorbě 3D modelů budou využity fotografie. Proto systém umožní k ulicím a strukturám přiřazovat fotografie.

F5.1: Systém umožní pro každou entitu ulice a struktury nahrávat libovolný počet fotografií. Již nahrané fotografie bude možné mazat

F5.2: Uživatelská aplikace nabídne pro každou ulici a strukturu komponentu galerie, díky které budou uživatelé moci fotografie procházet, nahrávat a mazat.

F6: Všechny funkcionality uživatelské aplikace budou přístupné pouze pro roli administrátor

2.4.2 Nefunkční požadavky

N1: Rozšířitelnost

Systém musí být pro účely dalšího vývoje snadno rozšířitelný.

N2: Jazyk

Aplikace bude poskytovat uživatelské rozhraní v češtině.

N3: Zpětná kompatibilita

Pokud bude při návrhu využito Private API z projektu VMČK, mělo by zůstat kompatibilní s aktuálním stavem v projektu VMČK kvůli možnosti aktualizace.

N4: React

Využití frameworku React je při tvorbě silně preferováno, kvůli zkušenostem vedoucích tohoto projektu.

Kapitola 3
Návrh

3.1 Architektura systému

Z analýzy vyplynulo, že Private API projektu Věnná města českých královen pokrývá mnoho požadavků na systém správy dat projektu Starý Most. Projekt Starý Most pracuje s ulicemi, strukturami a 3D objekty velmi podobně jako Private API pracuje s oblastmi, strukturami a 3D objekty. Jako backend bude tedy využito Private API. Protože jsou jeho součásti, zejména použité knihovny a software třetích stran, ale již zastaralé, budou tyto součásti aktualizovány. Do API bude rovněž přidána podpora fotografií, která se v něm nenachází, ale pro Starý Most je klíčová. Budou opraveny některé relevantní chyby zmiňované kolegy v pracích z projektu Věnná města českých královen. Využití Private API umožní budoucí využití i dalších technologií vyvinutých pro VMČK, jako například aplikace zabývající se zobrazením 3D objektů ve virtuální realitě. Díky tomu lze využít jako základ uživatelského rozhraní aplikaci pro schvalování 3D modelů, která byla rozebrána v analýze. Tato aplikace již obsahuje základní infrastrukturu pro komunikaci s mnohými částmi Private API a komponenty pro tvoření a úpravu 3D modelů, které mohou být znovu využity. Aplikace bude rozšířena o stránky na správu ulic, struktur, fotografií a s pomocí již existujících komponent budou implementovány stránky pro správu 3D modelů pro administrátory.

3.2 Případy užití

V této části budou rozebrány případy užití. Jelikož uživatelé interagují s frontendovou částí systému, bude se tato část týkat převážně uživatelského rozhraní a jeho interakcí s API.

Přihlášení

UC1 - Registrovat se

Uživatel klikne na tlačítko registrovat a zaregistruje se pomocí Google Auth

UC2 - Přihlásit se

Pre-condition

Uživatel má v systému existující účet

Basic Path

Uživatel klikne na tlačítko přihlásit se a přihlásí se do již evidovaného účtu pomocí Google Auth

Správa ulic

UC3 - Zobrazit seznam ulic

Pre-condition

Uživatel je přihlášen

Basic Path

Systém zobrazí seznam ulic

UC4 - Vložit novou ulici

Pre-condition

Uživatel je přihlášený a má přístup k seznamu ulic.

Basic Path

- Zobrazit seznam ulic.
- Uživatel na stránce ulic klikne na tlačítko vytvořit.
- Systém zobrazí formulář pro vytvoření ulice.
- Uživatel vyplní data o ulici a klepne na tlačítko uložit.
- Systém vytvoří novou ulici.

UC5 - Upravit ulici

Pre-condition

Uživatel je přihlášen a nachází se na stránce se seznamem ulic.

Basic Path

- Uživatel klikne na tlačítko upravit u ulice, kterou chce upravit.
- Systém zobrazí předvyplněný formulář s údaji o ulici.
- Uživatel údaje upraví a klepne na tlačítko uložit.
- Systém upraví vybranou ulici.

UC6 - Odstranit ulici

Pre-condition

Uživatel je přihlášen a nachází se na stránce se seznamem ulic.

Basic Path

- Uživatel klikne na tlačítko odstranit u ulice, kterou chce odstranit.
- Systém se uživatele zeptá, jestli si přeje ulici skutečně odstranit.
- Uživatel volbu potvrdí.
- Systém odstraní vybranou ulici.

UC7 - Zobrazit detail ulice

Pre-condition

Uživatel je přihlášen a nachází se na stránce se seznamem ulic.

Basic Path

- Uživatel klikne na tlačítko detail u ulice, již chce zobrazit.
- Systém zobrazí data o ulici.

Správa struktur u vybrané ulice

UC8 - Zobrazit seznam struktur

Pre-condition

Uživatel je přihlášen a nachází se na stránce s detailem ulice.

Basic Path

- Systém načte abecední seznam struktur a zobrazí ho uživateli.

UC9 - Vložit novou strukturu

Pre-condition

Uživatel je přihlášen a nachází se na stránce s detailem ulice. Je načtený seznam struktur.

Basic Path

- Uživatel klikne na tlačítko vytvořit strukturu.
- Systém zobrazí formulář pro vytvořen struktury.
- Uživatel vyplní data o struktuře a klepne na tlačítko uložit.
- Systém vytvoří novou strukturu přiřazenou k ulici, ve které se uživatel nachází.

UC10 - Upravit strukturu

Pre-condition

Uživatel je přihlášen a nachází se na stránce s detailem ulice. Je načtený seznam struktur.

Basic Path

- Uživatel klikne na tlačítko upravit u struktury, kterou chce upravit.
- Systém zobrazí předvyplněný formulář s údaji o struktuře.
- Uživatel údaje upraví a klepne na tlačítko uložit.
- Systém uloží změny u vybrané struktury.

UC11 - Odstranit strukturu

Pre-condition

Uživatel je přihlášen a nachází se na stránce s detailem ulice. Je načtený seznam struktur.

Basic Path

- Uživatel klikne na tlačítko odstranit u struktury, kterou chce odstranit.
- Systém se uživatele zeptá, jestli si přeje strukturu skutečně odstranit.
- Uživatel volbu potvrdí.
- Systém odstraní vybranou strukturu.

UC12 - Zobrazit detail struktury

Pre-condition

Uživatel je přihlášen a nachází se na stránce s detailem ulice. Je načtený seznam struktur.

Basic Path

- Uživatel klikne na tlačítko detail u struktury, již chce zobrazit.
- Systém zobrazí informace o struktuře.

Správa modelů u vybrané struktury

UC13 - Zobrazit seznam modelů

Pre-condition

Uživatel je přihlášen a nachází se na stránce s detailem struktury.

Basic Path

- Uživatel klikne na tlačítko pro zobrazení seznamu 3D objektů.
- Systém zobrazí seznam modelů.

UC14 - Nahrání modelu

Pre-condition

Uživatel je přihlášen a nachází se na stránce se seznamem 3D objektů vybrané struktury.

Basic Path

- Uživatel vybere soubor v nahrávacím formuláři a klikne na tlačítko nahrát.
- Systém nahraje model, nastaví výchozí transformační matici a uloží ho.

UC15 - Zobrazení informací o modelu

Pre-condition

Uživatel je přihlášen a nachází se na stránce se seznamem 3D objektů vybrané struktury.

Basic Path

- Uživatel klikne u modelu na tlačítko detail.
- Systém zobrazí informace o modelu, tedy: transformační matici, seznam assetů a seznam textur.

UC16 - Změna transformační matice modelu

Pre-condition

Uživatel je přihlášen a nachází se na stránce se seznamem 3D objektů vybrané struktury.

Basic Path

- Uživatel klikne na tlačítko upravit.
- Systém zobrazí stránku úpravy modelu.
- Uživatel upraví transformační matici.
- Systém vizuálně odliší upravené údaje od neupravených.
- Uživatel klikne na tlačítko uložit.
- Systém uloží upravenou transformační matici.

UC17 - Nahrát nový asset k modelu

Pre-condition

Uživatel je přihlášen a nachází se na stránce úpravy 3D objektu.

Basic Path

- Uživatel v tabulce assetů klikne na nahrát.
- Systém zobrazí dialog s možností nahrát soubor.
- Uživatel nahraje soubor a klikne na tlačítko nahrát.

UC18 - Nahrát novou texturu k modelu

Pre-condition

Uživatel je přihlášen a nachází se na stránce úpravy 3D objektu.

Basic Path

- Uživatel v tabulce textur klikne na nahrát.
- Systém zobrazí dialog s možností nahrát soubor.
- Uživatel nahraje soubor a klikne na tlačítko nahrát.

UC19 - Odstranit asset

Pre-condition

Basic Path

- Zobrazení informací o modelu.
- Uživatel u assetu klikne na tlačítko odstranit.
- Systém se uživatele zeptá, zda si skutečně přeje položku odstranit.
- Uživatel potvrdí.
- Systém odstraní asset.

UC20 - Odstranit texturu

Pre-condition

Basic Path

- Zobrazení informací o modelu.
- Uživatel u textury klikne na tlačítko odstranit.
- Systém se uživatele zeptá, zda si skutečně přeje položku odstranit.
- Uživatel potvrdí.
- Systém odstraní texturu.

UC21 - Zobrazit galerii a jednotlivé fotografie

Pre-condition

Uživatel se nachází na stránce s galerií, tedy na stránce s informacemi o ulici nebo struktuře.

Basic Path

- Systém zobrazí komponentu pro nahrávání fotografií.
- Systém zobrazí mřížku náhledů nahraných fotografií.
- Uživatel klikne na náhled nějaké fotografie.
- Systém zobrazí fotografii v plné velikosti.

UC22 - Nahrát novou fotografii

Pre-condition

Uživatel se nachází na stránce s galerií, tedy na stránce s informacemi o ulici nebo struktuře.

Basic Path

- Zobrazení galerie.
- Uživatel do nahrávací sekce přetáhne soubory nebo soubory vybere pomocí systémového dialogu.
- Systém zobrazí tlačítko nahrát, pokud jsou vybrány nějaké fotografie k nahrání.
- Uživatel klikne na tlačítko nahrát.
- Systém nahraje soubory s fotografiemi na server a zobrazí aktuální fotografie v galerii.

UC23 - Odstranit fotografii

Pre-condition

Uživatel se nachází na stránce s galerií a v galerii je alespoň jedna fotografie.

Basic Path

- Uživatel klikne na tlačítko křížku u fotografie, kterou si přeje smazat.
- Systém uživatele vyzve k potvrzení akce.
- Systém fotografii odstraní. Pokud není přiřazena k žádnému dalšímu objektu (ulici nebo struktuře), odstraní i soubor s fotografií.

3.3 Serverová část

3.3.1 Podpora fotografií

Protože se projekt zabývá rekonstrukcí již neexistujících objektů, bylo by vhodné, aby systém pro správu dat kromě ulic, struktur a 3D objektů dokázal k těmto entitám také ukládat fotografie. V této části budou rozebrány návrhy možných implementací podpory fotografií v Private API. Výstupem této části je výběr nejvhodnějšího způsobu implementace v závislosti na požadavcích projektu.

První možností je implicitně vytvořit u struktur a ulic prázdný objekt (tzv. null objekt) a tímto umožnit nahrávaní fotografií, které ještě nejsou rozřazeny. Fotky by byly uloženy ve struktuře nebo ulici, která by měla speciální status. Tato strategie má své výhody, v tom, že umožňuje využití stávající implementace API bez nutnosti zásadních úprav. Nicméně, přináší vyšší složitost v samotné správě a vnáší vyšší složitost do implementaci uživatelského rozhraní, které by muselo zajistit vytvoření těchto objektů, což může komplikovat údržbu a budoucí rozvoj. Tím pádem může být obtížnější udržovat systém aktualizovaný a připravený na budoucí změny.

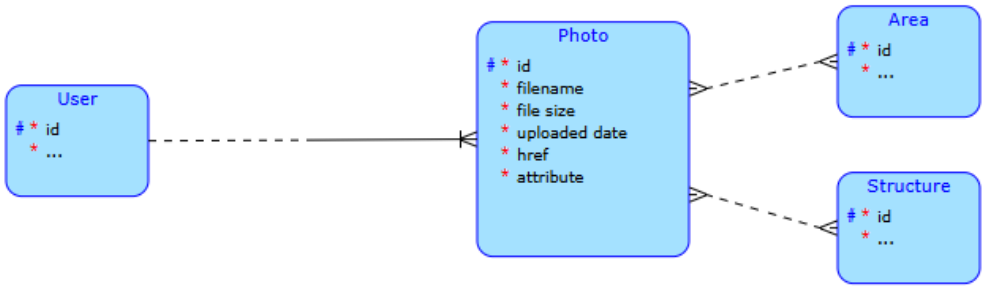
Druhou možností je fotografie ukládat v tabulkách, které by měly cizí klíče k objektům, ke kterým fotografie patří. Každá skupina objektů by měla vlastní tabulku, jako například PhotoStreet vázající se k ulici a PhotoStructure navázané na strukturu. Vazba na konkrétní entitu by nebyla povinná. Tímto by byly vyřešeny například nepřiřazené fotografie.

Třetí možností je fotky reprezentovat v jedné tabulce a u každé fotografie udržovat příznak, který by určoval, ke kterému objektu fotografie patří (například ve sloupci), v tomto případě by ale vazby nebyly jednoznačné. Toto by vedlo k tomu, že by takovou relaci nešlo vytvořit pomocí TypeORM, což by značně komplikovalo implementaci. Navíc nelze zajistit integritu na databázové úrovni.

Čtvrtou možností je fotografie ukládat v jedné tabulce a pomocí vazebních tabulek je propojovat s objektem (Photo-PhotoStreet-Street). Tato relace se jednoduše modeluje pomocí ORM a nezavádí atypické chování do kódu. Nevýhodou může být, že jsou potřeba vazební tabulky. Výhodou je pak že jedna fotografie může patřit k více objektům, na rozdíl od druhé možnosti nevzniknou žádné duplicity. Toto řešení umožní fotky i jednoduše přesouvat a kopírovat bez nutnosti vytvoření dalších kopií.

Jako nejlepší se jeví čtvrtá možnost a to z těchto důvodů:

- Oproti první možnosti nevyžaduje vytvoření implicitních objektů, které by zaváděly vyšší komplexitu do systému a tím ho dělaly složitějším na údržbu.
- Na rozdíl od druhé možnosti umožňuje sdílet stejné fotografie mezi různými entitami.
- Oproti třetí možnosti umožňuje zajistit integritu dat na databázové úrovni a tím tedy nekomplikuje implementaci a údržbu systému.



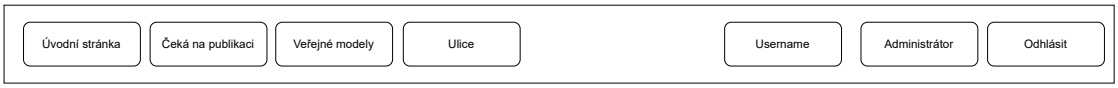
Obrázek 3.1 Návrh rozšíření databáze pro podporu fotografií. Vytvořeno pomocí nástroje DBS [31]

3.4 Uživatelské rozhraní

3.4.1 Návrh stránek

V této části budou navrženy jednotlivé stránky uživatelského rozhraní.
Návrh stránek bude sloužit jako reference při rozšíření stávají aplikace pro správu modelů. Grafická realizace bude provedena tak, aby nové stránky stylově zapadaly do již existující aplikace. Návrhy jednotlivých stránek a komponent byly provedeny v nástroji Diagrams.net [32], stejně jako v původní aplikaci [15].

Navigační menu je potřeba pro potřeby aplikace upravit a pro roli administrátor přidat tlačítko pro zobrazení seznamu ulic. Pro ostatní role nebudou nové funkcionality zpřístupněny.

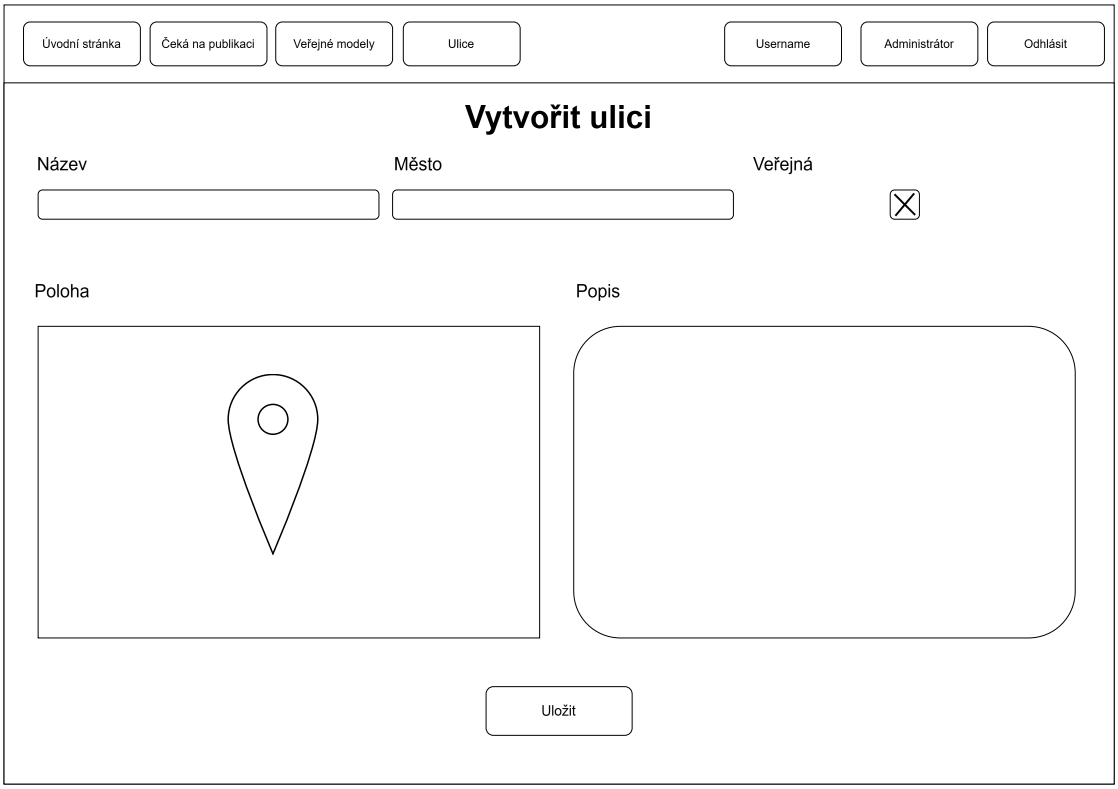


Obrázek 3.2 Stránka seznamu ulic

Stránka se seznamem ulic zobrazuje abecední seznam všech ulic v systému. Obsahuje tabulku, která reprezentuje tento seznam s akčními tlačítky pro přístup na detail ulice, úpravu a smazání.

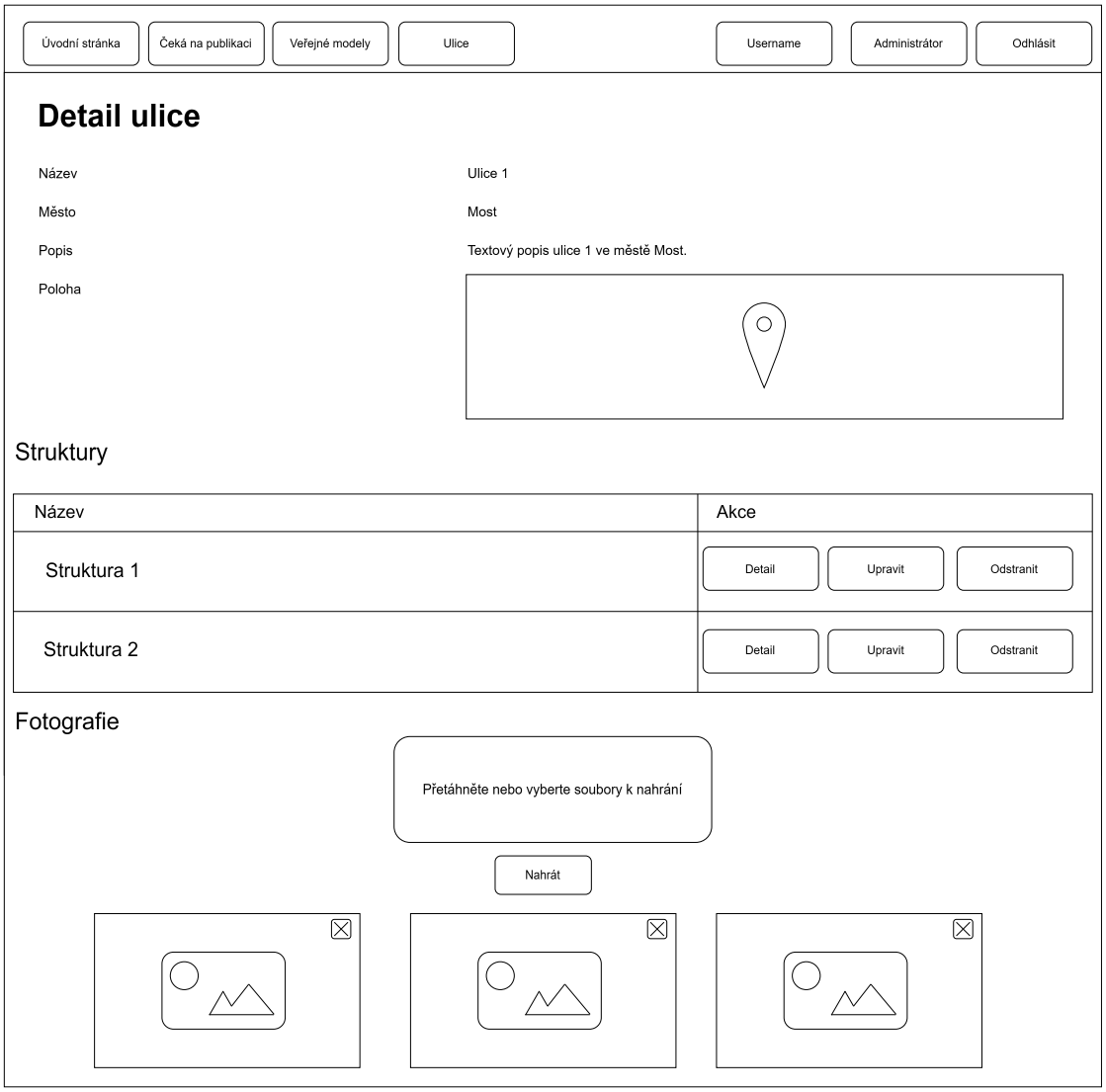
Obrázek 3.3 Stránka seznamu ulic

Stránka pro vytvoření a úpravu ulice je formulář, kde uživatel vyplní všechny informace o ulici. Samotný design je inspirovaný designem, který je již v aplikaci používán. Tato stránka bude také sloužit pro úpravu již existujících ulic. V takovém případě bude tento formulář po otevření již předvyplněný údaji o ulici. Na této stránce se nachází interaktivní mapa, kde může uživatel vybrat lokaci ulice.



Obrázek 3.4 Stránka vytvoření a úpravy ulice

Stránka detailu ulice zobrazí informace o ulici. Pod těmito informacemi bude seznam všech struktur, které jsou přiřazeny k této ulici. Vedle každé struktury zobrazí stejná akční tlačítka, jako zobrazuje seznam ulic.



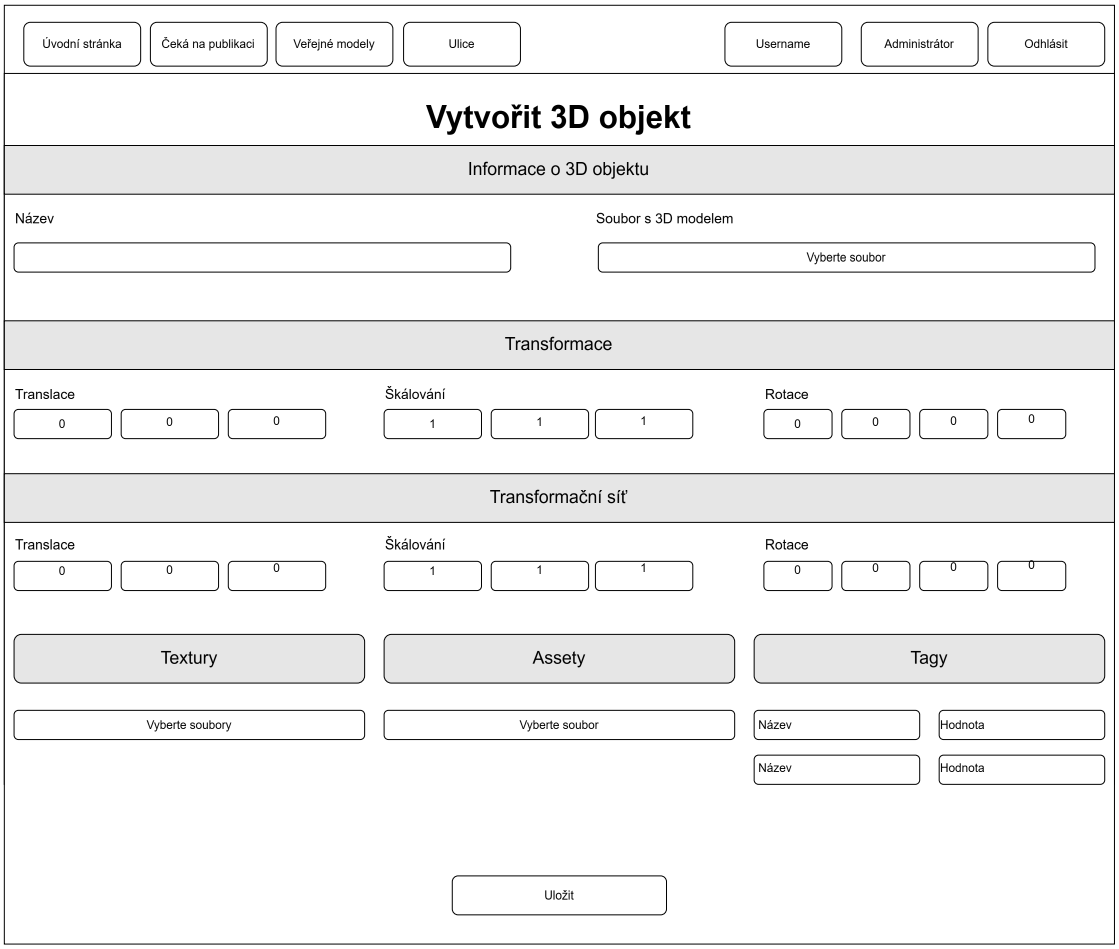
Obrázek 3.5 Stránka vytvoření a úpravy ulice

Stránka vytvoření a úpravy struktury je formulář podobný tomu pro vytvoření ulic. Umožní tvorbu nové a úpravu již existující struktury. Při provádění úprav bude formulář opět předvyplněn.

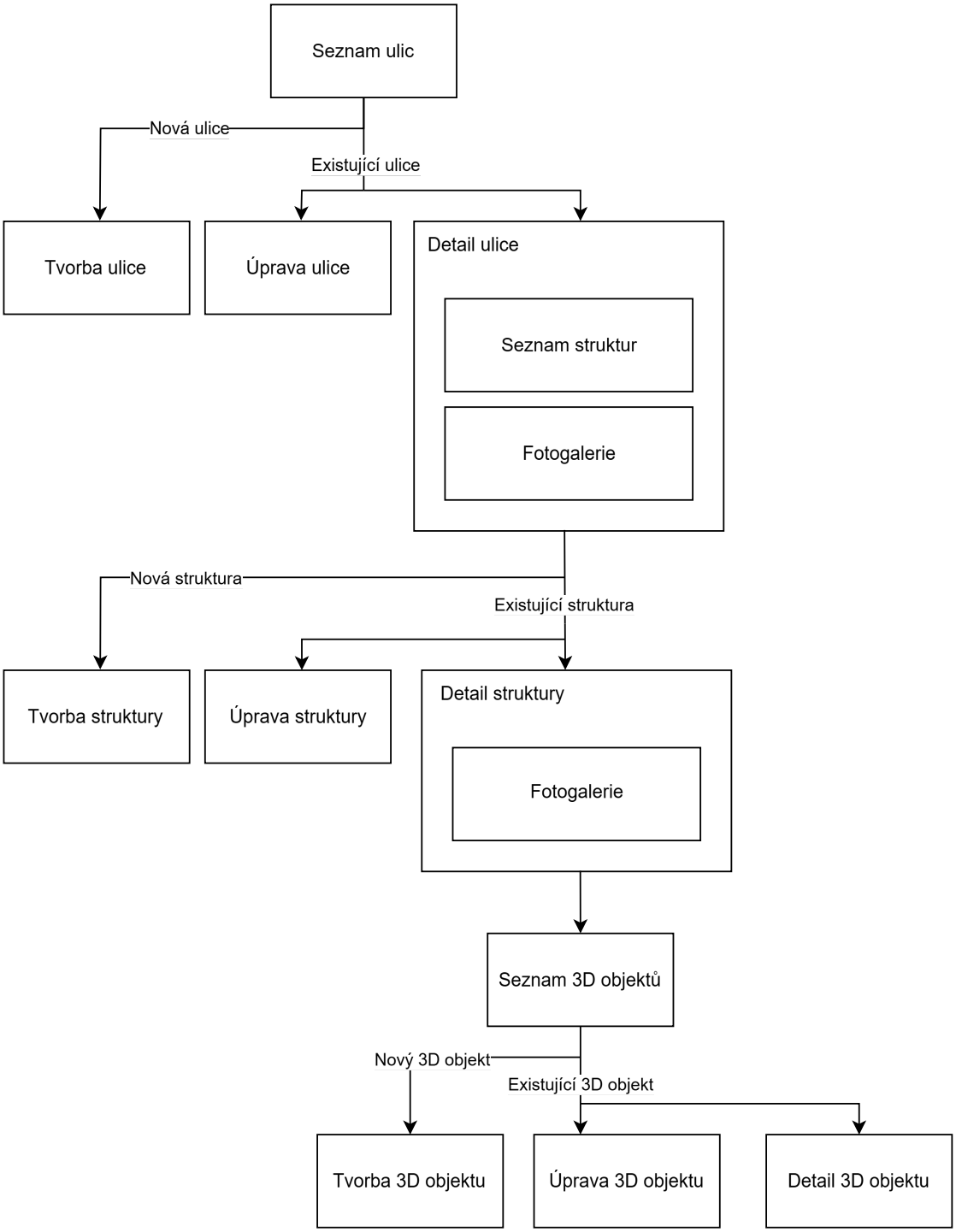
Stránka seznamu 3D modelů zvolené struktury zobrazí modely patřící ke struktuře včetně starších verzí. U každého modelu zobrazí tlačítka s proveditelnými akcemi.

Obrázek 3.6 Stránka seznamu 3D objektů vybrané struktury

Stránka úpravy 3D modelu zobrazí informace o 3D modelu a umožní upravit transformační matice a tagy, nahrát nové assety a textury a také vytvořit novou verzi tohoto modelu.



Obrázek 3.7 Stránka vytvoření a úpravy 3D objektu



Obrázek 3.8 Mapa a flow nových stránek

3.4.2 Galerie

Pro zobrazení a správu fotografií v uživatelském rozhraní je nutné navrhnout komponenty galerie. Tato galerie bude využita na stránkách detailu ulice a struktury. Proto i z hlediska budoucího využití je nutné, aby byly tyto komponenty co nejvíce znovupoužitelné.
Samotná galerie se bude skládat z komponent pro nahrávání obrázků, zobrazení náhledu obrázků a komponenty pro zobrazení zvětšeného obrázku přes celou obrazovku.

Komponenta nahrávání – tato komponenta bude sloužit k nahrání obrázků. Umožní uživateli přetáhnout do okna prohlížeče jeden či více obrázků nebo také vybrat obrázky pomocí systémového dialogu pro výběr souborů. Komponenta před nahráním nejdříve zobrazí zmenšeniny vybraných obrázků a umožní uživateli obrázky před nahráním odstranit pomocí symbolu křížku na jednotlivých náhledech.

Komponenta náhledu – zobrazí zmenšeniny obrázků přiřazených k ulici nebo struktuře v mřížce. Po kliknutí na obrázek se otevře komponenta zobrazení obrázku. V rohu obrázků bude symbol křížku, který odstraní obrázek z entity, do které je nahrán. Pokud je nahrán jen do této entity, zároveň odstraní obrázek z úložiště API.

Komponenta zobrazení obrázku – zobrazí obrázek v plné velikosti přes celou obrazovku. Při otevření překryje většinu obrazovky, ale nechá prostor kolem okrajů. Kliknutím do tohoto prostoru se obrázek zase zmenší. Komponenta nezobrazí obrázek na nové kartě, ale překryje již zobrazovanou stránku. Tato komponenta se také nazývá *Lightbox*.

Kapitola 4

Realizace

4.1 Zprovoznění CI/CD Private API

Prvním krokem k celkové údržbě a aktualizaci balíčků Private API je zprovoznění automatizovaných testů a na ně navázaný Gitlab Continuous Integration. Bez těchto testů by bylo ověření, že vše funguje tak jak má i po aktualizaci jednotlivých komponent, velmi složité.
Hlavním problémem, který se objevil, bylo že se vůbec nespouštěl první stage GitLab pipeline. Toto selhání bylo způsobeno nefunkčním připojením k Dockeru. Příčinou byla aktualizace školního GitLabu, který projekt využívá pro běh pipeline s Dockerem v Dockeru. Pro tento účel je nezbytné buď nakonfigurovat TLS nebo explicitně vypnout zabezpečení [33].

Když se pipeline nakonec spustila, ukázalo se, že testy neodpovídají současné verzi Private API a selhávají. Hlavním důvodem selhání bylo přidání validátorů pro formáty souborů modelů a assetů. Nová verze také vyžaduje specifikaci typu při nahrávání textur. Kvůli změnám v databázi, zejména povinným transformacím u modelů, bylo nutné upravit seedovací část testů a zahrnout nové parametry. Byly zjištěny chyby ve schématu databáze, které vyžadovaly povinné definice ikon u entity Area a Structure, a také kvůli duplicitní definici typu Struktura, což způsobilo označení některých sloupců v databázi jako povinných. Dále byla zjištěna chyba v migraci, která způsobovala povinné atributy u entity Area.

Původní CI/CD pipeline obsahovala stage „deploy“, který nasazoval aplikaci na vývojový server projektu VMČK pomocí SSH připojení a privátního klíče. Jelikož tento server není určen pro projekt Starý Most, byl tento stage zrušen.

4.2 Aktualizace balíčků

4.2.1 Aktualizace Docker kontejnerů

Verze Node byla aktualizována z 12.11 na 20.12, což je v současné době nejnovější verze s dlouhodobou podporou. Avšak, během této aktualizace se vyskytl problém s chybějící knihovnou libcrypto.so.1.1, kterou vyžaduje balíček MongoDB Memory Server používaný k vytvoření lokálního MongoDB serveru při testování. Tento problém byl vyřešen přechodem na kontejner Node 20.12-bullseye, který je založen na Debianu Bullseye, místo novější verze Debianu Bookworm.
Nicméně, při použití této verze se objevila další komplikace kvůli MongoDB a balíčku MongoDB Memory Server, kde se zobrazovala chybová zpráva „Status Code is 403 (MongoDB's 404)". Tento problém byl způsoben nekompatibilní starší verzí MongoDB. K řešení této situace byla MongoDB aktualizována na verzi 4.4 pomocí úpravy konfiguračního souboru package.json.

```
"config": {
  "mongodbMemoryServer": {
    "version": "v4.4-latest"
  }
}
```

Výpis kódu 4 Vybrání specifické verze MongoDBMemoryServeru

Dále byla aktualizována verze databáze PostgreSQL z verze 11, původně vydané v roce 2018, na verzi 16, která vyšla 14. září 2023 a jedná se v době psaní práce o nejnovější verzi tohoto databázového systému [34]. Tato verze by měla podle tvůrců přinést vylepšení ve výkonu. Podpora této verze je očekávána do 9. listopadu 2028, zatímco podpora verze 11 skončila 9. listopadu 2023 a tím pádem už nebude dostávat žádné aktualizace, včetně těch bezpečnostních. Konkrétně byl databázový kontejner aktualizován na verzi 16-alpine, který by měl v budoucnu zahrnovat všechny minoritní verze s hlavní verzí 16.
Protože byl MongoDBMemroryServer aktualizován na verzi 4.4 bez větších problémů, byl kontejner MongoDB využitý v produkční verzi aplikace také aktualizován na verzi 4.4. Upgrade na novější verze nebyl proveden, protože vyšší verze MongoDB i javascriptových balíčků pro připojení k MongoDB už obsahují hlubší změny, kvůli kterým by byla nutná hlubší revize celého kódu Private API.

4.2.2 Aktualizace JavaScript závislostí

V této sekci bude diskutován průběh aktualizace závislostí, které využívá jak frontend, tak API. Jelikož je frontend nová práce a využívá například knihovnu React v nejnovější verzi, nebyla aktualizace velmi obtížná. Analýzu zranitelností a možnosti aktualizace bylo možno provést pomocí příkazu *npm audit*. Identifikované zranitelnosti bylo pak možné odstranit aktualizací balíčků na novější kompatibilní verze pomocí příkazu *npm audit fix*.
V případě API byla tato aktualizace řádově složitější kvůli tomu, že API většinou využívá balíčky, které jsou minimálně 4 roky staré. Jak již vyplynulo z analýzy v těchto balíčcích bylo identifikováno mnoho zranitelností. Bohužel se ukázalo, že aktualizace způsobem, který byl využit u frontendu není možná, protože i verze označené jako kompatibilní obsahovaly změny, které rozbíjely testy i samotnou kompilaci Private API.
Jako hlavní zdroj problémů se ukázal balíček Jest pro testování kódu. Tento balíček měl zároveň velký počet závislostí, ve kterých byly rovněž identifikovány bezpečnostní zranitelnosti. Jelikož při větší aktualizaci přestala většina testů fungovat, byl nutný systematický přístup. Balíček Jest byl aktualizován vždy po jedné verzi a s každou novou verzí byly znovu spuštěny testy. S Jest

byly aktualizovány vždy i všechny jeho závislosti. Tedy balíčky *ts-jest*, *@types/jest* a *jest-cirrus*. Aktualizace tímto způsobem je velmi časově náročná a je třeba mít možnost se po každé inkrementální aktualizaci vrátit k poslední funkční verzi. Proto byla každá, byť sebemenší změna verzována jako commit v systému git. Toto vytvořilo desítky commitů, které byly na konci celého aktualizačního procesu metodou squash přetvořeny do jednoho jediného commitu, aby byla zachována čitelná historie změn.
Hned při aktualizaci z verze 25 na verzi 26 se objevily problémy s kompatibilitou. Bylo zjištěno, že způsob, jakým fungují asynchronní testy se změnil. Současné testy využívaly callback done pro hlášení dokončení testu. To jde ve verzi 26 ale jen u synchronních testů, protože asynchronní testy mají vracet *Promise* a testovací framework si jejich doběhnutí zajistí sám. Testy byly tudíž upraveny, aby reflektovaly tuto změnu. U asynchronních testů byl odstraněn callback done a v případě chyby se nyní propaguje výjimka až na nejvyšší úroveň, aby ji zachytil testovací framework a test selhal.
Dále byly testovací balíčky aktualizovány až na verzi 27 bez větších problémů. Problémy nastaly až u aktualizace na verzi 28, která vyžaduje Typescript 4.6, ale projekt využíval Typescript 3.9. Bylo tedy nutné aktualizovat i Typescript. Protože ale Typescript při přechodu z verze 3 na verzi 4 také obsahuje takzvané *breaking changes*, bylo nutné k aktualizaci přistupovat s podobnou opatrností jako v případě balíčku *Jest*. Typescript byl nejprve aktualizován na základní verzi 4. Zde se objevil problém se všemi konstrukty pro zachytávání výjimek v kódu (*try-catch*). Ve všech verzích Typescriptu až do verze 4 se v catch blocích automaticky používal typ *any*. V Typescriptu 4 toto bylo změněno a tento typ je nyní implicitně *unknown*. Je tedy nutné tuto proměnnou správně natypovat. Toto by ale rozbilo velmi mnoho projektů, a proto byl zaveden nový parametr, který vypíná použití unknown v catch proměnných.
V nastaveních Typescript *tsconfig* stačí tuto možnost přidat.

```
"compilerOptions": {
  "useUnknownInCatchVariables": false
}
```

Výpis kódu 5 Vypnutí implicitního unknown v catch proměnných

Způsobem postupné aktualizace se podařilo aktualizovat Typescript až na verzi 4.6.4, což je i dnes stále hojně využívaná verze.
Díky aktualizované verzi Typescriptu se následně podařilo aktualizovat balíček Jest a sním spojené balíčky na verzi 28 a následně i na verzi 29, což je v dnešní době nejnovější verze tohoto testovacího frameworku.
Další na řadě byl framework Express, který je nejdůležitější součástí celého projektu. Díky tomu, že tento framework je již relativně dospělý a velké změny v kompatibilitě nejsou časté, proběhla aktualizace na nejnovější verzi hladce. K tomu přispěla i aktualizovaná verze Typescriptu, protože nové verze frameworku Express fungují pouze s Typescriptem 4.1 a novějším.
Při aktualizaci Expressu se objevila menší chyba v neshodě verzí balíčku *@types/formidable* napříč závislostmi. Tento balíček poskytuje správné typy pro jazyk Typescript pro balíček *formidable*. Balíček *express-formidable* a *formidable* používali dvě vzájemně nekompatibilní verze *@types/formidable*. Řešením bylo zafixovat tuto závislost na stejné verzi pro oba balíčky.
Po těchto velkých změnách již bylo možné ostatní balíčky aktualizovat jednoduše tak, že pomocí *yarn audit* byli identifikováni kandidáti k aktualizaci a pomocí *yarn upgrade-interactive* byly tyto závislosti aktualizovány. Jediný problém představoval opět balíček mongodb-memory-server, jehož verze 6.10 byla podle autorů vydána omylem. Knihovna byla proto zafixována na verzi 6.9.6. Po těchto aktualizacích byl výsledek znovu podroben testům.
Tato aktualizace zásadně inovovala technologie, na kterých Private API závisí. V důsledku aktualizace byl počet zranitelností snížit z 608 na 88. To se stále může zdát jako mnoho, ale drtivá většina těchto zranitelností je v knihovnách, na kterých závisí testovací framework *Jest*. Tato část

Private API tedy existuje jen ve vývojovém prostředí a nenachází se v produkční verzi API. Navíc díky tomu, že se Jest podařilo aktualizovat na nejnovější verzi, by měla být aktualizace v případě, že správci frameworku vydají aktualizaci adresující tyto zranitelnosti, podstatně jednodušší.

4.3 Úpravy Private API

Tato sekce se zabývá návrhem úpravami, které bylo nutné provést v Private API, aby vyhovovalo požadavkům projektu Starý Most

4.3.1 Podpora fotografií

Do databáze byly přidány nové tabulky pro podporu fotografií. Hlavní je tabulka photo, která obsahuje základní o souboru fotografie. Tato tabulka obsahuje jméno souboru, velikost souboru, datum nahrání, odkaz pro získání souboru a id uživatele, který fotografii nahrál.
Dále byly přidány vazební tabulky PhotoArea a PhotoStructure, které reprezentují vazbu fotografie na struktury nebo oblast a obsahují identifikátory oblasti a struktury.
V API byly vytvořeny nové endpointy pro správu fotografií, tedy jejich nahrávání a mazání.

GET /photos – vrátí pole všech fotografií v systému.

GET /photos/{id} – vrátí soubor fotografie, lze využít jako url při zobrazení fotografií na webové stránce.

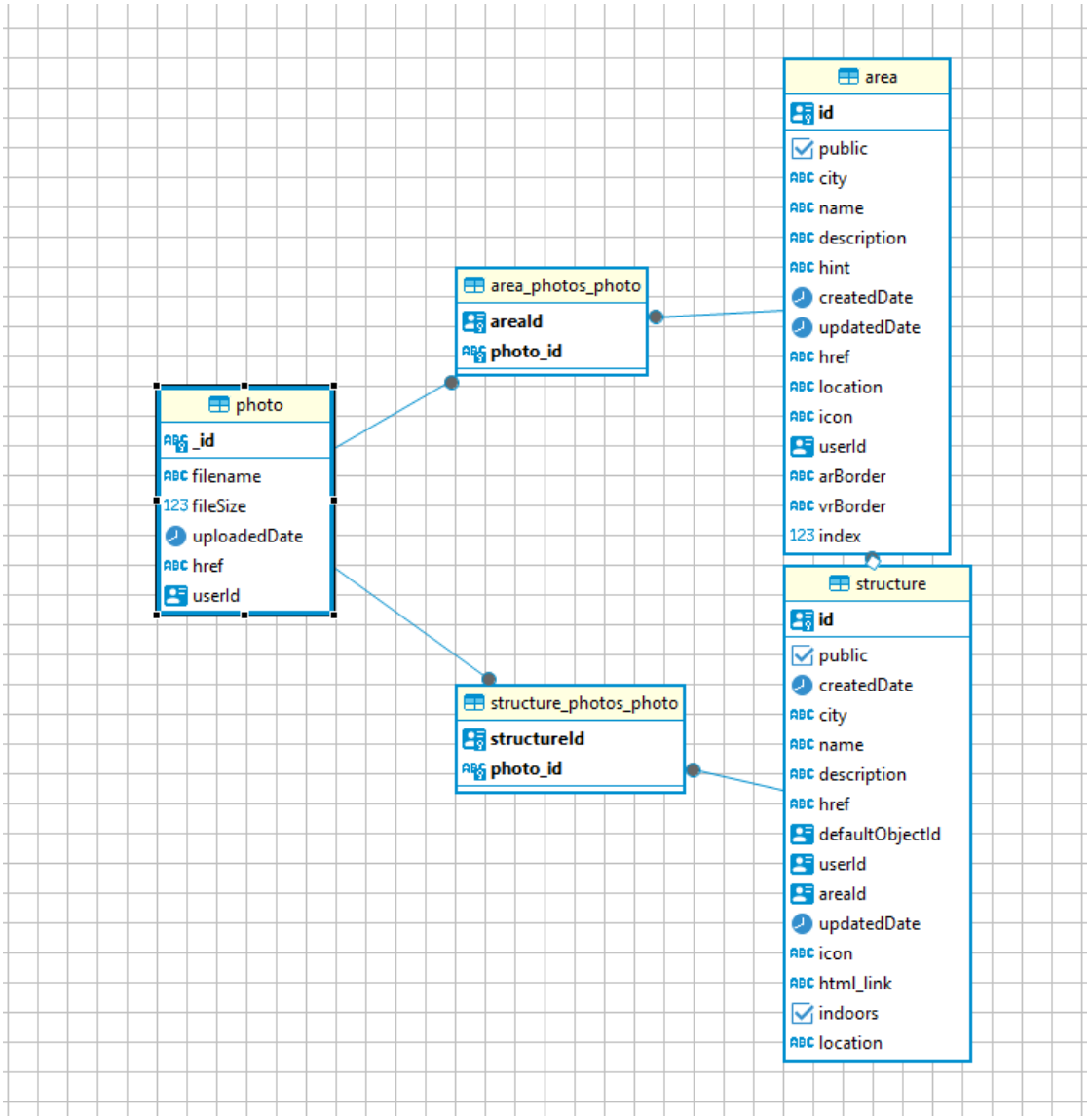
POST structures/{id}/photos – nahraje fotografii do systému a přiřadí ji ke struktuře.

POST areas/{id}/photos – nahraje fotografii do systému a přiřadí ji k ulici.

DELETE structures/{id}/photos/{id} – odstraní fotografii ze struktury a pokud není vázaná na jiný objekt, odstraní ji i ze systému.

DELETE – odstraní fotografii z ulice a pokud není vázaná na jiný objekt, odstraní ji i ze systému.

Entity area a structure nyní obsahují pole fotografií, které obsahuje odkazy na jednotlivé fotografie. Tato data lze získat požadavkem GET na seznam oblastí, struktur nebo požadavkem na konkrétní oblast nebo strukturu.



Obrázek 4.1 Databázové schéma fotografií

4.3.2 Oprava chyb

Nejprve bylo nutné opravit automatizované testy a seeding. Zde byl problém v tom, že testy byly zanedbané a již neodpovídaly aktuální specifikaci API. Testy byly opraveny tak, aby odesílaly požadavky ve správném formátu a rovněž očekávaly správný formát. Další problém spočíval v tom, že některé asynchronní testy byly špatně implementované. Testovací framework *Jest* u nich očekává *Promise*, ale tyto test využívaly callback done. To není v novějších verzích tohoto frameworku podporováno.

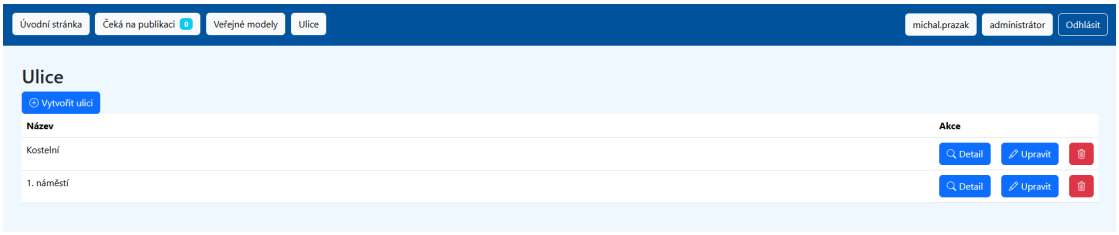
Byly opraveny drobné chyby ve validaci a chyby, které vznikly při migraci na novější verze Typescriptu. Zde se jednalo především o chyby v povinnosti parametru.

Dále byla opravena chyba mazání 3D objektů, kvůli které nebylo možné smazat 3D objekt, který byl označený jako výchozí pro danou strukturu. Také byla opravena chyba, kde API u modelů a 3D objektů vůbec neuvádělo formát souboru. API sice vyžadovalo při nahrávání souboru specifikovat v requestu také formát souboru, ale tuto informaci nikam neukládalo. API bylo tedy upraveno tak, aby tuto informaci ukládalo do tabulky TDObject a poskytovalo ji jako součást entity TDObject. Před touto změnou musely klientské aplikace rozpoznávat formát souboru samostatně.

4.4 Frontend

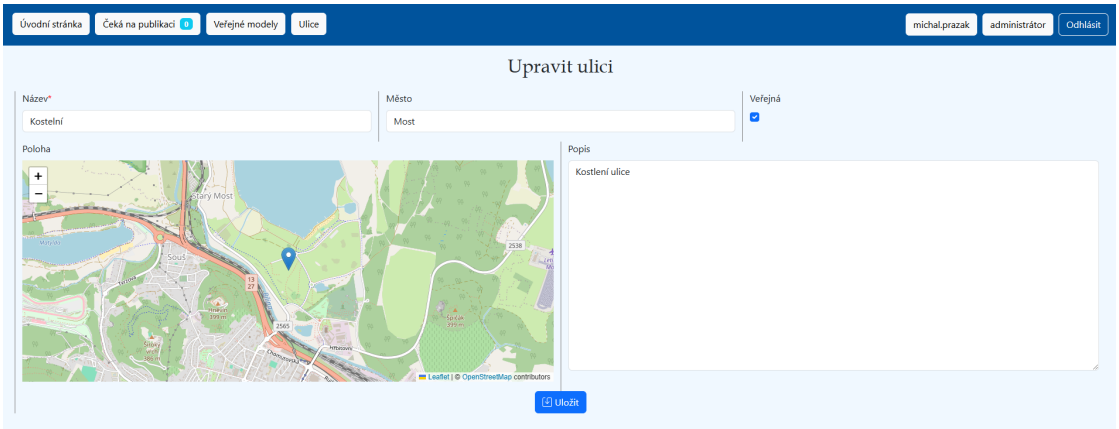
4.4.1 Nové stránky

Seznam ulic slouží jako hlavní bod přístupu k novým funkcionalitám uživatelského rozhraní. Zobrazuje seznam ulic evidovaných v systému a umožňuje administrátorovi vytvořit novou ulici, upravit nebo odstranit existující ulici a zobrazit její detail.



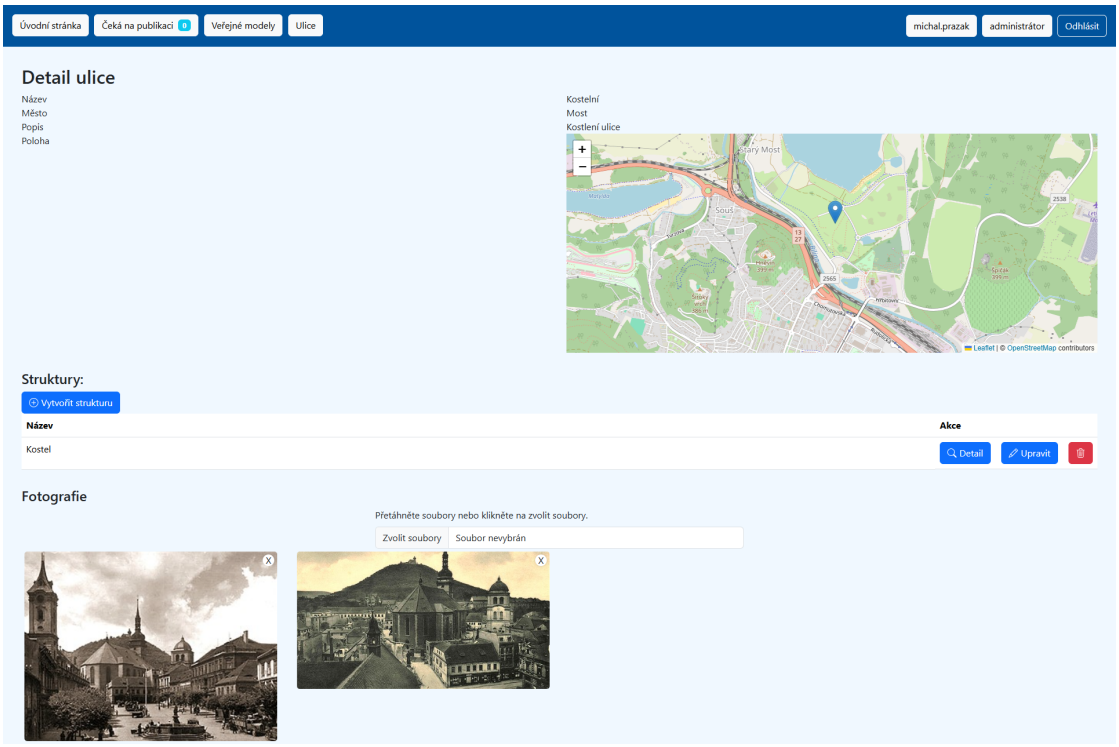
Obrázek 4.2 Stránka seznamu ulic

Tvorba a úprava ulice umožňuje vytvořit novou ulici zadáním základních informací o ulici, kterými jsou název, město, přepínač, jestli má být ulice veřejná, poloha, kterou uživatel vybere na interaktivní mapě a popis. Interaktivní mapa je realizována pomocí komponenty Leaflet, konkrétně její implementaci přímo pro framework React. Jako mapové podklady jsou použity OpenStreetMap. Tato komponenta ale podporuje jen zeměpisnou šířku a délku, zatímco API očekává zeměpisnou šířku, délku a nadmořskou výšku. Nadmořská výška se proto získává z otevřeného API pomocí vybraných zeměpisných souřadnic při kliknutí na tlačítko uložit. Toto API poskytují přímo služba OpenStreetMap.



Obrázek 4.3 Stránka vytvoření a úpravy ulice

Detail ulice zobrazuje informace spjaté s ulicí. Dále obsahuje seznam struktur, které se v této ulici nacházejí. Pod seznamem struktur se nachází pole pro nahrávání fotografií a interaktivní galerie.

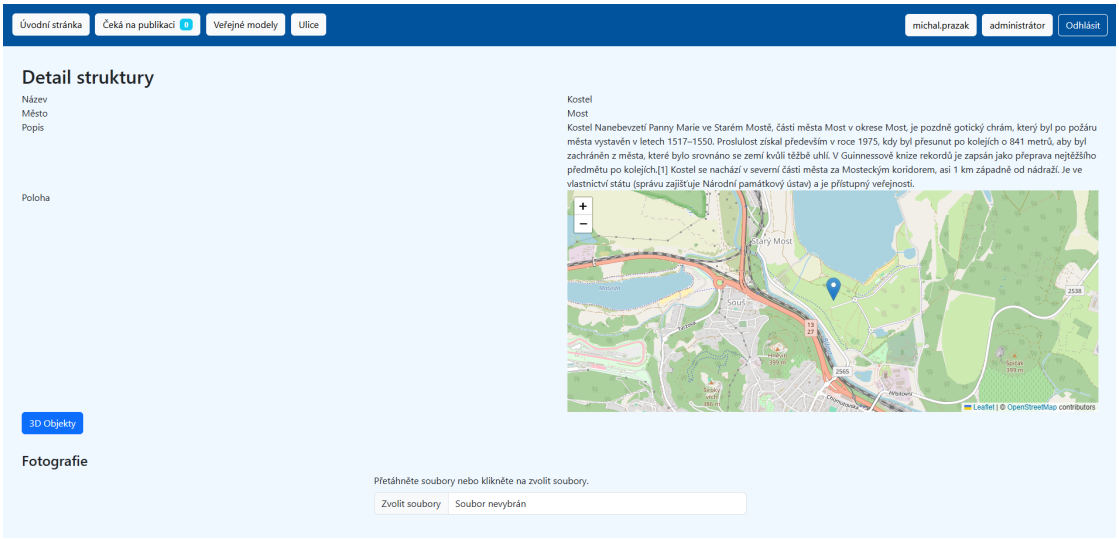


Obrázek 4.4 Stránka detailu ulice

Seznam struktur nabízí možnost vytvořit novou strukturu ve vybrané ulici. U existujících struktur nabízí možnost tyto struktury upravit, odstranit nebo zobrazit jejich detail.

Tvorba a úprava struktury jsou stránky velice podobné tvorbě a úpravě ulice. Umožňují vyplnit základní informace o struktuře. Stejně jako u ulice je formulář při úpravě existující struktury předvyplněný.

Detail struktury zobrazuje podobně jako detail ulice informace o struktuře. Oproti detailu ulice se zde nenachází už žádný další seznam. Na této stránce je tlačítko pro přechod na stránku se seznamem 3D objektů a fotogalerie.



Obrázek 4.5 Stránka detailu struktury

Seznam 3D objektů poskytuje seznam všech 3D objektů přiřazených ke konkrétní struktuře, včetně minulých verzí. Na této stránce lze vytvářet nové 3D objekty a u existujících objektů nabízí možnost prohlédnout si model ve 3D, zobrazit jeho detail a upravit nebo odstranit jej. Pro stránky detailu a úprav byly využity již existující komponenty pro tvorby 3D objektů. Pro potřeby této stránky byly komponenty mírně upraveny, aby přijímaly existující strukturu jako parametr, protože jejich původní verze vyžadovala vybrání struktury ze seznamu přímo v této komponentě.

Obrázek 4.6 Stránka se seznamem 3D objektů vybrané struktury

Galerie

Byly implementovány komponenty pro interaktivní galerii, které pracují s daty z nově implementovaného API. Galerie se skládá ze čtyř komponent:

Nahrávací komponenta umožňuje uživateli vybrat fotografie nebo je přetáhnout do nahrávacího pole. Po vybrání fotografií zobrazí náhledy s křížky pro odstranění fotografie z výběru a tlačítko „Nahrát“, které fotografie nahraje do datového úložiště. Po nahrání se fotogalerie obnoví, aby obsahovala nově nahrané fotografie. Aby byla tato komponenta univerzální, vyžaduje jako parametr funkci, která očekává jako parametr fotografii a postará se o nahrání fotografie na server.

```
<PhotoUpload uploadPhoto={uploadPhoto}  />
```

Výpis kódu 6 Použití nahrávací komponenty

```
const uploadPhoto = async (photo: any) => {
 if (!user || !street) return
 await uploadPhotoToApi(user.token, parentEntityId, photo)
 reload()
}
```

Výpis kódu 7 Ukázková funkce pro nahrávání fotografie

Komponenta galerie se stará o zobrazení náhledů fotografií. U každého obrázku se v pravém horním rohu nachází tlačítko pro odstranění fotografie. Po kliknutí na toto tlačítko se systém nejprve uživatele zeptá jestli si přeje fotografii smazat. Pokud uživatel volbu potvrdí, systém obrázek odstraní. Kliknutím na náhled obrázku se fotografie zvětší na celoobrazovkovou velikost. Tato komponenta očekává jako parametry pole informací o fotografiích, které standardně vrací API při požadavku na konkrétní strukturu nebo ulici, a funkci, která slouží pro odstranění fotografie ze systému. Tato očekává jako parametr identifikátor fotografie.

```
<PhotoGallery photos={street.photos} deletePhoto={deletePhoto} />
```

Výpis kódu 8 Vyvolání komponenty galerie

Authorized Photo je komponenta, která slouží pro načtení a zobrazení obrázku. V případě Private API vyžadují všechny fotografie přístup s autorizačním tokenem. Toho ale není možné dosáhnout pomocí standardních html5 komponent. Tato komponenta přistoupí na adresu obrázku s tokenem, načte ho a následně zobrazí. Tato komponenta je již součástí galerie, ale je možné jí využít i mimo ni. Komponenta očekává URL obrázku, alternativní text, přístupový token, identifikátor obrázku a funkci pro mazání stejnou jako očekává galerie.

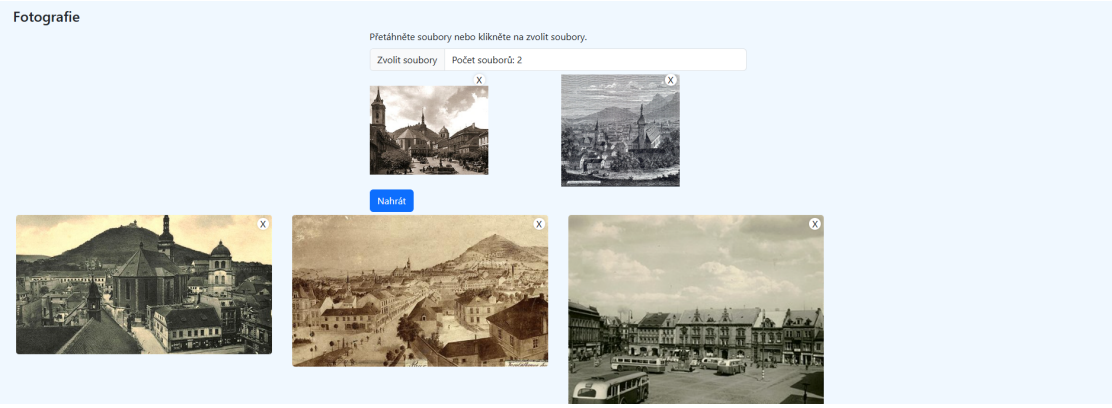
```
<AuthorizedImage
 src={photo.href}
 alt={photo.filename}
 token={currentUser.token}
 photoId={photo.id}
 deletePhoto={deletePhoto}
/>
```

Výpis kódu 9 Vyvolání komponenty galerie

Lightbox slouží k zobrazení fotografie přes celou obrazovku. Po otevření překrývá aktuální stránku a stejně jako Authorized Photo je již součástí galerie. Jako parametry očekává stavovou proměnnou *show*, která indikuje jestli je lightbox zobrazený nebo skrytý. Dále očekává funkci, která se zavolá po kliknutí mimo obrázek. Tato funkce slouží ke skrytí lightboxu a měla by změnit stav show na false. Jako zdroj dat (src) neočekává URL, ale přímo data obrázku.

```
<PhotoLightbox show={show} onHide={closeModal} src={imageData} alt={alt} />
```

Výpis kódu 10 Vyvolání komponenty PhotoLightbox



Obrázek 4.7 Fotogalerie

Obrázek 4.8 Zvětšený obrázek fotogalerie

4.4.2 Další změny

Byl přidán dialog pro potvrzení smazání, který v systému úplně chyběl. Nyní je použit při mazání ulic, struktur a fotografií, ale je univerzální a jeho použití lze rozšířit i do dalších součástí systému.

Dále byla přidána 3D komponenta pro zobrazení 3D modelu přímo na webu, která byla analyzována dříve v této práci. Původní aplikace sice také umožňovala zobrazení 3D modelu, ale využívala k tomu průvodce z projektu Věnná města českých královen, kterého zobrazovala pomocí iframe. V projektu Starý Most nelze tohoto průvodce použít. Navíc byl tento průvodce napojen přímo na produkční databázi Věnných měst a to znemožňovalo jeho použití ve vývojovém prostředí nebo odnoží pracujících s jinou databází.

Byly upraveny komponenty pro práci s modely, protože původně nebyly zaměřeny na práci s modely přímo ve strukturách. Proto nyní přijímají nepovinný parametr s identifikátorem struktury. Pokud bude tento parametr vyplněn, komponenty automaticky přiřadí modely ke struktuře a nenechají uživatele vybrat jinou strukturu. Pokud parametr vyplněn není, chová se komponenta jako před implementací této změny. Dále byly některé třídy v kódu rozšířeny o další parametry potřebné parametry a byly přidány nové entity pro práci s daty z API, jako například entita IStreet pro práci s ulicemi. Datová vrstva aplikace byla rozšířena o práci s těmito entitami.

Původní aplikace při práci s formuláři parsuje submit eventy. Jako lepší přístup se ale jeví použití knihovny react-hook-form, která umožňuje validaci a práci s formuláři pomocí React hooku, což se v prostředí frameworku React velmi dobře integruje s ostatními součástmi komponent. Tento přístup výrazně zpřehledňuje výsledný kód a aplikace je tak jednodušší na údržbu.

Kapitola 5
Testování

5.1 Integrační testování

Integrační testy jsou automatizované a lze je spustit pomocí příkazu *yarn test* přímo v docker kontejneru Private API. Jak bylo již zmíněno dříve, testy byly po převzetí projektu ve špatném stavu a neodpovídaly současnému stavu API. Testy proto byly opraveny, aby reflektovaly současný stav a odpovídaly očekávanému rozhraní API. Toto bylo velmi užitečné i při aktualizaci balíčků a opravách v projektu, kde testy sloužili k ověření, že API zůstává nezměněné.

Testy se nacházejí ve složce *test*, kde každý soubor odpovídá jednomu *test suite*. Testy probíhají v databázi test, která je oddělená od hlavní databáze. Obsah této databáze se vždy před spuštěním každého test suite smaže a provede se migrace. Tím je zajištěna nezávislost na testovacím prostředí a pořadí běhu testů.

Všechny tyto testy jsou také spouštěny automaticky v gitlab pipeline pomocí Continuous Deployment. Tím je zajištěno že vývojář je po každé změně informován o kompatibilitě API a případných chybách.

5.2 Uživatelské testování

Uživatelské testování bylo prováděno na uživatelské části webového systému. Testování se účastnilo 5 testerů v individuálních hovorech pomocí platformy *Discord*. Testeři postupovali podle předem připraveného scénáře a po provedení testování zodpovídali na předem připravené otázky. Poté mohli vznášet osobní připomínky k aplikaci.

5.2.1 Testovací scénář

1. Přihlašte se pomocí svého Google účtu do systému.
2. Přejděte do seznamu ulic a vytvořte novou ulici s libovolným názvem a polohou.
3. Přejděte na detail nově vytvořené ulice a nahrajte k ní libovolný počet obrázků.
4. Vyberte si jeden nahraný obrázek, ten zobrazte (zvětšete) a poté ho odstraňte.
5. V této ulici založte novou strukturu s libovolným názvem.

6. Vytvořte v této struktuře nový model.

7. Nově vytvořený model si prohlédněte ve 3D.

8. Stáhněte soubor tohoto modelu na svůj počítač.

9. Model odstraňte.

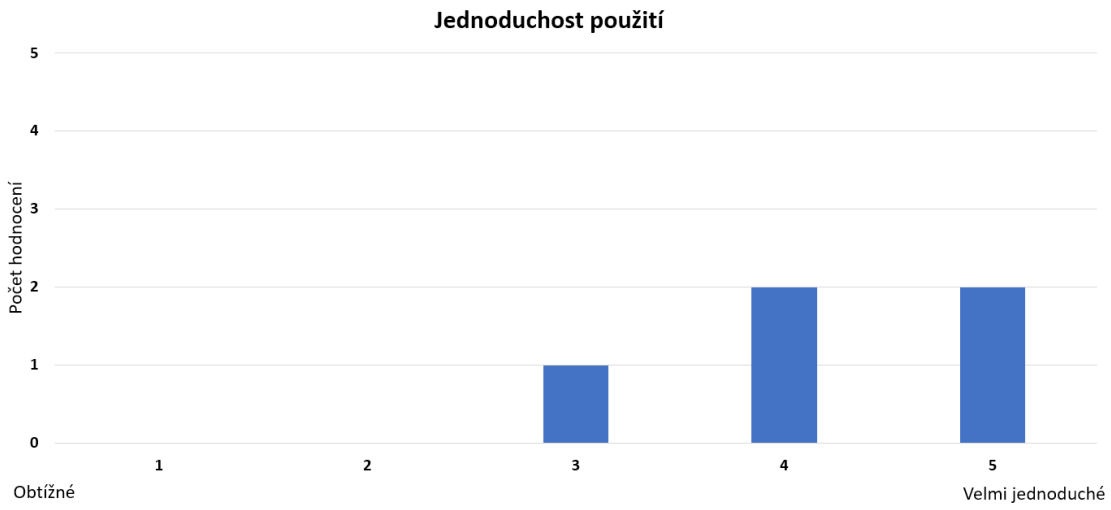
10. Odstraňte i strukturu nadřazenou tomuto modelu.

11. Přejděte zpět na seznam ulic a odhlaste se.

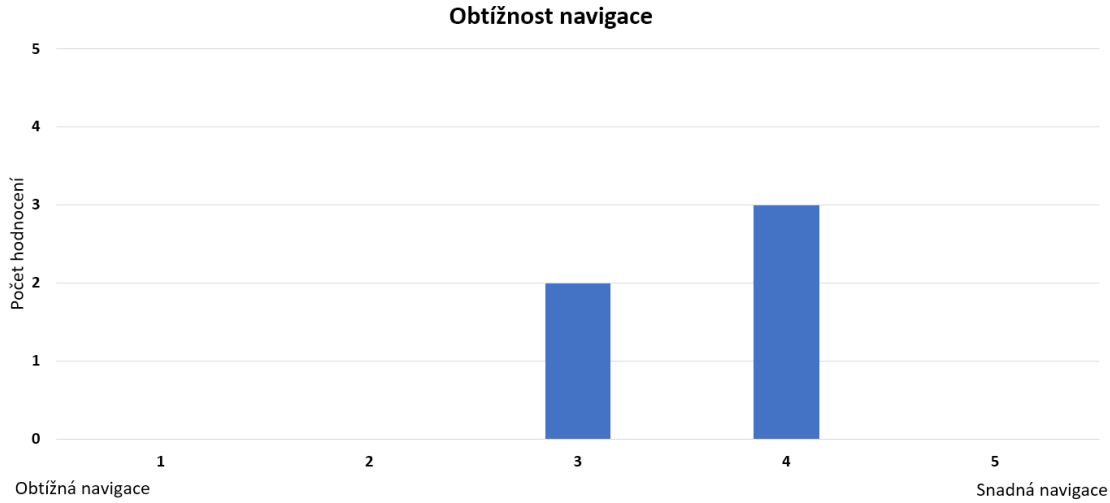
5.2.2 Výsledek testování

Po dokončení testovacího scénáře odpovídali testeři na tyto otázky. Aplikaci dávali známku mezi 1 a 5, kde 5 je nejlepší možná známka.

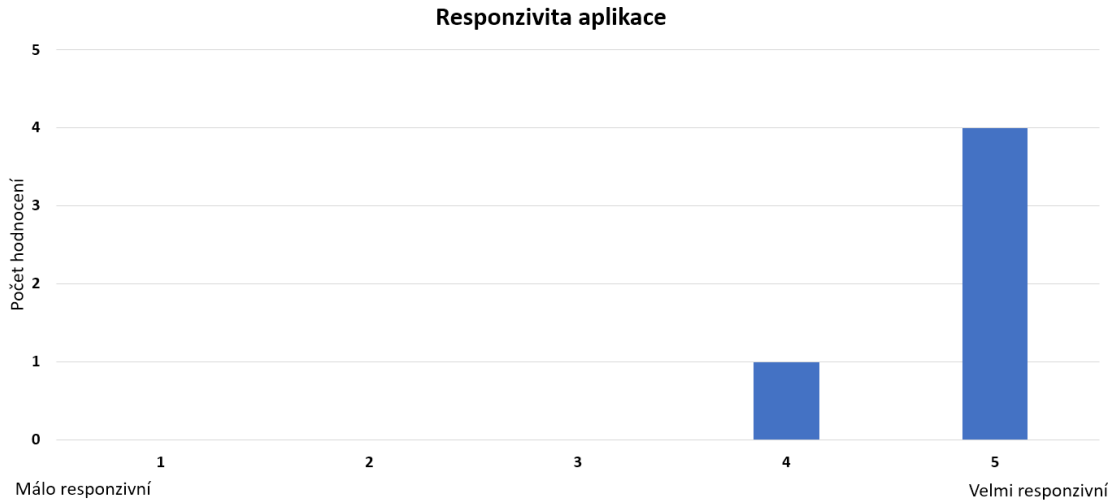
- Jak jednoduchá Vám přišla aplikace na použití?
- Jak jednoduchá Vám přišla navigace v aplikaci?
- Jak responzivní (rychlá) Vám aplikace přišla?



Obrázek 5.1 Graf hodnocení jednoduchosti použití



Obrázek 5.2 Graf hodnocení jednoduchosti navigace v aplikaci



Obrázek 5.3 Graf hodnocení responzivnosti aplikace

Poznatky testerů

V průběhu testování se u jednoho testera stalo to, že aplikace při vytváření nové ulice na chvíli zamrzla. Bylo zjištěno, že toto je způsobeno čekáním na odpověď API OpenStreetMap, které poskytuje ke geolokačním údajům nadmořskou výšku. Na tuto stránku byl přidán indikátor načítání, aby uživatele informoval o probíhajícím načítání. Dále byla nalezena chyba při odstraňování ulic a struktur způsobená výchozím modelem. Jak již bylo popsáno v kapitole realizace, tato chyba byla opravena. Zároveň bylo přidáno varování při odstraňování ulic a struktur s vnořenými objekty. Smazání takového objektu by mohlo způsobit nenávratnou ztrátu velkého množství dat. Z bezpečnostních důvodů bylo proto chování aplikace změněno tak, že je nejdříve nutné vnořené objekty manuálně odstranit.

5.3 Akceptační testování

Toto testování bylo provedeno s vedoucím práce a probíhalo tak, že autor předváděl funkcionality aplikace a tím demonstroval splnění funkčních požadavků. Vedoucí práce na základě prezentace rozhodl u každého funkčního požadavku, zda byl splněn, či ne.

F1.1: registrace	splněno
F1.2: přihlášení	splněno
F2.1: vytvoření ulice	splněno
F2.2: úprava a smazání ulice	splněno
F3.1: množina struktur v ulici	splněno
F3.2: vytvoření, úprava a smazání struktury	splněno
F4.1: tvorba 3D objektů	splněno
F4.2: verzování 3D objektů	splněno
F5.1: nahrávání fotografií	splněno
F5.2: komponenta galerie	splněno
F6: přístupné pouze pro roli administrátor	splněno

Tabulka 5.1 Tabulka pokrytí funkčních požadavků

Kapitola 6
Příručky

6.1 Instalační příručka

6.1.1 Private API

1. Nejprve je nutné naklonovat repozitář naklonovat repozitář z fakultního gitlabu pomocí příkazu *git clone*. Repozitáře jsou přístupné ve skupině *Věnná Města Českých Královen* v podskupině *Starý Most*.

2. V konfiguračním souboru *.env*, který se nachází ve složce *config* je třeba nastavit port na kterém se aplikace spustí, prostředí, které bude použitu (development nebo production), přístupové údaje k databázi, JWT secret, Google Client ID a Auth0 PEM URL. Tyto údaje lze získat registrací aplikace do Google OAuth pomocí Google Cloud Console a Auth0. Auth0 lze obejít zakomentováním *auth0Config* v souboru *AuthController* ve složce *src/controllers*, tímto se lze vyhnout konfiguraci Auth0, pokud to není nutné. Tento postup ale není doporučený

3. Dále je třeba mít nainstalovaný Docker a *docker compose*

4. Nyní lze vytvořit a spustit kontejnery s Private API pomocí příkazu *docker compose up*. Kontejnery se automaticky vytvoří a spustí, dojde ke kompilaci API a to se následně spustí. Kontejnery lze také spustit v detached režimu, kdy nejsou závislé na konzoli pomocí *docker compose up -d*.

5. API by mělo být nyní přístupné na zvoleném portu. Pro zpřístupnění API lze využít nginx reverse proxy, ale lze ho také zpřístupnit přímo.

6. Logy API mohou být zobrazeny pomocí *docker logs vmck-core-private-api* a pro přístup k příkazové řádce kontejneru lze použit *docker exec -it vmck-core-private-api <příkaz>*

6.1.2 Uživatelské rozhraní

1. Prvním krokem je naklonování repozitáře z fakultního gitlabu pomocí příkazu *git clone*. Repozitáře jsou stejně jako v případě Private API přístupné ve skupině *Věnná Města Českých Královen* v podskupině *Starý Most*.

2. Pro další kroky je nutné mít nainstalované běhové prostředí NodeJS ve verzi alespoň 20.

3. Dále je nutné nainstalovat závislosti aplikace spuštěním příkazu *npm install* v naklonované složce.

4. Konfigurační soubor *env.ts* se nachází ve složce *src*. Zde je nutné nastavit Google Client ID a API URL.
   Ve výchozím nastavení používá proměnné prostředí *REACT_APP_GOOGLE_CLIENT_ID* a *REACT_APP_PRIVATE_API_URL*, ale tyto údaje je také možné zapsat přímo do *env.ts*.

5. Nyní lze aplikaci spustit ve vývojovém prostředí pomocí příkazu *npm run start* nebo sestavit pomocí *npm run build*. Sestavení vyprodukuje statické html a js soubory, které lze nasadit na libovolný webový server. Doporučený a otestovaný server je nginx.

6.2 Programátorská příručka

6.2.1 Private API

Vstupním bodem pro aplikaci je soubor *app.ts*, kde jsou zaregistrovány základní služby a middleware pro autentifikaci. Také jsou tu zaregistrovány controllery a port, na kterém API běží. V souboru *server.ts* jsou vytvořeny připojení k databázím a spuštěn http server. TypeORM entity se nachází ve složce *entity*. Aplikace pro přístup k datům využívá modely, které se nacházejí ve složce *models*. Nejdůležitější je složka *controllers*, kde se nacházejí controllery, které obsluhují HTTP požadavky. Zde se jednotlivé metody mapují na cesty pomocí anotace *@Route*.

6.2.2 Uživatelské rozhraní

Obecné cesty jsou definovány v souboru *App.tsx*. Stránky uživatelského rozhraní a specifické cesty se nacházejí ve složce *routes*. Tyto stránky využívají komponenty ze složky *components*. Komponenty obsahují většinu funkcionalit aplikace, jako například galerie, vytvoření a úprava 3D modelů a tabulka se seznamem struktur. S API aplikace komunikuje pomocí API fetchers, které se nacházejí ve složce *utils*. Zde se také nacházejí další nástroje například pro parsování dat. Data jsou předávána mezi komponentami pomocí interfaců, které jsou definované ve složce *interfaces*.

6.3 Uživatelská příručka

Jako uživatelská příručka k uživatelskému rozhraní v této práci slouží rozbor stránek z kapitoly realizace (4.4). V této kapitole jsou popsány všechny nové stránky a jejich funkcionality. Dále je možné získat povědomí o tom, jak aplikace funguje a kde se co nachází pomocí diagramu mapy stránek 3.8.

Kapitola 7
Závěr

Tato práce se zabývala návrhem a implementací webového systému pro správu dat projektu Starý Most, jehož cílem je fungovat jako datové úložiště, nabídnout webové rozhraní pro přístup k těmto datům pro ostatní aplikace v podobě REST API a umožnit spolupráci historiků, modelářů a grafiků.

Na začátku práce byla provedena analýza projektu Věnná města českých královen, která odhalila možnou využitelnost aplikací, které vznikly v rámci tohoto projektu. Tato analýza ukázala, že znovupoužitelnost je zejména u novějších aplikací z tohoto projektu vysoká.

Poznatky z analýzy tvořili důležitý vstup pro kapitolu Návrh, kde bylo rozhodnuto, že bude webový systém pro správu dat projektu Starý Most založen na Private API z projektu VMČK. Pro toto API byl navržen modul pro správu fotografií a jejich přiřazení k oblastem a strukturám. Dále bylo rozhodnuto, že administrační rozhraní bude založeno na aplikaci pro schvalování 3D modelů pro projekt VMČK, protože tato aplikace již implementuje některé užitečné komponenty a komunikační vrstvu pro Private API.

V kapitole Realizace byl nejprve zprovozněn CI/CD pipeline a opravené automatizované testy. Poté byly všechny součásti Private API aktualizovány, což zásadně snížilo počet zranitelností v tomto projektu. Dále byly opraveny identifikované chyby a implementovány navrhované změny. V uživatelském rozhraní byly implementovány nové stránky pro správu ulic, struktur, fotografií a 3D modelů. Výsledky práce byly podrobeny integračním, uživatelským a akceptačním testům.

Dle autora byly cíle této práce splněné. Systém umožňuje vkládat a spravovat ulice, struktury, 3D modely a fotografie. Dále je možné pro další vývoj využít i schvalovacích procesů z projektu VMČK. Nové webové uživatelské rozhraní umožňuje s těmito daty jednoduše pracovat. Systém je navíc zpětně kompatibilní s projektem VMČK, což otevírá možnost aktualizovat i API používané ve VMČK nebo využít další aplikace původně vytvořené pro tento projekt i v projektu Starý Most.

Příloha A

Odkazy

Zdrojový kód Private API je dostupný na fakultním GitLabu FIT ČVUT:
`https://gitlab.fit.cvut.cz/vmck/star-most/private-api`

Zdrojový kód webového rozhraní je rovněž přístupný na GitLabu FIT ČVUT:
`https://gitlab.fit.cvut.cz/vmck/star-most/management-frontend`

Bibliografie

1. *Starfos | Most – město, které nezaniklo* [online]. 2023. [cit. 2024-04-15]. Dostupné z: `https://starfos.tacr.cz/cs/projekty/DH23P03OVV048`.
2. PRAŽÁK, Michal. *Konzultace s Ing. Jiřím Chludilem* [osobní konzultace]. 2024.
3. GRULICH, Petr et al. *Věnná města českých královen* [online]. 2023. [cit. 2024-05-10]. Dostupné z: `https://www.kralovskavennamesta.cz`.
4. *Docker overview* [online]. 2024. [cit. 2024-04-15]. Dostupné z: `https://docs.docker.com/get-started/overview/`.
5. *Node.js — About Node.js* [online]. 2024. [cit. 2024-04-15]. Dostupné z: `https://nodejs.org/en/about`.
6. *TypeScript Introduction* [online]. 2024. [cit. 2024-04-15]. Dostupné z: `https://www.w3schools.com/typescript/typescript_intro.php`.
7. *Express - Node.js web application framework* [online]. 2024. [cit. 2024-04-15]. Dostupné z: `https://expressjs.com/`.
8. *Case study: How & why to build a consumer app with Node.js | VentureBeat* [online]. 2012. [cit. 2024-04-15]. Dostupné z: `https://venturebeat.com/dev/building-consumer-apps-with-node/`.
9. *JSON Web Token Introduction – jwt.io* [online]. 2024. [cit. 2024-04-15], Dostupné z `https://jwt.io/introduction`.
10. SIVÁK, Dominik. *Věnná města českých královen - Backend administrační části*. Diplomová práce. České vysoké učení technické v Praze, Fakulta informačních technologií, 2021.
11. *PostgreSQL: About* [online]. 2024. [cit. 2024-04-15], Dostupné z: `https://www.postgresql.org/about/`.
12. *Stack Overflow Developer Survey 2023* [online]. 2023. [cit. 2024-04-15], Dostupné z: `https://survey.stackoverflow.co/2023/#most-popular-technologies-database`.
13. GILLIS Alexander S; BOTELHO, Bridget. *What is MongoDB? Features and how it works – TechTarget Definition* [online]. 2024. [cit. 2024-04-15], Dostupné z: `https://www.techtarget.com/searchdatamanagement/definition/MongoDB`.
14. VANČURA, Daniel. *Věnná města českých královen – jádro*. Bakalářská práce. České vysoké učení technické v Praze, Fakulta informačních technologií, 2020.
15. OPÁLKOVÁ, Tereza. *Věnná města českých královen – Webový klient pro schvalování a publikaci 3D modelů II*. Bakalářská práce. České vysoké učení technické v Praze, Fakulta informačních technologií, 2023.

16. DBEAVER TEAM. *DBeaver* [soft.]. 2024. [cit. 2024-04-15], Dostupné z: https://dbeaver.com/.
17. SIVÁK Dominik; Chludil, Jiří. *Private API for the project Věnná Města Českých Královen* [online]. 2023. [cit. 2024-04-15], Dostupné z: https://vmck-dev.sic.cz:3003/.
18. MARTÍNEK, Michal. *Administrační rozhraní pro projekt Věnná města českých královen.* Diplomová práce. České vysoké učení technické v Praze, Fakulta informačních technologií, 2020.
19. NYPL, Tomáš; OPÁLKOVÁ, Tereza; PRAŽÁK, Michal; MARTIN, Ševela; HIEU, Ta Minh. *Softwarový týmový projekt 1.* České vysoké učení technické v Praze, Fakulta informačních technologií, 2022. [cit. 2024-04-15], Dostupné z: https://gitlab.fit.cvut.cz/vmck/web-manager.
20. ANTOŠ, Pavel. *Webová komponenta pro interaktivní vizualizaci 3D modelů.* Diplomová práce. České vysoké učení technické v Praze, Fakulta informačních technologií, 2023.
21. *Developer Mapy.cz* [online]. 2024. [cit. 2024-04-15], Dostupné z: https://developer.mapy.cz/.
22. *Google Maps Platform* [online]. 2024. [cit. 2024-04-15], Dostupné z: https://mapsplatform.google.com/.
23. *Google Maps Platform Terms Of Service* [online]. 2024. [cit. 2024-04-15], Dostupné z: https://cloud.google.com/maps-platform/terms.
24. *Bing Maps Dev Center* [online]. 2024. [cit. 2024-04-15], Dostupné z: https://www.bingmapsportal.com/.
25. *OpenStreetMap* [online]. 2024. [cit. 2024-04-15], Dostupné z: https://www.openstreetmap.org/copyright.
26. *Leaflet - a JavaScript library for interactive maps* [online]. 2024. [cit. 2024-04-15], Dostupné z: https://leafletjs.com/.
27. *React Leaflet* [online]. 2024. [cit. 2024-04-15], Dostupné z: https://react-leaflet.js.org/.
28. *OpenLayers* [online]. 2024. [cit. 2024-04-15], Dostupné z: https://openlayers.org/.
29. *MapLibre* [online]. 2024. [cit. 2024-04-15], Dostupné z: https://maplibre.org/.
30. KORNIUSHYNA, Hanna. *Věnná města českých královen – Prototyp klienta v herním enginu Unreal Engine 4.* Bakalářská práce. České vysoké učení technické v Praze, Fakulta informačních technologií, 2022.
31. ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE, FAKULTA INFORMAČNÍCH TECHNOLOGIÍ. *DBS Portál* [soft.]. 2024. [cit. 2024-04-15], Dostupné z: https://dbs.fit.cvut.cz/.
32. DIAGRAMS.NET. *Diagrams.net* [soft.]. 2012. [cit. 2024-04-15], Dostupné z: https://app.diagrams.net/.
33. AZZOPARDI, Steve. *Update: Changes to GitLab CI/CD and Docker in Docker with Docker 19.03* [online]. 2019. [cit. 2024-04-23], Dostupné z: https://about.gitlab.com/blog/2019/07/31/docker-in-docker-with-docker-19-dot-03/.
34. *PostgreSQL: Versioning Policy* [online]. 2024. [cit. 2024-04-23], Dostupné z: https://www.postgresql.org/support/versioning/.

Obsah příloh

- `readme.txt` ..... stručný popis obsahu média
- `src`
  - `private-api` ..... zdrojové kódy Private API
  - `management-frontend` ..... zdrojové kódy uživatelského rozhraní
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF