



**MATEMATICKO-FYZIKÁLNÍ
FAKULTA**
Univerzita Karlova

BAKALÁŘSKÁ PRÁCE

David Nápravník

Softwarové řešení digitálních archivů

Katedra teoretické informatiky a matematické logiky

Vedoucí bakalářské práce: Mgr. Kateřina Macková

Studijní program: Informatika (B1801)

Studijní obor: IPSS (1801R048)

Praha 2021

Prohlašuji, že jsem tuto bakalářskou práci vypracoval(a) samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů. Tato práce nebyla využita k získání jiného nebo stejného titulu.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle §60 odst. 1 autorského zákona.

V dne

Podpis autora

TODO Poděkování:
Kateřina Macková
Anička Yaghobová
Monika Bošániová

Název práce: Softwarové řešení digitálních archivů

Autor: David Nápravník

Katedra: Katedra teoretické informatiky a matematické logiky

Vedoucí bakalářské práce: Mgr. Kateřina Macková, Katedra teoretické informatiky a matematické logiky

Abstrakt: Za použití moderních webových a databázových technologií byla vyvinuta aplikace jako náhrada za zastaralé knihovní systémy. Byly sníženy požadavky na výkon serveru a síťového rozhraní. Uživatel tak má intuitivní rychlou webovou aplikaci, jež kombinuje knihovní a redakční systém a další moduly, jež nějakým způsobem zobrazují historické objekty, jako jsou mapy nebo 3D modely. Vývojáři a automatizované systémy pak mají přístup k uchovávaným datům přes jednoduché a rychle API a umožní jim tak data napojit na své systémy.

Klíčová slova: digitální archiv, webová aplikace, databáze

Title: Software solution for digital archives

Author: David Nápravník

Department: Department of Theoretical Computer Science and Mathematical Logic

Supervisor: Mgr. Kateřina Macková, Department of Theoretical Computer Science and Mathematical Logic

Abstract: TODO Abstrakt en

Keywords: digital archive, web application, database

Obsah

1	Úvod	3
2	Analýza a požadavky	4
2.1	Požadavky na systém	4
2.2	Pro koho je systém určen	4
2.3	Existující produkty	5
2.3.1	KOHA	5
2.3.2	Evergreen	5
2.3.3	SLIMS	5
2.3.4	Porovnání existujících systémů	5
3	Návrh projektu	6
3.1	Výběr technologií	6
3.1.1	Frontend	6
3.1.2	Backend	6
3.2	Diagram systému	8
4	Implementace backendu	9
4.1	Server	9
4.2	Knihovny	9
4.2.1	Knihovna express.js	9
4.2.2	Knihovna mongoose	9
4.3	Dokumentace API	10
4.3.1	Vykreslovací engine pro dokumentaci	10
4.4	Routes	10
4.4.1	Uživatel	10
4.4.2	Autorizace	11
4.4.3	Záznam	11
4.4.4	Stránka	11
4.4.5	Nahrávání souborů	11
4.5	Modely	11
4.5.1	Záznam	11
4.5.2	Stránka	11
4.5.3	Uživatel	11
5	Implementace frontendu	13
5.1	Server	13
5.1.1	Kompilace	13
5.1.2	NPM	13
5.2	Knihovny	13
5.2.1	React	13
5.2.2	Material-UI	14
5.2.3	i18n	14
5.2.4	Babel	14
5.2.5	Webpack	14

5.3	Rozhraní	14
5.3.1	Scény	15
5.3.2	Komponenty	21
5.3.3	Moduly	23
5.4	Lokalizace	23
6	Provázání Backendu a Frontendu, API	24
6.1	Dokumentace online	24
6.1.1	Software pro online dokumentaci	25
6.2	Backend	25
6.3	API	25
6.3.1	Autentifikace	25
6.4	Frontend a volání API	26
6.4.1	Fetch	26
6.4.2	Existující programy pro práci a testování API	26
7	Moduly	28
7.1	Přidávání nových modulů	28
7.2	Aktuálně nasazené moduly	28
7.2.1	Modul hologram	28
7.2.2	Prohlížeč map	29
8	Instalace a spuštění	30
8.1	Prerekvizity	30
8.1.1	NodeJS	30
8.1.2	MongoDB	30
8.1.3	Nginx	30
8.1.4	Powershell	31
8.1.5	Chocolatey balík	31
8.2	Zdrojové soubory	31
8.3	Instalace	31
9	Řešení	32
9.1	Výsledný web	32
9.2	Uživatelská dokumentace	32
	Závěr	33
	Seznam použité literatury	34
	Seznam obrázků	35
	Seznam použitých zkratk	36

1. Úvod

Cílem této práce bylo vytvoření webové aplikace pro správu historických objektů jako jsou elektronické záznamy, odborné články, 3D modely a mapy z oblasti Krkonoš.

Systém byl vytvořen tak, aby splnil požadavky ze strany historického ústavu, který poskytl data a zároveň byl efektivnější a rychlejší než stávající knihovní systémy.

Výsledkem je webová aplikace, kterou si můžou návštěvníci zobrazit a vyhledávat v ní žádoucí informace, pročítat tématické články, nebo si zobrazit historické mapy či 3D objekty. Zadavatelé zde mohou pohodlně procházet, zadávat a editovat elektronické záznamy. Redaktoři mohou editovat stránky zobrazované návštěvníky a přidávat aktuality.

2. Analýza a požadavky

Vybíráme jen z open source produktů, abychom mohli nahlédnout do jejich kódu a lépe porozumět implementaci jejich komponent. Lépe se pro takový systém vyvíjejí externí moduly, protože obvykle mají o dost větší komunitu vývojářů a přispěvatelů.

2.1 Požadavky na systém

Systém bude přístupný jako webová aplikace skrze moderní webové prohlížeče. Zadavatel bude moci přidávat, prohlížet, měnit a mazat metadata a k nim příslušné indexy. Redaktor bude moci přidávat a měnit články a novinky na webu. Uživatel bude moci vyhledat záznam podle několika různých kritérií a poté jej zobrazit, též bude moci zobrazit příspěvky na webu, včetně novinek. Externí programátoři budou moci k systému přistupovat přes API a získávat nebo měnit data.

2.2 Pro koho je systém určen

Systém bude sloužit pro pracovníky historického ústavu jakožto pro zadavatele a redaktory stránek této aplikace a bude uchovávat historická data z oblasti Krkonoš, tak aby si je návštěvník případně vědecký pracovník mohl přehledně zobrazit na jednom místě a případně si data i automaticky stahovat přes připravené API. Zároveň systém obsahuje interaktivní moduly, jež budou k dispozici návštěvníkům výstavy Pramenů Krkonoš, jako je hologram 3D modelu nebo prohlížeč historických map.

2.3 Existující produkty

2.3.1 KOHA

Domovská stránka aplikace: <http://www.koha.cz/>
Koha je nejrozšířenější open source knihovní systém s širokou komunitou. Byla vyvinuta na Novém Zélandu roku 2000. Ale je stále udržovaná a stále rozšiřovaná (nejnovější update je z konce roku 2020)
Používá SQL databázi. Je psaná v Perlu, na frontendu využívající JavaScript (ale není jej tolik).



Obr. 2.1 | LOGO systému koha

2.3.2 Evergreen

Domovská stránka aplikace: <https://eg-wiki.osvobozena-knihovna.cz>
Další z řady knihovních systémů. Používán např. Pedagogickou a Teologickou Fakultou v Praze.
Používá SQL databázi, vykreslování probíhá na serveru a nemá uživatelsky přívětivé prostředí.



2.3.3 SLIMS

Domovská stránka aplikace: <https://slims.web.id/>
Systém se základní funkcionalitou a přívětivým vzhledem. Není v češtině. Není primárně určen jako knihovní systém, spíše je to univerzální systém na cokoli, proto není až tak efektivní a jeho nastavování by zabralo mnoho času.



2.3.4 Porovnání existujících systémů

Evergreen a SLIMS jsou systémy, které potřebují silně vyškolenou osobu, aby se o systém starala, narozdíl od systému KOHA, která je intuitivnější a pro nové uživatele přívětivější, při zachování stejné, možná i lepší funkcionality.

Obr. 2.3 | LOGO systému SLIMS

3. Návrh projektu

3.1 Výběr technologií

3.1.1 Frontend

Ačkoliv se trendy v oblasti vývoje webových aplikací mění velmi rychle, jednou z nejpodstatnějších změn, která přenesla vykreslování stránky na stranu uživatele a tím se výrazně odlišila od dosavadních konceptů vykreslujících stránku na straně serveru, se stala technologie **Single page application**. Celá aplikace pak je v tomto duchu implementována a přizpůsobena s důrazem na plynulost a rychlost aplikace.

Single page application

Single page application je technologie umožňující vykreslení jiné stránky, bez nutnosti, posílání requestu na server. Uživatel si při prvním spuštění webu stáhne celý balíček webu a při opětovném načtení většinou sahá jen do své lokální cache. JavaScriptová knihovna (v tomto případě React) poté stránku překresluje při uživatelské interakci. V případě nutnosti stažení / posílání dat mezi serverem a uživatelem (např. editace záznamu, nebo načtení existujícího záznamu) se volá pouze request k API webové služby a tělo requestu obsahuje pouze užitečné (ne-redundantní) informace.

React

Knihovna React poskytuje single page application technologii. Jedná se o dobře udržovanou knihovnu, jež byla vyvinuta Facebookem jakožto náhrada zastaralého konceptu renderování stránky na serveru. Díky tomu servery nemusejí ztrácet výkon s každou změnou na stránce a výkon k renderování se bere z PC uživatele. Jádro této knihovny je velmi dobře optimalizované a poskytuje i řadu debugovacích nástrojů, což je pro větší projekty nepostradatelná výhoda.

Další možné technologie

Běžnou praxí vykreslování dynamické stránky je její vykreslení na straně serveru, jako to má např. velmi populární redakční systém WordPress (jež je psaný v jazyce PHP). Takovýto systém je dobře uživatelsky přívětivý, ale za cenu masivního nárůstu potřebného serverového výkonu. V případě implementace knihovného systému by to znamenalo vykreslovat celou stránku (hlavičku, tělo i zápatí) na serveru, na druhé straně single page application na serveru nic nevykresluje a pouze minimalisticky posílá požadované informace.

3.1.2 Backend

Mít single page aplikaci na frontendu znamená, že na backendu musí existovat API, od kterého bude frontend čerpat data. Navíc zde potřebujeme i systém pro statické odesílání balíku celé webové stránky. V rámci udržitelnosti byl použit

stejný jazyk JavaScript, jaký je na frontendu. Knihovnou, která by umožňovala komplexní správu requestů a zároveň by byla i na robustnějších projektech programátorsky přehledná, byla zvolena Express.js.

Express.js

Express.js poskytuje nejen odesílání statických stránek, což je potřeba při odesílání balíku s aplikací React, ale umí i custom requesty, potřebné pro rozmanité API a také odesílání a lokální ukládání statických souborů, jako jsou obrázky a word nebo pdf dokumenty.

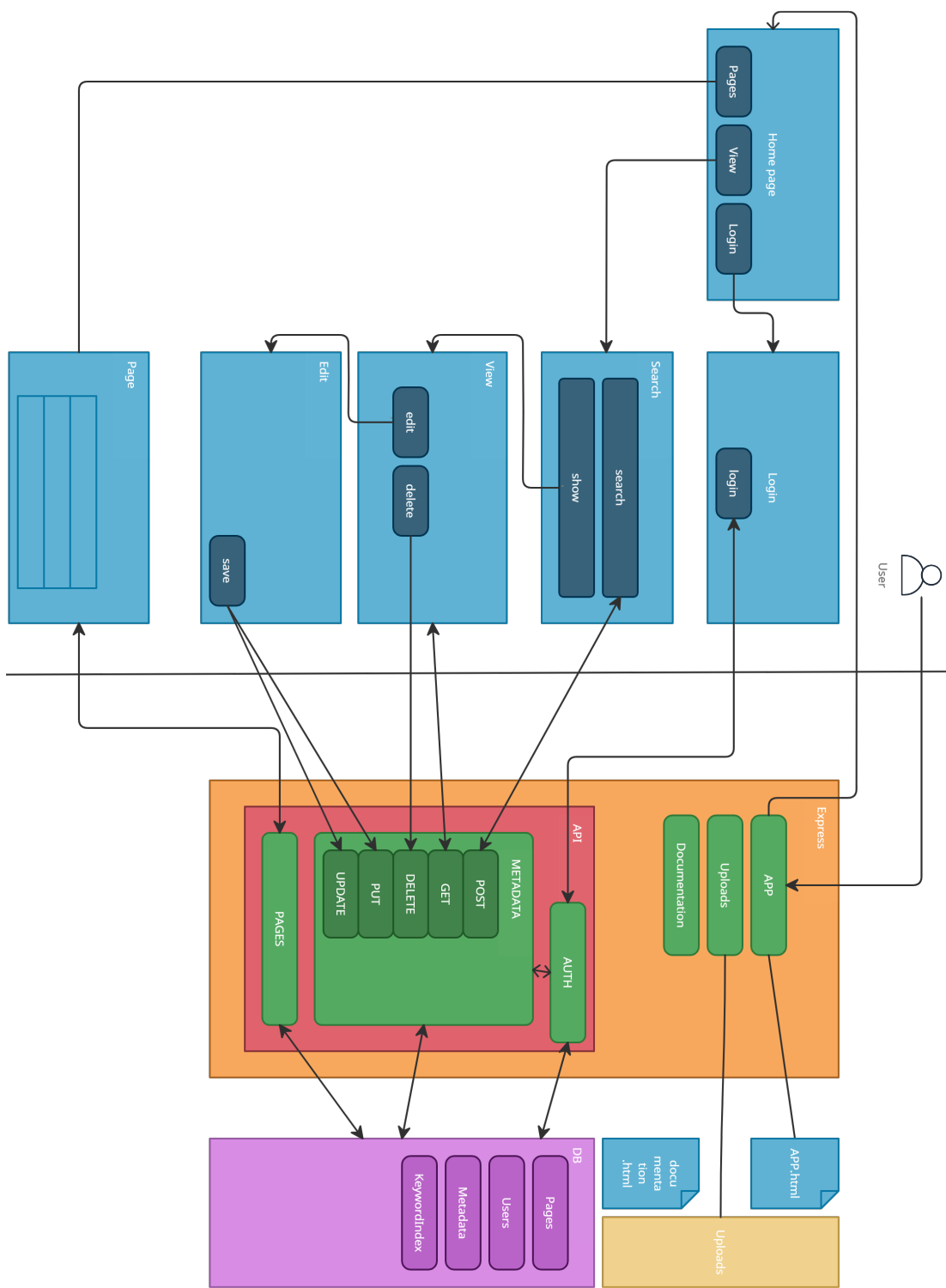
MongoDB

MongoDB je databázový systém typu non-SQL. Což primárně znamená, že data neuchovává v tabulkách, ale v tzv. schématech. Což má mnoho výhod, z nichž největší je, že nekompletní záznamy nezabírají svými nevyplněnými daty místo v DB a ukládá se opravdu jen to, co je potřeba. Další výhodou je styl ukládání dat a komunikace s DB. Databáze si data uchovává ve formátu BSON (binární JSON rozšířený o datové typy). O data si aplikace žádá pomocí query, která je zcela odlišná od těch u SQL databází, primárně se zde neposílá query ve formátu string ale jako JSON objekt, díky čemuž např. nenastane známá SQL injection. Znovu ve formátu JSON poté data vrací aplikaci.

Další možné technologie

Díky oddělení frontendu a backendu (narozdíl od např. WordPressu) je možné na backend nasadit téměř cokoliv co umí posílat requesty. Příkladem toho můžou být scripty v jazycích PHP, C#, Python, nebo Perl. Ale vzhledem k tomu, že jedním z modulů bude neuronová síť na pokročilé vyhledávání, vybírali jsme mezi jazyky Python a JavaScript, jakožto dvěma jazyky, které mají velmi dobré knihovny pro práci s neuronovými sítěmi.

3.2 Diagram systému



Obr. 3.1 | Diagram systému

4. Implementace backendu

Backend je část softwaru, která je umístěna na serveru a uživatel z ní vidí jen výstup, jež dostane. Slouží pro přijímání požadavků od klienta a odesílání případné odpovědi nebo provedení nějaké činnosti bez výstupu.

Součástí backendu je i API, které slouží jako toto spojení mezi serverem a klientem. Dále často obsahuje CRON tabulku, která periodicky vykonává nějaký script.

4.1 Server

Stroj, na kterém „běží“ aktuální verze systému je pod systémem Linux (přesněji Ubuntu). Na virtual machine poskytnuté Matematicko-fyzikální fakultou UK. Databáze je pak umístěna na serveru u Googlu, kvůli snazší konfiguraci. Není však problém kdykoliv tuto DB přesunout na stejný stroj, na kterém běží zbytek backendu a snížit tím prodlevu způsobenou prací s databází.

4.2 Knihovny

Pro rychlý začátek vývoje byla použita knihovna **Express.js**, která umožňuje práci s http requestsy potřebnými pro fungování API a to bez většího množství konfigurace ze začátku.

Pro práci s databází byla použita knihovna **mongoose**, která po rychlé konfiguraci umožňuje práci s databází typu MongoDB.

Dalšími menšími pomocnými knihovnami jsou **cors** (Cross-Origin Resource Sharing) napomáhající s nastavením hlavičky u API requestu, **md5** pro šifrování hesel a hašování dat a nakonec **cookie-parser** zjednodušující práci s cookies.

4.2.1 Knihovna express.js

Jedná se o minimalistickou a zároveň velmi silnou knihovnu poskytující všestranné prostředky pro web. Obsahuje funkce pro jednoduchou správu HTTP metod a díky tomu je vytváření i větších API jednoduché a přehledné. Má velmi dobrý výkon, který se sice nedá srovnávat s knihovnami psanými v jazyce C, ale z JS knihoven je jeden z nejrychlejších.

Nadstavba pro nahrávání souborů

Nadstavba **express-fileupload** umožňuje v těle requestu rozeznat a zpracovat soubor, který se následně pomocí knihovny **fs** ukládá na server, specificky do složky uploads.

4.2.2 Knihovna mongoose

Tato knihovna je ideální a nepostradatelná pro práci s databází typu MongoDB. Poskytuje funkce pro snadné připojení k databázi a komunikaci s ní. Hlavní výhodou této knihovny jsou modely, validace a vestavěné přetypování (JavaScript

je dynamicky typovaný jazyk). S modely se pracuje pomocí tzv. **promise**, která umožňuje řadit akce za sebe, ve stylu:

```
Model.find(body)
  .limit(_limit || 5)
  .exec()
  .then(result => { res.status(200).json(result) })
  .catch(err => { res.status(500).json("something went wrong") })
```

4.3 Dokumentace API

Na adrese <http://quest.ms.mff.cuni.cz/prak/api/documentation> se nachází statický soubor s dokumentací. Celá stránka je zapouzdřena do jediného souboru, který se po načtení vykreslí na straně uživatele, tudíž nezatěžuje server, ale především se dá stáhnout a prohlížet offline.

4.3.1 Vykreslovací engine pro dokumentaci

Aby bylo možné vykreslit stránku až na straně uživatele, je nutné přenášet i script, který to obstará. Tento script se skládá ze dvou částí. Engine na vykreslení a data samotná, která jsou uložena ve formátu JSON.

Script projde veškerá data a podle typu a kontextu postupně vytváří html elementy podle vestavěných šablon a přidává jim příslušné styly a ovládací prvky.

4.4 Routes

Pro rozpoznání, která akce se má vykonat při různých dotazech, se porovnává jak adresa, tak i metoda. Nejdříve se vezme v potaz cesta dotazu za statickou předponou „<http://quest.ms.mff.cuni.cz/prak/api/>...“. Jakmile máme vybranou cestu, zjistí se, co vlastně dotaz potřebuje udělat, a to jednou z těchto metod:

- POST - většinou obecný dotaz s query v těle
- GET - dotaz požadující 1 záznam, většinou podle ID
- PUT - vytvoření nového záznamu
- PATCH - změna v existujícím záznamu
- DELETE - smazání záznamu

Pokud uživatel má požadované oprávnění, akce se provede. V každém případě uživateli přijde zpětná vazba o úspěchu resp. neúspěchu dotazu. Tělo této zprávy může obsahovat požadovaná data, potvrzení o úspěchu, nebo i chybová zpráva a její příčina.

4.4.1 Uživatel

Slouží pro správu uživatelských účtů. Zakládání nového účtu může zavolat kdokoliv, avšak na zbylé akce, jako mazání nebo změnu hesla má právo pouze majitel a administrátor. Stejně tak heslo a sessionId nejsou přístupné zvenčí.

4.4.2 Autorizace

Pro ověření, zda přihlášený uživatel má příslušná práva (a to na straně serveru, jelikož lokálně si je může libovoně upravit) se používá **sessionID**, které se poté porovnává v databázi s uživatelem a jeho skutečnými právy.

SessionID získáme po odeslání korektních přihlašovacích údajů. Případně jej ztratíme při odhlášení nebo expiraci (která je aktuálně nastavena na 1 rok).

4.4.3 Záznam

Pro prohlížení záznamu, resp. jejich vyhledávání, není třeba žádné oprávnění. Avšak pro jejich editaci, resp. mazání, je potřeba mít přiřazena práva pro zápis.

4.4.4 Stránka

Zobrazení stránky též nevyžaduje žádná práva, ale k jejich vytváření je třeba mít roli tzv. „CMS editor“, která, jak název napovídá, definuje uživatele jakožto editora článků a příspěvků.

4.4.5 Nahrávání souborů

Pro nahrávání souborů je třeba práv pro zápis, stejně jako pro mazání. Zobrazení pak žádná práva nevyžaduje, ani nevyžaduje přístup ze stejné domény.

4.5 Modely

Model nebo též schéma je popis záznamu v databázi, obsahuje datový typ, formát a může obsahovat i referenční cestu. V podstatě se jedná o převodní tabulku, aby se JavaScript a MongoDB schodli na datovém typu a struktuře (především pro případ reference, nebo pole dat). Část, která je pro uživatele nejvíce viditelná, je požadavek na unikátní hodnotu pole nebo požadavek aby hodnota byla nenulová.

4.5.1 Záznam

Každý záznam má přesný popis v online dokumentaci API v pravém sloupci, kde je pravidelně aktualizován.

4.5.2 Stránka

Záznam stránky obsahuje název, jazyk, krátký popis, kategorii a obsah samotný. Pro možnost sledování změn je zde i automaticky generovaný seznam editorů a časů jejich editace.

4.5.3 Uživatel

Každý uživatel se přihlašuje e-mailem a heslem, s tím že heslo není uloženo v tzv. raw formátu, ale je výsledkem spojení hesla a „soli“ (anglicky *salt* - termín pro náhodný text pro zvýšení bezpečnosti proti bruteforce útoku) a výsledný hash

(metodou md5) se uloží do databáze. Při přihlášení pak stačí porovnat hash soli a hesla s údaji v databázi. Pro ověřování práv ale neslouží heslo, ale sessionID, které je unikátní pro každého uživatele, a časem nebo manuálně může expirovat a je nutné se přihlásit znovu. V záznamu uživatele je uloženo i jméno a příjmení pro případ, že by bylo potřeba zjistit, kdo např. psal jaký článek a zároveň nebyl prozrazen e-mail. Každý uživatel má určitá práva a to v libovolné kombinaci z následujících:

- read - právo pro čtení, nahlížení do záznamů a jejich vyhledávání (toto právo má i nepřihlášený uživatel).
- write - právo pro zápis v rejstřících, umožňuje editovat, vytvářet a mazat záznamy. Dále má navíc právo ukládat na server obrázky a dokumenty.
- execute - administrátorské právo, umožňuje vytváření uživatelských účtů a změnu hesla uživatele, stejně jako jeho údaje.
- cms - práva pro editační změny, co se týče obsahu stránek, jako jsou novinky a příspěvky.

5. Implementace frontendu

Frontend byl vyvíjen jako single-page application pomocí javascriptu a Reactu. Navržen tak, aby byl jednoduše ovladatelný přehledný a především velmi rychlý.

5.1 Server

Celá aplikace frontendu je uložena na serveru, včetně zdrojových kódů. Uživatel si ale stahuje pouze zkompilovanou aplikaci a případné externí zdroje.

5.1.1 Kompilace

Ačkoliv je JavaScript scriptovací jazyk a kompilaci provádí až za běhu, tak je možné jej zkompileovat předem. Při této kompilaci knihovny **webpack** a **babel** provedou několik zásadních kroků, z nichž nejdůležitějšími jsou převedení kódu na ECMAScript 5 (kvůli zpětné kompatibilitě), přetřídění a sloučení knihoven do jednoho zdrojového souboru (JavaScript se k uživateli dostane při prvním dotazu na server) a minifikace výsledného souboru, jež dokáže inteligentně projít kód a optimalizovat jeho textovou délku, čím výrazně sníží čas potřebný k načtení stránky.

5.1.2 NPM

Celá aplikace má strukturu balíčku NPM (node package manager), tudíž ji lze snadno spustit kdekoli, kde je nainstalovaný *nodejs* a knihovna *npm*. Zároveň udržuje pomocí souboru **package.json** přehled o potřebných knihovnách (dependencies) a obsahuje též různé spouštěcí skripty, např. skript pro build produkční verze aplikace.

5.2 Knihovny

Aplikace se skládá z více než tisíce knihoven a modulů, tudíž se podíváme pouze na ty největší a nejdůležitější z nich.

5.2.1 React

Tato JavaScriptová knihovna je vyvinuta společností Facebook a slouží pro tvorbu uživatelského rozhraní.

Její základem je vykreslování pomocí tzv. single-page a je vhodná zejména pro aplikace, kde se často mění data. Využívá pozměněnou JavaScriptovou syntaxi známou jako JSX (JavaScript XML), která umožňuje pracovat s HTML tagy uvnitř JavaScriptového kódu, bez nutnosti práce DOM objektem.

Výhodou single-page aplikace je pak i optimalizovaná práce s DOM objektem, který je bottleneck (úzkým hrdlem) při nesprávném použití (nebo lépe řečeno, při kterémkoliv použití, které něco s DOM objektem dělá).

5.2.2 Material-UI

Společnost Google (aktuálně Alphabet Inc.) se v roce 2014 rozhodla sjednotit svou grafickou podobu a vyvinula designový jazyk, jež se stal příručkou jak vytvářet uživatelsky přívětivé aplikace (nejen na webu, ale třeba i v aplikaci na mobilním telefonu nebo tabletu). Základem tohoto stylu je realistická práce se světlem a uživatelskou interakcí. Vše odlehčené a intuitivní. Součástí tohoto grafického jazyka jsou i lehce čitelné fonty (Roboto), pochopitelné ikonky a systém barev a jejich kombinací.

5.2.3 i18n

Jelikož je projekt mířen i na nečeské uživatele, vyskytla se potřeba rozhraní překládat. Jednoduchým přístupem by bylo zkompileovat několik aplikací, každou pro jiný jazyk, tudíž by se překlad odehrával na straně serveru. Bohužel to není neoptimálnější řešení a tudíž byla využita knihovna *i18n*, jež funguje na podobném principu jako single-page application a to sice, že dokáže měnit překlad stránky bez nutnosti stránku přenačíst a zároveň hlavní aplikace neobsahuje všechny jazyky při prvním načtení. Jazyky se postupně dostahují dle preference uživatele (vše samozřejmě na pozadí bez uživatelského zásahu).

Velkou výhodou tohoto systému může být například situace, kdy je potřeba přeložit kus textu na stránce, případně sehnat anglický ekvivalent a nechceme přenačtením stránky přijít o již vyplněná data.

5.2.4 Babel

Jak už bylo zmíněno výše, aplikace se kompiluje a za tuto část je zodpovědná právě knihovna Babel.

Hlavní výhodou je kompilace kódu ve verzi ES6+ (EcmaScript v6, neboli JavaScript v6) do verze ES5 (EcmaScript v5, neboli JavaScript v5). Tímto převodem získáme zpětnou kompatibilitu pro starší JavaScriptové enginy.

5.2.5 Webpack

Aby se z tak obrovského množství knihoven a souborů stal jediný spustitelný soubor pomáhá knihovna Webpack, která má za úkol kód zkomprimovat a sjednotit. Touto optimalizací si ušetříme množství dotazů, jež bude muset náš server přijmout (a ekvivalentně s tím uživatel odeslat).

Zároveň nám umožňuje používat formát **SCSS**, což je nadstavba nad klasickým CSS, jež rozpoznávají prohlížeče a dovoluje nám takto vytvářet **stylovací moduly**.

5.3 Rozhraní

Jelikož se jedná o složitější systém a všechno nemůže být „naházené“ v jediném souboru, tak je použita hierarchie která vypadá následovně:

- Hlavní soubor celého webu - provádí routing a obaluje celou aplikaci pomocnými wrapery (např. *i18n* překladač, *Mui* pro jednotný styl, *SnackbarProvider* pro vyskakovací toasty a *CookiesProvider* pro jednotný přístup ke cookies)
- Scény - Stránky diametrálně rozdílných vlastností a funkcionalit
- Komponenty - malé na sobě nezávislé „černé krabičky“ poskytující určitou funkcionalitu

5.3.1 Scény

Většina scén má klasické pojmenování, ale najdeme zde i uživatelsky atraktivní přesná pojmenování scén podle účelu jako je „Vyhledávátko“, „Zobrazovátko“ nebo „Upravovátko“.

Admin

Administrační rozhraní je přístupné pouze s oprávněním „execute“. Zde může administrátor nebo správce uživatelů vytvářet nové účty nebo konfigurovat stávající.

Jelikož se hesla v databázi ukládají zahešovaná, tak je nelze zobrazit ani administrátorovi. Dala by se přidat funkcionalita pro jejich crackování, aby se našli uživatelé se slabými hesly, ale to by bylo zbytečné plýtvání serverovými prostředky.

The screenshot shows the admin interface for user management. The top navigation bar includes links for 'PRAMENY', 'KRKONOŠ', and 'RIESENGBIRGSQUELLEN'. The main content area is divided into two sections: 'Přidat nového uživatele' (Add new user) and 'Všichni uživatelé' (All users).

Přidat nového uživatele

Form fields include:

- Email *
- Heslo *
- VYGENEROVAT BEZPEČNÉ HESLO (Generate secure password)
- Čist (Clean) checkbox
- Vkládat (Add) checkbox
- CMS (Redakční systém) checkbox
- Jméno (Name) field
- Příjmení (Surname) field
- PŘIDAT UŽIVATELE (Add user) button

Všichni uživatelé

Jméno	Příjmení	Email	Heslo	Čist	Vkládat	Spravovat (Admin)	CMS (Redakční systém)	ULOŽIT ZMĚNY	SMAZAT UŽIVATELE
admin		Cenzurováno	*****	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	ULOŽIT ZMĚNY	SMAZAT UŽIVATELE
Kateřina	Brožová	Cenzurováno	*****	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	ULOŽIT ZMĚNY	SMAZAT UŽIVATELE
David	Babka	Cenzurováno	*****	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	ULOŽIT ZMĚNY	SMAZAT UŽIVATELE
Olga	Hájková	Cenzurováno	*****	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	ULOŽIT ZMĚNY	SMAZAT UŽIVATELE
Václava	Horčáková	Cenzurováno	*****	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	ULOŽIT ZMĚNY	SMAZAT UŽIVATELE

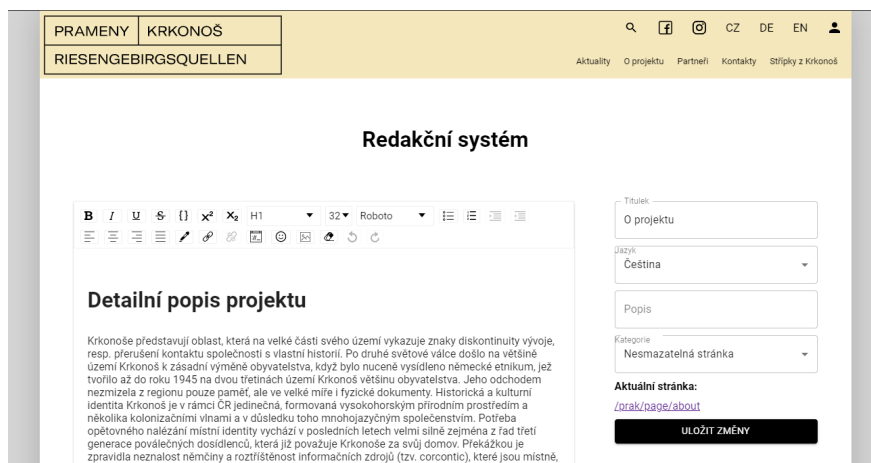
Obr. 5.1 | Frontend administrační scény

CMS

Vytváření a editace obsahu stránek se provádí v redakčním systému (Content Management System, zkráceně CMS). Do něj se uživatel dostane pouze po přihlášení a s platnými právy pro redakční systém.

Úpravy redaktor provádí v prostředí WYSIWYG (What you see is what you get, neboli co vidíš, to dostaneš). V pravém panelu pak zadá název, případně krátký

popis, který se hodí např. pro aktuality a kategorii. WYSIWYG editro podporuje velkou škálu stylování a formátování. Velkou výhodou je možnost nahrávání obrázků, kdy se po přetažení automaticky uloží na server do složky *uploads* a jejich odkaz se vloží do stránky.

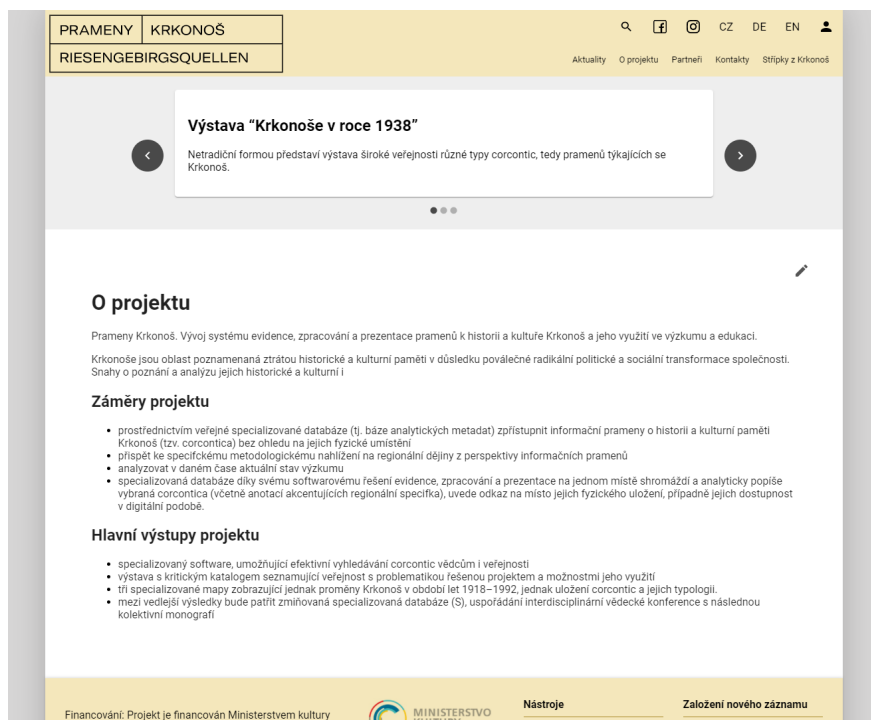


Obr. 5.2 | Frontend scény pro editaci stránek (CMS)

Domovská stránka

Výchozí stránka, na kterou se uživatel dostane, pokud nezadá přesnější cestu v url, je domovská stránka.

Na této stránce uživatel najde dynamické zobrazování novinek a krátkého popisu v horním panelu a stránku s názvem „homepage“, což je obyčejná stránka editovatelná v redakčním systému a dovoluje tedy redaktorům měnit i domovskou stránku.



Obr. 5.3 | Frontend hlavní stránky aplikace

Zobrazení stránky

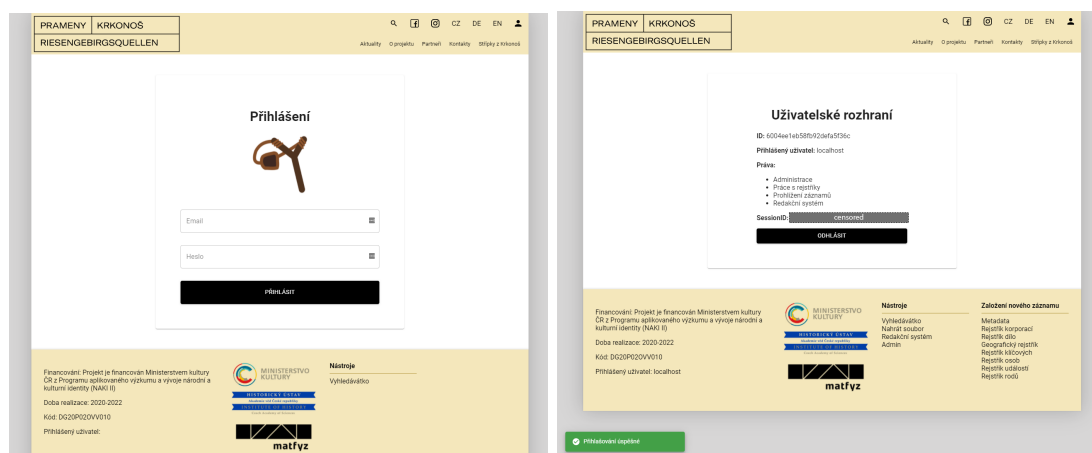
Každá stránka vytvořená v redakčním systému je přístupná na url adrese `.../prak/page/#jmenoStranky`. Zde si ji můžou uživatelé zobrazit. Pokud má uživatel i právo pro redakční systém, tak se mu v pravém horním rohu zobrazí ikonka pro editaci, díky které se snadno dostane do redakčního systému pro danou stránku.



Obr. 5.4 | Zobrazení normální informativní stránky

Přihlašování

Pokud uživatel klikne na ikonku postavičky vpravo v hlavičce, nebo se pokusí dostat na stránku, ke které nemá oprávnění, je přesměrován do přihlašovacího prostředí. Pokud se poté přihlásí, nebo již přihlášen byl, zobrazí se mu podrobnosti o jeho účtu, včetně dostupných oprávnění.



Obr. 5.5 | Přihlašovací stránka vlevo a stránka s informacemi o přihlášeném uživateli vpravo

Kontaktní formulář

Pro zpětnou vazbu či otázky k projektu, můžou návštěvníci využít kontaktní formulář.

PRAMENY | KRKONOŠ
RIESENGBIRGSQUELLEN

Aktuality O projektu Partneři Kontakty Stípký z Krkonoš

Kontakty

Máte-li otázku, podnět či připomínku, kontaktujte nás prosím prostřednictvím kontaktního formuláře.

Jméno *

Email *

Zpráva *

ODESLAT

Kontaktovat můžete také koordinátora projektu kohoutek@hiu.cas.cz či jednotlivé řešitele.

Financování: Projekt je financován Ministerstvem kultury ČR z Programu aplikovaného výzkumu a vývoje národní a kulturní identity (NAKI II)

Doba realizace: 2020-2022

Kód: DG20P020V010

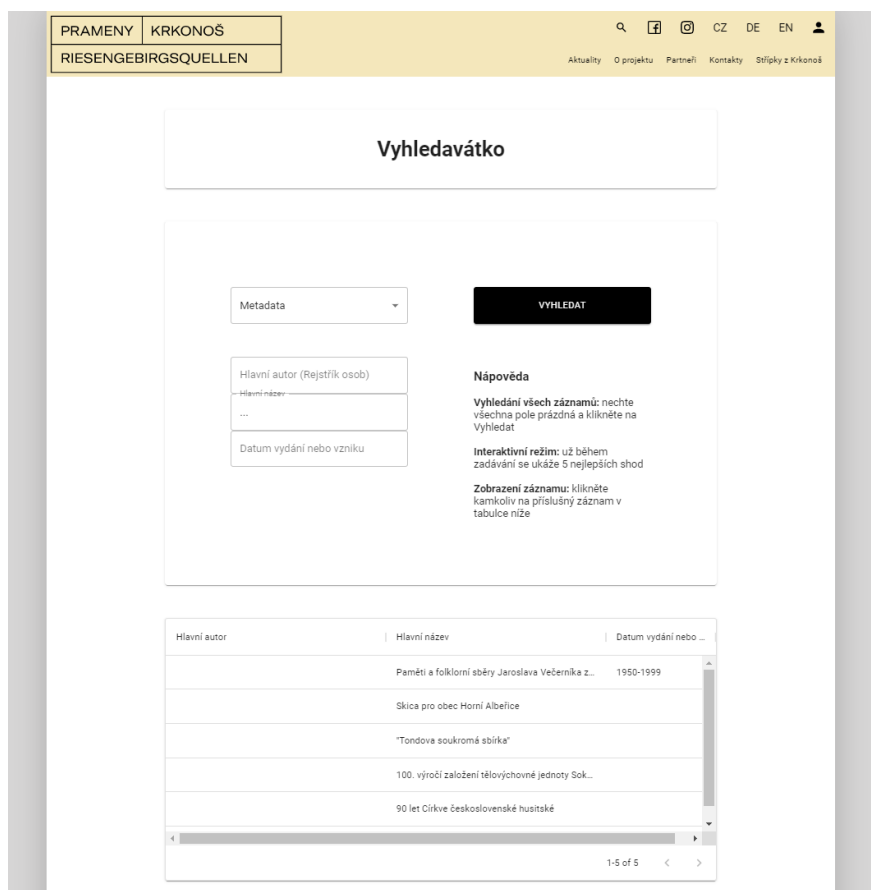
MINISTERSTVO KULTURY
HISTORICKÝ ÚSTAV
Kvalitní věda. Evropská spolupráce.
EUROPEAN RESEARCH INFRASTRUCTURE

Nástroje
Vyhledávátko

Obr. 5.6 | Kontaktní formulář

Vyhledávátko

Pomocí ikonky lupy v hlavičce se dostaneme do vyhledávacího prostředí. Po výběru, kde chceme hledat, se nám načtou políčka k vyplnění. Uživatel do nich může zadat libovolný výraz, a to s podporou regexp syntaxe. Ihned při vyplňování se načítá až 5 prozatím nejvhodnějších výskytů hledání. Při kliknutí na tlačítko „vyhledat“ se provede vyhledání všech odpovídajících záznamů. Nalezené záznamy se objevují v tabulce pod vyhledávacím polem. Tabulka dokáže záznamy třídit po kliknutí na hlavičku příslušného sloupce. Po kliknutí na záznam se otevře editační prostředí pro vybraný záznam.

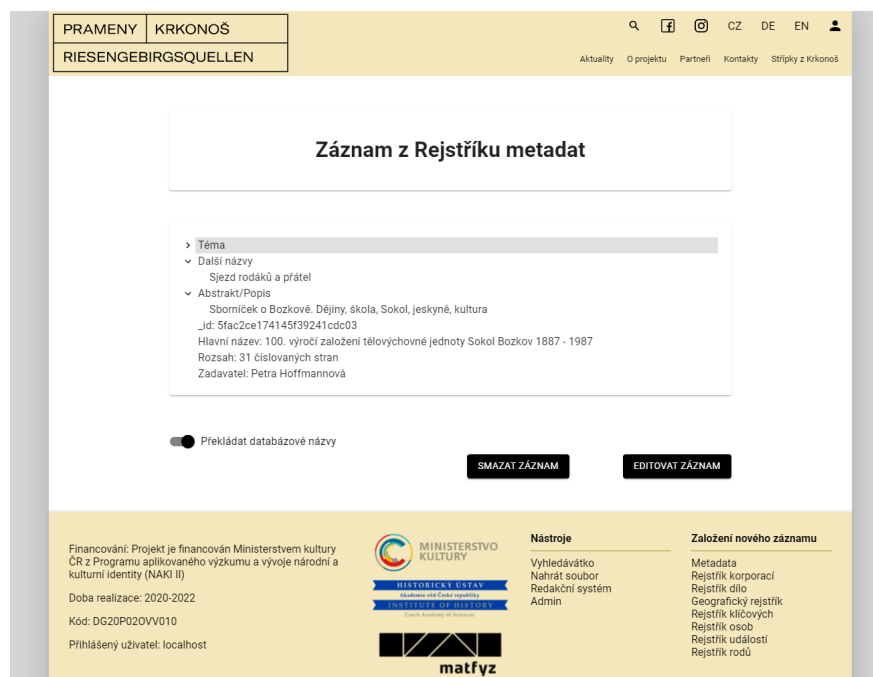


Obr. 5.7 | Frontend Vyhledavátka

Zobrazovátko

Záznam uložený v databázi si je možné zobrazit přes toto rozhraní, poté co se k němu dostanete přes vyhledávací rozhraní nebo pomocí unikátního url linku, který se po vytvoření záznamu již nemění.

Po načtení se uživateli zobrazí záznam tak, jak je uložen v databázi, případně s přeloženými názvy, pokud to má pomocí spodního přepínátka povolené. Dole též najde dvě tlačítka pro smazání záznamu a editaci, jež uživatele přesune do editačního rozhraní s aktuálním záznamem (samozřejmě jen pokud má dostatečná oprávnění).



Obr. 5.8 | Frontend Zobrazovátka

Upravovátko

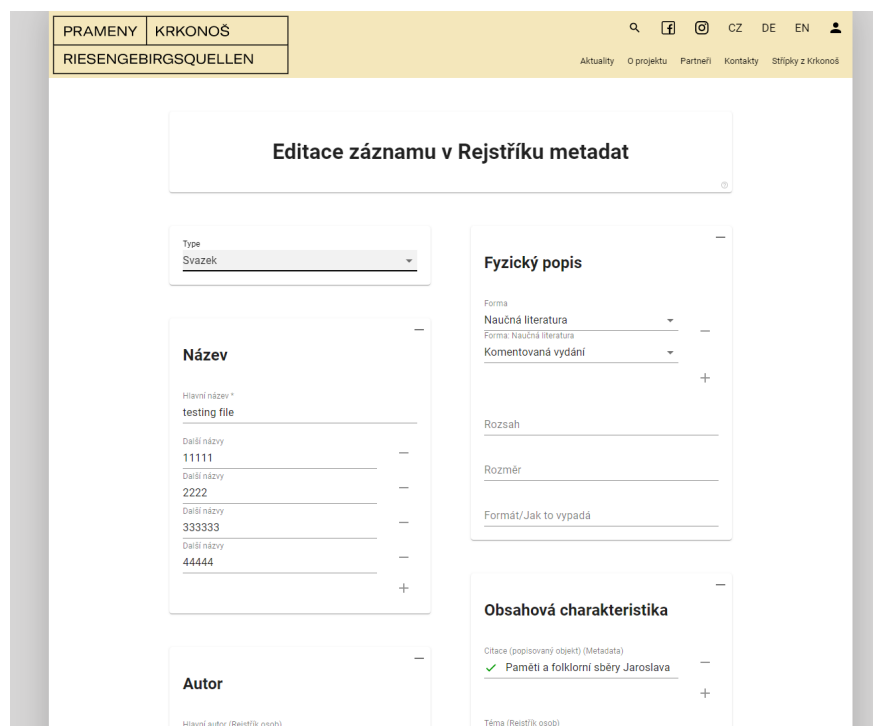
V tomto prostředí je možné záznam upravovat a jedná se o nejkomplikovanější scénu tohoto projektu.

Každý typ záznamu a každý podtyp metadatové struktury má vlastní příslušná datová pole. Ty, které umožňují mít více hodnot, mají pravo u sebe tlačítka plus a mínus, jehož velikost určuje velikost bloku, jež je jedním celkem. Plusové tlačítko přidává další datové pole, zatímco mínus datové pole vyprázdňuje a poté odebere z náhledu.

Datová pole jsou roztržena do menších podbloků, které lze minimalizovat (bez ztráty dat), aby uživateli usnadnily orientaci. Po minimalizaci, resp. maximalizaci, bloku může dojít k převážení sloupců a některé bloky tak mohou střídat levý a pravý sloupec, tak aby celková stránka byla co nejkratší a prázdného místa bylo co nejméně.

Některé záznamy mají možnost nahrát přílohu, ta se aktivuje po kliknutí na tlačítko uploadu, s ikonkou šipky směrem nahoru. Nahráný soubor se ihned přenáší na server a do příslušného pole automaticky zaznamená aktuální url nahrávaného souboru.

Po kliknutí nahrát se buď objeví červené vyskakovací okno s chybou, proč záznam nelze uložit (většinou zapříčiněný zadáním unikátního názvu, jež již je v databázi uložen). Nebo se stránka znovu přepne do zobrazovátka a vyskočí zelené potvrzovací okénko.





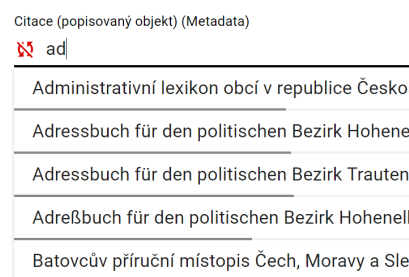
Obr. 5.9 | Frontend Upravovátka

5.3.2 Komponenty

Komponenty jsou vlastně zapouzdřené třídy, které se dají použít vícekrát. Každá komponenta je vhodná pro React systém a většinou je psaná bez tzv. hooků.

ComboBox

ComboBox je textové pole, které při vyplňování realtime vyhledává v databázi vyhovující shody, které zobrazuje ve formě nabídky pod tímto textovým vstupem. Po kliknutí na nabízenou položku se do pole vyplní hodnota a na pozadí se do pole uloží ID záznamu (to se poté odesílá na server). Dokud uživatel některý ze záznamů nevybere, pole není v konzistentním stavu a nemá možnost se odeslat na server, což značí ikonka , v případě že pole má přiřazené ID záznamu a zobrazuje název, rozsvítí se ikonka , která značí že data z tohoto pole se při uložení správně odešlou na server. Popisky často bývají dlouhé, a proto je zde i miniaturní posuvník, se kterým se dá pohodlně pracovat pomocí najetí myši na políčko, přidržení klávesy *Shift* a točením kolečka na myši.



Obr. 5.10 | Komponenta ComboBox

Zápatí

Jako každá správná stránka i v tomto projektu najdeme zápatí, jež obsahuje povinné údaje o projektu a rychlé odkazy.

Jelikož se jedná o projekt MK ČR - NAKI, nalezneme zde povinné údaje, kód projektu a loga sponzora, zadavatelů a vývojářů. Níže je ukazatel aktuálně přihlášeného uživatele. V pravé části jsou pak rychlé odkazy, které se zobrazují v závislosti na právech přihlášeného uživatele. Nepřihlášený uživatel tudíž v této části uvidí pouze jedno menu „Nástroje“ s jedinou položkou „Vyhledávátko“ a po přihlášení se viz obrázek 5.11 menu rozroste o další rychlé odkazy.



Obr. 5.11 | Zápatí stránky

Navigační menu

V navigačním menu nalezneme rychlé odkazy na Vyhledávátko (tlačítko s ikonkou lupy), Facebook a Instagram projektu Prameny Krkonoš a přihlašovací stránku. Tlačítka pro přepínání mezi třemi jazyky. Níže pak důležité stránky projektu a některé kategorie (např. kategorie aktuality).



Obr. 5.12 | Navigační menu

Textové pole s validací

To, že uživatel zadává data ve srozumitelném formátu (tak aby je dokázal pochopit další uživatel nebo aby je databáze správně interpretovala), je kontrolováno na straně klienta (a po odeslání i na straně serveru). V případě, že data požadovaný formát nemají, pod políčkem se zobrazí varovný nápis a záznam nepůjde uložit, dokud všechna pole nejsou ve správném formátu.

Souřadnice

N52.0 N53.0

Chybný formát souřadnic

Obr. 5.13 | Pole s kontrolou validity dat

Pole pro nahrávání souborů

U některých záznamů je užitečné uchovat přílohu ve formátu obrázku nebo dokumentu. Tyto soubory lze nahrát na server a do databáze uložit pouze odkaz na jejich umístění, protože databáze není stavěná na uchovávání obrázků a podobných relativně velkých souborů.

Příloha

/prak/uploads/tigg8.png



Obr. 5.14 | Pole pro nahrávání souborů

Indexy

Hlavní částí editačního rozhraní jsou indexové objekty, které určují, která pole se mají zobrazit při různých typech záznamů. Pro každý typ je zde JSON soubor,

ve kterém je seznam polí a informace o nich. Mezi tyto informace patří především popisek, struktura, jak data odeslat do databáze, nápověda přístupná pomocí malého otazníčku a nepovinná pole jako požadavek na nenulovou hodnotu, nebo regexpový výraz pro kontrolu formátu dat.

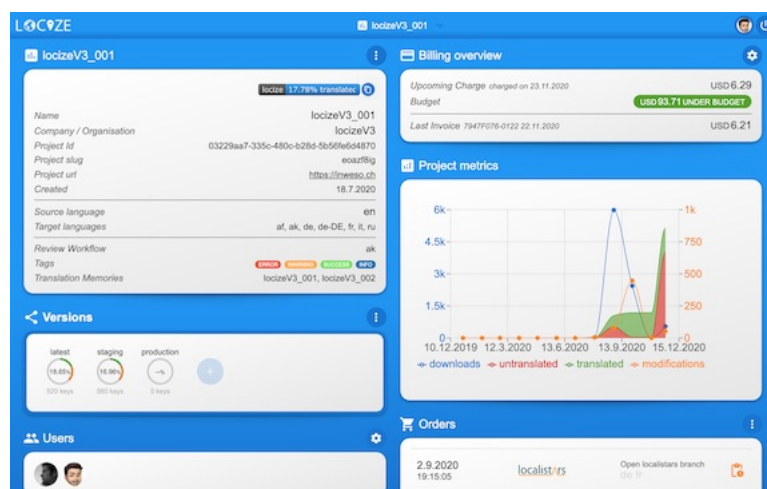
5.3.3 Moduly

Funkcionalita, která není až tak často používána, se do hlavního souboru aplikace nepřidává a tudíž čistý frontend žádné moduly nepodporuje. Moduly jsou výhradně načítány ze serveru a neobsahují hlavní frontend.

5.4 Lokalizace

Překlad stránky do různých jazyků je řešen pomocí knihovny i18n, která je velmi efektivní a zároveň poskytuje snadný způsob, jak texty překladu přidávat a editovat. Překlady navíc jsou rozloženy do více souborů a úrovní, takže se překlad velmi dobře škáluje i na větší projekty.

Výsledkem jsou lokalizační soubory ve formátu JSON, které frontend aplikace na požadovaných místech. Pokud ovšem překlad chybí, použije se nejbližší možná interpretace, podobný jazyk nebo zdrojová adresa překladu a vývojáři ukáže zprávu o chybějícím překladu.



Obr. 5.15 | Aplikace Locize umožňující správu překladů knihovny i18n

6. Provázání Backendu a Frontendu, API

6.1 Dokumentace online

Aktuální dokumentace, tak aby mohla být časem aktualizována a mohli do ní být připisovány další věci, se nachází na webu `.../prak/api/documentation`. Dokumentace je rozdělena na dvě části.

První část popisuje volání API.

Každá metoda (GET, POST atd.) má svůj vlastní účel, jež je popsán uvnitř url, a další možné parametry. Spolu s formátem requestu je zde i formát odpovědi. Podle kódu zjistíme, jestli byl náš požadavek úspěšný. Kódy jsou standardní podle "http status codes".

- **2xx** Všechno dopadlo dobře
- **4xx** Chyba je na straně klienta
- **5xx** Chyba je na straně serveru

V případě úspěchu (kód 200 - OK) se odešle odpověď na dotaz nebo prázdné tělo, pokud request neměl za funkci něco vracet. V případě neúspěchu pak v odpovědi najdeme zprávu o chybě, která nastala. Obvykle to u kódu 500 bývá odeslání duplicitního záznamu.

Druhá část popisuje strukturu schémat jednotlivých modelů.

Jelikož databáze má datové formáty, zatímco JSON soubor ne (nebo alespoň ne tak rozsáhlé), musí i odesílaná data mít správný formát, nebo alespoň být validní po automatickém přetypování. Schéma tvoří JSON objekt, jež toto schéma popisuje. Pokud je datový typ položky objekt buď se opravdu jedná o objekt, nebo se jedná pouze o upřesnění datového typu. Klíčovými slovy jsou:

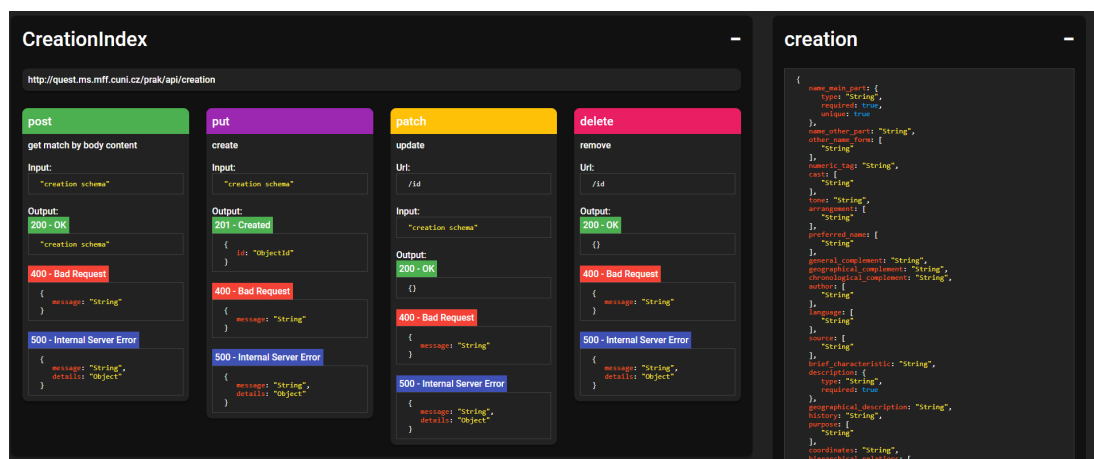
- **type**: datový typ
- **required**: true, pokud je tato položka povinná
- **unique**: true, pokud se zadaná hodnota nesmí shodovat s již existující položkou v DB
- **ref**: název schématu, na který se ID odkazuje
- **refPath**: speciální ref, umožňující uživateli zadat i název schématu, vůči kterému se odkazuje

Pokud je hodnota pouze string, pak je tato hodnota datovým typem.

6.1.1 Software pro online dokumentaci

Pro možnost stažení dokumentace a prohlížení offline, je vše zabaleno do jediného html souboru. Uvnitř je zdrojový kód programu, jež vykresluje stránku a zároveň data dokumentace.

Program vykresluje veškeré položky s daty dokumentace. Bloky zdrojových kódů jsou vysázeny fontem monospace a obarvený, aby uživateli poskytly rychlejší orientaci v kódu.



Obr. 6.1 | Dokumentace API jako webová stránka

6.2 Backend

Na backendu je spuštěný express server (běžící pod nodejs), který zachytává requesty s url `...prak/api/...`. Podle cesty, jež je uvedena za `/api` se request předává příslušnému routeru. Výjimku tvoří adresa `.../api/documentation`, která rovnou přeposílá soubor s dokumentací a není tedy pro získání dokumentace třeba rozumět systému requestů hlouběji.

Při převzetí requestu jedním z mnoha routerů, se porovnává typ requestu (POST, PUT atd.) a případné detaily cesty v url. Při plném nalezení shody se provede ověření práv, pokud je to potřeba. Pokud request má požadovaná oprávnění, je provedena příslušná funkce a uživateli je vrácena odpověď případně potvrzení o úspěchu. V případě, že request nemá příslušná oprávnění, je vrácen kód 401. V případě, že se na serveru něco pokazí, vrátí se kód 400, nebo 500, podle typu problému.

6.3 API

API je přístupné na adrese `quest.ms.mff.cuni.cz/prak/api/...`.

6.3.1 Autentifikace

S každým requestem přichází v hlavičce i cookie, ten pro autentifikaci se jmenuje "sessionID". Podle něj se najde příslušný uživatel a porovnají se jeho práva

a práva potřebná pro vykonání požadované funkce. Pokud jsou práva nedostatečná, vrátí se odpověď s kódem 401, v případě správného oprávnění router vykoná funkci, jež danému requestu přísluší a pokud se nepokazí nic jiného, vrátí validní odpověď.

6.4 Frontend a volání API

Jelikož je celý projekt zamýšlen jako webová aplikace, nejstandardnější použití je volání API pomocí JS funkce *fetch()*, což je pouze technický alias pro `XMLHttpRequest`. Ale je možné jej volat jakkoliv jinak, dokud to bude validní `http request`.

6.4.1 Fetch

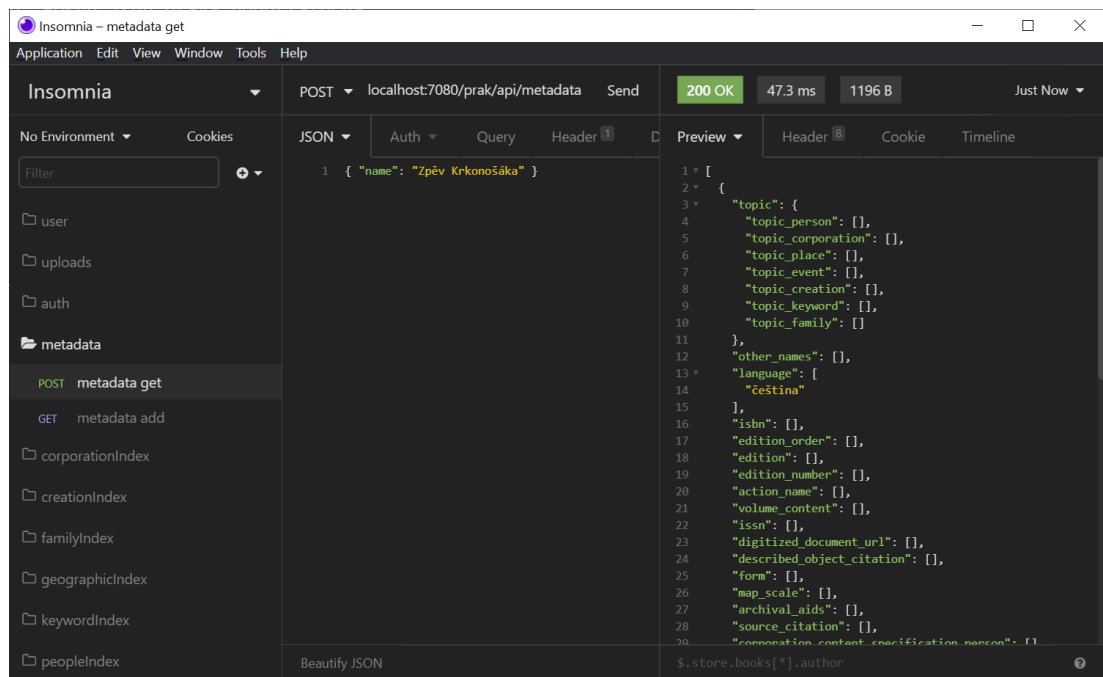
Dokumentace metody: developer.mozilla.org

Příklad použití:

```
const url = "/prak/api/metadata"
fetch(url, {
  method: "POST",
  headers: { "Content-Type": "application/json" },
  body: JSON.stringify({ "name": "... " }),
})
.then(response => {
  if(!response.ok) throw response
  return response.json()
})
.then(response => {
  console.log(response)
})
.catch(error => {
  console.error(error)
})
```

6.4.2 Existující programy pro práci a testování API

Příklad volání pomocí programu Insomnia:



Obr. 6.2 | Program insomnia s vyplněným dotazem pro odeslání požadavku na API

7. Moduly

7.1 Přidávání nových modulů

Nové moduly se nahrávají do složky *modules*, která leží vedle adresářů *frontend*, *backend* a *uploads*. Nahrávat lze jednotlivé html soubory, složky se složitější architekturou nebo celé npm balíčky s vlastní zkompilovanou stránkou. Moduly by měly být nezávislé na obsahu diskového prostoru v „rodiči“ a měly by tedy být samostatně spustitelné, maximálně s interakcí s API.

7.2 Aktuálně nasazené moduly

7.2.1 Modul hologram

Načítání modulu

Tento modul obsahuje velkou knihovnu, a tudíž není efektivní jej mít v primární aplikaci. Protože by se tyto knihovny načítaly, i když by uživatel s tímto modulem neměl v plánu pracovat. Což většinu času nebude chtít, a tudíž by to pouze vedlo ke zpomalení systému. Systémem LAZY načítání je tedy celý modul odříznut a načítá se až explicitně při jeho použití.

Grafický engine

Pro vykreslování 3D modelu na webu slouží WebGL, grafická knihovna podobná staré verzi OpenGL. Reálně jsou dostupné dvě propracované knihovny pro práci ve WebGL. ThreeJS, které se zaměřuje na statické vykreslování a předvádění. BabylonJS, která naopak napodobuje známé programy jako je *unity* a má podporu fyziky a dalších podknihoven užitečných pro tvorbu her. Pokud pomineme možnost naprogramovat vše od základu, nejvhodnější knihovnou je **ThreeJS**.

Z této knihovny využijeme moduly pro načítání modelu a textury, jejich spárování a efektu Peppers Ghost, který je implementován pomocí 4 nezávislých kamer.

Peppers Ghost Effect



Obr. 7.1 | Peppers Ghost efekt

Jedná se o divadelní trik, jež skládá pozorovateli dva obrazy přes sebe a vytváří iluzi hologramu nebo ducha. Původně byl vyvinut pro divadelní představení, v moderní době však dokáže simulovat i hologram, který známe ze sci-fi.

Modely a textury

Modely jsou z webu <https://3dwarehouse.sketchup.com> a textury z webu <https://www.textures.com/> a <https://texturehaven.com/>. V programu blender pak byly tyto textury namapovány na objekt a bylo do nich zapečeno ambient occlusion (stíny vytvořené stísněným prostorem).

7.2.2 Prohlížeč map

Dalším historickým prvkem, který bychom návštěvníkům této aplikace poskytly jsou 2D mapy z 19. století. Uživatel si může vybrat mapu a tu si velmi podrobně prohlížet pomocí přibližování a posouvání.

Zdroj map

Mapy byly získány od ČUZK výměnou za jejich kompletnost. Mapu jsme totiž dostali ve formě „tady máš puzzle a poslepujte si to sami“. Každá mapa je tedy pečlivě zrekonstruovaná, při zachování všech detailů.

8. Instalace a spuštění

8.1 Prerekvizity

Předpokládáme instalaci na operačním systému Windows 10 s následujícími předinstalovanými programy.

8.1.1 NodeJS

JavaScriptový interpret v nejnovější verzi lze stáhnout z oficiálního webu <https://nodejs.org/>.

Spolu s nodejs se nám nainstaluje i příkaz npm, ten ihned využijeme k nainstalování jeho lepšího „dvojčete“ a to sice pnpm, pomocí příkazu

```
npm install -g pnpm
```

8.1.2 MongoDB

Databázový nástroj lze stáhnout ze stránky <https://www.mongodb.com>, nebo využít službu „MongoDB Atlas“, což je přednastavená databáze na serveru Google (nebo u Amazonu podle preference).

Pro snazší konfiguraci je vhodná aplikace *MongoDB Compass*, poskytující oproti webové aplikaci i příkazový řádek.

8.1.3 Nginx

Webový server lze stáhnout z oficiální stránky vývojářů <https://www.nginx.com/>. Konfigurační soubor pak nalezneme dle cesty instalace: `.../ChocoInstallFolder/nginx-x.xx.x/conf/nginx.conf` do něj vložíme naši konfiguraci serveru (přizpůsobenou vůči místu instalace)

```
server {
    listen 80 default_server;
    server_name _;

    # React app & front-end files
    location / {
        root /var/www/naki/frontend/build;
        rewrite ^ /index.html break;
    }
    location /static/{ root /var/www/naki/frontend/build; }
    location /images/{ root /var/www/naki/frontend/build; }
    location /locales/{ root /var/www/naki/frontend/build; }
    location /public/{ root /var/www/naki/frontend/build; }

    location /modules/{ alias /var/www/naki/modules; }
    location /uploads/ { alias /var/www/naki/uploads; }

    # node API reverse proxy
    location /api/ {
```

```
    proxy_pass http://localhost:50081/;  
  }  
}
```

8.1.4 Powershell

Jelikož výchozí příkazová řádka od Windows má svá léta slávy za sebou, je vhodnější používat powershell, který se syntaxí více podobá linuxovému shellu. Pro multiplatformní účely je nejlepší jeho „core“ verze, kterou lze stáhnout z github repozitáře (v README.md je odkaz) <https://github.com/PowerShell/PowerShell>.

8.1.5 Chocolatey balík

Pro ty, kteří znají a mají balíčkovací manager „chocolatey“ pro Windows, je zde jednoduchý kód pro rychlou instalaci.

```
choco install nodejs  
choco install mongodb  
choco install mongodb-compass  
choco install nginx  
choco install powershell-core  
npm install -g pnpm
```

8.2 Zdrojové soubory

TODO zveřejnit kód na githubu

8.3 Instalace

Ve složce se zdrojovými soubory spustíme následující příkazy:

```
##### backend npm install  
cd backend  
pnpm install  
cd ..  
  
##### frontend npm install  
cd frontend  
pnpm install  
  
##### frontend build  
npm run build
```

9. Řešení

9.1 Výsledný web

9.2 Uživatelská dokumentace

Závěr

Při práci na tomto projektu

Seznam použité literatury

- BARTOŠEK, M. (2001). *Digitální knihovny*. Brno : Masarykova univerzita.
- BARTOŠEK, M. (2004). *Digitální knihovny - teorie a praxe*. Praha: Národní knihovna.
- BRATKOVÁ, E. (2008). *Otevřený přístup a digitální knihovny v oblasti vědy a výzkumu*. Praha: Ústav informačních studií a knihovnictví FF UK v Praze.
- CUBR, L. (2010). *Dlouhodobá ochrana digitálních dokumentů*. Praha: Národní knihovna České republiky. ISBN 978-80-7050-588-5.
- KNIHOVNY, D. (2008-2014). *Digitální knihovny*. NÁRODNÍ KNIHOVNA ČR. Knihovny.cz.
- KREJČÍŘ, V. (2006). *Systémy pro tvorbu digitálních knihoven*. In: INFORUM.
- ROBINSON, D. B. a LYN (2012). *An introduction to information science*. London: Facet. ISBN 978-185-6048-101.

Seznam obrázků

2.1	LOGO systému koha ze stránky https://www.mainelibit.org/node/77	5
2.2	LOGO systému Evergreen ze stránky https://eg-wiki.osvobozena-knihovna.cz/	5
2.3	LOGO systému SLIMS ze stránky https://slims.web.id/	5
3.1	Diagram systému	8
5.1	Frontend administrační scény	15
5.2	Frontend scény pro editaci stránek (CMS)	16
5.3	Frontend hlavní stránky aplikace	16
5.4	Zobrazení normální informativní stránky	17
5.5	Přihlašovací stránka vlevo a stránka s informacemi o přihlášeném uživateli vpravo	17
5.6	Kontaktní formulář	18
5.7	Frontend Vyhledávátka	19
5.8	Frontend Zobrazovátka	20
5.9	Frontend Upravovátka	21
5.10	Komponenta ComboBox	21
5.11	Zápatí stránky	22
5.12	Navigační menu	22
5.13	Pole s kontrolou validity dat	22
5.14	Pole pro nahrávání souborů	22
5.15	Aplikace Locize umožňující správu překladů knihovny i18n, zdroj: https://locize.com/	23
6.1	Dokumentace API jako webová stránka	25
6.2	Program insomnia s vyplněným dotazem pro odeslání požadavku na API	27
7.1	Peppers Ghost efekt - zdroj: https://commons.wikimedia.org/wiki/File:Peppers_Ghost.jpg	29

Seznam použitých zkratk

Administrativní	
MK ČR - NAKI	ministerstvo kultury - národní kulturní identita
ČUZK	Státní správa zeměměřictví a katastru
Programátorské	
API	Application Programming Interface