



**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

Poděkování

Chtěl bych poděkovat Mgr. Kateřině Mackové za odborné vedení práce a cenné rady, které mi pomohly tuto práci zkompletovat. Anně Yaghobové a Monice Bošániové za pomoc a rady při zpracování této práce. A mnoha dalším, jež mi pomáhali, a to i když jen jako „gumové kachničky“.

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: Tato bakalářská práce se zabývá modernizací knihovních systémů. Zaměřuje se na technologie jako single-page application, která zefektivňuje síťovou komunikaci a snižuje vytížení serveru. Cílem bylo sepsat a zakomponovat moderní technologie do fungující aplikace, která poskytuje uživateli moderní, přehledné a rychlé prostředí pro zadávání metadat s nadstavbou redakčního systému pro správu obsahu webu. Na serveru bylo vytvořeno přehledné API poskytující veškerá data nejen hlavní aplikaci, ale i souvisejícím modulům či dalším projektům. Součástí práce je též popis technologií, díky kterým k těmto zefektivnění došlo.

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: This bachelor's thesis is on the modernisation of library systems. It focuses on technologies such as single-page application, which streamlines network communication and reduces server usage. The goal is to describe and use modern technologies into a working application, that provides the user with a modern, streamlined and fast input environment metadata with the addition of the system for managing web content. A clear API is created on the server to provide all the data not only for the main application, as well as for the related modules or other projects. The work also describes the technologies that make these streamlining possible.

Keywords: digital archive, web application, database

Obsah

Úvod	3
1 Analýza a požadavky	4
1.1 Požadavky na systém	4
1.2 Pro koho je systém určen	4
1.3 Existující systémy	5
1.3.1 Koha	5
1.3.2 SLIMS	5
1.3.3 Evergreen	5
1.3.4 Porovnání existujících systémů	6
2 Návrh projektu	7
2.1 Výběr technologií	7
2.1.1 Frontend	7
2.1.2 Backend	7
2.2 Diagram systému	9
3 Implementace backendu	10
3.1 Server	10
3.2 Knihovny	10
3.2.1 Knihovna express.js	10
3.2.2 Knihovna mongoose	10
3.3 Dokumentace API	11
3.3.1 Vykreslovací engine pro dokumentaci	11
3.4 Routes	11
3.4.1 Uživatel	12
3.4.2 Autorizace	12
3.4.3 Záznam	12
3.4.4 Stránka	12
3.4.5 Nahrávání souborů	12
3.5 Modely	12
3.5.1 Záznam	12
3.5.2 Stránka	13
3.5.3 Uživatel	13
4 Implementace frontendu	14
4.1 Hostující server	14
4.1.1 Kompilace	14
4.1.2 NPM	14
4.2 Knihovny	14
4.2.1 React	14
4.2.2 Material-UI	15
4.2.3 i18n (internacionalizační knihovna)	15
4.2.4 Babel	15
4.2.5 Webpack	15

4.3	Rozhraní	15
4.3.1	Scény	16
4.3.2	Komponenty	22
4.3.3	Moduly	23
4.4	Lokalizace	24
5	Provázání Backendu a Frontendu, API	25
5.1	Dokumentace online	25
5.1.1	Software pro online dokumentaci	26
5.2	Backend	26
5.3	API	26
5.3.1	Autentifikace	26
5.4	Frontend a volání API	27
5.4.1	Fetch	27
5.4.2	Existující programy pro práci a testování API	28
6	Moduly	29
6.1	Přidávání nových modulů	29
6.2	Aktuálně nasazené moduly	29
6.2.1	Modul hologram	29
6.2.2	Prohlížeč map	30
7	Instalace a spuštění	31
7.1	Prerekvizity	31
7.1.1	NodeJS	31
7.1.2	MongoDB	31
7.1.3	Nginx	31
7.1.4	Powershell	32
7.1.5	Chocolatey balík	32
7.2	Zdrojové soubory	32
7.3	Instalace	32
8	Řešení	33
8.1	Výsledný web	33
8.2	Uživatelská dokumentace	33
8.2.1	Návštěvník (nepřihlášený uživatel)	33
8.2.2	Zadavatel	33
8.2.3	Redaktor	33
8.2.4	Admin	34
8.3	Co by se dalo vylepšit a přidat	34
	Závěr	35
	Seznam použité literatury	36
	Seznam obrázků	37
	Seznam použitých zkratk	38

Úvod

Knihovní systémy ve své podstatě dlouhodobě uchovávají data o naší historii a je důležité abychom tyto informace uchovali i pro další generace, ale bohužel tyto systémy stagnují v zastaralých verzích. Čím méně změn tyto systémy prodělají, tím spíše budou zpětně kompatibilní a uchovaná data se zachovají v co možná nejméně pozměněné verzi. Takovýto způsob je mezi programátory docela známý pod příslovím „dokud to funguje, tak na to nešahej“, což na jednu stranu funguje, ale je důležité si uvědomit, že změna může přinést zvýšení efektivnosti pracovníků a pohodlí při práci.

Motivací k napsání této práce byla participace na návrhu řešení pro projekt NAKI II - Prameny Krkonoš. Vývoj systému evidence, zpracování a prezentace pramenů k historii a kultuře Krkonoš a jeho využití ve výzkumu a edukaci.

Hlavním cílem práce je tedy nashromáždit nejnovější trendy v oblasti webových aplikací a zakomponovat je do fungující aplikace, principiálně podobné soudobým knihovním systémům. Nový systém chceme udělat řádově rychlejší, tak aby si frontend na většinu aktivit vystačil sám a nemusel se dotazovat na server, což je jeden z typických problémů zastaralých systémů. K datům uloženým v databázi musí přistupovat efektivně, a to přes dobře zdokumentované API, nasazené na serveru tak, aby zbytečně neplýtvalo výkonem. Server by tak měl zvládnout pracovat na aktuálním hardware a zároveň obsluhovat požadavky rychleji a zvládnout jich větší množství, než soudobé knihovní systémy.

Dalším cílem je zakomponování designového jazyka takového, který byl vyvinut pro přehlednost dat a efektivitu práce. Systém s dobrým popisem funkčnosti, který je pro uživatele intuitivní a vnese do aplikace přehlednost a nadčasový vzhled. Uživatel se tak oprostí od jednotvárného a „nahňácaného“ vzhledu, ve kterém se špatně orientuje.

Aby se dal systém dále rozvíjet i mimo hlavní projekt, musí podporovat systém modulů, které budou moci čerpat společná data přes API a přidat tak do systému novou funkcionalitu.

1. Analýza a požadavky

Pro počáteční analýzu jsme vybírali 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.

1.1 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.

1.2 Pro koho je systém určen

Systém bude sloužit pro (regionální) historiky (explicitně pro pracovníky Historického ústavu AV ČR), jakožto pro zadavatele a redaktory stránek této aplikace a bude uchovávat historická data z oblasti Krkonoš, tak aby si je návštěvník či 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 plánované výstavy Pramenů Krkonoš (jakožto jednoho z výstupů projektu), jako je hologram 3D modelu nebo prohlížeč historických map.

1.3 Existující systémy

Vzhledem k tomu, že v oblasti knihovních systémů existuje nepřehledné množství různých, českých i zahraničních, (open-source) knihovních systémů, některých i účelově navržených pro různé typy fondů, my jsme se při analýze zaměřili na tři nejrozšířenější z nich. - Systémy Koha, SLiMS a Evergreen. Ve všech třech případech se jedná právě o open-source systémy.

1.3.1 Koha

Domovská stránka aplikace: <http://www.Koha.cz/>

Koha je nejrozšířenější open source knihovní systém s širokou komunitou. Koha má několik tisíc instalací v knihovnách různých velikostí a zaměření v desítkách zemí. (Tento systém je vhodný i pro použití pro konsorcia knihoven.)

Byla vyvinuta na Novém Zélandu roku 2000. V roce 2005 došlo k zásadnímu vylepšení systému integrací fulltextového systému Zebra dánské firmy IndexData. 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). Jeho nespornou výhodou je ten fakt, že systém má obrovskou (mezinárodní) komunitu, která systém udržuje a nadále rozvíjí. A pak také to, že se jedná o cloudový systém.



Obr. 1.1 | logo systému Koha

1.3.2 SLiMS

Domovská stránka aplikace: <https://slims.web.id/>

Systém SLiMS je spíše orientován na malé knihovny. (Je hojně používán v Asii.) Jeho jedinou aplikací v ČR je systém obecníknihovna.cz. - Jedná se však o výrazně přepracovanou verzi tohoto systému. 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.



Obr. 1.2 | logo systému SLiMS

1.3.3 Evergreen

Domovská stránka aplikace:

<https://eg-wiki.osvobozena-knihovna.cz>

Další z řady otevřených knihovních systémů je Evergreen (zpřístupněný pod licencí *GNU public licenses content*). Evergreen byl vyvinut v roce 2006 jako systém pro konsorcium více než 270 veřejných knihoven amerického státu Georgia.

Poté se rozšířil i do dalších států v rámci USA a do Kanady. Rozšíření mimo anglicky mluvící státy není tak masivní. Nejvýrazněji se v rámci Evropy tento systém uplatňuje ve Finsku, neboť finská národní knihovna se rozhodla systém Evergreen nasadit jako národní knihovní systém. V rámci ČR lze jako nejvýraznějšího uživatele tohoto systému označit pražskou knihovnu JABOK. (Byť systém není v ČR nějak masivněji rozšířen, přesto Evergreen disponuje českou



Obr. 1.3 | logo systému Evergreen

lokalizací.) Bohužel jeho nevýhodou je, že jeho funkčnost je zaměřena především na tento konsorciální použití, ale postupně si nalézá cestu i do knihoven akademických. Ze zajímavých funkcí, kterými se odlišuje od systému Koha, lze vybrat mimo jiné zobrazení regálu nebo expertní prohledávání obsahu konkrétního pole. Systém používá také SQL databázi, vykreslování probíhá na serveru a nemá uživatelsky přívětivé prostředí.

1.3.4 Porovnání existujících systémů

Co se týče porovnání výše uvedených systémů, lze říct, že Koha a Evergreen vykazují jisté podobné znaky, co se týče funkcionality a uživatelské podpory v podobě komunity. Koha však disponuje nepoměrně rozsáhlejší komunitou. Obě komunity postupují přibližně stejně i při řešení chyb a rozvoji systému. Oba systémy používají systém pro hlášení chyb, navrhování chybějících funkcí apod. Systém Koha je plně webový (cloudový), což znamená, že není potřeba instalovat žádný speciální klient pro uživatele, což usnadňuje nasazení a aktualizace systému. Naopak součástí Evergreenu je i aplikace, kterou musí mít uživatel (knižovník) nainstalovanou na svém počítači, aby mohl se systémem pracovat. Podstatné je, že klient je multiplatformní a může být na rozdíl od komerčních systémů provozován mimo Windows i v operačních systémech Linux a Mac OSX. Pro usnadnění přechodu na nové verze nabízí Evergreen možnost automatických aktualizací klienta. Co se týče správy systému, Evergreen a SliMS jsou systémy, které potřebují silně vyškolenou osobu, aby se o systém starala, na rozdíl od systému Koha, která je intuitivnější a pro nové uživatele přívětivější, při zachování stejné, možná i lepší funkcionality.

2. Návrh projektu

2.1 Výběr technologií

2.1.1 Frontend

Ačkoliv se trendy v oblasti vývoje webových aplikací mění velmi rychle, jedním z nejpodstatnějších trendů, který přenesl vykreslování stránky na stranu uživatele a tím se výrazně odlišil 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 knihovního 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.

2.1.2 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 jako na frontendu, JavaScript. Knihovnou, která by umožňovala komplexní správu requestů a zároveň by byla i na robustnějších projektech programátorsky přehledná, byla zvolena Express.js, z důvodů uvedených níže.

Express.js

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

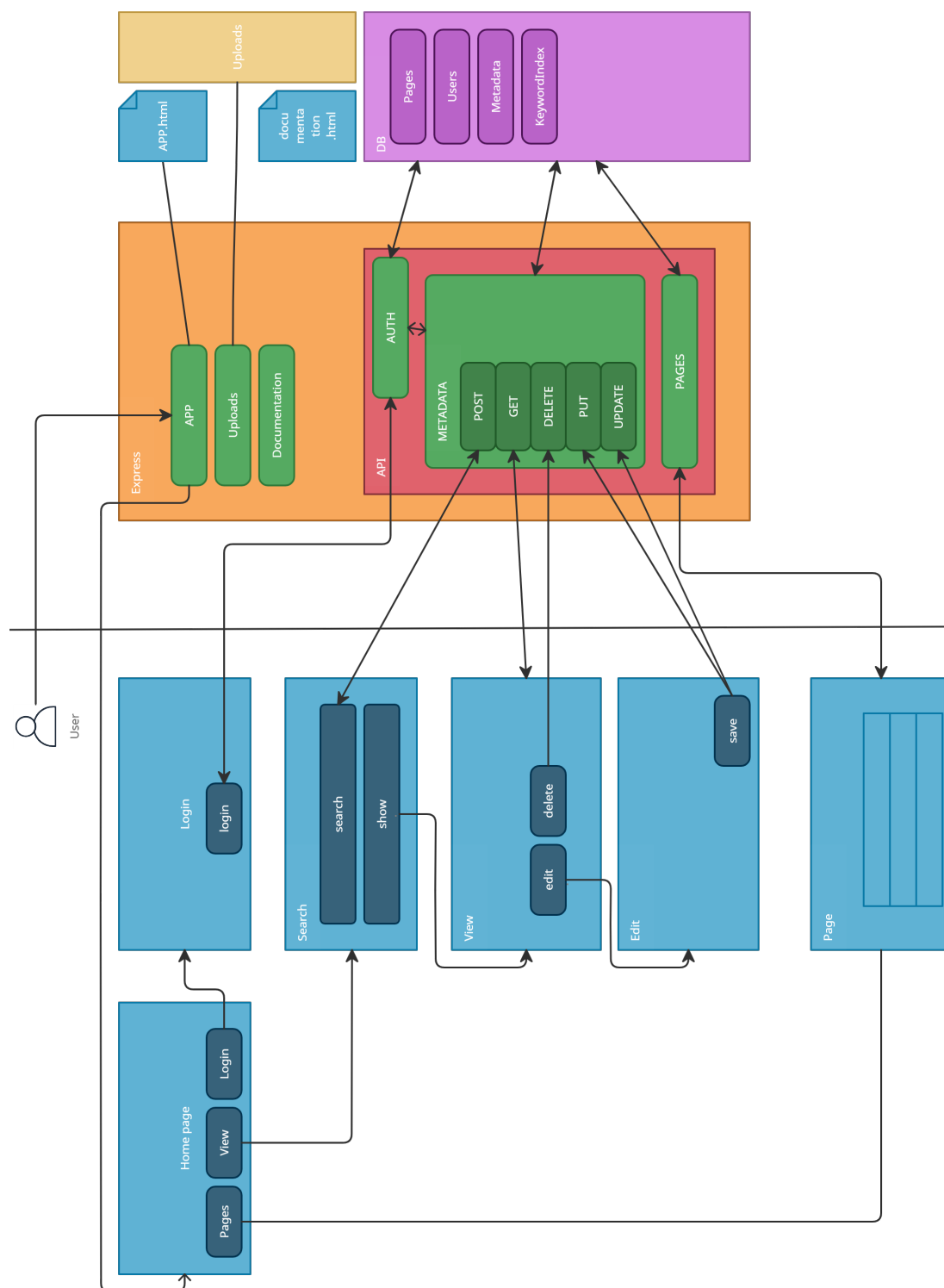
MongoDB

MongoDB je databázový systém typu non-SQL. To znamená, že data neuchovává v tabulkách, ale v tzv. schématech. To 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 (na rozdíl např. od WordPressu) je možné na backend nasadit libovolné technologie, které umí posílat requesty. Příkladem toho mohou být scripty v jazycích PHP, C#, Python, nebo Perl. Vzhledem k potřebě implementace neuronových sítí pro pokročilé vyhledávání jsme se rozhodli mezi dvěma jazyky, které jsou vhodné. Těmito jazyky s velmi dobrými knihovnami pro práci s neuronovými sítěmi, jsou Python a JavaScript.

2.2 Diagram systému



Obr. 2.1 | Diagram systému

3. Implementace backendu

Backend je software, který je umístěn na serveru a uživatel má přístup jen k výstupu, 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.

3.1 Server

Stroj, na kterém je dostupná aktuální verze systému, virtuální stroj poskytnutý Matematicko-fyzikální fakultou UK, s operačním systémem Linux (přesněji Ubuntu 20.04.2 LTS).

Databáze je pak umístěna na serveru Google 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.

3.2 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.

3.2.1 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.

3.2.2 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")
  })
```

3.3 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.

3.3.1 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.

3.4 Routes (Směrovače)

Pro rozpoznání, která akce se má vykonat při různých dotazech, se porovnává jak adresa, tak 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 chybovou zprávu a její příčinu.

3.4.1 Uživatel

Tato metoda slouží pro administraci uživatelských účtů pro tuto aplikaci. Zakládání nového účtu může zavolat kdokoliv, avšak na zbylé akce, jako mazání nebo změnu hesla, má právo pouze majitel účtu a administrátor. Velkou bezpečnostní výhodou je naprostá nepřístupnost k heslu a přístupnost k sessionID jen pro administrátora nebo vlastníka.

3.4.2 Autorizace

Pro ověření, zda přihlášený uživatel má příslušná práva (a to na straně serveru, jelikož lokálně si je může libovolně 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).

3.4.3 Záznam

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

3.4.4 Stránka

Zobrazení stránky též nevyžaduje žádná práva, ale k jejich vytváření je třeba mít roli CMS editora, která definuje uživatele jakožto editora článků a příspěvků.

3.4.5 Nahrávání souborů

Pro nahrávání souborů na server je třeba práv pro zápis, stejně jako pro mazání. Zobrazení souboru žádná práva nevyžaduje, dokonce nenastavuje ani cross-origin policy, neboli nevyžaduje přístup ze stejné domény.

3.5 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 shodli na datovém typu a struktuře (předešlím 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.

3.5.1 Záznam

Každý záznam má přesný popis v online dokumentaci API, přesněji v pravé části, kde je pravidelně aktualizován.

3.5.2 Stránka

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

3.5.3 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 náhodného textu pro zvýšení bezpečnosti (sůl, anglicky *salt*). Výsledný hash (metodou *md5*) se uloží do databáze. Při přihlášení pak stačí porovnat hash soli a hesla s údaji v databázi. Pro ověřování práv neslouží heslo, ale sessionID. To se generuje unikátní pro každého uživatele při přihlášení, ale časem, nebo manuálně, může expirovat a poté 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 oprávnění, umožňuje vytváření uživatelských účtů a změnu hesla uživatele, stejně tak jeho údaje
- CMS - práva pro editační změny, co se týče obsahu stránek, jako jsou novinky a příspěvky

4. Implementace frontendu

Frontend byl vyvíjen jako single-page application pomocí programovacího jazyku JavaScript a knihovny React. A byl navržen tak, aby byl jednoduše ovladatelný, přehledný a především velmi rychlý.

4.1 Hostující server

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

4.1.1 Kompilace

Ačkoliv je JavaScript skriptovací jazyk a kompilaci provádí až za běhu, tak je možné jej zkompilovat předem. Při této kompilaci knihovny **webpack** a **babel** provedou několik zásadních kroků. Nejdůležitější z nich 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, čímž výrazně sníží čas potřebný k načtení stránky.

4.1.2 NPM

Celá aplikace má strukturu balíčku NPM (node package manager), tudíž ji lze snadno spustit kdekoli, kde je nainstalovaný *nodejs* (ve verzi minimálně 16.0). Zároveň udržuje pomocí souboru **package.json** přehled o potřebných knihovnách (dependencies) a obsahuje též různé spouštěcí skripty, např. skript pro build produkční verze aplikace.

4.2 Knihovny

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

4.2.1 React

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

Jejím základem je vykreslování pomocí technologie *single-page application* a je vhodná zejména pro aplikace, kde se často mění data. Využívá pozměněné JavaScriptové syntaxe známe jako JSX (JavaScript XML), která umožňuje pracovat s HTML tagy uvnitř JavaScriptového kódu, bez nutnosti práce s DOM objektem. Výhodou single-page aplikace je pak i optimalizovaná práce s DOM objektem, který je bottleneckem (úzkým hrdlem) při nesprávném použití (nebo lépe řečeno, při valné většině použití, které něco s DOM objektem dělá).

4.2.2 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. Design je to odlehčený a intuitivní. Součástí tohoto grafického jazyka jsou i lehce čitelné fonty (například Roboto), pochopitelné ikonky a systém barev a jejich kombinací.

4.2.3 i18n (internacionalizační knihovna)

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 optimální 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 dostávají 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.

4.2.4 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.

4.2.5 Webpack

Aby se z tak obrovského množství knihoven a souborů stal jediný spustitelný soubor, pomůže nám knihovna Webpack, která dokáže 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**.

4.3 Rozhraní

Jelikož se jedná o složitější systém a všechno nemůže být „naházené“ v jediném souboru, je použita hierarchie o třech hlavních úrovních. Nejvýše je **hlavní soubor celého webu**, který provádí routing (směrování) a obaluje celou aplikaci

pomocnými wrapery (např. *i18n* překladač, *Mui* pro jednotný styl, *SnackbarProvider* pro vyskakovací toasty a *CookiesProvider* pro jednotný přístup ke cookies). Pod ním jsou jednotlivé **scény** neboli stránky diametrálně rozdílných vlastností a funkcionalit používající poslední úroveň, jež jsou **komponenty**, malé na sobě nezávislé černé krabičky¹ poskytující určitou funkcionalitu.

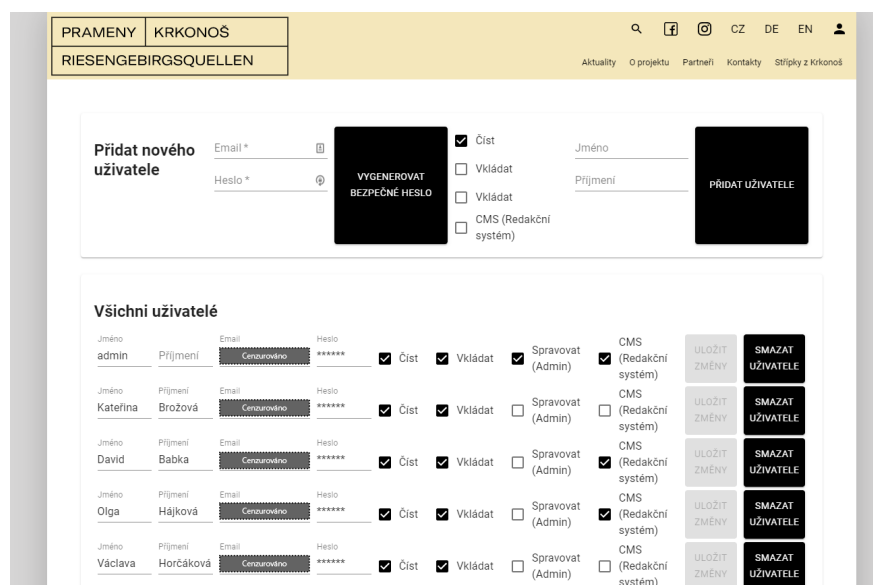
4.3.1 Scény

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.

Protože se hesla v databázi ukládají zašifrovaná, 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.



Obr. 4.1 | Frontend administrační scény

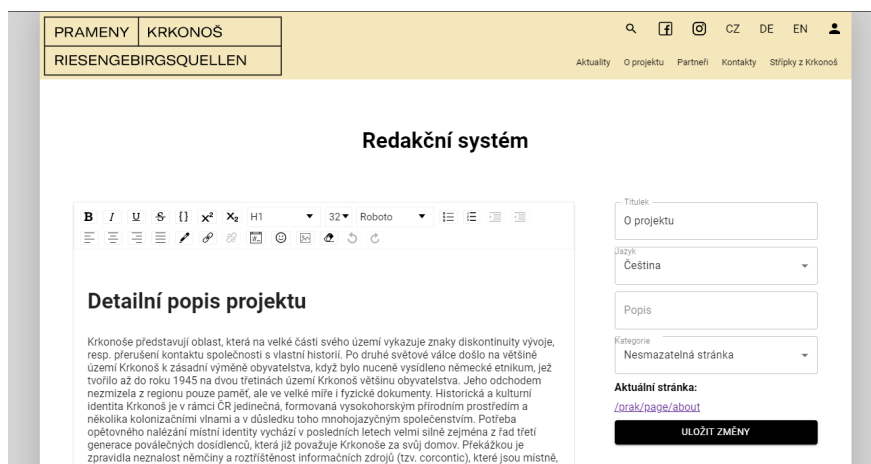
CMS

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 (a anglického „What you see is what you get“, přeloženo „Co vidíš, to dostaneš“). V pravém panelu pak zadá název, případně krátký popis, který se hodí např. pro aktuality a kategorii.

¹černa krabička - metoda u které není znám přesný algoritmus pro převod vstupních dat na výstupní

WYSIWYG editor 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.

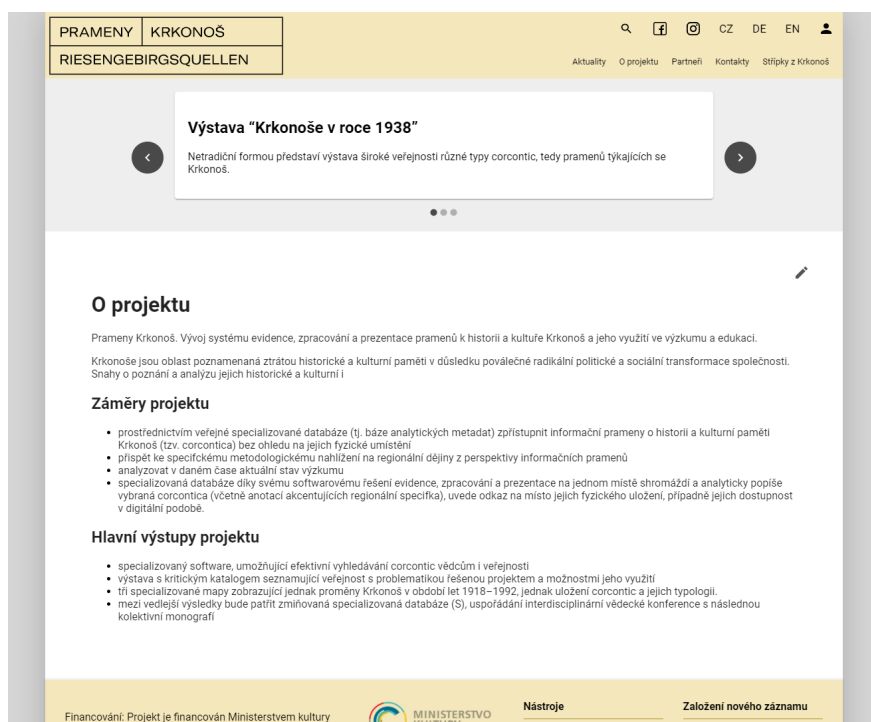


Obr. 4.2 | Frontend scény pro editaci stránek (CMS)

Domovská stránka

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

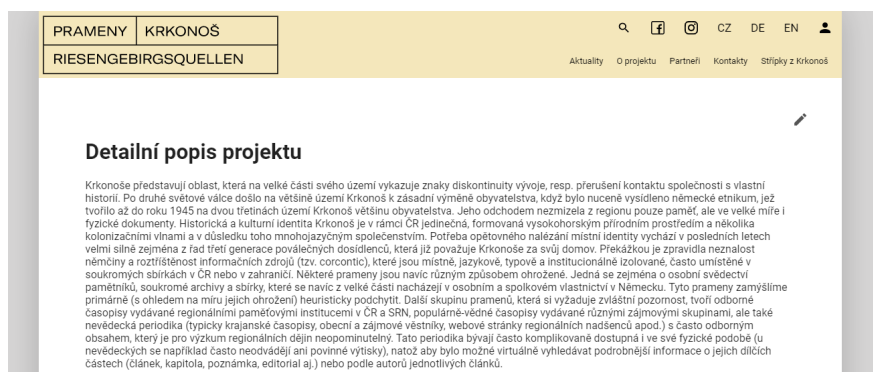
Na této stránce uživatel najde dynamické zobrazování novinek a krátkého popisu v horním panelu a stránku s názvem „homepage“, což je obyčejná stránka editovatelná v redakčním systému, a dovoluje tedy redaktorům měnit i domovskou stránku.



Obr. 4.3 | Frontend hlavní stránky aplikace

Zobrazení stránky

Každá stránka vytvořená v redakčním systému je přístupná na url adrese `.../prak/page/#jmenoStranky`. Zde si ji mohou 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. 4.4 | Zobrazení normální informativní stránky

Kontaktní formulář

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

Obr. 4.5 | 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 pět 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.

The screenshot shows a web application interface for searching. At the top, there is a header with navigation links: PRAMENY, KRKONOS, and RIESENGBIRGSQUELLEN. On the right, there are social media icons and language options (CZ, DE, EN). The main content area is titled "Vyhledávátko". Below the title, there is a search form with a dropdown menu labeled "Metadata" and a "VYHLEDAT" button. To the right of the form, there is a "Nápověda" (Help) section with instructions on how to use the search. Below the search form, there is a table of search results. The table has three columns: "Hlavní autor", "Hlavní název", and "Datum vydání nebo ...". The table contains five rows of results, each with a title and a date. At the bottom of the table, there is a pagination bar showing "1-5 of 5" and navigation arrows.

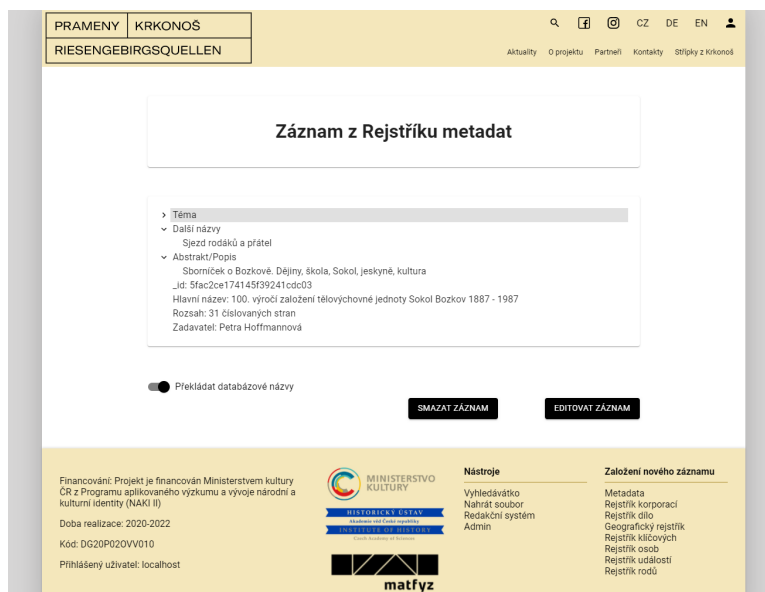
Hlavní autor	Hlavní název	Datum vydání nebo ...
	Paměti a folklorní sběry Jaroslava Večerníka z...	1950-1999
	Skica pro obec Horní Albeřice	
	"Tondova soukromá sbírka"	
	100. výročí založení tělovýchovné jednoty Sok...	
	90 let Církve československé husitské	

Obr. 4.6 | Frontend Vyhledávátka

Zobrazovátko

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

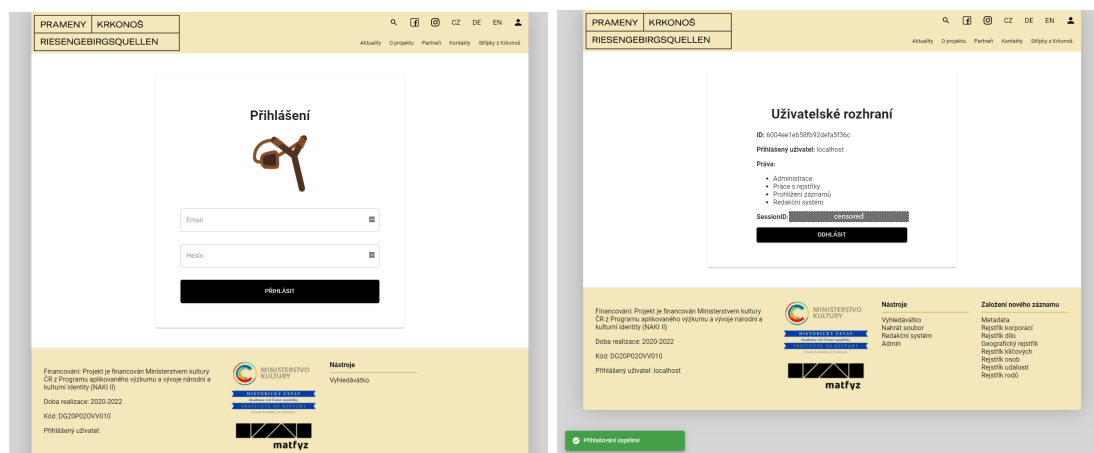
Po načtení se uživateli zobrazí záznam tak, jak je uložen v databázi, případně s přeloženými názvy, pokud to je pomocí spodního přepínače povoleno. Dole jsou též dvě tlačítka pro smazání záznamu a editaci záznamu, jež uživatele přesune do editačního rozhraní s aktuálním záznamem (samozřejmě jen pokud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



Obr. 4.7 | Frontend Zobrazovátko

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 na této stránce přihlásí, nebo již přihlášen byl, zobrazí se mu podrobnosti o jeho účtu, včetně dostupných oprávnění.



Obr. 4.8 | Přihlašovací stránka vlevo a stránka s informacemi o přihlášeném uživateli vpravo

Upravovátka

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 vpravo od sebe tlačítka plus a mínus, jehož vertikální velikost určuje velikost bloku, jež je jedním celkem. Plusové tlačítko přidává další datové pole, zatímco mínus datové pole vyprázdní a poté odebere z náhledu.

Datová pole jsou roztríděna do menších bloků a lze tlačítkem minimalizovat (vyplněná data se nezahodí, ale pouze skryjí), 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. Nahraný soubor se ihned přenáší na server a do příslušného pole automaticky zaznamená aktuální url nahrávaného souboru.

Po kliknutí na tlačítko 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.

PRAMENY KRKONOŠ
RIESENGBIRGSQUELLEN

Aktuality O projektu Partneři Kontakty Střípky z Krkonoš

Editace záznamu v Rejstříku metadat

Type
Svazek

Název

Hlavní název *
testing file

Další názvy
11111
2222
33333
44444

Fyzický popis

Forma
Naučná literatura
Forma: Naučná literatura
Komentovaná vydání

Rozsah

Rozměr

Formát/Jak to vypadá

Obsahová charakteristika

Citace (popisovaný objekt) (Metadata)
✓ Paměti a folklorní sběry Jaroslava

Autor

Hlavní autor (Rejstřík osob)



Téma (Rejstřík osob)

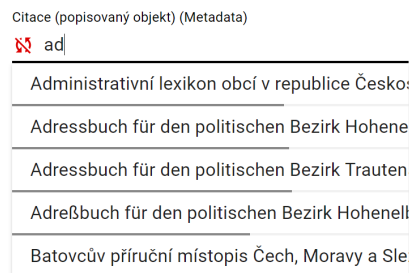
Obr. 4.9 | Frontend Upravovátka

4.3.2 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 (alternativní přístup k proměnným u komponent typu funkce, namísto typu třída).

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. 4.10 | Komponenta ComboBox

Zápatí

Jako každá správná stránka i tato má 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 „Vyhledávatko“ a po přihlášení se (viz obrázek 5.11) menu rozroste o další rychlé odkazy.



Obr. 4.11 | 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. 4.12 | 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. 4.13 | 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. 4.14 | 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 popis, 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.

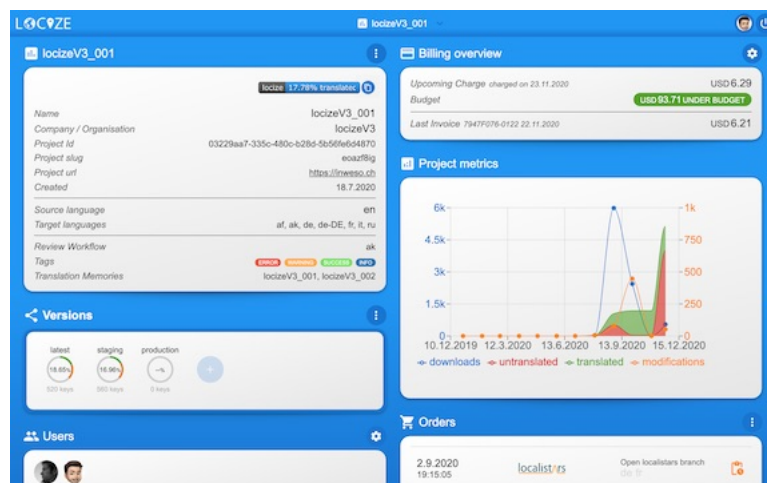
4.3.3 Moduly

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.

4.4 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 aplikuje na požadovaných místech. Pokud ovšem překlad chybí, použije se nejblíže možná interpretace, podobný jazyk nebo zdrojová adresa překladu a vývojáři ukáže zprávu o chybějícím překladu.



Obr. 4.15 | Aplikace Locize umožňující správu překladů knihovny i18n

5. Provázání Backendu a Frontendu, API

5.1 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.

Levá část popisuje volání API.

Každá sekce tu má příslušné http metody (GET, POST atd.), jejichž účel a formát je popsána uvnitř bloku, spolu s dalšími možnými 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.

Pravá 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:

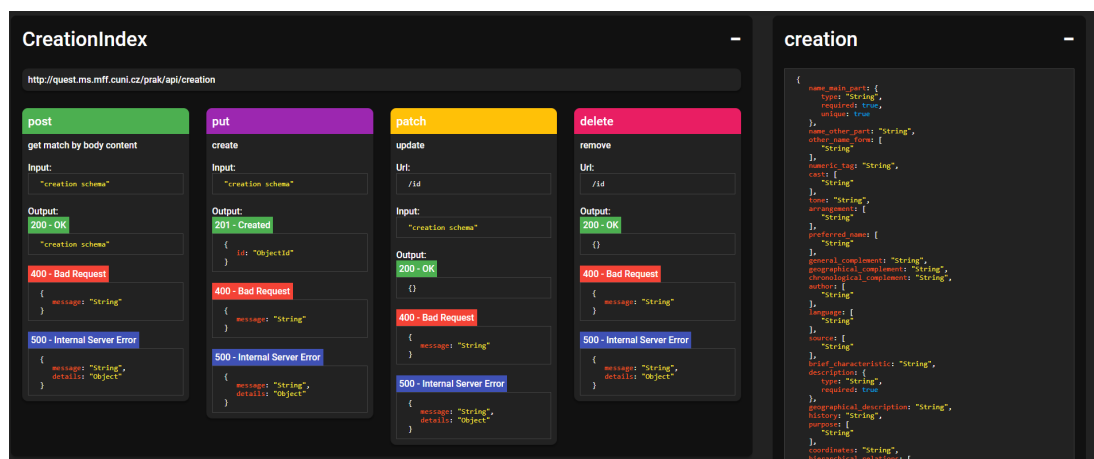
- **type**: datový typ
- **required**: true, pokud je tato položka povinná
- **unique**: true, pokud se zadaná hodnota nesmí shodovat s již existující položkou v DB
- **ref**: název schématu, na který se ID odkazuje
- **refPath**: speciální ref, umožňující uživateli zadat i název schématu, vůči kterému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

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

5.1.1 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 obarveny, aby uživateli poskytly rychlejší orientaci v kódu.



Obr. 5.1 | Dokumentace API jako webová stránka

5.2 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.

5.3 API

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

5.3.1 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.

5.4 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*.

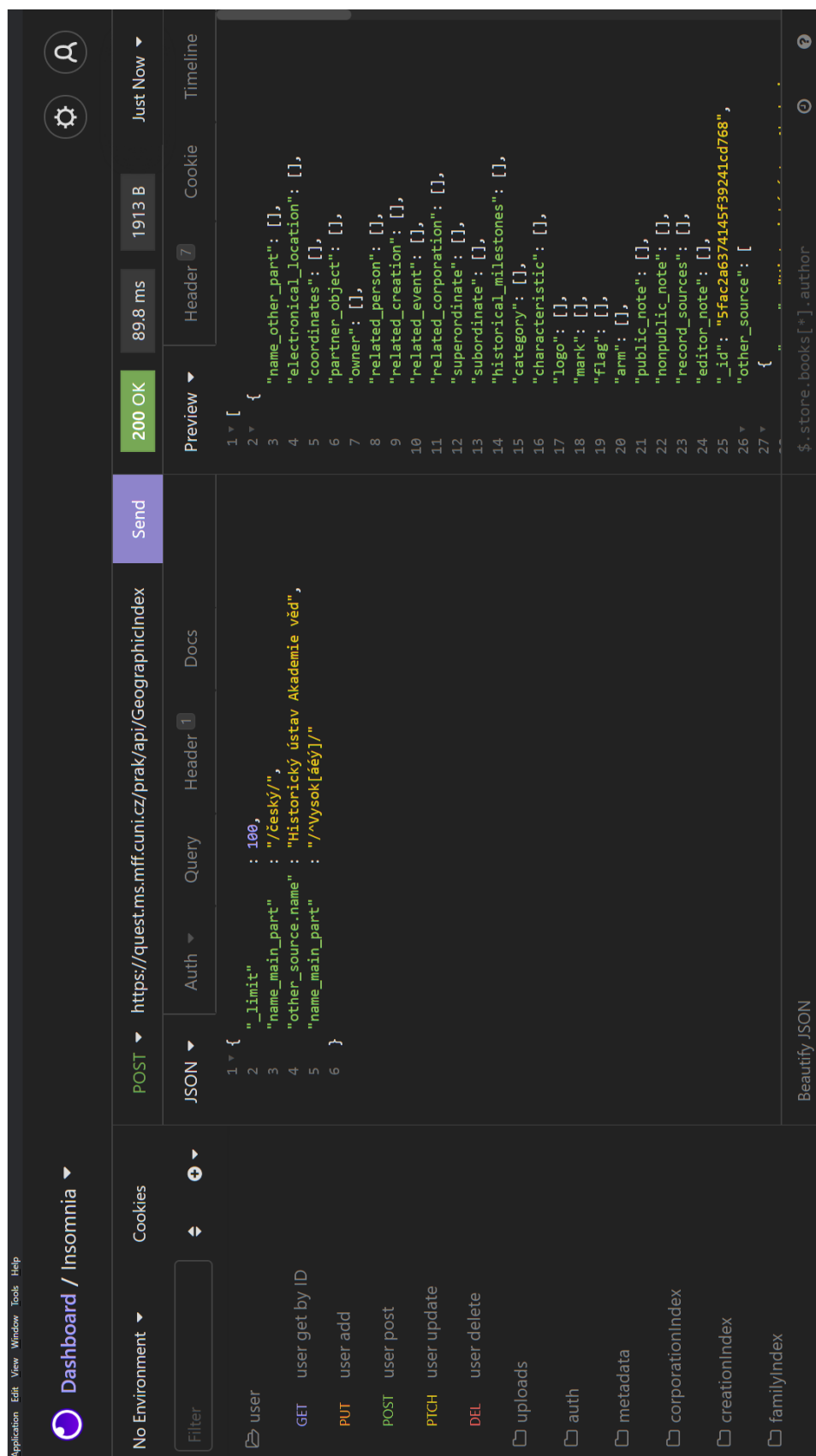
5.4.1 Fetch

Dokumentace metody: developer.mozilla.org

Příklad použití:

```
const url = "/prak/api/metadata"
fetch(url, {
  method: "POST",
  headers: { "Content-Type": "application/json" },
  body: JSON.stringify({
    "_limit" : 100, // max returned count by default is 5
    "name"   : "... "
  }),
})
.then(response => {
  if(!response.ok) throw response
  return response.json()
})
.then(response => {
  console.log(response)
})
.catch(error => {
  console.error(error)
})
```

5.4.2 Existující programy pro práci a testování API



Obr. 5.2 | Program Insomnia s vyplněným dotazem pro odeslání požadavku na API

6. Moduly

6.1 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.

6.2 Aktuálně nasazené moduly

6.2.1 Modul hologram

Tento modul bude sloužit na výstavě Pramenů Krkonoš k zobrazování 3D objektů ve formě hologramu, který se bude pomalu otáčet a ukáže se tak návštěvníkovi ze všech stran.

Načítání modulu

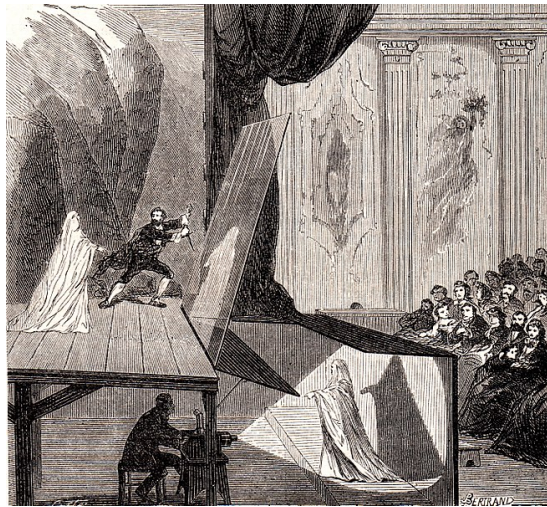
Tento modul obsahuje velkou knihovnu, a tudíž není efektivní jej mít v primární aplikaci. Protože by se tyto knihovny načítaly, i když by uživatel s tímto modulem neměl v plánu pracovat. Což většinu času nebude chtít, a tudíž by to pouze vedlo ke zpomalení systému. Systémem *Lazy* načítání je tedy celý 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. 6.1 | 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/ahttps://texturehaven.com/>.

V programu blender pak byly tyto textury namapovány na objekt a bylo do nich zapečeno ambient occlusion (stíny vytvořené stísňeným prostorem).

6.2.2 Prohlížeč map

Dalším historickým prvkem, který bychom návštěvníkům této aplikace poskytli, jsou 2D mapy z 19. století. Uživatel si může vybrat mapu a tu si velmi podrobně prohlízet 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“. Každou mapu jsme tedy pečlivě zrekonstruovali, při zachování všech detailů a historických přesností a bez jejich upravování či umělého přidávání.

7. Instalace a spuštění

7.1 Prerekvizity

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

7.1.1 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
```

7.1.2 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.

7.1.3 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/;
  }
}
```

7.1.4 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>.

7.1.5 Chocolatey balík

Pro ty, kteří znají a mají nainstalovaný balíčkovací manager „chocolatey“ pro Windows, stačí spustit tento jednoduchý kód pro snadnou instalaci.

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

7.2 Zdrojové soubory

Veškerá zdrojové soubory (kromě na platformě závislé konfiguraci Nginx) jsou uloženy na GitHubu <https://github.com/EbrithilNogare/PraK>

7.3 Instalace

Ve složce se zdrojovými soubory spustíme následující příkazy:

```
##### frontend npm install
cd frontend
npm install

##### frontend build
npm run build

##### backend npm install
cd ../backend
npm install
node server
```

8. Řešení

8.1 Výsledný web

Funkční web je možné si zobrazit na adrese <http://quest.ms.mff.cuni.cz/prak/homepage>. Nezapomínejte, že pro zobrazení většiny stránek jsou potřeba vyšší oprávnění, než má nepřihlášený uživatel.

8.2 Uživatelská dokumentace

8.2.1 Návštěvník (nepřihlášený uživatel)

Při vstupu na stránku se návštěvník dostává na hlavní webovou stránku, na které si může přečíst hlavní informace o projektu nebo aktuality.

Obsah stránky je převážně v češtině, ale lze přepnout jazyk ovládacích prvků pomocí tlačítek CZ, DE a EN vpravo nahoře. Pro zobrazení záznamů slouží ikonka lupy v pravém horním rohu, která směřuje na vyhledávací rozhraní, ve kterém lze záznamy vyhledat (systém podporuje i regexp výrazy), setřídít pomocí kliknutí na záhlaví požadovaného sloupce a následně požadovaný záznam i zobrazit v jeho detailní podobě.

V hlavním menu najdeme i několik odkazů na stránky s relevantními informacemi a rubriky jako jsou *Novinky* nebo *Střípky z Krkonoš*.

Trochu schované jsou pak odkazy na moduly:

- 3D model hologram
<http://quest.ms.mff.cuni.cz/prak/modules/hologram/>
- 2D mapa Krkonoš 19. století
<http://quest.ms.mff.cuni.cz/prak/maps>

8.2.2 Zadavatel

Zadavatel se stejně jako uživatel dokáže dostat k zobrazení detailu záznamu, navíc zde má ale i tlačítko pro editaci záznamu, jež jej přesune do editačního rozhraní. Zde může vyplňovat pole a následně záznam uložit. Specifikace atypických polí je v kapitole 5.3.2.

K vytvoření nového záznamu má zadavatel v patičce webu připraveny odkazy do scény *Zadávatka*, které je identické se vzhledem editačního prostředí.

V případě potřeby může zadavatel nahrát soubor na server pomocí odkazu v patičce webu s názvem *Nahrát soubor*. Po kliknutí na tlačítko s ikonkou šipky se mu zobrazí okno pro zvolení souboru a po jeho vybrání se soubor odešle na server a uživateli se zobrazí přímý odkaz na nahraný soubor, který může začít ihned používat.

8.2.3 Redaktor

Pro přístup do redakčního systému stránky stačí v jejím pravém rohu kliknout na ikonku tužky (viz *Obr. 5.4*) nebo použít odkaz *Redakční systém* v zápatí

stránky, kde se mu zobrazí veškeré stránky webu a odtud si může vybrat, kterou chce editovat. Po kliknutí na ikonku oka stránku zobrazí, případně po kliknutí na ikonku tužky ji začne editovat. Měnit lze vše kromě již existující url. Více o redakčním systému v *kapitole 5.3.1 - CMS*.

8.2.4 Admin

Veškerá správa uživatelů, jejich informací a oprávnění spadá pod administracní rozhraní, přístupné přes odkaz *Admin* v zápatí stránky.

Nového uživatele lze přidat v horní části administracního rozhraní, kam se zadají potřebné údaje a uživatel se uloží. Níže pak nalezneme seznam uživatelů a informace o nich, včetně možnosti změny jejich údajů a nebo smazání jejich účtů.

8.3 Co by se dalo vylepšit a přidat

Velké knihovní systémy mají velkou řadu funkcí pro práci nad daty, jako jsou např. **hromadné úpravy** nebo práce se **šablonami**. Ty ačkoliv nejsou aktuálně implementovány, tak je lze snadno vytvořit díky dobře vytvořenému API.

Další vylepšení by se dalo udělat pro data v záznamu, ačkoliv databáze je připravená v záznamu uchovat a vrátit cokoliv v jakémkoliv formátu, tak na frontendu je nutný zásah programátora, aby přidal pole, bylo by uživatelsky přívětivější, aby pole mohl přidávat i obyčejný zadavatel přes „klikací rozhraní“.

Závěr

Předmětem této práce byl návrh a vývoj softwarového řešení pro tematický digitální archiv. týkající se sběru, uchování a prezentaci dokumentů z oblasti Krkonoš. Výsledné softwarové řešení mělo agregovat databázi pro ukládání různých typů dokumentů (resp. objektů) a web pro prezentaci a úpravu těchto dat. - Přičemž rozhraní pro popis a evidenci dokumentů měl být navržen tak, aby byla oddělena vrstva pro běžné prohlížení dokumentů, vkládání záznamů o dokumentech a vrstva (prostředí) pro správu a editaci těchto záznamů. Navrhovaný software měl zohledňovat typologii dokumentů a zároveň reflektovat různé (oborové) normy pro bibliografický popis dokumentů. Při návrhu vkládacího modulu bylo třeba zohlednit normy knihovnické, muzejní i archivní. V tomto směru navržený software disponuje velkou inovativností v přístupu k popisu dokumentů. Zároveň bylo třeba při návrhu software (a jeho jednotlivých modulů) třeba zohlednit požadavek historiků na vzájemnou provázanost jednotlivých vložených údajů. - Tj. vytvořit takovou strukturu vkládaných záznamů, aby mezi bibliografickými údaji a rejstříky bylo možné z různých polí vzájemně odkazovat na jiná pole, resp. aby bylo možno odkazovat mezi bibliografickým záznamem a rejstříkem, a taktéž mezi rejstříky navzájem.

Výsledkem je webové rozhraní a serverová aplikace poskytující API pro komunikaci s jednotlivými moduly výsledného software. V práci je popsána jak implementace backendu, tak i implementace frontendu, ale i jejich vzájemné provázání prostřednictvím API. Dále jsou v práci popsány další přídatné moduly k výslednému software.

Vedle hlavní frontend aplikace je možné si načíst i oddělené moduly. První modul poskytuje hologram 3D modelu, který se hodí zejména na výstavy. Druhý modul přináší interaktivní prohlížení 2D map z Krkonoš 19. století.

Serverová aplikace poskytuje nejen samotnou webovou aplikaci, ale i moduly nezávislé na hlavní aplikaci. Ty pak komunikují se serverem pomocí API, které je dobře zdokumentováno a poskytuje tak přístup k datům i pro externí programátory a jejich programy.

Beta verze systému byla testována reálnými uživateli a bylo do ní zadáno přes 35000 záznamů různých typů a velikostí.

Během vývoje betaverze byl software testován „ostrými“ daty. Do systému bylo uživateli zadáno přes 35 000 záznamů různých typů, velikostí či úrovně kompletnosti. Na základě poznatků od uživatelů byl software laděn směrem k tomu, aby rozhraní pro vkládání dat a pro jejich editaci bylo uživatelsky co nejprívětivější a co nejintuitivnější.

Seznam použité literatury

- BARTOŠEK, M. (2001). *Digitální knihovny*. Brno : Masarykova univerzita, <http://www.ics.muni.cz/mba/dl-datakon01.pdf>.
- 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.
- CZ (2008-2014). *Digitální knihovny*. NÁRODNÍ KNIHOVNA ČR. Knihovny.cz, <http://archiv.knihovny.cz/digitalni-knihovny>.
- KREJČÍŘ, V. (2006). *Systémy pro tvorbu digitálních knihoven*. In: INFORUM, http://www.inforum.cz/pdf/2006/Krejcir_Vlastimil.pdf.
- ROBINSON, D. B. a LYN (2012). *An introduction to information science*. London: Facet. ISBN 978-185-6048-101.

Seznam obrázků

1.1	logo systému Koha ze stránky https://www.manelibit.org/node/77	5
1.2	logo systému SLIMS ze stránky https://slims.web.id/	5
1.3	logo systému Evergreen ze stránky https://eg-wiki.osvobozena-knihovna.cz/	5
2.1	Diagram systému	9
4.1	Frontend administrační scény	16
4.2	Frontend scény pro editaci stránek (CMS)	17
4.3	Frontend hlavní stránky aplikace	17
4.4	Zobrazení normální informativní stránky	18
4.5	Kontaktní formulář	18
4.6	Frontend Vyhledávátka	19
4.7	Frontend Zobrazovátka	20
4.8	Přihlašovací stránka vlevo a stránka s informacemi o přihlášeném uživateli vpravo	20
4.9	Frontend Upravovátka	21
4.10	Komponenta ComboBox	22
4.11	Zápatí stránky	22
4.12	Navigační menu	23
4.13	Pole s kontrolou validity dat	23
4.14	Pole pro nahrávání souborů	23
4.15	Aplikace Locize umožňující správu překladů knihovny i18n, zdroj: https://locize.com/	24
5.1	Dokumentace API jako webová stránka	26
5.2	Program Insomnia s vyplněným dotazem pro odeslání požadavku na API	28
6.1	Peppers Ghost efekt - zdroj: https://commons.wikimedia.org/wiki/File:Peppers_Ghost.jpg	30

Seznam použitých zkratek

Administrativní

MK ČR - NAKI	Ministerstvo kultury ČR - Národní Kulturní Identita
ČUZK	Český úřad zeměměřický a katastrální

Programátorské

API	Application Programming Interface
Regexp	Regular expression, více zde: https://en.wikipedia.org/wiki/Regular_expression
DB	Databáze
cors	Cross-origin Resource Sharing, soubor pravidel pro komunikaci napříč rozdílnými doménami
md5	message-digest algorithm, rodina hašovacích funkcí v kryptografii
WYSIWYG	What you see is what you get, neboli „co vidíš, to dostaneš“