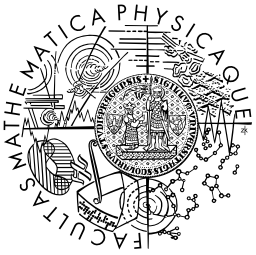


MATEMATICKO-FYZIKÁLNÍ
FAKULTA
Univerzita Karlova

BAKALÁŘSKÁ PRÁCE

David Nápravník

Softwarové řešení digitálních archivů

Katedra teoretické informatiky a matematické logiky

Vedoucí bakalářské práce: Mgr. Macková Kateřina

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 Podekovani:
Petra Hoffmannová
Kateřina Macková

Název práce: Softwarové řešení digitálních archivů

Autor: David Nápravník

Katedra: Katedra teoretické informatiky a matematické logiky

Vedoucí bakalářské práce: Mgr. Macková Kateřina, katedra

Abstrakt: TODO Abstrakt cz

Klíčová slova: digitální archiv web databáze

Title: Software solution for digital archives

Author: David Nápravník

Department: Department of Theoretical Computer Science and Mathematical Logic

Supervisor: Mgr. Macková Kateřina, department

Abstract: TODO Abstrakt en

Keywords: digital archive web database

Obsah

Úvod

1. zadani

Cilem tohoto projektu je vytvoreni weboveho rozhrani pro ukladani a zobrazovani historickych i soucasnych zaznamu z oblasti Krkonos.

2. existujici produkty

Vybirame jen z open source produktu, abychom mohli nahlednout do jejich kodu a lepe porozumet implementaci jejich komponent. Lepe se pro takovy system vyvyjeji nezavisle moduly a obvikle maji o dost vetsi komunitu vyvojaru a prispevatelu.

2.1 KOHA

url: http://www.koha.cz/
Koha je nejrozsirenejsi open source system s sirokou komunitou. Byla vyvynuta na Novem Zelande roku 2000. Ale je stale udrzovana a stale rozsirovana (nejnovejsi update je z konce roku 2020)
Pouziva SQL tabulky. Psana v Perlu, na frontendu vyuzivajici javascript (ale neni ho tolik).

2.2 Evergreen

url: https://eg-wiki.osvobozena-knihovna.cz
Dalsi z rady knihovnich systemu. Pouzivan napr Pedagogickou a Teologickou Fakultou v Praze.
Pouziva sql databazi, vykreslovani na serveru, nema uzivatelsky privetive prostredi.

2.3 SLIMS

url: https://slims.web.id/
System se zakladni funkcionalitou a privetivym vzhledem. Neni v cestine. Neni primarne urcet jako knihovni system, spis je to univerzalni system na jakykoliv system, proto neni az tak efektivni a jeho nastavovani by zabralo mnoho casu.

2.4 Zaverem

Evergreen a SLIMS jsou systemy, ktere poterbuji silne vyskolenou osobu, aby se o system starala, narozdil od KOHY, ktera je intuitivnejsi a pro nove uzivatele privetivejsi. Pri zachovani stejne, mozna i lepsi funkcionality.

3. Vyber technologii

3.1 Frontend

Vzhledem k rychle se menicim trendum v oblasti webovych technologii, jsem se rozhodl jit cestou kterou vyvynul Facebook a jeho tym programatoru. Jedna se o technologii **Single page aplication**, jez je implementovana v knihovne **React**.

3.1.1 Single page aplication

Single page aplication je technologie umoznujici vykresleni jine stranky, bez nutnosti posilani requestu na server. Uzivatel si pri prvnim spusteni webu stahne cely balicek webu a pri opetovnem nascteni vetsinou saha jen do sve cache. Javascriptova knihovna (v tomto pripade React) pote stranku prekresluje pri uzivatelske interakci. V pripade nutnosti stazeni / posilani dat mezi serverem a uzivatelem (napr. editace zaznamu, nebo nacteni existujiciho zaznamu) se vola pouze request k API webove sluzby a telo requestu obsahuje pouze uzitecne informace.

3.1.2 React

Knihovna React je knihovna poskytujici single page aplication technologii. Jedna se o dobre udrzovanou knihovnu, jez byla vyvinuta Facebookem, jakozto nahrada zastaraleho konceptu renderovani stranky na serveru. Diky tomu servery nemuseli ztracet vykon s kazdou zmenou na strance a vykon k renderovani se bere z PC uzivatele. Jadro teto knihovny je velmi dobre optimalizovate a poskytuje i radu debugovacich nastroju, coz je pro vetsi projekty nepostradatelna vyhoda.

3.1.3 Dalsi mozne technologie

Velmi casto vykreslovani stranek probiha na serveru, se systemy jako jsou WordPress, psany PHP. Takovyto system je velmi dobre uzivatelsky privetivy, ale z pohledu vykonu ma velmi obrovsky overhead. V pripade implementace knihovniho systemu by to znamenalo vykreslovat celou stranku (hlavicku, telo i zapati) na serveru, na druhe strane single page aplication nic nerenderuje, pouze posle informaci o knize.

3.2 Backend

Mit single page aplikaci na frontendu znamena, ze na backendu musi existovat API, od ktereho bude frontend cerpat data. Navic zde potrebujeme i system pro staticke odesilani baliku cele webove stranky. V ramci udrzitelnosti jsem se rozhodl vyuzit jazyk Javascript stejny jako pro frontend. Express.js je knihovna ktera umoznuje komplexni spravu requestu a stala se tudiz jasnou volbou.

3.2.1 Express.js

Express.js poskytuje odesilani statickych stranek (Reactiho baliku v nasem pripade), custom requesty pro rozmanite API a take odesilani a lokalni ukladani statickych souboru, jako obrazku, word i pdf dokumentu atd.

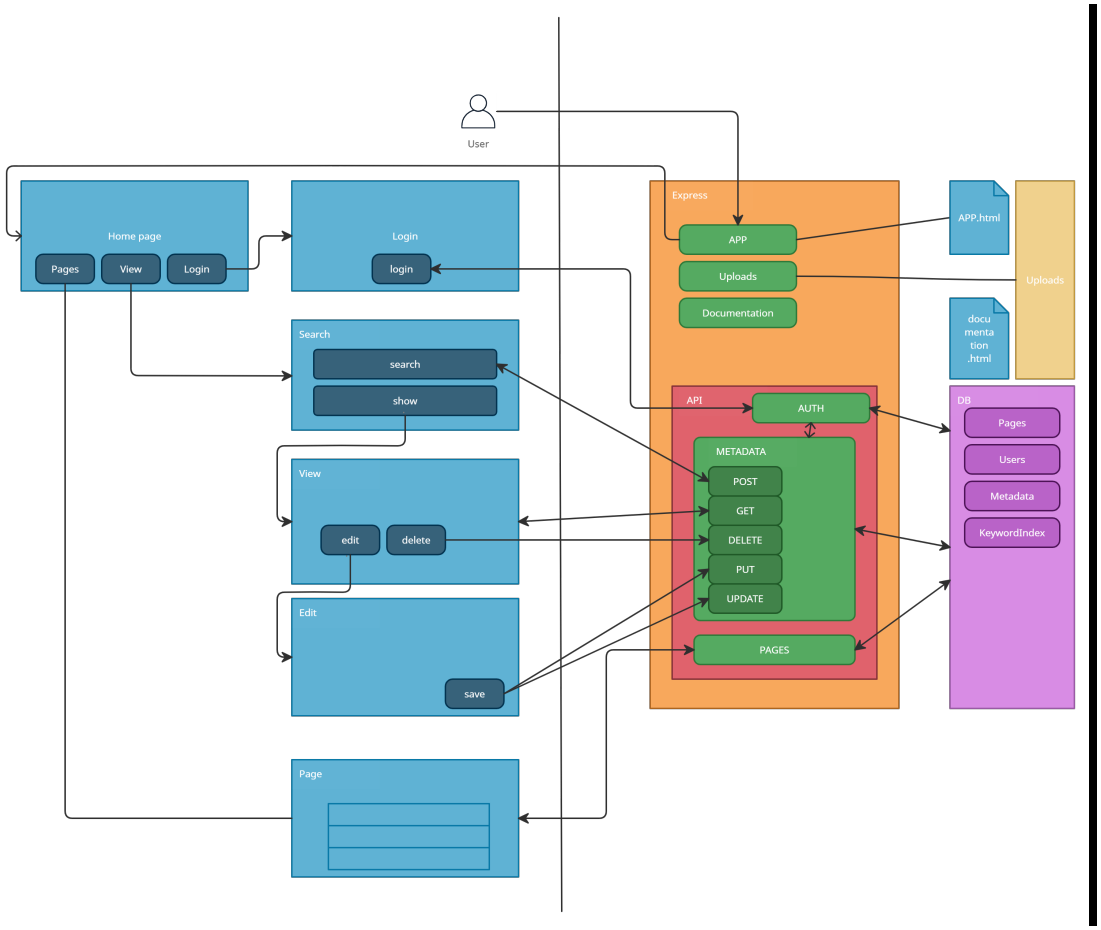
3.2.2 MongoDB

MongoDB je databazovy system typu non-sql. Coz primarne znamena, ze data neuchovava v tabulkach, ale v tkz. schematech. Coz ma mnoho vyhod, nejvetsi je, ze nekompletni zaznamy nezabiraji svymi nevyplnenymi daty misto v DB a uklada se opravdu jen to co je potreba. Dalsi vyhodou, je styl ukladani dat a komunikace s DB. Databaze si data uchovava ve formatu BSON (binarni JSON rozsireny o datove typy). O data si aplikace zada pomoci query, ktera je zcela odlisna od tech u sql-like databazi, primarne se zde neposila query ve formatu string ale JSON, diky cemuz napr. nenastane znama SQL injection. Znovu ve formatu JSON pote data vraci aplikaci.

3.2.3 Dalsi mozne technologie

Diky oddeleni frontendu a backendu (narozdil napr. u WordPressu) je mozne na backend nasadit temer cokoliv co umi posilat requesty. Prikladem tomu muzou byt scripty v jazicich PHP, C#, Python, nebo Perl. Ale vzhledem k tomu, ze jednim z modulu bude neuronova sit na vyhledavani, vybiral jsem mezi Pythonem a JavaScriptem, jakozto 2mi jazyky, ktere maji velmi dobre knihovny pro praci s neuronovymi sitemi.

4. diagram systemu



5. implementace backendu

5.1 server

5.2 knihovny

5.2.1 Express.js

nadstavba pro nahravani souboru

5.2.2 mongoose

5.3 dokumentace

/api/documentation

5.3.1 vlastni mini knihovna pro dokumentaci

5.4 routes

5.4.1 uzivatel

5.4.2 autorizace

5.4.3 zaznam

5.4.4 stranka

5.4.5 nahravani souboru

5.5 models

5.5.1 zaznam

5.5.2 stranka

5.5.3 uzivatel

6. implementace frontendu

6.1 server

6.1.1 kompilace

6.1.2 npm

6.2 knihovny

6.2.1 React

6.2.2 Material-ui

6.2.3 i18n

6.2.4 babel

6.2.5 webpack

6.3 rozhrani

6.3.1 sceny

amin

cms

homepage

page

login

kontakt

search

show

edit

6.3.2 komponenty

KomboBox

Zapati

Navigacni menu

validationTextField

Uploadfile

Indexy

6.3.3 moduly

hologram

6.4 Lokalizace

7. Moduly

7.1 obecne pridavani modulu

7.2 aktualni moduly

7.2.1 hologram

nacitani modulu

Tento modul obsahuje velkou knihovnu a tudiz neni efektivni jej mit v defaultni aplikaci. Protoze by se tyto knihovny nacitaly, i kdyz by uzivatel s timto modulem nemel v planu pracovat. Coz vetsinu casu nebude chtit a tudiz by to pouze vedlo k zpomaleni systemu. Systemem LAZY loading je tedy cely modul odriznut a nacita se az explicitne pri jeho pouziti.

8. Provazani Backendu a Frontendu, API

8.1 Dokumentace online

Aktualni dokumentace, tak aby mohla byt casem aktualizovana a mohly do ni byt pripisovany dalsi veci, se nachazi na webu quest.ms.mff.cuni.cz/prak/api/documentation. Dokumentace je rozdelena na dve casti.

Prvni cast popisuje volani API. Kazda metoda (GET, POST atd.) ma svuj vlastni ucel jez je popsan uvnitr, url, na kterou se dotazovat a mozne parametry. Spolu s formatem requestu je i zde format odpovedi. Podle kodu zjistime, jak uspesny byl nas pozadavek. Kody jsou standartni podle "http status codes".

- **2xx** Vsechno dopadlo dobre
- **4xx** Chyba je na strane klienta
- **5xx** Chyba je na strane serveru

V pripade uspechu (kod 200 - OK) se odesle odpoved na dotaz, nebo nic pokud request nemel za funkci neco vracet. V pripade neuspechu pak v odpovedi najdeme zpravu o chybe ktera nastala. Obvykle to u kodu 400 byva, ze odesilame duplicitni zaznam.

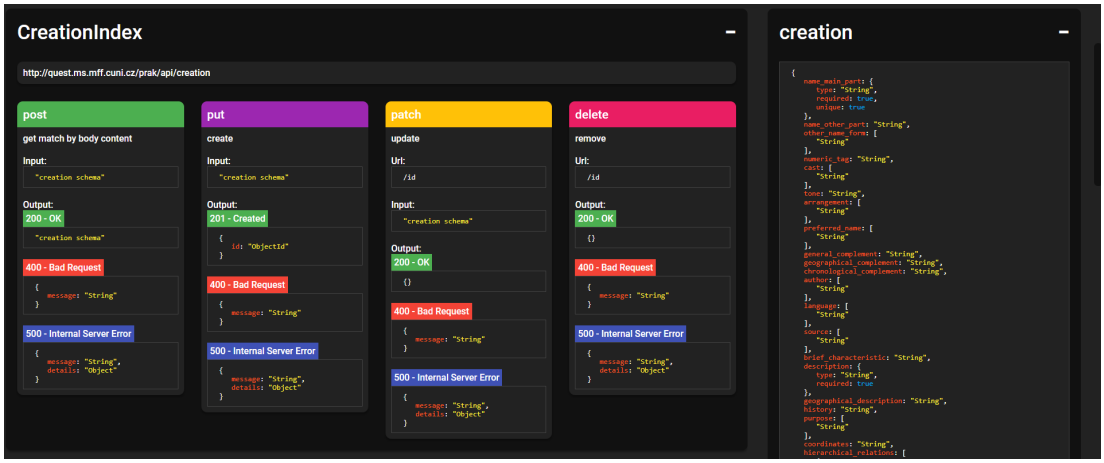
Druha cast popisuje strukturu schemat jednotlivych modelu
Jelikoz databaze ma datove formaty, zatimco JSON soubor ne (nebo alespon ne tak rozsahle) musi i odesilana data mit spravny format, nebo alespon byt validni po automatickem pretypovani. Schema tvori JSON objekt popisujici schema. pokud je datovy typ polozky objekt, bud se opravdu jedna o objekt, nebo se jedna pouze o upresneni datoveho typu. Klicovimy slovy jsou:

- **type**: datovy typ
- **required**: true, pokud je tato polozka povinna
- **unique**: true, pokud se zadana hodnota, nesmi schodovat s jiz existujici polozkou v DB
- **ref**: nazev schematu, na ktery se ID odkazuje
- **refPath**: specialni ref, umoznujici uzivateli zadat i nazev schematu, vuci kteremu se odkazuje

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

8.1.1 sowtware na online dokumentaci

Pro moznost stazeni dokumentace a prohlizeni offline, je vse zabaleno do jednoho html souboru. Uvnitr je zdrojovy kod programu, jez vykresluje stranku a zaroven data dokumentace samotne.
Program vykresluje veskere polozky s daty dokumentace. Bloky zdrojovych kodu jsou vysazeny monospacem a obarveny, aby uzivateli poskytly rychlejsi orientaci v kodu.



8.2 Backend

Na backendu je spusteny express server (bezici pod nodejs), ktery zachytava requesty s url ...prak/api/... . podle cesty, jez je uvedena za /api se request predava prislosnemu routeru. Vyjimku tvori adresa .../api/documentation, ktera rovnou preposila soubor s dokumentaci a neni tedy pro ziskani dokumentace trba rozumet systemu hloubeji.

Pri prevzeti requestu jednim z mnoha routeru, se porovnava typ requestu (POST, PUT atd.) a pripadne detaily cesty v url. Pri plne nalezeni schody se provede overeni prav pokud je potreba. Pokud request ma pozadovana opravneni je provedena prislusna funkce a uzivateli je vracena odpoved / potvrzeni o uspechu. V pripade ze request nema prislusna opravneni, je vracen kod 401. V pripade ze se na serveru neco pokazi vrati se kod 400, nebo 500, podle typu problemu.

8.3 API

dostupne na adrese quest.ms.mff.cuni.cz/prak/api/

8.3.1 Autentifikace

S kazym requestem prichazi v hlavicce i cookies. Pro autentifikaci pouzivam cookie se jmenem "sessionID". Podle nej se najde prislusny uzivatel a porovnaji se jeho predava a prava potrebna pro vykonani funkce. Pokud jsou prava nedostatecna, vrati se odpoved 401, v pripade spravneho opravneni, router vykona

funkci jez danemu requestu prislusi a pokud se nepokazi nic jineho, vrati validni odpoved.

8.4 Frontend a volani API

Jelikoz je cely projek zamyslen jako webova aplikace, nejstandartnejsi pouziti je volani API pomoci JS funkce fetch(), coz je pouze zastita pro XMLHttpRequest. Ale je mozne jej volat jakkoliv jinak, dokud to bude validni http request.

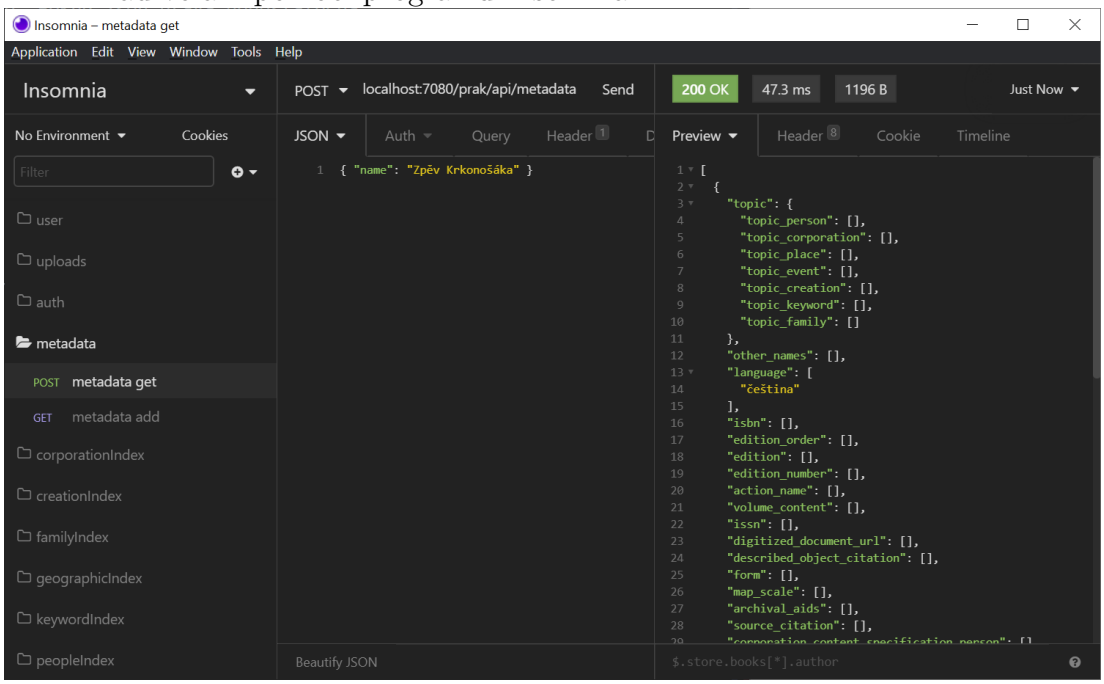
8.4.1 Fetch

Dokumentace metody: mozilla
Priklad pouziti:

```
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)
})
```

8.4.2 Existujici programy pro praci a testovani APi

Priklad volani pomoci programu Insomnia:



9. instalace a spusteni

10. vysledny web

11. vyuziti

Závěr

Seznam použité literatury

Seznam obrázků

Seznam tabulek

Seznam použitých zkratek

A. Přílohy

A.1 První příloha