Gymnázium, Praha 6, Arabská 14

Programování

MATURITNÍ PRÁCE

Viktor Korladinov

květen 2020

Gymnázium, Praha 6, Arabská 14

MATURITNÍ PRÁCE

Předmět: Programování
Téma: Knihovní systém

Autor: Viktor Korladinov
Třída: 4.E
Školní rok: 2019/2020
Třídní učitelka: Mgr. Jana Urzová, Ph.D.
Konzultant: Mgr. Jan Lána

Prohlášení

Prohlašuji, že jsem jediným autorem tohoto projektu, všechny citace jsou řádně označené a všechna použitá literatura a další zdroje jsou v práci uvedené. Tímto dle zákona 121/2000 Sb. (tzv. Autorský zákon) ve znění pozdějších předpisů uděluji bezúplatně škole Gymnázium, Praha 6, Arabská 14 oprávnění k výkonu práva na rozmnožování díla (§ 13) a práva na sdělování díla veřejnosti (§ 18) na dobu časově neomezenou a bez omezení územního rozsahu.

V Praze dne ..... …….....

Viktor Korladinov

Anotace

Rychlý a uživatelsky přívětivý školní knihovní systém, který pomůže žákům nejen zjistit, zda knihu, kterou hledají v knihovně je, aniž by museli osobně knihovnu navštívit, ale také knížku zarezervovat na několik dny. Kromě rezervačního systému software umožní chytře přidávat nové knihy – sám vyplní všechny potřebné informace jenom podle ISBN nebo název knihy.

Abstract

A swift and user-friendly school library system, that will not only help students find out whether the book they are searching for is present at their school library without physically checking for it themselves, but also allow them to reserve the aforementioned book for a few days, making sure they have time to get it. A main feature of this software will be “smart add” – just by an ISBN or a title the system should be able to find all needed information alone, saving a lot of the librarian’s time.

Keywords

Library; Library system; ISBN

Zadání

Knihovní systém, v němž si studenti budou moct ověřit, zdali je knížka, kterou hledají, ve školní knihovně, nebo jestli je půjčena. Poté si budou moct svůj výběr zarezervovat na několik dni, během kterých by si měli knížku přijít vyzvednout. Učitelovi webová aplikace umožní půjčovat knihy, a také je označovat jako vrácené v momentě, kdy je žák vrátí. Knihovník by měl taky moct přidávat knihy do databázi. Kromě toho, že knihovník může vyplnit sám všechna políčka, systém mu dovolí zadat jenom ISBN nebo název a vyplní automaticky zbytek. Knihovník bude moct i ukládat vlastní fotky u knížek, kde fotka není.

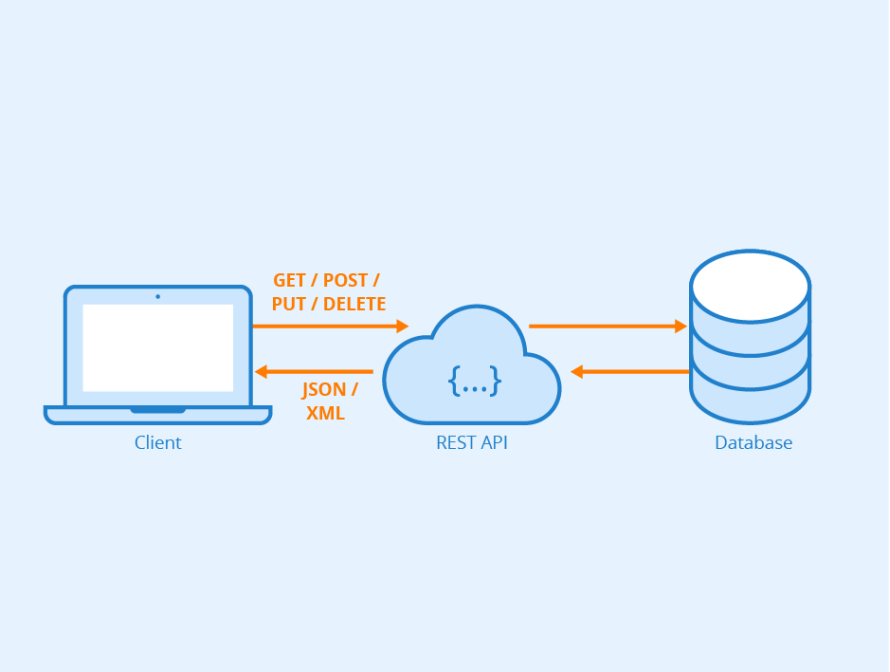
Obsah

1. Úvod

Většina studentů středních škol by se mnou jistě souhlasila, že navzdory výraznému technologickému pokroku naší doby, školní knihovna pořád hraje klíčovou roli v našem vzdělání. Jelikož téměř všichni mé spolužáci tráví svůj volný čas především „online“, myslím si, že můj projekt, tj. softwarová verze běžného knihovního systému, výrazně usnadní jak přístup ke knížkám na straně žáků, tak i samotná správa knihovny.

Projekt se skládá ze tří základní částí – front end (prezentační vrstva), back end (server webu) a databáze. Prezentační vrstva aplikace komunikuje se serverem pomocí tzv. Rest API, což je sbírka procedur obsahující všechny potřebné metody a příkazy. Front end (klient) posílá požadavky na server, jenž je zpracovává a na základě jejich obsahu manipuluje s databází jedním ze čtyř způsobů (CRUD): vytváří nové prvky (create), čte, upravuje nebo maže již existující záznamy (read, update, delete).

Jedním z nejdůležitějších aspektů každého softwaru je nesporně bezpečnost. V mém případě aplikace rozpoznává tři úrovně zabezpečení – „návštěvník“ (visitor), „student“ a „knihovník“ (teacher). Systém následně dynamicky přizpůsobuje povolené operace a vzhled a obsah stránky podle skupiny, do které klient patří. Detailní popis práv jednotlivých typů uživatelů v kapitole Components.



Obr. 1: Základní princip komunikace[1]

[1] Zdroj: https://www.seobility.net/en/wiki/images/f/f1/Rest-API.png

2. POUŽITÉ TECHNOLOGIE

Celá práce je napsaná v Javascriptu. Front end projektu běží na frameworku React a serverová část této webové aplikace je vytvořená v Expresu, což je framework pro vývojové prostředí NodeJS. Zvolil jsem NOSQL databázi MongoDB kvůli vynikající integraci s NodeJS. Systém byl vyvinut za pomoci verzovacího nástroje Git a webovou službu GitHub ve vývojovém prostředí (IDE) IntelliJ WebStorm. Podrobnější informace o každém nástroji jsou uvedené níže.

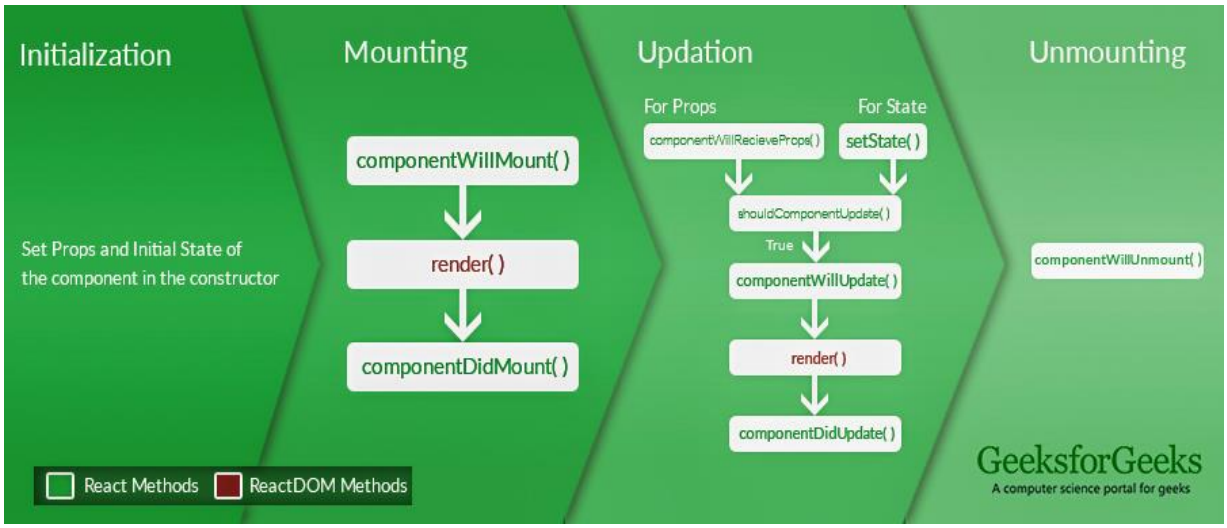
2.1 Front end

Prezentační vrstva projektu využívá nejmodernější technologie pro vývoj webových aplikaci – React a Redux.

2.1.1 React

React je framework, který kombinuje JavaScript a HTML do souboru *.jsx*. Je open-source a na vývoji se podílí i firma Facebook. Vybral jsem tuto knihovnu, jelikož je ideální pro vývoj SPA neboli *single page applications*. Základním stavebním blokem v *Reactu* jsou komponenty – *Components* reprezentující různé prvky webové stránky. Důležitou součástí komponent je takzvaný *state* (stav), do kterého se ukládají data, jež má komponenta zobrazovat. Dále mohou komponenty dostávat data od rodičovských prvků – tato data se nazývají *props*.

Komponenty mají vlastní životní cyklus. Můžeme ho v komponentách využívat, jelikož existuje několik funkcí, jež se spustí v patřičných fázích životního cyklu, např.: funkce componentDidMount se spustí, když se komponenta poprvé vyrenderuje na stránce, a funkce componentDidUpdate se spustí poté, co dojde ke změně dat v komponentě (komponenta dostala nové *props*, nebo byl změněn její *state*), kvůli čemuž je nutno ji znovu vyrenderovat.



Obr. 2:Životní cyklus React komponentu[2]

[2] Zdroj: https://media.geeksforgeeks.org/wp-content/uploads/lifecycle_reactjs.jpg

2.1.2 Redux

Hlavní myšlenkou *Reduxu* je vytvoření uložiště – *store* dat pro celou aplikaci, ke kterému se může každá komponenta připojit a zažádat si o patřičná data. Tato knihovna je potřebná kvůli způsobu fungování Reactu – protože defaultně framework umožňuje přenos dat jenom jedním směrem, používáme third-party nástroje, abychom mohli měnit, nastavovat a rušit globální proměnné.

Reduxův *store* se hodí k:

- Přihlášení uživatele a následné autentizaci
- Dynamickému spojení s Rest API webového serveru *(GET a POST požadavky)*

Toto globální uložiště se nastavuje pomocí takzvaných *Reducers.* V reducerů se konfigurují defaultní hodnoty proměnných a také se definuje chování uložiště pro požadavky GET / SET / DELETE. Popis jednotlivých proměnných a také jejich význam a účel najdete v kapitole Front end – Reducers.

2.2 Back end

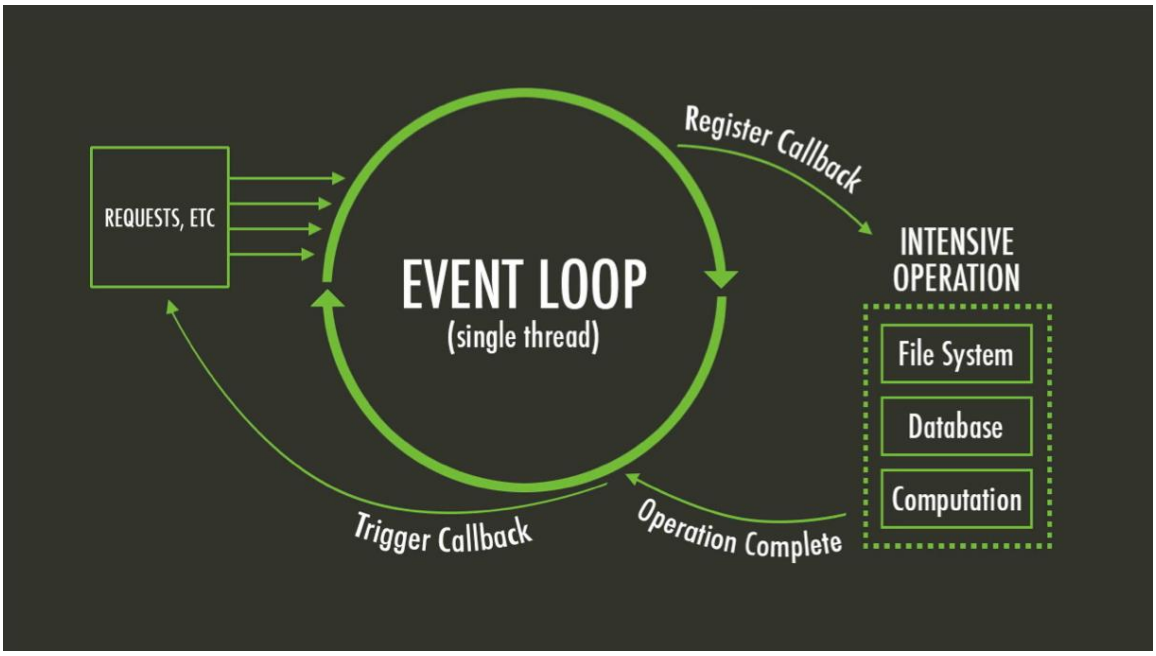
Server tohoto softwaru je napsán ve frameworku *Express* ve vývojovém prostředí *Node.js*. Zvolil jsem toto prostředí kvůli tomu, že je velmi rychlé, událostmi řízené, neblokující a asynchronní. Vybral jsem Express, neboť je minimalistický a skvěle doplňuje Node.js. MongoDB byla zvolena kvůli jednoduché integraci.

2.2.1 Node.js

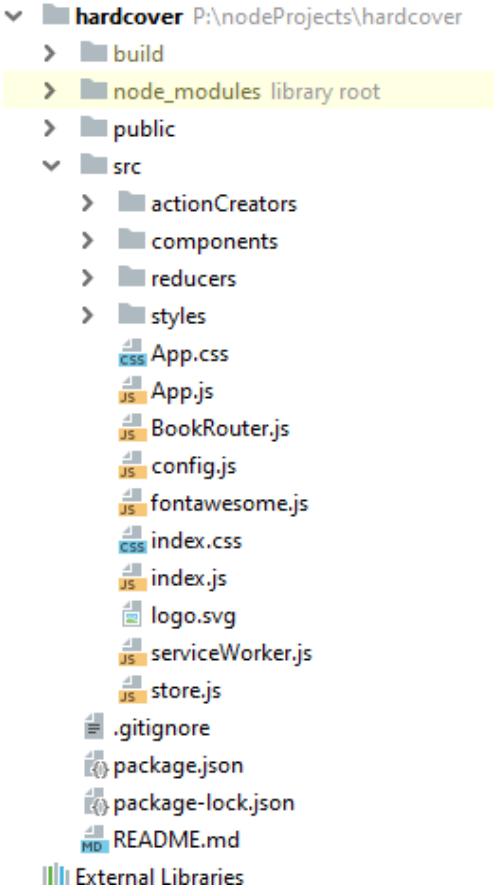
Node.js je prostředí umožňující spouštět JavaScript kód mimo webový prohlížeč. Je postaveno na Chrome V8 JavaScript enginu, takže základ tohoto JS prostředí je stejný jako ve webovém prohlížeči Google Chrome. JavaScript se tedy díky tomuto prostředí dá používat i na serveru a ne jen na druhém konci, u klienta. Avšak na rozdíl např. od PHP je v Node.js kladen důraz na vysokou škálovatelnost, tzn. schopnost obsloužit mnoho připojených klientů naráz. Pro tuto vlastnost a vysokou výkonnost je dnes Node.js velmi oblíbený pro tvorbu tzv. API serverů pro klientské single page aplikace rovněž v JavaScriptu. (Máca, 2018)

2.2.2 Express.js

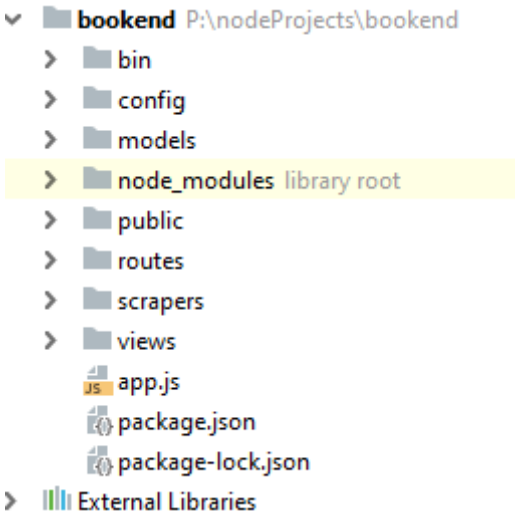
Express.js je nejpoužívanější framework pro Node, protože je velmi minimalisticky a flexibilní. Doplňuje a vylepšuje Node v místech, kde je potřeba, ale na rozdíl od Reactu nebo jiných podobných frameworku ponechává strukturu a styl psáni kódu stejnou. Vybral jsem ho právě z tohoto důvodu.



Obr. 5: Jednoduchý obrázek pro znázornění asynchronní způsob operaci Nodu[3]



Obr. 3: Struktura Front endu



Obr. 4 Struktura Back endu

[3] Zdroj: https://s3.amazonaws.com/oodles-technologies1/blog-images/bbae3887-8839-43e1-a07e-29c961f2a04f.png

2.3 Společné nástroje

Verzovácí systém a správce balíčků.

2.3.1 Git & GitHub

Git je verzovácí systém, pomocí něhož se snadno a rychle nacházejí bugy a problémy v kódu. Také značně ulehčuje kolaboraci a přenos programu do nového počítače. *GitHub* je webová služba využívající *Git*. Nabízí bezplatné uložiště projektu, pokud je program open-source.

Front end: https://github.com/ViktorKorladinov/hardcover

Back end: https://github.com/ViktorKorladinov/bookend

2.3.2 Node package manager

Node package manager neboli *npm* je správce balíčků – jednoduše instaluje vše, co projekt potřebuje ke spuštění. Chování *npm* je definováno v souboru *package.json*. Tento manager také spouští projekt.

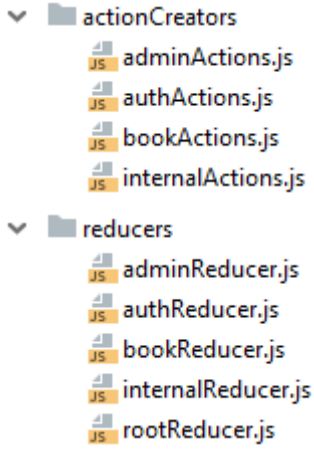
- `Příkaz pro nainstalování balíčků: npm install`
- `Příkaz pro spuštění aplikace: npm start`

3. FRONT END

App.js je hlavní komponentou aplikace a **index.js** komponentu rendruje. **BookRouter.js** je konfigurační soubor pro knihovnu *react-router*. Uvnitř jsou uvedené cesty a komponenty, které by se měly na dané stránce načíst. Například, na `/auth` se načte komponenta *Auth,* na `/dashboard` – komponenta *Dashboard* atd. *Navigation* a *Errors* se rendrují na každé stránce. V případě, že jsi uživatel zadal neplatnou adresu nebo stránka přestala existovat, ukáže se komponenta *PageNotFound*.

Dále **config.js** slouží k přepínání adresy serveru, podle toho, zda aplikace běží lokálně, nebo na serveru. Soubor **fontawesome.js** je „podknihovnou" knihovny fontawesome. Vytvořil jsem ji z optimizačních důvodů – program načítá jenom ikony, jež používá, a tak výrazně snižuje objem dat, která se musí stáhnout. **App.css** a **index.css** jsou kaskádové styly. Zbytek kaskádových stylů se nachází ve složce **styles**. **store.js** obsahuje konfiguraci pro Reduxův *store*.

3.1 Action Creators & Reducers



Obr. 6: actionCreators a reducers

Action Creators jsou takzvané „továrny". V nich jsou různé „akce", neboli metody, které se spouští, když je některá komponenta zavolá. Většina těchto metod slouží k interakci s API serverem. Informace, které pak dostávají, posílají do reducerů, kde se zpracují a poté uloží do globálního uložiště *store*. Jak **actionCreators,** tak i **reducers** jsou součástí ekosystému knihovy *react-redux*.

adminActions.js obsahuje všechna API volání, která vyžadují administratorská práva (knihovnika).

authActions.js má na starost autentizaci – registrace, přihlášení, registrace/přihlášení pomocí Googlu.

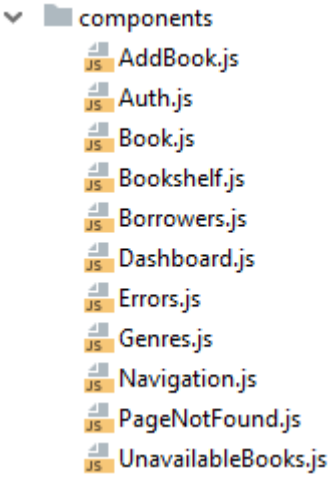
bookActions.js obsahuje metody pro manipulaci s knížkami[4] – rezervace, hledání, přidávání knížek atd.

internalActions.js je soubor všech metod, které aplikace potřebuje používat na více místech – proto jsou uložené v globálném uložišti.

[4] Poznámka – samotné propůjčení knížek se nachází v **adminActions.js**, protože na to je potřeba přítomnost a práva knihovníka.

Reducers mají stejná jména jako „továrny“, za které jsou zodpovědné. Reducery přebírají objekty, jež akce dostaly od serveru ve formatu JSON, a zpracovávají je do podoby, do které je chtějí do *store* uložit.

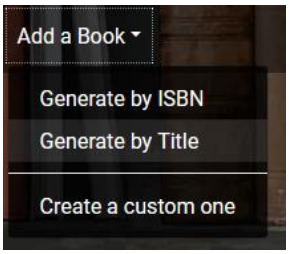
3.2 Components



Obr. 8: components

AddBook.js má tři režimy:

- Uložit knihu podle ISBN
- Najít a uložit knihu podle názvu
- Sepsat vlastnoručně celou informaci o knížce a následně ji uložit.



Obr. 7: AddBook možnosti

Auth.js - přihlášení a registrace uživatelů

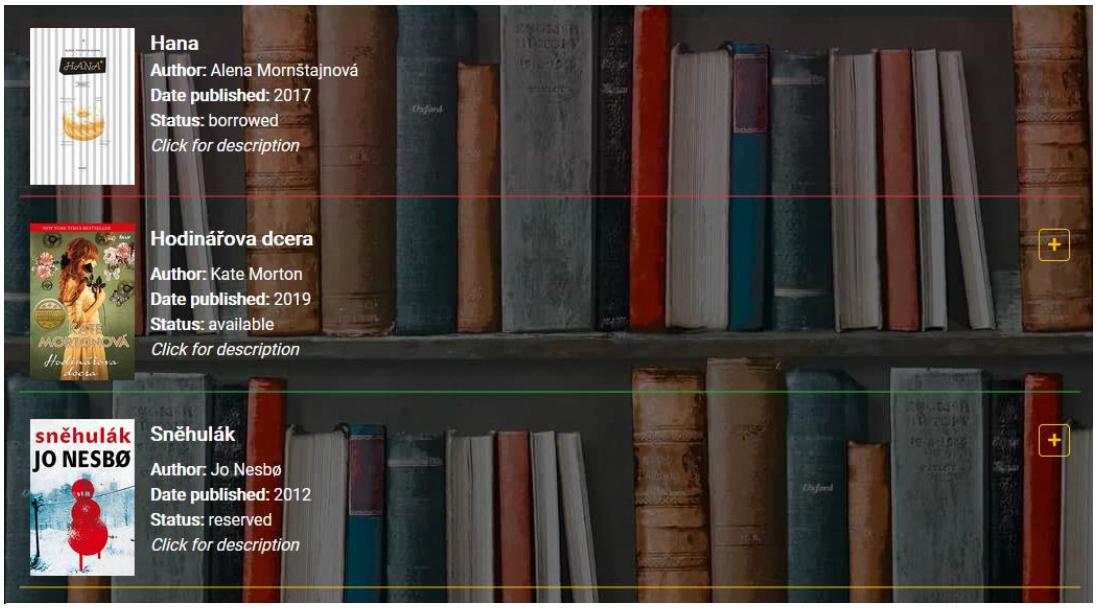
Token uživatele se uchovává v local storage browseru klienta a také v globálním uložišti. Obsah stránek a povolených operací se dynamicky mění podle toho, zda klient je přihlášen. Dále obsah se mění i podle úrovně uživatele – knihovník má přístup k zakázaným pro běžné uživatele komponentám.

Book.js

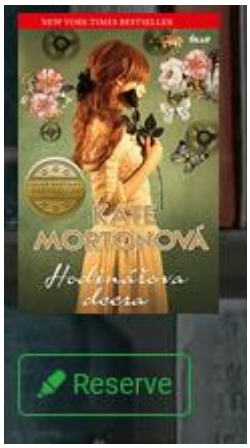
Komponenta, jež reprezentuje jednu knihu. Kniha může být zarezervovaná (reserved), volná (available), nebo půjčena (borrowed). Komponenta ma různé zabarvení, vzhledem k jejímu stavu: zelená – volná, žlutá – zarezervovaná, červená – půjčena (Obr. 9). Dále **Book.js** kontroluje profil klienta a mění počet dovolených operací vzhledem k jejich bezpečnostnímu oprávnění.

Operace/ Oprávnění	**Nepřihlašený uživatel**	**Student**	**Knihovník**
Číst (Vidět knížky)			
Rezervovat			
Půjčovat			

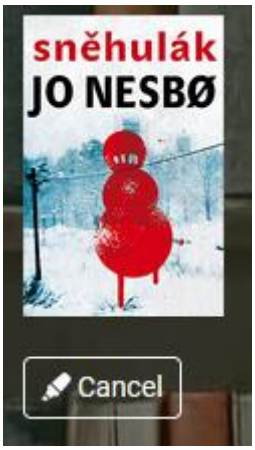
Pokud je knížka zarezervovaná, zrušit ji může knihovník nebo uživatel, který rezervaci provedl. Pokud byla někomu půjčena, jen knihovník ji může označit jako vracenou. Možnost knížku zarezervovat, popřípadě zrušit se ukazují po kliknutím *„Click for description“*(Obr. 11, Obr. 10).



Obr. 9: Půjčena vs. volná vs. zarezervovaná knížka



Obr. 11: Rezervace



Obr. 10:Zrušení rezervace

Bookshelf.js

Komponenta, která obaluje seznam knížek, vyhledávač, a filtry. Filtry mění pořadí knížek na přání klienta. Umí řadit podle jména autora, titulu, nebo data přidání. První kliknutí řadí odshora dolů, pak každé další kliknutí mění směr. Defaultní řazení je podle názvu.

Dashboard.js, Genres.js Borrowers.js & UnavailableBooks.js

UnavailableBooks.js je komponenta, která se ukazuje jen knihovníkům. Ukazuje všechny knížky, které jsou půjčene nebo zarezervované a umožňuje jednotlivé knížky označit jako vracené nebo rezervaci zrušit. **Borrowers** ukazuje seznam uživatelů, kteří mají půjčenou nebo zarezervovanou knížku. **Dashboard** je obálkou těchto dvou komponent + **Bookshlef.js** a **Genres.js**, což je komponenta, jež umí ukazovat knížky jen určitého žánru.

4. BACK END

Soubory **app.js** a **/bin/www** mají několik důležitých funkcí:

- Konfiguruje Express.js (**app.js**)
- Navazuje spojení s databází (**app.js**)
- Konfiguruje port, na kterém běží (**www**) a také cesty API (**app.js**) – například `/admin/...` je cesta pro všechna volání, která vyžadují administrátorské oprávnění. Podrobný popis API níže.
- Definuje chování serveru při vzniklé chybě (oba)

4.1 Config

Složka s konfiguračními soubory.

config.js:

```
module.exports = {
    studentLendingDays: 30,
    teacherLendingDays: 60,
    renewDays: 30,
    borrowDays: 3,
    borrowLimit: 4,
};
```

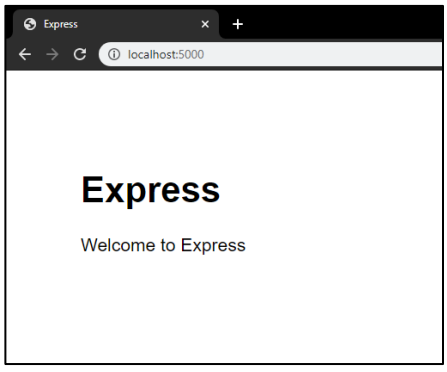
První řádek nastavuje půjčovní dobu pro žáka, druhý pro učitele. Třetí definuje dobu prodloužení. Předposlední řádek určuje délku rezervace a poslední – maximální počet rezervovaných knížek najednou na uživatele.

keys.js: Obsahuje různé adresy, jako například adresu vzdálené databáze.

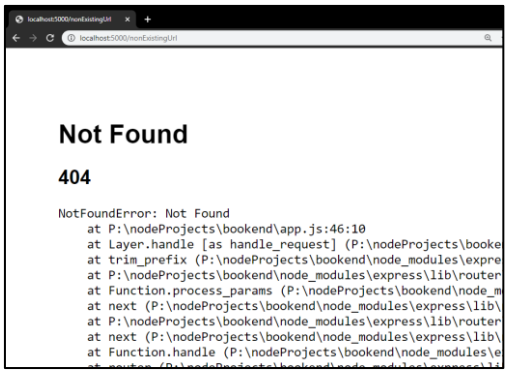
passport.js: Konfigurační soubor pro knihovnu *passport.js*. Pomocí tuto knihovnu autentizuji uživatele.

4.2 Views

Pomocí knihovny *mustache* vytváříme šablony – HTML stránky, které dostávají informace z jejich řadičů – v našem případě řadičem je *app.js*, který posílá chyby, pokud se nějaké vyskytly (a také název projektu.). **index.hbs** je hlavní stránka, **error.hbs** ukazuje chyby serveru v browseru, a **layout.hbs** konfiguruje rozložení těchto stránek.



Obr. 12: index.hbs



Obr. 13: error.hbs (404)

4.3 Scrapers

Tato složka obsahuje jeden soubor – **bookWorm.js**, který používá knihovnu *cheerio*, aby parsoval HTML databází knih (https://www.databazeknih.cz/). Tento *scraper* voláme vždy, když knihovník chce uložit novou knížku, pomocí názvu nebo ISBN. Náš *scraper* umí parsovat dva typy stránek: stránka knížky a stránka s výsledky vyhledávání.

Stránka s vyhledáváním je potřeba z jednoho prostého důvodu: v případě, že administrátor chce přidat knihu podle názvu, aplikace nabídne vše, co v databázi webu je, místo jen prvního odkazu. Z tohoto důvodu přidávání knihy podle názvu je mnohem složitější operace než podle ISBN.

První krok je POST požadavek klienta, kde se nachází název nebo část názvu knihy. Dále v dotazovací metodě je i počet výsledků, které chceme dostat. Defaultní hodnota je 5. Minimum je 1, maximum není – omezení vytváří počet knih v databázi webu. Server dostane data, zpracuje je a vyšle *scraper*, aby získal tolik výsledků ze stránky databazeknih.cz, kolik klient požadoval. V odpovědi (*response*) pak zasílá front endu knihy, které našel. UI dostane knihy a jejich obálky a vybídne uživatele, aby tu správnou vybral. Poté, co si knihovník vybere knížku, front end opět posílá POST *request* serveru, už s přesnou adresou knihy, kterou chce do databáze přidat. Back end vyšle (volá *parseBook()* - o této funkci níže.) *scraper* parsovat znovu – tentokrát stránku určité knížky.

Uložení knihy podle ISBN je jednodušší – *scraper* zadá číslo do vyhledavače stránky a pak následuje několik redirektů, které ho nakonec dostanou do stránky knihy se stejným ISBN. Poté volá *parseBook()*.

parseBook() očekává jediný parametr – adresu, kterou má parsovat. Ukládá dva typy dat – veškerou informaci o knížce v objektu **Book** (model objektu je popsaný v sekci Databáze.) a také obálku. Obálku uloží nejdřív lokálně, poté uloží na serveru Imgur pomocí jejich API a když se ujistí, že se fotka uložila, smaže ji z lokálního uložiště. Pak hotlink fotky přidá do objektu **ook** jako řetězec.

4.4 API

Celá API dokumentace se nachází na adrese:
https://documenter.getpostman.com/view/5103979/SzS7RmLv?version=latest.

Databazeknih.cz ...Váš knižní svět

Přihlásit | Vytvořit profil

Novinky | Knihy | Bazar | Autoři | Žebříčky | Komunita • 645 | Diskuze

Vyhledávání

Pokročilé hledání

1984

Knihy | Autoři | Bazar | Uživatelé | Povídky | Diskuze | Ostatní

- 1984
  2009, G. Orwell (pseudonym)
- Kronika zániku Evropy 1984-2054
  2019, V. Vondruška
- Marzi 1984-1987
  2012, M. Sowa
- Veselé příběhy čtyřlístku - 1982-1984
  2007, L. Stiplová
- Úžasné příběhy Čtyřlístku - 1984-1987
  2008, L. Stiplová
- Hovory lidí (1984)
  1983, B. Hrabal
- Když klec je pořád na spadnutí. Vzájemná korespondence Antonína Přidala a Jana Zábrany z let 1963–1984
  2018, J. Zábrana

Databazeknih.cz ...Váš knižní svět

Přihlásit | Vytvořit profil

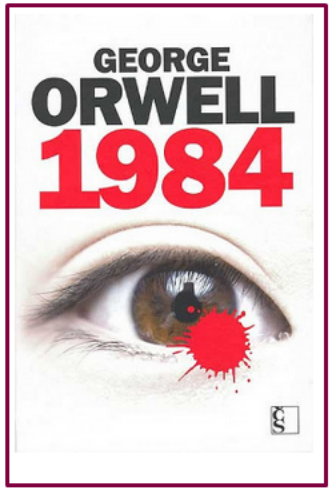
Novinky | Knihy | Bazar | Autoři | Žebříčky | Komunita • 635 | Diskuze

1984

kniha od George Orwell (p)

92 % | 8 527 hodnocení

Přidat do mých knih



Koupit | Koupit eknihu

1984, dnes již klasické dílo antiutopického žánru, je bezesporu jedním z nejpozoruhodnějších románů 20. století a brilantní analýzou všech totalitních systémů. "Anti-utopie" má i svého "antihrdinu": je jím Winston Smith, obyčejný, ne příliš statečný člověk, který si nepřeje nic světoborného: chce jen poznat skutečnou, ne zm...celý text

Žánr	Literatura světová, Romány, Sci-fi
Vydáno	2009, Levné knihy
Orig. název	Nineteen Eighty-Four, 1949
Počet stran	262
Jazyk vydání	český
Překlad	Eva Šimečková
Edice	Československý spisovatel
Autor obálky	Milan Jaroš
Vazba knihy	pevná / vázaná
ISBN	978-80-7309-808-7

4.5 Databáze

Databáze je MongoDB. Z optimalizačních důvodů jsem zvolil použít externí server pro databázi – Atlas MongoDB. Pro mé účely je tato služba ideální a poměrně rychlá. Mongo a Node komunikují pomocí knihovny *mongoose*. Tato knihovna umí vytvářet z JavaScript objektů Mongo dokumenty a z dokumentů objekty pomocí takzvaných schémat. Má aplikace má celkově 3 schémata neboli modelů – **Book** (kniha), **Genre** (žánr) a **User** (uživatel). Každý model je ve vlastním souboru a vždy popisuje jaká data budeme do objektu ukládat a také typ jednotlivých dat.

```
const GenreSchema = new mongoose.Schema({
    name: {
        type: String,
        required: true
    },
    amount: {
        type: Number,
        default: 1,
    },
});
```

4.5.1 Genre

Jeden dokument je jeden žánr. Má dvě povinná pole: název (řetězec) a množství. Množství je číslo a při vytvořením dokumentu se automaticky nastavuje na jedničku. Tyto dokumenty jsou potřeba, aby front end mohl vypsat seznam všech žánrů. Po přidání nové knihy se zkontrolují její žánry a buď se vytvoří nový dokument, pokud tento žánr v systému není, nebo se zvětší číslo, označující množství o jedna, v případě, že žánr už v bázi je.

4.5.2 User

Každý uživatel je reprezentován dokumentem User. První pole (**firstName**) je povinné a označuje křestní jméno studenta, či knihovníka. Dále objekt obsahuje povinný řetězec **surname**, což reprezentuje příjmení. Email klienta se uchovává v povinném poli **email**. Poslední dva řetězce jsou **password** a **pic**. První *string* – **password** uchovává zašifrovanou verzi hesla. Pole není povinné, jelikož heslo se neukládá v případě, že se uživatel přihlásil pomocí služby Google Auth. Poslední řetězec **pic** je *hotlink* pro avatara. Defaultní obrázek je nastaven v případě, že klient nenastavil vlastní. Při přihlášení pomocí Googlu aplikace avatar vezme z Google účtu uživatele. **roles** definuje oprávnění. Jsou dvě možnosti – *student* a *teacher*, přičemž *teacher* má více práv. **reserved** obsahuje knihy, které uživatel zarezervoval a **borrowed** – knihy, které si půjčil.

```
const UserSchema = new mongoose.Schema({
    firstName: {
        type: String,
        required: true
    },
    surname: {
        type: String,
        required: true
    },
    email: {
        type: String,
        required: true
    },
    password: {
        type: String,
        required: false
    },
    date: {
        type: Date,
        default: Date.now
    },
    roles: {
        type: Array,
        default: ['student']
    },
    reserved: {
        type: Array,
        default: [],
    },
    borrowed: {
        type: Array,
        default: [],
    },
    pic: {
        type: String,
        default:
'https://secure.gravatar.com/avatar/a9c4b6
    }
```

4.5.3 Book

```
const BookSchema = new mongoose.Schema({
    title: {
        type: String,
        required: true
    },
    author: {
        type: String,
        required: true
    },
    desc: {
        type: String,
    },
    genres: {
        type: Array,
        required: true
    },
    dateAdded: {
        type: Date,
        default: Date.now
    },
    datePublished: {
        type: String,
    },

    numberOfPages: {
        type: String,
    },
    ISBN: {
        type: String,
        required: true,
    },
    image_link: {
        type: String,
    },
    state: {
        type: String,
        default: 'available'
    },
    lastAccessed: {
        type: Date,
    },
    returnDate: {
        type: Date,
    },
    borrower: {
        type: String,
    },
    borrowerId: {
        type: String,
    },
}
```

První je název knihy. Je to řetězec a je povinný. Mongoose ohlásí chybu, pokusíme-li se uložit dokument s prázdným polem **title.**

Další pole je jméno a příjmení autora, o něm platí to samé, co platí o názvu.

Třetí v řadě je popis knihy, což je nepovinné, jelikož některé knihy nemají popis.

Genres je povinné pole prvků, kde jeden prvek je jeden žánr. Tato *array* slouží k filtrování knih podle žánrů.

datePublished je rok vydání knihy a **numberOfPages** počet stran.

dateAdded je pole, které se automaticky nastavuje. Označuje čas přidání knihy do databáze a je typu *Date*.

ISBN je ISBN knihy. Je povinen, jelikož hraje roli při ukládání obalu knihy.

image_link je *hotlink* obalu, podrobnější informace najdete v kapitole Scrapers.

state označuje jeden ze čtyř stavů knihy. Při přidání se automaticky nastaví jako volná.

lastAccessed je kdy byla naposledy půjčena a **returnDate** označuje termín vracení. Obě pole jsou typu *Date*.

borrower je jméno uživatele, jenž si knížku půjčil a **borrowerId** jeho id.

Kromě těchto parametrů máme ještě jeden virtuální – **status**. Ten udává, zda knížka je například opravdu rezervovaná, nebo rezervace vypršela, aniž by si ji uživatel půjčil – v takovém případě *state* knížky by byl *reserved*, ale status – available. Také přidává čtvrtý typ stavu – overdue neboli zpožděné vrácení.

5. ZÁVĚR

Dle mého názoru se mi povedlo realizovat původní nápad, neboť výsledek odpovídá, ba i přesahuje mé představy ze začátku školního roku. Splnil jsem všechny cíle, které jsem si stanovil, a v daném časovém rámci jsem rovněž stihl doplnit projekt o některé „extry“, jako například nastavení avatarů uživatelů nebo přidání vlastních obalů/obrázků knížek. Z optimalizačních důvodů jsem se v tomto roce také rozhodl použít externí servery pro databázi a uložiště obrázků.

Snažil jsem se navrhnout a strukturovat projekt takovým způsobem, který umožňuje snadné přidání či upravování nabízených funkcí. Tato zkušenost byla pro mě velmi užitečná a přínosná v kontextu mého budoucího povolání.

7. POUŽITÁ LITERATURA

1. **Máca, Jindřich.** Úvod do Node.js. *ITnetwork.* [Online] 18. 4 2018. https://www.itnetwork.cz/javascript/nodejs/uvod-do-nodejs.

2. **Abramov, Dan.** Async actions. *Redux.* [Online] Redux, 12. 4 2016. [Citace: 2. 3 2020.] https://redux.js.org/advanced/async-actions.

3. **Stover, Charles.** Optimal file structure for React applications. *Medium.* [Online] 5. 3 2019. [Citace: 2019. 12 21.] https://medium.com/@Charles_Stover/optimal-file-structure-for-react-applications-f3e35ad0a145.

4. **Karnik, Nick.** Introduction to Mongoose for MongoDB. *Free Code Camp.* [Online] 11. 2 2018. [Citace: 15. 12 2019.] https://www.freecodecamp.org/news/introduction-to-mongoose-for-mongodb-d2a7aa593c57/.

5. **Gonzalo, Pablo.** Structure of a NodeJS API Project. *Medium.* [Online] Medium, 27. 3 2018. [Citace: 24. 11 2019.] https://medium.com/codebase/structure-of-a-nodejs-api-project-cdecb46ef3f8.

8. SEZNAM OBRÁZKŮ A TABULEK