FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání bakalářské práce

Název:	Návrh a implementace frontend pro projekt Česká elektronická knihovna
Student:	Filip Hladej
Vedoucí:	Ing. Michal Valenta, Ph.D.
Studijní program:	Informatika
Obor / specializace:	Webové a softwarové inženýrství, zaměření Webové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	do konce letního semestru 2021/2022

Pokyny pro vypracování

Ústavu pro českou literaturu (UČL) se podařilo digitalizovat cca 1700 sbírek obsahujících přes 80 000 básní. Obsahy sbírek jsou uloženy v XML souborech a doplněny i nascanovanými stránkami daných fyzických knih.
Cílem této práce je analyzovat, navrhnout a vytvořit frontend (uživatelské rozhraní), který umožní zejména zpřístupnění a opravy dat. Postupujte v těchto krocích:
• Seznamte se se strukturou dat.

• Shromážděte požadavky na frontend projektu.

• Ve spolupráci s autorem souběžné diplomové práce zaměřené na backend tohoto projektu navrhněte strukturu API zejména pro účely prezentace dat a jejich opravy.

• Proveďte průzkum vhodných technologií a zvolte technologii pro realizaci uživatelského rozhraní. Bude se jednat o webovou aplikaci s důrazem na možnost práce i z mobilního zařízení.

• Navrhněte, implementujte, otestujte a zdokumentujte uživatelské rozhraní (frontend) pro práci s daty.

Elektronicky schválil/a Ing. Jaroslav Kuchař, Ph.D. dne 29. ledna 2021 v Praze.

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

Návrh a implementace frontendu pro projekt Česká elektronická knihovna

Filip Hladej

Katedra softwarového inženýrství
Vedoucí práce: Ing. Michal Valenta, Ph.D.

12. května 2021

Poděkování

Nejprve bych chtěl poděkovat vedoucímu mé bakalářské práce Ing. Michal Valenta, Ph.D. za možnost pracovat na reálném projektu a také za podporu, které se mi během psaní práce dostalo. Dále bych chtěl poděkovat Tomášovi Chvostovi za spolupráci na projektu.

Velké díky také patří mým skvělým přátelům, kteří mě podporují v těžkých chvílích studia a dokáží mě pokaždé vyhrotit k podání lepších výsledků.

Na závěr bych chtěl poděkovat své rodině, která mě po celou dobu podporovala.

Prohlášení

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

V Praze dne 12. května 2021 . . . . . .

Odkaz na tuto práci

Hladej, Filip. *Návrh a implementace frontendu pro projekt Česká elektronická knihovna*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2021.

Abstrakt

Tato práce se zabývá vytvořením klientské části webové aplikace pro projekt Česká elektronická knihovna Akademie věd České republiky. Jedná se o aplikaci pro správu sbírek básní, která by měla sloužit jako inovace stávajícího řešení. Práce nejprve analyzuje a srovnává vhodné technologie a přístupy v tvorbě webových aplikací. Následně se věnuje vytváření požadavků na systém pokrývající veškeré funkcionality původního řešení. Součástí je také návrh uživatelského rozhraní a jeho následná implementace. Uživatelská část je v práci otestována a doplněna o dokumentaci.

Klíčová slova elektronická knihovna, knihovna básní, responzivní webová aplikace, React, Material UI, frontend, JavaScript

Abstract

This work deals with the creation proccess of the client side of a web application for Czech electronic library project created by the Academy of Sciences of the Czech Republic. The primary use of the application is poetry anthology management intended as an innovation to the existing solution. An analysis and comparison of suitable technologies and approaches of web development was performed and based on its results, a set of requirements covering all functionalities was created. Based on its results, a user interface was designed and it was implemented into a working solution. The work includes proper testing and is supplied with sufficient documentation.

Keywords electronic library, library of poems, responsive web application, React, Material UI, frontend, JavaScript

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Používání webových aplikací se stalo běžnou součástí našich životů. Silný vliv má samozřejmě neustále rostoucí technologický vývoj a s tím spojené využívání nových postupů či přístupů ve všech odvětvích lidského života. Aplikace za nás v dnešní době dělají spoustu práce, usnadňují nám život a tím se k nám dostávají blíže.

Spousta uměleckých děl, které mají či mohly mít nevyčíslitelnou hodnotu i v dnešním světě, není kompletní, zachovaly se pouze jisté části a celkový odkaz těchto děl je tak znehodnocen. Z historie se ovšem vždy dá poučit a vytvořit tak nové možnosti pro budoucí generace. Ústavu pro českou literaturu Akademie věd se podařilo digitalizovat velké množství sbírek básní. Nová umělecká díla v dnešní době stále vznikají a jejich celkový počet rychle roste. Zmíněné digitalizované sbírky, dále jen data, takového rozsahu je mnohem snazší spravovat a uchovávat pomocí technologií dnešní doby než jen v jejich originální fyzické podobě.

Ústav pro českou literaturu Akademie věd přišel s projektem Česká elektronická knihovna, jehož cílem je vytvoření aplikace, která by umožnila tato data prezentovat, spravovat či opravovat. V praxi se dá webová aplikace rozdělit na dvě části, které spolu úzce spolupracují, server a klient. První částí této webové aplikace je tvorba rozhraní umožňující administraci a zpracování dat, zmíněná serverová část (backend). Této části se věnuje autor souběžné diplomové práce Bc. Tomáš Chvosta. Předkládaná bakalářská práce si dává za cíl realizaci uživatelské části (frontendu), pro umožnění komunikace uživatele a knihovního systému.

Motivací pro tvorbu této práce bylo nahlédnutí za oponu tvorby webové aplikace, od návrhu, přes implementaci až k samotnému testování. Při výběru tématu hrálo důležitou roli vědomí toho, že se potýká s reálným projektem.

Rešeršní část se zaobírá analýzou zvolených technologií a rozebrání důvodů, proč byly tyto technologie upřednostněny. Následuje popis komplexního návrhu uživatelského rozhraní a jeho vliv na tvorbu zbylých částí práce. Související implementační část popisuje postup tvorby frontendu včetně propojení se serverovým řešením. Nedílnou součástí práce je také vytvoření testů, které si dávají za cíl ověřit funkčnost implementace a vyhledat případné chyby, které během ní mohly nastat. V závěrečné kapitole je popsána dokumentace, která slouží jako průvodce aplikací.

Výsledek práce bude využíván Ústavem pro českou literaturu Akademie věd a je možné, že i samotnými autory uměleckých děl. Projekt se má časem zdokonalovat, naše práce jsou inicializačními kroky, tedy je zde reálný předpoklad na tuto práci v budoucnu navázat.

Cíl práce

Hlavním cílem této práce je analýza, návrh a implementace uživatelské části webové aplikace v rámci projektu Česká elektronická knihovna pro Ústav pro českou literaturu Akademie věd České republiky. Za tímto účelem bude v práci nutné nastudovat, popsat a porovnat vhodné přístupy pro tvorbu uživatelského rozhraní. Stěžejní je také možnost využívat aplikaci z mobilního zařízení, tedy je kladen požadavek na responzivitu. Podstatnou částí práce bude realizovat komunikaci s rozhraním backendu a finálně tak uvést do provozu plně funkční aplikaci umožňující správu uměleckých děl. Závěrem bude nezbytné vytvořit testy a dokumentaci.

Kapitola 1

Rešerše

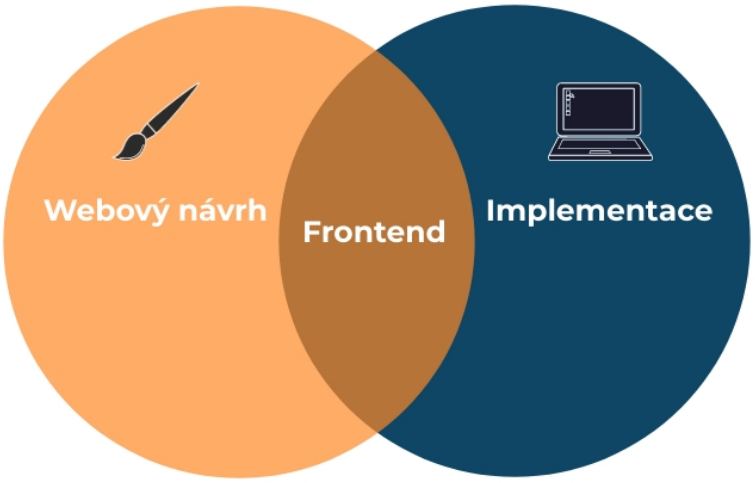
Rešerše je rozdělena na dvě části. První část se zabývá představením technologií, které budou využity při implementaci uživatelského rozhraní a jejich srovnáním s konkurencí. Druhá část se zabývá technologiemi, které ve své práci využije autor backendu a následně budou úzce spjaty s touto bakalářskou prací.

1.1 Frontend

Jako frontend je označována ta část webu, která je viditelná pro uživatele. Je na ni kladen velký důraz, co se týče propracování z hlediska přístupnosti, použitelnosti a vzhledu. Zahrnuta je i celková grafická podoba webu, různé formuláře, ikony, efekty a tedy by měla návštěvníka na první pohled zaujmout. Uživatelská část by také měla být navrhnuta tak, aby byla lehce zapamatovatelná, umožňovala snadnou orientaci a byla intuitivní.

Tvorba frontendu se skládá ze dvou částí. Nejprve je třeba navrhnout webový design nebo také návrh webu. Nejedná se pouze o grafický návrh webu [1], jak se mylně spousta lidí domnívá, ale také o analýzu požadavků, stanovení cílů a popis toho, jak by měl celý web fungovat. Tuto část má na starosti návrhář. Druhou částí je implementace, což je zase práce programátora. Jeho cílem je navázat na návrh a vytvořit dle jeho specifikací odpovídající uživatelské rozhraní. Na obrázku 1.1 lze vidět popis vztahu návrhu a implementace v kontextu vývoje frontendu.

Dle [3] se vývoj frontendu odvíjí od zkušeností programátora s jazyky HTML, CSS a JavaScript. Tyto tři jazyky jsou hlavní součástí tvorby uživatelské části webových aplikací. Zatímco HTML definuje strukturu webové stránky a její obsah pomocí řady elementů, které má k dispozici, CSS slouží k definování způsobu zobrazení těchto elementů a tedy zastává funkci designu. Pro pokročilejší funkcionality je třeba sáhnout po jiných programovacích jazycích



Obrázek 1.1: Frontend [2]

a knihovnách. V dnešní době je velice populární použití jazyka JavaScript, který je popsaný v sekci 1.1.4.

1.1.1 Uživatelské rozhraní

Uživatelské rozhraní (ang. User Interface, UI) je systém, který umožňuje uživateli komunikovat s počítačem. Jeho primárním cílem je, aby tato komunikaci byla intuitivní. Je lidem známý, neboť se jedná kupříkladu o domovskou obrazovku mobilního telefonu, příkazový řádek nebo webovou aplikaci. Je také snadno zapamatovatelný, neboť jej lze vidět, slyšet nebo si na něj lze i sáhnout.

Dle [4] je uživatelské rozhraní vůbec ta nejdůležitější část jakéhokoliv počítače. Lidé mají neustále rostoucí nároky na kvalitní graficky propracované rozhraní. Technologický posun v posledních letech dvacátého století v oblasti UI ukázal, že vývojáři na takové požadavky slyší. To vedlo ke vzniku grafického uživatelského rozhraní (ang. Graphical User Interface, GUI).

GUI [5] je UI, které umožňuje komunikovat uživateli s počítačem pomocí řady intuitivních symbolů a vizuálních vjemů. Slouží jako náhrada za textová rozhraní, které byla pro většinu uživatelů příliš nejasná.

1.1.2 Webový framework

Webový framework [6] je softwarová struktura, která obvykle nabízí sadu nástrojů, které pomáhají při vývoji webové aplikace. Může se jednat kupříkladu o ulehčení práce se směrováním (routováním), databázovým připojením, sty-

lováním, bezpečnostními prvky nebo také zpracováváním HTTP požadavků a odpovědí. Při výběru je potřeba vzít v potaz několik aspektů. Jedním z nich může být absence předchozí zkušenosti vývojářů s daným frameworkem, kdy získání potřebných znalostí bude tvořit významnou část celého životního cyklu projektu. Důležitá je také zkušenost s jazykem samotným, jinak získání těchto znalostí může být velmi obtížné a kontraproduktivní.

Některé webové frameworky nabízí lepší podporu ohledně bezpečnosti proti základním kybernetickým útokům než jiné, a jsou i takové, které ji sice nabízí, ale ne ve výchozí konfiguraci.

Je důležité zmínit rozdíl mezi frameworkem a knihovnou [7], neboť velmi často dochází k záměně těchto pojmů. Knihovny nabízí určité možnosti, jak využít již hotový kód, tedy stejně jako framework pomáhají při vývoji, ovšem použití tohoto kódu je nepovinné. U frameworku je tomu přesně naopak, jedná se o rámec nad daným jazykem. Určuje tedy strukturu celého projektu a nelze se rozhodnout, že v jisté části bude jednoduše vynechán.

1.1.3 Srovnání frameworků

Mezi nejpopulárnější a nejpoužívanější JavaScriptové frameworky [8] patří AngularJS, VueJS a ReactJS.

1.1.3.1 AngularJS

AngularJS [9] je rozšiřitelný, zaměřený na tvorbu single-page aplikací. Funguje na principu datové vazby (ang. Data Binding), tedy automaticky aktualizuje uživatelské rozhraní, kdykoliv dojde ke změně. Tímto způsobem se zbavuje nutnosti často nepříjemné manipulace s objektovým modelem dokumentu. Stejně jako Vue a React umožňuje vzít webovou stránku a rozdělit ji na znovupoužitelné komponenty. Dalším společným znakem těchto tří frameworků je bezproblémová podpora knihoven. Ovšem na rozdíl od Reactu a Vue jsou AngularJS modely napsány v obyčejném JavaScriptu, tedy údržba a testování jsou značně jednodušší.

1.1.3.2 VueJS

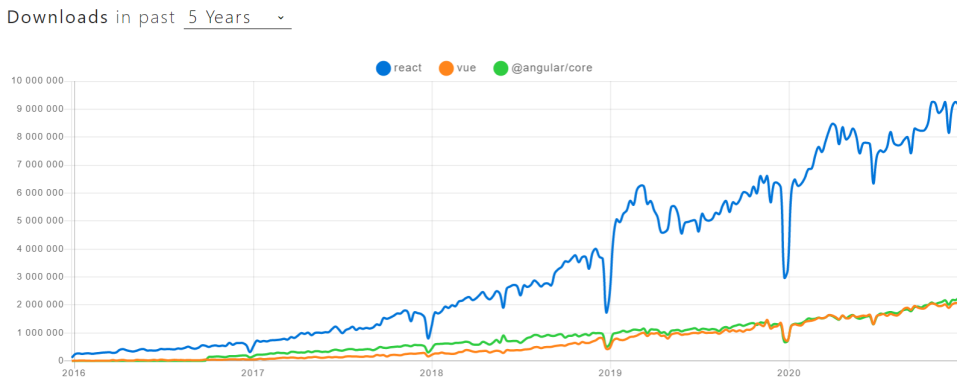
VueJS [10] je progresivní, reaktivní, má vlastní podporované knihovny a balíčky, které zjednodušují tvorbu pokročilých funkcí, jako je například routování a sestavování. Také se používá pro tvorbu single-page aplikací. Stejně jako React využívá virtuální objektový model dokumentu. Syntaxe je zase podobná Angularu, ovšem Vue je pružnější a oproti Angularu jednodušší na naučení.

V rozhovoru [11] pro Between the Wires tvůrce VueJS, Evan You, který pracoval pro Google, ocenil jako tehdejší programátor AngularJS principy, na kterých

Angular fungoval, jako byly datové vazby nebo způsob, jakým se vypořádává s objektovým modelem dokumentu. V rozhovoru také zmínil, že chtěl extrahovat pouze jisté části, které se mu na Angularu líbily a vytvořit tak něco méně robustního, co ovšem pokryje všechny potřebné funkcionality.

1.1.3.3 ReactJS

Jedná se o nejpopulárnější frontendový framework. Jeví se jako ideální JavaScriptový framework pro začínajícího frontend vývojáře [12]. Má velmi propracovanou dokumentaci a je nejpoužívanějším (viz obrázek 1.2) ze všech zmíněných. Více se mu bude věnovat kapitola 1.1.5. Po provedené analýze a následné konzultaci s mým vedoucím a autorem backendu, byl React zvolen jako vhodná technologie pro tuto bakalářskou práci.



Obrázek 1.2: Poměr počtu stažení jednotlivých frameworků skrz NPM [8]

1.1.4 JavaScript

JavaScript (JS) [13] je dynamický, netypovaný, vysokoúrovňový, událostmi řízený skriptovací jazyk, vhodný převážně k funkcionálnímu, ale také k objektovému programování.

Využití lze najít jak na straně serveru, tak na straně klienta. Tento jazyk umožňuje vytvářet dynamicky se měnící obsah webových stránek, tedy zobrazovat různé animace, přehrávat audio či video záznamy a spoustu dalších interaktivních prvků. [14]

JS je závislý na prohlížeči, neboť v různých verzích prohlížečů může být jeho chování odlišné. Uživatel má také možnost jej vypnout, čímž může docílit větší rychlosti či většího zabezpečení, záleží ovšem na konkrétní webové stránce. Může nastat situace, že JS bude neustále zbytečně aktualizovat určité prvky na webové stránce, což může vést k jejímu celkovému zpomalení. Stejně tak může shromažďovat určité informace ze strany uživatele webové stránky, které

by nerad sdílel. Z tohoto pohledu se zdá jeho vypnutí v určitých situacích jako ideální řešení, ovšem jako prevence stejně tak stačí nenavštěvovat nedůvěryhodné weby.

Zvětšit výkon, interaktivitu či responzivitu webových aplikací jde ruku v ruce s používám AJAX (Asynchronous JavaScript and XML) [15] technologií. Tyto technologie umožňují změnit obsah webové stránky bez nutnosti jejich znovunačítání. Využívají dynamického HTML (DHTML). V praxi se jedná o kombinací technologií HTML, CSS, JS a také dokumentového objektového modelu. Když AJAX posílá data ze serveru na prohlížeč, data předtím převede do serializovatelné podoby ve formátu XML (Extensible Markup Language).

1.1.5 React

React (nebo také ReactJS) [16] je nejpopulárnější (viz obrázek 1.2) frontendová JavaScriptová knihovna pro tvorbu uživatelského rozhraní založená na principu vytváření znovupoužitelných UI komponent, které mají vlastní logiku a vnitřní stav.

Tento stav (ang. state) [17] je reprezentace těch částí aplikace, které se dají měnit. Pokud je stav nějaké komponenty změněn, React zareaguje překreslením této komponenty v jejím novém stavu. Spojení z předchozí věty: „React zareaguje", přesně popisuje (jak i název knihovny napovídá), co v podstatě React dělá, tedy reaguje na změny stavu.

Je vhodný pro tvorbu single-page nebo mobilních aplikací, jelikož umožňuje vytvářet webové aplikace, které dokáží měnit svá data bez potřeby znovunačítání stránky. Je také snadno integrovatelný s dalšími JavaScriptovými knihovnami nebo frameworky, jako například AngularJS. Za vývojem stojí převážně Facebook, mimo jiné také komunita [18]. Základy stojí [19] na konceptu virtuálního DOMu, kterému se detailněji věnují následující sekce.

1.1.5.1 DOM

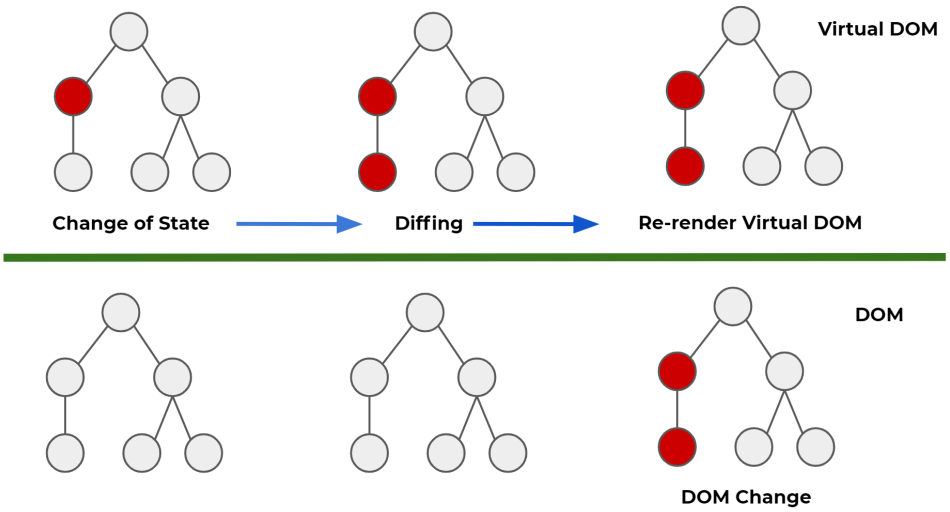
Objektový model dokumentu (ang. Document Object Model, DOM) [17] je rozhraní pro modifikaci HTML a XML dokumentů. Reprezentuje dokument jako objekty a uzly, díky čemuž lze pomocí programovacího jazyka měnit jeho obsah, strukturu nebo také styl.

Je reprezentován stromovou strukturou [20], což vede k rychlému provedení změn v rámci DOMu samotného. Problém nastává ve chvíli, kdy musí každý aktualizovaný element, včetně jeho dětí, překreslit jemu příslušnou část UI, u které ke změně došlo, čímž se celý proces značně zpomaluje.

1.1.5.2 Virtuální DOM

Virtuální Objektový Model Dokumentu (ang. Virtual Document Object Model, VDOM) [16] je paměťová reprezentace skutečného DOMu vytvářena komponentami.

Při příchodu požadavku na změnu obsahu stránky, tedy při úpravě stavu (popsáno v kapitole 1.1.5) se nejprve vytvoří nový VDOM, kde každý element je reprezentován jako uzel stromu. Poté se pomocí porovnávacího algoritmu porovná strom nového VDOMu se stromem předchozího VDOMu, dojde k výpočtům, v čem přesně se tyto stromy liší a jak nejlépe provést změny v reálném modelu. React pak už jen tyto změny provede aktualizováním příslušných objektů [20]. Tomuto procesu se říká smíření [18] (ang. reconciliation) (viz obrázek 1.3), přičemž se vykreslují pouze ty komponenty, které se skutečně mění. Počet modifikací je tedy v případě virtuálního modelu znatelně nižší, jelikož nemusíme aktualizovat celý DOM, z čehož vyplývá větší rychlost aplikace.



Obrázek 1.3: Proces smíření [21]

1.1.6 Material Design

Material Design [22] je návrhový jazyk orientovaný převážně na Android, dále iOS, Flutter a webové aplikace. Byl vytvořen Googlem za účelem pomoci designérům využívat širokého spektra interaktivních prvků pro tvorbu vysoce kvalitních digitálních vizualizací.

Material Design reflektuje objekty v reálném světě [23], jak reagují na světlo nebo také třeba, jak vrhají stíny. Jako základní stavební kámen pro tvorbu

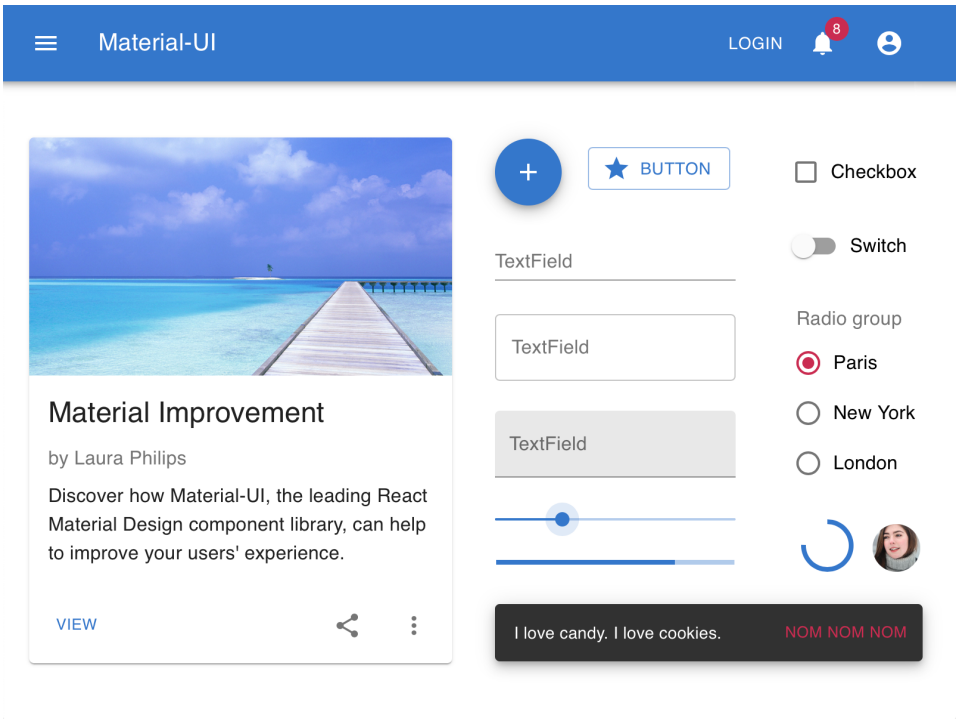
uživatelského rozhraní se využívají Material komponenty. Tyto komponenty umožňují například pokročilé ovládání vstupů, akcí, navigace nebo také organizování obsahu.

1.1.6.1 Material UI

Podobně jako Vuetify nebo Angular Material je Material UI jedna z neoficiálních implementací Material Designu s otevřeným zdrojovým kódem.

Tato knihovna následuje filosofii Material Designu, ovšem jedná se o samostatnou knihovnu plnou komponent pro tvorbu uživatelského rozhraní, nelze o ní tedy hovořit jako o pouhé implementaci. Také je třeba zmínit, že má velmi propracovanou dokumentaci, do které bylo ze strany vývojářů vloženo velké úsilí [24].

Material UI komponenty [25] byly vytvořeny tak, aby byly dostupné, v souladu s HTML5 a fungovaly na principu využívání pouze těch stylů, které potřebují zobrazovat. Na obrázku 1.4 lze vidět ukázku použití základních komponent, které knihovna nabízí. Na žádném prvku z této ukázky nebylo třeba manuálních změn jakýchkoliv CSS souborů.



Obrázek 1.4: Ukázka komponent Material UI [25]

1.1.7 Responzivita

Jeden z velkých trendů týkající se vývoje webových aplikací je responzivita [26], jinak řečeno vytváření optimalizovaného zobrazení webové stránky pro větší množství zařízení, tedy v praxi schopnost dynamické adaptace webového rozhraní na prostředí.

Existují 3 hlavní složky pro tvorbu responzivního rozhraní: [27]

- přizpůsobivé rozložení stránky, typicky založené na gridu
- přizpůsobivé obrázky a média
- media queries

Grid [28] je CSS3 modul rozložení stránky usnadňující stylování webových stránek založený na principu mřížky. Jedná se o dvoudimenzionální systém řádků a sloupců. Umisťování prvků na stránce je pomocí gridu výrazně jednodušší než komplikované a často nejasné pozicování pomocí tradičních CSS stylů.

Může se kupříkladu stát, že při zmenšení velikosti okna, tedy například při spuštění aplikace z mobilu, bude obrázek, který býval na stolním počítači přizpůsobený velikosti obrazovky, až příliš velký. V takové chvíli obrázek přesahuje element, v němž by měl být obalen. Něco takového lze vyřešit jednoduše pomocí přidání CSS stylu `max-width:100%` k obrázku. Tím se velikost elementu, ke kterému je tento styl přidán, tedy zmíněného obrázku, limituje na maximální velikost elementu, ve kterém se nachází.

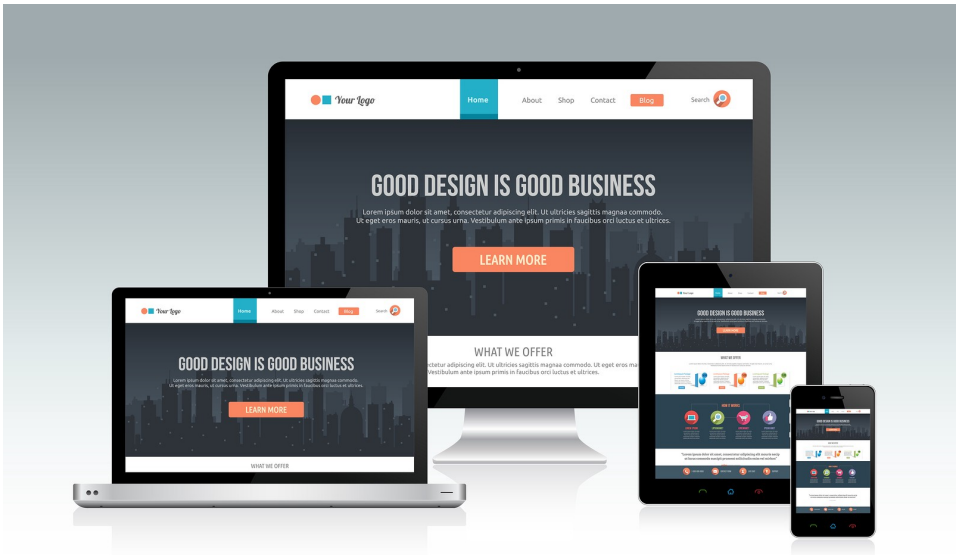
Media query je CSS technika, která umožňuje manipulovat s rozhraním při změnách rozlišení, což z ní dělá užitečný nástroj při škálování pro různá zařízení.

K tvorbě responzivního designu se využívá relativních jednotek (ang. relative units) [29], jako jsou třeba procenta. Jejich výsledná délka se dynamicky mění na základě jiné délky. Mezi nejpoužívanější relativní jednotky mimo zmíněné procenta patří:

- emy a remy – u emů délka vychází z velikosti fontu rodičovského elementu, kdežto u remů z velikosti fontu dokumentu
- vh a vw – viewport height a viewport width udávají délku na základě velikosti výšky, respektive šířky okna (prohlížeče)

Jelikož pixely jsou absolutní jednotky udávající fixní velikost, tak neškálují dle velikostí ostatních elementů stránky. Pro tvorbu aplikace, která by měla

působit uživatelsky příjemně na více zařízeních se nejeví jako nejlepší volba. Samozřejmě je na programátorovi, zda se chce držet volby fixních velikostí prvků, ovšem na úkor nepříliš pružného rozhraní. Dle Ethana Marcotta [27], amerického programátora, který roku 2010 přišel s pojmem responzivní webový design, by používání relativních jednotek při tvorbě responzivního rozhraní mělo být standardem.



Obrázek 1.5: Ukázka responzivní webové aplikace [30]

1.2 Backend

Backend je jádrem klientské aplikace, stará se o její logiku a o práci s daty. Nejčastěji je realizována pomocí API (1.2.1.3), které slouží jako prostředník mezi těmito daty a klientskou částí.

Při příchodu požadavku z klienta je povinností backendu na požadavek vytvořit odpověď a poslat ji zpět klientovi. Jsou tři hlavní části, které se na tomto principu podílejí: [31]

- server – zařízení, které přijímá požadavky z klienta, typicky počítač
- databáze – systém, který ukládá a organizuje data
- aplikace – jedná se o aplikaci běžící na serveru, pracuje s daty uloženými v databázi a posílá odpověď klientovi

1.2.1 REST API

RESTful API nebo také REST API je API, které vyhovuje architektuře fungující na principu klient-server, tedy RESTu. Jedná se o velmi populární způsob komunikace mezi klientem a serverem.

Neustále se diskutuje [32] o tom, jak vlastně správně navrhnout RESTful webové služby. Hlavním důvodem bude chybějící výchozí model, který by je přesně popisoval, což vede k tomu, že spousta již existujících služeb nutně nedodržuje všechny konvence a to může vést kupříkladu ke ztrátě rozšiřitelnosti a škálovatelnosti.

1.2.1.1 REST

REpresentation State Transfer (REST) [33] je architektura rozhraní využívána pro distribuované prostředí pro usnadnění komunikace mezi počítačovými systémy. Formálně byla představena Roy Fieldingem v roce 2000 jako součást jeho disertační práce.

Základní charakteristikou RESTful systémů [34] je rozdělení aplikace na serverovou a uživatelskou část. Velmi často využívaným přístupem je nezávislá implementace klienta a serveru tak, že klient neví o existenci serveru a naopak. Velkou výhodou tohoto přístupu je možnost modifikace kódu těchto implementací, aniž by na sebe měly vzájemný vliv. Pokud by tedy existovalo více klientských částí, všechny by na libovolný dotaz dostaly od serveru stejnou odpověď.

Definován je také přístup pro manipulaci se zdrojem realizovaný pomocí čtyř základních funkcí, tzv. CRUD operací: [35]

- create – vytvoření záznamu
- read – čtení záznamu
- update – modifikace záznamu
- delete – smazání záznamu

Komunikace, tedy čtení, editování, mazání a vytváření obsahu v REST prostředí se provádí pomocí HTTP (1.2.1.2) metod.

1.2.1.2 HTTP

Hypertext Transfer Protocol (HTTP) je internetový protokol sloužící pro komunikaci mezi webovým klientem a webovým serverem. Tato komunikace stojí na principu odesílání požadavků a přijímání odpovědí, přičemž transportním

kanálem vytvářejícím spojení je Transmission Control Protocol (TCP). Mezi základní HTTP metody patří: [36]

- GET – používá se k vyžádání dat ze serveru
- POST – slouží k vytváření nebo aktualizování obsahu na serveru
- PUT – modifikuje již existující obsah
- DELETE – maže konkrétní obsah na serveru

HTTP metod je samozřejmě více a mají své specifické výhody a úskalí. Některé z nich jsou idempotentní, tedy při jejich opakovaném volání bude výsledek vždy stejný. Rozlišuje se také, zda jsou metody bezpečné či nikoliv. Metoda je označována jako bezpečná v případě, že slouží pouze k získávání informací, tedy „pouze ke čtení“ (ang. read only).

Tabulka 1.1: Shrnutí bezpečnosti a idempotence HTTP metod [37]

Metoda	Bezpečná	Idempotentní
CONNECT	ne	ne
DELETE	ne	ano
GET	ano	ano
HEAD	ano	ano
OPTIONS	ano	ano
POST	ne	ne
PUT	ne	ano
TRACE	ano	ano

HTTP požadavek může být odchycen, přečten a dokonce i změněn, tedy může dojít k zfalšování obsahu požadavku a narušení komunikace. V případě nutnosti odeslat informace o kreditní kartě nebo heslo nějakého účtu lze využít Hypertext Transfer Protocol Secure (HTTPS). Ten umožňuje šifrování dat, ať už při jejich odesílání či přijímání. [36]
Odpovědi odeslané serverem na klienta pomocí HTTP stavového kódu:

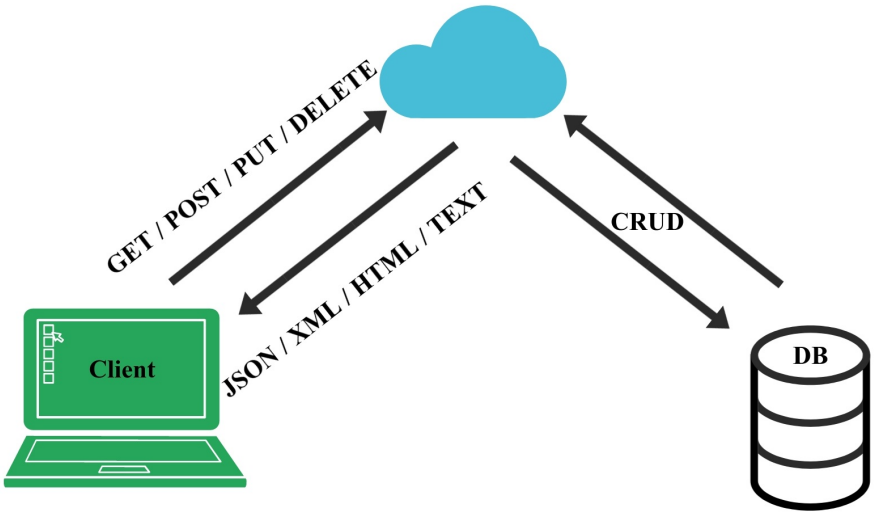
- 100 – podání informace
- 200 – úspěch
- 300 – přesměrování
- 400 – nastala chyba klienta
- 500 – nastala chyba serveru

1.2.1.3 API

Rozhraní pro programování aplikací (ang. Application Programming Interface, API) [38] je termín užívaný v softwarovém inženýrství. Jedná se o soubor procedur a různých funkcí, které programátoři využívají k vývoji aplikací.

V praxi funguje API jako prostředník (ang. middleware) pro komunikaci mezi serverem a klientem. Typicky klient pošle dotaz na API, součástí dotazu je typ požadavku a konkrétní identifikace zdroje, v praxi HTTP metoda a Uniform Resource Identifier (URI) [39].

Webové služby byly původně vymyšleny tak, aby využívaly Simple Object Access Protocol (SOAP), který funguje na principu odesílání dat ve formátu XML pomocí HTTP metod. V dnešní době se však častěji používá REST API.



Obrázek 1.6: Přiblížení architektury REST API [40]

1.2.2 JSON

JavaScript Object Notation nebo také JSON [41] je způsob zápisu dat určený primárně k jejich přenosu, typicky se používá k výměně dat u webových aplikací. Pro počítače je snadné jej generovat a zpracovat, stejně jako pro člověka je snadné jej číst i psát. Jedná se o nezávislý textový formát, avšak držící se konvencí podobným programovacím jazykům rodiny C.

1.2.3 NoSQL databáze

NoSQL (dle některých „non SQL"nebo „not only SQL") databáze jsou netabulkové, ukládající data jiným způsobem než tradiční relační databáze.

Tento koncept umožňuje lepší kontrolu, jednoduchou vertikální i horizontální škálovatelnost, a tak se tyto databáze často využívají k práci s velkými daty. V porovnání s SQL databázemi jsou méně náročné co se výkonu týče, ovšem využívají větší množství paměti. V roce 2000, když se objevily NoSQL databáze, cena datových úložišť začala prudce klesat [42]. V dnešní době je tedy paměť v porovnání s výkonem levná záležitost a tak i z tohoto důvodu se tyto databáze stále více používají.

1.2.4 MongoDB

MongoDB [43] je dokumentová databáze patřící mezi NoSQL databáze. Je pružná, škálovatelná a má dynamické schéma, díky čemuž je integrace dat podstatně rychlejší. Data ukládá jako JSON dokumenty, jedná se totiž o přirozený způsob, jak nad daty přemýšlet oproti klasickým databázovým modelům typu řádek/sloupec. Hodnoty v databázi ovšem mohou být i různé zřetězené objekty a pole. Lze také využít Mongo dotazovacího jazyka, který umožňuje pokročilé filtrování. Dotazy jsou psány v JSONu.

Kapitola 2

Návrh

Tato kapitola se zabývá analýzou vedoucí k tvorbě požadavků na aplikaci. Následně popisuje tvorbu grafického návrhu, jehož model lze najít na přiložené SD kartě. Závěrem kapitoly jsou rozebrány jednotlivé REST API endpointy vytvořené ve spolupráci s autorem serverové části.

2.1 Analýza

Cílem této sekce je identifikace podstatných vlastností a funkcionalit, kterými by výsledná aplikace měla disponovat.

2.1.1 Analýza existujícího řešení

Projekt Česká elektronická knihovna [44], respektive jeho první iterace, se uskutečnila již roku 2005. Tento projekt byl vytvořen ve spolupráci s inSophy, Centrem softwarových řešení pro vědu a špičkové technologie. Cílem projektu bylo zpřístupnění jednotlivých knih pro uživatele formou knihovny, jež fyzicky neexistuje. Projekt měl sloužit badatelům či zájemcům o poezii.

Obsahově se jedná o poezii 19. století reprezentující 1200 básnických knih. Existující řešení obsahuje a umožňuje:

- autentizaci
- zobrazení textu jednotlivých knih
- vyhledávání na různé úrovni textu
- filtrování a třídění seznamu sbírek
- statistické údaje o sbírkách
- editaci jednotlivých sbírek

- uživatelské poznámky
- administraci aplikace

2.1.2 Analýza požadavků

Analýza požadavků a jejich specifikace slouží k identifikaci potřebných funkcionalit. Jedná se o nejvýznamnější [45] část tvorby softwarového návrhu. Požadavky lze rozdělit do dvou kategorií:

- Funkční požadavky – Jedná se o funkcionality, které popisují chování systému a jsou pro uživatele nezbytné. Příkladem může být požadavek na filtrování knih v seznamu.
- Nefunkční požadavky [46] – Jedná se o specifikace funkcionalit, které zajišťují bezpečnost, škálovatelnost, spolehlivost a rozšířitelnost systému. Může se například jednat o požadavek na responzivitu aplikace.

2.1.2.1 Souhrn požadavků

Hlavním cílem je renovace existující aplikace vytvořením aplikace nové, která nabídne inovativní řešení a moderní vzhled. Velmi důležitá je možnost editace sbírek a s tím související filtrování či vyhledávání. Jedná se především o usnadnění práce editorům, kteří byli momentálně nuceni sbírky editovat v Microsoft Wordu.

Na základě jednání s mým vedoucím, mým spolupracovníkem a se zadavateli z Ústavu pro českou literaturu Akademie věd České republiky byly identifikovány funkční (tabulka 2.1) a nefunkční požadavky (tabulka 2.2) na systém.

Tabulka 2.1: Funkční požadavky

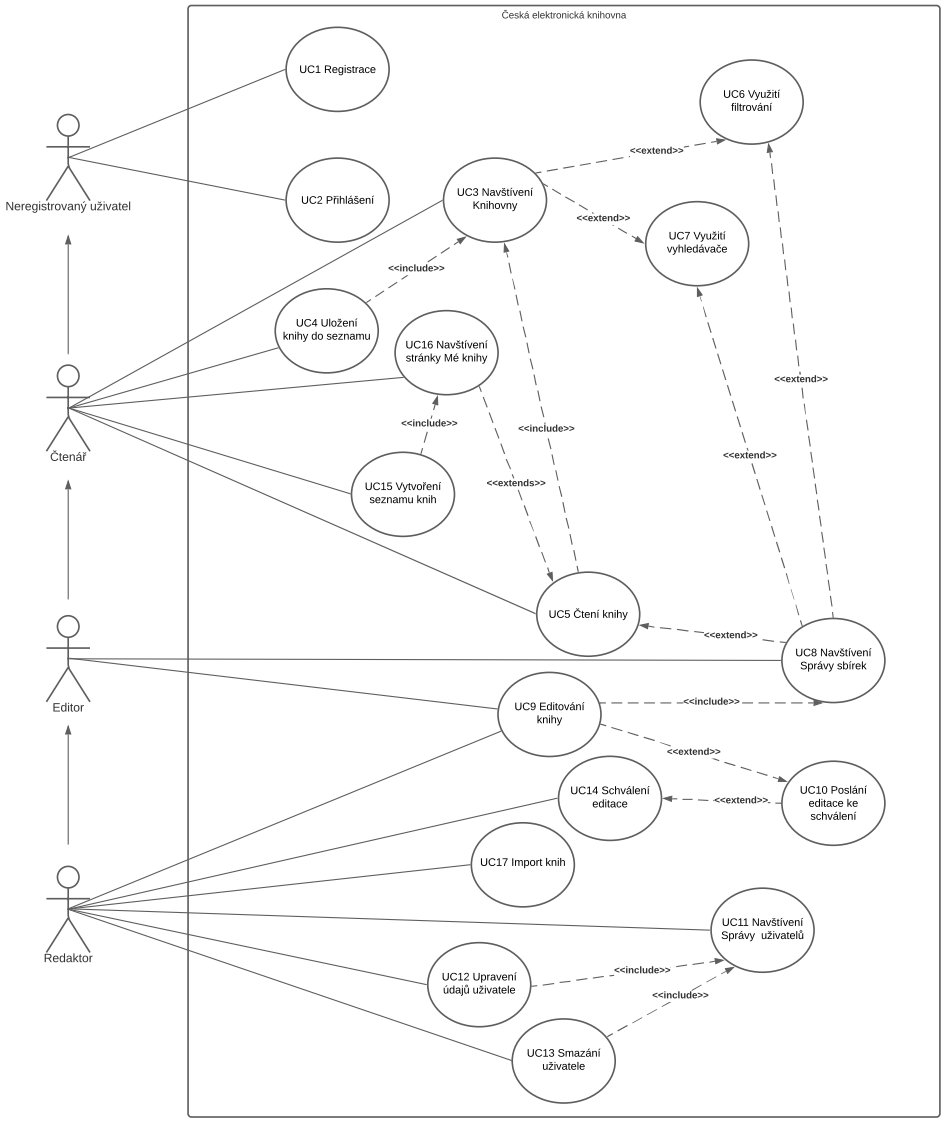
Požadavek	Popis	Priorita
FP-1 Registrace	Systém by měl uživateli umožnit vytvořit účet skrz registrační formulář. Pro úspěšnou registraci je nutné vyplnit jméno, příjmení, e-mail a heslo, přičemž e-mail musí být unikátní. Registrace je samostatná stránka jednoznačně určena svým URL.	Vysoká
FP-2 Přihlášení a odhlášení	Systém by měl umožnit registrovanému uživateli přihlášení k účtu pomocí příslušného e-mailu a hesla. Po přihlášení by uživatel měl mít možnost se z účtu odhlásit. Přihlášení je samostatná stránka jednoznačně určena svým URL.	Vysoká
FP-3 Čtení knih	Systém by měl přihlášenému uživateli umožnit čtení vybrané knihy. Knihy jsou přístupné v Knihovně nebo ve Správě sbírek formou seznamu. Knihovna i Správa sbírek jsou samostatné stránky jednoznačně určeny svým URL. Do Správy sbírek má přístup pouze privilegovaný uživatel s rolí editora či redaktora.	Vysoká
FP-4 Vyhledávání knih	Systém by měl přihlášenému uživateli umožnit vyhledávat knihy podle jejich názvu. Vyhledávač knih je dostupný v Knihovně a ve Správě sbírek.	Vysoká
FP-5 Filtrování knih	Systém by měl přihlášenému uživateli umožnit abecední řazení knih podle jména autora, názvu knihy, místa a roku vydání. Filtrování knih je dostupné v Knihovně a ve Správě sbírek.	Vysoká
FP-6 Ukládání knih	Systém by měl přihlášenému uživateli umožnit uložit vybranou knížku do seznamu knih. Ukládání knih je dostupné v seznamu knih v Knihovně a Správě sbírek.	Vysoká
FP-7 Editace knih	Systém by měl privilegovanému uživateli v roli editora umožnit editovat text vybrané knihy a požádat o schválení editace. Privilegovaný uživatel v roli redaktora může text vybrané knihy editovat bez procesu schválení editace. Editace sbírek je dostupná ve Správě sbírek. Správa sbírek je samostatná stránka jednoznačně určena svým URL.	Vysoká
FP-8 Vytvoření seznamu knih	Systém by měl přihlášenému uživateli umožnit vytvářet vlastní seznamy knih, jejichž počet není omezen. Vytvoření seznamu knih je dostupné na stránce Mé knihy. Mé knihy je samostatná stránka jednoznačně určena svým URL.	Vysoká
FP-9 Správa uživatelů	Systém by měl privilegovanému uživateli v roli redaktora umožnit editovat údaje jiných uživatelů. Editovatelné je jméno, příjmení a uživatelská role. Privilegovaný uživatel v roli redaktora může také jiného uživatele odstranit a tím mu zamezit přístup do aplikace.Správa uživatelů je samostatná stránka jednoznačně určena svým URL.	Vysoká
FP-10 Schvalování editací	Systém by měl privilegovanému uživateli v roli redaktora umožnit schvalovat editace knih méně privilegovaných uživatelů. Schvalování editací je dostupné ve Správě sbírek.	Střední
FP-11 Import knih	Systém by měl privilegovanému uživateli v roli redaktora umožnit importovat nové knihy do systému.	Střední

Tabulka 2.2: Nefunkční požadavky

Požadavek	Popis	Priorita
NP-1 Autorizace	Systém by měl umožnit přihlášenému uživateli provádět operace, ke kterým má příslušná oprávnění. Tato oprávnění jsou jednoznačně určena uživatelskými rolemi. Mezi tyto role patří čtenář, editor a redaktor.	Vysoká
NP-2 Responzivní rozhraní	Systém by měl umožnit přihlášenému uživateli využívat aplikaci z mobilního zařízení.	Střední

2.1.3 Případy užití

Případ užití (ang. Use Case) je seznam funkcí, které jsou vymezeny pro uživatele s danou rolí. Jednotlivé případy užití a s nimi spojené uživatelské role lze vidět na UML (Unified Modeling Language) diagramu reprezentující chování systému z pohledu uživatele (viz obrázek 2.1). Jedná se o diagram případů užití (ang. Use Case diagram).
Je důležité, aby byly všechny uživatelské požadavky pokryty případy užití. Pro přehled a ověření pokrytí všech požadavků slouží tabulka 2.3.



Obrázek 2.1: Diagram případů užití

Tabulka 2.3: Pokrytí funkčních požadavků případy užití

	Požadavky										
Případy užití	FP-1	FP-2	FP-3	FP-4	FP-5	FP-6	FP-7	FP-8	FP-9	FP-10	FP-11
UC1	X										
UC2		X									
UC3			X	X	X	X					
UC4						X					
UC5			X								
UC5			X				X				
UC6			X	X	X	X	X				
UC7			X	X	X	X	X				
UC8							X				
UC9							X				
UC10							X				
UC11									X		
UC12									X		
UC13									X		
UC14										X	
UC15								X			
UC16								X			
UC17											X

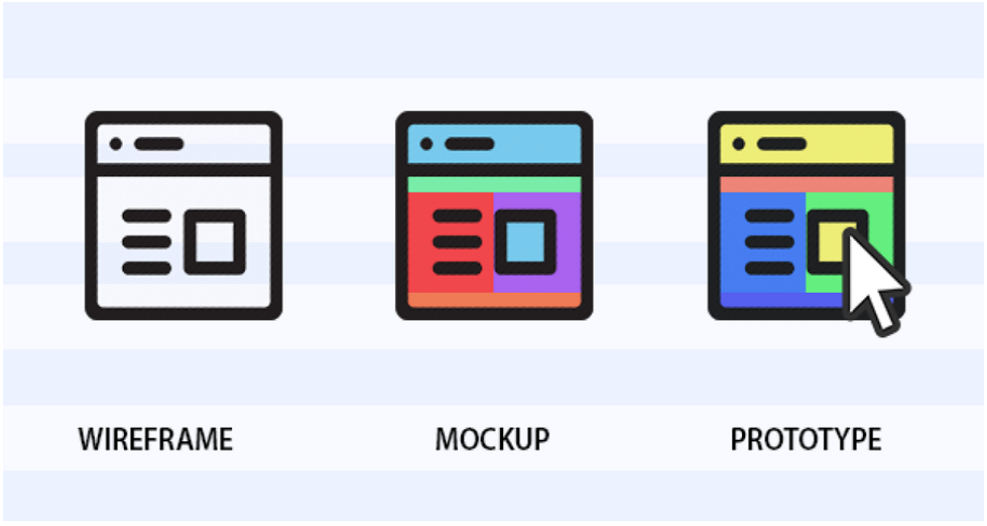
2.2 Grafický návrh

Posledním krokem v návrhu aplikace je vytvoření modelu aplikace. Oblíbená je tvorba drátěného modelu (ang. wireframe), jehož cílem je prezentace nového řešení klientovi. V tomto řešení bývají typicky zahrnuty klientské požadavky, a tak je tento model odrazem fungování webu včetně rozmístění požadovaných prvků na stránce. Nejedná se však o grafický návrh, za který je často zaměňován. Existují však i jiné typy modelů.

Dalším typem modelu aplikace je mockup. Jedná se v podstatě o wireframe s detailnějším zaměřením na vzhled aplikace s hlediska barev a typografie.

Posledním zmíněným způsobem vytváření modelu aplikace bude Prototyp [47]. Prototypy jsou pokročilejší formou wireframu. Jsou často velmi podobné výslednému vzhledu aplikace. Oproti drátěnému modelu využívají k prezentaci řešení grafických prvků. Jedná se o ideální způsob, jakým předvést komunikaci, vzhled a interakci prvků, které aplikace obsahuje.

K vytvoření modelu aplikace bylo využito nástroje Justinmind [48], který nabízí širokou škálu možností pro vykreslení produktu a jeho designu. Tvorba si zakládala na splnění stanovených klientských požadavků, které ovšem tou dobou nebyly kompletní, respektive právě tvorba tohoto modelu odhalila jisté nedostatky. Nedostatky se týkaly především možností editace knih, kde po návazných schůzkách s členy Ústavu pro českou literaturu byly upřesněny další specifikace. Model byl původně zamýšlen jako wireframe, v průběhu tvorby se však přidaly grafické prvky a tím byl model povýšen na mockup.



Obrázek 2.2: Typy modelu aplikace [49]

2.3 REST API endpointy

Koncový bod, dále jen endpoint, je jeden konec komunikačního kanálu, který reprezentuje umístění ze kterého API zpřístupňuje přiřazená data. Typicky endpoint obsahuje URL serveru a při poslání dotazu vrací API odpověď právě ve formě dat.

2.3.1 Implementované rozhraní

Nejprve je potřeba identifikovat data, které by rozhraní na konkrétním URL mělo zpřístupnit. Pro správnou funkci sekcí *knihovna*, *mé knihy* a *správa sbírek* je potřeba mít k dispozici název knihy, jméno autora, rok a místo vydání. Rozhraní by také mělo zpřístupnit jednoznačný identifikátor knihy, díky němuž se pomocí API lze dostat k informacím jedné konkrétní knihy. Sekce *správa uživatelů* a přihlášení zase potřebuje přístup k datům uživatelů. Konkrétně se jedná o jméno, příjmení, email a uživatelskou roli. I v tomto případě by rozhraní mělo zpřístupnit jednoznačný identifikátor pro případ získání dat konkrétního uživatele. Důležitá je také práce se seznamy zahrnující potřebu získání dat pro implementování funkcionalit obsažených v stávajícím řešení České elektronické knihovny. Tyto funkcionality budou rozebrány v následující kapitole. Rozhraní musí také umožnit mazání a editaci výše zmíněných dat. Základní endpointy pro výše zmíněné operace jsou uvedeny v následujících tabulkách.

Tabulka 2.4: Endpointy ke zdroji Kniha

Typ požadavku	**Cesta**	**Popis**
GET	/books	Vrací základní informace o všech knihách
GET	/books/{id}	Vrací veškeré informace o konkrétní knize
GET	/books/{id}?content	Vrací text konkrétní knihy
PUT	/books/{id}	Aktualizuje informace konkrétní knihy
DELETE	/books/{id}	Odstraňuje konkrétní knihu

Tabulka 2.5: Endpointy ke zdroji Uživatel

Typ požadavku	**Cesta**	**Popis**
GET	/users	Vrací veškeré údaje o všech uživatelích
POST	/users	Vytváří nového uživatele
GET	/users/{id}	Vrací veškeré údaje konkrétního uživatele
PUT	/users/{id}	Aktualizuje údaje konkrétního uživatele
DELETE	/users/{id}	Odstraňuje konkrétního uživatele

Tabulka 2.6: Endpointy ke zdroji Seznam knih

Typ požadavku	**Cesta**	**Popis**
GET	/users/{userId} /bookLists	Vrací veškeré údaje o všech uživatelích
POST	/users/{userId} /bookLists	Vytváří nového uživatele
GET	/users/{userId} /bookLists/{bookListId}	Vrací veškeré údaje konkrétního uživatele
PUT	/users/{userId} /bookLists/{bookListId}	Aktualizuje údaje konkrétního uživatele
DELETE	/users/{userId} /bookLists/{bookListId}	Odstraňuje konkrétního uživatele
GET	/users/{userId}/bookLists /{bookListId}?alphabeticalDictionary	Vrací abecední slovník konkrétního seznamu knih
GET	/users/{userId}/bookLists /{bookListId}?frequencyDictionary	Vrací frekvenční slovník konkrétního seznamu knih
GET	/users/{userId}/bookLists /{bookListId}?stats	Vrací statistické údaje konkrétního seznamu knih
GET	/users/{userId}/bookLists /{bookListId}?fulltext	Vrací výsledek fulltextového vyhledávání aplikovaného na konkrétní seznam knih
GET	/users/{userId}/bookLists /{bookListId}?allTexts	Vrací všechny texty knih, které jsou obsaženy v konkrétním seznamu knih

Kapitola 3

Realizace

3.1 Architektura frontendu

Stěžejní část tvoří knihovna React (1.1.5), díky níž lze využívat již jednou vytvořené komponenty znovu na více místech v kódu. Díky principům, na kterých je React založen, kupříkladu VDOM (1.1.5.2), se zvyšuje rychlost aplikace, přesněji nároky na výkon nejsou tak velké. Zároveň neexistuje případ, kdy by bylo potřeba znovu načítat stránku, tedy veškerý přístup v aplikaci včetně využívání jejích funkcionalit je uživatelsky velmi příjemný a zanechává moderní dojem. Důležitou součástí je také knihovna Material UI (1.1.6.1), díky níž lze stavět tvorbu UI na principech Material Designu a to především díky hotovým komponentám, které Material UI nabízí. Mezi další knihovny uplatněné při implementaci patří Material Table, React-Json-Schema-Form a Ant Design.

3.2 Popis aplikace

Dostupné stránky pro veřejnost, tedy nepřihlášené uživatele, jsou pouze domovská stránka, stránka o projektu, přihlášení a registrace. Po přihlášení se uživateli rozšíří navigační menu o nové položky. Jedná se o samostatné stránky, jejichž obsah se vykreslí namísto původní stránky v případě, že uživatel provede příslušnou akci. Jednotlivé položky jsou, stejně jako jejich URL adresy, přístupné pouze určitým skupinám uživatelů.

Standardní uživatel, jemuž je přiřazena role čtenáře má z navigačního menu mimo domovskou stránku a stránku projektu přístup k sekcím *knihovna* a *mé knihy*. Uživatel s rolí editora má navíc přístup do sekce *správa sbírek*. Naproti tomu uživatel, jemuž je přiřazena role redaktora má kromě přístupu do sekce *správa sbírek* navíc i přístup do sekce *správa uživatelů*.

3.2.1 Autentizace

Registrace a přihlášení jsou veřejně dostupné formou formulářů. Tyto formuláře jsou implementovány za pomocí předem připravené komponenty <Form/> knihovny Material UI. Po přihlášení má uživatel možnost se odhlásit pomocí položky v navigačním menu. Uživatel, který se jednou přihlásí, bude po znovuotevření stránky stále přihlášen, a to díky uloženým údajům v Local Storage. Local Storage je úložiště v prohlížeči, které slouží k uložení dat klienta. Úložiště je přístupné JavaScriptem. Po odhlášení se tato data z Local Storage vymažou. Údaje o uživateli jsou po celou dobu přihlášení drženy aplikací v hlavní komponentě pomocí state objektu. Důležitý je údaj o uživatelské roli, podle něhož se v jednotlivých komponentách zpřístupňují roli odpovídající funkcionality. Všechny tyto údaje jsou globálně rozšířeny mezi všechny komponenty a to pomocí React Context, který umožňuje průchod dat skrz strom komponent.

3.2.2 Knihovna

Sekce *knihovna*, stejně jako všechny její funkcionality, je přístupná pro libovolného přihlášeného uživatele. Slouží k prezentaci seznamu sbírek básní dodaných Ústavem pro českou literaturu. Tento seznam je reprezentován formou tabulky vytvořené pomocí knihovny Material Table. Je zde k dispozici stránkování a zobrazování určitého počtu záznamů, díky čemuž se snižují nároky na výkon. Tabulka umožňuje také abecední filtrování podle sloupců, jimiž jsou název knihy, autor, místo vydání a rok vydání. K dispozici je také vyhledávání na základě hledaného řetězce mezi všemi sloupci ve všech záznamech. Filtrování i vyhledávání je implementováno jako součást Material Table. S tímto množstvím údajů a stránkováním jsou tyto vestavěné funkcionality velmi efektivní. Posledním sloupcem jsou akce, kde se nachází jednotlivá tlačítka spouštějící požadované události. Konkrétně se jedná o čtení knihy, reprezentované zobrazením textu v dialogovém okně. Dále uložení knihy, reprezentované zobrazením existujících seznamů taktéž formou dialogového okna s možností vybrání právě toho seznamu, do kterého má být kniha uložena.

3.2.3 Mé knihy

Tato sekce, včetně veškerých funkcionalit, je zpřístupněna pro libovolného přihlášeného uživatele. Umožňuje vytvoření nového seznamu knih pomocí textového pole a tlačítka. Do takto vytvořeného seznamu lze následně skrz *knihovnu* nebo *správu sbírek* ukládat libovolné množství knih. Vytvořené seznamy jsou formou rozbalovacího seznamu (ang. drop-down list) vyobrazeny na stránce spolu s dalšími funkcemi reprezentovanými pomocí tlačítek. Po rozbalení tohoto listu jsou k dispozici jednotlivé knihy, které lze ze seznamu

smazat, anebo stejně jako v knihovně, využít akce k přečtení jejich obsahu. Samotné seznamy knih lze také smazat anebo přejít na stránku s rozšířenými filtry.

Stránka s rozšířenými filtry obsahuje tři části tvořené sloupci. Prvním sloupcem je list knih, které jsou obsaženy v daném seznamu, na němž se přechod na stránku s rozšířenými filtry aplikoval. Prostřední sloupec slouží k zobrazování všech požadovaných dat. Posledním sloupcem jsou nástroje pro badatele, které již byly obsaženy v původním řešení a bylo je potřeba do nové aplikace reimplementovat. Mezi tyto nástroje patří statistika, která pro seznam, na něž byly rozšířené filtry aplikovány, vypíše do prostředního sloupce počet jednotlivých básnických figur. Dalšími nástroji jsou slovníky. Abecední slovník vypíše všechna slova a počet jejich výskytů v jednotlivých sbírkách seznamu seřazená abecedně. Frekvenční slovník vypíše taktéž všechna slova a jejich počet výskytů, ovšem slova jsou seřazena dle nejvyššího počtu výskytů. Dalším nástrojem je fulltextové vyhledávání, které pro zadané slovo vypíše jeho celkový počet výskytů v rámci všech knih daného seznamu. Dále vypíše kolikrát je toto slovo obsaženo v jednotlivých knihách.

3.2.4 Správa sbírek

Sekce *správa sbírek* je reprezentována stejně jako *knihovna* formou tabulky. Obsahuje stejné funkcionality, tedy možnost čtení, ukládání knih a vestavěné možnosti filtrování a vyhledávání. Navíc nabízí dvě uživatelské akce přidané do posledního sloupce tabulky akce. Jednou z nich je možnost smazání libovolné knihy z databáze. Druhou akcí je editace knih.

Tato akce vykreslí novou komponentu, která pracuje s knihovnou React-Json-Schema-Form. Jedná se o komponentu knihovny React, která dokáže z validního JSON schématu vygenerovat webový formulář, který toto schéma popisuje. Tento nástroj také umožňuje předem nahrát data do formuláře skrz React rekvizity (ang. properties, props). Tato funkcionalita umožňuje naplnit formulář daty konkrétní knihy, na které je proces editace zavolán. Tímto způsobem uživatel vidí, jaká data jsou spojená s konkrétní knížkou a může je jednoduše přepsat. Na konci formuláře se nachází tlačítko editace, které spustí potřebné akce pro přepis dat v databázi.

Správa sbírek je dostupná pouze uživateli s přidělenou uživatelskou rolí editora či redaktora. Tato stránka by také měla dle funkčních požadavků rozdělovat funkcionality dle těchto rolí. Prozatím však bylo rozhodnuto, že systém schvalování editací bude vytvářen mimo tuto práci, a to až po vytvoření systému pro správu jednotlivých uživatelských změn. Aktuálně v sekci *správa sbírek*

mají tedy uživatel s rolí redaktora i editora stejné oprávnění.

Import knih popsaný ve funkčních požadavcích, respektive jeho reprezentace na frontendu, nebude součástí této práce, nýbrž následné spolupráce s Ústavem věd. Momentálně lze knihy importovat skrz aplikaci běžící na backendu.

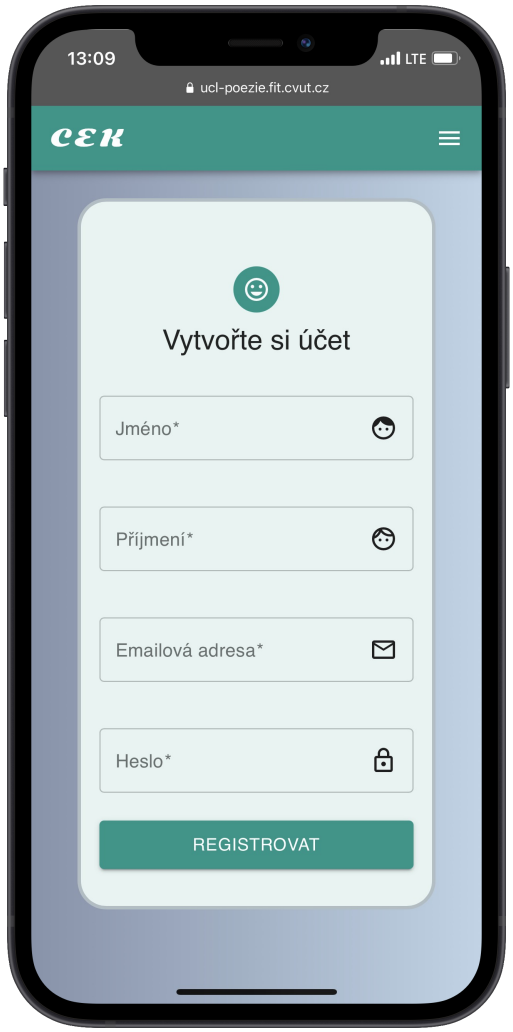
3.2.5 Správa uživatelů

Do této sekce má přístup pouze uživatel s rolí redaktora. Jedná se o přehled všech uživatelů aplikace reprezentovaný pomocí tabulky knihovny Material Table. Stejně jako v *knihovně* a *správě sbírek* je zde možnost filtrování podle sloupců a vyhledávání mezi všemi daty najednou. Tabulka obsahuje sloupce jméno, příjmení, email, role a akce. Redaktor může uživatele smazat a tím mu úplně zamezit přístup do aplikace. Může také uživatele editovat. Editovatelné jsou všechny sloupce kromě emailu. Ten je totiž jednoznačným identifikátorem uživatele v databázi, a tak je zakázáno jej editovat. Nejpodstatnější je ovšem možnost nastavit libovolnému uživateli jednu ze tří definovaných rolí a tím mu přidělit či odebrat určitá práva.

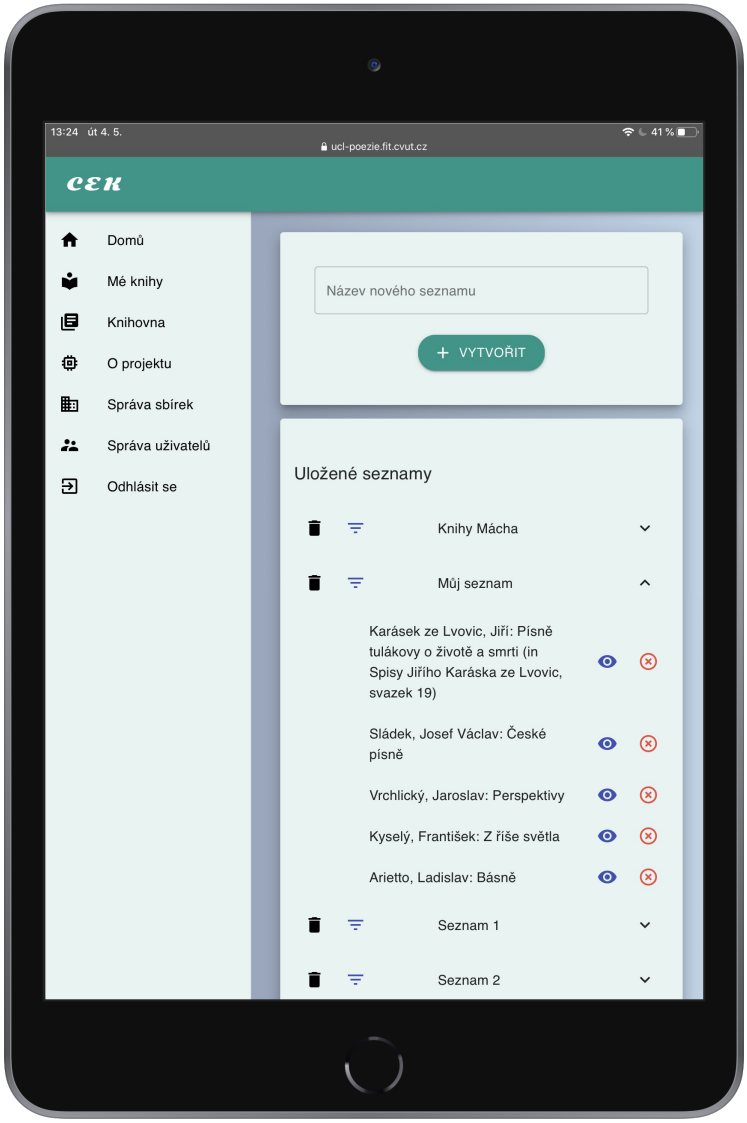
3.3 Výsledná ukázka

Tato sekce obsahuje ukázky aplikace, která je momentálně dostupná na `https://ucl-poezie.fit.cvut.cz/`. Aplikace je stále ve vývoji, nejedná se o produkci. Její prezentace tímto způsobem slouží k získání zpětné reakce z AV ČR.

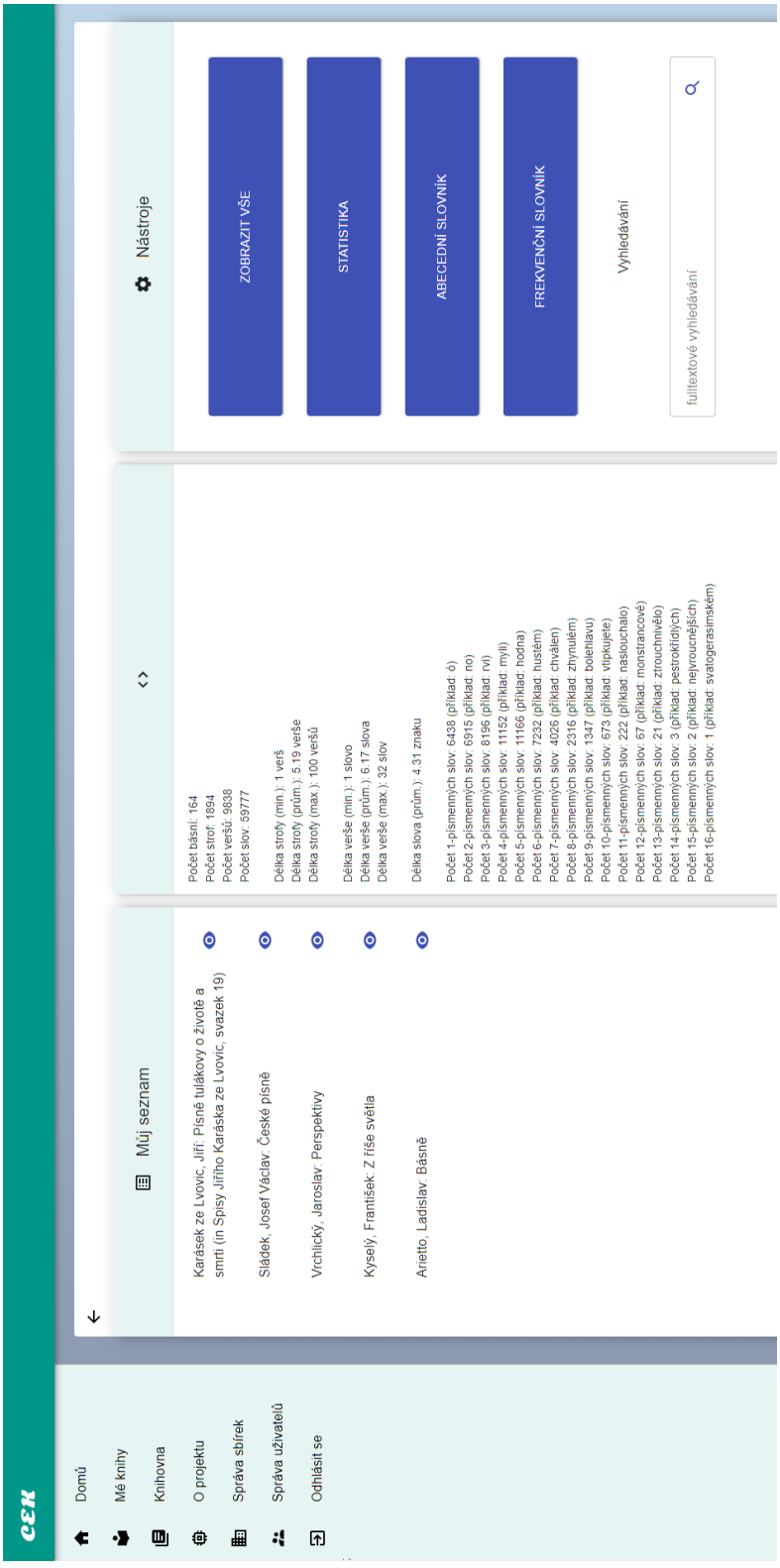
Jednotlivé snímky byly pořízeny na různých zařízeních a různých stránkách aplikace. Jedná se o ukázku vzhledu a responzivity aplikace.



Obrázek 3.1: Registrační formulář – iPhone 12 2020



Obrázek 3.2: Sekce Mé knihy – iPad Pro 2016



Obrázek 3.3: Sekce Rozšířené filtry – Desktop

Kapitola 4

Testování

4.1 Uživatelské testování

K testování klientské části jsem se rozhodl využít metodu uživatelského testování [50]. Jedná se o metodu, která je nejčastěji používaná pro otestování použitelnosti a slouží k odhalování chyb, které způsobují, že se uživatelé v aplikaci ztrácí. Důležité je nenechat uživatele zbytečně přemýšlet, kde se na webu nachází hledané informace. Veškerá navigace v aplikaci by měla být intuitivní.

Základním krokem je vybrání částí aplikace, které bude nutné testování podrobit. Podstatnými oblastmi testování jsou proces autentizace a průchod napříč celou aplikací. Důležité je také zaměřit se na detaily kolem pohybu a funkcionalit v sekcích *knihovna*, *mé knihy* a *správa sbírek*.

Podstatné je také vhodně zvolit testery a jejich počet. Pro tento projekt jsem si podle vzoru Jakoba Nielsena [51] vybral 5 testerů. Ten totiž tvrdí, že 5 testerů dokáže odhalit 75 % problémů s použitelností. Mezi první tři vybrané testery patří absolventi bakalářského studia FIT ČVUT Bc. Tomáš Halama, Bc. Iveta Šárfyová a Bc. Oleksandr Chmel, kteří se věnují vývoji webových aplikací a AI. Dále pro tento projekt byli vybráni tester společnosti NanguTV Honza Štefánik a tester společnosti NCR Pavel Kosek.

Poslední důležitou částí je sepsání testovacích scénářů dle kterých budou testeři postupovat. Scénáře by měly pokrýt využití veškerých funkcionalit a průchodů aplikací. Jednotlivé scénáře jsou popsány v tabulce 4.1

Tabulka 4.1: Testovací scénáře

Scénář	Popis
TS-1 **Autentizace**	Zaregistrujte se do aplikace. Následně se pokuste přihlásit. Pokud jste se úspěšně přihlásili, tak se odhlaste a znovu zkuste přihlásit.
TS-2 **Manipulace se seznamy**	Přejděte do sekce Mé knihy a vytvořte tři seznamy. Jeden seznam smažte.
TS-3 **Funkce tabulky**	Přejděte do sekce Knihovna. V této sekci vyzkoušejte alespoň dva řadící filtry, které nabízí a zkuste si vyhledat knihu jménem kytice. Následně smažte text ve vyhledávacím okně, nastavte zobrazení deseti záznamů na stránku a vyzkoušejte přechod mezi stránkami.
TS-4 **Provázanost knih a seznamů**	Přejděte do sekce Knihovna. Přečtěte si jednu knihu. Uložte tři knihy do libovolně zvoleného seznamu. Následně přejděte do sekce Mé knihy, přečtěte si libovolnou knihu ze seznamu a odeberte ji.
TS-5 **Rozšířené filtry**	V sekci Mé knihy si vyberte seznam obsahující knihy a přejděte na stránku s rozšířenými filtry. Na této stránce vyzkoušejte všechny nástroje, které jsou k dispozici a vraťte se na předchozí stránku.
TS-6 **Manipulace s knihami**	Zadejte kombinaci emailu *editor@editor.cz* a hesla *editor*. Přejděte do sekce Správa sbírek. Smažte knihu, jejíž název je poslední v abecedě. Následně najděte knihu, jež byla napsána v roce 1965 a upravte jméno autora. Knihu poté znovu najděte a zkontrolujte, že údaje byly skutečně změněny.
TS-7 **Manipulace s uživateli**	Zadejte kombinaci emailu *redaktor@redaktor.cz* a hesla *redaktor*. Přejděte do sekce Správa uživatelů. V této sekci najděte libovolného uživatele se jménem Tester a přiřaďte mu práva editora nebo čtenáře. Zkontrolujte, že skutečně došlo k požadovaným změnám, uživatele smažte a odhlaste se.

4.1.1 Průběh testování

Testerům bylo pro větší rozmanitost umožněno vybrat si zařízení, na kterém budou aplikaci testovat. Aplikace byla testována na následujících zařízeních:

- iPhone 12 2020
- iPhone 10 2018
- Samsung Galaxy S10 2019
- MacBook Pro 2019
- iPad Pro 2016
- 3 různá desktopová zařízení

Dohromady proběhlo 12 průchodů testovacími scénáři, z toho 7 na mobilních zařízeních s operačními systémy iOS a Android. 2 takovéto průchody proběhly na tabletovém zařízení iPad se systémem iOS. Na desktopových zařízeních s operačním systémem Windows a Linux proběhly 3 průchody scénáři.

4.1.2 Výsledek testování

Uživatelským testováním byly odhaleny následující chyby:

- Velmi pomalá editace knih.
- Proměnlivé chování při vytváření a mazání seznamů. Občas měla jedna z operací zpoždění a jediným řešením bylo obnovení stránky.
- Navigační bar byl horizontálně posuvný (scrollovatelný).
- Obnovení libovolné stránky vedlo k přesměrování na domovskou stránku.
- Dlouhé názvy seznamu knih vedly k přetečení textu z příslušného elementu.
- Skrz Správu uživatelů se dalo nastavit jméno či příjmení uživatele na prázdný řetězec.
- Zobrazování prvků na stránce s rozšířenými filtry nebylo kompatibilní s prohlížečem Safari.

Výše zmíněné chyby až na problémy s kompatibilitou prohlížeče Safari byly opraveny, ovšem z důvodu objevení nepředpokládaného množství chyb byla provedena ještě druhá iterace testování. Jejím účelem bylo zkontrolovat, zda byly chyby skutečně vyřešeny a případně zjistit další potenciální problémy.

Editace sbírek je však stále velmi pomalá, důvodem je velmi složité schéma, které knihy popisuje. Nejprve by bylo potřeba data očistit do co nejjednodušší podoby. Po schůzkách s mým vedoucím a spolupracovníkem jsme se rozhodli, že nejspíše zvolíme jiný způsob editace knih anebo použijeme tento postup za předpokladu, že bude možné vytvořit jednodušší schéma.

Druhá iterace již neodhalila žádnou další chybu, ovšem některým z testerů chyběly jisté funkcionality, které by jako uživatelé aplikace uvítali. Jednou z nich byla možnost zobrazení a schování hesla při registraci a přihlašování. Taktéž při přihlašování a registraci nebylo možné provést odeslání formuláře ke zpracování pomocí klávesy enter. Poslední připomínkou bylo nekvalitně navržené dynamické měnění velikosti elementů na stránce s rozšířenými filtry.

Všichni testeři ocenili orientaci v aplikaci, její moderní vzhled a použité UI prvky. Aplikace byla celkově hodnocena jako uživatelsky příjemná a intuitivně ovladatelná. Testování bylo úspěšné, neboť odhalilo a poukázalo na spoustu chyb a možných inovací. Veškeré důležité nedostatky byly opraveny a aplikace je tak připravena k přechodu na další verzi.

Kapitola 5

Dokumentace

Základní dokumentaci, která slouží k sestavení projektu, obsahuje soubor `README.md`. Tento soubor lze najít v kořenovém adresáři projektu. Dokumentaci frontendu popisující navigaci v projektu obsahuje soubor `doc.md`. Tato dokumentace obsahuje také seznam všech komponent včetně popisu jejich funkce v projektu. Uvedeny jsou také všechny knihovny, které byly použity při implementaci. K vytvoření dokumentace bylo využito editoru MarkdownPad2 [52].

Závěr

Cílem této práce bylo vytvoření webové aplikace pro projekt Česká elektronická knihovna Ústavu pro českou literaturu Akademie věd České republiky. Tato aplikace měla sloužit ke správě sbírek básní, které se podařilo digitalizovat. Důležité bylo také zachování funkcionalit původního řešení, jež tato práce úspěšně zanalyzovala. Práce také srovnala jednotlivé technologie potřebné k implementaci a na základě analýzy zvolila vhodné postupy tvorby.

Ve spolupráci s autorem backendu byly vytvořeny požadavky na aplikaci, následně proveden návrh a zkonzultovány jednotlivé REST API endpointy.

Uživatelskou část aplikace se s využitím knihovny React povedlo úspěšně implementovat a propojit ji s řešením Bc. Tomáše Chvosty. Výsledkem je responzivní webová aplikace splňující požadavky zadavatelů.

Práce také pomocí uživatelského testování úspěšně ověřila správnost implementace. Závěrem byla vytvořena dokumentace sloužící k orientaci v projektu.

Veškeré cíle práce byly úspěšně splněny, ovšem aplikace ještě není připravena nahradit původní řešení. Je zde tedy možnost pokračovat ve vývoji aplikace, a to v rámci projektu TAČR.

Bibliografie

1. *What do design and implementation mean?* [Software Engineering Stack Exchange] [online] [cit. 2021-02-08]. Dostupné z: https://softwareengineering.stackexchange.com/questions/275288/what-do-design-and-implementation-mean.
2. *What is Frontend? What is Backend?* [Frontend GmbH] [online]. 2019-11-01 [cit. 2021-02-02]. Dostupné z: https://www.frontend-gmbh.de/en/blog/what-is-frontend-what-is-backend/. Section: Frontend.
3. *Front-end Developer Handbook 2019 - Learn the entire JavaScript, CSS and HTML development practice!* [Online] [cit. 2021-02-02]. Dostupné z: https://frontendmasters.com/books/front-end-handbook/2019/.
4. GALITZ, Wilbert O. *The Essential Guide to User Interface Design: An Introduction to GUI Design Principles and Techniques*. USA: John Wiley & Sons, Inc., 2007. ISBN 0470053429.
5. *Graphical user interface — computing* [Encyclopedia Britannica] [online] [cit. 2021-02-05]. Dostupné z: https://www.britannica.com/technology/graphical-user-interface.
6. *Server-side web frameworks - Learn web development — MDN* [online] [cit. 2021-02-05]. Dostupné z: https://developer.mozilla.org/en-US/docs/Learn/Server-side/First_steps/Web_frameworks.
7. *The Difference Between a Framework and a Library* [freeCodeCamp.org] [online]. 2019-02-01 [cit. 2021-02-05]. Dostupné z: https://www.freecodecamp.org/news/the-difference-between-a-framework-and-a-library-bd133054023f/.
8. 262588213843476. *Front-end frameworks popularity (React, Vue and Angular)* [Gist] [online] [cit. 2021-02-08]. Dostupné z: https://gist.github.com/tkrotoff/b1caa4c3a185629299ec234d2314e190.
9. *AngularJS — Superheroic JavaScript MVW Framework* [online] [cit. 2021-02-08]. Dostupné z: https://angularjs.org/.

10. *Vue.js* [online] [cit. 2021-02-08]. Dostupné z: https://vuejs.org/.

11. V.CROMWELL. *Between the Wires: An interview with Vue.js creator Evan You* [freeCodeCamp.org] [online]. 2017-05-30 [cit. 2021-02-08]. Dostupné z: https://www.freecodecamp.org/news/between-the-wires-an-interview-with-vue-js-creator-evan-you-e383cbf57cc4/.

12. DAITYARI, Shaumik. *Angular vs React vs Vue: Which Framework to Choose in 2021* [CodeinWP] [online]. 2019-01-10 [cit. 2021-02-08]. Dostupné z: https://www.codeinwp.com/blog/angular-vs-vue-vs-react/.

13. SUCHÝ, Bc. Ondřej. Metodika vývoje webových mapových aplikací. [N.d.], s. 86. Dostupné také z: https://dspace.cvut.cz/bitstream/handle/10467/70118/F3-DP-2017-Suchy-Ondrej-suchyon6_dp_Metodika_vyvoje_webovych_mapovych_aplikaci.pdf?sequence=1&isAllowed=y.

14. *What is JavaScript Used For?* [Online] [cit. 2021-02-02]. Dostupné z: https://www.hackreactor.com/blog/what-is-javascript-used-for.

15. PAULSON, L. D. Building rich web applications with Ajax. *Computer*. 2005, roč. 38, č. 10, s. 14–17. ISSN 1558-0814. Dostupné z DOI: 10.1109/MC.2005.330.

16. *React – A JavaScript library for building user interfaces* [online] [cit. 2021-02-02]. Dostupné z: https://reactjs.org/.

17. CEDDIA, Dave. *A Visual Guide to State in React* [Dave Ceddia] [online] [cit. 2021-02-03]. Dostupné z: https://daveceddia.com/visual-guide-to-state-in-react/.

18. AGGARWAL, Sanchit. Modern Web-Development using ReactJS. *International Journal of Recent Research Aspects*. 2018, roč. 5, č. 1, s. 2349–7688. ISSN 23497688. Dostupné také z: http://search.ebscohost.com/login.aspx?direct=true&db=a9h&AN=129311347&site=ehost-live&scope=site.

19. KUMAR, Anurag; SINGH, Ravi Kumar. Comparative analysis of angularjs and reactjs. *International Journal of Latest Trends in Engineering and Technology*. 2016, roč. 7, č. 4, s. 225–227.

20. RAVICH, Adhithi. *React Virtual DOM Explained in Simple English* [Programming with Mosh] [online]. 2018-12-03 [cit. 2021-02-03]. Dostupné z: https://programmingwithmosh.com/react/react-virtual-dom-explained/. Section: React.

21. CODINGMEDIC. *The Virtual DOM* [Coding Medic] [online]. 2020-11-10 [cit. 2021-02-02]. Dostupné z: https://codingmedic.wordpress.com/2020/11/10/the-virtual-dom/.

22. *What is Material Design?* [The Interaction Design Foundation] [online] [cit. 2021-02-03]. Dostupné z: https://www.interaction-design.org/literature/topics/material-design.

23. *Introduction* [Material Design] [online] [cit. 2021-02-03]. Dostupné z: https://material.io/design/introduction#principles.

24. TASSINARI, Olivier. *Material-UI v1 is out* [Medium] [online]. 2019-07-22 [cit. 2021-02-03]. Dostupné z: https://medium.com/material-ui/material-ui-v1-is-out-e73ce13463eb.

25. *Material-UI: A popular React UI framework* [online] [cit. 2021-02-03]. Dostupné z: https://material-ui.com/.

26. NEBELING, Michael; NORRIE, Moira C. Responsive design and development: Methods, technologies and current issues. In: *International Conference on Web Engineering*. 2013, sv. 7977, s. 510–513. Dostupné z DOI: 10.1007/978-3-642-39200-9_47.

27. MARCOTTE, Ethan. *Responsive web design: A book apart n 4*. Editions Eyrolles, 2017.

28. *A Complete Guide to Grid* [CSS-Tricks] [online] [cit. 2021-02-07]. Dostupné z: https://css-tricks.com/snippets/css/complete-guide-grid/.

29. NOCIK. *Lekce 3 - Jednotky (em, rem, px, ...)* [Online] [cit. 2021-02-08]. Dostupné z: https://www.itnetwork.cz/jednotky-em-rem-px.

30. NAEEM, Subhan. *Angular — Code Design for responsive websites* [Medium] [online]. 2018-03-01 [cit. 2021-02-09]. Dostupné z: https://itnext.io/angular-code-design-for-responsive-websites-acd4259a478c.

31. *Back-End Web Architecture* [Codecademy] [online] [cit. 2021-02-04]. Dostupné z: https://www.codecademy.com/articles/back-end-architecture.

32. ZHOU, Wei; LI, Li; LUO, Min; CHOU, Wu. REST API design patterns for SDN northbound API. In: *2014 28th international conference on advanced information networking and applications workshops*. 2014, s. 358–365. ISBN 978-1-4799-2653-4.

33. FIELDING, Roy Thomas; TAYLOR, Richard N. *Architectural styles and the design of network-based software architectures*. University of California, Irvine Irvine, 2000. ISBN 0599871180. AAI9980887.

34. *What is REST?* [Codecademy] [online] [cit. 2021-02-04]. Dostupné z: https://www.codecademy.com/articles/what-is-rest.

35. *What is CRUD?* [Codecademy] [online] [cit. 2021-02-04]. Dostupné z: https://www.codecademy.com/articles/what-is-crud.

36. *HTTP Requests* [Codecademy] [online] [cit. 2021-02-04]. Dostupné z: https://www.codecademy.com/articles/http-requests.

37. FIELDING, Roy; RESCHKE, Julian. *Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content* [online] [cit. 2021-02-09]. Dostupné z: https://tools.ietf.org/html/rfc7231#section-8.1.3.

38. FREEMAN, Jonathan. *What is an API? Application programming interfaces explained* [InfoWorld] [online]. 2019-08-08 [cit. 2021-02-04]. Dostupné z: https://www.infoworld.com/article/3269878/what-is-an-api-application-programming-interfaces-explained.html.

39. *HTTP Methods – REST API Verbs* [REST API Tutorial] [online]. 2018-05-24 [cit. 2021-02-04]. Dostupné z: https://restfulapi.net/http-methods/.

40. YILDIRIM, Mehmet Emin. *Learn The Basics of RESTful APIs* [Medium] [online]. 2020-02-13 [cit. 2021-02-09]. Dostupné z: https://medium.com/@metyildirim/learn-the-basics-of-restful-apis-b53e5d157c76.

41. *JSON* [online] [cit. 2021-02-09]. Dostupné z: https://www.json.org/json-en.html.

42. *What is NoSQL? NoSQL Databases Explained* [MongoDB] [online] [cit. 2021-02-09]. Dostupné z: https://www.mongodb.com/nosql-explained.

43. L.SCHAEFER. *What Is MongoDB?* [MongoDB] [online] [cit. 2021-02-09]. Dostupné z: https://www.mongodb.com/what-is-mongodb.

44. SVADBOVÁ, B. *Česká elektronická knihovna – poezie 19. století* [online] [cit. 2021-02-28]. Dostupné z: https://www.scienceworld.cz/clovek/ceska-elektronicka-knihovna-poezie-19-stoleti-1752/.

45. SUBRAMANIAN, Naryanan. Requirements Specification and Analysis. In: *Wiley Encyclopedia of Computer Science and Engineering*. American Cancer Society, 2008, s. 1–10. ISBN 9780470050118. Dostupné z DOI: https://doi.org/10.1002/9780470050118.ecse355.

46. *Quality of Service Requirements (Sun Java Enterprise System Deployment Planning Guide)* [online] [cit. 2021-02-28]. Dostupné z: https://docs.oracle.com/cd/E19636-01/819-2326/gaxqg/index.html.

47. *What's the difference between wireframes and prototypes?* [Online] [cit. 2021-03-24]. Dostupné z: https://www.justinmind.com/blog/whats-the-difference-between-wireframes-and-prototypes/.

48. *Free prototyping tool for web & mobile apps - Justinmind* [online] [cit. 2021-05-11]. Dostupné z: https://www.justinmind.com/.

49. LINDA. *Basic UI/UX Design Concept Difference Between Wireframe, Prototype, and Mockup* [Medium] [online]. 2017-11-17 [cit. 2021-03-25]. Dostupné z: https://medium.com/@linda1858231/basic-ui-ux-design-concept-difference-between-wireframe-prototype-and-mockup-8611d9824279.

50. TAN, Wei-siong; LIU, Dahai; BISHU, Ram. Web evaluation: Heuristic evaluation vs. user testing. *International Journal of Industrial Ergonomics*. 2009, roč. 39, č. 4, s. 621–627. ISSN 0169-8141. Dostupné z DOI: https://doi.org/10.1016/j.ergon.2008.02.012.

51. EXPERIENCE, World Leaders in Research-Based User. *How Many Test Users in a Usability Study?* [Nielsen Norman Group]. Dostupné také z: https://www.nngroup.com/articles/how-many-test-users/.

52. *MarkdownPad - The Markdown Editor for Windows* [online] [cit. 2021-05-11]. Dostupné z: http://markdownpad.com/.

Příloha A

Seznam použitých zkratek

AI Artificial Intelligence

AJAX Asynchronous JavaScript and XML

API Application Programming Interface

AV Akademie věd

CRUD Create, Read, Update, Delete

CSS Cascading Style Sheets

ČVUT České vysoké učení technické

DHTML Dynamic HyperText Markup Language

DOM Document Object Model

FIT Fakulta informačních technologií

FP Funkční požadavek

GUI Graphical User Interface

HTML Hypertext Markup Language

HTTP Hypertext Transfer Protocol

HTTPS Hypertext Transfer Protocol Secure

JS JavaScript

JSON JavaScript Object Notation

NP Nefunkční požadavek

REST Representational State Transfer

SOAP Simple Object Access Protocol

SQL Structured Query Language

TAČR Technologická agentura České republiky

TCP Transmission Control Protocol

TS Testovací scénář

UC Use Case

UI User Interface

UML Unified Modeling Language

URI Uniform Resource Identifier

URL Uniform Resource Locator

VDOM Virtual Document Object Model

XML Extensible markup language

Příloha **B**

Obsah přiložené SD karty

- `readme.txt` ..... stručný popis obsahu SD karty
- `src`
  - `frontend` ..... zdrojové kódy klientské část
  - `latex` ..... zdrojová forma práce ve formátu LaTeX
  - `model` ..... model aplikace popsaný diagramem
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF
  - `doc` ..... dokumentace klientské části