FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání diplomové práce

Název:	Portál České elektronické knihovny pro Ústav české literatury
Student:	Bc. Filip Hladej
Vedoucí:	Ing. Michal Valenta, Ph.D.
Studijní program:	Informatika
Obor / specializace:	Webové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	do konce letního semestru 2023/2024

Pokyny pro vypracování

Práce navazuje na výsledky závěrečných prací Tomáše Chvosty [1] a Fllipa Hladeje [2] a integruje další výsledky projektu podpořeného v rámci soutěže TAČR ETA (2021-2023) "Analýza motivických klastrů z oblasti aktuálních kulturně-společenských témat a jejich aplikace na materiál uměleckých textů 19. a počátku 20. století".

Postupujte v těchto krocích:
1. Shromážděte a zkompletujte požadavky na uživatelské role v portálu a funkcionalitu poskytovanou jednotlivým rolím.
2. Na základě požadavků navrhněte vizuální podobu portálu ve formě wireframes.
3. Proveďte uživatelské testování a případně opravte návrh dle výsledků testování.
4. S využitím technologií použitých ve výše zmíněných ZP (MongoDB, NodeJS, ReactJS) implementujte portál. Implementace obnáší zejména:
 - rozšíření REST API backend o funkcionalitu poskytovanou týmem pracujícím na výše zmíněném projektu (našeptávání s podporou funkcí AI, full-textové vyhledávání, ...),
 - spolupráce na rozšíření datového modelu portálu a odpovídajících služeb backendu,
 - kompletní realizace frontend a zapracování grafického návrhu dodaného profesionálním grafikem.
5. Řešení řádně zdokumentujte a vhodně otestujte.

Literatura:
[1] Chvosta Tomáš: Návrh a implementace backend pro projekt Česká elektronická knihovna. Magisterská práce ČVUT FIT. 2021.

Elektronicky schválil/a Ing. Jaroslav Kuchař, Ph.D. dne 2. ledna 2023 v Praze.

Diplomová práce

PORTÁL ČESKÉ ELEKTRONICKÉ KNIHOVNY PRO ÚSTAV ČESKÉ LITERATURY

Bc. Filip Hladej

Fakulta informačních technologií
Katedra softwarového inženýrství
Vedoucí: Ing. Michal Valenta, Ph.D.
17. dubna 2023

Odkaz na tuto práci: Hladej Filip. *Portál České elektronické knihovny pro Ústav české literatury.* Diplomová práce. České vysoké učení technické v Praze, Fakulta informačních technologií, 2023.

Obsah

Poděkování vii

Prohlášení viii

Abstrakt ix

Seznam zkratek x

1 Úvod 1
1.1 Cíl práce . . . 2

2 Teoretická část 3
2.1 Frontend . . . 3
2.1.1 Design . . . 4
2.1.2 Grafické uživatelské rozhraní . . . 4
2.1.3 Heuristiky Jakoba Nielsena . . . 4
2.1.4 Webový prohlížeč . . . 5
2.1.5 Webová stránka vs. Webová aplikace . . . 6
2.1.6 HTML . . . 6
2.1.7 CSS . . . 6
2.1.8 Responzivita . . . 8
2.1.9 JavaScript . . . 9
2.1.10 Single-page aplikace . . . 10
2.1.11 Framework vs. knihovna . . . 11
2.1.12 ReactJS . . . 11
2.2 Backend . . . 13
2.2.1 NodeJS . . . 14
2.2.2 NOSQL databáze . . . 15
2.2.3 MongoDB . . . 15
2.2.4 API . . . 16
2.2.5 HTTP . . . 16
2.2.6 REST . . . 17
2.2.7 Endpoint . . . 17
2.2.8 REST API . . . 17
2.2.9 XML . . . 18
2.2.10 JSON . . . 18
2.2.11 Autentizace . . . 19
2.2.12 Autorizace . . . 20
2.2.13 JWT Token . . . 21
2.2.14 Cookies . . . 21
2.2.15 HTTP Session . . . 21
2.2.16 Local Storage . . . 22
2.2.17 Mikroslužby . . . 22

3 Praktická část **25**
3.1 Analýza . . . 25
3.1.1 Analýza požadavků . . . 25
3.1.2 Analýza stávajícího řešení . . . 28
3.1.3 Srovnání . . . 29
3.1.4 Zhodnocení . . . 29
3.2 Návrh . . . 30
3.2.1 Persony . . . 30
3.2.2 Uživatelské cíle . . . 32
3.2.3 Případy užití . . . 32
3.2.4 Task list . . . 34
3.2.5 Task graph . . . 35
3.2.6 Wireframe . . . 36
3.2.7 LOFI . . . 36
3.2.8 Prototyp . . . 41
3.2.9 HIFI . . . 41
3.2.10 Uživatelské testování . . . 43
3.2.11 Zhodnocení testování a změny v návrhu . . . 44
3.2.12 Komunikace s Ústavem pro českou literaturu . . . 45
3.3 Implementace . . . 46
3.3.1 Architektura backendu . . . 46
3.3.2 Architektura frontendu . . . 47
3.3.3 Ukázka aplikace . . . 48
3.4 Testování . . . 53
3.4.1 Scénáře . . . 53
3.4.2 Výsledek testování . . . 54
3.5 Nasazení . . . 54
3.6 Dokumentace . . . 54

4 Závěr **57**

A Wireframy **59**

Obsah přiložené SD karty **65**

Seznam obrázků

2.1 Frontend . . . 3
2.2 Backend . . . 13
2.3 Event loop . . . 14
2.4 Diagram JSON notace páru stringu a hodnoty . . . 19
2.5 Autentizace a Autorizace . . . 20
2.6 Monolit a mikroslužba . . . 23

3.1 Diagram případů užití . . . 33
3.2 Vizuální reprezentace jednotlivých úkolů pomocí task graph . . . 35
3.3 Ukázka přihlášování . . . 37
3.4 Ukázka domovské stránky . . . 38
3.5 Ukázka stránky sbírek . . . 39
3.6 Ukázka detailu sbírky . . . 39
3.7 Ukázka detailu básně . . . 40
3.8 Ukázka detailu autora . . . 40
3.9 Interaktivita HIFI prototypu . . . 42
3.10 Registrační formulář . . . 48
3.11 Přihlašovací formulář . . . 49
3.12 Domovská stránka . . . 49
3.13 Stránka Knihy . . . 50
3.14 Detail sbírky . . . 50
3.15 Detail básně . . . 51
3.16 Stránka Autoři . . . 51
3.17 Detail autora . . . 52

A.1 Modální okno vyhledávání . . . 59
A.2 Varianta vyhledávače . . . 60

Seznam tabulek

2.1 Shrnutí bezpečnosti HTTP metod. . . . 17

3.1 Funkční požadavky . . . 27
3.2 Nefunkční požadavky . . . 28
3.3 Persona A . . . 31
3.4 Persona B . . . 31

3.5 Persona C 31
3.6 Testovací scénáře 43
3.7 Otázky na uživatele 43
3.8 Odpovědi testerů 44
3.9 Endpointy ke zdroji Kniha 46
3.10 Testovací scénáře 53

Chtěl bych poděkovat především vedoucímu mé diplomové práce Ing. Michalu Valentovi, Ph.D. Dále bych rád poděkoval Ing. Karlu Kloudovi, Ph.D., který mi pomáhal s komunikací se zadavateli. Velký dík patří také mé rodině za podporu, jež se mi dostávalo během doby ztrávené prací na tomto projektu. Závěrem bych chtěl zmínit také své přátelé, kteří mě vždy svými výkony ženou kupředu, a to i díky nim tu dnes mohu psát pro mne závěr této práce, děkuji.

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 17. dubna 2023 .....

Abstrakt

Tato práce se zabývá přepracováním aktuálního řešení webového portálu české elektronické knihovny pro Ústav pro českou literaturu Akademie věd České republiky. Nejprve byla provedena obšírná analýza požadavků, následně byly vytvořeny modely reprezentující vzhled a funkčnost výsledné aplikace. Tyto modely byly poté otestovány, schváleny a jejich výsledná podoba byla použita při implementaci. Nově vytvořené uživatelské rozhraní v rámci klientské části nového webového řešení bylo otestováno, nasazeno a zdokumentováno.

Klíčová slova knihy, knihovna, básně, webová aplikace, React, Material UI, frontend, JavaScript, Akademie věd

Abstract

This thesis is about a rework of the current website solution representing the Czech electronic library for the Institute for Czech Literature of Academy of Sciences of the Czech Republic. At first a large scaled analysis of the requirements has been made, after that models representing design of the website have been created. These models have been tested, approved and used during the implementation process. The brand new User Interface of the new frontend web solution has been tested, deployed and documented.

Keywords books, library, poems, web application, React, Material UI, frontend, JavaScript, The Academy of Sciences

Seznam zkratek

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
FSv	Fakulta stavební
FTVS	Fakulta tělesné výchovy a sportu
GUI	Graphical User Interface
HIFI	High fidelity
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
JS	JavaScript
JSON	JavaScript Object Notation
LOFI	Low fidelity
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

Kapitola 1

Úvod

Lidé si v dnešní době již zvykli na pohodlí, kdy mohou řešit stále více problémů jen odemknutím mobilního telefonu a spuštěním aplikace. S tímto masivním využíváním technologií se ovšem pojí i vyšší nároky na výkon a design. Výkon hraje velkou roli při rychlosti běhu aplikace, zatímco její vzhled by měl uživateli ulehčit její používání.

Důležitým pojmem dnešní doby je digitalizace. Jedná se o přenos libovolného dokumentu do podoby, která je čitelná počítačem. I tohoto procesu lidé stále více využívají. Digitální úložiště je levnější než to fyzické. Má dokonce i více místa, samozřejmě v kontextu ukládaných dat. Digitálním souborům také nehrozí, že budou ukradeny při vloupání nebo zničeny při požáru, tedy i bezpečnost je u tohoto řešení výhodou.

Ústav pro českou literaturu Akademie věd České republiky nashromáždil velké množství sbírek básní poezie konce 18. a 19. století. Před několika lety přišel také s nápadem tyto sbírky básní digitalizovat. Představou bylo vytvoření portálu, který by výzkumníkům z Ústavu pro českou literaturu umožnil provádět srovnávací studie literatury. Jako součást projektu TAČR v rámci diplomové práce Ing. Tomáše Chvosty a mé bakalářské práce ve spolupráci s Ing. Michalem Valentou, Ph.D. byla vytvořena elektronická knihovna formou webové aplikace.

Tato diplomová práce bude pod vedením Ing. Michal Valenta, Ph.D. a v rámci projektu TAČR navazovat na stávající webové řešení, které na základě definovaných požadavků přetvoří aplikaci do finální podoby vyhovující Ústavu pro Českou literaturu.

Hlavní motivací pro práci na tomto projektu je důsledné navázání na výsledky mé bakalářské práce. Cílem této práce je vytvořit užitečný a praktický produkt, který bude skutečně využíván. Zároveň mám zájem o rozšíření svých znalostí v oblasti tvorby moderních webových aplikací pomocí současných technologií.

Práce je rozdělena na dva hlavní celky, které jsou mezi sebou velice úzce spjaty. Nejprve se práce zabývá teoretickou částí, kde jsou rozebrány a diskutovány s touto prací související technologie a pojmy, již se využívá při tvorbě aplikace samotné. Jedná se od popisu základních pojmů jako jsou webová a klientská část až po detailní rozebrání procesu autentizace či autorizace pomocí moderních technologií.

Druhá část práce se zpočátku zabývá analýzou. Analyzuje požadavky vytvořené na základě spolupráce s Akademií věd a jejich představou o projektu. Dále analyzuje existující řešení, které vzniklo v rámci zmíněných závěrečných prací. Následně tyto poznatky porovnává a vyvozuje závěry, na které je navázáno při tvorbě návrhu. Návrhu je dedikována kapitola, která definuje základní prvky, na kterých je postaveno uživatelské testování modelu aplikace. Toto testování pomáhá hlubšímu porozumění požadavků a stanovení vhodnějšího výchozího bodu pro implementaci. Výsledkem této kapitoly je jasná představa o funkčnosti a designu aplikace. Implementační část popisuje průběh implementace založený na návrhu. Dále se zabývá popisem architektury serverové a klientské části. Závěrem pak prezentuje ukázky aplikace. V neposlední řadě se praktická část zabývá testováním výsledného produktu pomocí vytvořených uživatelských scénářů. Závěrem druhé části práce je popsán způsob nasazení projektu do provozu a vytvoření dokumentace, která má za cíl pomoci administrátorům aplikaci spravovat.

Výsledná elektronická knihovna formou webové aplikace bude sloužit Ústavu pro českou literaturu Akademie věd České republiky jako vhodné úložiště a správa dat. Bude také sloužit široké odborné i laické veřejnosti včetně studentů středních škol.

1.1 Cíl práce

Hlavním cílem je vytvořit webovou aplikaci reprezentující elektronickou knihovnu sbírek básní poezie konce 18. a 19. století pro Ústavu pro českou literaturu Akademie věd České republiky. Nejprve je potřeba analyzovat stávající řešení a pomocí konzultací s Ústavem věd dojít k jednotné představě o výsledné podobě aplikace. Dále bude nutné nastudovat a popsat všechny související pojmy. Poté vytvořit návrh pomocí kterého se aplikace implementuje a otestuje. Finální bodem je nasazení aplikace do provozu a vytvoření dokumentace usnadňující práci administrátorům i případně dalším vývojářům, kteří na projekt navážou.

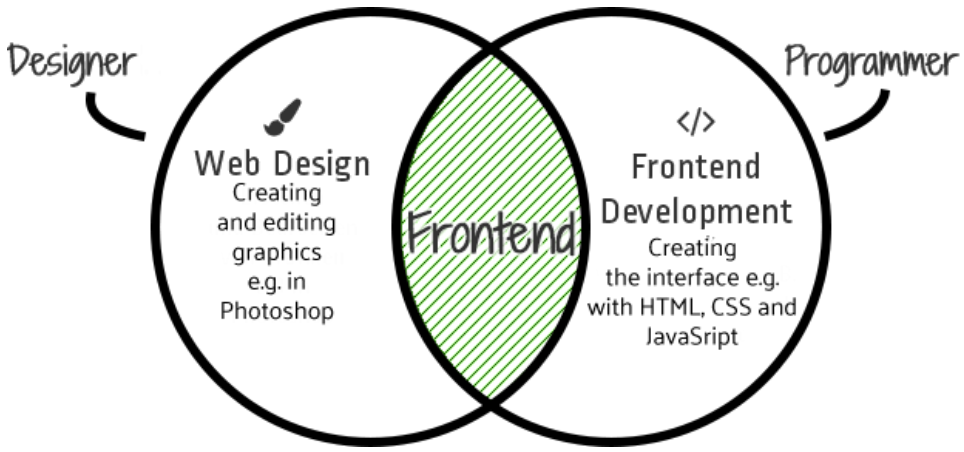
Kapitola 2

Teoretická část

Teoretická část je rozdělena na dvě kapitoly. V první kapitole jsou uvedeny a popsány technologie a pojmy vztahující se ke klientské části webové aplikace. Druhá kapitola se zabývá serverovou částí. Diskutované pojmy teoretické části jsou důležité pro pochopení aplikace a její tvorbu.

2.1 Frontend

Klientská část nebo také frontend [1] je část aplikace, kterou uživatel vidí a může s ní interagovat. Jedná se o reprezentační vrstvu webu. Je hlavní složkou grafického uživatelského rozhraní (viz 2.1.2). Jedná se o veškerý obsah a styly, zkrátka o ty části, které lze vnímat pomocí smyslů. Reprezentuje něčí představu o designu (viz 2.1.1), kterou je typicky potřeba uvést tak, aby byla na pohled zajímavá a snadno zapamatovatelná, neboť frontend je první věc, kterou uživatel na webu uvidí. Uživatel by měl mít při tomto prvním pohledu na klientskou část jasnou představu o tom, jak aplikaci používat. Tuto vlastnost systému lze také popsat jako jednoduchost a jednu z 10 heuristik Jakoba Nielsena (viz 2.1.3) pojednávající o základních principech, které by design libovolné aplikace měl splňovat pro zajištění použitelného uživatelského rozhraní.



Obrázek 2.1 Frontend [2]

Prvním krokem při tvorbě frontendu je analýza požadavků dle stanovených cílů a popis toho, co by aplikace dělat měla a co by dělat neměla. Tuto práci má typicky na starost návrhář. Druhou částí této tvorby je implementace uživatelského rozhraní odpovídající designu návrháře. Tuto část má na starosti programátor. Frontend je tedy kombinace návrhu vzhledu webu nebo také jeho designu a jeho implementace pomocí jazyků tuto tvorbu umožňujících, jak je i znázorněno na obrázku 2.1.

Mezi základní jazyky, které programátor využívá při tvorbě uživatelské části webové aplikace patří HTML (viz 2.1.6), CSS (viz 2.1.7) a JavaScript (viz 2.1.9).

2.1.1 Design

Princip tvorby designu [3] je představa a následné plánování tvorby interaktivního systému. V kontextu webových aplikací je hlavním cílem při návrhu designu myšlenka ulehčení práce s uživatelským rozhraním řešící problém uživatele. Design není jen o tom, jak věci vypadají a jak se s nimi pracuje, ale také jak by měly fungovat. Kvalitního designu je dosaženo ve chvíli, kdy si uživatel ani neuvědomuje jeho přítomnost, tedy ve chvíli, kdy je práce s rozhraním pro uživatele přirozená a vnímá její jednoduchost skrze definované objekty či barvy.

2.1.2 Grafické uživatelské rozhraní

Grafické uživatelské rozhraní [4] nebo také GUI je přirozenou součástí většiny počítačových systémů. Vývoj GUI je iterativní proces orientovaný na potřeby uživatele, který se může měnit v závislosti na jeho aktivitách. Je esenciální složkou uživatelské části webové aplikace mající za cíl podchytit uživatelské cíle. Jedná se o most mezi uživatelem a systémem, neboť uživatel často chápe aplikaci jen v kontextu GUI, protože práci systému na pozadí nevidí.

2.1.3 Heuristiky Jakoba Nielsena

Jakob Nielsen se snažil poukázat [5] na to, že design není jen záležitost hezkého GUI a je potřeba si uvědomit, že musí být skutečně navržen tak, aby také efektivně řešil problémy uživatele:

„Even the best designers produce successful products only if their designs solve the right problems. A wonderful interface to the wrong features will fail.” – Jakob Nielsen

Dle [6] existuje 10 heuristik, které každé uživatelské rozhraní musí splňovat, aby se dalo považovat za dobře fungující. Těchto 10 heuristik je dokonce popisováno jako pravidla tvorby UI. Při splnění těchto pravidel by mělo být UI uživatelsky přívětivé, intuitivní a jednoduché pro práci. Často jsou využívány k testování použitelnosti, a to jak u webových stránek, tak i u mobilních aplikací či jiných zařízení.

- Viditelnost stavu systému - Systém musí uživateli dávat signály, co se právě děje a reagovat na jeho požadavky.
- Shoda mezi systémem a reálným světem - Systém by měl nabízet přirozené logické značení a informativní sdělení, které uživatel zná z reálného světa.

- Uživatelská kontrola a svoboda - Uživatel by se nikdy neměl dostat do slepé uličky, vždy by měl být schopen najít cestu zpět, respektive vzít zpět určitá rozhodnutí.
- Konzistence a standardy - Uživatel by neměl být zahlcen složitostí nekonzistence, kde se například stejné ikony nebo názvy sekcí používají na více místech, které spolu ani nesouvisí.
- Předcházet chybám - Systém by měl být navržen tak, že místo nutnosti zobrazování chybových hlášek, uživateli nedovolí prostor pro chyby, jako je třeba vypnutí možnosti kliknutí na tlačítko, dokud nejsou splněné požadavky pro provedení takové akce.
- Rozpoznání spíš než vzpomínání - Uživatel by neměl být nucen pamatovat si, kudy se dá dostat do jisté části systému, místo toho by systém měl být navržen tak, že jej tam vždy právě díky své jednoduchosti a intuitivnosti dovede.
- Flexibilita a efektivita používání - Systém by měl uživateli poskytnout efektivní způsoby práce, může se jednat o různé zkratky nebo případná nastavení systému dle uživatelských preferencí.
- Estetika a minimalistický design - Systém by měl být jednoduchý a nabízet a umožňovat cesty skutečně jen do té míry, do které je uživatel využije.
- Pomoc uživatelům rozpoznat, porozumět a zotavit se z chyb - Systém by měl být navržen tak, že pokud se uživatel přece jen chybě nevyhne, bude mu naznačeno, jak se z takové chyby dostat, jako je tomu například u registrace, kde systém žádá jedno velké písmeno a číslici pro zadání hesla a při nesplnění této podmínky to dá uživateli jasně najevo.
- Nápověda a dokumentace - Důležitým prvkem je i dokumentace, která uživateli pomůže v případě, že mu nějaký koncept systému není jasný a potřebuje si jej detailně nastudovat.

2.1.4 Webový prohlížeč

Jedná se o aplikaci [7] využívající se k přístupu na web. Uživatel chce přistoupit na nějakou webovou stánku, prohlížeč dostává data z webové serveru a zobrazuje příslušný obsah uživateli. Takto probíhá typické využívání webového prohlížeče. Cílem prohlížeče je tedy vytáhnutí dat z webových stránek a jejich prezentace uživateli. Komunikace prohlížeče se serverem probíhá za pomocí protokolu HTTP. Data zase typicky přijímá ve formátu HTML. Prohlížeč data, která přichází z různých webů i ukládá. Těmto datům se říká cookies.

V dnešní době již webové prohlížeče nabízí celou řadu funkcí. Může se jednat o možnost spuštění soukromého módu, automatické ukládání historie prohlížení, využití záložek či různá další rozšíření a doplňky, pro které se vžil anglický termín „addons“. Nabízí také zpětné tlačítka nebo zkratky, které mohou zefektivnit práci uživatele. Existuje ovšem celá řada různých prohlížečů pro desktop nebo mobilní využití.

Je důležité si uvědomit, že skrze prohlížeč jsou realizovány veškeré přenosy dat, řešení autentifikací, ukládání dat v podobě cookies nebo local storage. Prohlížeč je v podstatě reprezentace uživatele samotného při komunikaci ze serverem. Prohlížeči se také někdy říká User-agent.

2.1.5 Webová stránka vs. Webová aplikace

Velmi často zaměňovanými pojmy, které ovšem ve skutečnosti nenesou stejné významy, jsou webová stránka a webová aplikace. Zdánlivě reprezentují stejnou věc, ale existují rozdíly v principech, které zastupují a k čemu se využívají.

Jako webovou stránku lze označit [8] skupinu stránek, které se všechny nacházejí pod jednou adresou. Pomocí této adresy k ní lze přistoupit s využitím webového prohlížeče. Je lehce nalezitelná pomocí vyhledávacích systémů jako je třeba Google. Má jednoduchou funkci. Není interaktivní, často slouží k prezentaci statického obsahu. Takový typ obsahu typicky nemusí být za přihlašovací bránou. Autentizace tedy nebývá častou součástí webové stránky. Konkrétně se může jednat například o obsah sdělující důležité informace zákazníkům nebo portfolio.

Webová aplikace je naproti tomu škálovatelný systém, který je typicky určen pro více zařízení. Často bývá rozdělen na serverovou a klientskou část. Jeho cílem je zaměření se na interakci s konkrétními cílovými uživateli. Jeho funkce bývá oproti webové stránce komplexnější. Je snadno udržitelný, neboť používá stejný kód jak pro mobilní, tak pro desktop verze. Naproti webové stránce často vyžaduje autentizaci.

2.1.6 HTML

Hypertext Markup Language [9] nebo také HTML je základní jazyk při tvorbě frontendu definující strukturu webu. Jedná se o strukturu nadpisů, seznamů, tabulek a textu. Umožňuje využívat hypertextové linky, tlačítka a tvorbu formulářů. Pro vyšší míru uživatelské přívětivosti se typicky kombinuje s jazykem CSS (viz 2.1.6).

Vhodné modularity webové aplikace lze dosáhnout pomocí HTML elementů. Ty říkají prohlížeči, jakým způsobem se má obsah na webu zobrazit. Elementy jsou vymezeny pomocí kombinace počáteční a ukončující značky. Součástí HTML elementu jsou také atributy. Ty mají 2 části. První je jméno, které slouží jako označení typu atributu. Druhou částí je samotná hodnota atributu. Tímto způsobem lze vkládat například inline CSS do HTML elementu.

HTML není dynamické, proto se nepovažuje za programovací jazyk. Funkcionalita je čistě statická. Je však považován za webový standard. World Wide Web Consortium nebo také W3C se stará o udržování HTML a jeho neustálé zlepšování. Proto také možnosti, které nabízel roku 1992, jsou zcela odlišné od těch, které nabízí od roku 2014 jeho prozatimní nejnovější verze HTML5. I tato verze prošla sérií aktualizací a nutno podotknout, že se již nějakou dobu mluví o jejím nástupci HTML6 [10].

2.1.7 CSS

Kaskádové styly [11] nebo také CSS definují, jak je web prezentován. Pomáhá definovat rozložení jednotlivých prvků a jejich barvy. Je zcela nezávislý na HTML a lze jej kombinovat i s XML. Nezávislost HTML a CSS umožňuju rozdělovat vrstvy struktury webu od jeho prezentace. CSS

umožňuje prezentovat web pro různé typy zařízení, tedy pro různou velikost okna. Této vlastnosti se říká responzivita a pomocí kaskádových stylů jí lze dosáhnout využitím media queries.

CSS je pravidly řízený jazyk [12]. Uživatel definuje pravidla dle kterých se mají aplikovat příslušné styly. Pravidly lze rozumět za jakých okolností by se které styly měly aplikovat a kde by k aplikaci těchto stylů mělo dojít. Při psaní CSS se vždy nejprve zvolí element nebo také selektor, na který mají být příslušné styly aplikovány. Samotné deklarace popisující styly se potom píší do složených závorek. Deklarace těchto stylů jsou typicky realizovány párem vlastnosti (ang. property) a hodnoty (ang. value). Jednoduchým příkladem může být změna barvy u hlavního nadpisu dokumentu na zelenou. Selektorem bude HTML element `<h1>`, vlastností bude `color` a její hodnotou bude `green`. Výsledné pravidlo poté vypadá následovně: `h1 {color: red;font-size: 5em;}`.

Flexbox a Grid

Existují dva způsoby, jak docílit sofistikované rozložení prvků na stránce. Jedná se o Flexible Box Module nebo také Flexbox a CSS Grid Layout nebo také Grid.

Flexbox [13] nabízí kvalitní manipulaci z rozprostřením prostoru mezi prvky v jedné dimenzi. Manipulací v jedné dimenzi je myšlena úprava rozložení prvků v řádcích nebo ve sloupcích. K tomuto účelu Flexbox definuje konstrukty jako je hlavní a křížící osa (ang. main axis and cross axis). Pomocí hlavní osy umisťuje prvky dle výběru v řádku či sloupci. Pomocí křížící osy, která je kolmá na osu hlavní zase určuje opačný směr. Pokud tedy pomocí hlavní osy programátor definuje rozložení prvků v řádcích, potom pomocí křížící osy definuje toto rozložení ve sloupcích. Dohromady tedy tvoří dvoudimenzionální prostor, který lze upravovat dle potřeby. Vždy je ale možné takto upravovat dimenze pouze po jedné, proto je Flexbox jednodimenzionální.

Druhým případem je Grid. Ten umožňuje rozprostření prvků v prostoru do řádků a sloupců, stejně jako je tomu u tabulek. Oproti tabulkám však nabízí [14] propracovanější způsoby návrhu rozložení a to vše velice jednoduše díky vestavěným konstruktům. Grid dokáže pracovat se dvěmi dimenzemi najednou, čímž se liší oproti Flexboxu. Ne vždy je ale tato vlastnost výhodná, takže použití těchto dvou způsobů rozložení prvků se liší dle situace.

CSS Framework

CSS Framework [15] obsahuje předem připravené CSS konstrukty a komponenty, které lze využívat jako pomocné prvky při tvorbě GUI. Často se CSS frameworky využívají ke zrychlení práce v prvotních fázích vývoje a není tak potřeba začínat úplně od nuly. Lze tak vytvářet položky jako jsou navigační menu, zápatí nebo různé posuvníky velice rychle a efektivně. Tento způsob práce šetří spoustu času, ale také dává projektu koherentní tvář. Některé frameworky nabízí vylepšení v podobě funkcí založených na JS. Hlavně však nabízí standardy jednotného stylu mezi které patří typografie, fonty, ikony, tlačítka, formuláře a samozřejmě ovládání rozložení prostoru pomocí gridových vlastností.

Často využívanými CSS frameworky jsou Bootstrap, TailwindCSS a Semantic UI. V rámci knihovny ReactJS se však často využívají CSS knihovny MaterialUI, ChakraUI a React Bootstrap.

2.1.8 Responzivita

Moderní způsoby tvorby webových aplikací umožňují tvorbu aplikace přizpůsobující se velikosti okna dle právě používaného zařízení. Výsledek je uživatelsky přívětivý a v dnešní době takřka nepostradatelný. Tomuto způsobu stylování HTML se říká responzivita. Dle Ethan Marcotte lze responstivní design rozdělit [16] do 3 částí:

- Rozložení stránky by mělo být pohyblivé a umožňovat přizpůsobovat se maximální velikosti prohlížeče.
- Obrázky by se měly umět flexibilně hýbat a měnit v závislosti na rozložení stránky.
- Pro potřeby změny rozložení obsahu stránky při zobrazování pro různé typy zařízení a naplnění uživatelské přívětivosti by se měly používat media queries (viz 2.1.8).

Takovéto flexibility se v praxi dosahuje definováním velikostí pomocí relativních jednotek. Lze používat procenta, které jsou vhodné pro definici velikosti písma. Velikostí elementu je dále v této kapitole na mysli velikost jeho typografie. Pro úpravu velikosti dle rodičovského elementu se používají jednotky `em`. Pro stejný efekt dle velikosti celého dokumentu zase jednotky `rem`. Velice užitečné jsou jednotky `viewport width` nebo také `vw` a `viewport height` nebo také `vh`. Tyto jednotky umožňují úpravu elementů relativně dle velikosti okna.

Klasické definování velikostí pomocí pixelů nebo také `px` je v dnešní době využitelné už jen pro definování velikostí specifických věcí, jako třeba rámečky. Problém s pixely totiž nastává už ve chvíli, kdy si uživatel může měnit velikost písma na webu dle vlastní libosti. Stránka takto definovaná v pixelech může pro dané rozlišení s výchozí velikostí písma splňovat žádaný efekt. Při změně velikosti písma však může dojít ke kompletnímu chaosu, neboť jednotka pixel není relativní a tak není schopna svou velikost relativně upravovat. Výsledkem tak může být stránka, která již nesplňuje ani základní požadavky, aby se s ní vůbec dalo pracovat.

Media Queries

Media Queries jsou [16] základní metodou umožňující aplikovat rozdílné CSS styly v závislosti na různém technického kontextu. Jsou rozděleny na dvě části. První části je typ média. Ten se typicky definuje jako `@media screen`. Druhou částí je podmínka, která aplikuje příslušející styly při jejím splnění. Ta se definuje například jako `(max-width: 60em)`. Výsledkem je potom kompletní sekce `@media screen and (max-width: 60em) {}`, kde se mezi složené závorky píší příslušné CSS styly, které si pro tento případ přejeme aplikovat. V tomto příkladě je tedy řečeno, že pokud platí, že maximální šířka prekročila velikost 60 `emů`, tak je potřeba aplikovat příslušné styly.

Obsahuje také logické operátory `and`, `not` a `only`. Toho lze využít k definici komplexnějších media queries, které mohou mít větší množství podmínek pro aplikace stylů. Další možností je zřetězení více media queries pomocí čárky. Styly lze poté ve větším množství situací.

Media Queries jsou v dnešní době již doporučovaným standardem v tvorbě webových aplikací. Jedná se o základ tvorby responzivního webu [16].

2.1.9 JavaScript

JavaScript [17] nebo také JS je skriptovací, netypovaný, objektově orientovaný a multiplatformní jazyk, který je řízený událostmi a umožňuje vytvářet dynamicky se aktualizující obsah.

Pro správný běh programu napsaného v jazyku JS je potřeba jej převést do binárního kódu, který je přímo spustitelný na počítači. Převod vyšších programovacích jazyků do binárního kódu se provádí buď pomocí tzv. interpreteru, což je obecně pomalejší, ale dynamičtější přístup, anebo pomocí překladače. JS patří do skupiny programovacích jazyků, které používají interpreter. Z toho důvodu není JS jazyk kompilovaný, ale interpretovaný. JS nabízí práci s již definovanými objekty jako je například `Array`. Tyto objekty mají řadu vlastností a funkcí, které se nad nimi dají volat. Díky těmto objektům a jejich funkcím nabízí JS okamžité využití jednoduchý funkcionalit oproti ostatním jazykům, kde si je uživatel musí definovat sám. Na druhou stranu tyto definice mohou být mnohem efektivnějším kódem než jsou již nabízené funkce JS.

JS je zodpovědný za veškeré pohyblivé prvky na stránce. Jedná se o animované prvky jako je například dynamicky se měnící pozadí. Umožňuje také přehrávat audio či video záznamy. Lze jej využít ke stahování obsahu z jiných stránek. Často bývá využíván také ke tvorbě serverové části webové aplikace pomocí Node.js.

JS se při tvorbě webových aplikací často využívá jako součást HTML. Tento způsob je velice jednoduchý, ovšem plynou z něho jistá bezpečností omezení. Neboť se tento kód spouští až po načtení stránky na straně klienta, což je přesně opačně než je tomu například u jazyka PHP, tak jeho případná práce se soubory ohrožuje soukromí uživatele. U jazyka PHP je tato práce orientovaná čistě na serverovou část a tento bezpečnostní problém je eliminován.

Existují 3 způsoby zápisu JS do HTML:

- Inline Javascript, který se vpisuje přímo k příslušnému HTML prvku

```
<button onclick="alert('Button Click')">Click me!</button>
```

- Internal Javacript, kterého lze využít pomocí `script` tagu.

```
<script> function(){alert("Some message..")} </script>
```

- External Javascript, který je součástí jiného souboru. K tomuto kódu lze přistoupit přes `src` tag.

```
<script src="./script.js"></script>
```

Může být ale také použit k vytváření škodlivých skriptů, které mohou napadnout webovou stránku nebo její uživatele. Cílem útočníků bývá otevírání zadních vrátek, vytváření neoprávněných přístupů k datům nebo ukládání škodlivého kódu. Tímto způsobem ho lze použít k získání citlivých informací uživatele. Uživatel má ovšem také možnost JavaScript vypnout, čímž by měl docílit vyššího stupně zabezpečení i vyšší rychlosti při prohlížení obsahu webu. Často nastává totiž případ, kdy se JS neustále aktualizuje a uživateli tím prohlížení webové stránky zpomaluje. Záleží ovšem na konkrétní webové stránce a nelze toto tvrzení generalizovat.

Mezi typickou zbraň útočníků na webu využívající JS patří [18] Cross-site scripting nebo také XSS a Cross-site request forgery nebo také CSRF. XSS spočívá v umístění pro uživatele škodlivého kódu jako součást webové stránky, který jej libovolnou interakcí může spustit. CSRF útok je založen na přinucení uživatele k poslání požadavku. Typickým příkladem může být obrázek, který je ve skutečnosti odkazem na bankovní převod, který tímto kliknutím může být realizován, pokud jsou stále aktivní cookies uživatele a uživatel je ke svému bankovnímu účtu přihlášen. V dnešní době však existují již vyšší stupně ochrany jak ze strany banky, tak ze strany uživatelů samotných. Může se jednat o speciální aplikace bank, které vyžadují potvrzení o platbě nebo několika faktorové ověření skrze potvrzovací kódy chodící jako sms zprávy.

Existuje velké množství JS frameworků a knihoven ulehčující práci pomocí předem vytvořených konstruktů. Mezi často používané patří AngularJS, VueJS a ReactJS.

AJAX

Asynchronous JavaScript And XML nebo také jako AJAX lze označit technologie využívající se k vývoji [19] webových stránek, které se dokáží aktualizovat bez potřeby znovunačítání. Po kliknutí na tlačítko spouštějící akci se tradičně musí webová stránka kvůli této malé změně celá znovu načíst. Tato metoda ovšem tuto neefektivitu vyřešila tím způsobem, že se na pozadí pošle na server pouze informace o těch částech stránky, které je potřeba změnit a pouze ty se aktualizují. Tuto vlastnost lze popsat jako asynchronnost a jedná se o jednu z hlavních charakteristik této technologie. AJAX tedy dokáže asynchronně nebo také na pozadí komunikovat ze serverem. Prohlížeč nejdříve načte AJAX engine a až poté webovou stránku. AJAX poté na pozadí slouží jako prostředník mezi klientem a serverem. Uživatelské akce na webové stránce jsou tradičně interpretované jako HTTP požadavku odesílané přímo na server. V tomto případě jsou však tyto požadavky poslány na AJAX engine a ten je asynchronně zpracovává, aniž by si byl uživatel vědom jeho existence.

Posílat i přijímat dokáže informace ve formátech jako je JSON, XML, HTMl nebo také obyčejný text. Často se používá v kombinaci s DHTML[1], JavaScriptem a XML. [20]

XML v názvu AJAX může svádět k myšlence, že aplikace, které AJAX využívají, posílají data pomocí XML. Tyto data jsou však často ve formátu JSON nebo jako obyčejný text.

2.1.10 Single-page aplikace

Dle [21] jsou webové stránky pomalé z velké části kvůli architektuře MVC se zaměřením na přepínání mezi stránkami pomocí nějaké akce definující link. Tento způsob akcí typicky zapříčiní

[1] DHTML - Dynamické HTML

okamžitý efekt bílé obrazovky, kde už je jasno, že se právě celá stránka znovu načítá. Jedná se o filosofii tvorby webu, kdy jsou překresleny všechny prvky na stránce včetně navigačního baru, sekcí v zápatí a dalších komponent, u kterých nedochází ve skutečnosti k žádné změně. Toto řešení je pomalé, velice zastaralé a v dnešní době je k podivu, že je ještě aplikováno. Přitom dnes existují moderní a jednoduché způsoby, jak vytvořit komplexní web, který reaguje na potřeby uživatele.

Za Single-page aplikaci nebo také SPA je označována dynamická webová aplikace, která dokáže překreslovat obsah aktuální webové stránky bez potřeby jejího znovunačítání. Cílem tvorby webu jako single-page aplikace je vysoká rychlost a reaktivnost. Knihovnami a frameworky, které korelují s touto tvorbou jsou AngularJS, VueJS a také knihovna, která je využívána v rámci této práce, ReactJS.

2.1.11 Framework vs. knihovna

Frameworky i knihovny jsou bloky kódu, které lze využívat jako součást projektu. Jedná se o sady nástrojů, které pomáhají při vývoji webových aplikací. Při používání knihovny si programátor může určit, zda využije konstrukty, které knihovna nabízí či nikoliv. Naproti tomu framework určuje konstrukci celého projektu a nelze si zvolit, že jej v jisté chvíli programátor přestane používat. Jedná se tedy o rámec nad daným jazykem.

2.1.12 ReactJS

ReactJS je [22] JavaScriptová knihovna usnadňující tvorbu uživatelského rozhraní. Často bývá využívána k vytváření mobilních i single page webových aplikací. Dokáže totiž velice efektivně pracovat s rychle se měnícími daty. React zastupuje View v softwarové architektuře Model-View-Controller nebo také MVC. Nespolupracuje přímo s modelem DOM (viz 2.1.12), který je vytvářen prohlížečem, ale s jeho částí, která je uložená v paměti. Odtud také pramení hlavní charakteristika Reactu. Spousta jiných knihoven a frameworků pracuje s celým modelem DOM a to má za následek jeho neustálé upravování kvůli každé akci provedené klientem. Kvůli tomuto důvodu jsou jednoduché operace zbytečně složité a režijně náročné. React ovšem využívá principů Virtuálního modelu DOM, kterému je věnována sekce 2.1.12 této kapitoly.

V Reactu se využívají komponenty, které lze znovu využívat na více místech v kódu. Jsou dva způsoby, jak lze komponenty vytváře. První způsob využívá třídy. Tato metoda je však již v dnešní době mezi programátory méně oblíbená. Častěji se využívá druhého způsobu pomocí funkcí. Většinou se vytváří větší množství těchto komponent, kde každá reprezentuje jeden logický celek na stránce. Komponenty Reactu lze psát v čisté podobě jazyka JavaScript anebo pomocí JSX. JSX nebo taky JavaScript XML umožňuje do Reactu přidávat HTML elementy. Umožňuje tedy konverzi HTML na React elementy.

V Reactu se využívá stavů (ang. states). Typicky existuje jedna komponenta mající takovýto stav reprezentující měnící se data. Tato komponenta potom posílá ostatním komponentám data uložená v tomto stavu. Komponenty tyto data přijímají přes tzv. props a na jejich základě vykreslují příslušný obsah. V dnešní době se však využívá již moderního přístupu, který zjednodušuje práci se stavy. Jedná se o tzv. hooks, které props nahrazují a velice usnadňují sdílení kódu napříč celou aplikací. Komponenty Reactu také implementují metodu render, která vrací to, co se má zobrazit na stránce.

DOM

Objektový model dokumentu [23] nebo také DOM je API pro HTML a XML dokumenty. Reprezentuje webovou stránku jako stromovou strukturu uloženou v paměti. Každý uzel stromu reprezentuje objekt. Díky této struktuře může programátor jednoduše manipulovat s obsahem modelu DOM. Při těchto změnách, ať už obsahu nebo struktury dokumentu, je však potřeba vždy překreslit celý DOM. Pokud je totiž upraven nějaký uzel, musí jak on, tak i všechny jeho děti projít aktualizací. Tento způsob aktualizace zbytečně aktualizuje uzly, které ovšem žádnou změnou neprošly a celý proces je tak značně zpomalen. Chytré řešení tohoto problému nabízí Virtuální Objektový model dokumentu.

VDOM

Virtuální Objektový model dokumentu [24] nebo také VDOM je virtuální reprezentace UI uložená v paměti. Tato reprezentace je v synchronizaci s klasickým modelem DOM pomocí ReactDOM. Tomuto procesu se říká smíření (ang. reconciliation). Při požadavku na změnu obsahu stránky se namísto manipulace se samotným modelem DOM vytvoří VDOM tyto změny reprezentující. Nový VDOM již obsahující požadované změny se poté porovná s modelem DOM generovaným prohlížečem a pomocí porovnávacího algoritmu provede změny v reálném modelu DOM. Změny jsou provedeny pouze na těch uzlech, u kterých byla změna provedena. Tento princip zajišťuje vysokou rychlost měnění obsahu stránky, které React nabízí. Je to hlavní důvod, proč je React považován za vysoce efektivní z hlediska rychlosti a je často upřednostňován oproti konkurenčním knihovnám a frameworkům.

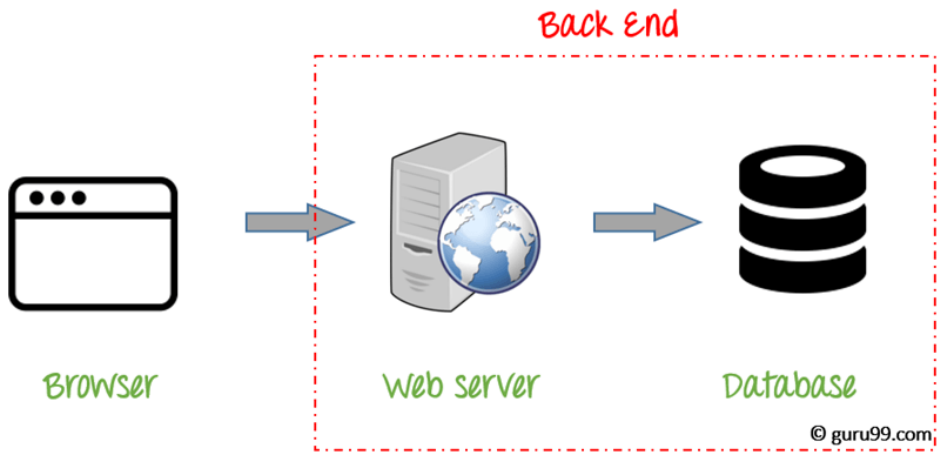
2.2 Backend

Serverová část webové aplikace nebo také backend slouží k administraci webu. Obstarává logiku aplikace a stará se o potřebné výpočty. Je to kód, který běží na straně serveru, jehož hlavním cílem je mít definovány principy, které dokáží obstarávat příchozí dotazy. Při dotazování z frontendu formou požadavku backend dotaz zpracuje za pomocí dat získaných z databáze a odesílá žádané data jako odpověď zpět na frontend (viz obrázek 2.2). Existují 3 hlavní části, které se na tomto procesu výměny informací podílejí: [25]

- Server – Jedná se o zařízení, které očekává příchod požadavků. Jakýkoliv počítač může vystupovat jako server, je-li součástí sítě.
- Databáze – Systém, který organizuje data a stará se o jejich perzistentní uložení.
- Aplikace – Aplikace, který běží přímo na serveru, očekává požadavky, pracuje s informace uloženými v databází a posílá odpovědi klientovi.

Při tvorbě backendu je třeba znát cíle aplikace a přicházet s efektivním řešením problémů. Backend se také stará o autentizaci a autorizaci uživatele. Měl by tedy zobrazovat data jen takovému uživateli, které k nim má skutečně přístup.

Běžný uživatel tyto procesy běžící na serveru nemůže zaznamenat a tak je pro něj backend narozdíl od frontendu skrytý.

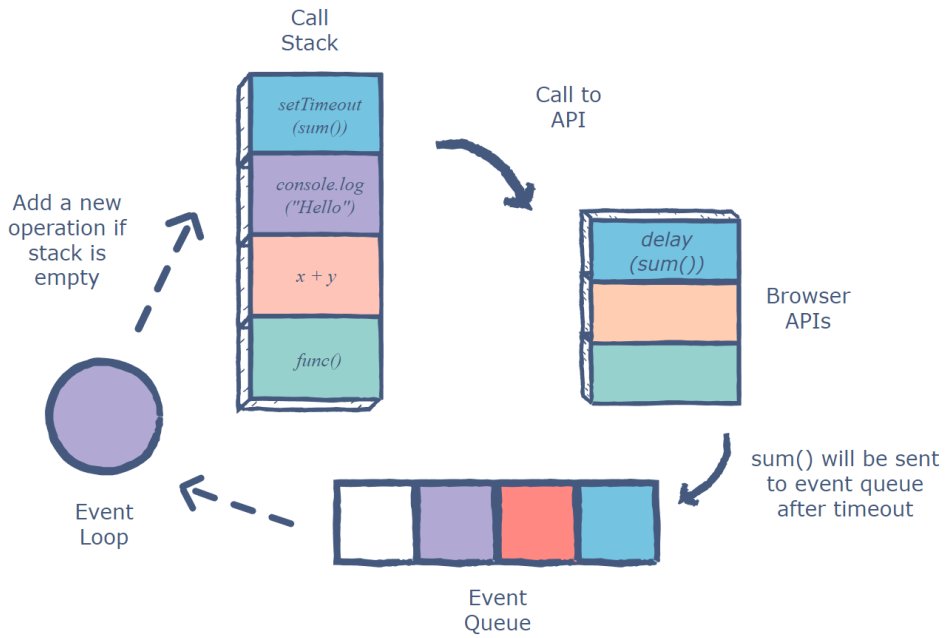


Obrázek 2.2 Backend [26]

2.2.1 NodeJS

Je to systém vhodný pro tvorbu škálovatelných systémů. Největší využití se nachází v tvorbě webových aplikací, konkrétně serverové části. NodeJS nikdy neprovádí I/O operace přímo, ale zpracování požadavků probíhá naopak asynchronně. Nepoužívá se žádných zámků, tedy nedochází k vzájemnému blokování mezi procesy, stejně jako nehrozí deadlock. Tímto způsobem lze dosáhnout vyššího výkonu než je tomu u konkurečních systémů. Smyčku událostí nebo také event loop definuje NodeJS jako konstrukt doby běhu. Tedy oproti chování jiných systémů, které event loop spouští pomocí příkazu v blokujícím čekání, jej NodeJS spustí v momentě spuštění skriptu na vstupu. [27]

Event loop je důvod, proč je JS asynchronní. Jedná se o smyčku využívající pouze jednoho vlákna, kde ovšem díky chytře definovaným strukturám lze sledovat iluzi vícevláknového běhu. Tato smyčka (viz obrázek 2.3) začíná u zásobníku volání (ang. call stack), který se stará o veškeré operace, které musí být vykonány. Při splnění určité operace, je tato operace jednoduše odebrána z tohoto zásobníku. Požadavek se zpracovává jeho posláním z call stacku na API, které jej zpracuje za využití jednoho vlákna. Odpověď poté odesílá do fronty událostí (ang. event queue). Z event queue putuje informace o splnění požadavku na event loop, který zašle informaci o splnění požadavku na call stack spolu s další operací, kterou je potřeba provést. Call stack následně splněný požadavek odebere ze zásobníku. Poté přidá nový požadavek, který právě dorazil z event loop. Tato smyčka se opakuje, dokud není call stack prázdný, tedy dokud nejsou splněny všechny požadavky. Ikdyž je NodeJS v základu navrženo jako systém bez potřeby využívání vláken, není problém je i přesto využít. Pro tento účel existují vestavěné příkazy, které jsou součástí API.



Obrázek 2.3 Event loop [28]

2.2.2 NOSQL databáze

NoSQL databáze jsou často jednou skupinou označovány jako non SQL databáze, ale přesnější označení je not only SQL. Mezi hlavní výhody patří horizontální škálování, jednoduchost a flexibilní schéma. Velmi vhodné bývá použití NoSQL databází k ukládání velkého množství dat. Z tohoto důvodu jsou ale velice náročné na množství paměti. Typicky ale úložiště není drahá záležitost a tak se nejedná o zásadní problém. Oproti relačním databázím navíc nepotřebují takový výkon. Dotazování u NoSQL databází bývá obecně rychlé, ovšem vždy záleží na datovém modelu.

Jedná se o [29] databáze nevyužívající tabulky k ukládání dat. Existuje spousta různých typů těchto databází založených na různých datových modelech. Mezi typické příklady patří Wide-column databáze, kde jsou data ukládána data do flexibilních sloupců, příkladem je Apache Cassandra. Neo4j je databází využívající grafový model, kde je jako struktura k ukládání dat a dotazování nad nimi využíván graf. Existují také key-value databáze založené na ukládání dat jako kolekci páru klíče a hodnoty, příkladem je Riak KV. Často se využívají i dokumentové databáze, které s daty pracují ve formě dokumentů. Nejpoužívanější z těchto databází je MongoDB (viz 2.2.3).

2.2.3 MongoDB

MongoDB je NoSQL, dokumentová, key-value, multiplatformní a distribuovaná databáze. Jedná se o databázi s otevřeným zdrojovým kódem [30]. Nabízí vysokou dostupnost díky kopiím dat zvaným repliky. Repliky fungují na principu, že pokud selže hlavní zdroj dat, replika jej jednoduše nahradí. Této vlastnosti se říká replikace. Nabízí také horizonální škálování díky metodě zvané sharding. Jedná se o distribuci jednoho kusu dat v dekomponované podobě mezi více strojů. Díky této technice umí Mongo v případě chyby vyvažovat zátěž. Neobsahuje žádné pevné schéma. Tato vlasnost je výhodou z hlediska nepotřeby schéma neustále udržovat. Tato vlastnost ovšem má i své nevýhody, jako třeba problémy s validací.

Je založena na dokumentovém modelu, který velmi usnadňuje práci s databází [31], neboť k ní lze přistupovat objektovým způsobem. Místo tradičního využívání tabulek, jako je tomu u SQL databází, využívá Mongo k ukládání dat dokumenty, které jsou uloženy jako součást kolekcí. Tyto dokumenty mají formát binary JSON nebo také BSON. Jedná se o serializaci formátu JSON vhodnou pro ukládání a přenos dat. Dokumenty samotné již obsahují libovolné data. Umožňují ukládat i vnořené data, které mezi sebou mohou mít složitější vztahy.

Umožňuje indexaci a agregace v reálném čase využívající MapReduce. Dotazování probíhá pomocí jazyka MongoDB Query Language nebo také MQL.

2.2.4 API

Application Programming Interface [32] nebo také API je programátorem využívaná kolekce funkcí, tříd, protokolů a procedur. Jedná se o software, který umožňuje dvoum aplikacím komunikovat mezi sebou. API je spojovací rozhraní umožňující sdílet data napříč různými aplikacemi. Application v názvu API reprezentuje software s nějakou funkcí. Interface zase mluví o spojení dvou aplikací k sobě. Samotný název je tedy velice informativní.

Mezi klasické standardy moderního API patří HTTP a REST. Existují však také SOAP API, které funguje na principu výměny zpráv pomocí XML, a také RPC API nebo Websocket API. Každý způsob tvorby API má své specifické charakteristiky.

API je lehce dostupné a zjednodušuje práci pro programátora. Obecně se jedná o koncept a ne technologii. Vytvářet API lze v různých programovacích jazycích. Často se pro tvorbu tohoto rozhraní využívá jazyků Java nebo JavaScript.

2.2.5 HTTP

Hyper Text Transfer Protocol [33] nebo také HTTP je bezestavový internetový protokol, který umožňuje komunikaci s World Wide Web servery. Komunikace mezi klientem a serverem probíhá pomocí požadavků a odpovědí. Klient posílá požadavek na server jako HTTP požadavek. Server tento požadavek přijme a zpracuje. Poté posílá odpověď zpět na klienta jako HTTP odpověď.

Součástí odpovědi je také stavový kód, který slouží jako informace, zda požadavek proběhl v pořádku nebo naopak nastala nějaká chyba. Existuje 5 rozdělení stavových kódů: [34]

- 100-199 – Informace.
- 200-299 – Úspěch.
- 300-399 – Přesměrování.
- 400-499 – Chyba na straně klienta.
- 500-599 – Chyba na straně serveru.

HTTP přenáší data různých formátů, typicky je to ovšem HTML nebo XML. Protokol samotný není šifrován a není zaručena integrita dat. Pro zabezpečenou komunikaci lze využít Hypertext Transfer Protocol Secure nebo také HTTPS. HTTPS zajišťuje integritu dat, ověření identity a zabezpečení založené na asymetrické kryptografii.

Mezi 4 základní HTTP metody REST systémy patří:

- GET – Slouží k získání dat.
- POST – Slouží k vytváření nebo aktualizaci obsahu.
- PUT – Slouží k modifikaci obsahu.
- DELETE – Slouží k mazání obsahu.

U HTTP metod existují dvě důležité vlastnosti. První vlastností je bezpečnost. Metoda je považována za bezpečnou, pokud se dá použít pouze ke čtení, tedy nelze s jejím využitím přes

API cokoliv měnit. Druhou vlastností je idempotence. Metoda je idempotentní, pokud se jejím opakovaným voláním dosáhne vždy stejného výsledku. V tabulce 2.1 jsou uvedeny HTTP metody a jejich vlastnosti bezpečnosti a idempotence.

Tabulka 2.1 Shrnutí bezpečnosti a idempotence HTTP metod [35]

HTTP metoda	Bezpečná	Idempotentní
GET	ano	ano
HEAD	ano	ano
OPTIONS	ano	ano
TRACE	ano	ano
PUT	ne	ano
DELETE	ne	ano
POST	ne	ne
PATCH	ne	ne

Ke komunikaci lze také využít XML HTTP Request nebo také XHR. ikdyž je v názvu HTTP, tak se XHR používá s jinými protokoly než HTTP. Obecně se jedná o JavaScriptový objekt pomocí kterého se posílají data mezi prohlížečem a serverem.

2.2.6 REST

Representational State Transfer [36] nebo také REST je architektura rozhraní usnadňující komunikaci na webu mezi systémy. Systémy REST jsou stateless, tedy při příchodu požadavku na server vrátí odpověď bez potřeby ukládání jakýchkoliv dat. S tím souvisí i další vlastnost RESTful systémů, a to nezávislost klienta na serveru. Proběhnou-li nějaké změny na straně klienta, nijak neovlivní fungování serveru a stejně tak to platí i opačně. Díky rozdělení datového úložiště a uživatelského rozhraní je systém flexibilní a lehce škálovatelný. Server se dá totiž lehce rozdělit na více částí, které se můžou vyvíjet samostatně. Na stejné endpointy nad jedním rozhraním REST se může dotazovat více klientů najednou a vždy obdrží stejnou odpověď pošlou-li stejný požadavek.

Systém typu REST by měl vždy umožňovat provádět CRUD[2]operace. Měl by tedy být schopný obsah vytvářet, číst, aktualizovat a mazat. Pro tyto čtyři funkčnosti je vhodné použít HTTP metody POST, GET, UPDATE a DELETE, kterým se věnuje předchozí kapitola.

2.2.7 Endpoint

Koncový bod nebo také Endpoint je bod, ve kterém dochází ke komunikaci dvou aplikací, konkrétně API a klienta. Klient se dotáže na API endpoint, API tak umožní přístup ke zdroji na serveru a poté vrátí odpověď zpět na klienta. Často existuje více než jen jeden endpoint. Každý potom bývá zodpovědný za přístup ke specifickému zdroji, tedy konkrétním datům.

2.2.8 REST API

Popisuje sadu zdrojů nebo také endpointů, ke kterým lze přistupovat skrz klienta a volat nad nimi jisté operace. Z klienta se operace volají pomocí HTTP nebo také JavaScriptu, který je

[2]CRUD - Create, Read, Update, Delete

spouštěn na webovém prohlížeči. Veškeré odkazy na zdroje jsou definovány jako relativní cesty vzhledem k té výchozí. Různé výchozí cesty lze využít pro definici cest k rozdílným REST API. Obecně lze říci, že REST API je API, které splňuje principy rozhraní definované architekturou REST.

2.2.9 XML

Značkovací jazyk je obecně mechanismus, který identifikuje strukturu nějakého dokumentu. Extensible Markup Language [37] nebo také XML je značkovací jazyk využívaný pro ukládání a posílání dat. Cílem XML je obecnost, jednoduchost a využitelnost napříč celým Internetem. Obsahuje specifické tagy, které dávají datům strukturu a obsahují metadata. Součástí dat mohou být také obrázky. Jedná se o zjednodušenou podobu dnes již tolik nepoužívaného jazyka SGML. XML se obecně nezabývá vzhledem, lze jej tedy kombinovat s CSS. Lze také využít jazyka XSLT a transformovat XML na jiný typ dokumentu. Data v XML jsou kódovány způsobem, že jsou čitelné člověkem i strojem. XML je také využíváno jako základní jazyk komunikačních protokolů jako SOAP nebo XMPP. Příklad XML formátu lze vidět na ukázce kódu číslo 1.

```
<note>
    <to>Michal</to>
    <from>Petr</from>
    <heading>Dinner</heading>
    <body>Do not forget we have a dinner with the client in the evening !</body>
</note>
```

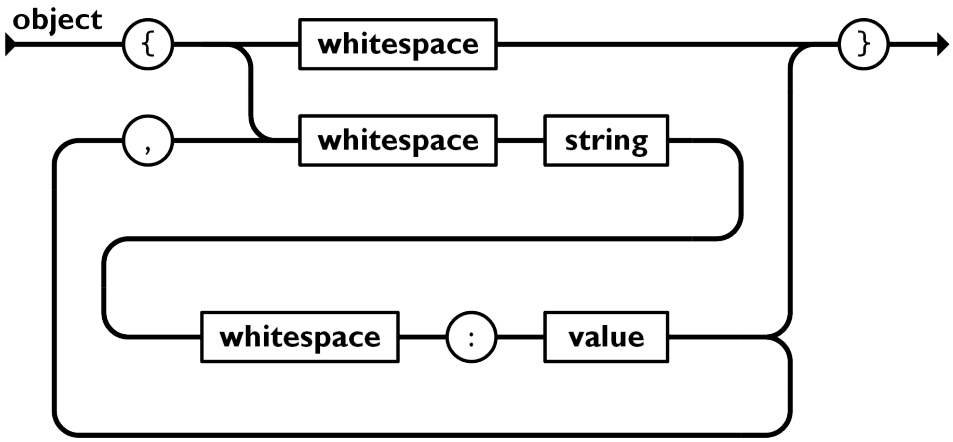
Výpis kódu 1 Ukazka kódu v XML formátu

Simple Object Access Protocol nebo také SOAP je protokol sloužící pro komunikaci na webu pomocí HTTP. Posílané zprávy mají formát XML. Posílat data ve formátu XML má své pro i proti. Jednou z již zmíněných výhod je, že se formát dá číst člověkem i strojem. U stroje je čtení v tomto případě pomalejší, neboť parsování XML je složité a zvyšuje tak nároky na výkon. XML velice dobře umí pojmout kontext přenášených dat, bohužel ale posílá z většiny spoustu tagů a atributů. Z důvodu zbytečné složitosti a režie bývá tedy často využíván jako jazyk pro ukládání a komunikaci na webu spíše JSON.

2.2.10 JSON

JavaScript Object Notation [38] nebo také JSON je datový formát používaný pro výměnu dat mezi 2 stroji. Vyskytuje se jak na straně klienta, tak i na straně serveru. Jak už název napovídá, JSON je odvozen z jazyka JavaScript. Jedná se tedy o jeho podmnožinu, která ovšem neslouží k programování. Je to textová reprezentace dat obsahující určitou strukturu definovanou dle daných pravidel. Jeho syntax je jednoduchá na psaní i čtení, a to jak pro člověka, tak pro stroj. Narozdíl od XML počítač dokáže JSON generovat i parsovat velice rychle, což z něj dělá silný nástroj pro ukládání i posílání dat. Posílaná data lze posílat v polích nebo jako součást objektů. Přenos dat ve formátu JSON bývá často realizován pomocí technologie AJAX. Tento formát je nezávislý na počítačové platformě i jazyku, ovšem používá konvence podobné rodinám jazyků C. Lze jej tedy využívat v libovolném programovacím jazyku. JSON je odlehčenější a jednodušší než XML. Lze však mezi těmito formáty libovolně převádět pomocí známých metod.

Základem JSON notace je objekt. JSON objekt je neseřazený seznam mnoha párů typu jméno a hodnota. Jeho definice vždy začíná levou složenou závorkou a končí naopak pravou složenou závorkou. Autor formátu JSON Douglas Crockford popsal reprezentaci struktury JSONu pomocí syntaktických diagramů. Na obrázku 2.4 je uvedena ukázka jednoho z těchto diagramů ilustrující reprezentaci gramatiky pro pár stringu a hodnoty.



Obrázek 2.4 Diagram JSON notace páru stringu a hodnoty [39]

2.2.11 Autentizace

Autentizace je proces ověřování hodnověrnosti entity. Jedná se o důležité bezpečnostní opatření, které pomáhá identifikovat daný subjekt a naopak chrání před jeho zfalšováním. Přihlašováním na webovou stránku pod uživatelským účtem dochází k autentizaci osoby. Posíláním zprávy jako součásti šifrované komunikace, při které je typicky využíváno veřejného a privátního klíče, zase dochází k autentizaci zprávy [40].

Namísto hesel lze v dnešní době využít zmíněného asymetrického veřejného klíče. Hesla s sebou nesou spousty problémů. Uživatelé často využívají stejné heslo pro přihlašování na různé webové stránky. Pokud by došlo k úniku dat z jiné stránky, kde se uživatel pomocí takového hesla přihlašuje, útočník jej tak může zneužít ve svůj prospěch a dostat se tak na stránku, která je cílem útoku. Tímto způsobem se také dá velice efektivně bránit proti phishingu. Útočník s neplatnou webovou stránkou umožňující přihlášení se jako uživatel nepřihlásí. Důvodem je změna podpisu dle originu. Další výhodou je omezení úniku dat díky existenci kombinace veřejného a privátního klíče. Když útočník získá veřejný klíč, stále jej nebude schopen využít k autentizaci, neboť mu chybí zmíněný privátní klíč. Tyto principy silné autentizace zastává specifikace W3C a FIDO Web Authentification API.

V kontextu REST API existuje několik pro autentizaci využívaných metod. První metoda je autentizace využívající schémat HTTP Authentication Schemes [41]. To nejjednodušší je přímočaré a nenabízí vysoký stupeň ochrany. Je založeno na posílání jména a hesla v HTTP headeru. Druhé schéma již používá ochranné tokeny nazývané bearer tokens. Tento název indikuje, že přístup bude povolen pouze „nositeli“ (ang. bearer) tokenu. Ovšem jeho využitelnost je stále doporučovaná jen v rámci HTTPS. Existuje také metoda API klíčů (ang. API key), která je považována za vysoký stupeň ochrany. Funguje na principu vygenerování unikátní hodnoty

uživateli, který je znám. Když tedy uživatel znovu navštíví daný systém, je kontrolován skrze unikátní klíč. Mezi metodu, která je v dnešní době velice populární je OAuth 2.0.

OAuth 2.0

Funguje na principu vytvoření tokenu uživateli, který se jím může prokazovat. [42] Token má ovšem typicky omezenou dobu platnosti. Přístupový token (ang. access token) funguje podobně jako API klíč. Umožňuje uživateli přístup k datům, dokud neskončí jeho platnost. Existuje však také obnovovací token (ang. refresh token). Ve chvíli, kdy přístupovému tokenu vyprší platnost, refresh token získá nový přístupový token. OAuth 2.0 řeší i autorizaci, umožňující vyšší stupeň ochrany a větší kontrolu. Nad OAuth 2.0 může pracovat také další vrstva OpenID Connect.

OpenID

Jedná se o způsob, kterým lze autentifikovat uživatele bez potřeby implementace vlastního řešení [43]. O bezpečnost se stará poskytovatel dané OpenID. Tato metoda je decentralizovaná. Při ověřování uživatele se pošle požadavek správci příslušné OpenID entity. Proces je velice jednoduchý a v dnešní době se už stává zažitým trendem, a to nejspíš právě díky uživatelskému pohodlí, které nabízí. Nejčastěji se k autentizaci touto metodou využívá Google, Facebook, Instagram nebo Yahoo.

2.2.12 Autorizace

Autorizace je úzce spjatá s autentizací, jedná se však o zcela jiný proces [41]. Během tohoto procesu není cílem ověřit totožnost dané entity, ale spíše co je této entitě dovoleno (viz obrázek 2.5). Jejím cílem je ověřit práva k přístupu. Jde tedy o způsob, jak jsou zpřístupněny části systému uživateli dle jeho přidělené uživatelské role. Uživatel může být sice systémem autentizován, ovšem v důsledku nedostatečné úrovně přidělených práv už nemusí být autorizován. V takovou chvíli bude jeho požadavek zamítnut.

Obrázek 2.5 Autentizace a Autorizace [41]

2.2.13 JWT Token

JSON Web Token nebo také JWT Token je standard umožňující bezpečný přenos dat mezi 2 stranami. Obsah dat má strukturu JSON. Server vygeneruje token, jehož nositel se z jeho využitím kupříkladu legitimuje jako admin. Tento token je poté zpřístupněn klientské aplikaci. Uživateli je poté povoleno provádět úkony příslušející roli admina, má-li tento token k dispozici. Typicky bývá token podepsán soukromým klíčem ze strany serveru a tak si lze jednoduše ověřit jeho pravost. Bezpečnost zajišťuje právě digitální podpis. JWT je podepsán pomocí HMAC algoritmu nebo zmíněného soukromého klíče či veřejného klíče. Často se využívá v kombinaci s RSA šifrou. Využitelnost JWT je tedy orientovaná na autorizaci a zasílání zpráv. Jakmile je uživatel ověřen, každý jednotlivý požadavek už bude obsahovat JWT token. K zasílání zpráv je zase využíváno zmíněného digitálního podpisu. Dalšími standardy úzce spjatými s JWT jsou webové šifrování a webový podpis, které jsou taktéž založené na JSONu. [44]

2.2.14 Cookies

Jsou to malé kusy kódu, které jsou vytvořené webovou stránkou, ovšem uložené na straně klienta. Prohlížeč má tyto data uloženy pro každou stránku zvlášť. Existuje více druhů cookies. Starší Magic Cookies a v dnešní době rozšířené HTTP cookies. Jejich cílem je sledování a ukládání informací o uživateli během každého sezení (ang. session) (viz 2.2.15). Webové server pošle identifikační data na webový prohlížeč, cookies jsou takto identifikovány a začínají být posílány se specifickými datami. Typicky se jedná o pár jména a hodnoty, který uživatele dokáže identifikovat. Cookies jsou základem moderního internetu, ovšem znepokojivé z hlediska soukromí. [45]

Dle [46] je třeba zvážit všechny výhody a nevýhody a nastavit na cookies limity. Díky cookies dokáže spousta obchodů předpovídat nákupní požadavky uživatele. Zde je třeba uvážit sběr informací o uživateli a jeho možné dopady.

Cookies zajišťují, že je klient pro webovou stránku znám, je schopen se automaticky přihlásit nebo si pamatovat třeba i obsah košíku daného webu. Problém ovšem nastává ve chvíli, kdy jsou k těmto pro klienta neškodným informacím sbírány další kusy dat, které se dají zneužít a naopak mohou představovat potenciální hrozbu. K těmto typům sběru dat však nedochází již tak často, většina cookies je bezpečná a věrohodné webové stránky často nejprve chtějí potvrzení uživatele o jeho souhlasu s cookies. [47]

2.2.15 HTTP Session

Relace nebo také sezení (ang. session) je obecně spojení mezi klientem a serverem. HTTP session, o které se budeme bavit ve zbytku této kapitoly, definuje způsob, jak uložit informace o jednotlivých uživatelích. HTTP je bezestavový protokol, tedy neukládá žádné informace, které by mu daly kontext o uživateli. Session pomáhá HTTP tento kontext dát. Session se předává buď jako URL nebo součástí HTTP cookie. Může ovšem nastat problém s bezpečností.

Spousta systémů podléhá bezpečnostní hrozbě zvané Session Hijacking. Uživatel po autentizaci během sezeních neustále posílá velké množství citlivých informací. Útočník může zaútočit na síťovou vrstvu a ukrást informace potřebné pro provedení útoku na aplikační vrstvě. Často se jedná o hijack na TCP a UDP session. TCP hijack lze provést více způsoby. Jedním z nich

je IP spoofing, kdy se útočník vydává za někoho jiného nebo. Další metodou je man in the middle útok, jehož cílem je krádež paketů. Co se týče UDP session, ta není tak zapezpečená jako TCP a je tedy poměrně snadným cílem útočníků. Samotné session hijack na aplikační vrstvě potom spočívá v krádeži existující session. Lze ovšem také z nasbíraných dat vytvořit session novou a útok tak provést tímto způsobem. Hlavním cílem útočníků pro úspěšnou krádež bývá session ID, které je unikátním identifikátorem dané session. ID lze získat například krádeži paketů v Cookies. Tento útok je velice podobný Man in the middle útoku. Dalším způsobem je snažit se odhadnout vzor, který session ID nesou a dle toho metodou hrubé síle (ang. Brute force) odpovídající ID najít. Účinnou ochranou je šifrování paketů pomocí SSL nebo SSH. Session potom disponuje silným ID, které je pro útočníka v rozumném časovém horizontu nerozluštitelné [48].

2.2.16 Local Storage

Jedná se o typ webového úložiště, které umožňuje přístup k lokálnímu Objektu Úložiště (ang. Storage Object). Do toho úložiště lze ukládat data a to po neomezenou dobu. Po zavření prohlížeče a jeho následném otevření jsou tedy data stále perzistentně uložena. K jejich smazání může dojít v případě, že je smazána cache [49].

Local Storage ukládá data jako pár klíč a hodnoty. Ukládaná data by však měly být pouze pro uživatele necitlivé údaje. Příkladem dat, která se do tohoto úložiště ukládají, mohou být jazyky nebo témata (ang. themes). Local Storage nikdy neměla za cíl bezpečnost a je tedy snadným cílem útočníků. Mezi časté útoky patří XSS, fyzický přístup k prohlížeči nebo využití slabin JavaScriptu. Efektivní ochranou je ukládání citlivých dat na straně serveru.

2.2.17 Mikroslužby

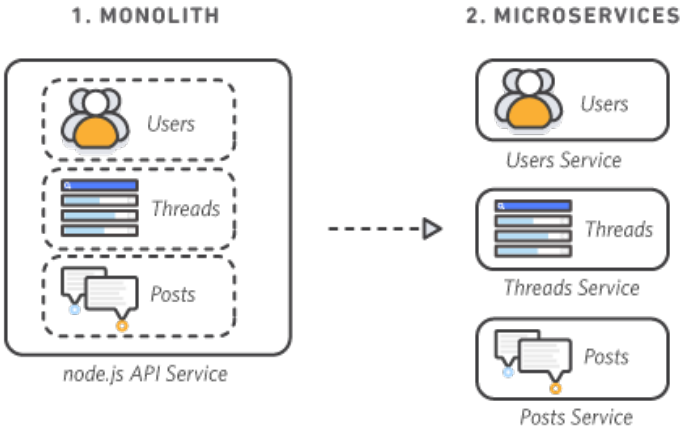
Mikroslužby (ang. Microservices) jsou architektonický styl, nejedná se tedy o software, jako spíše o myšlenku. Jde o přístup ve vývoji systému kompozicí malých nezávislých služeb, které komunikují skrze API. Nabízí velice rychlou a spolehlivou tvorbu komplexních systémů, neboť kdykoliv umožňují vytvoření jedné služby zastávající nějakou novou funkci, kterou přidají k celému technologickému stacku [50].

Dle [51] se jedná se o kolekce jednotlivých služeb, které mají specifické charakteristiky. Důležitou charakteristikou je nezávislost jednotlivých služeb. Ty se dají upravovat, škálovat i nasazovat, aniž by ovlivnily ostatní služby, neboť spolu vzájemně nesdílí žádný kód. Každá služba má svou vlastní specifickou funkcionalitu, kterou zastává. Kdyby došlo k zrobustnění jedné služby, typicky lze velice jednoduše tuto službu opět rozdělit na menší dílčí komponenty.

Výhody mikroslužeb spočívají ve flexibilitě, škálovatelnosti a snadné rozšířitelnosti. Kód jedné služby lze snadno přepoužívávat v jiných částech celého systému. Další výhodou je vysoká odolnost vůči výpadku. Výpadek jedné služby neovlivní chod celé aplikace, což se ovšem nedá říct o systému, který odpovídá architektuře monolitu.

Mikroslužby vs. monolity

Pro složitější aplikace se dnes již mikroslužby používají naprosto běžně [51] namísto robustního monolitického architektonického stylu. U monolitu jsou narozdíl od mikroslužeb veškeré procesy velice úzce spjaty a běží i fungují jako jeden celek (viz obrázek 2.6). Přidávání nových funkcionalit je mnohem složitější než u mikroslužeb, neboť je třeba takovou funkcionalitu zanést komplexně do už tak robustního systému. Složitost přidávání nových funkcionalit samozřejmě také roste se zvětšujícím se množstvím kódu a ještě větším zrobustňováním celého systému. Jelikož jsou veškeré systémy úzce spjaty, je časově náročné vnášet do monolitu nové nápady a se zvětšováním se systému se také snižuje dostupnost. Může se stát, že kvůli těmto provázanostem selhání jediné části systému může vést ke komplexnímu výpadku.



Obrázek 2.6 Monolit a mikroslužba [51]

Kapitola 3

Praktická část

V této části se diplomová práce zaobírá praktickými částmi využívající principy popsané v teoretické části. První krokem je analýza stávajícího řešení a vytvoření představy o aktuálním cíli dle požadavků klienta. Na základě analýzy je dále vytvořen komplexní návrh obsahující od definic cílových skupin, přes tvorbu různých druhů modelů, až k samotnému testování jednoho z těchto modelů. Na základě návrhu, který jasně stanoví, jak by měl vypadat výsledný produkt, navazuje implementační část. Ta nejprve popisuje architekturu jednotlivých částí aplikace a poté prezentuje výslednou ukázku. Výsledná aplikace je poté testována v produkci a následně nasazena jako funkční řešení. Posledním krokem praktické části práce je vytvoření dokumentace.

3.1 Analýza

Tato sekce se nejprve věnuje analýze požadavků. Důležitá je zde komunikace s klientem pro vymezení požadavků, které jsou splnitelné, jasně zadané a vzájemně konzistentní. Popsána je dále analýza stávajícího řešení. Na základě této analýzy je potřeba určit, jaké změny se musí provést, aby nový systém vyhovoval zadaným požadavkům, a tedy cílům klienta. Kombinace těchto dvou analýz následně vede k jejich srovnání a výslednému zhodnocení, které slouží jako základ pro tvorbu návrhu.

3.1.1 Analýza požadavků

Při analýze požadavků je nejprve potřeba identifikovat zúčastněné strany, konkrétně klienta. Analýza totiž probíhá formou dialogu. Je tedy k věci uvážit, zda není prospěšné přizvat další účastníky k tomuto dialogu. Ti mohou být schopni pomoci s komunikací. Konkrétně se mohou s klientem domluvit na jasně definovaných požadavcích. pochopení klientova záměru je základem této analýzy. Není totiž horší scénář, než když jedna strana špatně pochopí záměry strany druhé a vývoj přejde do další fáze s chybou. Analýza požadavků je obecně pojem softwarového inženýrství popisující úkoly, které je potřeba splnit pro uspokojení potřeb klienta kladených na systém. Od těchto požadavků se odvíjí vlastnosti, které by systém měl splňovat. Lze také říci, že zachycují omezení a vymezují hranice systému. Požadavky musí být v rámci systému konzistentní, splnitelné a testovatelné. Je nutné je také detailně popsat kvůli pozdějším fázím návrhu.

Typicky se požadavky rozdělují do 2 kategorií:

- Funkční požadavky - Určují konkrétní funkce, které musí nutně fungovat. Obecně popisují, co by měl systém dělat. Proto se jedná o funkční požadavky, protože bez jejich splnění, systém funkční není.
- Nefunkční požadavky - Kladou požadavky spíše na návrh. Nesouvisí s funkčností a popisují charakteristiky systému, které jsou očekávané. Konkrétně se může jednat o vymezení vůči designu nebo požadavky na výkon systému. Jedná se však o typ požadavků bez jejichž splnění systém bude fungovat korektně, proto jsou také nazývány nefunkčními požadavky.

Hlavním cílem v rámci projektu TAČR je vytvoření portálu, který by mohl sloužit výzkumníkům pro srovnávací studie literatury. Cílem je aplikovat moderní metody zpracování přirozeného jazyka. Portál by měl sloužit také vyučujícím českého jazyka na středních školách, stejně tak jako studentům středních škol, které čeká maturitní zkouška. Využití portálu cílí také na širokou laickou veřejnost.

Shrnutí požadavků

Tato sekce shrnuje požadavky kladené na serverovou a klientskou část. Funkční požadavky jsou uvedeny v tabulce 3.1. Nefunkční požadavky lze nalézt v tabulce 3.2.

Tabulka 3.1 Funkční požadavky

	Popis	Priorita
FP01	Registrace: Uživatel by měl mít možnost se zaregistrovat pomocí jména a hesla	Vysoká
FP02	Autentizace: Pokud je uživatel zaregistrován, měl by mít možnost se přihlásit pomocí kombinace jména a hesla	Vysoká
FP03	Domovská stránka: Uživateli by měla nabídnout základní údaje ohledně počtu sbírek, autorů, básní a veršů. Dále by měla obsahovat informaci o básni dne. Uživateli by také měla umožnit full-textové vyhledávání básní i autorů	Vysoká
FP04	Seznam sbírek: Uživatel by měl mít možnost zkoumat sbírky básní skrze seznam, který umožňuje vyhledávání a dále řazení dle názvu sbírky, jména autora či roku vydání. Tato stránka by také měla odkazovat na detail sbírky	Vysoká
FP05	Detail sbírky: Uživatel by měl mít možnost dostat se na stránku detail sbírky obsahující informace o konkrétní sbírce. Požadován je interaktivní obsah sbírky, možnost stránkování mezi básněmi, bibliografické údaje o sbírce, možnost přepínání různých vrstev náhledu, a také možnost dostat se na detail básně.	Vysoká
FP06	Detail básně: Uživatel by měl mít možnost dostat se na stránku detail básně obsahující informace o konkrétní básni.	Vysoká
FP07	Seznam autorů: Uživatel by měl mít možnost zkoumat autory skrze seznam, který umožňuje vyhledávání a dále řazení dle jména autora, počtu jeho sbírek a roku narození. Tato stránka by také měla odkazovat na detail autora	Vysoká
FP08	Detail autora: Uživatel by měl mít možnost dostat se na stránku detail autora obsahující informace o konkrétním autorovi. Mezi požadované informace patří rok i místo narození a úmrtí, seznam sbírek autora, spoluautoři a sekce témata, která definuje význačné znaky autora.	Vysoká
FP09	Pro školy: Uživatel by měl mít možnost shlédnout si informace cílené jako sdělení pro školy. Jedná se o statickou stránku	Střední
FP10	Pro experty: Přihlášený uživatel by měl mít možnost přejít do sekce pro experty. Ta prozatím není více definována	Střední

Tabulka 3.2 Nefunkční požadavky

	Popis	Priorita
NP1	Rychlost vyhledávání: Vyhledávání by mělo být optimalizované pro rychlejší vracení výsledků	Střední
NP2	Responzivita: Systém by se měl umět přizpůsobit dle velikosti okna	Nízká
NP3	Stránkování: Stránky Sbírky a Autoři by měly umožnit stránkování obsahu	Nízká

3.1.2 Analýza stávajícího řešení

Stávající řešení, které vzniklo v rámci mé bakalářské práce a diplomové práce Ing. Tomáše Chvosty, je momentálně dostupné na `https://ucl-poezie.fit.cvut.cz/`. Serverová část je vytvořena pomocí NodeJS a databáze MongoDB. Ke tvorbě klientské částí bylo využito knihovny ReactJS a principů Material designu, konkrétně knihovny MaterialUI. Pro nepřihlášeného uživatele je dostupná domovská stránka a sekce ohledně informací o projektu. Obě tyto stránky mají pouze statický obsah. Součástí obsahu pro nepřihlášeného uživatele je možnost registrace pomocí uživatelského jména a hesla. Řešení také nabízí možnosti autentizace i autorizace, ovšem citlivé údaje jsou ukládány do Local storage, a tak toto řešení nepatří mezi bezpečné. Uživatel je navíc po autentizaci přihlášen na neomezeně dlouhou dobu, což může také vést k bezpečnostním problémům.

Přihlášený uživatel má k dispozici 1 ze 3 uživatelských rolí. Mezi ně patří role čtenáře, který má oproti nepřihlášenému uživatele k dispozici navíc Knihovnu a sekci Mé knihy. Další uživatelskou rolí je editor, která má stejné pravomoce jako uživatel s rolí čtenáře, ovšem navíc má přístup do sekce Správa sbírek. Poslední uživatelskou rolí, které stávající řešení nabízí je redaktor. Redaktor má již zpřístupněné veškeré možnosti, které portál nabízí, které oproti předchozí roli editora obsahuje navíc přístup do sekce Správa uživatelů.

Knihovna je stránka obsahující jednotlivé sbírky básní formou tabulky. Tabulka umožňuje vyhledávání a řazení sbírek básní dle názvu knihy, autora, roku vydání a místa vydání. Implementováno je také stránkování a odkazy na akce čtení či ukládání knih. Akce ukládání knihy uloží sbírku do vlastního seznamu sbírek, kterých může uživatel vytvořit neomezené množství. S těmito listy lze poté pracovat v sekci Mé knihy.

Sekce Mé knihy umožňuje vytvářet seznamy sbírek a přidávat nebo odebírat sbírky z jednotlivých seznamů. Nad daným seznamem lze přejít do sekce s rozšířenými filtry, která umožňuji aplikovat nejrůznější nástroje. Jedná se o statistiky, abecední slovníky, frekvenční slovníky, full-textové vyhledávání a vyhledávání slov pro kontexty.

Správa sbírek slouží k přehledu všech sbírek básní, které jsou v databázi. V této sekci s nimi lze manipulovat buď formou smazání sbírky nebo její úpravou. Úprava je ovšem ve stávajícím řešení velice neefektivní a v podstatě nepoužitelná.

3.1.3 Srovnání

Oproti myšlence, jak by nový portál měl fungovat a vypadat se aktuální řešení dost vzdaluje. Budou existovat pouze dvě uživatelské role. První je neregistrovaný, běžný uživatel. Druhou rolí bude uživatel s rolí výzkumníka, u kterého bude vyžadováno přihlášení. Autentizace a autorizace musí být provedena modernějšími metodami, které zajistí bezpečnost. Požadavky jsou z velké částí v nesouladu z aktuálním řešením. V návrhu je důležité zohlednit tvorbu stránek, které jsou oproti původnímu řešení nové:

- Domovská stránka
- Detail sbírek
- Detail básně
- Seznam autorů
- Detail autora
- Pro školy
- Pro experty

Některé částí stávajícího řešení lze však stále využít jako základ k urychlení vývoje, neboť si jsou z aktuálními požadavky poměrné blízké a jejich úpravy nemusí nutně znamenat vytvoření nové stránky. Je však možné, že se úpravám, které budou znamenat v podstatě vytvoření nové stránky, vyhnout nepůjde:

- Registrace
- Přihlášení
- Seznam sbírek

3.1.4 Zhodnocení

Staré řešení již funguje s běžící serverovou částí napojenou na databází s daty, které jsou potřeba i v aktuálním projektu. Klientská část má rovněž již rozdělenu modularitu a různá nastavení včetně směrování stejným způsobem, jako je to vhodné i pro tento projekt. Z tohoto důvodu nemá smysl zakládat úplně nový projekt. Lepší variantou se jeví modifikace serverové i klientské části stávající aplikace. Některé části stávajícího řešení, jako je Registrace, Přihlášení nebo Seznam sbírek mají ukotvenou logiku, kterou půjde snadno přepoužít. Úpravou v těchto částech bude z velké části pouze design. Co se týče nových stránek, ty je potřeba od počátku navrhnout.

3.2 Návrh

Tvorba návrhu UI probíhá v cyklech. Prvním krokem je návrh, následuje implementace a poté vyhodnocení. Tento proces se opakuje, dokud není dosaženo uspokojivého výsledku. Míra propracovanosti návrhu koreluje s množstvím těchto cyklů. Tato metodika se ovšem lépe aplikuje u větších projektů.

Kvalitní návrh se snaží postihnout veškeré akce a scénáře, na které lze na základě analýzy reagovat. Důležité je schválení všech těchto detailů klientem, což indikuje, že se jeho představa shoduje s představou návrháře. Cílem této sekce je právě takový návrh, kde se naše představa bude shodovat s představou klienta. K tomuto účelu by měly značně přispět konzultace s Ústavem pro českou literaturu a zpracované informace z těchto sezení, které jsou již součástí předchozí analytické sekce. Tímto způsobem by měl v rámci této práce být dostatečný pouze jeden cyklus návrhu UI, kdy dojde ke komplexnímu návrhu založeném na analýze, implementaci dle tohoto návrhu a závěrečném zhodnocení výsledků.

3.2.1 Persony

Jedná se o profily fiktivních lidí, kteří spadají do některé ze skupin uživatelů, které můžou odrazit případné potřeby či nedostatky v rámci projektu. Vizualizací pomocí těchto příběhů lze pochopit emocionální stav uživatele a uvědomit si, zda ubírání se aktuálním směrem, má smysl. Persony se vytváří za účelem definování byznysové představy o potenciálních zákaznících. V tomto projektu lze rozdělit cílové skupiny na:

- Výzkumníci
- Čtenáři

Již byly zkonzultovány konkrétní nástroje, které by nový portál měl nabídnout pro výzkumníky z Ústavu pro českou literaturu. Případné další softwarové pomůcky lze kdykoliv přidat k těm stávajícím. Realizovatelná je také úprava stávajících pomůcek tak, aby odpovídaly přesné představě výzkumníků. Jelikož je výzkumník spíše uživatelem na úrovni administrátora a představy o jeho činnosti jsou zřejmé, vytváření person reprezentující tuto cílovou skupinu by návrhu výrazně nepřispělo.

Skupina čtenářů by však mohla poskytnout konkrétní představu o tom, jak by vlastně vypadala osoba, která by portál měla motivaci navštívit. Stejně tak i představu o osobě, která by portál s velkou pravděpodobností nikdy nenavštívila. Nejprve je nutné definovat osobnost, potřeby a další charakteristiky, které fiktivní charakter ztvární. Existují 3 typy person:

- Persona A - Typický uživatel, vážný zájemce o poezii (viz tabulka 3.3)
- Persona B - Příležitostný čtenář, občasný zájemce o poezii (viz tabulka 3.4)
- Persona C - Antipersona, nejeví zájem o poezii (viz tabulka 3.5)

Tabulka 3.3 Persona A

Jméno	Dominika
Věk	46
Pohlaví	Žena
Povolání	Učitelka literatury
Koníčky	Učení, literatura, divadelní hry
Příběh	Dominika je učitelkou literatury na gymnáziu. Vyučuje literaturu již 20 let a její hlavní životní náplní je předávání znalostí studentům. Volné chvíle tráví s rodinou a čtením poezie.

Tabulka 3.4 Persona B

Jméno	Iveta
Věk	34
Pohlaví	Žena
Povolání	Prodavačka obuvi
Koníčky	Literatura, čtení, stolní hry
Příběh	Iveta každé ráno vstane v 7 ráno a jde do práce. Vlastní podnik, kde prodává obuv. Ve volných chvílích v práci čte poezii, která ji pomáhá jako výplň času, kdy v obchodě nejsou žádní zákazníci. Po práci chodí s přáteli na pivo a hrát stolní hry.

Tabulka 3.5 Persona C

Jméno	Alex
Věk	23
Pohlaví	Muž
Povolání	Programátor
Koníčky	Posilování, jezení jídla
Příběh	Alex 6x v týdnu dálkově pracuje. Zbytek času, který mu zbývá během běžného dne, stráví v posilovně, případně na jídle s kamarády. Jeho časové možnosti jsou silně limitovány

Zmíněné persony jsou založeny na mých kamarádech a jejich pohledu k potenciálu tohoto portálu. Je zřejmé, že portál bude navštěvovat nadšenec do poezie. Dalším případným návštěvníkem může být člověk, který se o poezii alespoň trochu zajímá a stránka jej zaujme. Návštěvníkem může být také člověk, který prostě jen rád čte a na stránku náhodně narazí. U posledních dvou případů je však nutností, aby si dotyčný na takové aktivity byl schopen najít čas. Naopak člověk, který je časově vytížen, nemá zájem o poezii nebo čtení obecně, určitě tento portál navštěvovat nebude.

Portál by měl obsahovat také sekci Pro školy, která by měla nést zajímavé informace pro žáky. Dalšími potenciálními návštěvníky by tedy mohli být právě žáci. Detailnější informace o vývoje této sekce ovšem prozatím nejsou známy, a tak tuto úvahu nelze dále rozvíjet.

3.2.2 Uživatelské cíle

Jedná se o popis stavů, kterých chce uživatel dosáhnout. Jako uživatelský cíl však není myšlen stav dosažený v softwarovém systému, nýbrž stav, který popisuje reálný cíl uživatele [52]. Porozumění hlavních cílů uživatele usnadňuje tvorbu návrhu. Snižuje také pravděpodobnost chyby, která může vést k nesprávnému vývoji systému. Získat informace o uživatelských cílech však není jednoduché. Uživatel totiž nemusí úplně rozumět, co se vlastně jeho cíly v daném kontextu myslí. Je proto potřeba při komunikaci řešit vyšší míru detailu, aby se obě strany zadavatele i vykonavatele skutečně chápaly.

Na uživatele je zde potřeba dívat se jako na čtenáře a uživatelské cíle tomu také přizpůsobit. Uživatelské cíle se typicky řadí od nejdůležitějších po ty nejméně důležité. Jako uživatelské cíle čtenáře našeho portálu byly identifikovány následující:

- Přečíst báseň
- Přečíst informace o sbírce
- Přečíst informace o autorovi
- Vyhledat báseň
- Vyhledat sbírku
- Přihlásit se
- Vytvořit účet

3.2.3 Případy užití

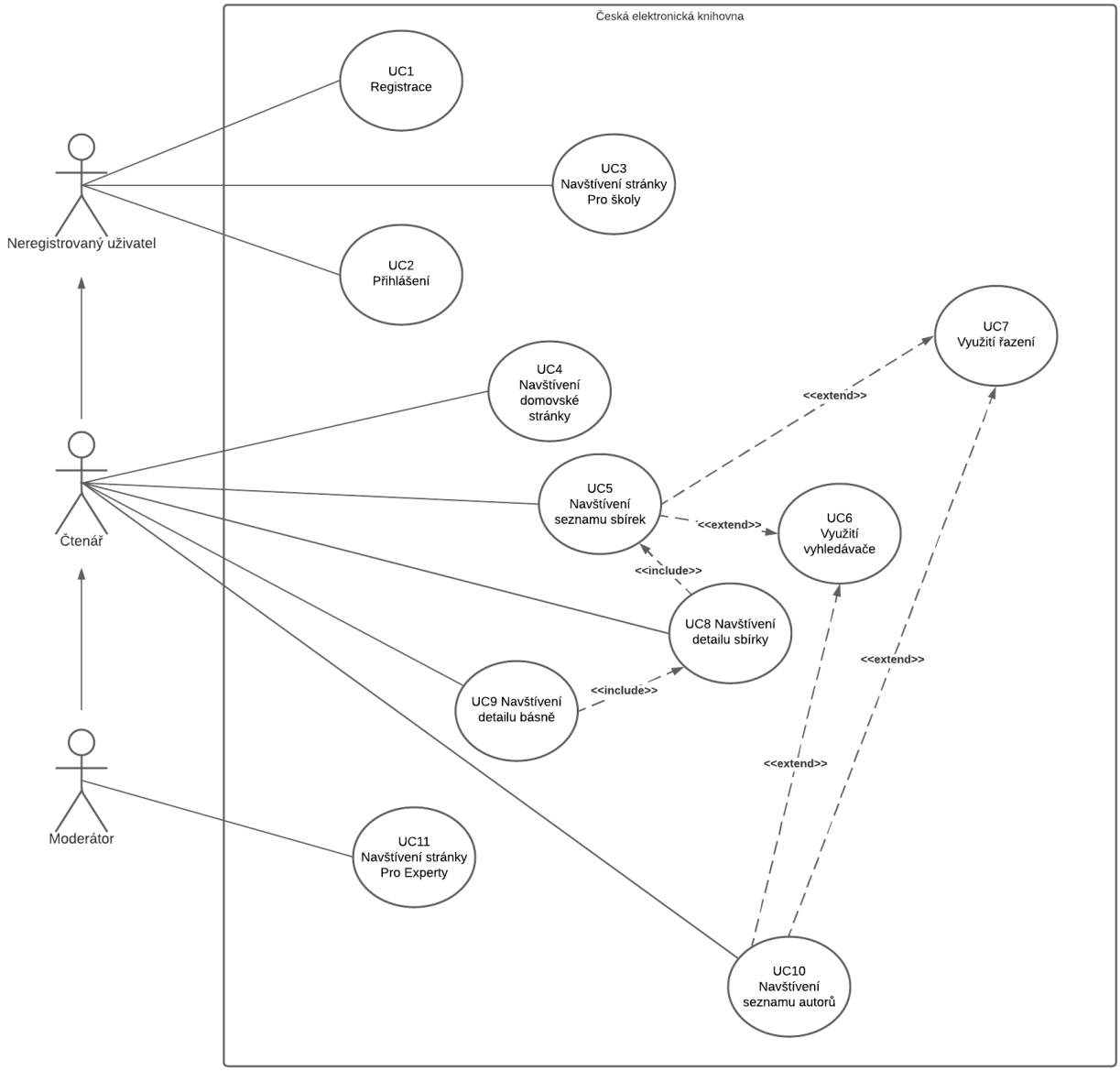
Případy užití (ang. use cases) jsou pojmem softwarového inženýrství [53] popisující seznam akcí, kterými lze dosáhnout nějakého cíle. Typicky je má na starost byznysový analytik. Tyto akce probíhají mezi rolí a systémem. Role se v UML jazyce nazývá aktér (ang. actor). Aktér může být jakýkoliv systém, většinou to však bývá člověk. Případy užití reprezentují funkční požadavky a jsou odvozeny od uživatelských cílů (viz 3.2.2). Pomáhají odhalit případné chyby. Jejich využití lze najít v různých fázích softwarového vývoje. Pomáhají u řešení systémových požadavků a otázkami ohledně návrhu. Jsou také přínosné při testování nebo jako uživatelské manuály. Existují dva typy případu užití:

- Byznysové - popisují pouze byznysový proces a zapojené aktéry

- Systémové - popisují detailní kroky k dosáhnutí uživatelského cíle, typicky se vizualizuje vytvořením diagramu případů užití (viz 3.2.3).

Diagram případů užití

Diagram případů užití (ang. use case diagram) je grafické zobrazení [53]. Cílem je popsat interakci mezi uživatelem a systémem. Nejprve je nutné identifikovat všechny uživatelské role. Poté vybrat jednu z nich a definovat ji uživatelské cíle. Poté vybrat jeden uživatelský cíl a popsat cestu vedoucí k jeho splnění. Opakováním těchto kroků vyjma prvního a za použití jazyku UML byl vytvořen Diagram případů užití pokrývající funkční požadavky systému (viz 3.1).



Obrázek 3.1 Diagram případů užití

3.2.4 Task list

Pracovní prostor lze rozdělit na dvě části [54]. První je úkol nebo také task. Ten je zodpovědný za splnění individuální části nějakého cíle. Splněním více úkolů lze však konkrétního cíle dosáhnout. Druhou částí je skupina úkolů nebo také task group. Task group v sobě typicky obsahuje více úkolů. Jedná se o kategorizace úkolů dle jejich typů. Nejprve byl vytvořen seznam všech úkolů nebo také task list:

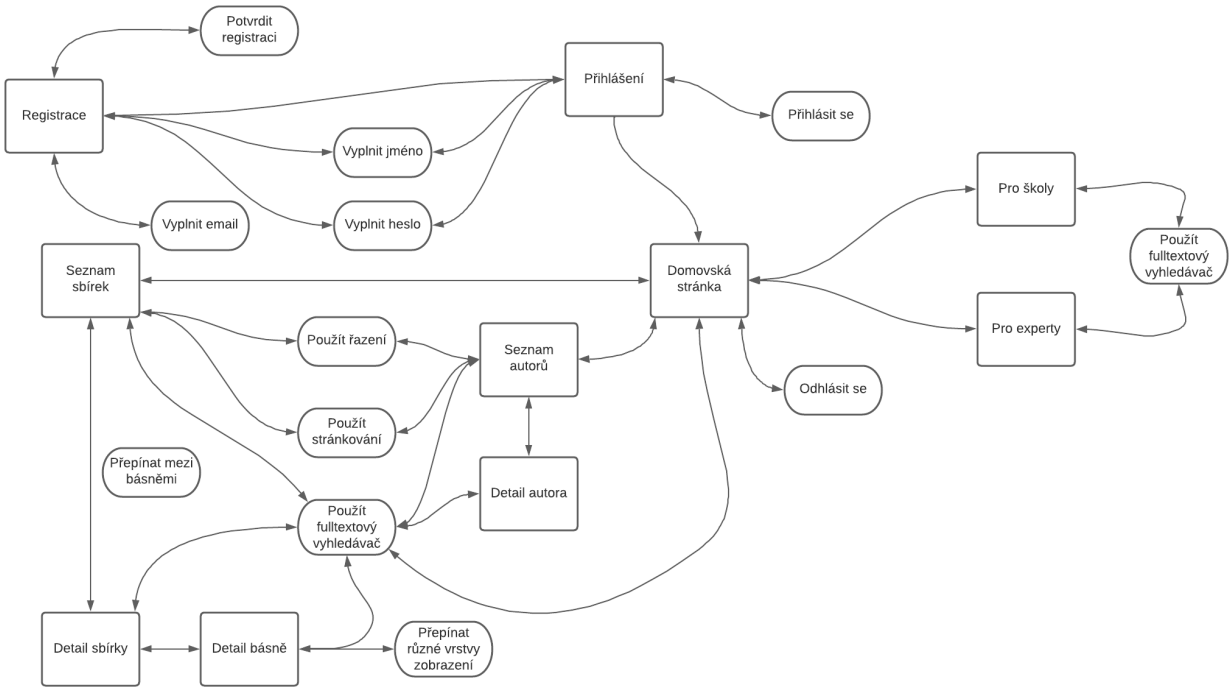
- Přečíst si informace na domovské stránce
- Použít fulltextové vyhledávání
- Registrovat se
- Přihlásit se
- Hledat v seznamu sbírek a použít řazení
- Přejít na detail sbírky
- Přejít na detail básně
- Hledat v seznamu autorů a použít řazení
- Přejít na detail autora
- Přečíst si informace na stránce Pro školy
- Přejít na stránku Pro experty
- Odhlásit se

Následně byly jednotlivé úkoly kategorizovány do dvou skupin úkolů. Vizuální task group, která reprezentuje zobrazení všech stránek, které s nimi souvisí. Jedná se o Domovskou stránku, registrační a přihlašovací formuláře, seznamy sbírek a autorů, detaily sbírek, básní a autorů i stránky Pro Školy a Pro experty. Druhou skupinou úkolů jsou uživatelské akce:

- Domovská stránka - Použít fulltextový vyhledávač, odhlásit se.
- Registrace - Vyplnit jméno, email, heslo, heslo podruhé a potvrdit registraci
- Přihlášení - Vyplnit jméno, heslo a přihlásit se.
- Seznam sbírek - Použít stránkování, řazení, fulltextový vyhledávač a přejít na detail sbírky.
- Detail sbírky - Použít fulltextový vyhledávač, přepínat mezi básněmi, přejít na detail básně.
- Detail básně - Použít fulltextový vyhledávač, přepínat různé vrstvy zobrazení, vrátit se zpět na detail sbírky.
- Seznam autorů - Použít stránkování, řazení, fulltextový vyhledávač a přejít na detail autora.
- Detail autora - Použít fulltextový vyhledávač.
- Pro školy - Použít fulltextový vyhledávač.
- Pro experty - Použít fulltextový vyhledávač.

3.2.5 Task graph

Task graph (viz 3.2) je vizuální reprezentace jednotlivých úkolů popsaných pomocí task list a rozdělených s využitím task groups. Jedná se o orientovaný graf, kde jedním typem uzlů jsou jednotlivé stránky a druhým typem uzlů uživatelské akce. Hrany grafu reprezentují průchod systémem pomocí přechodů mezi stránkami, anebo prováděné akce na těchto konkrétních stránkách.



Obrázek 3.2 Vizuální reprezentace jednotlivých úkolů pomocí task graph

3.2.6 Wireframe

Drátěný model nebo také wireframe je základním pilířem tvorby návrhu. Prezentuje rozmístění prvků na jednotlivých stránkách. Nejedná se ovšem o grafický návrh. Neobsahuje barvy a není interaktivní. Existují 3 typy drátěných modelů:

- Low-fidelity (viz 3.2.7)
- Mid-fidelity
- High-fidelity (viz 3.2.9)

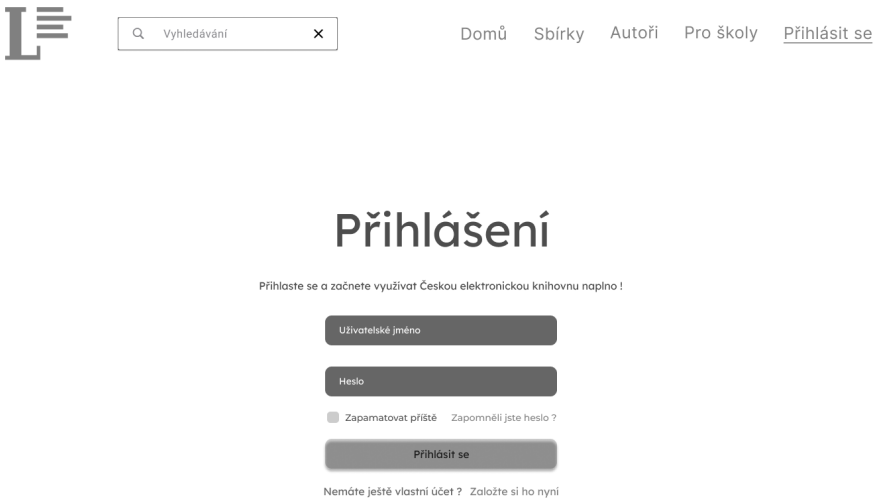
3.2.7 LOFI

V UX designu se využívá celé řady typů modelů. Některé jsou pouhými papírovými náčrty ucelující základní představu o vzhledu systému, jiné jsou komplexními prototypy aplikace, které odráží skutečný vzhled a fungování aplikace.

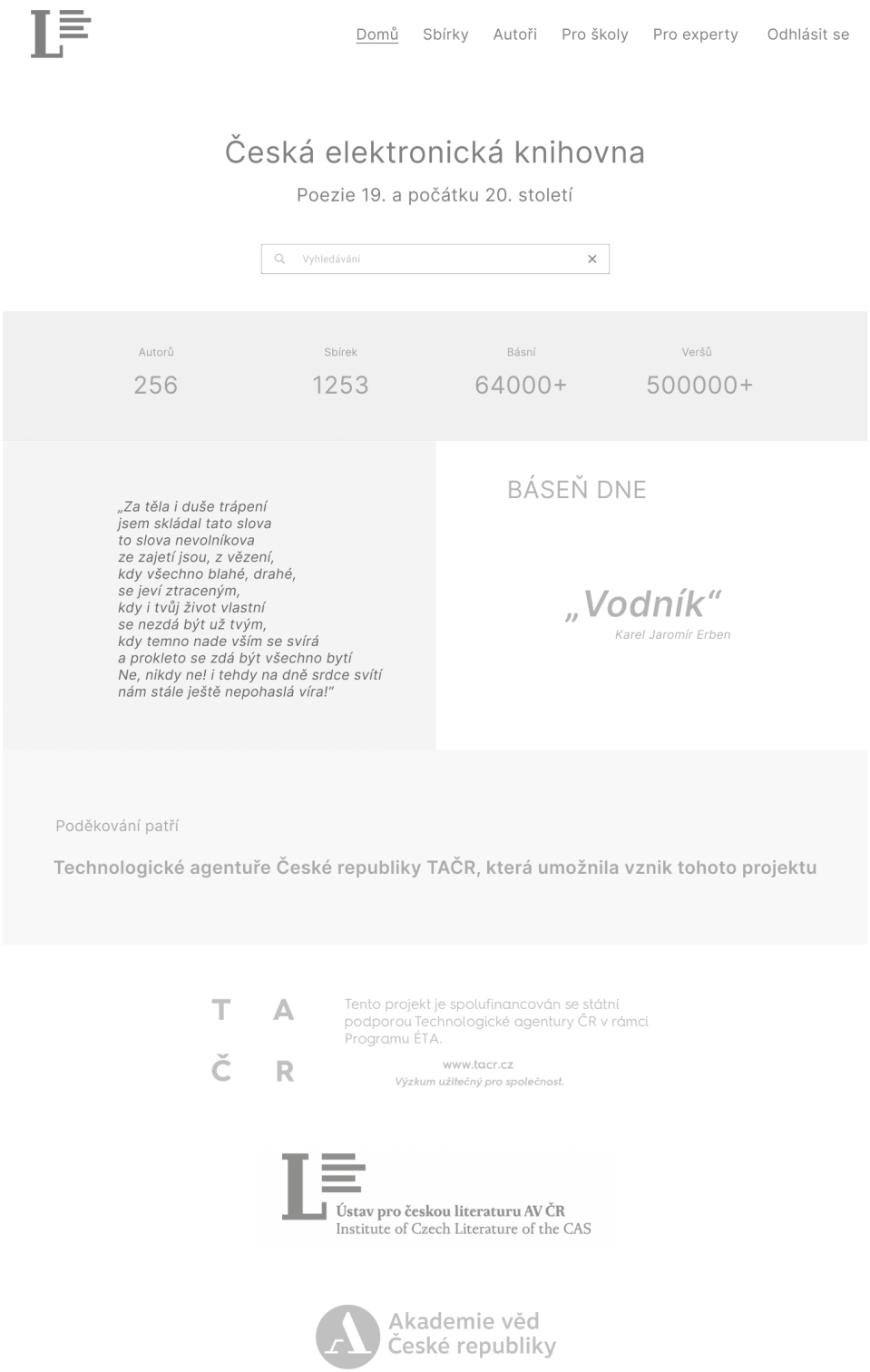
Low-fidelity wireframe nebo také LOFI je nadstavbou papírového modelu [55]. Popisuje základní obsah stránek a v omezené míře i jejich vizuál. Nejedná se však o interaktivní model jako spíše o ukázku kostry systému. Pojem fidelity lze v tomto kontextu popsat jako míru přesnosti, složitosti nebo kvality. Low-fidelity tedy mluví o méně detailní propracovanosti, jednoduchosti a samozřejmě i nepřesnosti. Low-fidelity model nebo také low-tech model tedy indikuje využití minimálního množství technologií pro jeho vytvoření. Výhodou tohoto modelu jsou nízké náklady. Do jeho tvorby se mohou zapojit také laici na poli programování. Umožňuje reprezentovat složitější myšlenky jednoduchým způsobem. Lze díky němu odhalit počáteční chyby, které by se v pozdějších fázích návrhu mohly řešit zbytečně složitě. Často slouží jako vzor pro složitější HIFI model (viz 3.2.9).

Ukázka LOFI wireframů

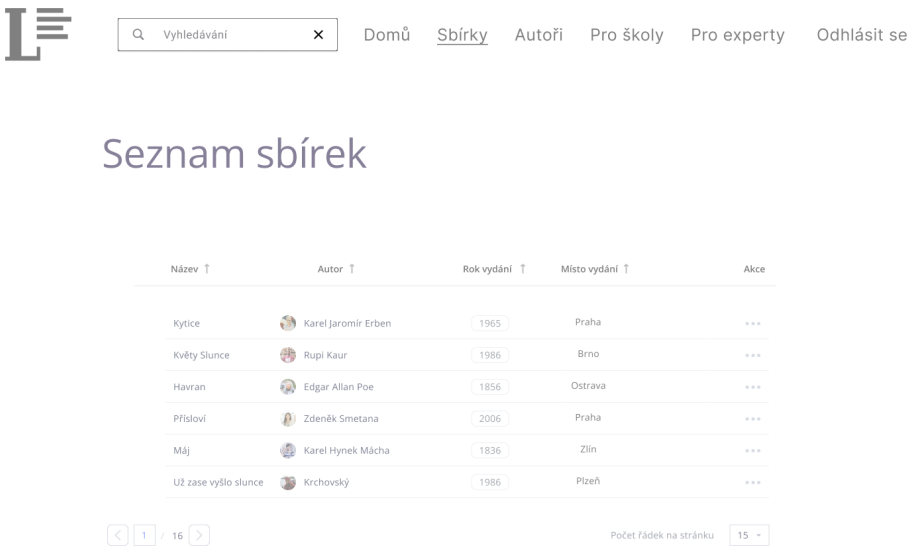
Níže lze vidět ukázku několika vytvořených wireframů. Kompletní kolekci wireframů lze dohledat mezi přílohami.



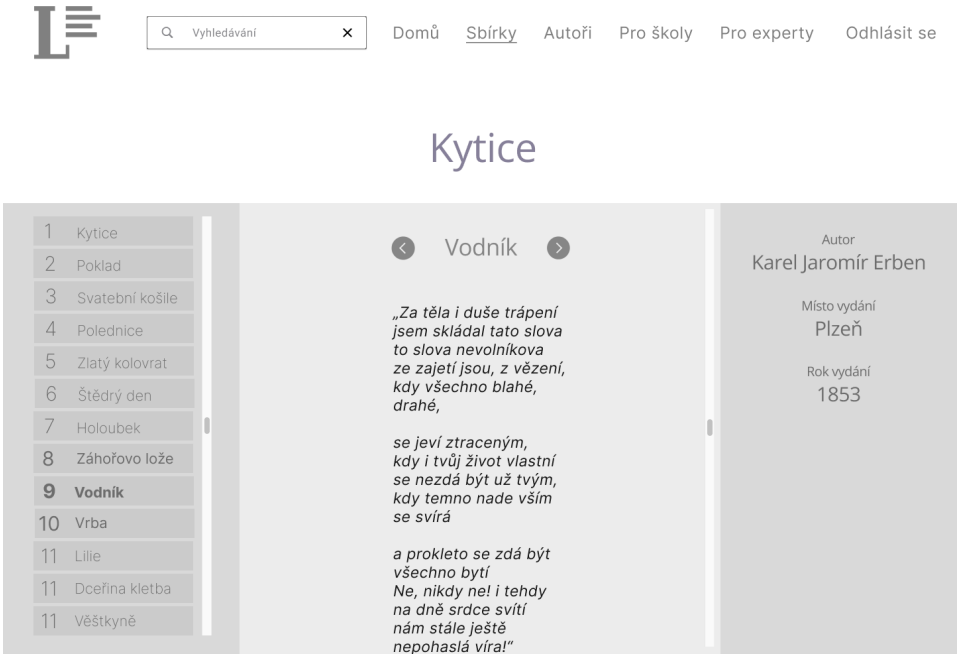
Obrázek 3.3 Ukázka přihlášování



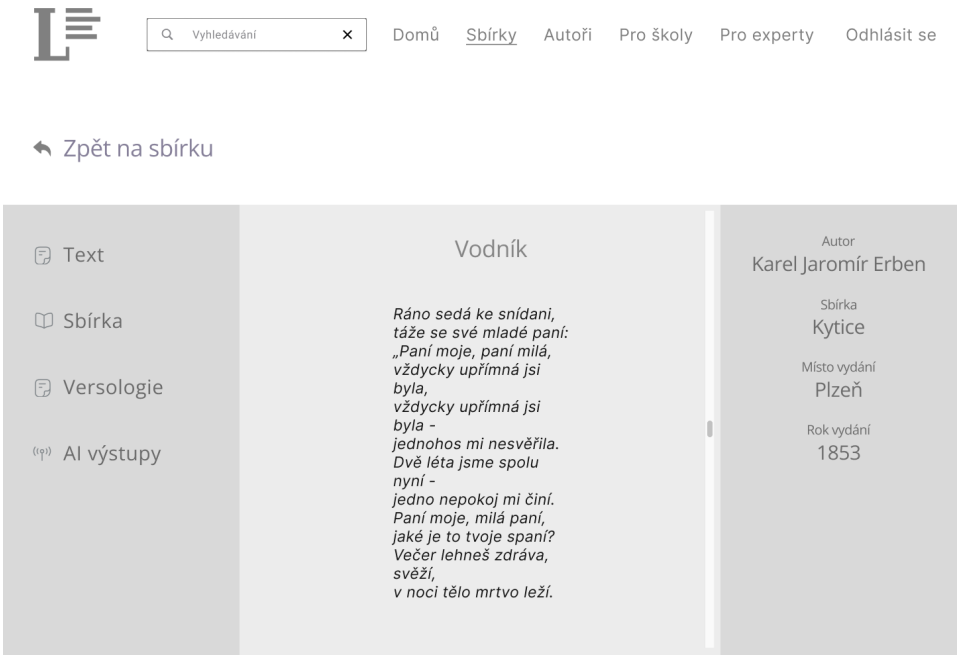
Obrázek 3.4 Ukázka domovské stránky



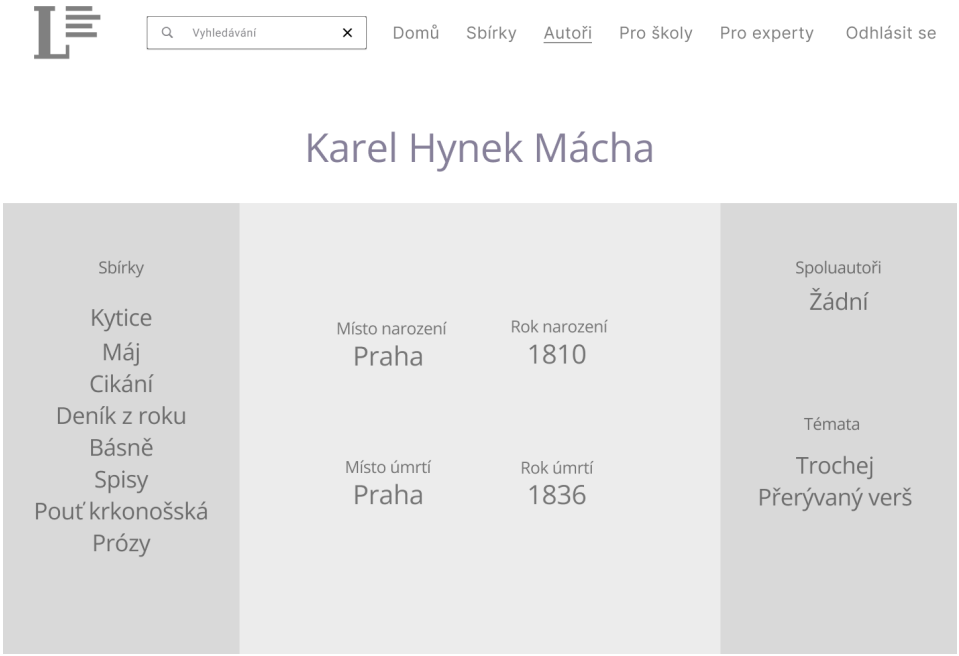
Obrázek 3.5 Ukázka stránky sbírek



Obrázek 3.6 Ukázka detailu sbírky



Obrázek 3.7 Ukázka detailu básně



Obrázek 3.8 Ukázka detailu autora

3.2.8 Prototyp

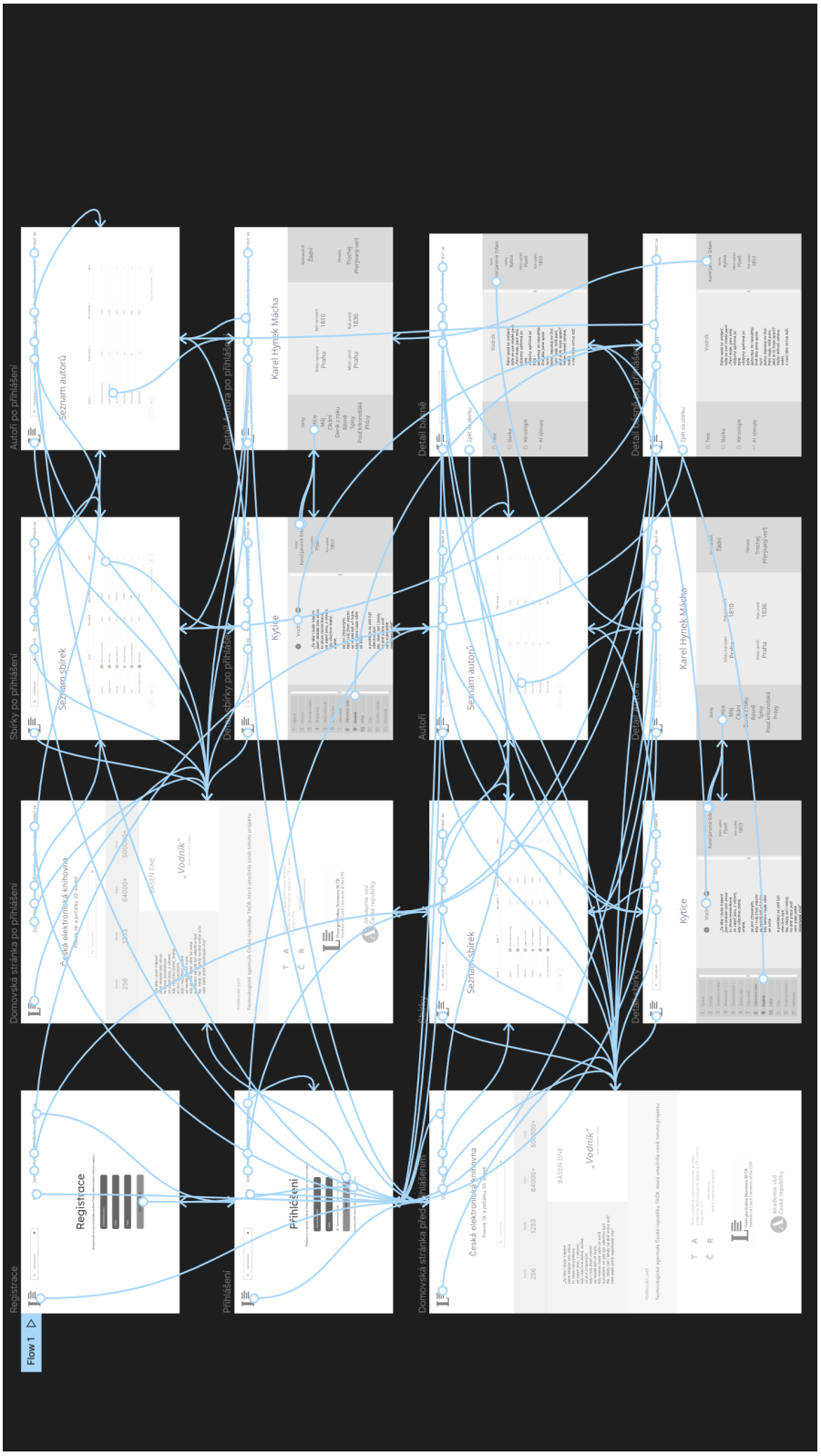
Pojmem prototyp lze označit i LOFI varianty wireframů. Často se však využívá právě pojmu wireframe pro model typu LOFI a pojmu prototyp pro model typu HIFI. V tomto kontextu se tedy jedná o model umožňující realistické uživatelské interakce, typicky pomocí klávesnice a myši. Jedná se o model s nejvyššími úrovněmi detailu, který je v rámci daného projektu vytvářen. Existují 3 typy HIFI prototypů [56]:

- Vizuální - Reprezentuje vysokou úroveň vizuálních detailů.
- Obsahový - Reprezentuje vysokou úroveň obsahu.
- Interaktivní - Reprezentuje variantu simulovatelnou člověkem.

3.2.9 HIFI

High-fidelity wireframe nebo taky HIFI je detailněji zpracovaná varianta LOFI [57]. Někdy se také využívá názvu high-tech, který indikuje vyšší množství technických detailů. To má samozřejmě za následek vyšší náklady na jeho výrobu. Na tomto modelu se už také nemohou podílet laici způsobem, jako tomu bylo u LOFI. Kvůli vysoké míře technických detailů, které s sebou často nesou implementaci úplné funkcionality budoucího systému, je nutná práce odborníka. Po vizuální stránce bývá tento model propracovanější a často využívá barev. Model je interaktivní, uživatel tedy může provádět příslušné akce a systém mu na ně odpoví. Toto teoretické finální řešení umožňuje návrháři provést uživatelské testování, které může odhalit značné množství chyb. Tyto chyby jsou složitější k opravě oproti LOFI, ale mnohem jednodušší než by tomu bylo až při implementaci. Jelikož HIFI slouží jako předloha pro implementaci, je tedy rozumné jej upravovat, dokud nejsou splněny požadavky klienta. Značně se tak sníží množství vytvořených chyb při implementaci. Obrázek zachycující interaktivitu vytvořeného HIFI prototypu lze vidět na obrázku 3.9. Prototyp je dostupný ve figmě[1], kde bylo také prováděno uživatelské testování (viz 3.2.10).

[1] `https://figma.fun/M8Nnd3`



Obrázek 3.9 Interaktivita HIFI prototypu

3.2.10 Uživatelské testování

Uživatelské testování (ang. Usability testing) nebo také testování použitelnosti má za cíl odhalit chyby sledováním chování uživatelů při práci se systémem. Této metody se využívá již v návrhu, což vede k přesnější tvorbě během implementace. Je lepší menší množství testování dříve než obrovské množství testování později [58]. Tuto metodu je vhodné použít i po nasazení aplikace. Neustále tak lze sledovat interakci uživatelů se systémem a problémy, které s ním uživatelé mají. Při větších projektech se tato metoda často používá každý měsíc, což vede ke skutečnému zkvalitnění poskytovaného systému.

Nejprve je nutné najít několik dobrovolníků. Poté zadat konkrétní úkoly, které uživatel splní. Důležité je zjistit, zda uživatel plní úkoly dostatečně rychle a efektivně. Při splnění úkolu musí uživatel také ze systému poznat, že úkol skutečně splnil. Podnětná je také spokojenost s rozložením prvků na stránce, respektive vzhledem celého systému. Neměl by být jen jednoduchý pro práci, ale taky zaujmout oko a líbit se. Na tyto aspekty jsme se zaměřili při tvorbě testovacích scénářů a k nim doplňujících otázek.

Testeři byli zvoleni náhodně, neboť problémy uživatelů se systémem, které má následující testování odhalit, může být testováno skutečně kýmkoliv. Dle [59] odhalí 3 testeři většinu uživatelských problémů a 5 testerů dokonce až 80 % těchto chyb. Větší množství testerů tedy nesplňuje očekávaný přírustek v nalezení množství chyb. Pro testování v této části vývoje jsme se tedy rozhodli pro 3 testery. K dispozici dostali prototyp, testovací scénáře (viz 3.6) a poté otázky k zodpovězení (viz 3.7).

Tabulka 3.6 Testovací scénáře

	Scénář
TS1	Prohlédněte si domovskou stránku a veškeré informace, které poskytuje.
TS2	Přejděte na stránku sbírky a zobrazte si detail sbírky Kytice, po prohlédnutí detailu sbírky přejděte na detail básně Vodník a po shlédnutí stránky přejděte zpět na detail sbírky.
TS3	Přejděte na stránku autoři a zobrazte si detail autora Karel Hynek Mácha, následně si stránku prohlédněte.
TS4	Přejděte na stránku přihlášení a přihlaste se. Nemáte-li vytvořen účet, tak se nejprve zaregistrujte.
TS5	Zopakujte testovací scénáře TS2 a TS3, následně přejděte na domovskou stránku a odhlašte se ze svého účtu.

Tabulka 3.7 Otázky na uživatele

	Otázka
O1	Jak se vám se systémem pracovalo?
O2	Jak hodnotíte rozložení prvků na stránkách a co byste případně změnili?
O3	Jak přínosné hodnotíte informace na domovské stránce a co byste přidali nebo naopak odebrali?
O4	Co říkáte na stránku sbírky a autoři? Zaměřte se na seznamy definované pomocí tabulek.
O5	Jak přehledné vám přišly stránky detail sbírky a detail básně?
O6	Jak přehledná vám přišla stránka detail autora?

Seznam testerů:

- Tester1 - Bc. Oleksandr Chmel, student FIT ČVUT.
- Tester2 - Martin Kulíšek, student FTVS UK.
- Tester3 - Nikola Vránová, studentka FSv ČVUT.

3.2.11 Zhodnocení testování a změny v návrhu

Odpovědi testerů na zadané otázky po splnění testovacích scénářů lze vidět v tabulce 3.8.

Tabulka 3.8 Odpovědi testerů

	Tester1	**Tester2**	**Tester3**
O1	Jednoduchý design, dobrá navigace.	Vše je na jednom místě, není těžké nic najít, přidal bych barvy.	Chvíli zabralo zorientovat se v seznamech sbírek. Celkově však přehledné a uživatelsky přívětivé.
O2	Chtělo by to zpětné tlačítka na seznamy sbírek i autorů.	Naprosto v pořádku.	Vyhledávač v navigačním baru je matoucí.
O3	Přehledné, jednoduché.	Hezky prezentované, neměnil bych je.	Jednoduché, vhodné.
O4	Funkčnost dobrá, ale ocenil bych více možností náhledu.	Do tabulek bych přidal další sloupec žánr, jinak v pořádku.	Tabulky vhodné, vyhledávač nesmyslně umístěn.
O5	Chtěl bych možnost číst na celé obrazovce.	Přehledné, nic bych neměnil.	Chytře rozdělené, jednoduché pro orientaci.
O6	Ocenil bych další informace k autorům.	Mohla by se přidat fotografie autora.	Přehlednost v pořádku, ale chybí více informací o autorovi.

Na základě provedeného uživatelského testování se dospělo k jistým změnám. První změnou je přidání zpětných tlačítek na stránkách detailu básně, detailu sbírky a detailu autora. Další změnou je umístění a vzhled vyhledávacího pole pro fulltextové vyhledávání. Vhodnější je jeho přesunutí na druhou stranu navigačního baru, kde již nenarušuje uživatelovu pozornost vůči ostatním prvkům. Změnil se také jeho vzhled v méně prostoru zabírající ikonku lupičky, na kterou lze kliknout. Až teprve po tomto kliknutí se objeví vyhledávací pole, do kterého lze psát. Ostatní výtky testerů, jako je doplnění více informací o autorovi, přidání fotky autora či přidání sloupce žánr na stránku sbírky jsou bohužel nesplnitelné z důvodu nedostatečného množství dodaných dat. Jelikož ale tyto testery zmíněné nedostatky nesouvisí s požadavky Ústavu pro Českou literaturu, není nutné se jimi dále zabývat.

3.2.12 Komunikace s Ústavem pro českou literaturu

Dohromady bylo vytvořeno větší množství druhů modelů, které reprezentují možnou podobu výsledného portálu dle požadavků Ústavu pro českou literaturu. Součástí byla sada lofi modelů, které sloužily jako jednoduchá představa o vzhledu výsledného webu. Komunikaci s Ústavem pro českou literaturu a tedy prezentaci mé práce měl na starosti Ing. Karel Klouda, Ph.D. Po prezentaci této sady Lofi modelů se ze stany Ústavu pro českou literaturu dospělo k jiné představě o vzhledu a fungování výsledného webu. Došlo tak ke změně požadavků a k vytvoření 6 drátěných modelů, které aktuální představu splňovaly.

Po další schůzce, kterou Ing. Karel Klouda, Ph.D. s Ústavem pro českou literaturu vedl, došlo na další úpravy vytvořených drátěných modelů a dodefinování dalších stránek s hrubou představou o informacích, které by měly obsahovat. Jednalo se o nové požadavky na stránky detail sbírky, detail básně nebo detail autora. Následně proběhla další iteraci tvorby drátěných modelů, kdy došlo k úpravám stávajících 6 stávajících modelů a vytvoření 13 nových modelů.

Jelikož se nedařilo delší dobu sjednat schůzku s Ústavem pro českou literaturu, aby poskytli požadovanou zpětnou reakci na nově vytvořené modely, byl vytvořen interaktivní prototyp, který sloužil jako věrná imitace naší přestavy o výsledném portálu. Tento model Ing. Karel Klouda, Ph.D. později představil Ústavu pro českou literaturu, který dodefinoval další, byť menší změny v návrhu.

Cílem vytvoření drátěných modelů byla jejich neustálá iterace až do podoby webu schválené zadavatelem. Následné vytvoření interaktivního prototypu mělo pramenit již z této schválené podoby portálu. Bylo ovšem nutné tento proces urychlit a interaktivní model vytvořit dříve. Stejně tak původní představa tvorby interaktivního modelu byla, stejně jako u tvorby drátěných modelů, jeho neustálé přepracovávání až do podoby schválené zadavatelem. V takové podobě by model sloužil jako vhodná předloha pro implementaci.

Poslední požadavky ze strany Ústavu pro českou literaturu tak bylo z časových důvodů nutné zanést až jako součást implementace. I přes mírně chaotický průběh však návrh posloužil k ujasnění představy o vzhledu a fungování jednotlivých stránek portálu.

3.3 Implementace

Tato kapitola se zabývá implementací. Zcela plynule navazuje na návrh, který jednoznačně definoval, co přesně je potřeba nově implementovat, a co je potřeba pouze upravit. Díky návrhu je tedy představa o výsledném vzhledu portálu přímočará. Nejprve se tato část práce zabývá popsáním architektur serverové a klientské části nového řešení. Závěrečná sekce se poté věnuje ukázce výsledného portálu.

3.3.1 Architektura backendu

Funkční backend byl vytvořen již dříve jako součást diplomové práce Ing. Tomáše Chvosty. Pro ukládání dat byla zvolena dokumentová databáze MongoDB. Důvodem této volby byla forma poskytnutých dat potřeba k nim přistupovat jako k dokumentům. Není to však nic překvapivého vzhledem k tomu, že poskytnutá data jsou sbírky básní. Přístup k databázi je realizován skrze NodeJS server. Z tohoto původního řešení lze stále využít některé již vytvořené endpointy. V tabulce 3.9 níže je popsán zdroj Kniha [60] vedoucí k datům ohledně básní.

Tabulka 3.9 Endpointy ke zdroji Kniha [60]

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.

Nové změny týkající se serverové části souvisí s novými stránkami. Jedná se především o stránky Autoři a Detail autora spojené s novou entitou Autor. Bohužel jsme stále od Ústavu pro českou literaturu neobdrželi slíbená data o autorech, a tak nebylo v rámci této práce možné požadované nové části backendu vytvořit.

Mezi nové změny v rámci serverové části také patří stránka Detail básně, která by měla umožnit zobrazení různorodých vrstev náhledu konkrétní básně. Jedná o vrstvy Text, Sbírka, Versologie a AI výstupy. Kromě vrstvy Text však není možné aktuálně pracovat na tvorbě dalších vrstev. Důvodem je čekání na data, která by měl poskytnout AI tým vedený Ing. Karlem Kloudou, Ph.D., který momentálně čeká na více dodaných dat ze strany Ústavu pro Českou literaturu.

Existují i další změny, které by bylo možné realizovat v rámci serverové části. Jednou z nich je informace o básni dne či doporučování obsahu. Všechny tyto záležitosti však opět spadají pod zmíněný AI tým, který opět čeká na dodaná data od Ústavu pro českou literaturu.

Dalším diskutovaným bodem je autentizace. Požadavky Ústavu pro českou literaturu se změnily takovým způsobem, že existují pouze uživatelské role neregistrovaného uživatele a přihlášeného uživatel. Přihlášený uživatel má navíc oproti neregistrovanému uživateli pouze přístup do sekce Pro experty, která není momentálně blíže definována. Ze schůzek však jasně vyplynulo, že se bude jednat pouze o stránku s nástroji, kterou bude moci používat kdokoliv, kdo si účet vytvoří. Myšlenka autentizace skrze JWT token a zajištění bezpečnosti uživatele zde tedy není aktuálním tématem. Prozatím bylo rozhodnuto, že se prozatimní funkční, byť nezabezpečená forma autentizace v rámci aplikace ponechá, neboť je více než dostačující.

3.3.2 Architektura frontendu

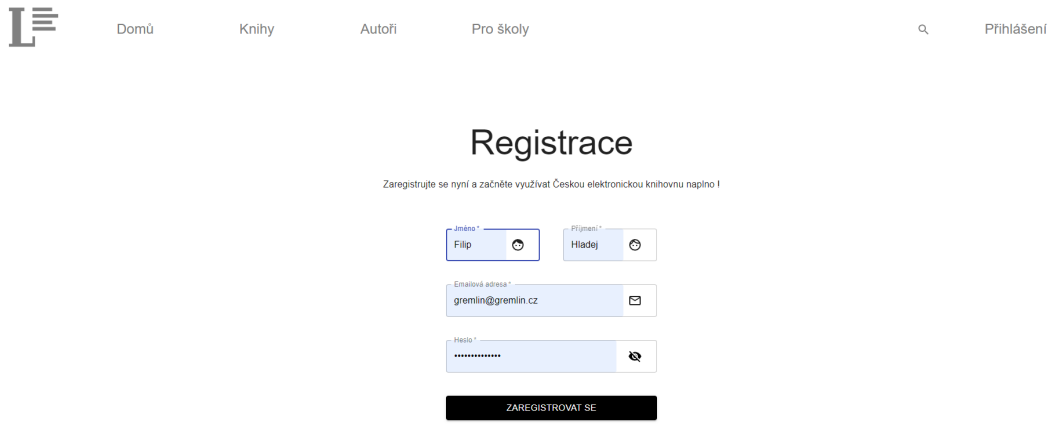
Uživatelská část byla taktéž vytvořena již dříve, a to v rámci mé bakalářské práce. Požadavky Ústavu pro českou literaturu však definovaly zcela jiný koncept výsledného portálu. Z tohoto důvodu byla klientská část kompletně předělána. Nejprve byly naimplementovány nové stránky umožňující registraci a přihlášení. Součástí implementace jsou také stránky Knihy, Detail sbírky a Detail básně. Tyto stránky vyjma Detailu básně interagují se serverovou částí a splňují tak potřebné požadavky. Stránky Autoři a Detail autora jsou byly také implementovány, ovšem jak již bylo dříve zmíněno, tyto stránky ze serverovou částí prozatím nekomunikují. Vytvořena byla také domovská stránka, která však taktéž zobrazuje pouze smyšlené data a prozatím nekomunikuje se serverovou částí aplikace.

K implementaci současného řešení je využito knihovny ReactJS. Díky této knihovně a principům VDOMu, na kterých je založena, bylo docíleno rychlých reakcí na interakce uživatele s webovou stránkou. Celá klienstká část nese prvky Material Designu. V každé implementované částí je určitým způsobem využito knihovny MaterialUI, která nabízí celou řadu výhod v znovupoužitelnost již hotových komponent. Další využitou knihovnou v rámci frontendu je Material Table. Tato knihovna nabídla jednoduché a režijně přijatelné řešení pro tvorbu tabulky, kterou lze najít na stránkách Knihy, Autoři a Detail autora. Jak již název knihovny napovídá, taktéž vychází z konceptu Material Designu.

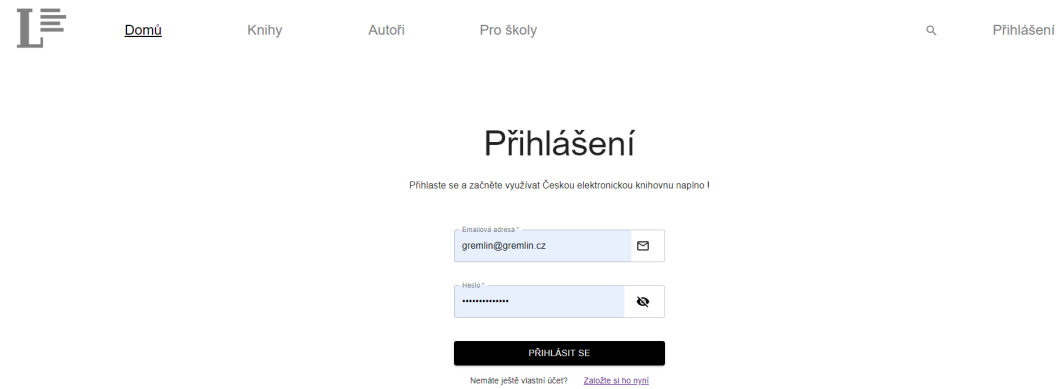
Vytvořeny byly také prozatím prázdné stránky Pro Školy a Pro Experty. Stránky budou doplněny o potřebné informace a funkcionality, a to ve chvíli, kdy dojde k jejich definici. To vše už bude však součástí tvorby mimo tuto závěrečnou práci.

Co se týče tématu responzivity, zde se na základě schůzek došlo k závěru, že se prozatím jedná o nepotřebnou funkcionalitu.

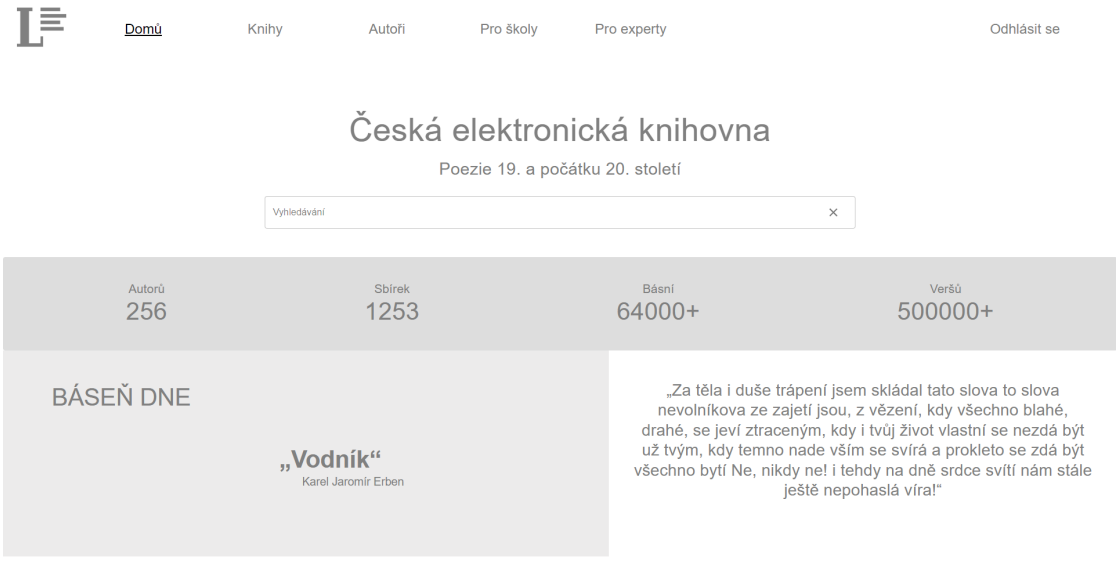
3.3.3 Ukázka aplikace



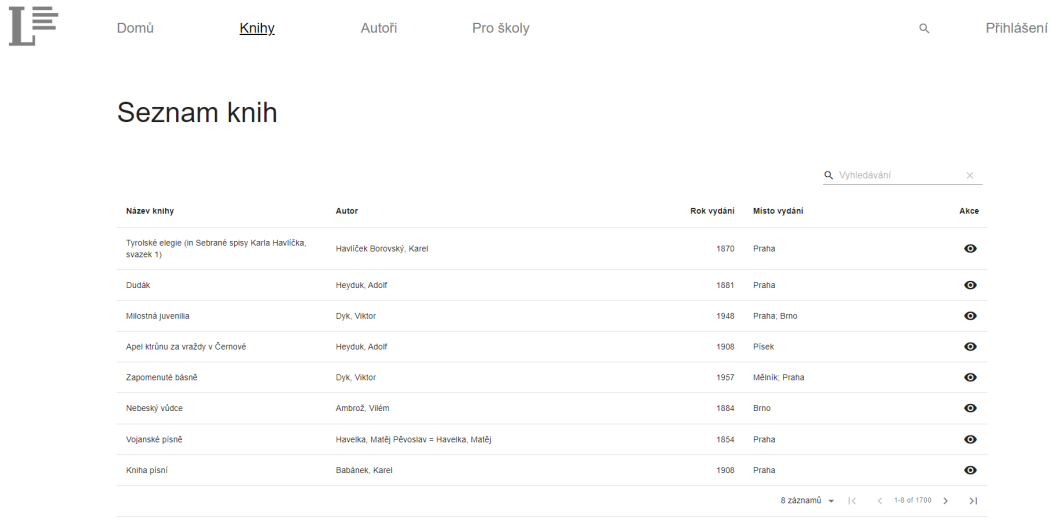
Obrázek 3.10 Registrační formulář



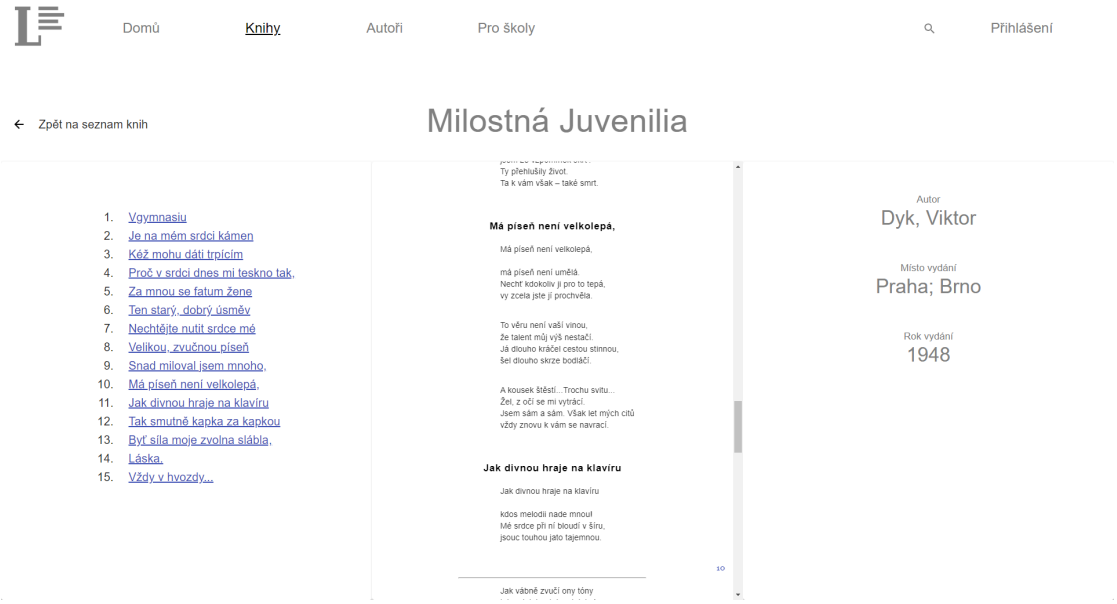
Obrázek 3.11 Přihlašovací formulář



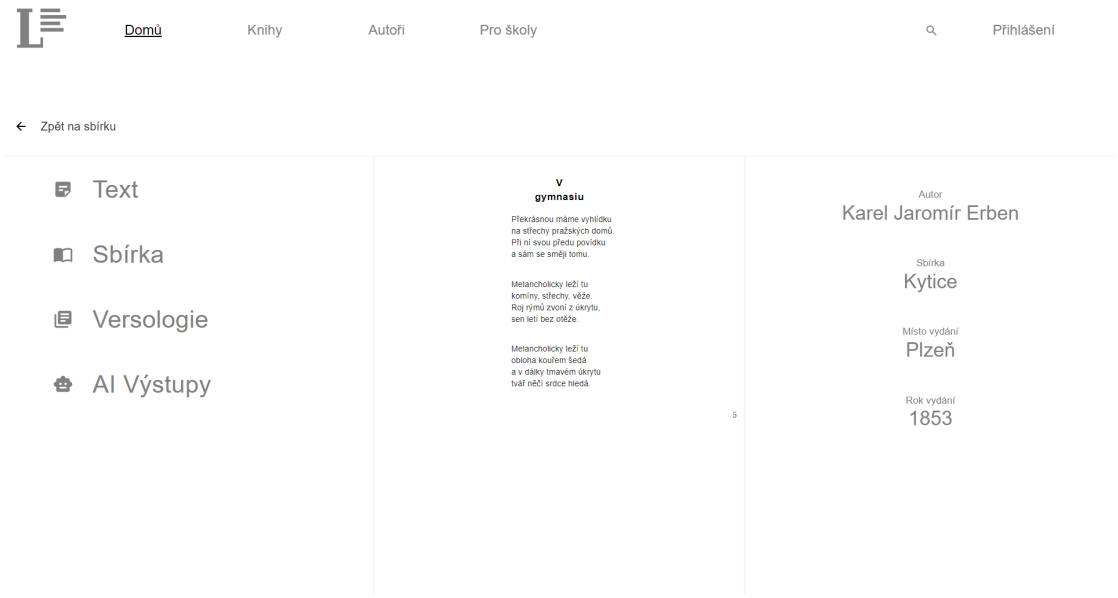
Obrázek 3.12 Domovská stránka



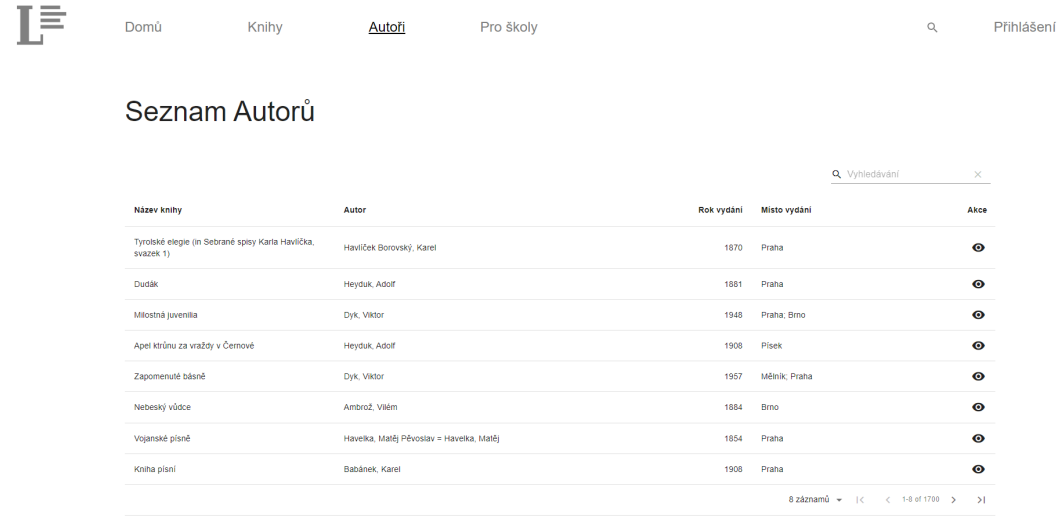
Obrázek 3.13 Stránka Knihy



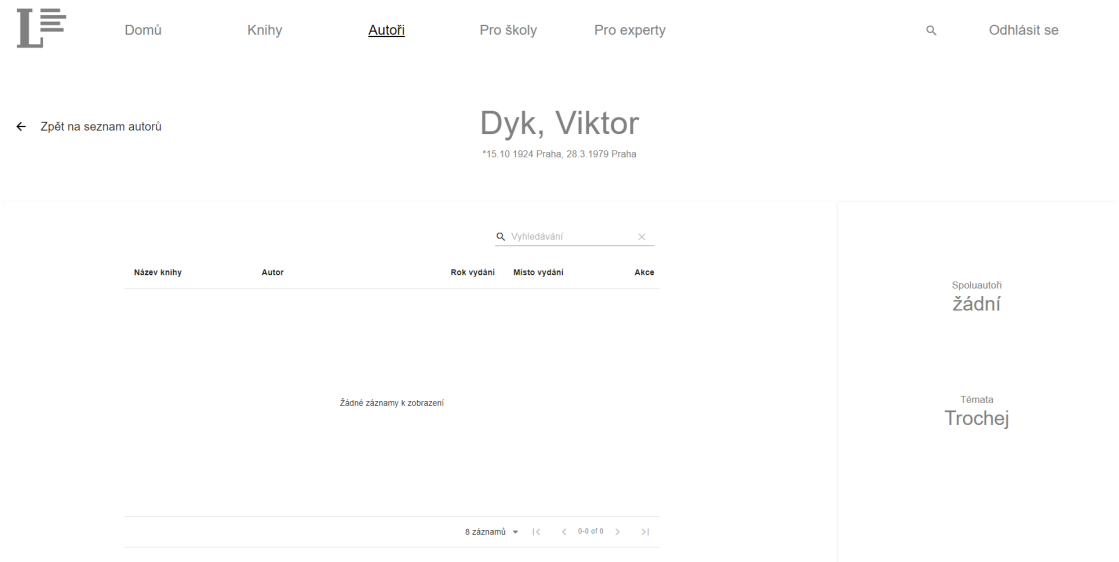
Obrázek 3.14 Detail sbírky



Obrázek 3.15 Detail básně



Obrázek 3.16 Stránka Autoři



Obrázek 3.17 Detail autora

3.4 Testování

Tato sekce se bude věnovat testování aplikace. Došlo již k testování interaktivního prototypu, díky kterému byly odhaleny jisté nedostatky. Tyto nedostatky byly již v rámci implementace odstraněny. Nyní je důležité otestovat aplikaci samotnou a ověřit, zda během implementace nevznikly další chyby. Jelikož se převážně bude jednat o testování designu výsledného portálu, tak se jako nejvhodnější způsob jeví metoda uživatelského testování. K tomuto účelu vznikly testovací scénáře.

3.4.1 Scénáře

V rámci uživatelského testování výsledné aplikace byli zvoleni 3 náhodní testeři, kteří by měli odhalit většinu chyb vzniklých při implementaci:

- Tester1 - Bc. Vít Pekárek, bývalý student FIT ČVUT Bc.
- Tester2 - Jakub Jurečka, student MUNI FSpS.
- Tester3 - Bc. Simona Procházková, dentální hygienistka.

Testeři dostanou testovací scénáře, které pokrývají celkový průchod aktuální verze aplikace. Otázky k zodpovězení jim v tomto případě poskytnuty nebudou. Cílem je zde pozorovat a naslouchat každé zpětné reakci, kterou testeři během plnění testovacích scénářů poskytnou. Vytvořené testovací scénáře lze vidět v tabulce 3.10.

Tabulka 3.10 Testovací scénáře

	Scénář
TS1	Prohlédněte si domovskou stránku. Poté se zaregistrujte a přihlašte se.
TS2	Najděte sbírku básní s názvem Zapomenuté básně. Následně přejděte na detail sbírky. Poté otestujte funkčnost obsahu sbírky. Následně se vraťte zpět na seznam knih.
TS3	Najděte libovolného autora. Následně přejděte na detail autora a po prohlédnutí stránky přejděte zpět na seznam autorů.
TS4	Přejděte do sekce Knihy a poté do detailu libovolné sbírky básní. Následně se vraťte zpět na seznam knih. Poté zopakujte TS4 s tím, že do seznamu knih se vraťte jiným způsobem, než který jste využili v prvním případě.
TS5	Zopakujte TS4 ze sekcí Autoři.
TS6	Odhlašte se z aplikace a následně přejděte do sekce Pro školy. Poté přijďte na způsob, jak se dostat do sekce Pro experty.
TS7	Přejděte do libovolné sekce vyjma domovské stránky. Následně se pokuste dostat se na domovskou stránku. Poté zopakujte TS7 s tím, že na domovskou stránku se vraťte jiným způsobem, než který jste využili v prvním případě.
TS8	Pokuste se něco vyhledat na domovské stránce. Následně přejděte do jiné sekce a zkuste opět využít vyhledávání.

3.4.2 Výsledek testování

Testování odhalilo chybu zpětného tlačítka v sekci detail sbírky, a to ve chvíli, kdy před jeho kliknutím došlo k manipulaci s obsahem sbírky. Zpětné tlačítko poté namísto směrování předchozí stránky odpovídající seznamu knih pouze vracelo zpět odkazy z obsahu sbírky. Další výtky směrem k portálu ze strany testerů se týkaly barev. Tento problém je momentálně, stejně jako většina těch zbývajících, v čekací fázi ohledně vyjádření ze strany Ústavu pro českou literaturu. Stejný problém je i se sekcemi Pro školy a Pro experty, na něž byly taktéž vznášeny poznámky během testování. Testeři odhalili také chybu, kdy během snahy přejít na detail autora, akce odpovídající tomuto přechodu nesla popis „číst knihu“. Toto zmatení se stalo ve 2 ze 3 případů.

Naopak pochválen byl jednoduchý design a rychlé reakce ze strany webové stránky. Konkrétně opět 2 ze 3 testerů pochválili proklik z přihlašovací formuláře na registrační jako jedinou možnou formu, jak se k registraci dostat. Líbilo se jim, jak je stránka jednoduše vede k cíli a není nutné přemýšlet, jak se do zmíněného místa vlastně dostat.

3.5 Nasazení

Posledním krokem je nasazení aplikace pomocí veřejně dostupné url adresy. K tomuto účelu nám byla zpřístupněna doména `fit.cvut.cz`. Po nahrání serverové a klientské části aplikace bylo nutné vyřešit, jakým způsobem bude řešení vystaveno. K tomuto účelu bylo využito procesového manažera PM2. Ten běží jako dlouhodobě spuštěný proces, který není v přímem kontaktu s uživatelem. Takovému typu procesu se také říká démon. Pomocí jednoduchých bash příkazů `pm2 start backend` a `pm2 start frontend`, které PM2 nabízí, byla serverová i klientská část spuštěny do doby, než dojde k jejich ukončení pomocí příkazu `pm2 stop` nebo dojde k nějaké chybě v jedné z těchto částí, která zapříčiní její ukončení. K takové situaci může typicky dojít při aktualizaci systému, která do něj přinese novou chybu.

PM2 se jeví také jako skvělý nástroj pro budoucí vývoj aplikace. Umožňuje totiž sledování chyb a vyjímek (ang. exceptions). Nabízí také logování v reálném čase, monitorování zvolených metrik a reporting. Tyto nástroje bude vhodné využít ve chvíli, kdy začně být aplikace doplněna o žádané funkcionality související s daty od AI týmu vedeným Ing. Karlem Kloudou, Ph.D. a nástroji, které s těmito daty budou v rámci aplikace pracovat.

Aplikace je dostupná na adrese `https://ucl-poezie.fit.cvut.cz/`.

3.6 Dokumentace

Dokumentace je součástí přílohy SD karty, stejně jako soubor s názvem `README.md`, který popisuje sestavení projektu. Dokumentace obsahuje popis všech modulů a komponent k nim příslušejících. Je v ní také popsáno, k čemu jednotlivé komponenty slouží. Jsou zde také vypsány všechny použité externí knihovny, které bylo v rámci projektu zapotřebí. Dokumentaci lze najít v příloze jako soubor s názvem `doc.md`.

Dokumentace by měla sloužit každému, který bude v budoucnu na projektu pracovat. Měla by dotyčnému pomoci zorientovat se v aplikaci a rychleji přijít na to, jak přesně aplikace funguje.

Momentálně je dohodnuto, že na projektu budu mimo tuto závěrečnou práci pokračovat já, a to i s ostatními členy, které jsem v rámci této závěrečné práce zmínil. V blízké budoucnosti tak bude dokumentace sloužit spíše našemu aktuálnímu týmu.

Kapitola 4

Závěr

Cílem této práce bylo vytvoření webového portálu, který by sloužil jako elektronická knihovna sbírek básní poezie konce 18. a 19. století pro Ústav pro českou literaturu Akademie věd České republiky.

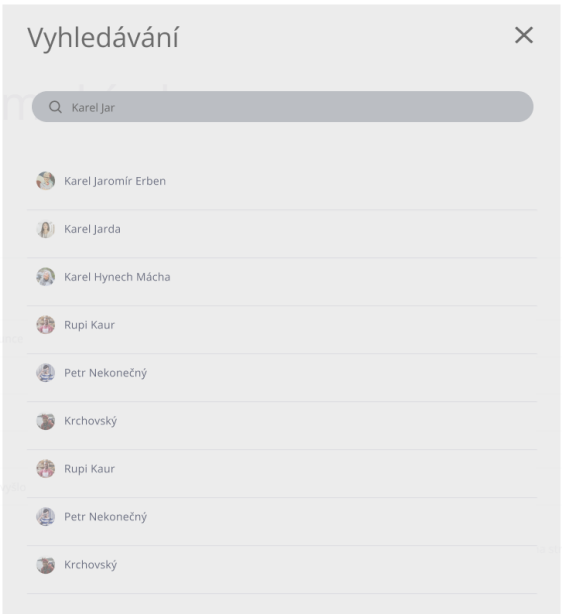
Úspěšně se povedlo analyzovat aktuální řešení a potřebné požadavky. Dále byl vypracován návrh, jehož průběh byl neustále diskutován pomocí schůzek Ústavu pro českou literaturu a Ing. Karla Kloudy, Ph.D. Během návrhu došlo k vytvoření diagramů, wireframů a interaktivního prototypu, který po testování a následném schválení reprezentoval aktuální představy Ústavu pro českou literaturu o možném vzhledu aplikace.

Na základě takto schválených vizualizací byla implementována nová klientská část komunikující s již dříve vytvořenou serverovou částí Ing. Tomášem Chvostou. Aplikace byla testována a nasazena. Závěrem bylo řešení vzniklé při této práci řádně zdokumentováno.

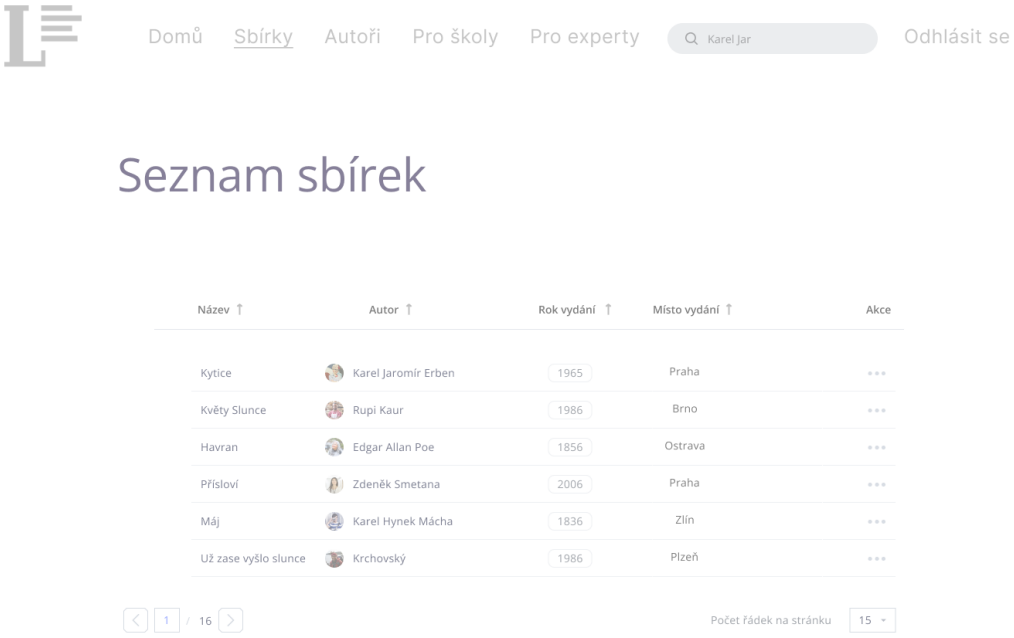
Veškeré cíle této práce byly naplněny. Jedná se však stále o nedokončený projekt, který již brzy čekají nové aktualizace. V rámci projektu TAČR a spolu s Ing. Tomášem Chvostou, Ing. Michalem Valentou, Ph.D. a Ing. Karlem Kloudou, Ph.D. se budu na tomto projektu nadále podílet.

Příloha A

Wireframy



Obrázek A.1 Modální okno vyhledávání



Obrázek A.2 Varianta vyhledávače

Bibliografie

1. *Frontend VS Backend – What's the Difference?* [online]. 2022. [cit. 2023-02-08]. Dostupné z: https://www.freecodecamp.org/news/frontend-vs-backend-whats-the-difference/.
2. *What is Frontend? What is Backend?* [online]. 2019. [cit. 2023-02-08]. Dostupné z: https://www.frontend-gmbh.de/en/blog/what-is-frontend-what-is-backend/. Section: Frontend.
3. MASTERADMINSTRATE. *What is design?* [online]. 2017. [cit. 2023-02-08]. Dostupné z: https://www.strate.education/gallery/news/design-definition.
4. STONE, Debbie; JARRETT, Caroline; WOODROFFE, Mark; MINOCHA, Shailey. *User Interface Design and Evaluation*. Elsevier, 2005. ISBN 978-0-08-052032-2. Google-Books-ID: VvSoyqPBPbMC.
5. EXPERIENCE, World Leaders in Research-Based User. *Nielsen Norman Group: UX Research, Training, and Consulting* [online]. [B.r.]. [cit. 2023-02-08]. Dostupné z: https://www.nngroup.com/articles/the-myth-of-the-genius-designer/.
6. EXPERIENCE, World Leaders in Research-Based User. *10 Usability Heuristics for User Interface Design* [online]. [B.r.]. [cit. 2023-02-08]. Dostupné z: https://www.nngroup.com/articles/ten-usability-heuristics/.
7. TAIVALSAARI, Antero; MIKKONEN, Tommi; INGALLS, Dan; PALACZ, Krzysztof. Web Browser as an Application Platform. In: *2008 34th Euromicro Conference Software Engineering and Advanced Applications*. 2008, s. 293–302. Dostupné z DOI: 10.1109/SEAA.2008.17. ISSN: 2376-9505.
8. DESIGNS, Essential. *Website vs Web App: What's the Difference?* [online]. 2019. [cit. 2023-02-09]. Dostupné z: https://medium.com/@essentialdesign/website-vs-web-app-whats-the-difference-e499b18b60b4.
9. *HTML — Definition & Facts — Britannica* [online]. 2023. [cit. 2023-02-08]. Dostupné z: https://www.britannica.com/technology/HTML.
10. S, Astari. *What Is HTML? Hypertext Markup Language Basics Explained* [online]. 2018. [cit. 2023-02-08]. Dostupné z: https://www.hostinger.com/tutorials/what-is-html.
11. *HTML & CSS - W3C* [online]. [B.r.]. [cit. 2023-02-08]. Dostupné z: https://www.w3.org/standards/webdesign/htmlcss.html.
12. *What is CSS? - Learn web development — MDN* [online]. 2022. [cit. 2023-02-08]. Dostupné z: https://developer.mozilla.org/en-US/docs/Learn/CSS/First_steps/What_is_CSS.

13. *Basic concepts of flexbox - CSS: Cascading Style Sheets — MDN* [online]. 2022. [cit. 2023-02-08]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Flexible_Box_Layout/Basic_Concepts_of_Flexbox.
14. *CSS Grid Layout - CSS: Cascading Style Sheets — MDN* [online]. 2022. [cit. 2023-02-08]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Grid_Layout.
15. LAWRENCE, Matt. *What is a CSS Framework?* [online]. 2019. [cit. 2023-02-08]. Dostupné z: https://medium.com/html-all-the-things/what-is-a-css-framework-f758ef0b1a11.
16. GARDNER, Brett S. Responsive web design: Enriching the user experience. *Sigma Journal: Inside the Digital Ecosystem*. 2011, roč. 11, č. 1, s. 13–19.
17. WILTON, Paul. *Beginning JavaScript*. John Wiley & Sons, 2004. ISBN 978-0-7645-5855-9.
18. *Types of attacks - Web security — MDN* [online]. 2022. [cit. 2023-02-08]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/Security/Types_of_attacks.
19. GARRETT, Jesse James et al. Ajax: A new approach to web applications. 2005.
20. *Getting started - Developer guides — MDN* [online]. 2023. [cit. 2023-02-08]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/Guide/AJAX/Getting_Started.
21. MIKOWSKI, Michael; POWELL, Josh. *Single Page Web Applications: JavaScript end-to-end*. Simon a Schuster, 2013. ISBN 978-1-63835-134-4. Google-Books-ID: eDszEAAAQBAJ.
22. *React – A JavaScript library for building user interfaces* [online]. [B.r.]. [cit. 2023-02-08]. Dostupné z: https://reactjs.org/.
23. *Document Object Model (DOM) - Web APIs — MDN* [online]. 2023. [cit. 2023-02-08]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model.
24. *Virtual DOM and Internals – React* [online]. [B.r.]. [cit. 2023-02-08]. Dostupné z: https://reactjs.org/docs/faq-internals.html.
25. *Back-End Web Architecture* [online]. [B.r.]. [cit. 2023-02-08]. Dostupné z: https://www.codecademy.com/article/back-end-architecture.
26. MARTIN, Matthew. *What is Backend Developer? Skills Need for Web Development* [online]. 2020. [cit. 2023-02-08]. Dostupné z: https://www.guru99.com/what-is-backend-developer.html.
27. NODE.JS. *About* [online]. [B.r.]. [cit. 2023-02-08]. Dostupné z: https://nodejs.org/en/about/.
28. *What is an event loop in JavaScript?* [online]. [B.r.]. [cit. 2023-02-17]. Dostupné z: https://www.educative.io/answers/what-is-an-event-loop-in-javascript.
29. *What Is NoSQL? NoSQL Databases Explained* [online]. [B.r.]. [cit. 2023-02-08]. Dostupné z: https://www.mongodb.com/nosql-explained.
30. *What Is MongoDB? — MongoDB* [online]. [B.r.]. [cit. 2023-02-09]. Dostupné z: https://www.mongodb.com/what-is-mongodb.
31. *What Is MongoDB?* [online]. [B.r.]. [cit. 2023-02-08]. Dostupné z: https://www.mongodb.com/what-is-mongodb.
32. *What is an API? - API Beginner's Guide - AWS* [online]. [B.r.]. [cit. 2023-02-08]. Dostupné z: https://aws.amazon.com/what-is/api/.
33. *An overview of HTTP - HTTP — MDN* [online]. 2023. [cit. 2023-02-08]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview.
34. *HTTP response status codes - HTTP — MDN* [online]. 2023. [cit. 2023-02-08]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/HTTP/Status.

35. *HTTP request methods - HTTP — MDN* [online]. 2022. [cit. 2023-02-08]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods.
36. *What is REST?* [online]. [B.r.]. [cit. 2023-02-08]. Dostupné z: https://www.codecademy.com/article/what-is-rest.
37. *XML.com: What is XML?* [online]. [B.r.]. [cit. 2023-02-08]. Dostupné z: http://www.ce.unipr.it/people/bianchi/Teaching/IntelligenzaArtificiale/rdf_pl/XML-RDF/xmlguide1.html.
38. SMITH, BEN. *Beginning JSON*. Apress, 2015. ISBN 978-1-4842-0202-9. Google-Books-ID: ZYYnCgAAQBAJ.
39. *JSON* [online]. [B.r.]. [cit. 2023-02-08]. Dostupné z: https://www.json.org/json-en.html.
40. *Web Authentication API - Web APIs — MDN* [online]. 2023. [cit. 2023-02-08]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/Web_Authentication_API.
41. *4 Most Used REST API Authentication Methods* [online]. 2019. [cit. 2023-02-08]. Dostupné z: http://blog.restcase.com/4-most-used-rest-api-authentication-methods/.
42. *OAuth 2.0 — OAuth* [online]. [B.r.]. [cit. 2023-02-09]. Dostupné z: https://oauth.net/2/.
43. *OpenID Connect — OpenID* [online]. 2011. [cit. 2023-02-09]. Dostupné z: https://openid.net/connect/.
44. AUTH0.COM. *JWT.IO - JSON Web Tokens Introduction* [online]. [B.r.]. [cit. 2023-02-08]. Dostupné z: http://jwt.io/.
45. *What Are Internet Cookies and How Are They Used?* [online]. 2015. [cit. 2023-02-09]. Dostupné z: https://allaboutcookies.org/what-is-a-cookie.
46. AZIZ, Arslan; TELANG, Rahul. What is a cookie worth? In: *WEIS*. 2015, s. 3.
47. *What are Cookies?* [online]. 2022. [cit. 2023-02-08]. Dostupné z: https://www.kaspersky.com/resource-center/definitions/cookies. Section: Resource Center.
48. KAPOOR, Shray. Session hijacking exploiting TCP, UDP and HTTP sessions. *infosecwriters. com/text_resources/.../SKapoor_SessionHijacking. pdf*. 2006.
49. *Window.localStorage - Web APIs — MDN* [online]. 2023. [cit. 2023-02-08]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/Window/localStorage.
50. *What are microservices?* [online]. [B.r.]. [cit. 2023-02-08]. Dostupné z: http://microservices.io/index.html.
51. *What are Microservices? — AWS* [online]. [B.r.]. [cit. 2023-02-08]. Dostupné z: https://aws.amazon.com/microservices/.
52. *User Goals — Usability & Web Accessibility* [online]. [B.r.]. [cit. 2023-02-20]. Dostupné z: https://usability.yale.edu/understanding-your-user/user-goals?fbclid=IwAR1-CZRV_7NpPMjxv4XHU99-WDNzTOyPtu1j85L-HqO1oQEFDFWktFvfuGM.
53. *What is a Use Case?* [online]. [B.r.]. [cit. 2023-02-20]. Dostupné z: https://www.techtarget.com/searchsoftwarequality/definition/use-case.
54. *Workspaces, task groups, and tasks* [online]. [B.r.]. [cit. 2023-02-20]. Dostupné z: https://help.hcltechsw.com/commerce/7.0.0/com.ibm.commerce.workspaces.doc/concepts/cwoparts.html.
55. *Low fidelity vs high fidelity wireframes: what's the difference?* [online]. [B.r.]. [cit. 2023-02-23]. Dostupné z: https://www.justinmind.com/wireframe/low-fidelity-vs-high-fidelity-wireframing-is-paper-dead.

56. BABICH, Nick. *Prototyping 101: The Difference between Low-Fidelity and High-Fidelity Prototypes and When to Use Each* [online]. [B.r.]. [cit. 2023-02-23]. Dostupné z: https://blog.adobe.com/en/publish/2017/11/29/prototyping-difference-low-fidelity-high-fidelity-prototypes-use.

57. *Low Fidelity Wireframe Template & Example for Teams — Miro* [online]. [B.r.]. [cit. 2023-02-23]. Dostupné z: https://miro.com/templates/low-fidelity-wireframes/.

58. *The well-kept secret behind great UX: Usability Testing* [online]. 2018. [cit. 2023-02-27]. Dostupné z: https://www.freecodecamp.org/news/the-well-kept-secret-behind-great-ux-usability-testing-b788178a64c3/.

59. COLLINS, Peter. *How many testers in usability testing is enough?* [online]. [B.r.]. [cit. 2023-02-27]. Dostupné z: https://info.webusability.co.uk/blog/usability-testing/how-many-testers-are-enough.

60. HLADEJ, Filip. Návrh a implementace frontend pro projekt Česká elektronická knihovna [online]. 2021 [cit. 2023-03-27]. Dostupné z: https://dspace.cvut.cz/handle/10467/95373. Accepted: 2021-06-17T22:51:37Z Publisher: České vysoké učení technické v Praze. Vypočetní a informační centrum.

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` ..... modely aplikace
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF
  - `doc` ..... dokumentace klientské části