Masarykova univerzita
Fakulta informatiky

Návrh UI a vývoj aplikácie pre editáciu a správu terminológie

Diplomová práca

Bc. Ján Mach

Brno, jeseň 2020

Masarykova univerzita
Fakulta informatiky

Návrh UI a vývoj aplikácie pre editáciu a správu terminológie

Diplomová práca

Bc. Ján Mach

Brno, jeseň 2020

Na tomto mieste sa v tlačenej práci nachádza oficiálne podpísané zadanie práce a vyhlásenie autora školského diela.

Vyhlásenie

Vyhlasujem, že táto diplomová práca je mojím pôvodným autorským dielom, ktoré som vypracoval samostatne. Všetky zdroje, pramene a literatúru, ktoré som pri vypracovaní používal alebo z nich čerpal, v práci riadne citujem s uvedením úplného odkazu na príslušný zdroj.

Bc. Ján Mach

Vedúci práce: RNDr. Adam Rambousek, Ph.D.

Poďakovanie

Ďakujem vedúcemu práce, RNDr. Adamovi Rambouskovi, Ph.D., Mgr. et Mgr. Marii Starej ako aj ostatným kolegom z Centra spracovania prirodzeného jazyka FI MU za sprostredkovanie zaujímavej témy, cenné rady a ochotu.

Zhrnutie

Cieľom práce je vytvorenie webovej aplikácie pre editáciu a správu terminológie rozličných verejných inštitúcií na základe zadaných funkčných a nefunkčných požiadaviek. Práca sa zaoberá opisom riadenia vývoja systému, analýzou požiadaviek klienta, návrhom užívateľského rozhrania – tvorbou wireframov – a implementáciou front-endu webových aplikácií. Súčasťou práce je porovnanie a výber rôznych technológií určených pre tvorbu bežných webových aplikácií. Spomedzi Angularu, Vue.js, Material UI, Ant Designu som vybral React a Semantic UI, pomocou ktorých som implementoval požadovanú funkcionalitu.

Kľúčové slová

Tezaurus, webová aplikácia, front-end, JavaScript, React, UI, UX

Obsah

Úvod **1**

1 Systém webových tezaurov **2**
1.1 Funkčné a nefunkčné požiadavky 3
1.2 Užívateľské roly 4
1.2.1 Aplikácia centrálneho systému 4
1.2.2 Aplikácie jednej inštancie tezauru 6
1.3 Riadenie vývoja systému 10

2 Návrh užívateľského rozhrania **12**
2.1 UI vs UX 14
2.2 Drôtené modely 15
2.2.1 Navigačná lišta 15
2.2.2 Vstupná stránka 16
2.2.3 Zoznam tezaurov 18
2.2.4 Stránka nastavení 19
2.2.5 Pokročilé vyhľadávanie 21
2.2.6 Detail jedného termínu 23

3 Použité technológie **26**
3.1 React 27
3.1.1 Knižnica Redux 29
3.1.2 Knižnica Informed 30
3.1.3 Ďalšie nástroje z ekosystému Reactu 30
3.2 TypeScript 32
3.3 Komponentový framework 33
3.3.1 Ikony 35

4 Implementácia **36**
4.1 Architektúra projektu 36
4.1.1 Enviroment variables 36
4.1.2 DevOps 37
4.1.3 Hierarchia súborov 37
4.1.4 Štruktúra React komponent 38
4.1.5 Komponent editácie termínu 39
4.2 Zmeny podľa spätnej väzby 40

Záver **41**

Bibliografia **42**

A Funkčné a nefunkčné požiadavky **44**

B Príručky pre používateľov aplikácie **45**

C Zdrojové kódy **46**

Zoznam obrázkov

1.1 Prípady použitia aplikácie centrálneho systému 5
1.2 Prípady použitia jednej inštancie tezauru 8

2.1 Walterova hierarchia užívateľských potrieb 12
2.2 Mockup navigačnej lišty 15
2.3 Mockup responzívnej verzie navigačnej lišty 16
2.4 Mockup vstupnej stránky inštancie tezauru 17
2.5 Mockup zoznamu registrovaných slovníkov 18
2.6 Mockup stránky zmeny grafických nastavení 20
2.7 Mockup stránky pokročilého vyhľadávania termínov 22
2.8 Mockup stránky detailu jedného termínu 24

4.1 Hierarchia súborov aplikácie 38
4.2 Editačný formulár so zvýraznenými poľami 39

Úvod

Projekt webových tezaurov Centra spracovania prirodzeného jazyka Fakulty informatiky Masarykovej univerzity v Brne má za cieľ vytvorenie webovej aplikácie tezauru – výkladového slovníka – a oddelenej aplikácie pre správu jej rozličných inštancií. Účelom je tvorba webových aplikácií prístupných cez prehliadač bez nutnosti inštalovať si akýkoľvek špecializovaný software. Inštancie vytvorenej aplikácie tezauru budú použité vo verejných inštitúciách pre editáciu a zverejňovanie oficiálnej odbornej viacjazyčnej terminológie. Súčasťou požiadaviek zadávateľa projektu je, aby každá užívateľská inštitúcia mala možnosť prispôsobiť si vlastnú inštanciu aplikácie slovníka podľa svojich predstáv a potrieb, či už definíciou vlastných terminologických odborov a skupín, do ktorých sa termíny môžu zoskupovať alebo upraviť si grafickú podobu aplikácie slovníka svojej vizuálnej identite.

Mojou úlohou bolo riadenie čiastkového projektu pre tvorbu webového rozhrania, tvorba grafického návrhu užívateľského rozhrania a implementácia front-endu aplikácií. Riadenie spočívalo v komunikácii so zákazníkom, analýze jeho požiadaviek ako aj v plánovaní a rozdelení programátorských prác vývoja rozhrania a tvorby užívateľskej a editorskej príručky. Vytvoril som grafické návrhy, takzvané mockupy, užívateľského rozhrania pre každú významnú podstránku aplikácie, pričom som sa sústredil hlavne na kvalitu UX designu.

Kľúčovou a najnáročnejšou činnosťou v rámci diplomovej práce bola implementácia front-endu webových aplikácií. Tá pozostávala z kvalifikovaného výberu vhodných technológií a frameworkov, návrhu architektúry front-endu webových aplikácií, implementácie ich prototypu a zberu spätnej väzby od užívateľov. Na základe analýzy reakcií a požiadaviek sme postupne vytvorili niekoľko iterácií aplikácií a finálnu sme nasadili na produkčné prostredie zákazníka.

1 Systém webových tezaurov

Tezaurus je vo všeobecnosti pomerne úplný výkladový slovník hierarchicky usporiadanej odborovej terminológie. Zachytáva vzťahy medzi lexikálnymi jednotkami, zvyčajne obsahuje taktiež zoznam synoným, niekedy aj antoným či homoným. Môže sa využívať pre vyhľadávanie vhodných slov či informácií bez nutnosti poznať presný odborný termín. Označenie „tezaurus" vzniklo z latinského slova, ktoré znamená poklad či pokladnica. [1]

Zákazníkom pre vývoj systému webových tezaurov je Ministerstvo vnútra Českej republiky. Celý komplexný systém je rozdelený do dvoch samostatných aplikácií: aplikácie centrálneho systému a aplikácie jednej inštancie tezauru.

Aplikácia centrálneho systému tezaurov

Jedná sa o systém centralizovaného zoznamu všetkých tezaurov spadajúcich pod rovnakú organizáciu, ktorá ich spravuje. Účelom tejto aplikácie je len jednoduché združenie jednotlivých inštancií výkladových slovníkov do jedného zoznamu. Vďaka tomu sa dá jednoducho vyhľadávať konkrétny špecializovaný tezaurus a v prípade jeho výpadku môžu zamestnanci správcovskej organizácie jednoducho nájsť kontaktné údaje na osoby zodpovedné za danú problémovú inštanciu slovníka.

Pre každý tezaurus je uvedená jeho webová URL adresa, názov organizácie, ktorá ho spravuje, dátum a čas jeho registrácie – pridania do centrálneho systému –, celkový počet termínov, ktoré obsahuje, dátum a čas poslednej úpravy jeho informácií a krátky slovný popis. Tieto základné údaje vidia všetci, aj neprihlásení užívatelia. Prihlásený užívateľ s adekvátnym administrátorským oprávnením vidí taktiež jedinečné ID aplikácie inštancie tezauru, meno, kontaktný telefón a email osoby, ktorá je zodpovedná za jeho správu a korektný chod. Taktiež môže pridávať, mazať a vykonávať úpravy uložených informácií o jednotlivých inštanciách tezaurov.

Aplikácia jednej inštancie tezauru

Jedná sa o samostatnú aplikáciu tezauru, ktorej inštancie sa môžu registrovať do centrálneho systému. Obsahuje všetku potrebnú funkcionalitu pre zobrazovanie, vyhľadávanie, vytváranie, upravovanie a mazanie termínov, správu užívateľov, funkcionalitu notifikácií pre prihlásených užívateľov o zmenách v systéme a funkcie pre správu a menenie nastavení seba samého.

Základnou funkciou aplikácie je takzvané napovedacie pole vyhľadávania, s ktorého pomocou môžu užívatelia jednoducho a rýchlo prehľadávať obsah slovníka. Aplikácia taktiež podporuje pokročilé vyhľadávanie či filtrovanie termínov a možnosť stiahnuť si informácie v rozličných dátových formátoch. Podporuje taktiež pokročilé možnosti prispôsobenia, napríklad konfiguráciu zoznamu jazykov, do ktorých následne umožňuje prekladanie termínov. Ku každému termínu je taktiež možné pridávanie dodatočných údajov – definícií, odkazov na nadradené pojmy, synonymá, antonymá, homonymá, obrázky, videá, odkazy do iných tezaurov a redakčné poznámky. Každá definícia môže obsahovať mimo jej samotného textu aj zdroj/citáciu odkiaľ definícia pochádza, zdroj zo zákona, vzorec, tabuľku, odbor, druh, typ, doplňujúce informácie a neverejnú poznámku. Každý záznam termínu v tezaure je taktiež možné označiť, že sa jedná o skratku.

Aplikácia podporuje taktiež možnosť jednoduchého menenia vizuálnej identity – farebnej palety a loga prevádzkovateľa – za účelom prispôsobenia a odlíšenia jednotlivých inštancií tezaurov. Cieľom je snaha minimalizovať pravdepodobnosť ich zamieňania a prípadného zmätenia užívateľov pri súbežnej práci s viacerými rôznymi inštanciami tezaurov.

1.1 Funkčné a nefunkčné požiadavky

Prvým krokom pri začínajúcom projekte vývoja softvéru je definovanie požiadaviek a očakávaní, ktoré klient od systému má. Požiadavky je možné deliť na funkčné a nefunkčné.

Funkčné požiadavky sa zameriavajú na to, čo by systém mal, prípadne nemal robiť, definujú, ako by mal systém reagovať na rozličné vstupy či situácie, t. j. hovoria o funkcionalite systému. Nefunkčné

požiadavky definujú vlastnosti a obmedzenia systému ako celku. Patria medzi ne napríklad požiadavky na dostupnosť, spoľahlivosť či bezpečnosť a podobne.

Zoznam funkčných a nefunkčných požiadaviek systému webových tezaurov sa nachádza v prílohe.

1.2 Užívateľské roly

Celý systém má možnosť používať mnoho rôznych užívateľov, a to nielen registrovaných, ale aj neregistrovaných. Pre zabezpečenie jeho efektívneho fungovania je nutné, aby jeho správcovia jednoznačne definovali právomoci a zodpovednosti jednotlivých užívateľov či zamestnancov. Za týmto účelom nám boli poskytnuté zjednodušené zoznamy jednotlivých pracovných pozícií zamestnancov s ich pracovnými povinnosťami, ktorí budú so systémom interagovať. Jeden užívateľ môže mať pridelených aj niekoľko rolí. Pri každom prístupe do aplikácie systém automaticky rozoznáva, o akého užívateľa sa jedná, a uspôsobí užívateľské rozhranie pre jeho jedinečné potreby. Aplikácie centrálneho systému a jednej inštancie tezauru rozoznávajú rozdielne užívateľské roly a ich definície sa líšia.

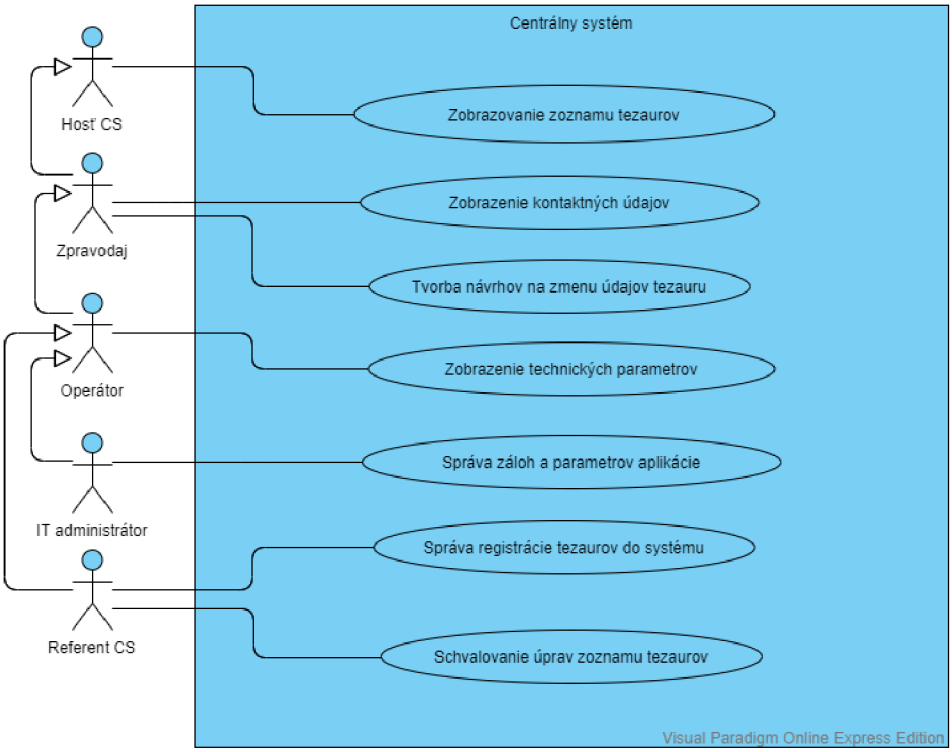
1.2.1 Aplikácia centrálneho systému

- Referent centrálneho systému (CS)
  - » spravuje zaregistrovanie a odregistrovanie dátového systému do/z centrálneho systému
  - » schvaľuje úpravy zoznamu centrálneho systému
- Operátor prevádzky centrálneho systému
  - » operátor sleduje a zabezpečuje plynulú prevádzku IT systému
- IT administrátor centrálneho systému
  - » správa záloh, nastavení a zmien parametrov aplikácie
- Spravodajca centrálneho systému

  - » spravodaj s možnosťou pasívneho sledovania obsahu a stavu CS
  - » má možnosť zobrazenia kontaktných informácií jednotlivých užívateľov

- Hosť CS
  - » užívateľ majúci prístup k užívateľskej časti CS
  - » zobrazovanie len verejných informácií

Z týchto údajov bol vypracovaný zoznam prípadov použitia a definované jednotlivé prístupové roly k systému.



Obr. 1.1: Prípady použitia aplikácie centrálneho systému

Bližšie informácie k jednotlivým prípadom použitia:

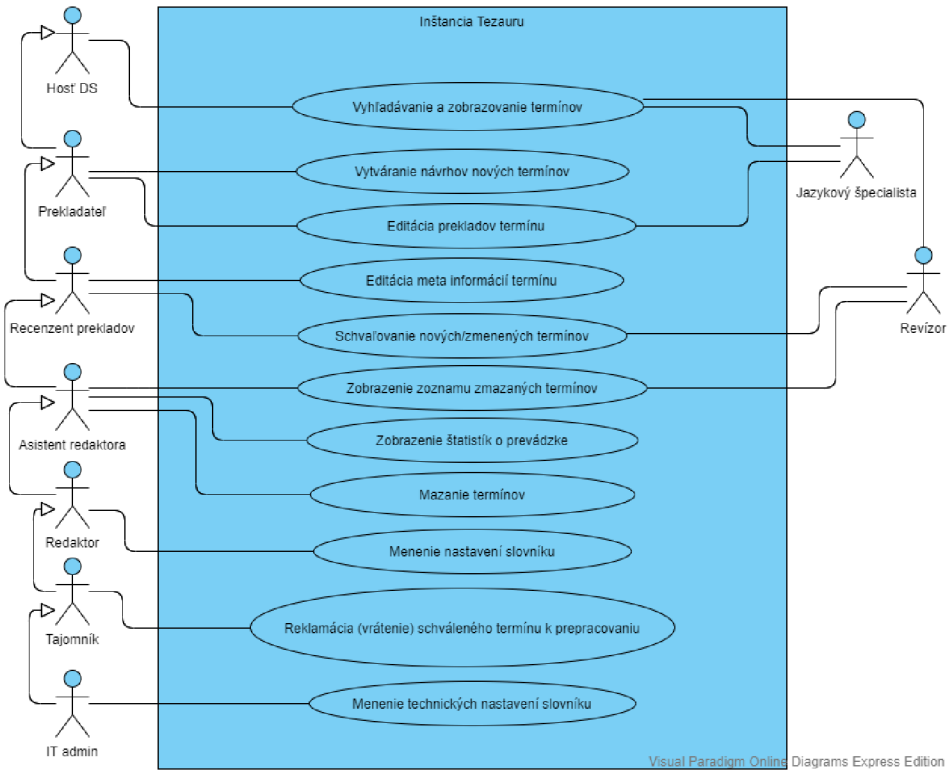
1. **Zobrazenie zoznamu tezaurov** – možnosť zobrazenia zoznamu všetkých registrovaných tezaurov a základných informácií o nich.
2. **Zobrazenie kontaktných údajov** – pridanie kontaktných údajov na zodpovedné osoby k základným údajom každého tezauru.
3. **Tvorba návrhov na zmenu údajov tezauru** – funkcionalita zaslania požiadavky na zmenu uložených údajov jednotlivých tezaurov.
4. **Zobrazenie technických parametrov** – možnosť zobrazenia technických údajov systému.
5. **Správa záloh a parametrov aplikácie** – menenie technických nastavení aplikácie.
6. **Správa registrácie tezaurov do systému** – možnosť pridávať, meniť a mazať jednotlivé inštancie tezaurov.
7. **Schvaľovanie úprav zoznamu tezaurov** – funkcionalita schvaľovania žiadostí o úpravy, ktoré boli zaslané užívateľmi.

1.2.2 Aplikácie jednej inštancie tezauru

- IT administrátor dátového systému
  - » IT zabezpečenie chodu aplikácie
- Operátor prevádzky dátového systému
  - » operátor sledujúci prevádzku IT systému
- Hlavný redaktor
  - » schvaľovanie vystavenia termínu
- Asistent hlavného redaktora
  - » rutinná agenda chodu (priraďovanie rolí užívateľom, spracovanie štatistík apod.)
- Redaktor odboru

  - » tvorca termínu a výkladov v češtine
- Recenzent odboru
  - » kontrola termínov a výkladov po odbornej stránke
- Prekladateľ jazyka
  - » tvorca prekladov termínov a výkladov v jednotlivých jazykoch
- Recenzent prekladov jazyka
  - » recenzenti prekladov termínov a výkladov v jednotlivých jazykoch
- Jazykový špecialista
  - » špecialista na českú jazykotvorbu (tvorba a kontrola jazykových novotvarov v českom jazyku)
- Jazykový korektor
  - » bežná jazyková korektúra výkladov v českom jazyku
- Revízor
  - » osoba, ktorá má na starosti celkovú kontrolu a validáciu obsahu tezauru
- Tajomník terminologickej komisie
  - » prístup k štatistikám, informáciám o prevádzke a obsahu
  - » tlmočí stanoviská a názory terminologickej komisie
- Hosť tezauru
  - » vyhľadávať a zobraziť informácie k jednotlivým termínom
  - » registrovaný či neregistrovaný užívateľ

Rovnako ako v prípade centrálneho systému bol z týchto údajov vypracovaný zoznam prípadov použitia a definované jednotlivé prístupové roly k systému.



Obr. 1.2: Prípady použitia jednej inštancie tezauru

Bližšie informácie k jednotlivým prípadom použitia:

1. **Vyhľadávanie a zobrazovanie termínov** – komplexná funkcionalita vyhľadávania termínov a zobrazenie jedného zvoleného termínu. Užívateľ disponuje celkovo tromi rôznymi spôsobmi vyhľadávania, základným vyhľadávacím poľom s funkciou automatického dopĺňania slov, vyhľadávaním podľa abecedy vo všetkých podporovaných jazykoch a možnosťou pokročilého filtrovania podľa textu definície, autora, dátumu zmeny a založenia, podľa druhu (aktuálny, neplatný, zastaralý...) a typu

(norma, monografia...) termínu. Nájdené termíny je možné zobrazovať priamo v aplikácii alebo exportovať do dokumentu vo formáte PDF, RTF, XML, SKOS alebo JSON.

2. **Vytváranie návrhov nových termínov** – možnosť vytvorenia návrhu na pridanie nového termínu do korpusu. Pri vytváraní návrhu systém automaticky vyhľadáva potencionálne duplicity nielen v momentálne otvorenej inštancií tezauru, ale aj vo všetkých ďalších napojených inštanciách.

3. **Editácia prekladov termínu** – možnosť meniť preklady termínu do jednotlivých jazykov. Každý používateľ môže mať obmedzené, do ktorých jazykov preklady môže upravovať.

4. **Editácia metainformácií termínu** – funkcionalita umožňujúca pridávanie a menenie dodatočných údajov o termíne, prípadne podľa roly prihláseného užívateľa vytváranie návrhov na ich zmenu.

5. **Schvaľovanie nových/zmenených termínov** – možnosť schvaľovať návrhy zmien korpusu vytvorené inými užívateľmi.

6. **Zobrazenie zoznamu zmazaných termínov** – zobrazenie zoznamu zmazaných termínov s možnosťou ich obnovy.

7. **Zobrazenie štatistík o prevádzke** – zobrazenie štatistických údajov o prevádzke tezauru. Napríklad počet návštev, najnavštevovanejšie heslá a podobne.

8. **Mazanie termínov** – možnosť zmazať vybraný termín.

9. **Menenie nastavení slovníku** – aplikácia umožňuje zmenu metadát slovníka (napríklad názov a skrátený názov slovníka, URL aplikácie, názov organizácie, komisie, kontaktnej osoby...), zmenu designu, pridávanie podporovaných jazykov, voľbu povinných jazykov, pridávanie možností domén, typov a druhov termínov, správu užívateľov a import/export nastavení.

10. **Reklamácia schváleného termínu k prepracovaniu** – funkcionalita pre jednoduché vrátenie termínu k prepracovaniu.

11. **Menenie technických nastavení slovníka** – k ostatným nastaveniam pridáva technické nastavenia – nastavenia zálohovania (cesta, URL, užívateľ, heslo, interval tvorby), adresy logov a konfiguráciu SMTP servera (adresa, port, užívateľ a heslo).

1.3 Riadenie vývoja systému

Systém webových tezaurov je založený na už existujúcom softvéri korpusu, ktorý bol vyvinutý Centrom spracovania prirodzeného jazyka Fakulty informatiky Masarykovej univerzity v Brne. Bolo potrebné do existujúceho riešenia implementovať drobné úpravy, vyvinúť celé webové rozhranie a komunikačnú vrstvu back-endu. Pre tento vývoj boli projektu pridelení dvaja programátori: jeden front-end (autor tejto diplomovej práce) a jeden back-end vývojár. Medzi požiadavkami bola aj tvorba príručiek, ako správne aplikácie používať. Ich vypracovanie som delegoval na kolegyňu. Príručky tvoria jednu z príloh tejto práce.

Vývoj systému webových tezaurov som rozdelil do dvoch fáz. Prvou fázou bol vývoj prototypu a druhou implementácia zmien a oprava chýb na základe spätnej väzby používateľov.

Pretože MV ČR, zadávateľ zákazky na vývoj tohto systému, na začiatku pevne definoval požiadavky na výsledný softvér a implementácii sa venovali len dvaja vývojári, tak som sa rozhodol pre prvú fázu – fázu vývoja prototypu – využiť vodopádový vývojový model. Po pevnej špecifikácii požiadaviek zákazníka sme definovali podobu REST API pre komunikáciu medzi back-endom a front-endom, podľa ktorých sme požadovanú funkcionalitu následne implementovali.

Po dokončení implementácie a odovzdaní hotového prototypu na testovanie prešiel projekt do druhej fázy. Túto som riadil využitím princípov agilnej metodiky Scrum. Z priebežne prichádzajúcich požiadaviek od užívateľov som vytváral tickety. Každý pracovný deň sme sa (ja – front-end vývojár) s back-end vývojárom stretli na standupe a dohodli, ktoré tickety budeme v daný deň implementovať, a vyjasnili všetky potrebné informácie. Pretože vývoj prebiehal nepravidelne museli byť niektoré Scrum ceremónie adekvátne upravené. Nemalo zmysel stanovovať si časový interval retrospektív a namiesto toho sme

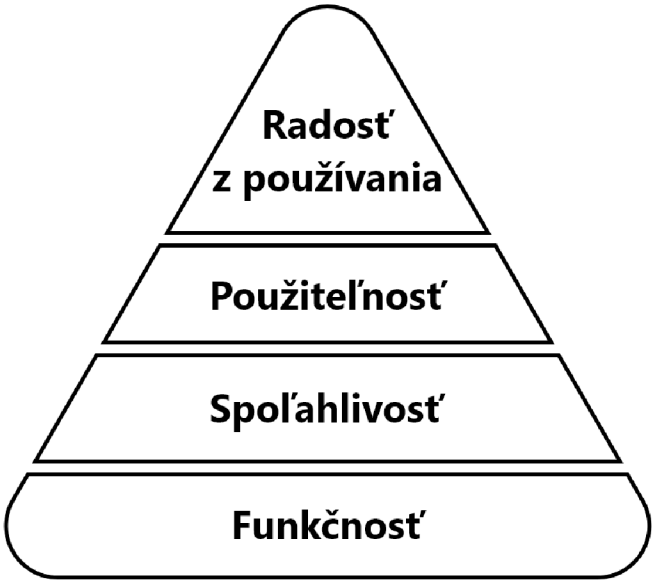
si určili, že si retrospektívu naplánujeme vždy po dokončení desiatich ticketov. Časom sme tento interval posunuli na dvadsať ticketov.

Po vyriešení všetkých návrhov získaných spätnou väzbou od užívateľov systému sme projekt nasadili na produkčné prostredia MV ČR a vývoj bol ukončený.

2 Návrh užívateľského rozhrania

Dôležitou súčasťou tvorby každej profesionálnej webovej aplikácie je vypracovanie návrhu užívateľského rozhrania. Mimo samotnej funkčnosti a robustnosti portfólia podporovaných funkcií má na výslednú užívateľskú prívetivosť významný dopad aj jej celkový design – od rozloženia tlačidiel a ovládacích prvkov, cez responzivitu až po použitú farebnú paletu.

Aarron Walter vo svojej knihe Designing for Emotion definoval hierarchiu potrieb užívateľov aplikácií, ktorá výrazne pripomína známu definíciu základných ľudských potrieb Abrahama Maslowa. Maslow vo svojej teórii argumentoval, že ľudia sú najvýraznejšie motivovaní naplnením svojich základných potrieb, napríklad snahou zaistiť si prístrešie alebo dostatok potravy, a až potom, čo sú tieto bezpečne zaistené, začínajú ľudia postupne napĺňať ďalšie potreby vyšších úrovní, napríklad potrebou spolupatričnosti, prijatia, sebarealizácie či kreativity. V obdobnom duchu Walter definoval pyramídu základných potrieb užívateľov aplikácií so štyrmi úrovňami. [2, 3]



Obr. 2.1: Walterova hierarchia užívateľských potrieb

Funkčnosť

Úplným základom každej aplikácie je funkcionalita, ktorú má za úlohu poskytovať svojím používateľom. Ak by aplikácia nedokázala poskytnúť užívateľovi riešenie problému či inej potreby, ktorú od nej očakáva, stráca celý systém zmysel svojej existencie.

Spoľahlivosť

Sekundárnou prioritou je spoľahlivosť. Okrem samozrejmej požiadavky na absenciu chýb a nepretržitú dostupnosť sa aplikácia musí chovať konzistentne a v intuitívne rovnakých situáciách by mala reagovať z pohľadu používateľa identicky.

Použiteľnosť

Výrazným faktorom v používateľovej spokojnosti je taktiež použiteľnosť či intuitívnosť aplikácie. Aplikácia s dobrým designom by nemala byť neprimerane náročná na pochopenie či používanie a mala by dodržiavať všeobecne akceptované konvencie rozloženia ovládacích prvkov. Znamením dobre zvládnutej použiteľnosti je, keď užívateľ nemusí rozmýšľať nad vykonávaním základných, napríklad navigačných, úkonov a už pri prvej návšteve ich zvláda intuitívne.

Radosť z používania

Ak sú všetky prechádzajúce faktory na dobrej úrovni, posledným faktorom, ktorý užívateľ vníma, sú rôzne drobné grafické vylepšenia či rozličné funkcie a nastavenia smerované na personalizáciu a spríjemnenie dlhodobého používania skúseným užívateľom.

V praxi sa často stáva, že vznikajú verejne dostupné aplikácie poskytujúce prvé tri priority na slušnej úrovni ale, zvyčajne z dôvodu obmedzeného rozpočtu, nevenujú pozornosť poslednému faktoru. Pretože poskytujú užitočné funkcie, sú spoľahlivé a aj relatívne pohodlné na použitie, stanú sa populárnymi. S rastúcou popularitou často prichádzajú aj finančné zdroje, ktoré umožnia majiteľom finan-

covať kompletné prepracovanie a nový design, v ktorom sa zamerajú hlavne na moderný vzhľad a profesionálny vizuálny dojem, čo má niekedy za následok zanedbanie základných princípov dobrého designu, ktorý bol doteraz vnímaný ako samozrejmosť. Ak sa tak stane, po zverejnení novej verzie aplikácia začne strácať užívateľov a môže prísť aj k jej úplnému zániku. Preto je dôležité vždy budovať design aplikácií od základov a nerobiť ústupky v prospech vizuálnych prvkov na úkor prvých troch princípov Walterovej hierarchie užívateľských potrieb.

2.1 UI vs UX

Návrh dizajnu webovej aplikácie je zvyčajne tvorený z dvoch základných častí, design užívateľského rozhrania (UI) a design užívateľského zážitku (UX). Design UI sa orientuje hlavne na farby, animácie a vizuálne aspekty všeobecne, kým design UX rieši aspekt interakcie so systémom a rozvrhnutia, hlavne ovládacích prvokov stránky. [4] Zvyčajne je najprv vytvorený návrh UX designu, často v podobe drôteného modelu, takzvaného wireframu, na ktorého základe je následne vytvorený finálny design užívateľského rozhrania pridaním farieb, aplikovaním rôznych rezov písma či obrázkov a animácií. UX design sa zaoberá hlavne splnením prvých troch princípov hierarchie užívateľských potrieb definovaných Aarronom Walterom (viac informácií sa nachádza v predchádzajúcej kapitole) a návrh UI sa venuje hlavne poslednej – radosti z používania. Z toho vyplýva, že správne spracovaný UX design je pre výslednú užívateľskú spokojnosť výrazne dôležitejší než design UI.

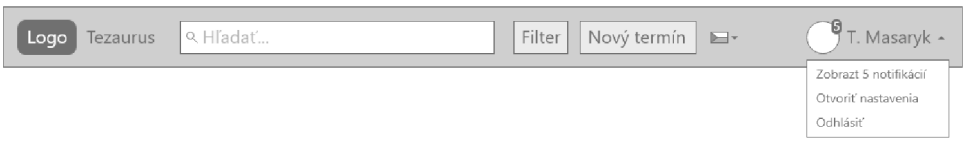
Táto diplomová práca sa venuje tvorbe systému, s ktorým budú interagovať zväčša vzdelaní zamestnanci, ktorí budú mať čas a priestor na prvotné naštudovanie systému. Z tohto dôvodu bol pri tvorbe designu kladený dôraz skôr na dlhodobé pohodlné používanie, aj keď zo začiatku menej intuitívne, než na jednoduché a rýchlo naučiteľné ovládanie. Súčasťou aplikácie sú dva manuály, na ktoré sa môže užívateľ v prípade potreby obrátiť a kde sa môže dozvedieť viac informácií o všetkých funkciách systému.

2.2 Drôtené modely

Táto kapitola obsahuje návrhy UX v podobe drôtených modelov. V praxi sa takýto návrh často označuje termínom mockup. Keďže súčasťou aplikácie je možnosť meniť farebnú paletu, a teda jednotlivé inštancie systému sa budú farebne líšiť, návrhy UI som nevytváral. Ďalším dôvodom je aj fakt, že ide o internú aplikáciu, pri ktorej upútanie užívateľov silným vizuálnym dojmom nie je prioritou.

2.2.1 Navigačná lišta

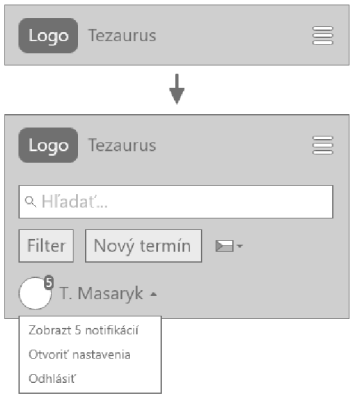
Základným ovládacím prvkom každej aplikácie je navigačná lišta. Dobre navrhnutá navigačná lišta obsahuje všetky základné a najčastejšie používané funkcie, ako aj informácie o momentálnom globálnom stave aplikácie, ako napríklad meno prihláseného používateľa či počet nových, ešte neobslúžených, notifikácií. Rovnako ako ďalšie časti, vytvoril som návrh len pre aplikáciu jednej inštancie slovníka, komponenty centrálneho systému majú rovnakú, prípade zjednodušenú podobu bez niektorých funkcií.



Obr. 2.2: Mockup navigačnej lišty

Prvou položkou je logo a názov inštancie systému. Ďalšou v poradí je vyhľadávacie textové pole. Po zadaní textu do poľa užívateľom o dĺžke minimálne troch znakov sa aktivuje funkcia automatického dopĺňania slov a zobrazia sa všetky zhody pre zadané znaky. Nasledujú odkazy v podobe tlačidiel na vyhľadávanie pomocou pokročilého filtrovania a na stránku vytvorenia nového termínu. Poslednou možnosťou ľavej strany panelu je voľba jazyka aplikácie. Na pravej časti navigačnej lišty je zobrazené meno momentálne prihláseného užívateľa, počet nových notifikácií a možnosti súvisiace s jeho účtom. V prípade, že nie je prihlásený žiaden užívateľ, sa na tomto mieste zobrazí text odkazujúci na prihlasovaciu obrazovku.

Aplikácia je primárne zameraná pre použitie na stolných počítačoch so širokými plnohodnotnými monitormi. Napriek tomu je nutné, aby bolo možné aplikáciu ovládať aj na menších zariadeniach. Pretože podoba navigačnej lišty vyobrazená vyššie by pre takéto zariadenia mohla byť príliš široká a nezmestila by sa na ich obrazovku, vytvoril som model aj jej responzívnej verzie.



Obr. 2.3: Mockup responzívnej verzie navigačnej lišty

2.2.2 Vstupná stránka

Prvou časťou aplikácie, ktorú väčšina užívateľov uvidí, je vstupná stránka, bežne označovaná anglickým výrazom landing page. Mala by obsahovať základné informácie o systéme a ukazovať užívateľom cestu, ako sa dostať k informáciám či nastaveniam, ktorých získanie bolo cieľom ich návštevy. [5]

V aplikácii jednej inštancie tezauru je navrchu vstupnej stránky logo a názov organizácie, ktorá ju spravuje, pod ktorými je uvedený názov tejto konkrétnej inštancie a krátky popis, na akú oblasť sa slovník zameriava. Pod tým nasledujú navigačné tlačidlá, slúžiace ako rozcestník do jednotlivých častí aplikácie. Vyobrazená verzia obsahuje všetky navigačné tlačidlá. Aplikácia automaticky zobrazuje len tie tlačidlá, ku ktorým má momentálne prihlásený užívateľ prístupové práva.

Návrh vstupnej stránky pre centrálny systém vyzerá obdobne, v podstate jediným rozdielom sú rozdielne tlačidlá odkazujúce na jed-

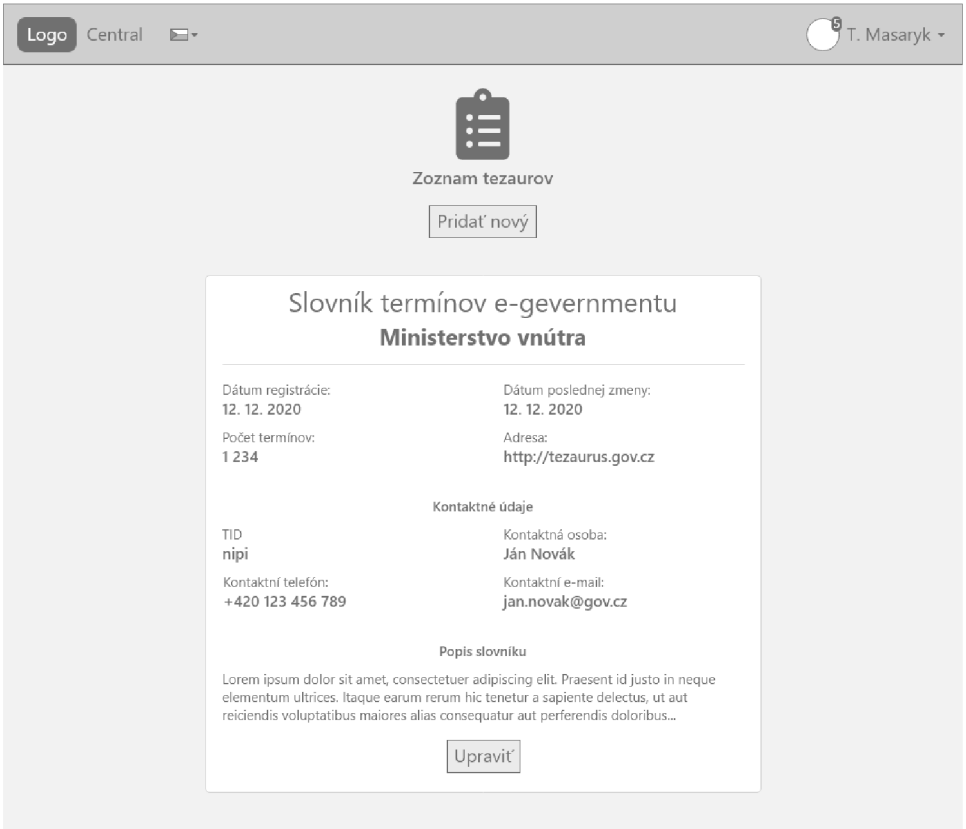
notlivé podstránky aplikácie. Na zariadeniach s nižším rozlíšením sa text jednoducho zúži a navigačné tlačidlá vytvoria jeden stĺpec, rozložením každého do nového riadku. Jedná sa relatívne o zanedbateľné zmeny, preto som nepovažoval za nutné vytvoriť unikátny drôtený model responzívnej verzie vstupnej stránky aplikácie, ani špecifickú verziu pre centrálny systém.

Obr. 2.4: Mockup vstupnej stránky inštancie tezauru

Tezaurus umožňuje abecedné prechádzanie záznamov, ktoré užívateľ môže využiť po presmerovaní cez prvé tlačidlo. Okrem krátkeho popisu si môže užívateľ zobraziť aj pokročilé informácie o slovníku stlačením nasledujúceho tlačidla. Tlačidlo s titulkom Nápoveda vedie k externému dokumentu manuálu, ktorý prislúcha k aplikácii a užívateľ tam môže nájsť informácie o jej jednotlivých funkciách a ako aplikáciu správne používať.

2.2.3 Zoznam tezaurov

Stránka zoznamu tezaurov zobrazuje zoznam všetkých inštancií slovníkov napojených na tento centrálny systém. Základným komponentom je blok symbolizujúci jednu inštanciu, ktorého súčasťou je názov inštancie, jej správca, metainformácie o zázname, kontaktné údaje na zodpovednú osobu a krátky popis, čím je daná inštancia špecifická. Okrem zobrazovania statických informácií, stránka umožňuje otvoriť formulár určený na vytvorenie nového či editáciu existujúceho záznamu o napojenom slovníku. Editačný formulár je relatívne jednoduchý, obsahuje len textové polia pre všetky dáta potrebné pre napojenie a preto som nepovažoval za potrebné vytvárať preň wireframe.



Obr. 2.5: Mockup zoznamu registrovaných slovníkov

Drôtený model zoznamu tezaurov je jediný grafický návrh vytvorený výhradne pre aplikáciu centrálneho systému. Aplikácia jednej inštancie tezauru je výrazne komplikovanejšia než aplikácia centrálneho systému. Z tohto dôvodu boli všetky ostatné drôtené modely spracované výhradne pre ňu, alebo v prípade veľmi podobných komponentov bola vytvorená jedna verzia s maximálnou komplexnosťou a následne pri implementácii zjednodušená pre potreby konkrétnej implementovanej aplikácie.

2.2.4 Stránka nastavení

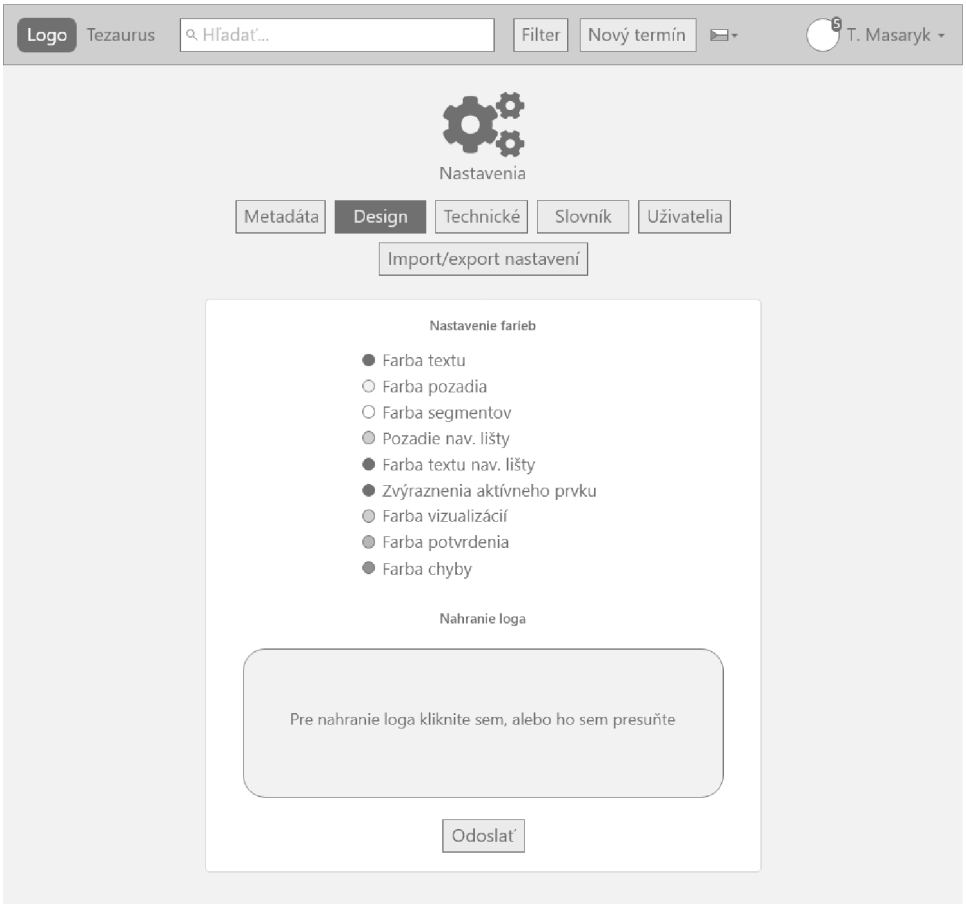
Autorizovaní užívatelia s pridelenými adekvátnymi užívateľskými rolami si môžu zobraziť stránku nastavení aplikácie a meniť jednotlivé nastavenia. Táto stránka zoskupuje všetky nastavenia naprieč celou aplikáciou, ktoré je možné meniť počas jej behu z jej užívateľského prostredia. Akékoľvek iné komplikovanejšie zmeny nastavení vyžadujú pripojenie na server, priamu editáciu konfiguračných súborov systému a následné reštartovanie systému.

Všetky nastavenia sú rozdelené do celkovo piatich sekcií, na základe ich logických podobností. Aplikácia umožňuje aj import a export všetkých nastavení prostredníctvom ďalšej, šiestej, sekcie. Medzi sekciami sa užívateľ môže navigovať pomocou navigačných tlačidiel. Tlačidlo momentálne aktívnej sekcie je farebne zvýraznené pre zlepšenie prehľadnosti.

Sekcia metadát obsahuje nastavenia týkajúce sa hlavne verejných informácií o inštancii tezauru. Medzi tie patrí napríklad jej názov, kontaktné informácie osoby, ktorá zodpovedá za jej chod, popis jej tematiky a špecifík v češtine aj angličtine. Druhá sekcia pomenovaná Design umožňuje správcom systému jednoducho meniť grafické prvky aplikácie, teda farebnú paletu a nahrať logo správcu inštancie tezauru. Sekcia technických nastavení obsahuje nastavenia SMTP servera a umožňuje vytvárať zálohy či meniť nastavenia ich intervalu a podobne. Pod sekciou pomenovanou Slovník sa nachádzajú nastavenia definujúce chovanie rozličných funkcií tezauru, ako napríklad zoznam domén či typov, pod ktoré je následne možné termíny zaraďovať. Taktiež umožňuje konfiguráciu zoznamu podporovaných jazykov a výber povinných jazykov, do ktorých musí byť každý termín preložený. Posledná sekcia nastavení obsahuje zoznam všetkých užívateľov, taktiež

aj funkcionalitu na editáciu uložených informácií o nich či možnosť registrovania nových osôb do systému. Šiesta sekcia neobsahuje žiadne ďalšie nastavenia, umožňuje len jednoduchý export a import všetkých nastavení do jedného súboru.

Aplikácia automaticky autentizuje momentálne prihláseného užívateľa a zobrazí mu len tlačidlá spadajúce pod jeho autorizáciu. Niektorým užívateľom taktiež zobrazuje podstránky len v režime pre čítanie a neumožňuje im uskutočňovať žiadne zmeny, ak nemajú dostatočné oprávnenia.



Obr. 2.6: Mockup stránky zmeny grafických nastavení

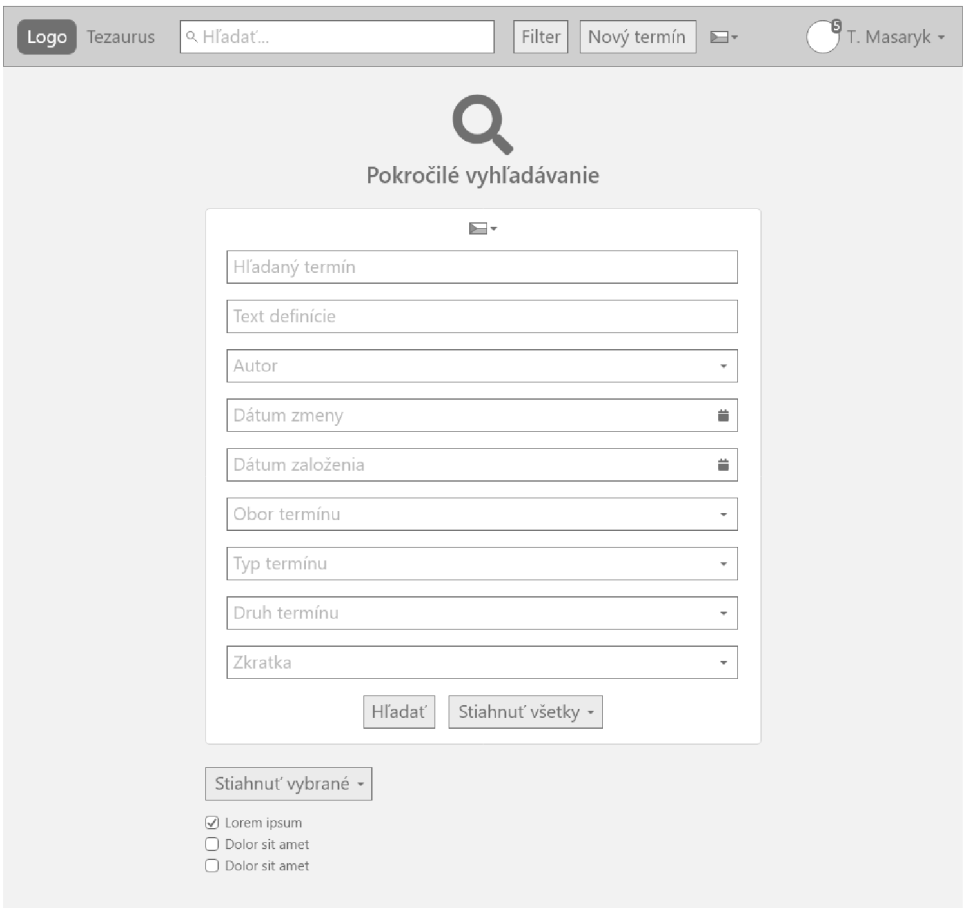
Pre umožnenie prispôsobenia vizuálnej identity som rozdelil celú aplikáciu do niekoľkých opakovane používaných farieb, eventuálne len s drobnými variáciami jasu, kontrastu či odtieňa farby. Stránka nastavení designu tak umožňuje jednoducho meniť farebnú paletu. Po kliknutí na jeden element sa otvorí okno pre výber farby. Po zvolení požadovaného odtieňa sa zmena hneď aplikuje na momentálnu obrazovku, vďaka čomu si správcovia systému nemusia len predstavovať, ale môžu hneď vidieť, ako bude aplikácia po aplikovaní novej farebnej platy vyzerať.

Všetky ostatné sekcie nastavení sa skladajú len z vizuálne jednoduchých textových formulárov, pre ktoré som nepovažoval za nutné vytvárať vlastné mockupy.

2.2.5 Pokročilé vyhľadávanie

Okrem jednoduchého základného vyhľadávania len podľa samotného znenia termínu je vhodné, aby užívatelia mali možnosť aj pokročilého vyhľadávania a filtrovania podľa rozličných kritérií, ako napríklad podľa odboru termínu, dátumu vytvorenia či podľa autorstva. Zároveň je potrebné, aby stránka umožňovala vyhľadávanie aj v iných jazykoch, než je momentálne vybraný jazyk aplikácie užívateľom. Táto funkcionalita je nutná, nakoľko aplikácia nie je preložená do každého jazyka, v ktorom môžu byť v slovníku záznamy termínov, a základné vyhľadávacie pole v navigačnej lište vyhľadáva termíny len v jazyku, v ktorom je momentálne aplikácia nastavená.

Okrem typického použitia pre vyhľadávanie termínu, ktorého znenie nie je užívateľovi známe, podľa definície či ostatných parametrov je stránka vhodná aj v prípade potreby hromadného exportu viacerých termínov. Vzhľadom k potencionálnej veľkosti dát môže vyhľadávanie trvať relatívne dlho, preto je vhodné okrem zobrazenia vyhľadávaných termínov implementovať funkcionalitu aj pre ich priamy export. Tým sa užívateľovi ušetrí čas a spríjemní obsluha aplikácie.



Obr. 2.7: Mockup stránky pokročilého vyhľadávania termínov

Z pohľadu UX designu považujem za samozrejmosť, aby užívateľ vkladal dáta pomocou výberových polí, anglicky známych ako select boxes, ktoré mu zobrazia všetky možnosti, ktoré pre dané pole dávajú zmysel, napríklad všetkých autorov ktorí do aplikácie vkladali nové termíny. Rovnako polia dátumov po kliknutí zobrazujú drobný komponent kalendára, v ktorom si užívateľ vyberie dátum, ktorý požaduje, a ten sa automaticky naformátuje do správneho tvaru. Vďaka tomu užívateľovi uľahčíme interakciu s aplikáciou a predídeme chybám spôsobeným napríklad preklepmi v zadávaní odboru termínu či nesprávneho formátu dátumu.

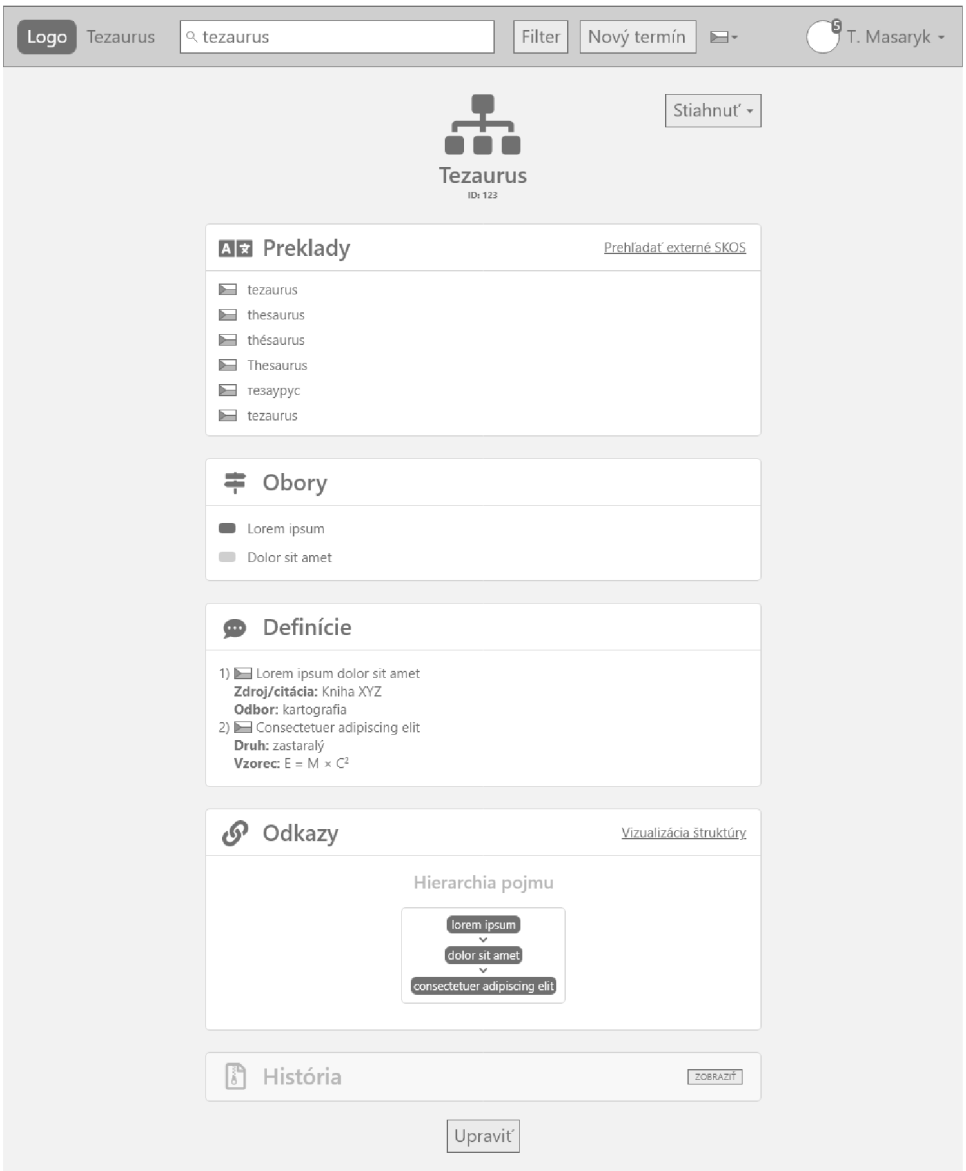
2.2.6 Detail jedného termínu

Stránka detailu jedného termínu je v podstate pointou a teda najdôležitejšou časťou celého systému. Na tejto stránke sa zobrazujú nahraté dáta o samotných termínoch uložených v tezaure. Zároveň stránka poskytuje aj funkcionalitu pre stiahnutie momentálne zobrazeného termínu v rôznych dátových formátoch, obsahuje odkazy na externé zdroje a autorizovaným užívateľom poskytne možnosť prejsť na stránku editácie termínu.

Navrchu stránky je zobrazený samotný termín, jeho interné identifikačné číslo v systéme a tlačidlo pre výber formátu na stiahnutie. Zvyšná časť stránky je rozdelená do blokov. Každý blok zobrazuje logicky súvisiace informácie o termíne. Bloky sú vždy v rovnakom poradí a označené rovnakou ikonkou, čo užívateľovi uľahčuje navigáciu a orientáciu pri hľadaní konkrétnej informácie. Niektoré dlhé bloky obsahujúce menej často vyhľadávané informácie (napríklad blok histórie editácie alebo príkladov použitia termínu) sa zobrazujú v základnom pohľade zabalené do jedného riadku len s nadpisom a tlačidlom pre rozbalenie bloku. Vďaka tomu je stránka kratšia a jednoduchšie sa na nej vyhľadávajú požadované informácie. Taktiež sa dáta začnú načítavať až v momente, keď užívateľ klikne na tlačidlo rozbalenia bloku, vďaka čomu je prvotná odpoveď zo servera menšia, a teda rýchlejšia, a šetrí sa množstvo prenášaných dát, nakoľko sa môže jednať o pomerne objemné dáta a pri väčšine návštev stránky termínu ich užívatelia nepotrebujú.

Prvým blokom je blok prekladov, kde sa zobrazujú všetky uložené preklady a varianty termínu. Jeden termín môže mať zaznamenaných aj viacero prekladov pre rovnaký jazyk. Blok odborov obsahuje zoznam odborov, pod ktoré termín spadá. Každý odbor má definovanú vlastnú farbu a eventuálne aj špeciálny symbol pre lepšiu identifikáciu. Následujúcim blokom je blok definícií. Ide o najrozsiahlejší blok. Obsahuje definície o termíne v rôznych jazykoch, ktoré môžu pozostávať z informácií o ich zdroji, druhu, definovať vzorec alebo obsahovať tabuľku. V bloku s titulkom Odkazy sa nachádzajú informácie o vzťahoch a hierarchii pojmu v relácii k ostatným termínom uloženým v slovníku. Každý termín je zobrazený ako odkaz a po kliknutí naň je užívateľ presmerovaný na stránku detailu daného termínu. Poslednými, pri načítaní stránky zabalenými, blokmi sú blok histórie a blok

príkladov použitia termínu. Blok histórie obsahuje záznamy o editácii termínu a odkazy na jednotlivé verzie termínu v minulosti. Blok príkladov použitia obsahuje vety, ktoré obsahujú tento termín. Taktiež ho v každej vete zvýrazňuje zmenou rezu písma.



Obr. 2.8: Mockup stránky detailu jedného termínu

Stránka editácie termínu má obdobný vzhľad ako stránka detailu jedného termínu. Namiesto zobrazovania jednotlivých informácií v podobe jednoduchého textu zobrazuje textové polia s východiskovou hodnotou reprezentujúcou momentálne uložené dáta. Taktiež neobsahuje blok odborov, pretože tie nie je potrebné definovať zvlášť, ich zoznam sa načíta z odborov definícií. Stránka taktiež kontroluje splnenie všetkých podmienok a upozorní užívateľa pri pokuse o uloženie termínu v nekorektnej podobe.

3 Použité technológie

Jedným z cieľov mojej diplomovej práce bol aj výber vhodných technológií k implementácii systému. Pretože ide o aplikáciu, ku ktorej užívatelia budú pristupovať cez webové rozhranie, použitie HTML, CSS a skriptovacieho jazyka JavaScript bolo samozrejmosťou. Pre zjednodušenie tvorby užívateľského rozhrania som sa rozhodol využiť Javascriptovú knižnicu React. Bližšie zdôvodnenie jej použitia a porovnanie s konkurenčnými technológiami, ako napríklad s knižnicou AnglularJS či Vue.js, sa nachádza v nasledujúcej kapitole.

Back-end systému pozostáva z rozšírenej verzie už existujúceho riešenia patriaceho Centru spracovania prirodzeného jazyka (CZPJ) Fakulty informatiky Masarykovej univerzity, pod ktorého záštitou bol celý projekt realizovaný. Už existujúce back-endové riešenie je postavené pomocou programovacieho jazyka Ruby. Ide o plne objektovo orientovaný interpretovaný skriptovací vysoko úrovňový programovací jazyk. [6] Nakoľko toto už existujúce riešenie zodpovedalo veľkej časti požiadaviek, rozhodol som sa, že bude vhodné len doplniť chýbajúce funkcie. Toto rozhodnutie taktiež viedlo k tomu, že Ruby bolo použité ako back-endová technológia. Keďže nemám veľa skúseností s programovaním back-endu a konkrétne s jazykom Ruby žiadne, a zadanie mojej diplomovej práce to nevyžadovalo, rozhodol som sa, že bude vhodné celú prácu na back-ende delegovať na zamestnancov CZPJ.

Komunikácia medzi front-endom a back-endom aplikácie je postavená na architektúre REST, ktorá je v dnešnej dobe v prostredí webových technológií veľmi rozšírená. Pre zjednodušenie a zároveň aj automatické zálohovanie vývoja systému bol použitý verzovací systém Git, konkrétne implementácia GitLab hostovaná na serveroch Masarykovej univerzity. GIT projekt je zaradený pod skupinou NLP Centra. Využitie GitLab-u mi taktiež umožnilo používať jeho robustné nástroje automatizovaného reťazeného spracovania, takzvaných pipeline. Tie nám umožnili dosiahnutie pohodlného, relatívne rýchleho a zväčša bezproblémového nasadzovania aplikácie na servery produkčného či testovacieho prostredia.

3.1 React

React je Javascriptová knižnica pôvodne vytvorená a spravovaná spoločnosťou Facebook Inc. Ide o jeden z najpopulárnejších nástrojov určených na tvorbu dynamických užívateľských rozhraní webových stránok či aplikácií. [7] Okrem Reactu som zvažoval aj použitie knižníc Vue.js a Angular.

React vs Vue.js

Knižnica Vue.js je v mnohých ohľadoch veľmi podobná knižnici React. Obe sú určené pre tvorbu moderných dynamických webových aplikácií so zameraním na užívateľské rozhranie. Obe sú založené na funkcionalite virtuálizácií DOMu a aplikácie vytvorené pomocou nich zvyčajne nasledujú architektúru znovu použiteľných komponentov. Kým React vznikal ako interný korporátny nástroj a až časom bol uvoľnený ako open source projekt, Vue.js bolo vytvorené jedným človekom a dnes je udržiavané a ďalej vyvíjané len svojou komunitou ako open source technológia. [8]

Vue.js je všeobecne považované ako jednoduchšia technológia na naučenie aj pre začiatočníkov a to aj vďaka svojej kvalitnejšej dokumentácii. Na druhej strane sa React teší korporátnej podpore svojej materskej spoločnosti Facebook Inc. a okrem nej aj silnej komunite vývojárov, čo, dá sa povedať, zaručuje pokračovanie jeho vývoja a údržby. React je hlavne v Európe celkovo výrazne populárnejší a rozšírenejší, vďaka čomu je ďaleko jednoduchšie nájsť zamestnancov, ktorí už s ním majú skúsenosti. Taktiež poskytuje bohatší ekosystém naviazaných knižníc, integrácií s IDE a iných zdrojov či technológií. Všeobecne sa React považuje za vhodnejšiu voľbu pre tvorbu rozsiahlych významných systémov či aplikácií. [9]

Systém tezaurov je štátnym systémom, ktorý sa bude pravdepodobne používať a udržiavať ešte dlhú dobu. Preto stabilita Reactu, vďaka jeho korporátnemu pôvodu, je pre tento projekt z môjho pohľadu výraznou výhodou. Taktiež by v budúcnosti nemal nastať problém s hľadaním programátorov, ktorí s ním majú bohaté skúsenosti. Pred pár rokmi bolo Vue.js vyzdvihované pre lepšiu výkonnosť a optimalizáciu, ale dnes sú s Reactom už na podobnej úrovni. Z týchto

dôvodov považujem knižnicu React za vhodnejšiu voľbu pre tvorbu tejto aplikácie než Vue.js.

React vs Angular

Najvýznamnejším rozdielom medzi technológiami React a Angular je fakt, že React je len knižnicou, zatiaľčo Angular je komplexným frameworkom. Z pohľadu MVC modelu React vo svojom základe poskytuje len časť V – View –, teda stará sa čisto len o uľahčovanie práce so zobrazovaním dát užívateľovi a dáva architektovi aplikácie slobodu si pre ostatné časti voliť ľubovolné nástroje či nevyužiť žiadne knižnice a použiť natívne riešenia Javascriptu. Na druhej strane Angular obsahuje okrem zobrazovacej i pevne definovanú komplexnú funkcionalitu aj pre ostatné časti MVC modelu. Vďaka tomuto programátor nemusí premýšľať nad tým, ktoré knižnice použije, ani riešiť potencionálne problémy s ich nekompatibilitou a môže jednoducho využiť overené all-in-one riešenie.

Hlavným prínosom, ktorý React ponúka, je jeho technológia virtuálneho DOMu, vďaka ktorému je dynamické aktualizovanie rôznych častí aplikácie rýchle a jednoduché. Angular na druhej strane využíva priamo len reálny DOM prehliadača, čím, obzvlášť u veľmi rozsiahlych aplikácií, šetrí výrazné množstvo vyžadovanej operačnej pamäte. React využíva takzvaný JSX syntax, ktorá spája zápis podobný HTML s Javascriptovou logikou, na rozdiel od Angularu, ktorý programovú logiku a zápis štruktúry HTML striktne oddeľuje a do HTML len pridáva direktíva pre obojstranné naviazanie dát.

Rovnako ako React, Angular je dielom veľkej nadnárodnej spoločnosti. Zatiaľčo React bol vytvorený spoločnosťou Facebook Inc., za Angularom stojí Google. React je staršou technológiou – jeho prvá verzia bola vydaná v marci 2013. Na rozdiel od toho prvá verzia Angularu, ako ho poznáme dnes, bola zverejnená v septembri 2016. Angular už ale existoval aj predtým pod názvom AngularJS, ktorý bol vydaný už v októbri 2010. Okrem premenovania sa jednalo o celkový prepis a dokonca aj zmenu programovacieho jazyka z Javascriptu na TypeScript a kvôli tomu neexistuje možnosť jednoduchej spätnej kompatibility. Google oznámil, že podpora originálneho frameworku AngularJS skončí ku koncu roku 2021 a naďalej bude vyvíjať len modernú verziu Angularu založenú na Typescripte. [10, 11]

React je dnes celkovo populárnejšou technológiou, je modulárnejší a nevyžaduje od programátora dodržiavanie presne definovaného štýlu zápisu kódu tak, ako Angular. Taktiež poskytuje lepšiu spätnú kompatibilitu, ktorej absencia by u štátneho projektu s potencionálne dlhou životnosťou v prípade Angularu mohla v budúcnosti spôsobovať problémy. [11] Angular pre jeho celkovú architektúru považujem za vhodnejší skôr pre komplexnejšie projekty, než akým je systém tezaurov. Z týchto dôvodov som pre vývoj užívateľského rozhrania systému zvolil knižnicu React.

3.1.1 Knižnica Redux

Veľmi populárnou knižnicou, hlavne v spojení s Reactom, je open source knižnica Redux.[1] Jedná sa o nástroj určený pre komplexnú správu stavu aplikácie vrátane funkcionality pre vysielanie notifikácií o zmene momentálneho stavu jednotlivým komponentom aplikácie. Jeho primárnym cieľom je unifikovanie stavu a poskytnutie takzvaného jedného zdroja pravdy, vďaka čomu sa predchádza problémom nekonzistencie dát.

Redux je komplexné riešenie s robustnou paletou podporovaných funkcií, ktoré ale nie sú v každom projekte potrebné. Hlavne v jednoduchších projektoch si programátor dokáže postrážiť zabezpečenie konzistentnosti dát aj pomocou tzv. Context API, ktoré už v základe poskytuje samotný React a ktoré aj Redux na pozadí využíva. [12]

Aplikácie systému tezaurov nevyžadujú komplexnú manipuláciu stavu a obsahujú len niekoľko jasne oddelených dátových objektov. Aplikácia jednej inštancie tezauru za behu pracuje so svojimi globálnymi nastaveniami, dátami o vyhľadávanom či momentálne zobrazenom termíne a informácie o autorizovanom užívateľovi v prípade, že je nejaký užívateľ v daný okamih prihlásený. Ako príklad uvádzam aplikáciu jednej inštancie tezauru, nakoľko je jej dátový model komplexnejší, než dátový model aplikácie centrálneho systému. Z týchto dôvodov som sa rozhodol, že nie je potrebné využiť knižnicu Redux a bude vhodnejšie nezvyšovať celkovú veľkosť nasadenej aplikácie a teda veľkosť súborov, ktoré potrebujú užívatelia stiahnuť pri návšteve stránky.

1. Hlavná stránka Redux: https://redux.js.org/

3.1.2 Knižnica Informed

Z pohľadu implementácie najkomplexnejšou časťou systému je stránka editácie jedného termínu aplikácie jednej inštancie tezauru. Obsahuje veľa rôznych polí, každé s rozdielnymi validačnými pravidlami a rozličnou definíciou práv pre zobrazenie či zápis. Väčšinu polí je taktiež potrebné dynamicky pridávať podľa toho, koľko prekladov či definícií si užívateľ praje k termínu pridať.

Pre uľahčenie práce s formulármi som sa rozhodol využiť k tomuto účelu určenú predpripravenú knižnicu. Z rozličných možností som sa rozhodol použiť knižnicu Informed, ktorá podporuje všetku potrebnú funkcionalitu.[2]

3.1.3 Ďalšie nástroje z ekosystému Reactu

Pri tvorbe systému tezaurov som okrem technológií spomínaných v ostatných kapitolách využil ešte niekoľko ďalších, bežne používaných knižníc z ekosystému Reactu, ktoré uvádzam ďalej.

React Router je kolekcia navigačných komponentov pre kompletnú správu navigácie a navigačnej histórie užívateľa cez jednotlivé podstránky aplikácie.

Babel je Javascriptovým kompilerom, vďaka ktorému je možné používať všetky moderné funkcie Javascriptu ešte predtým, než je bezpečné predpokladať, že dostatočné množstvo užívateľov používa aktuálny internetový prehliadač, ktorý ich podporuje.

webpack je takzvaný static module bundler, teda nástroj, ktorý spracováva hotovú aplikáciu tak, že vygeneruje graf dependencií, pomocou ktorého následne zabalí všetky potrebné súbory do jedného, prípadne viacerých modulov.

Lodash je malá knižnica moderných drobných nástrojov či rôznych pomocných funkcií, ktorých účelom je spríjemňovať život programátorovi. Z mnohých poskytovaných funkcií som pri tvorbe systému tezaurov využil napríklad funkciu debounce, ktorá poskytuje jedno-

2. Hlavná stránka Informed: https://joepuzzo.github.io/informed/

duchú definíciu pre oneskorené volanie požadovaného kódu, ktorej časomiera sa resetuje pri každom jej zavolaní. Túto funkcionalitu je, samozrejme, možné dosiahnuť aj pomocou zopár natívnych funkcií Javascriptu, ale Lodash ich elegantne poskytuje už zaobalené v jednom volaní.

react-dropzone je nástroj pre vytvorenie takzvanej drag'n'drop zóny pre nahratie súboru. Táto funkcionalita spríjemňuje užívateľský zážitok používateľa aplikácie pri nahrávaní súboru, keďže nemusí prechádzať cez klasický formulár nahratia súboru a požadovaný súbor môže jednoducho preniesť myšou do okna prehliadača. V prípade, že by niekomu klasický spôsob vyhovoval viac, môže kliknutím na zónu vyvolať formulárové okno a súbor nahrať jeho pomocou, takže zostáva zachovaná i táto funkcionalita.

react-translate, ako názov napovedá, umožňuje jednoduchú internacionalizáciu aplikácie.

React Color je kolekcia rôznorodých komponentov pre definovanie farieb s mnohými užitočnými podpornými funkciami. Táto knižnica významne uľahčuje tvorbu stránky prispôsobenia dizajnu s instantnou odozvou systému.

react-lineto je priamočiarym nástrojom pre generovanie úsečiek medzi dvomi bodmi či jednotlivými komponentmi aplikácie. Tento nástroj som využil pri tvorbe vizualizácií hierarchie medzi jednotlivými termínmi slovníku.

react-simple-keyboard je knižnica obsahujúca komponent virtuálnej klávesnice spolu s rôznymi užitočnými pomocnými funkciami a nástrojmi. Podporuje napríklad vlastné rozloženia kláves a obsahuje aj natívne integrovanú funkcionalitu pre prepínanie medzi nimi.

query-string je jednoduchá knižnica určená k parsovaniu a generovaniu štandardizovaných parametrov pre URL adresy.

url-regex je knižnice obsahujúca funkcionalitu pre vyhľadávanie URL adries v texte za využitia regulárneho výrazu.

whatwg-fetch je polyfillom, ktorý je nutný pre použitie natívnej funkcie fetch moderného Javascriptu v starších prehliadačoch, ako napríklad Internet Explorer.

Všetky knižnice sú dostupné cez správcu Javascriptových balíčkov npm (Node.js Package Manager): https://npmjs.com/.

3.2 TypeScript

Jednou z dnes často používaných technológií spolu s Reactom je TypeScript.[3] TypeScript je open source programovací jazyk vytvorený a spravovaný spoločnosťou Microsoft. Jedná sa o striktnú nadmnožinu Javascriptu, presnejšie povedané jazyka ECMAScript 2015, ktorý je nadmnožinou jazyka ECMAScript 5, ktorý je často nazývaný originálnym Javascriptom.

TypeScript pridáva do Javascriptu anotácie pre statickú typovú kontrolu v dobe kompilácie. Pri pedantnom typovaní premenných a funkcií zaručuje typovú korektnosť programu v dobe kompilácie. Pretože sa anotácia typov z kódu počas kompilácie odstraňuje, TypeScript nedokáže vykonávať typovú kontrolu za behu aplikácie. Z toho vyplýva, že v prípade nesprávneho nastavenia typov, napríklad dát prichádzajúcich zo serveru, TypeScript skontroluje len korektné zaobchádzanie s predpokladanými dátami, nedokáže ale zaručiť, že dáta prijaté zo servera v dobe behu programu skutočne dodržujú vopred definovanú deklaráciu typov, kvôli čomu nedokáže zachytiť jednu z najčastejších chýb v súvislosti s typmi. Ak sa má typovanie vykonávať korektne, vyžaduje od programátora pedantnosť a vynaloženie nezanedbateľného množstva času. Rovnako typová korektnosť programu nezaručuje celkové korektné fungovanie systému. Taktiež TypeScript neodhaľuje chyby v samotnej špecifikácii programu, ktoré tvoria okolo 78 % všetkých nahlásených chýb v open source programoch na GitHub-e. [13]

Vychádzajúc z týchto informácií som usúdil, že v mojom prípade by TypeScript vyžadoval viac práce a pridávaním typov by som strávil

3. Hlavná stránka TypeScript: https://www.typescriptlang.org/

viac času, než koľko by mi ich automatickou kontrolou vo výsledku ušetril. K podobnému záveru dospeli v prípade podobných, teda relatívne menších projektov aj programátori s výrazne väčším množstvom skúseností, než mám ja. [14]

3.3 Komponentový framework

Pri tvorbe webových aplikácií môže výrazne uľahčiť prácu použitie komponentového frameworku. Komponentové frameworky zväčša obsahujú predpripravené, otestované a responzívne komponenty, ktoré sú pripravené na okamžité použitie. Týmto sa môže ušetriť významné množstvo času, ktoré by inač programátori museli stráviť tvorbou všetkých potrebných komponentov. Na druhej strane, nič nie je zadarmo. Hlavným kompromisom je obmedzenie designu užívateľského rozhrania. V prípade, že sa vyžaduje dodržanie presne vopred definovaného grafického jazyka, môže byť, a v praxi aj často býva, snaha o prispôsobenie už hotového komponentového frameworku, aby nasledoval potrebnú vizuálnu identitu, kontraproduktívna. Výnimkou môžu byť niektoré frameworky, ktoré podporujú pokročilé funkcie prispôsobenia, ktoré vďaka tejto funkcionalite hlavne v poslednej dobe získavajú výrazný náskok v popularite v porovnaní s ich striktnejšou konkurenciou.

Aplikácie systému tezaurov nevyžadujú žiadnu špeciálnu grafickú podobu, a preto rozhodnutie použiť k implementácii jeden z komponentových frameworkov bolo jednoduché. Vzhľadom na to, že aplikácie budú vyvíjané za pomoci knižnice React, vyberal som medzi komponentovými frameworkmi, ktoré sú v nej taktiež napísané. Celkovo som zvažoval štyri rôzne frameworky, vyberal som z najpopulárnejších, ktoré boli v danom čase (jún 2018) verejne dostupné pre komerčné použitie. [15]

Ant Design

V dobe písania tejto diplomovej práce (december 2020) je framework Ant Design[4] najpopulárnejším komponentovým frameworkom Reactu.

4. Domovská stránka Ant Design: https://ant.design/

Popularitu frameworkov som porovnával podľa počtu hviezdičiek na platforme GitHub.

Ant Design bol vytvorený a je udržiavaný čínskou spoločnosťou Alibaba Group zo začiatku ako ich interný nástroj, ktorý bol neskôr uvoľnený pod MIT licenciou aj pre verejnosť ako open source projekt. Je určený na použitie hlavne v korporátnom prostredí, ktorému má poskytovať potrebnú kvalitu. Sústreďuje sa hlavne na poskytnutie dobrého užívateľského zážitku. Veľký dôraz kladie taktiež na elegantné a prirodzeným dojmom pôsobiace užívateľské prostredie. Z českých spoločností tento framework používa napríklad spoločnosť Home Credit pre svoje interné nástroje.

Dôvodom, prečo som pre tvorbu front-endu systému tezaurov framework Ant Design nezvolil, bolo jeho napojenie na čínske spoločnosti. Diplomová práca je súčasťou projektu pre Ministerstvo vnútra Českej republiky (MV ČR), aj keď mi to nebolo výslovne zakázané, neprišlo mi v dobe, keď sa vyhlasovali rôzne sankcie a zákazy spolupráce s čínskymi spoločnosťami, za vhodné framework tohoto pôvodu používať. Taktiež je Ant Desgin dnes na tom celkovo lepšie, než bol v dobe výberu. Vtedy bol výrazne menej populárny a potýkal sa s väčším množstvom problémov, než ostatné frameworky.

Material UI

Material UI[5] bol v dobe výberu najpopulárnejším komponentovým frameworkom z ekosystému Reactu a aj dnes za Ant Designom zaostáva len veľmi tesne. Jedná sa taktiež o open source projekt šírený pod MIT licenciou.

Material UI je špecifický v tom, že, ako už názov napovedá, riadi sa a implementuje filozofiu Material Designu definovanú spoločnosťou Google. Z tohto dôvodu sa nápadne podobá produktom od tejto spoločnosti, čo bol tiež hlavný dôvod, prečo som ho z výberu eliminoval. Systém tezaurov nevyžaduje dodržiavanie žiadnej konkrétnej grafickej identity, ale nebolo by vhodné, keby aplikácie pôsobili, akoby boli vytvorené spoločnosťou Google.

5. Domovská stránka Material UI: https://material-ui.com

React Bootstrap

Jedná sa o kompletnú prestavbu najpopulárnejšieho všeobecného komponentového frameworku Bootstrap pre React.[6] Za svoju popularitu však pravdepodobne vďačí hlavne svojmu menu a dedičstvu. V porovnaní s ostatnými v čase môjho výberu frameworku obsahoval závažnejšie chyby a neobsahoval žiadnu podporu pre prispôsobenie designu užívateľského rozhrania. Napriek tomu sa dostal do finále výberu spolu s frameworkom Sematic UI.

Semantic UI

Semantic UI[7] je univerzálny komponentový framework, ktorý sa sústreďuje na pocit intuitívneho užívateľského zážitku namiesto efektného dizajnu. Podľa slov jeho tvorcov je jeho hlavným účelom tvorba responzívnych a pre ľudí priateľsky pôsobiacich aplikácií. Jeho tvorcovia taktiež vytvorili oficiálne integrácie s Reactom, Angularom, Emberom a ďalšími knižnicami určenými na tvorbu užívateľských rozhraní.

V porovnaní s ostatnými frameworkmi považujem jeho dokumentáciu za najdetailnejšie spracovanú. Obsahuje taktiež základnú funkcionalitu pre vytváranie rôznych grafických tém, ktorá sa môže pri programovaní požadovanej podpory zmien v grafickej palete aplikácie hodiť. Je síce menej populárny, než React Bootstrap, ale podľa môjho názoru, ten za svoju popularitu z veľkej časti vďačí svojmu pôvodu. Z týchto dôvodov som sa rozhodol pre implementáciu využiť práve tento komponentový framework.

3.3.1 Ikony

Vhodnou grafickou nápovedou pre užívateľa sú ikony, charakterizujúce jednotlivé časti či funkcie aplikácie. K tomuto účelu som použil sadu fontov Font Awesome (šírenú pod CC BY 4.0 licenciou[8]), ktorá je taktiež súčasťou frameworku Semantic UI.

6. Domovská stránka React Bootstrap: https://react-bootstrap.github.io
7. Domovská stránka Semanti UI: https://semantic-ui.com/
8. Licencia Font Awesome: https://fontawesome.com/license/free

4 Implementácia

Po zvolení vhodných technológií k implementácii bola nasledujúcim krokom samotná tvorba prototypu aplikácií. Prácu som rozdelil na front-end a back-end. Implementácii front-endu som sa venoval ja a celý vývoj back-endu som delegoval na kolegov z CZPJ.

Vývoj front-endu som začal pomocou Create React App (CRA). CRA je veľmi užitočný nástroj, ktorý umožňuje kompletnú inicializáciu prostredia pre vývoj projektu používajúceho knižnicu React. Obsahuje v sebe väčšinu knižníc a nástrojov ako závislostí, ktoré pri vývoji typických aplikácií programátori obyčajne potrebujú, ako napríklad webpack, Babel a ESLint.

4.1 Architektúra projektu

Prvotným intuitívnym riešením bolo vytvorenie dvoch oddelených front-endových projektov: jedného pre aplikácie centrálneho systému a druhého pre aplikácie jednej inštancie tezauru. Pri ďalšej analýze som si uvedomil, že oba projekty by obsahovali veľa zdieľaného kódu či súborov, ako napríklad systém prihlasovania, logiku základného rozloženia stránky s navigačnou lištou či rôzne logá alebo obrázky a ikonky. Z týchto dôvodov som sa rozhodol pre vytvorenie jedného spoločného projektu len s niektorými oddelenými súbormi pre centrálny systém a aplikáciu slovníka.

4.1.1 Enviroment variables

Pri spúšťaní vývojárskeho servera či inicializácii sekvencie zostavy produkčnej verzie je pre výber požadovaného variantu aplikácie možné použiť jednoduchý prepínač `REACT_APP_CENTRAL` nastaviteľný ako premennú prostredia (Environment Variable). Pre voľbu verzie aplikácie pre centrálny systém je potrebné nastaviť túto premennú na hodnotu 1, pre aplikáciu jednej inštancie na 0.

Pomocou druhej podporovanej premennej prostredia je možné definovať URL adresu, na ktorej sa nachádza rozhranie pre komunikáciu s back-endom. Hodnota tejto premennej prostredia je nastaviteľná pomocou kľúča `REACT_APP_API_URL`.

Posledná premenná umožňuje definovanie koreňového reťazca URL adresy, t. j. napríklad reťazec `/tezaurus`. Kľúč `PUBLIC_URL` premennej prostredia nastavuje túto hodnotu.

4.1.2 DevOps

Termín DevOps vychádza zo spojenia anglických slov Development a Operations. Ide o prístup k vývoju softvéru, ktorý kladie dôraz na komunikáciu medzi vývojármi a odborníkmi na prevádzku produkčnej infraštruktúry. [16]

Pre automatizáciu nasadzovania aplikácií na servery, známe pod anglickým termínom Continuous Integration & Continuous Deployment (CI/CD), som sa rozhodol využiť nástroj GitLab Pipelines. Vďaka jeho priamej integrácii s GITom stačilo napísať jednoduchý konfiguračný súbor vo formáte YML s definovanou adresou serveru, kam sa má aplikácia nasadiť. Následne pre nasadenie poslednej verzie aplikácie na, napríklad, produkčný server, stačí kliknutím na jedno tlačidlo spustiť automatizovaný skript, ktorý sa o všetko postará.

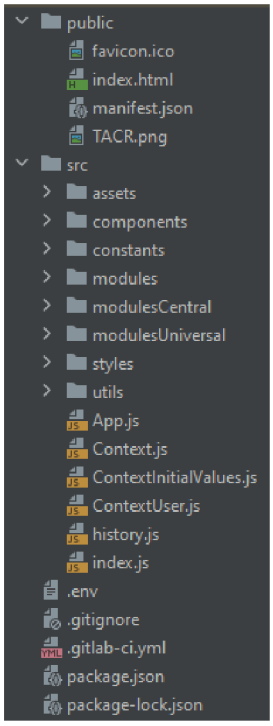
4.1.3 Hierarchia súborov

Medzi aspekty s výrazným dopadom na výslednú prehľadnosť a jednoduchosť navigácie v kóde patrí aj hierarchia súborov aplikácie. Pre zachovanie jednoduchosti a podobnosti s inými projektmi som nasledoval klasickú hierarchiu vychádzajúcu z hierarchie súborov CRA.

Kód aplikácie je rozdelený do zložky `public` (ktorá obsahuje univerzálne, statické súbory) a zložky `src` (ktorá obsahuje súbory zväčša Javascriptového kódu, z ktorých pozostáva samotná implementácia systému). Okrem toho sa na koreňovej úrovni nachádza aj súbor `package.json`, ktorý obsahuje metainformácie pre nástroj npm, definuje závislosti na iné Javascriptové knižnice a obsahuje príkazy skriptov pre spustenie vývojárskeho servera či zostavenie produkčnej verzie projektu.

Samotný kód v zložke `src` je ďalej rozdelený medzi zložku `assets` (ktorá obsahuje obrázky), zložku `components` (v ktorej sa nachádzajú jednotlivé komponenty Reactu, z ktorých sa následne skladajú jednotlivé stránky) a zložky `modules` (ktoré obsahujú kód pre jednotlivé

stránky aplikácie). Obsah ostatných zložiek intuitívne vyplýva z ich názvu.



Obr. 4.1: Hierarchia súborov aplikácie

4.1.4 Štruktúra React komponent

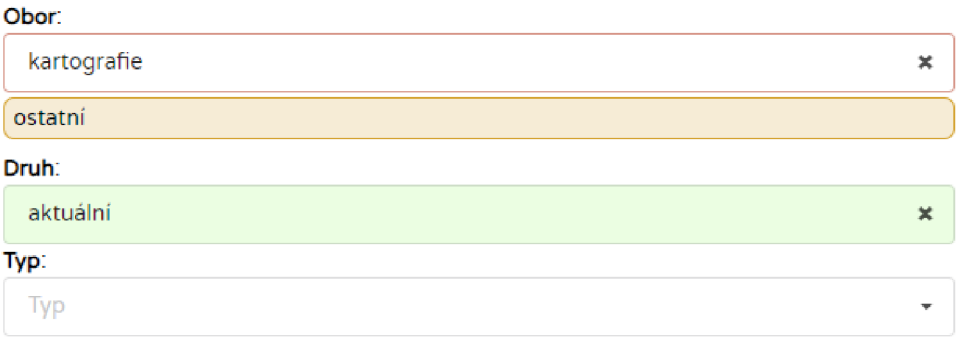
Celý projekt je obalený dvomi oddelenými kontextmi React API. Kontext Reactu umožňuje prístup k dátam kdekoľvek hlbšie v hierarchii aplikácie. Prvý z dvoch globálnych kontextov tohto projektu obsahuje konfiguračné dáta aplikácie a druhý informácie o momentálne prihlásenom užívateľovi.

Nasleduje opis smerovania jednotlivých aplikácií definovaného pomocou knižnice React Router. Na tomto mieste sú uvedené URL adresy všetkých podstránok aplikácie spolu s určeným komponentom, ktorý danú stránku reprezentuje. Následne sa spúšťa jedinečný kód modulu určeného momentálnou URL adresou.

4.1.5 Komponent editácie termínu

Najkomplexnejšou a pre implementáciu najnáročnejšou časťou aplikácie bol pre mňa komponent editácie jedného termínu. Pre zlepšenie prehľadnosti som sa ju rozhodol rozdeliť na niekoľko častí. Do samostatných súborov som oddelil definície vstupných polí pre každý z blokov dát a taktiež logiku validácie vložených dát pri odosielaní formuláru.

Jednou z požiadaviek bolo aj zvýrazňovanie zmenených polí v editačnom formulári. Systém podporuje taktiež komplexnú funkcionalitu schvaľovania zmien termínov a z tohto dôvodu je nutné rozlišovanie dvoch druhov zvýrazňovania. Oranžovou farbou sú zvýraznené zmeny čakajúce na schválenie a zelenou farbou systém zvýrazňuje zmeny vykonané počas momentálnej relácie. Knižnica Informed síce poskytuje jednoduchú funkcionalitu, ktorá vracia množinu vstupných polí, do ktorých užívateľ vpísal hodnotu, tú ale nebolo možné využiť. Je potrebné, aby sa farba poľa zmenila na pôvodnú, ak užívateľ vráti zmeny, ktoré vykonal. Pre dosiahnutie tejto funkcionality som musel implementovať vlastnosť ukladania počiatočnej hodnoty poľa, s ktorou sa po zmene porovnáva momentálna hodnota a pole sa zafarbí na zeleno len v prípade, ak sa hodnoty líšia.



Obr. 4.2: Editačný formulár so zvýraznenými poľami

Na obr. 4.2 je vidieť podobu editačného formuláru. Momentálne prihlásený užívateľ zmenil hodnotu poľa „druh" a vidí zmenu oboru z „ostatní" na „kartografie" vykonanú iným užívateľom, ktorú môže zamietnuť alebo schváliť.

4.2 Zmeny podľa spätnej väzby

Po skončení implementácie prototypu a dokončení aj všetkých potrebných zmien na back-ende sme nasadili systém na testovacie prostredie MV ČR. V priebehu niekoľko mesiacov trvajúceho testovacieho obdobia nám používatelia zasielali spätnú väzbu s ovládaním systému. Spätnú väzbu som delil na návrhy na pridanie nových funkcií či zmenu chovania stávajúcich a požiadavky na opravu chýb.

Spätnú väzbu používatelia zasielali zväčša elektronickou poštou. Každé hlásenie som vyhodnotil, v prípade nejasností kontaktoval používateľa, ktorý ho zaslal, a vytvoril ticket v systéme s popisom čo je potrebné zmeniť. Následne som jednotlivým ticketom priradil programátora, ktorý bude zodpovedať za vyhotovenie danej požiadavky. Každý ticket symbolizuje systémom automaticky vygenerovaný unikátny identifikátor. Po tom, ako programátor implementoval vyžadovanú funkcionalitu alebo opravil chybu, pridal identifikátor odpovedajúceho ticketu pre zlepšenie prehľadnosti do takzvanej commit message verzovacieho systému GIT.

Celkovo sme spolu implementovali 169 zmenových požiadaviek a opráv chýb. Jednou z najväčších zmien vo fungovaní systému, ktorá bola požadovaná, bolo pridanie niekoľkých ďalších dátových blokov do stránky detailu jedného termínu. Táto požiadavka vyžadovala nielen jednoduché pridania zobrazovacích blokov, ale aj pridanie logiky pre editáciu termínu vrátane validácie a autorizačných pravidiel užívateľských rolí a aj zmeny back-endu.

Záver

Hlavným cieľom diplomovej práce bolo vytvorenie webových aplikácií pre editáciu a správu systému terminologických tezaurov. Celý nami vytvorený systém sa skladá z dvoch rozdielnych webových aplikácií. Prvou je aplikácia centrálneho systému pre správu tezaurov. Predstavuje zoznam všetkých inštancií tezaurov patriacich pod jednu doménu, spoločnosť, organizáciu či majúcich spoločné zameranie. Každý tezaurus má uvedené základné údaje o sebe, svojom prevádzkovateľovi a prípadne aj kontakt na zodpovedné osoby a ďalšie technické informácie. Druhou je aplikácia jednej inštancie samotného slovníka, ktorá obsahuje jednotlivé termíny a zoznam používateľov s nastavenými užívateľskými rolami.

Aplikácia inštancie tezauru je samostatne použiteľná a môže existovať aj nezávisle bez napojenej aplikácie centrálneho systému. Na druhej strane aplikácia centrálneho systému sama o sebe neposkytuje žiadne odborné či lexikálne informácie, slúži len ako rozcestník odkazujúci na jednotlivé inštancie tezaurov, takže sama o sebe, bez napojených slovníkov, je nepoužiteľná.

Súčasťou mojej práce bola aj komunikácia s klientom a riadenie vývoja webového rozhrania. Jednalo sa ale prvorado o reálny projekt, ku ktorému som sa dostal až v momente, keď bol zoznam funkčných a nefunkčných požiadaviek vypracovaný. Preto som ho nevypracovával ja a prikladám ho k diplomovej práci mimo hlavného textu len ako prílohu. Okrem toho som sa ale venoval všetkým ďalším povinnostiam v rámci riadenia vývoja softvéru.

Cieľ diplomovej práce považujem za dosiahnutý – vyvinuli sme dve funkčné webové aplikácie, ktoré svojim užívateľom už dnes v praxi pomáhajú a poskytujú reálny prínos. Implementácia front-endu aplikácií je k dispozícii v GIT repozitári[1] a v dobe písania diplomovej práce je testovacia verzia aplikácie dostupná na adrese http://tezaurus-cuzk.ogibeta2.gov.cz/tezaurus.

1. Odkaz na projekt v GitLab FI MU: https://gitlab.fi.muni.cz/xmach5/tezaurus

Bibliografia

1. ČERMÁK, František. *Jazyk a jazykověda*. Praha: Karolinum, 2009. ISBN 9788024619460.
2. WALTER, Aarron. *Designing for emotion*. New York: A Book Apart / Jeffrey Zeldman, 2011. ISBN 9780132858311.
3. FESSENDEN, Therese. *A Theory of User Delight: Why Usability Is the Foundation for Delightful Experiences* [online]. Nielson Norman Group, 2017 [cit. 2020-12-13]. Dostupné z: `https://www.nngroup.com/articles/theory-user-delight/`.
4. BUXTON, Bill. *Sketching User Experiences: Getting the Design Right and the Right Design*. Morgan Kaufmann, 2007. ISBN 9780123740373.
5. *Landing Pages: The Complete Guide to Effective UX Design* [online]. UXPin, 2010/2020 [cit. 2020-12-13]. Dostupné z: `https://www.uxpin.com/studio/blog/landing-pages-guide/`.
6. MATSUMOTO, Yukihiro. *The Ruby Programming Language: Everything You Need to Know*. O'Reilly Media, 2008. ISBN 9780596516178.
7. JAVINPAUL. *10 of the Most Popular JavaScript Frameworks Libraries for Web Development in 2021* [online]. Javarevisited – Medium, 2019-09-15 [cit. 2020-12-13]. Dostupné z: `https://medium.com/javarevisited/10-of-the-most-popular-javascript-frameworks-libraries-for-web-development-in-2019-a2c8cea68094`.
8. *What is Vue.js?* [Online] [cit. 2020-12-13]. Dostupné z: `https://vuejs.org/v2/guide/#What-is-Vue-js`.
9. NOWAK, Maja. *Vue vs React in 2020: Which Framework to Choose and When* [online]. Monterail, 2020-08-06 [cit. 2020-12-13]. Dostupné z: `https://www.monterail.com/blog/vue-vs-react-2020`.
10. *Introduction to Angular* [online] [cit. 2020-12-13]. Dostupné z: `https://angular.io/docs`.
11. MOOR, Thomas De. *React vs Angular: Their Biggest Differences* [online]. X-Team, 2020 [cit. 2020-12-13]. Dostupné z: `https://x-team.com/blog/react-vs-angular/`.

12. *Redux vs Context vs Local Component State: state management solutions for React* [online]. Merixstudio, 2020 [cit. 2020-12-13]. Dostupné z: https://www.merixstudio.com/blog/redux-vs-context-vs-local-component-state/.

13. BARR, Earl T. *To Type or Not to Type: Quantifying Detectable Bugs in JavaScript* [online]. 2017 [cit. 2020-12-13]. Dostupné z: http://earlbarr.com/publications/typestudy.pdf.

14. ELLIOTT, Eric. *The TypeScript Tax: A Cost vs Benefit Analysis* [online]. Medium, 2019 [cit. 2020-12-13]. Dostupné z: https://medium.com/javascript-scene/the-typescript-tax-132ff4cb175b.

15. JADHAO, Vishal. *10 Best React UI Component Frameworks* [online]. Medium, 2018-06-07 [cit. 2020-12-13]. Dostupné z: https://medium.com/@vishal75j_14174/10-best-react-ui-component-frameworks-6303b4447766.

16. PANT, Rajiv. *Organizing a Digital Technology Department of Medium Size in a Media Company* [online]. RAJIV.COM, 2009-03-17 [cit. 2020-12-13]. Dostupné z: https://www.rajiv.com/blog/2009/03/17/technology-department/.

A Funkčné a nefunkčné požiadavky

Zoznam funkčných a nefunkčných požiadaviek je k dispozícii v Informačnom systéme Masarykovej univerzity.

B Príručky pre používateľov aplikácie

Užívateľská a editorská príručka pre aplikáciu tezauru sú k dispozícii v Infomačnom systéme Masarykovej univerzity.

C Zdrojové kódy

Zdrojové kódy sú k dispozícii v Infomačnom systéme Masarykovej univerzity a GIT repozitári: https://gitlab.fi.muni.cz/xmach5/tezaurus