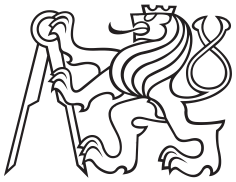
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra počítačů

Mobilní aplikace pro tvorbu sémantických tezaurů

Filip Kopecký

Vedoucí: Ing. Petr Křemen, Ph.D.
Studijní program: Softwarové inženýrství a technologie
Květen 2021

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Kopecký** Jméno: **Filip** Osobní číslo: **483821**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Softwarové inženýrství a technologie**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Mobilní aplikace pro tvorbu sémantických tezaurů

Název bakalářské práce anglicky:

Mobile application for semantic vocabularies

Pokyny pro vypracování:

Cílem práce je navrhnout a implementovat mobilní aplikaci pro prohlížení a správu propojených sémantických tezaurů. Aplikace tak bude moci sloužit jako vyhledávač slovníků, pojmů, jejich vzájemných významových ontologických vztahů, dále pak jako nástroj pro jejich komentování a úpravy.
1) Seznamte se se standardy sémantického webu, zejména SKOS, RDF, SPARQL. Seznamte se s uživatelským a aplikačním rozhraním systému TermIt pro správu sémantických tezaurů.
2) Navrhněte mobilní aplikaci pro vyhledávání, prohlížení a správu propojených sémantických tezaurů spravovaných v systému TermIt.
3) Implementujte mobilní aplikaci pomocí zvolené technologie (např. React Native)
4) Navrhněte a proveďte uživatelské testy aplikace na vybraných propojených sémantických tezaurech.

Seznam doporučené literatury:

[1] M. Ledvinka, P. Kremen, L. Saeeda, M. Blasko. TermIt: A Practical Semantic Vocabulary Manager. ICEIS (1), 759-766
[2] SKOS Simple Knowledge Organization System Reference. W3C recommendation 2009. Online at https://www.w3.org/TR/2009/REC-skos-reference-20090818/.
[3] P. Kremen, M. Necaský. Improving discoverability of open government data with rich metadata descriptions using semantic government vocabulary. J. Web Semant. 55: 1-20 (2019)

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Petr Křemen, Ph.D., skupina znalostních softwarových systémů FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **11.02.2021** Termín odevzdání bakalářské práce: **21.05.2021**

Platnost zadání bakalářské práce: **30.09.2022**

Ing. Petr Křemen, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

.	
Datum převzetí zadání	Podpis studenta

Poděkování

Rád bych poděkoval vedoucímu práce Ing. Petru Křemenovi, Ph.D. za jeho ochotu, trpělivost a věcné připomínky v průběhu vypracovávání bakalářské práce. Díky nechybí ani testerům, kteří poskytli nezbytnou zpětnou vazbu. V neposlední řadě bych rád poděkoval své matce, která mě plně podporovala během celého trvání mého studia.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně, a že jsem uvedl veškerou použitou literaturu.

V Praze, 21. května 2021

Abstrakt

Motivací této bakalářské práce bylo vytvoření softwarové analýzy multiplatformní mobilní aplikace pro systém TermIt a její následná implementace. Práce objasňuje principy a technologie, které systém TermIt používá, společně s popisem jeho funkcionalit a vlastností. Zabývá se problematikou vývoje mobilních aplikací a předkládá argumenty pro možné volby vývojových přístupů, spjatých s mobilním vývojem. Popisuje implementační část práce spolu s rozhodnutími, které byly během vývoje učiněny. Vyhodnocuje výsledky uživatelského testování mobilní aplikace a srovnává současné řešení nabízené systémem TermIt s nově vzniklým.

Klíčová slova: softwarová analýza, mobilní aplikace, TermIt, Android, sémantický web

Vedoucí: Ing. Petr Křemen, Ph.D.

Abstract

The motivation for this thesis was the creation of software analysis for cross-platform mobile application and its later implementation. The thesis explains the core principles and technologies around which is system TermIt built. Description of features and properties of the system are explained as well. Then discusses new trends and possibilities of modern mobile app development. The thesis then explains the implementation details of the mobile application and important decisions that were made during development. Last but not least, the conclusion of whether a newly created mobile app is a usable and preferable option for users interacting with system TermIt is drawn.

Keywords: software analysis, mobile application, TermIt, Android, semantic web

Title translation: Mobile application for semantic vocabularies

Obsah

Obrázky

Tabulky

Kapitola 1

Úvod

Otevřená data [18] jsou informace zveřejňované takovým způsobem, aby přístup k nim byl možný realizovat dálkově, v otevřeném a strojově čitelném formátu. Jedná se o data, která je možné volně používat a šířit. Příkladem mohou být otevřená data veřejné správy (její datové sady). Nejen že mohou být nápomocné uživatelům v každodenním životě, ale zároveň dodávají i jistou transparentnost do fungování státu.

K tvorbě dokumentace datových sad lze využít sémantické slovníky pojmů. Tyto slovníky reprezentují významy jednotlivých pojmů obsažených v datových sadách, jejich definice a vzájemné sémantické vazby. Tento popis je vhodný, jelikož vyhledávače nejsou schopné, pouze na základě názvu pojmu, rozlišit jejich kontextuální význam. Příkladem může být pojem *stavba*, který lze chápat jako podstatné jméno, ale i jako sloveso v závislosti na kontextu. Zároveň i samotný význam podstatného jména *stavba* je nutné specifikovat, jelikož rozdílné zákony se mohou ve svém výkladu pojmu rovněž lišit.

Systém TermIt[1], jako nástroj pro správu takových slovníků, vznikl ve skupině znalostních a softwarových systému na Katedře počítačů FEL ČVUT. Nástroj umožňuje uživatelům pracovat s propojenými slovníky, rozlišovat a identifikovat jednotlivé pojmy v zdrojových dokumentech a vytvářet jejich vzájemné vazby. Tyto pojmy jsou použity pro sémantickou anotaci datových zdrojů a pro následné vyhledávání.

TermIt je v současné době uživatelům dostupný pouze jako webová aplikace, která byla vytvořena primárně pro používání na počítači. Webová aplikace je sice částečně responzivní (uzpůsobující se rozměrům obrazovky), ale výsledek, který je prezentován na mobilních zařízeních, není uspokojivý. Rozmístění prvků webové aplikace spuštěné na mobilním zařízení působí nahodile a nepřehledně. Tento problém má vyřešit nově vzniklá mobilní aplikace.

Mobilní aplikace je navržena tak, aby její uživatelé měli k dispozici základní funkcionalitu nástroje TermIt v přehledné podobě na mobilních zařízeních. Aplikace se od webové verze liší především svým uzpůsobeným vzhledem pro mobilní zařízení, využíváním interakce pomocí gest a samotnou rychlostí svého používání. Nadále přidává funkce, které webová verze neposkytuje v podobě možnosti změny koncové adresy serveru, správu oblíbených položek či evidenci historie prohlížených položek.

Začátek dokumentu se zabývá sémantickým webem a technologiemi, které ho tvoří. Dále přibližuje funkcionalitu systému TermIt a popisuje jeho technické zpracování.

Další část dokumentu obsahuje návrh systému mobilní aplikace. Definuje její cíl a zmiňuje technologie, díky kterým je aplikace vytvořena. Definuje funkční a nefunkční požadavky společně s případy užití. Součástí je i vysvětlený doménový model aplikace.

Následující kapitola je věnována technické části projektu. Rozebírá dva možné přístupy vývoje mobilních aplikací, které posléze srovnává. Dalším předmětem porovnání a popisu byly frameworky umožňující tvorbu multiplatformních aplikací. Následuje detailní popis frameworku React Native, který byl zvolen pro vývoj mobilní aplikace. V neposlední řadě objasňuje způsob, jakým byl projekt pro zvolený framework založen a zmiňuje technologie, které se v projektu vyskytují.

[1] http://kbss.felk.cvut.cz/termit navštíveno: 30.12.2020

Další kapitola se zabývá implementační částí mobilní aplikace. Popisuje, jakým způsobem je řešena problematika globálního stavu v aplikaci. Dále se zabývá navigační strukturou aplikace spolu s její implementací. Rovněž popisuje, jak probíhá komunikace aplikace se serverem a ukládání dat do paměti mobilního zařízení.

Předposlední kapitola je věnována testování mobilní aplikace. Popisuje druhy testů, které byly provedeny a nástroje k tomu použité. Zhodnocuje výsledky uživatelských testů a nabízí možné řešení nedostatků či problémů, které byly v aplikaci nalezeny.

Poslední kapitola srovnává mobilní aplikaci s již existujícím řešením mobilní webové verze systému TermIt. Srovnání ukazuje, zdali má nově vzniklá mobilní aplikace své opodstatnění a jestli splnila své předem vytyčené cíle uživatelské přívětivosti.

Kapitola 2

Popis problematiky

Tato kapitola popisuje technologické pozadí projektu, konkrétně technologie a architektury, na kterých stojí systém TermIt, jehož API je klíčovou složkou nové mobilní aplikace.

2.1 Sémantický web

Sémantický web [38] je rozšíření World Wide Webu, na kterém pracuje konsorcium W3C[1], kde namísto webových stránek, které jsou čitelné pouze lidmi, bude obsah reprezentován na základě svého významu ve strojově čitelné podobě. Sémantický web pro svojí funkcionality využívá hned několik technologií, mezi které například patří: RDF (sekce: 2.2), RDFS [30], SKOS (sekce: 2.3), SPARQL (sekce: 2.4) a OWL [28]. Použitím těchto technologií vzniká web navzájem propojených dat, které jsou pochopitelné strojem. Hlavním cílem je mezi těmito daty efektivně vyhledávat, automatizovat práci s nimi, sjednocovat je a použít je napříč mnoha aplikacemi.

2.2 RDF

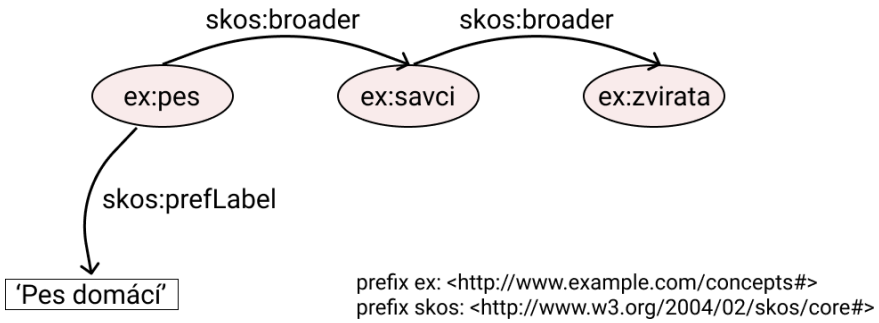
RDF [11] (neboli Resource Description Framework) je datový model podobný klasickému konceptuálnímu modelování. Jeho základní myšlenkou je vytváření tvrzení o zdrojích ve tvaru *subject* - *predicate* - *object* (tzv. triplets). Jinými slovy, zdroj (subject) má vlastnost (predicate) o určité hodnotě (object). Jednoduchý příklad může být *obloha* - *má barvu* - *modrou*. Tento způsob zápisu je strojově i lidsky čitelný. Množina RDF tvrzení tvoří popsaný orientovaný graf.

[1] https://www.w3.org/ navštíveno: 25.12.2020

2.3 SKOS

SKOS [26] (neboli Simple Knowledge Organization System) je doporučení konsorcia W3C pro reprezentaci tezaurů. SKOS poskytuje model pro popis základní struktury a obsahu koncepčních schémat jakou jsou tezaury, taxonomie, klasifikační schémata či jakékoliv jiné řízené slovníky v rámci sémantického webu využívající standardy RDF, RDFS a OWL.

Základním prvkem je třída *Concept* a jejich množina reprezentovaná třídou *Concept scheme*. Jednotlivé prvky třídy *Concept* můžeme popsat dalšími vlastnostmi, jakými například jsou: *prefLabel* (preferovaný název) nebo *definition* (definice). Rovněž je můžeme zařadit do hierarchické struktury například pomocí vlastností *narrower* (prvek má bližší specifikací) nebo *broader* (prvek má obecnější specifikaci). [7] Níže (obrázek: 2.1) je ukázán RDF graf obohacený o SKOS třídy a vlastnosti. Díky nim je například specifikováno, že pes patří do savců.



Obrázek 2.1: RDF graf s SKOS

2.4 SPARQL

Jedná se o sémantický dotazovací jazyk pro data ve formátu RDF. Jinými slovy, SPARQL [34] umožňuje vytvářet dotazy nad RDF daty a nadále pracovat s jejich výsledky. Jeho syntaxe může připomínat dotazovací jazyk SQL [15] pro relační databáze. I zde totiž existují klíčová slova, jakou jsou např: SELECT, WHERE, FILTER nebo agregační funkce: COUNT, SUM, AVG. Podobností v syntaxi je více, není je ale nutné vyjmenovávat všechny. Klíčové slovo SELECT určuje požadovanou proměnnou či proměnné na výstupu, WHERE určuje grafový vzor a FILTER vymezuje výsledky.

Pro názornou ukázku použití těchto klíčových slov v jazyce SPARQL si dovolím jednoduchou ukázku dotazu (obrázek: 2.3), který na přiloženém datasetu (obrázek 2.2) vrátí název a cenu knih, jejíž cena je menší než 30.

V dotazu se objevuje klíčové slovo PREFIX, díky kterému tvoříme zkratku pro jmenný prostor. Dataset (RDF graf) je popsán za pomocí Turtle [8] syntaxe.

```
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix : <http://example.org/book/> .
@prefix ns: <http://example.org/ns#> .

:book1 dc:title "SPARQL Tutorial".
:book1 ns:price 42 .
:book2 dc:title "The Semantic Web".
:book2 ns:price 23 .
```

Obrázek 2.2: Ukázkový dataset [29]

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX ns: <http://example.org/ns#>
SELECT ?title ?price
WHERE { ?x ns:price ?price .
FILTER (?price < 30)
?x dc:title ?title . }
```

Obrázek 2.3: Ukázka dotazu v dotazovacím jazyce SPARQL [29]

2.5 REST API

Jedná se o architekturu rozhraní navrženou pro distribuované prostředí. REST [10] na rozdíl od jiných architektur (např. SOAP [22]) je orientován datově, nikoli procedurálně. Nejčastěji ho nalézáme v kombinaci s protokolem HTTP [33]. Výhodou rozhraní je, že nabízí jednotný a jednoduchý přístup ke zdrojům. Zdroje jsou přístupné pod jednoznačným identifikátorem URI a přístup k nim je řízen předem definovanými metodami. Základní metody jsou uvedeny níže (tabulka: 2.1), společně s jejich vazbou na CRUD [19] a jejich popisem.

HTTP metoda	**CRUD**	**Popis**
GET	R (Retrieve)	Vrátí reprezentaci dotazovaného zdroje
POST, (PUT)	C (Create)	Vytvoří reprezentaci zdroje na serveru
PUT, PATCH	U (Update)	Změní stav zdroje
DELETE	D (Delete)	Smaže dotazovaný zdroj

Tabulka 2.1: Popis základních HTTP metod

2.6 TermIt

TermIt [25] je nástroj pro správu SKOS (sekce: 2.3) tezaurů, postavený na architektuře sémantického webu (sekce: 2.1). Umožňuje vyhledávání a práci se sémantickými slovníky. Tyto slovníky obsahují pojmy, které si mohou uživatelé prohlížet. Slovníkům a pojmům je možné přidávat a odebírat atributy, které je nadále rozvádějí, čímž upřesňují jejich význam. Zároveň lze definovat vzájemné vztahy mezi pojmy a slovníky a tím vytvářet jejich hierarchickou strukturu. Rovněž systém umožňuje spravovat dokumenty obsahující pojmy ze slovníků, popřípadě pojmy v dokumentech vyhledávat.

Jelikož se jedná o data, která jsou strojově čitelná, je možné je rovnou publikovat jako propojená data [5] nebo je použít k datovému modelování.

Systém TermIt dělíme na dvě části:

1. Serverová část - Poskytující REST API
2. Front-endová část - Webová aplikace

2.6.1 Technické zpracování

Serverová část[2] je open source aplikací napsanou v programovacím jazyce Java. Data, se kterými aplikace pracuje, jsou uložena v RDF (sekce: 2.2) databázi. Nad touto databází jsou prováděny dotazy v sémantickém dotazovacím jazyce SPARQL (sekce: 2.4). Zpracovaná data jsou poskytnuta díky REST API (sekce: 2.5). Ta jsou ve výchozí podobě ve formátu JSON-LD [23], je ale možné použít obyčejný JSON formát [16]. Tabulka v příloze (sekce: F.1) obsahuje API koncové body[3], které mobilní aplikace využívá.

Front-endová část[4] systému TermIt je aplikace, psaná v programovacím jazyce TypeScript, používající React framework.

2.6.2 Pojmy

Uživatelé systému TermIt si mohou pojmy vyhledávat a zobrazovat jejich atributy. Ukázku předních atributů naleznete dále v textu (tabulka: 2.3). Pro větší přehlednost bylo jejich URI zapsáno za pomocí prefixů. Prefixy jmenných prostorů jsou vypsány na další stránce (tabulka: 2.2).

TermIt rovněž umožňuje vyhledávat související pojmy na základě ontologických vztahů. Příkladem může být pojem *Plocha* ze slovníku *Slovník zákona č. 183/2006 Sb. (Stavební zákon)*, který ontologicky souvisí s pojmem *Transformační plocha* ze slovníku *Slovník Pražských stavebních předpisů 2016*. Ačkoliv jsou oba záznamy z jiných slovníku, pojem *Plocha* je pojmu *Transformační plocha* nadřazený.

Příklad části výstupu, resp. jeho hodnot, které vidí uživatel systému TermIt, jsou ukázány na další stránce (tabulka: 2.4). Ačkoliv jsou hodnoty atributů prezentovány pouze jako texty (či odkazy), jejich význam je mnohem přesnější, díky definici atributů za pomocí SKOS vlastností. Příkladem může být již

[2] Zdrojové kódy serverové části: https://github.com/kbss-cvut/termit navštíveno: 18.5.2020

[3] Dokumentace celého API: https://app.swaggerhub.com/apis/ledvima1/TermIt/ navštíveno: 18.5.2020

[4] Zdrojové kódy front-end části: https://github.com/kbss-cvut/termit-ui navštíveno: 18.5.2020

zmiňovaný atribut podřazeného pojmu, který systém skutečně chápe jako určení hierarchického postavení, odkazující se na skutečný pojem.

prefix	**jmenný prostor**
skos	`http://www.w3.org/2004/02/skos/core#`
rdfs	`http://www.w3.org/2000/01/rdf-schema#`
rdf	`http://www.w3.org/1999/02/22-rdf-syntax-ns#`
popdat	`http://onto.fel.cvut.cz/ontologies/slovník/agendový/popis-dat/pojem/`
dc	`http://purl.org/dc/elements/1.1/`

Tabulka 2.2: Výpis jmenných prostorů a jejich prefixů

Název	**URI**
Název	skos:prefLabel
Definice	skos:definition
Zdroj definice	dc:source
Typ pojmu	rdf:type
Nadřazené pojmy	skos:broader
Podřazené pojmy	skos:narrower
Doplňující poznámka	skos:scopeNote
Zdroj pojmu	popdat:zdroj
Slovník	popdat:je-pojmem-ze-slovníku

Tabulka 2.3: Výpis atributů pojmů a jejich URI

Atribut	**Hodnota zobrazená uživateli**
Název	Plocha
Definice	Část území tvořená jedním či více pozemky nebo jejich částí, která je vymezena v politice územního rozvoje, zásadách územního rozvoje nebo územním plánu, popřípadě v územně plánovacích podkladech s ohledem na stávající nebo požadovaný způsob jejího využití a její význam
Zdroj definice	https://www.zakonyprolidi.cz/cs/2006-183#p2-1-g
Typ pojmu	Typ objektu
Nadřazené pojmy	Část území Plocha, část plochy, nebo soubor ploch
Podřazené pojmy	Transformační plocha Koridor Stavební blok
Slovník	Slovník zákona č. 183/2006 Sb. (Stavební zákon) - slovník

Tabulka 2.4: Ukázka zpracovaného výstupu TermIt

2.6.3 Slovníky

Uživatelé mohou slovníky vyhledávat a zobrazovat si jejich atributy. Hlavní výpis slovníku, který uživatel systému TermIt vidí, obsahuje informace o jeho názvu, popisu, importovaných slovnících a obsažených pojmech.

Kapitola 3

Návrh systému

Tato kapitola popisuje návrh systému mobilní aplikace. Nově vzniklá mobilní aplikace má přenést základní funkcionalitu systému TermIt na mobilní zařízení. Rozsah systému je přiblížen za pomocí funkčních požadavků a případů použití. Dále jsou definovány nefunkční požadavky a je popsán datový model mobilní aplikace.

3.1 Technické parametry projektu

Aplikace bude vypracována za pomocí frameworku React Native (sekce: 4.3). Hlavním zdrojem dat bude REST API serverové části systému TermIt. Jednotlivé aspekty technického zpracování jsou popsány v technické analýze projektu (kapitola: 4).

3.2 Funkční požadavky

Funkční požadavky [41] popisují funkce systému nebo jeho částí, které musí systém poskytovat. Mezi požadavky můžeme řadit výpočty, technické detaily, manipulace a zpracování dat či jiné procesy, které musí systém vykonávat. Jedná se o charakterizaci určitého výsledku systému. Funkční požadavky pro mobilní aplikaci jsou vypsány na další stránce.

- FR:1 Přihlášení
  - Aplikace musí umožňovat uživatelům se přihlásit.
- FR2: Odhlášení
  - Aplikace musí umožňovat uživatelům se odhlásit.
- FR3: Vyhledávání pojmů a slovníků
  - Aplikace musí uživateli poskytnout vyhledávání pojmů a slovníků.
- FR4: Zobrazení detailu pojmu
  - Aplikace musí být schopná zobrazit uživateli název, definici pojmu, typ pojmu, zdroj definice, příslušný slovník a podřazené pojmy.
- FR5: Zobrazení detailu slovníku
  - Aplikace musí být schopná zobrazit uživateli popis slovníku a seznam přiřazených pojmů k němu. Seznam přiřazených pojmů bude zobrazen hierarchicky, tj. pojmy obsahující pojmy podřazené je budou moci na vyžádání zobrazit.
- FR6: Vytvoření pojmu
  - Aplikace musí umožňovat oprávněným (neomezeným) uživatelům vytvořit nový pojem pro vybraný slovník. Uživatel musí vyplnit unikátní název pojmu pro daný slovník, dále může vyplnit následující atributy: definice, zdroj definice, synonyma, typ a nadřazené pojmy v daném slovníku. Nadřazené pojmy se volí vzhledem k jejich hierarchické pozici ve slovníku vůči ostatním pojmům.
- FR7: Vytvoření slovníku
  - Aplikace musí umožňovat oprávněným (neomezeným) uživatelům vytvořit nový slovník. Uživatel musí vyplnit unikátní název a může vyplnit popis slovníku.
- FR8: Upravení pojmu
  - Aplikace musí umožňovat oprávněným (neomezeným) uživatelům upravovat pojmy ve slovníku, tzn. jejich název, definici, zdroj definice, synonyma, typ a nadřazené pojmy v daném slovníku. Nadřazené pojmy se volí vzhledem k jejich hierarchické pozici ve slovníku vůči ostatním pojmům.
- FR9: Upravení slovníku
  - Aplikace musí umožňovat oprávněným (neomezeným) uživatelům upravovat slovníky, tzn. jejich název a popis.

- FR10: Mazání pojmu
  - Aplikace musí umožňovat oprávněným (neomezeným) uživatelům odstranit pojem ze slovníku.
- FR11: Mazaní slovníku
  - Aplikace musí umožňovat oprávněným (neomezeným) uživatelům odstranit slovník, neobsahující žádné pojmy.
- FR12: Okomentování pojmu
  - Aplikace musí umožňovat všem přihlášeným uživatelům přidávat komentáře k pojmům.
- FR13: Ukládání pojmu
  - Aplikace musí umožňovat všem uživatelům ukládat pojmy do seznamu oblíbených položek.
- FR14: Ukládání slovníku
  - Aplikace musí umožňovat všem uživatelům ukládat slovníky do seznamu jejich oblíbených položek.
- FR15: Zobrazení oblíbených položek
  - Aplikace musí umožňovat všem uživatelům zobrazit seznam jejich oblíbených položek.
- FR16: Odebrání oblíbené položky
  - Aplikace musí umožňovat všem uživatelům odebrání pojmu či slovníku z jejich seznamu oblíbených položek.
- FR17: Zobrazení nedávné historie
  - Aplikace musí umožňovat všem uživatelům zobrazit jejich nedávnou historii vyhledávaných pojmů a slovníků (posledních 5 položek).
- FR18: Vyčištění historie
  - Aplikace musí umožňovat všem uživatelům vymazání všech položek jejich nedávné historie.
- FR19: Změna URL adresy serveru
  - Aplikace musí umožňovat všem uživatelům před přihlášením nastavit URL adresu serveru.
- FR20: Kopírování do schránky
  - Aplikace musí umožňovat kopírovat definice, zdroje definic, názvy a synonyma pojmů do schránky. Kopírovat lze i názvy a popisy slovníků.

- FR21: Výpis slovníků
  - Aplikace musí umožňovat zobrazení všech slovníků dostupných na serveru.

3.3 Nefunkční požadavky

Nefunkční požadavky [42] jsou kritéria, podle kterých lze hodnotit kvalitativní funkce systému. Definují celkové vlastnosti systému a jeho omezení. Nefunkční požadavky pro mobilní aplikace jsou uvedeny níže.

- NFR1: Podpora operačního systému Android
  - Aplikace musí fungovat na systémech Android (verze 6 a vyšší).
- NFR2: Přizpůsobení velikost
  - Aplikace se musí přizpůsobit rozměru mobilního zařízení, na kterém je spuštěna.
- NFR3: Startovací čas
  - Aplikaci by nemělo trvat déle než 3 vteřiny načíst úvodní stránku.
- NFR4: Množství použití dat
  - Aplikaci by měla vytvářet co možná nejmenší počet API volání, aby nedocházelo k zbytečnému zatěžování sítě či plýtvání mobilních dat.

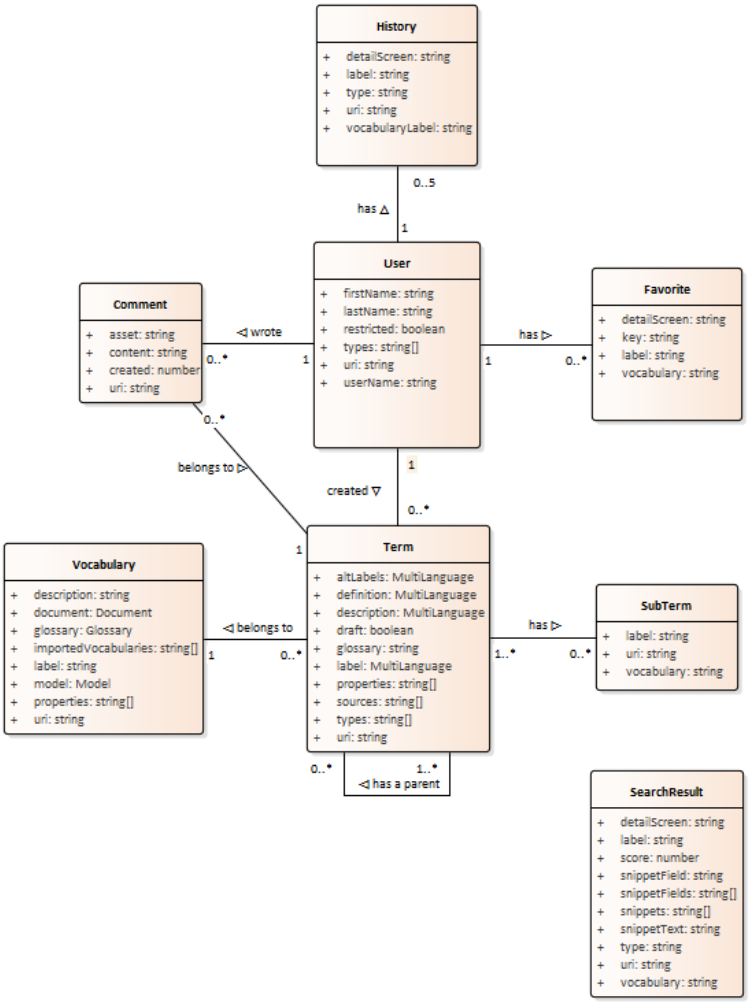
3.4 Případy použití

Případy užití [6] jsou množiny akcí, které vedou k dosažení určitého cíle. Jedná se o definici funkcionalit systému, se kterým účastníci interagují. Účastníkem může být uživatel, externí systém nebo čas. Z pohledu systému se jedná o externí entitu. Charakterizují tedy jakým způsobem vnější subjekty používají systém. Individuální případy užití popisují jednotlivé funkční požadavky a nadále je rozvádějí. V příloze (příloha: A) jsou uvedeny jednotlivé případy užití mobilní aplikace (anglicky. use cases) i s vysvětlením jejich struktury.

3.5 Doménový model

Doménový model představuje strukturovanou vizuální reprezentaci entit s jejich vlastnostmi a vzájemnými vazbami. [9] Doménový model, se kterým pracuje mobilní aplikace, vznikl odvozením doménového modelu serverové části systému TermIt. Z něj jsou převzaty ty entity, které poskytuje server na svém API rozhraní. Zároveň je model obohacen o entity specifické pro mobilní aplikaci, jakou je například entita Favorite (oblíbená položka). Entity s jejich atributy jsou popsány v příloze (příloha: B).

Diagram (obrázek: 3.1) vyobrazuje entity, jejich atributy a vzájemné vazby mezi nimi.



Obrázek 3.1: Doménový model mobilní aplikace

Kapitola 4

Technická analýza

Tato kapitola se zabývá technickou stránkou projektu. Popisuje možné vývojové postupy tvorby mobilních aplikací. Tyto postupy následně porovnává a zdůvodňuje proč byl daný způsob tvorby zvolen. Vybraný typ tvorby sebou nese další rozhodnutí v podobě použitého frameworku. Nejdříve jsou frameworky a jejich základní koncepty představeny a následně porovnány. Vybraný framework je posléze popsán do hloubky i s technologiemi s ním spjatými.

4.1 Typ aplikace

Před zahájením vývoje mobilní aplikace je důležité rozhodnout, zdali se bude jednat o aplikaci nativní (sekce: 4.1.1) pro daný operační systém, nebo o multiplatformní aplikaci (sekce: 4.1.2).

4.1.1 Nativní aplikace

Nativní aplikace je taková aplikace, která je vyvíjena pro specifickou platformu. Aplikace jsou tedy psané v takovém programovacím jazyce, který je danou platformou podporován. Operační systém od firmy Apple (iOS) podporuje jazyk Objective C a Swift. Android (od společnosti Google) se přiklání k jazyku Java nebo Kotlin. Nativní aplikace se pyšní především svojí výkonností a možností robustní funkcionality.

4.1.2 Multiplatformní aplikace

Multiplatformní aplikace je vyvíjena pro více operačních systémů současně (například iOS a Android). Aplikace tedy sdílí společnou code-base (zdrojový kód). Pro tvorbu multiplatformních aplikací existuje nespočet frameworků. V současné době mezi ty nejpopulárnější patří React Native, Flutter a Xamarin (více o těchto frameworcích v sekci 4.2).

4.1.3 Srovnání přístupů

Bohužel nelze jednoznačně určit který vývojový přístup tvorby mobilních aplikací je bezprostředně lepší. Existují ale jisté rozhodující faktory, které mohou být pro potřeby programátora klíčové. Ve srovnání níže (tabulka: 4.1) jsou jednotlivé klíčové faktory zmíněny a porovnány pro jednotlivé vývojové přístupy.

Parametr	Nativní aplikace	Multiplatformní aplikace
Výkon aplikace	Vysoký	Nižší
Přístupnost	Pouze na vyvíjené platformě	Více platforem současně
Dostupné funkce	Plný přístup k funkcím systému	Omezený přístup k funkcím systému

Tabulka 4.1: Srovnání nativního a multiplatformního přístupu vývoje

Pro potřeby bakalářské práce se zdá být cesta **multiplatformní aplikace** tou vyhovující. Plný přístup ke všem funkcím mobilního zařízení není nutný a výkon aplikace není relevantním hodnotícím parametrem (aplikace nebude vykonávat náročné grafické operace). Možnost rozšíření aplikace na více platforem je rovněž žádoucí.

4.2 Frameworky

Oblíbenost multiplatformních aplikací roste a s ní i množství nástrojů, díky kterým se dají tyto aplikace tvořit. V následujících podkapitolách jsou stručně popsány nejoblíbenější frameworky na trhu.

4.2.1 Flutter

Flutter[1] je open-source projekt vyvíjen společností Google umožňující vývoj webových aplikací a aplikací pro Android, iOS, Linux, Mac, Windows a Google Fuchsia pod jedním zdrojovým kódem. Tento zdrojový kód je psán v programovacím jazyce Dart. Ten je relativně nový (svoji první verzi uvedl v roce 2013) a nedisponuje velikou komunitou vývojářů. Framework (během vývoje aplikace) nabízí tzv. *hot reload*, který umožňuje vývojářům aplikovat změny ve zdrojovém kódu instantně, bez nutnosti rekompilace celého projektu, čímž se značně urychluje vývoj.

4.2.2 React Native

React Native[2] je open-source projekt vyvíjen společností Facebook a umožňuje vývoj pro Android, Android TV, iOS, macOS, tvOS, Web, Windows a UWP. React Native (stejně jako Flutter) disponuje hot-reloadem. Jeho velkou výhodou je nepochybně fakt, že zdrojový kód React Native aplikací je psán v programovacím jazyce JavaScript, jehož komunita je obrovská (zhruba 12.4 milionu aktivních vývojářů ve třetím kvartálu roku 2020 [40]).

4.2.3 Xamarin

Xamarin[3] je framework vyvíjený společností Microsoft umožňující vývoj pro Android, iOS a Windows. Pro tvorbu aplikací využívá programovací jazyk C#. Na rozdíl od předchozích dvou zmíněných nedisponuje hot-reloadem, což značně zpomaluje rychlost vývoje. Zároveň ne vždy se vývojář může spolehnout na znalosti C# a občas je nucen přepsat části UI (uživatelské rozhraní) do nativního kódu cílené platformy (př: Java, Kotlin).

[1] https://flutter.dev/ navštíveno: 2.1.2020
[2] https://reactnative.dev/ navštíveno: 2.1.2020
[3] https://dotnet.microsoft.com/apps/xamarin navštíveno: 2.1.2020

4.2.4 Výběr frameworku

Výběr mezi frameworky jsem zúžil pouze na React Native a Flutter. Ačkoliv počet vývojářů používající Xamarin pomalu roste, jeho komunita pořád není tak veliká jako u předchozích dvou. Zdaleka největším odrazením od tohoto frameworku byla absence hot-reloadu a nutnost psaní nativního kódu pro danou platformu. Ve srovnání (tabulka: 4.2) ukazuji hlavní rozdíly mezi Flutterem a React Nativem.

	React Native	**Flutter**
Jazyk	JavaScript	Dart
Rok vzniku	2015	2018
Náročnost učení	Střední	Těžší
Hot-reload	Ano	Ano
Výkon	Střední	Rychlý
UI konzistence (více OS)	Potřeba knihoven 3. stran	Plně konzistentní

Tabulka 4.2: Srovnání React Native a Flutteru

Po delší úvaze jsem se uchýlil k možnosti psát aplikaci za pomocí **React Native** frameworku. Rozhodujícím faktorem byla má znalost JavaScriptu a doba kterou je framework dostupný (tzn. větší komunita vývojářů). Jak jsem již avizoval v kapitole **Srovnání přístupu** (sekce: 4.1.3), výkon aplikace není klíčovým parametrem. Toto rozhodnutí podpořil fakt, že samotné UI systému TermIt je psáno pomocí React frameworku, ze kterého React Native vychází.

4.3 React Native

Tato část blíže popisuje základní principy a architekturu React Native frameworku.

4.3.1 Obecné

React Native (jak název implikuje) vychází z JavaScriptové knihovny React, používanou pro tvorbu uživatelských rozhraní. Framework tedy staví na stejných základních principech jakými jsou:

1. Komponenty
2. Props – vstupní parametry komponent
3. State – vnitřní stav komponent
4. JSX

Základním prvkem jsou komponenty [12], které umožňují rozdělení jednotlivých UI prvků do nezávislých, oddělených a znovupoužitelných kusů kódu. Tyto dílčí kusy lze definovat jako JavaScriptové funkce nebo třídy.

Komponenty (stejně jako funkce) mohou mít definované vstupní parametry, které nadále zpracovávají. Tyto parametry jsou neměnné a slouží jako definice vlastností pro danou komponentu.

Komponenty si mohou rovněž udržovat svůj vnitřní stav (mít paměť). Na rozdíl od vstupních parametrů se jedná o data, která se mění za běhu aplikace a je nutné je aktualizovat.

Vnitřní stav komponent je izolován od ostatních komponent. Izolace spočívá v utajení vnitřních stavů napříč komponentami. Není tedy možné, aby jedna komponenta přímo četla či jinak manipulovala s vnitřním stavem komponenty druhé. Jediná možná přímá manipulace vnitřního stavu jiných komponent je přes jejich vstupní parametry. Ty mohou být uloženy do vnitřního stavu komponenty, která je obdržela. Informace o uložení ale stále zůstává jiným komponentám utajena a izolace je tedy zachována. Nastavování vstupních parametrů je možné pouze směrem „shora dolů“ tzn. komponenta, která nastavuje vstupní parametry komponentě jiné, se musí nacházet v hierarchické struktuře výše než komponenta, která vstupní parametry přijímá.

JavaScriptový kód je možné obohatit o JSX [13] (neboli JavaScript XML). Jedná se o rozšíření umožňující psát syntax značkovacího jazyka uvnitř

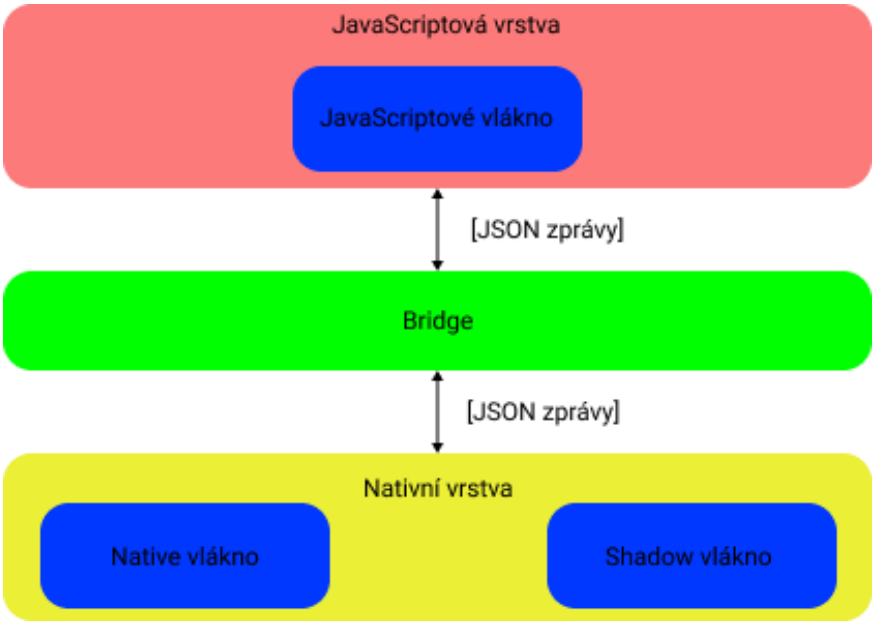
JavaScriptu [21]. Programátor tedy nemusí volat funkci na tvorbu komponent, nýbrž napíše pouze název komponenty uvnitř značky. To ve svém důsledku zvyšuje čitelnost kódu, jelikož UI komponenty jsou jasně rozlišitelné od zbytku JavaScriptového kódu.

4.3.2 Architektura

React Native pro svoji funkcionalitu využívá tři vlákna.

1. Native vlákno – Vlákno, na kterém je aplikace spuštěna. Hlavními cíli vlákna jsou vykreslování obrazu a schopnost reagovat na vstupy od uživatele.
2. JavaScriptové vlákno – Vlákno vykonávající JavaScriptový kód.
3. Shadow vlákno – Vlákno starající se o výpočet vzhledu (rozložení) aplikace.

Rozložení do tří vláken je výhodné, jelikož nedochází k blokaci hlavního vlákna, se kterým uživatel interaguje, během vykonávání postranních výpočtů.



Obrázek 4.1: Architektura React Native

Pro zajištění komunikace mezi JavaScriptovou vrstvou (kód v jazyce JavaScript) a nativní vrstvou (kód v jazyce Java/Objective C) využíváme tzv. bridge. Hlavním úkolem bridge je přeposílat serializované JSON zprávy mezi

vrstvami. Samotné zprávy jsou uloženy ve frontě. Při každém tiku jsou odebrány položky ze začátku fronty a odeslány ke svému zpracování. Jedná se o asynchronní způsob komunikace.

Tento způsob řešení má ale i své nedostatky. [3] Absence sdílené paměti napříč vrstvami je jedním z nich. Veškeré zdroje, které jsou potřeba v obou vrstvách, se musí serializovat do JSON zpráv a ty následně odeslat přes bridge. Dalším úskalím je samotná asynchronnost zpráv. V současné chvíli není možné aktualizovat uživatelské rozhraní z JavaScriptového vlákna synchronně. Při zahlcení bridge velkým množstvím zpráv, může docházet k nekonzistentnímu vzhledu uživatelského rozhraní. Příkladem může být situace, kdy uživatel rychle prochází velký seznam položek. Informace o těchto položkách může být pozdržena ve frontě bridge, což způsobuje viditelné načítání položek (seznam je chvíli prázdný, než se položky objeví). Výše zmíněné problémy budou adresovány v nové React Native architektuře (pracovní název Fabric [4]), jejíž konkrétní datum vydání ještě není známo.

React Native komponenty jsou pouze mapovány na nativní pohledy operačního systému, na kterém je aplikace spuštěna. Pohled je základním prvkem ve vývoji nativních aplikací pro Android či iOS. Nejedná se tedy o kompilaci JavaScriptových komponent do nativního kódu, pouze o jejich vyvolávání za pomocí zpráv ve formátu JSON, odeslaných přes bridge.

Aby byly komponenty vykresleny správně, využívá nativní část Shadow vlákno. [31] Hlavním úkolem tohoto vlákna je vytvořit hierarchickou strukturu prvků. Na vlákně je rovněž spuštěn Yoga Engine[4], starající se o správnou reprezentaci rozložení prvků na obrazovce. Konkrétně vypočítává pozici a velikost prvků definovaných pomocí CSS flexbox layoutu [39]. Po kalkulaci oznámí nativnímu vláknu, že může aplikaci (jednotlivé prvky) zobrazit.

[4] https://yogalayout.com/docs navštíveno: 29.4.2020

4.3.3 Založení projektu

Pro založení nového projektu v React Native frameworku se nabízí dvě možnosti.

1. Managed workflow - Expo
   - Vývojář má omezenou kontrolu nad projektem a pouze píše JavaScriptový zdrojový kód.
2. Bare workflow
   - Vývojář má plnou kontrolu nad projektem i se všemi jeho komplexnostmi.

Níže (tabulka: 4.3) jsou oba přístupy porovnány.

	Expo	**Bare**
Potřeba zkušeností s mobilním vývojem	Ne	Ano
Správa knihoven	Frameworkem	Programátorem
Správa buildů	Frameworkem	Programátorem (vývojové prostředí)
Velikost aplikace na disku	Větší (nelze optimalizovat)	Menší (lze optimalizovat)

Tabulka 4.3: Srovnání managed workflow pomocí Expo a Bare režimu

Hlavním rozhodujícím faktorem pro mě byla zkušenost s tvorbou mobilních aplikací. Založení projektu v Bare režimu je doporučována spíše pro zkušenější vývojáře mobilních aplikací. Tento režim sice nabízí větší svobodu vývoje, ale vyžaduje vlastní správu knihoven. Možnou nevýhodou Bare režimu, pro nového vývojáře mobilních aplikací, může být i vlastní správa buildů. Expo řeší buildy aplikací samo, tudíž není nutné používat robustní vývojová prostředí (např. Android Studio a Xcode) pro kompilaci a spouštění aplikací. Zároveň nabízí i mobilního klienta, který aplikaci dovoluje otestovat na fyzickém zařízení prakticky okamžitě. Všechny výše uvedené důvody vedly k rozhodnutí založit projekt za pomocí **Expo** frameworku.

4.4 Transpilace

Pro zachování zpětné kompatibility verzí JavaScriptu je nutné kód transpilovat do starších verzí JavaScriptu, které umí spouštět starší JavaScriptové enginy. Transpilace je proces, kde transpilátor (překladač) na vstupu obdrží zdrojový kód, který přepíše do zdrojového kódu jazyka jiného. Překlady se nejčastěji týkají standardu ECMAScript 2015+ [37] a kódu obohaceného o JSX.

Možným řešením tohoto překladu je transpilátor Babel[5]. Ukázkovým předmětem překladu může být následující kód (obrázek: 4.2).

```
1         function greet(input) {
2             return input ?? "Hello world";
3           }
```

Obrázek 4.2: Kód psaný ve standardu ES2020

Po překladu je kód změněn následujícím způsobem (obrázek: 4.3).

```
1         function greet(input) {
2             return input != null ? input : "Hello world";
3           }
```

Obrázek 4.3: Kód psaný ve standardu ES5

Nutnost transpilovat kód v projektu je očividná, vzhledem k používání JSX (sekce: 4.3.1) a novějších standardů JavaScriptu ve zdrojovém kódu aplikace.

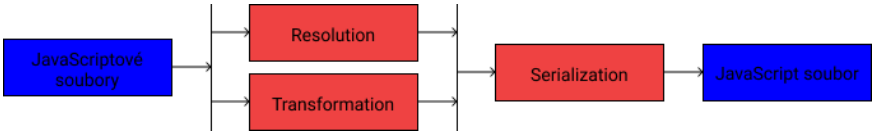
4.5 JavaScript bundler

JavaScriptový bundler [17] umožňuje shromáždit vícero JavaScriptový souborů do jednoho souboru. Shromáždíme-li veškerý kód na jedno místo, nemusíme se obávat špatných referencí na jiné JavaScriptové soubory. React Native projekty používají JavaScriptový bundler Metro[6]. Ten svojí činnost dělí do tří fází:

1. Resolution
    - Vytvoří graf modulů, které aplikace potřebuje. Probíhá paralelně s transformační fází.

[5] https://github.com/babel/babel navštíveno: 5.4.2020
[6] https://facebook.github.io/metro/ navštíveno: 5.4.2020

2. Transformation
   - Všechny moduly, které aplikace potřebuje projdou transformací. Jedná se o transpilaci kódu do takové podoby, kterou budou schopné cílové platformy spustit. Překlad probíhá za pomoci transpilátoru Babel (viz kapitola transpilace).
3. Serialization
   - Po překladu všech modulů dojde k jejich serializaci. Serializátor vezme všechny přeložené moduly a ty shromáždí do jednoho JavaScriptového souboru.



Obrázek 4.4: Operace Metro bundleru

Metro [36] nabízí vývojářům hlavně rychlé znovu-načítací časy, rychlé spouštěcí časy, kompatibilitu/shromáždění kódu a konverzi zdrojů (př. obrázků) do podoby, kterou umí nativní komponenty zobrazit (př. Image komponenta).

4.6 Node moduly

Node moduly jsou psané v programovacím jazyce JavaScript. Může se jednat o jeden či celou řadu JavaScriptových souborů, které je možné spouštět mimo prostředí webového prohlížeče. Moduly poskytují funkcionalitu, kterou lze použít opakovaně napříč aplikací. Každý modul má svůj vlastní kontext a je tedy nemožné, aby jakkoliv narušoval chod ostatních modulů či jinak ovlivňoval global scope (proměnné deklarovoné mimo funkci). O popularitě node modulů svědčí i jejich veřejně dostupné množství. NPM[7] má již ve svém registru přes 1.5 milionů modulů. Konkrétní použité node moduly jsou popsány v následující kapitole.

[7] Node package manager: https://www.npmjs.com/ navštíveno: 1.5.2020

Kapitola 5

Implementace

Tato kapitola se zabývá implementační částí projektu. Ten je naprogramován v programovacím jazyce TypeScript (rozšíření JavaScriptu, rovněž podléhá transpilaci). Popisuje klíčové node moduly použité v aplikaci, dále popisuje datový model a strukturu projektu.

5.1 Globální stav

Jak již bylo avizováno v kapitole **React Native Obecné** (sekce: 4.3.1), interní stavy komponent jsou navzájem utajeny. To sebou nese značná omezení během vývoje. Ačkoliv je možné předávat veškerá data za pomocí vstupních parametrů komponent, v hluboké struktuře se tento přístup stává značně nepřehledným. Zároveň není možné, aby komponenta ovlivnila vnitřní stav komponenty, která je hierarchicky výše.

Tento problém je řešen za pomocí implementace globálního/aplikačního stavu. Globální stav představuje společné úložiště pro všechny komponenty. Tím mizí restrikce na směr komunikace „shora dolů“ a zároveň nadbytečné předávání vstupních parametrů napříč komponentami.

Hlavním důvodem použití globálního stavu v mobilní aplikaci je sdružení dat o přihlášeném uživateli a jeho právech. Ta jsou zapotřebí pro určení, k jakým částem aplikace má uživatel přístup. Rovněž jsou udržovány informace o vnitřním stavu aplikace, který vymezuje, jaký obsah bude uživateli zobrazen. Příkladem může být stav, ve kterém již spuštěná aplikace čeká na potřebná data pro své další použití (tzn. je ve stavu „načítání“). Tento stav je specifický, jelikož aplikace musí zobrazit obrazovku tomuto stavu předem určenou.

5.1.1 Redux

V současné době patří Redux[1] mezi nejrozšířenější knihovnu pro řešení problému globálního stavu v JavaScriptových aplikacích. Na Redux se dá pohlížet jako na knihovnu a zároveň jako na návrhový vzor, řešící správu centralizované paměti v aplikaci za pomoci předem definovaných akcí.

Pro vysvětlení funkcionality musíme zavést pár základních termínů, se kterými Redux pracuje.

- Store
  - Jedná se o místo, kde je uložen globální stav aplikace. Hodnoty, které zde ukládáme jsou POJOs[2], pole a primitiva. Abychom mohli store vytvořit, musíme při jeho zakládání specifikovat, jaké reducers bude používat.
- Reducer
  - Jedná se o funkci, jejíž parametry jsou stav (state) a akce (action). Funkce se na základě přijaté akce rozhodne, jak má upravit stav, který posléze z funkce vrátí.

    Je nutné poznamenat fakt, že stav je neměnný, nelze ho tedy přímo upravovat. Změna probíhá výpočtem nového stavu ze stavu předchozího. Nově vytvořený objekt je z funkce vrácen jako návratová hodnota.

    Pro reducer platí určitá pravidla, která běžně asociujeme s „pure functions“. To jsou funkce, které splňují základní dvě vlastnosti:

    1. Pro stejný vstup mají vždy stejný výstup
    2. Nedochází k „side effects“

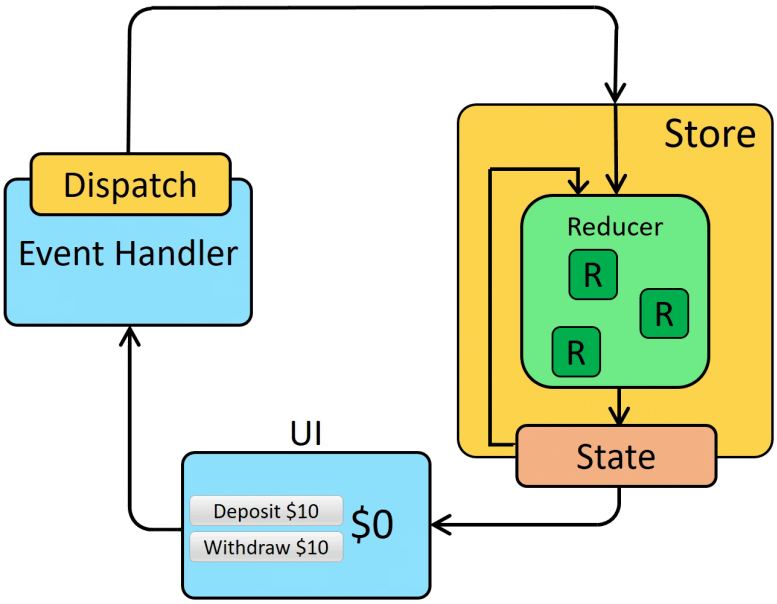
    Mezi typické „side effects“ řadíme například:

    1. Úprava vstupních proměnných funkce
    2. Úprava proměnných mimo rámec funkce
    3. Logování do konsole
    4. Provádění asynchronních operací (př. AJAX HTTP požadavky)
    5. Provádění vstupně-výstupních operací
    6. Generování náhodných čísel

[1] https://github.com/reduxjs/redux navštíveno: 6.4.2020
[2] plain old java objects

- Actions
  - Akce popisující, co se v aplikaci stalo. Tento popis slouží k modifikaci globálního stavu. Jedná se o objekt, obsahující název akce (type) a případná data k akci přidružená (payload).
- Dispatch
  - Aby mohlo dojít ke změně stavu, musíme store říct, že nastala akce. Pro tento účel má store metodu dispatch. Ta jako svůj parametr přijímá objekt akce, kterou chceme vyvolat. Funkce zavolá reducer a uloží nově vypočítaný stav vrácený reducerem.

Diagram (obrázek: 5.1) znázorňuje tok data napříč Reduxem. Jedná se o typický příklad, kde uživatelské rozhrání zobrazuje aktuální hodnotu globálního stavu, nastane-li událost, event handler ji zachytí a zavolá metodu dispatch s příslušnou akcí, ta je následně zpracována reducerem, jehož výstup je uložen ve store.

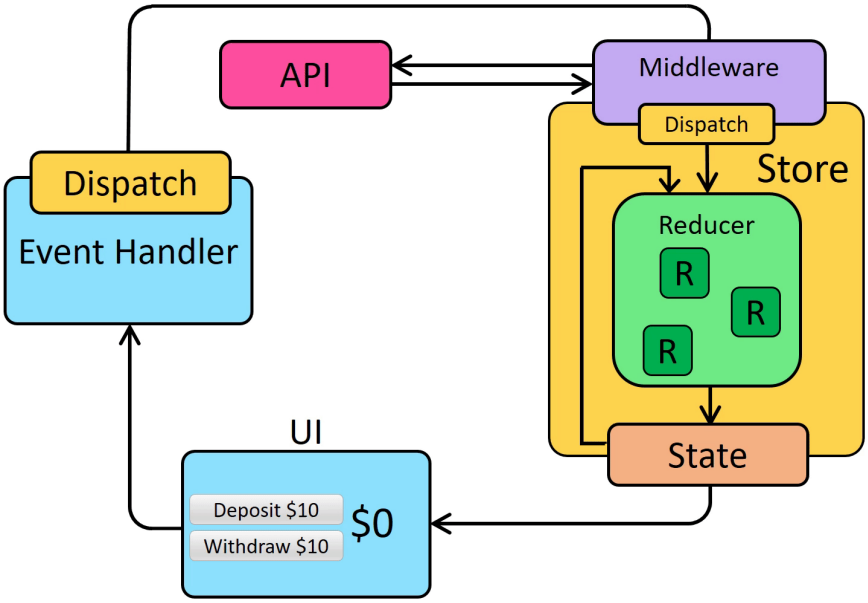


Obrázek 5.1: Redux architektura [1]

Jelikož se reducers musí řídit výše zmíněnými pravidly „pure functions“, musíme zavést asynchronní logiku na jiném místě. Pro tyto účely využíváme middleware, který nám umožňuje měnit stav i za použití asynchronních operací.

Redux nabízí svojí vlastní implementaci toho middlewaru nazývanou Redux Thunk Middleware[3]. Ten umožňuje poslat metodě dispatch parametr, který není akcí ale asynchronní funkcí. Tato funkce se spustí a ve svém průběhu může zavolat metodu dispatch znovu. Volání ale nyní opatří objektem akce jako jejím parametrem. To je následně zpracované reducerem, jehož výstup je rovněž uložen do store.

Následující diagram (obrázek: 5.2) je obohaceným předchozím diagramem datového toku napříč Reduxem o asynchronní logiku ve formě volání vzdáleného serveru (HTTP AJAX požadavek). Tok událostí je podobný jako bez použití middlewaru. Vyvolaná událost na uživatelském rozhraní je zachycena event handlerem, který zavolá dispatch metodu s asynchronní funkcí jako jejím parametrem. Funkce je zpracovaná middlewarem, který následně vytvoří nové volání metody dispatch, nyní již s akcí jako parametrem. Tu reducer rozpozná a vypočítá nový stav, který je následně uložen.



Obrázek 5.2: Redux architektura s middlewarem [2]

[3]https://github.com/reduxjs/redux-thunk navštíveno: 6.4.2020

5.2 Navigace v aplikaci

Problém navigace v aplikaci je řešen za pomocí react-navigation/native node modulu[4]. Tento modul je doporučován samotnými vývojáři React Nativu v jeho oficiální dokumentaci [14]. V současné chvíli se jedná o nejstahovanější navigační modul pro React Native (dle React Native Directory[5]).

Základní stavební bloky navigace tvoří samotné obrazovky a tzv. navigátory, do kterých dané obrazovky patří. K dispozici modul nabízí tři druhy navigátorů:

1. Tab Navigator
   - Navigace pomocí lišty
2. Stack Navigator
   - Navigace pomocí odkazů na obrazovkách
3. Drawer Navigator
   - Navigace pomocí vyjíždějící postranní nabídky

V aplikaci je použit navigátor typu Stack, jehož historie obrazovek je reprezentována pomocí zásobníkové datové struktury. Tato reprezentace je vhodná, jelikož zachovává časový tok událostí, které uživatel v aplikaci vyvolal. Rovněž je v aplikaci přítomen navigátor typu Drawer, který umožňuje rychlý přístup k často používaným obrazovkám.

Pro použití více navigátorů současně, je nutné do sebe navigátory vzájemně vnořovat. Nicméně vývojáři modulu nedoporučují navigátory vnořovat excesivně [32]. Hluboké struktury mohou totiž způsobovat výkonnostní potíže na slabších mobilních zařízeních.

[4] https://github.com/react-navigation/react-navigation navštíveno: 1.5.2020
[5] https://reactnative.directory/ navštíveno: 18.5.2020

Struktura navigátorů a jejich příslušných obrazovek v aplikaci je ukázána níže.

- Přihlašovací obrazovka
- Domů (Stack Navigator)
  - Domů
  - Drawer (Drawer Navigator)
    - Vyhledávání (Stack Navigator)
      - Vyhledávání
      - Detail pojmu
      - Detail slovníku
      - Úprava pojmu
      - Úprava slovníku
      - Nový pojem
    - Oblíbené
    - Slovníky (Stack Navigator)
      - Slovníky
      - Nový slovník

5.3 Komunikace s API

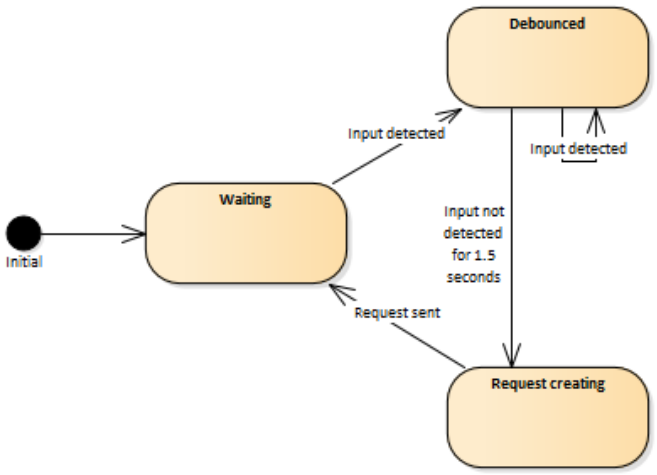
Komunikace s REST API systému TermIt je zajištěná node modulem SWR[6]. Jedná se o React Hook[7] knihovnu, která umožňuje jednoduché vytváření API volání. Zároveň nabízí načítání dat z cache paměti, což značně urychlí odezvu aplikace. Načítání dat z cache je řízeno strategií HTTP RFC 5861 [27]. Proběhne tedy kontrola HTTP hlavičky odpovědi serveru (Cache-Control: max-age, stale-while-revalidate), na základě které se rozhodne, zdali jsou data pořád platná či nikoliv. Sekvenční diagram (příloha: C.1) ukazuje chování SWR knihovny, konkrétně v případě vyhledávání pojmů a slovníků.

5.4 Optimalizace vyhledávání

Pro naplnění očekávaného chování vyhledávací obrazovky bylo nezbytné implementovat tzv. debounce funkci. O její implementaci se postará stejnojmenný modul[8]. Její použití slouží k detekování situace, kdy uživatel přestal psát do

[6] https://github.com/vercel/swr navštíveno: 3.5.2020
[7] https://reactjs.org/docs/hooks-intro.html navštíveno: 3.5.2020
[8] https://github.com/component/debounce navštíveno: 3.5.2020

vyhledávacího okna aplikace. Pouze tehdy se má kontaktovat API systému TermIt. Bez této detekce by aplikace vytvářela nespočet API volání (při každé změně textu ve vyhledávacím okně) a obsah aplikace by se co chvíli měnil. Následující stavový diagram (obrázek: 5.3) ukazuje její konkrétní implementaci v projektu.



Obrázek 5.3: Stavový diagram vyhledávacího okna

5.5 Ukládání dat

V aplikaci je zapotřebí některá data uložit takovým způsobem, aby se neztratila s jejím vypnutím. Pro tyto účely je v aplikaci nasazen modul react-native-async-storage/async-storage[9]. Jedná se o oblíbený komunitní balíček, čítající okolo 500 tisíc stažení měsíčně (uvádí React Native Directory[10]). Právě na komunitní balíčky se vývojáři React Native odkazují v oficiální dokumentaci, protože řešení, které je v frameworku implicitně zahrnuto, již není v současné době aktivně spravováno.

Data se ukládají jako textový řetězec s vnitřní strukturou dat ve formátu JSON, tedy klíč-hodnota. V aplikaci jsou tímto způsobem ukládány následující data:

1. Autentizační token (Bearer token [24])
2. Oblíbené položky
3. Historie vyhledávání

[9] https://github.com/react-native-async-storage/async-storage navštíveno: 18.5.2020
[10] https://reactnative.directory/ navštíveno: 18.5.2020

Autentizační token se v aplikaci ukládá, aby po uživateli nebyla zbytečně vyžadovaná autentizace při každém zapnutí mobilní aplikace. Ta se při svém zapnutí nejdříve pokusí spojit se serverem za použití již uloženého autentizačního tokenu. Pokud server tento token vyhodnotí jako neplatný, je uživatel přesměrován na přihlašovací obrazovku. V opačném případě je uživatel rovnou vpuštěn do aplikace. Důvody pro ukládání oblíbených položek a historie vyhledávání není nutné vysvětlovat.

5.6 Struktura

Projekt je pro větší přehlednost rozložen do několika adresářů (tabulka: 5.1).

Adresář	**Obsah**
___mocks___	Mocky pro testování
actions	Action creators pro Redux
assets	Doplňkový obsah (multimédia)
components	React komponenty
models	Datové modely
node_modules	Node moduly (př. Redux, react-navigation)
reducers	Reducers pro Redux
screens	Obrazovky
store	Definice typů pro jednotlivé reducers
utils	Nápomocné funkce, typy a konstanty

Tabulka 5.1: Popis adresářů a jejich obsahu

Kapitola 6

Testování

Tato kapitola popisuje zvolené způsoby testování aplikace a technologie k tomu použité.

6.1 Použité nástroje

Automatizované testování v aplikaci probíhá za pomoci Jest[1] frameworku . Jedná se o JavaScriptový framework, který nevyžaduje složitou konfiguraci a cílí na jednoduchost a rychlost svého používání. Framework svým uživatelům umožňuje psaní unit testů, integračních testů, mocků (více v sekci: 6.2) a generování statistik týkající se code coverage (množství kódu pokrytého testy).

[1] https://github.com/facebook/jest navštíveno 26.12.2020

Pro potřeby testování mobilní aplikace bylo nutné do projektu dodat ještě další dvě knihovny:

- react-test-renderer[2]
  - Umožňuje vykreslenou React komponentu převést do JavaScriptového objektu, který je nezávislý na DOM (Document Object Model) nebo na mobilním prostředí.
- redux-mock-store[3]
  - Umožňuje namockovat Redux store.

Vzájemná spolupráce výše zmíněných knihoven umožňuje otestovat uživatelské rozhraní a operace spojené s životním cyklem aplikace.

6.2 Unit tests

Jedná se o automatizované testování, které ověřuje funkčnost a korektnost implementace částí systému [20]. Rozsah testované části je malý a samotné testované segmenty by měli být na sobě nezávislé. Abychom dosáhli izolace testovaných části, používáme tzv. mocks které simulují chování jiných částí systému, na kterých může být námi testovaný kód závislý. Tím zajistíme funkčnost jednotlivých částí, ze kterých se systém skládá.

Unit testy v mobilní aplikaci testují následující části zdrojového kódu.

- Action creators
  - Kontrola, zdali vytvářené Redux akce mají správný typ a adekvátní payload.
- Reducers
- Pomocné funkce

[2] https://reactjs.org/docs/test-renderer.html navštíveno: 10.5.2020
[3] https://github.com/reduxjs/redux-mock-store navštíveno: 10.5.2020

- Uživatelské rozhraní
  - Test probíhá vykreslením komponenty a jejího následného exportu do JSON formátu. Pokud je test spuštěn prvně, vytvoří se snímek komponenty (reprezentativní vzorek), který je porovnáván při dalším spuštění testu. Pokud se nově vytvořený snímek komponenty neshoduje s reprezentativním vzorkem, došlo ke změně uživatelského rozhraní, kterou je potřeba ručně ověřit.

Jako názornou ukázku použití Jest frameworku přikládám kód, který testuje správné vytváření Redux akcí.

```
1        describe('search actions', () => {
2            it('should create an action to SEARCH', () => {
3                const searchQuery='searched text';
4
5                const expectedAction= {
6                    type: SEARCH,
7                    payload: searchQuery
8                }
9
10               expect(search(searchQuery)).toEqual(
                     expectedAction);
11           })
12       })
```

Obrázek 6.1: Ukázka testu používající Jest framework

6.3 Uživatelské testy

Uživatelské testování [35] se provádí ke konci vývojového cyklu softwaru. Jedná se o tzv. *přejímací zkoušku*, kde se ověřuje, zdali byly splněny všechny požadavky zadavatele. V této fázi testování přistoupí k softwaru testeři, kterým jsou přiděleny scénáře (situace vycházející z funkčních požadavků), podle kterých software testují. Cílem tohoto testování je ověření, že produkt splňuje vytyčené funkční a nefunkční požadavky.

Uživatelské testování se ukázalo jako velice přínosným zdrojem informací. Nejen že odhalilo několik drobných chyb, které bude nutné opravit, ale poskytlo i cenný náhled uživatelů na samotné chování aplikace. Celkem se testování zúčastnili tři testeři. Každý tester vykonal 15 testovacích scénářů (příloha: D). U každého testu jsem byl přítomen a pozoroval jsem, jak uživatelé s aplikací interagují. Po samotném testu aplikace byly otestovány ty samé funkcionality na mobilní webové verzi systému TermIt. Cílem bylo zjistit, zdali není toto řešení dostačující a nově vzniklá aplikace tedy nadbytečnou (více v kapitole: 7 **Srovnání**).

Níže (tabulka: 6.1) uvádím problémy či nedostatky, na které uživatelé během testování narazili a popisuji jejich možné řešení.

Problém/Nedostatek	**Možné řešení**
Je-li server během přihlášování nedostupný, uživatel na to není upozorněn.	Kontrolovat http stavové kódy odpovědi serveru. Zobrazovat chybovou zprávu je-li server nedostupný.
Chybí vyhledávání na výpisu dostupných slovníků.	Přidat textové pole pomocí nějž bude možné filtrovat seznam dostupných slovníků.
Chybí potvrzení akcí při vytváření a ukládání pojmů a slovníků.	Přidat dialogové okno, které se zobrazí po stisknutí tlačítka „uložit“.
Kliknutím na zdroj pojmu, který je URL adresou, nedojde k otevření webového prohlížeče na dané URL adrese.	Přidat ověření, jestli je zdroj platnou URL adresou, pokud ano, automaticky otevřít výchozí prohlížeč na dané URL adrese.
Chybějící detailní popis vyplňovaných atributů při tvorbě/úpravě pojmů. Uživatel by ocenil bližší vysvětlení vyplňovaného atributu.	Přidat našeptávač pro jednotlivé atributy.
Při kliknutí na název obrazovky vedle ikony postranního menu, uživatel očekává vyjetí postranní nabídky.	Přidat naslouchač událostí k zobrazenému názvu obrazovky. Při zaregistrování události, ukázat postranní nabídku aplikace.
Má-li uživatel odsazenou klávesnici z důvodů ergonomie, aplikace detekuje stisky tlačítek pod klávesnicí.	Problém se týká obrazovek starající se o vytváření a úpravu pojmů či slovníků. Znepřístupnit tlačítko uložení je-li klávesnice aktivní.
Při kliknutí na prvek obrazovky, během aktivovaného vstupního textového pole, nedojde k prokliknutí tlačítka.	Opravit předávání událostí na obrazovce.

Tabulka 6.1: Testery odhalené chyby a nedostatky

Kapitola 7

Srovnání

Tato kapitola srovnává nově vzniklou mobilní aplikaci s aktuálně dostupným řešením.

7.1 Mobilní webová verze

Hlavním cílem mobilní aplikace bylo vytvoření uživatelsky přívětivé varianty systému TermIt na mobilních zařízeních. Jejím jediným možným konkurentem je současná implementace mobilní webové verze systému TermIt. Srovnání těchto dvou řešení probíhalo bezprostředně po vykonání uživatelských testů mobilní aplikace. Testerům byly předloženy stejné úkoly, jaké vykonávaly v mobilní aplikaci. Srovnání probíhalo formou dotazníku, jehož výsledky jsou vypsány dále v textu (tabulka 7.1). Testeři byli dotázáni, aby ke každému úkolu přiřadili preferovanou variantu implementace. Pokud neupřednostňovali ani jednu z nich, volili možnost *Srovnatelný zážitek*. Čísla v tabulce tedy reprezentují počet testerů, kteří volili danou implementaci.

Popis úkolu	**Mobilní aplikace**	**Mobilní webová verze**	**Srovnatelný zážitek**
Vyhledávání pojmů a slovníků	3	0	0
Vytváření a úprava pojmů	2	0	1
Vytváření a úprava slovníků	1	0	2
Detailní výpis pojmu	3	0	0
Detailní výpis slovníku	3	0	0
Výpis všech slovníků spravovaných serverem	2	1	0
Psaní komentářů	3	0	0
Mazání pojmů a slovníků	0	0	3
Hlasy celkem	**17**	**1**	**6**

Tabulka 7.1: Srovnání verzí

Z tabulky je patrné, že skoro ani jeden tester by si pro jakoukoliv funkcionalitu poskytovanou mobilní aplikací, raději zvolil mobilní webovou verzi než mobilní aplikaci. Jedinou výjimkou je výpis všech slovníků spravovaných serverem. Tester raději volil mobilní webovou verzi z důvodu možné filtrace názvů slovníků. Tuto funkci plánuji v budoucnu naimplementovat.

Největším problémem pro testery na mobilní webové verzi bylo vyhledávání, které se zasekávalo či neposkytovalo výsledky vůbec. Dalším úskalím byly detailní výpisy pojmů a slovníků. Testeři označili výpis za nepřehledný, působící nenačteným dojmem. Výpis v mobilní aplikací popisují jako srozumitelný a rychle přístupný (přehledný). Pro psaní komentářů rovněž všichni dotázaní volili cestu mobilní aplikace. Důvodem byla možnost psát komentář okamžitě, bez nutnosti přepnutí kontextového menu.

Testeři se jednohlasně shodli na tom, že mobilní aplikace je jím příjemnější na používání a upřednostnili by ji před mobilní webovou verzí. Rovněž ocenili funkcionality navíc, kterou mobilní aplikace nabízí.

Ukázky uživatelského rozhrání mobilní aplikace a její následné srovnání s mobilní webovou verzí naleznete v příloze (příloha: E).

Kapitola 8

Závěr

Motivací této bakalářské práce byla analýza technologií pracujících na sémantickém webu, popis funkcionality systému TermIt, vytvoření softwarové analýzy nové mobilní aplikace, její následná implementace, otestování aplikace a srovnání s mobilní webovou verzí. Mobilní aplikace má poskytovat uživatelsky přívětivý přístup do systému TermIt na mobilních zařízeních s operačním systémem Android.

8.1 Zhodnocení

S výsledkem bakalářské práce jsem spokojený. Nově vzniklá mobilní aplikace poskytuje základní funkcionalitu, kterou by uživatel systému TermIt očekával na mobilním zařízení. To znamená rychlé vyhledávání pojmů a slovníků, zobrazovaní jejich detailních popisů, úpravu pojmů a slovníků a komentování pojmů.

Aplikace zároveň poskytuje funkcionalitu, kterou webová implementace nenabízí. Mezi nové funkcionality řadíme možnost změny koncové adresy serveru přímo v aplikaci, ukládání pojmů a slovníků do seznamu oblíbených položek a evidence historie vyhledávání. To vše v uživatelsky přívětivém, jednoduše pochopitelném prostředí mobilní aplikace.

O úspěšnosti výsledku svědčí uživatelské srovnání mobilní aplikace s mobilní webovou verzí, které dopadlo ve prospěch mobilní aplikace.

8.2 Budoucí práce

Ačkoliv byly splněny všechny předem vytyčené cíle, stále je zde prostor pro vylepšení ve formě optimalizace výkonu aplikace a oprav nově objevených chyb. Rovněž je možné mobilní aplikaci zprovoznit na zařízeních s operačními systémy iOS. V neposlední řadě lze aplikaci nadále rozšiřovat přidáváním dalších funkcionalit systému TermIt.

8.2.1 Oprava chyb

Během uživatelského testování bylo odhaleno několik chyb, které je nutné opravit. Nejedná se o chyby, které by znepřístupňovaly funkcionalitu aplikace, ale o chyby, týkající se uživatelského zážitku. Jedná se o připomínky drobnějšího charakteru, tudíž i jejich samotná oprava by neměla zabrat velké množství času.

8.2.2 Optimalizace

Aplikace trpí propady snímkovací frekvence, při procházení velkého množství pojmů za pomocí navigačních tlačítek (př. podřazený pojem) v jejich detailních výpisech (aplikace si pamatuje všechny uživatelovy předchozí kroky). Mezi možné řešení patří odstranění cyklických vazeb či nastavení limitu maximálně možných kroků zpět.

8.2.3 Operační systém iOS

Aby bylo možné aplikaci spustit na platformě iOS, je zapotřebí prověřit, zdali se ve zdrojovém kódu nevyskytují neošetřené Android specifické volání komponent. React Native totiž nabízí i komponenty, které mají nativní implementaci pro Android, ale nikoliv pro iOS. Stejně tak nabízí i iOS proprietární komponenty. Aby byl přechod na iOS úspěšný, je rovněž zapotřebí aplikaci důkladně otestovat právě na tomto operačním systému.

8.2.4 Další funkcionality

Samotný systém TermIt je stále ještě vývoji a je tedy velice pravděpodobné, že bude obohacen o nové funkcionality, které bude vhodné rovněž přenést do mobilní aplikace. Zároveň může být mobilní aplikace inspirací pro nové požadavky na systém TermIt. Nabízí se využití fotoaparátů na mobilních zařízeních. Fotografie z nich by mohly sloužit jako doplňující materiál k definicím pojmů.

Literatura

[1] D. Abramov and the Redux documentation authors. Redux fundamentals, part 2: Concepts and data flow. https://redux.js.org/tutorials/fundamentals/part-2-concepts-data-flow. [Navštíveno: 10.5.2021].

[2] D. Abramov and the Redux documentation authors. Redux fundamentals, part 6: Async logic and data fetching. https://redux.js.org/tutorials/fundamentals/part-6-async-logic#redux-fundamentals-part-6-async-logic-and-data-fetching. [Navštíveno: 10.5.2021].

[3] S. Alpert. State of react native 2018. https://reactnative.dev/blog/2018/06/14/state-of-react-native-2018. [Navštíveno: 29.4.2021].

[4] axemclion. React native's new architecture - glossary of terms. http://blog.nparashuram.com/2019/01/react-natives-new-architecture-glossary.html. [Navštíveno: 29.4.2021].

[5] T. Berners-Lee. Linked data. https://www.w3.org/DesignIssues/LinkedData.html, 2009. [Navštíveno: 10.5.2021].

[6] K. Bittner, I. Spence, and I. Jacobson. *Use Case Modeling*. The Addison-Wesley object technology series. Addison Wesley, 2003.

[7] D. Brickley and A. Miles. SKOS core guide. https://www.w3.org/TR/2005/WD-swbp-skos-core-guide-20051102/. [Navštíveno: 10.5.2021].

[8] G. Carothers and E. Prud'hommeaux. RDF 1.1 turtle. https://www.w3.org/TR/2014/REC-turtle-20140225/. [Navštíveno: 10.5.2021].

[9] O. Chursin. A brief introduction to domain modeling. https://olegchursin.medium.com/a-brief-introduction-to-domain-modeling-862a30b38353. [Navštíveno: 06.5.2021].

[10] W. W. W. Consortium. 3.1.3 Relationship to the World Wide Web and REST Architectures. https://www.w3.org/TR/2004/NOTE-ws-arch-20040211/#relwwwrest. [Navštíveno: 30.12.2020].

[11] R. Cyganiak, D. Wood, and M. Lanthaler. RDF 1.1 concepts and abstract syntax. W3C recommendation, W3C, Feb. 2014. https://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/ [Navštíveno: 30.12.2020].

[12] Facebook Inc. Components and Props. https://reactjs.org/docs/components-and-props.html. [Navštíveno: 29.4.2021].

[13] Facebook Inc. Introducing JSX. https://reactjs.org/docs/introducing-jsx.html. [Navštíveno: 29.4.2021].

[14] Facebook Inc. React native. https://reactnative.dev/docs/navigation. [Navštíveno: 30.12.2020].

[15] I. O. for Standardization. Information technology — Database languages — SQL — Part 2: Foundation (SQL/Foundation). Standard, International Organization for Standardization, Geneva, CH, Dec. 2016.

[16] I. O. for Standardization. Information technology — The JSON data interchange syntax. Standard, International Organization for Standardization, Geneva, CH, Nov. 2017.

[17] A. Gimeno. How javascript bundlers work. https://medium.com/@gimenete/how-javascript-bundlers-work-1fc0d0caf2da. [Navštíveno: 5.5.2021].

[18] M. Grünhagen. *The Semantic Web*. Springer Nature, 2007.

[19] M. Heller. Rest and crud: the impedance mismatch. https://www.infoworld.com/article/2640739/rest-and-crud--the-impedance-mismatch.html. [Navštíveno: 6.1.2021].

[20] ISO. Software and systems engineering — software testing — part 1: Concepts and definitions. ISO 29119-1:2013, International Organization for Standardization, Geneva, Switzerland, 2013.

[21] James H. Coombs, Allen H. Renear, Steven J. DeRose. Markup systems and the future of scholarly text processing. http://xml.coverpages.org/coombs.html. [Navštíveno: 29.4.2021].

[22] Jean-Jacques Moreau and Marc Hadley and Martin Gudgin and Henrik Frystyk Nielsen and Noah Mendelsohn. SOAP Version 1.2 Part 1: Messaging Framework. W3C recommendation, W3C, June 2003. https://www.w3.org/TR/2003/REC-soap12-part1-20030624/ [Navštíveno: 28.12.2020].

[23] G. Kellogg, D. Longley, and P.-A. Champin. JSON-ld 1.1. https://www.w3.org/TR/2020/REC-json-ld11-20200716/. [Navštíveno: 10.5.2021].

[24] M. Jones and D. Hardt. The OAuth 2.0 Authorization Framework: Bearer Token Usage. http://www.rfc-editor.org/rfc/rfc6750.txt. [Navštíveno: 10.5.2021].

[25] Martin Ledvinka, Kremen Petr, Saeeda Lama and Blasko Miroslav. Termit: A practical semantic vocabulary manager. https://www.researchgate.net/publication/341541783_TermIt_A_Practical_Semantic_Vocabulary_Manager, 2020. [Navštíveno: 10.5.2021].

[26] A. Miles and S. Bechhofer. SKOS simple knowledge organization system reference. W3C recommendation, W3C, Aug. 2009. https://www.w3.org/TR/2009/REC-skos-reference-20090818/ [Navštíveno: 25.12.2020].

[27] M. Nottingham. Http cache-control extensions for stale content. https://tools.ietf.org/html/rfc5861. [Navštíveno: 9.1.2021].

[28] P. Patel-Schneider, P. Hayes, and I. Horrocks. OWL web ontology language semantics and abstract syntax. W3C recommendation, W3C, Feb. 2004. https://www.w3.org/TR/2004/REC-owl-semantics-20040210/ [Navštíveno: 30.12.2020].

[29] E. Prud'hommeaux and A. Seaborne. SPARQL query language for RDF. https://www.w3.org/TR/2008/REC-rdf-sparql-query-20080115/. [Navštíveno: 18.5.2021].

[30] Ramanathan Guha and Dan Brickley. RDF Schema 1.1. https://www.w3.org/TR/2014/REC-rdf-schema-20140225/. [Navštíveno: 5.1.2021].

[31] A. Rashevskaya. React native re-architecture — what to expect from the popular cross-platform framework? https://litslink.com/blog/new-react-native-architecture. [Navštíveno: 29.4.2021].

[32] React Navigation Community. Nesting navigators. https://reactnavigation.org/docs/nesting-navigators/. [Navštíveno: 29.4.2021].

[33] Roy Fielding, Tim Berners-Lee, Jean Paul Getty. Hypertext transfer protocol – http/1.1. https://tools.ietf.org/html/rfc2616. [Navštíveno: 9.1.2021].

[34] A. Seaborne and S. Harris. SPARQL 1.1 query language. W3C recommendation, W3C, Mar. 2013. https://www.w3.org/TR/2013/REC-sparql11-query-20130321/ [Navštíveno: 30.12.2020].

[35] M. Setter. User acceptance testing – how to do it right! https://usersnap.com/blog/user-acceptance-testing-right/. [Navštíveno: 10.5.2021].

[36] R. Sharma. Role of metro bundler in react native. https://rishabh0297.medium.com/role-of-metro-bundler-in-react-native-24d178c7117e. [Navštíveno: 5.5.2021].

[37] S. Stefanov. *JavaScript Patterns: Build Better Applications with Coding and Design Patterns*. O'Reilly Media, 2010.

[38] P. Tetlow, E. Wallace, D. Oberle, and H. Knublauch. A semantic web primer for object-oriented software developers. W3C note, W3C, Mar. 2006. https://www.w3.org/TR/2006/NOTE-sw-oosd-primer-20060309/ [Navštíveno: 30.12.2020].

[39] Tobias Ahlin Bjerrome. Common CSS Flexbox Layout Patterns with Example Code. https://tobiasahlin.com/blog/common-flexbox-patterns/, 2019. [Navštíveno: 5.1.2021].

[40] L. Tung. Programming language popularity: JavaScript leads – 5 million new developers since 2017. https://www.zdnet.com/article/programming-language-popularity-javascript-leads-5-million-new-developers-since-2017/. [Navštíveno: 5.1.2021].

[41] U. S. G. US Army. *Systems Engineering Fundamentals*. CreateSpace Independent Publishing Platform, 2013.

[42] Vishwas Ng. Non-functional requirement of the mobile development system. https://medium.com/@vishwasng/non-functional-requirement-of-the-mobile-development-system-e0ed98f2a872. [Navštíveno: 06.5.2021].

Příloha A

Případy užití

A.1 Struktura případů užití

- ID
  - Identifikátor daného případu užití.
- Popis
  - Stručný popis přibližující daný případ užití.
- Účastník
  - Určuje, jaký účastník daný případ užití vyvolal. V případě že účastníkem je uživatel mající specifickou roli v systému, je tato role uvedena. Není-li role požadována, píšeme pouze *Uživatel*.
- Prerekvizity
  - Jedná se o vstupní podmínky, které musí být splněny, aby mohlo dojít ke spuštění případu užití.
- Úspěšný scénář
  - Posloupnost akcí které se musí stát, aby mohl být případ užití vyhodnocen jako úspěšný.
- Neúspěšný scénář
  - Definuje, za jakých okolností může určitá akce popsaná v úspěšném scénáři vyvolat neúspěch (neúspěšné vykonání celého případu užití). Pokud k těmto okolnostem dojde, neúspěšný scénář k danému bodu popisuje posloupnost akcí, které následně nastanou. Příkladem může být zobrazení chybové zprávy uživateli nebo vrácení uživatele do bodu úspěšného scénáře před vyvoláním chyby.

A.2 Use case 1 - Přihlásit se

UC: 1	**Přihlásit se**
Popis	Uživatel se přihlásí do aplikace
Účastník:	Uživatel
Prerekvizity:	Žádné

Úspěšný scénář:

1. Systém žádá autentizaci uživatele
2. Účastník vybere název serveru
3. Účastník zadá uživatelské jméno a heslo
4. Systém ověří zadané údaje
5. Systém přihlásí uživatele

Neúspěšný scénář:

4.a Oveření se nezdařilo:

1. Systém zobrazí zprávu o neúspěšném přihlášení
2. Účastník se vrací na krok č.1

A.3 Use case 2 - Odhlásit se

UC: 2	**Odhlásit se**
Popis	Uživatel se odhlásí z aplikace
Účastník:	Uživatel
Prerekvizity:	Uživatel je přihlášený

Úspěšný scénář:

1. Účastník si vyžádá odhlášení z aplikace
2. Systém uživatele odhlásí

A.4 Use case 3 - Vyhledat pojem nebo slovník

UC: 3	**Vyhledat pojem nebo slovník**
Popis	Uživatel vyhledá pojmy a slovníky odpovídající hledanému textu
Účastník:	Uživatel
Prerekvizity:	Uživatel je přihlášený

Úspěšný scénář:

1. Účastník si vyžádá vyhledávání pojmů a slovníků
2. Systém zobrazí vyhledávací obrazovku
3. Účastník vyplní hledaný text
4. Systém zobrazí pojmy a slovníky odpovídající hledanému textu

Neúspěšný scénář:

4.a Hledanému textu neodpovídají žádné výsledky:

1. Systém zobrazí zprávu o nenalezení žádných odpovídajích výsledků

A.5 Use case 4 - Zobrazit detail slovníku

UC: 4	**Zobrazit detail slovníku**
Popis	Uživatel si zobrazí detailní výpis slovníku
Účastník:	Uživatel
Prerekvizity:	Uživatel je přihlášený

Úspěšný scénář:

1. Účastník si vyžádá zobrazení detailního výpisu slovníku
2. Systém zobrazí detailní výpis slovníku

Neúspěšný scénář:

2.a Systém nenalezl detailní informace o slovníku:

1. Systém zobrazí zprávu o chybě

A.6 Use case 5 - Zobrazit detail pojmu

UC: 5	**Zobrazit detail pojmu**
Popis	Uživatel si zobrazí detailní výpis pojmu
Účastník:	Uživatel
Prerekvizity:	Uživatel je přihlášený

Úspěšný scénář:

1. Účastník si vyžádá zobrazení detailního výpisu pojmu
2. Systém zobrazí detailní výpis pojmu

Neúspěšný scénář:

2.a Systém nenalezl detailní informace o pojmu:

1. Systém zobrazí zprávu o chybě

A.7 Use case 6 - Vytvořit nový pojem

UC: 6	**Vytvořit nový pojem**
Popis	Účastník vytvoří slovníku nový pojem
Účastník:	Neomezený uživatel
Prerekvizity:	Neomezený uživatel je přihlášený

Úspěšný scénář:

1. Účastník si vyžádá vytvoření nového pojmu
2. Systém zobrazí obrazovku sloužící k vytváření nových pojmů
3. Účastník vyplní informace o pojmu
4. Účastník si vyžádá uložení nového pojmu
5. Systém uloží nový pojem
6. Systém zobrazí detailní výpis nově vytvořeného pojmu

Neúspěšný scénář:

3.a Účastník zadal název pojmu, který již ve slovníku existuje:

1. Systém zobrazí zprávu o nalezení kolize názvů
2. Účastník se vrací ke kroku č. 3

4.a Účastník nevyplnil název pojmu:

1. Účastník se vrací ke kroku č. 3

A.8 Use case 7 - Vytvořit nový slovník

UC: 7	**Vytvořit nový slovník**
Popis	Účastník vytvoří nový slovník
Účastník:	Neomezený uživatel
Prerekvizity:	Neomezený uživatel je přihlášený

Úspěšný scénář:

1. Účastník si vyžádá vytvoření nového slovníku
2. Systém zobrazí obrazovku sloužící k vytváření nových slovníků
3. Účastník vyplní informace o slovníku
4. Účastník si vyžádá uložení nového slovníku
5. Systém uloží nový slovník
6. Systém zobrazí detailní výpis nově vytvořeného slovníku

Neúspěšný scénář:

4.a Účastník zadal název slovníku, který již server eviduje:

1. Účastník se vrací ke kroku č. 3

A.9 Use case 8 - Upravit pojem

UC: 8	**Upravit pojem**
Popis	Účastník upraví již existující pojem
Účastník:	Neomezený uživatel
Prerekvizity:	Neomezený uživatel je přihlášený

Úspěšný scénář:

1. Účastník si vyžádá úpravu pojmu
2. Systém zobrazí obrazovku sloužící k úpravě pojmu
3. Účastník upraví informace o pojmu
4. Účastník si vyžádá uložení upraveného pojmu
5. Systém uloží pojem s upravenými parametry
6. Systém zobrazí detailní výpis upraveného pojmu

Neúspěšný scénář:

3.a Účastník zadal název pojmu, který již ve slovníku existuje:

1. Systém zobrazí zprávu o nalezení kolize názvů
2. Účastník se vrací ke kroku č. 3

4.a Účastník nevyplnil název pojmu:

1. Účastník se vrací ke kroku č. 3

A.10 Use case 9 - Upravit slovník

UC: 9	**Upravit slovník**
Popis	Účastník upraví již existující slovník
Účastník:	Neomezený uživatel
Prerekvizity:	Neomezený uživatel je přihlášený

Úspěšný scénář:

1. Účastník si vyžádá úpravu slovníku
2. Systém zobrazí obrazovku sloužící k úpravě slovníku
3. Účastník upraví informace o slovníku
4. Účastník si vyžádá uložení upraveného slovníku
5. Systém uloží slovník s upravenými parametry
6. Systém zobrazí detailní výpis upraveného slovníku

Neúspěšný scénář:

4.a Účastník nevyplnil název slovníku:

1. Účastník se vrací ke kroku č. 3

4.a Účastník zadal název slovníku, který již server eviduje:

1. Účastník se vrací ke kroku č. 3

A.11 Use case 10 - Smazat pojem

UC: 10	**Smazat pojem**
Popis	Účastník odstraní pojem
Účastník:	Neomezený uživatel
Prerekvizity:	Neomezený uživatel je přihlášený

Úspěšný scénář:

1. Účastník si vyžádá smazání pojmu
2. Systém zobrazí dialogové okno
3. Účastník potvrdí dialogové okno
4. Systém smaže pojem

Neúspěšný scénář:

3.a Účastník nepotvrdí dialogové okno:

1. Systém se vrací na detailní výpis pojmu

A.12 Use case 11 - Smazat slovník

UC: 11	**Smazat slovník**
Popis	Účastník odstraní slovník
Účastník:	Neomezený uživatel
Prerekvizity:	Neomezený uživatel je přihlášený

Úspěšný scénář:

1. Účastník si vyžádá smazání slovníku
2. Systém zobrazí dialogové okno
3. Účastník potvrdí dialogové okno
4. Systém smaže slovník

Neúspěšný scénář:

3.a Účastník nepotvrdí dialogové okno:

1. Systém se vrací na detailní výpis slovníku

3.b Účastník se pokouší odstranit slovník, který obsahuje pojmy:

1. Systém se vrací na detailní výpis slovníku

A.13 Use case 12 - Přidat komentář k pojmu

UC: 12	**Přidat komentář k pojmu**
Popis	Účastník přidá komentář k existujícímu pojmu
Účastník:	Uživatel
Prerekvizity:	Uživatel je přihlášený

Úspěšný scénář:

1. Účastník si vyžádá zobrazení detailního výpisu pojmu
2. Systém zobrazí detailní výpis pojmu
3. Účastník vyplní text komentáře
4. Účastník potvrdí odeslání komentáře
5. Systém uloží komentář k danému pojmu
6. Systém zobrazí aktualizovaný detailní výpis pojmu

Neúspěšný scénář:

4.a Účastník nevyplnil text komentáře:

1. Systém nezpracuje komentář
2. Účastník se vrací na krok č. 3

A.14 Use case 13 - Uložit pojem do oblíbených

UC: 13	**Uložit pojem do oblíbených**
Popis	Účastník uloží pojem do seznamu svých oblíbených položek
Účastník:	Uživatel
Prerekvizity:	■ Uživatel je přihlášený ■ Ukládaný pojem není přítomen v seznamu oblíbených položek uživatele

Úspěšný scénář:

1. Účastník si vyžádá uložení pojmu do seznamu oblíbených položek
2. Systém uloží pojem do seznamu oblíbených položek
3. Systém zobrazí informaci o uložení pojmu do seznamu oblíbených položek

A.15 Use case 14 - Uložit slovník do oblíbených

UC: 14	**Uložit slovník do oblíbených**
Popis	Účastník uloží slovník do seznamu svých oblíbených položek
Účastník:	Uživatel
Prerekvizity:	■ Uživatel je přihlášený ■ Ukládaný slovník není přítomen v seznamu oblíbených položek uživatele

Úspěšný scénář:

1. Účastník si vyžádá uložení slovníku do seznamu oblíbených položek
2. Systém uloží slovník do seznamu oblíbených položek
3. Systém zobrazí informaci o uložení slovníku do seznamu oblíbených položek

A.16 Use case 15 - Zobrazit oblíbené

UC: 15	**Zobrazit oblíbené**
Popis	Účastník si zobrazí výpis všech svých oblíbených položek
Účastník:	Uživatel
Prerekvizity:	Uživatel je přihlášený

Úspěšný scénář:

1. Účastník si vyžádá zobrazení seznamu oblíbených položek
2. Systém zobrazí všechny účastníkovy oblíbené položky

Neúspěšný scénář:

2.a Účastník nemá žádné oblíbené položky:

1. Systém vypíše informaci o prázdném seznamu oblíbených položek

A.17 Use case 16 - Odebrat položku z oblíbených

UC: 16	**Odebrat položku z oblíbených**
Popis	Účastník odebere položku ze seznamu oblíbených položek
Účastník:	Uživatel
Prerekvizity:	Uživatel je přihlášený

Úspěšný scénář:

1. Účastník si vyžádá odebrání položky ze seznamu oblíbených položek
2. Systém odebere položku ze seznamu oblíbených položek

A.18 Use case 17 - Zobrazit nedávnou historii

UC: 17	**Zobrazit nedávnou historii**
Popis	Účastník si zobrazí svojí nedávnou historii vyhledávání
Účastník:	Uživatel
Prerekvizity:	Uživatel je přihlášený

Úspěšný scénář:

1. Účastník si vyžádá zobrazené své nedávné historie
2. Systém zobrazí seznam posledních pěti vyhledávaných položek

A.19 Use case 18 - Odstranit položky nedávné historie

UC: 18	**Odstranit položky nedávné historie**
Popis	Účastník odstraní všechny položky z nedávné historie vyhledávání
Účastník:	Uživatel
Prerekvizity:	Uživatel je přihlášený

Úspěšný scénář:

1. Účastník si vyžádá odstranění všech položek nedávné historie
2. Systém odstraní všechny položky z výpisu nedávné historie

A.20 Use case 19 - Změnit URL adresu serveru

UC: 19	**Změnit URL adresu serveru**
Popis	Účastník si zvolí server, se kterým bude aplikace komunikovat
Účastník:	Uživatel
Prerekvizity:	Žádné

Úspěšný scénář:

1. Účastník si vyžádá výběr dostupných serverů
2. Systém zobrazí názvy dostupných serverů
3. Účastník si vybere server
4. Systém nastaví koncovou URL adresu serveru

A.21 Use case 20 - Kopírovat text do schránky

UC: 20	**Kopírovat text do schránky**
Popis	Účastník si označí text, který následně zkopíruje do schránky
Účastník:	Uživatel
Prerekvizity:	Přihlášený uživatel

Úspěšný scénář:

1. Účastník označí text
2. Systém označený text zvýrazní
3. Účastník si vyžádá zkopírování označovaného textu do schránky
4. Systém uloží označený text do schránky

Neúspěšný scénář:

1.a Účastník se pokusí označit text, který není názvem pojmu, synonymem pojmu, definicí pojmu, zdrojem definice pojmu, názvem slovníku nebo popisem slovníku:

1. Systém text neoznačí

Příloha B

Popis doménového modelu

B.1 User - uživatel

Atribut	Popis atributu
firsName	křestní jméno uživatele
lastName	příjmení uživatele
restricted	určuje, zdali má uživatel plná práva v mobilní aplikaci
types	uživatelské role
uri	jednoznačný identifikátor uživatele
userName	uživatelské jméno

Tabulka B.1: Popis atributů User entity

B.2 Term - pojem

Atribut	Popis atributu
altLabels	synonyma pojmu (možné ve více jazycích)
definition	definice pojmu (možné ve více jazycích)
description	popis pojmu (možné ve více jazycích)
draft	určuje, zdali se jedná o koncept pojmu
glossary	identifikuje glosář
label	název pojmu (možné ve více jazycích)
properties	ostatní vlastnosti pojmu
sources	zdroje definic
types	typy pojmu
uri	jednoznačný identifikátor pojmu

Tabulka B.2: Popis atributů Term entity

B.3 Vocabulary - slovník

Atribut	Popis atributu
description	popis slovníku
document	přidružený dokument ke slovníku
glossary	glosář (obsahuje identifikátor, a kořenové pojmy)
importedVocabularies	importované slovníky
label	název slovníku
model	jednoznačný identifikátor modelu
properties	ostatní vlastnosti slovníku
uri	jednoznačný identifikátor slovníku

Tabulka B.3: Popis atributů Vocabulary entity

B.4 SubTerm - podřazený pojem

Atribut	Popis atributu
label	název pojmu
uri	jednoznačný identifikátor pojmu
vocabulary	jednoznačný identifikátor slovníku

Tabulka B.4: Popis atributů SubTerm entity

B.5 Favorite - oblíbená položka

Atribut	Popis atributu
detailScreen	určuje název obrazovky obsahující detailní výpis oblíbené položky
key	jednoznačný identifikátor pojmu/slovníku
label	název pojmu/slovníku
vocabulary	název slovníku kterému pojem náleží (prázdné, pokud je oblíbená položka slovníkem)

Tabulka B.5: Popis atributů Favorite entity

B.6 SearchResult - výsledek hledání

Atribut	Popis atributu
detailScreen	určuje název obrazovky obsahující detailní výpis hledané položky
label	název hledané položky
score	číslo udávající relevanci hledané položky
snippetField	typ náhledu
snippetFields	typy náhledů
snippets	náhledy
snippetText	náhled
type	typ výsledků (pojem/slovník)
uri	jednoznačný identifikátor pojmu/slovníku
vocabulary	název slovníku (prázdné, pokud je hledaná položka pojmem)

Tabulka B.6: Popis atributů SearchResult entity

B.7 History - položka historie

Atribut	Popis atributu
detailScreen	určuje název obrazovky obsahující detailní výpis položky historie
label	název položky historie
type	typ historické položky (pojem/slovník)
uri	jednoznačný identifikátor pojmu/slovníku
vocabularyLabel	název slovníku (prázdné, pokud je hledaná položka pojmem)

Tabulka B.7: Popis atributů History entity

B.8 Comment - komentář

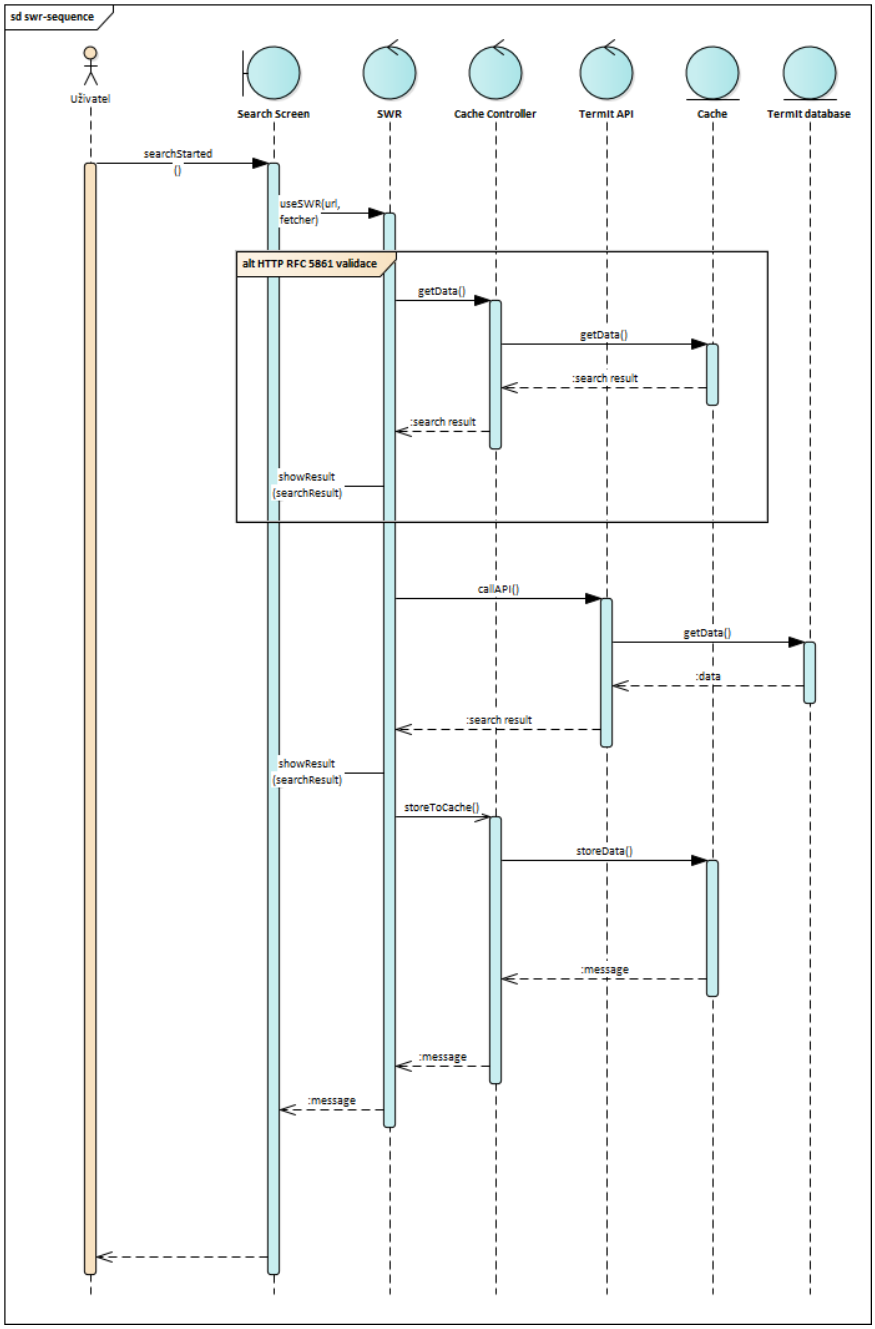
Atribut	Popis atributu
asset	jednoznačný identifikátor pojmu
content	obsah komentáře
created	časová značka vytvoření komentáře
uri	jednoznačný identifikátor komentáře

Tabulka B.8: Popis atributů Comment entity

Příloha C

Diagramy

C.1 Sekvenční diagram API komunikace



Obrázek C.1: Sekvenční diagram vyhledávání pojmů a slovníku za pomocí SWR

Příloha D

Testovací scénáře

D.1 Přihlášení do aplikace

Test ID	LOGIN_01
Název testu	Přihlášení do aplikace
Shrnutí testu	Uživatel se přihlásí přes vybraný server do aplikace.
Popis testu	Uživatel zvolí server, se kterým chce, aby aplikace komunikovala. Po platném přihlášením je uživatel přesměrován na domovskou stránku aplikace.
Kroky testu	1. Vyberte název serveru
	2. Vyplňte přihlašovací jméno
	3. Vyplňte heslo
	4. Klikněte na tlačítko „Přihlásit se“
Vstupní podmínky	
Testovací data	Název serveru: server-dev
	Přihlašovací jméno: test
	Heslo: test
Očekávaný výsledek	Uživatel se úspěšně autentizuje vůči zvolenému serveru a je přesměrován na domovskou obrazovku aplikace.

Tabulka D.1: Testovací scénář: Přihlášení do aplikace

D.2 Vyhledávání pojmu

Test ID	SEARCH_01
Název testu	Vyhledávání pojmu
Shrnutí testu	Vyhledání pojmu, zobrazení jeho detailu
Popis testu	Přihlášený uživatel vyhledá pojem pomocí jeho názvu, následně si zobrazí jeho detail.
Kroky testu	1. Na hlavní obrazovce klikněte na tlačítko „hledat“
	2. Do vstupního textového pole napište název pojmu
	3. Klikněte na tlačítko nesoucí název hledaného pojmu v příslušném slovníku
Vstupní podmínky	· Přihlášený uživatel
	· Uživatel se nachází na domovské obrazovce
Testovací data	Název pojmu: Budova
	Příslušný slovník: Decree No. 268/2009
Očekávaný výsledek	Uživateli je zobrazena detailní stránka, popisující hledaný pojem

Tabulka D.2: Testovací scénář: Vyhledávání pojmu

D.3 Vyhledávání slovníku

Test ID	SEARCH_02
Název testu	Vyhledávání slovníku
Shrnutí testu	Vyhledání slovníku, zobrazení jeho detailu
Popis testu	Přihlášený uživatel vyhledá slovník pomocí jeho názvu, následně si zobrazí jeho detail.
Kroky testu	1. Na hlavní obrazovce klikněte na tlačítko „hledat“
	2. Do vstupního textového pole napište název slovníku
	3. Klikněte na tlačítko nesoucí název hledaného slovníku
Vstupní podmínky	· Přihlášený uživatel
	· Uživatel se nachází na domovské obrazovce
Testovací data	Název slovníku: Portál covid-19
Očekávaný výsledek	Uživateli je zobrazena detailní stránka, popisující hledaný slovník

Tabulka D.3: Testovací scénář: Vyhledávání slovníku

D.4 Vyhledání slovníku přes výpis všech slovníků

Test ID	SEARCH_03
Název testu	Vyhledání slovníku přes výpis všech slovníků
Shrnutí testu	Navigace na detailní výpis slovníku přes výpis všech dostupných slovníků na serveru.
Popis testu	Přihlášený uživatel si zobrazí všechny slovníky, které server spravuje. Posléze vybere hledaný slovník a zobrazí si jeho detailní výpis.
Kroky testu	1. Na hlavní obrazovce klikněte na tlačítko „slovníky“.
	2. Klikněte na hledaný slovník
Vstupní podmínky	· Přihlášený uživatel
	· Uživatel se nachází na domovské obrazovce
Testovací data	Hledaný slovník: Portál covid-19
Očekávaný výsledek	Uživateli je zobrazen výpis všech dostupných slovníků. Po kliknutí na položku je uživatel přesměrován na detailní výpis daného slovníku.

Tabulka D.4: Testovací scénář: Vyhledání slovníku přes výpis všech slovníků

D.5 Uložení slovníku do oblíbených

Test ID	FAVORITES_01
Název testu	Uložení slovníku do oblíbených
Shrnutí testu	Uložení slovníku do seznamu oblíbených položek
Kroky testu	1. Na hlavní obrazovce klikněte na tlačítko „hledat“
	2. Do vstupního textového pole napište název pojmu
	3. Klikněte na tlačítko nesoucí název hledaného slovníku
	4. Na spodním panelu detailu slovníku klikněte na tlačítko oblíbit
Vstupní podmínky	· Přihlášený uživatel
	· Ukládaný slovník se nenachází v seznamu oblíbených položek
	· Uživatel se nachází na domovské obrazovce
Testovací data	Název slovníku: Portál covid-19
Očekávaný výsledek	Uživatel uložil slovník do seznamu oblíbených položek. Tlačítko spodního panelu se změnilo z „oblíbit“ na „oblíbeno“. Uložená položka je k dispozici na výpisu oblíbených položek.

Tabulka D.5: Testovací scénář: Uložení slovníku do oblíbených

D.6 Odebrání položky z oblíbených

Test ID	FAVORITES_02
Název testu	Odebrání položky z oblíbených
Shrnutí testu	Odebrání položky ze seznamu oblíbených položek
Popis testu	Přihlášený uživatel odebere položku ze seznamu oblíbených položek.
Kroky testu	1. Odebíranou položku potáhněte směrem do levé strany obrazovky
	2. Klikněte na tlačítko „odebrat“
Vstupní podmínky	· Přihlášený uživatel
	· Uživatel se nachází na domovské obrazovce
	· Odebíraná položka se nachází v seznamu oblíbených položek
Testovací data	
Očekávaný výsledek	Odebíraná položka se nenachází v seznamu oblíbených položek

Tabulka D.6: Testovací scénář: Odebrání položky z oblíbených

D.7 Přidání položky do historie

Test ID	HISTORY_01
Název testu	Přidání položky do historie
Shrnutí testu	Vyhledávaná položka se přidá do historie vyhledávaných položek. Položka v historii slouží jako přesměrovaní na zobrazení detailního výpisu dané položky.
Popis testu	Přihlášený uživatel vyhledá pojem. Po zobrazení detailu pojmu se pojem přidá do seznamu vyhledávaných položek. Tento seznam posléze uživatel využije pro zobrazení detailu, jím dříve vyhledávaného pojmu.
Kroky testu	1. Na hlavní obrazovce klikněte na tlačítko „hledat“
	2. Do vstupního textového pole napište název pojmu
	3. Klikněte na tlačítko nesoucí název hledaného pojmu v příslušném slovníku
	4. Klikněte na tlačítko menu v pravém horním rohu
	5. Klikněte na možnost „hledat“
	6. Ve výpisu položek historie klikněte na vámi předešlou vyhledávanou položku
Vstupní podmínky	· Přihlášený uživatel
	· Uživatel se nachází na domovské obrazovce
Testovací data	Název pojmu: Služební předpis
	Příslušný slovník: Slovník veřejné správy

Tabulka D.7: Testovací scénář: Přidání položky do historie

D.8 Odebrání všech položek historie

Test ID	HISTORY_02
Název testu	Odebrání všech položek historie
Shrnutí testu	Ze seznamu historie položek jsou odebrány všechny položky
Popis testu	Přihlášený uživatel odstraní veškeré položky v seznamu historie prohlížených položek
Kroky testu	1. Klikněte na tlačítko „hledat“
	2. Klikněte na tlačítko „Vymazat historii“
Vstupní podmínky	· Uživatel se nachází na domovské obrazovce
Testovací data	
Očekávaný výsledek	Seznam historie vyhledávaných položek je prázdný

Tabulka D.8: Testovací scénář: Odebrání všech položek historie

D.9 Vytvoření slovníku

Test ID	CREATE_VOCABULARY_01
Název testu	Vytvoření slovníku
Popis testu	Uživatel vytvoří nový slovník s danými parametry.
Kroky testu	1. Klikněte na tlačítko „slovníky“
	2. Na spodním panelu výpisu všech slovníků klikněte na tlačítko „nový slovník“
	3. Vyplňte název
	4. Vyplňte popis
	5. Klikněte na tlačítko „uložit“
Vstupní podmínky	· Přihlášený neomezený uživatel
	· Uživatel se nachází na domovské obrazovce
	· Název nového slovníku se v databázi nevyskytuje
Testovací data	Název: Test nový slovník 123
	Popis: Popis nového testovacího slovníku 123
Očekávaný výsledek	Uživatel vytvoří nový slovník s vyplněnými parametry, který server uloží. Po jeho vytvoření je uživatel přesměrován na detailní výpis nově vytvořeného slovníku. Detailní výpis obsahuje všechny uživatelem zadané parametry.

Tabulka D.9: Testovací scénář: Vytvoření slovníku

D.10 Vytvoření pojmu

Test ID	CREATE_TERM_01
Název testu	Vytvoření pojmu
Shrnutí testu	Uživatel vytvoří nový pojem
Popis testu	Neomezený uživatel vytvoří nový pojem s danými parametry.
Kroky testu	1. Na spodním panelu detailu slovníku klikněte na tlačítko „nový pojem“
	2. Vyplňte název
	3. Přidejte synonyma
	4. Vyberte typ pojmu
	5. Vyplňte text definice
	6. Vyplňte zdroj definice
	7. Klikněte na tlačítko „uložit“
Vstupní podmínky	· Přihlášený neomezený uživatel
	· Uživatel se nachází na detailním výpisu slovníku
Testovací data	Název: TestXXX
	Synonyma: Test XXX, XXX
	Typ pojmu: Objekt
	Text definice: Definice XXX
	Zdroj definice: Zdroj XXX
Očekávaný výsledek	Uživatel vytvoří nový pojem, který server uloží se všemi vyplněnými parametry. Po vytvoření pojmu je uživatel přesměrován na detailní výpis nově vytvořeného pojmu. Detailní výpis obsahuje všechny uživatelem zadané parametry.

Tabulka D.10: Testovací scénář: Vytvoření pojmu

D.11 Editace slovníku

Test ID	EDIT_VOCABULARY_01
Název testu	Editace slovníku
Shrnutí testu	Existujícímu slovníku jsou upraveny parametry
Popis testu	Neomezený uživatel upraví slovník změnou jeho parametrů
Kroky testu	1. Na spodním panelu detailu slovníku klikněte na tlačítko „upravit“
	2. Upravte název
	3. Upravte popis
	4. Klikněte na tlačítko „uložit“
Vstupní podmínky	· Přihlášený neomezený uživatel
	· Uživatel se nachází na detailním výpisu slovníku
Testovací data	Název: Test nový slovník 123 - změna
	Popis: Úprava popisu 123
Očekávaný výsledek	Uživatel upraví již existující slovník s nově zadanými parametry, které server uloží. Po úpravě slovníku je uživatel přesměrován na detailní výpis nově upraveného slovníku. Detailní výpis obsahuje všechny uživatelem zadané parametry.

Tabulka D.11: Testovací scénář: Editace slovníku

D.12 Editace pojmu

Test ID	EDIT_TERM_01
Název testu	Editace pojmu
Shrnutí testu	Existujícímu pojmu jsou upraveny parametry
Popis testu	Neomezený uživatel upraví pojem změnou jeho parametrů
Kroky testu	1. Na spodním panelu detailu pojmu klikněte na tlačítko „upravit“
	2. Upravte název
	3. Upravte synonyma
	4. Upravte typ pojmu
	5. Upravte text definice
	6. Upravte zdroj definice
	7. Klikněte na tlačítko „uložit“
Vstupní podmínky	· Přihlášený neomezený uživatel
	· Uživatel se nachází na detailním výpisu pojmu
Testovací data	Název: TestXXX - upravený
	Synonyma: Test XXX
	Typ pojmu: Individuál
	Text definice: Definice XXX - upravená
	Zdroj definice: Zdroj XXX - upravená
Očekávaný výsledek	Uživatel upraví již existující pojem s nově zadanými parametry, které server uloží. Po úpravě pojmu je uživatel přesměrován na detailní výpis nově upraveného pojmu. Detailní výpis obsahuje všechny uživatelem zadané parametry.

Tabulka D.12: Testovací scénář: Editace pojmu

D.13 Odstranění pojmu

Test ID	DELETE_TERM_01
Název testu	Odstranění pojmu
Shrnutí testu	Odstranění pojmu z databáze
Popis testu	Neomezený uživatel odstraní pojem z databáze
Kroky testu	1. Na spodním panelu detailu pojmu klikněte na tlačítko „smazat“
	2. Potvrďte dialogové okno kliknutím na tlačítko „smazat“
Vstupní podmínky	· Přihlášený neomezený uživatel
	· Uživatel se nachází na detailním výpisu pojmu
Testovací data	
Očekávaný výsledek	Uživatel odstraní pojem z databáze. Po odstranění pojmu je uživatel přesměrován na domovskou obrazovku.

Tabulka D.13: Testovací scénář: Odstranění pojmu

D.14 Odstranění slovníku

Test ID	DELETE_VOCABULARY_01
Název testu	Odstranění slovníku
Shrnutí testu	Odstranění prázdného slovníku z databáze
Popis testu	Neomezený uživatel odstraní prázdný slovník z databáze
Kroky testu	1. Na spodním panelu detailu slovníku klikněte na tlačítko „smazat“
	2. Potvrďte dialogové okno kliknutím na tlačítko „smazat“
Vstupní podmínky	· Přihlášený neomezený uživatel
	· Uživatel se nachází na detailním výpisu slovníku
	· Slovník neobsahuje žádné pojmy
Testovací data	
Očekávaný výsledek	Uživatel odstraní slovník z databáze. Po odstranění slovníku je uživatel přesměrován na domovskou obrazovku.

Tabulka D.14: Testovací scénář: Odstranění slovníku

D.15 Okomentování pojmu

Test ID	ADD_COMMENT _01
Název testu	Okomentování pojmu
Shrnutí testu	Vytvoření nového komentáře k pojmu
Popis testu	Uživatel přidá komentář k pojmu.
Kroky testu	1. Vyplňte text komentáře
	2. Klikněte na tlačítko „Přidat komentář“
Vstupní podmínky	· Přihlášený uživatel
	· Uživatel se nachází na detailním výpisu pojmu
Testovací data	Text komentář: Test komentáře
Očekávaný výsledek	Uživatel přidá komentář k pojmu. Po uložení komentáře serverem je komentář zobrazen v detailním výpisu pojmu.

Tabulka D.15: Testovací scénář: Okomentování pojmu

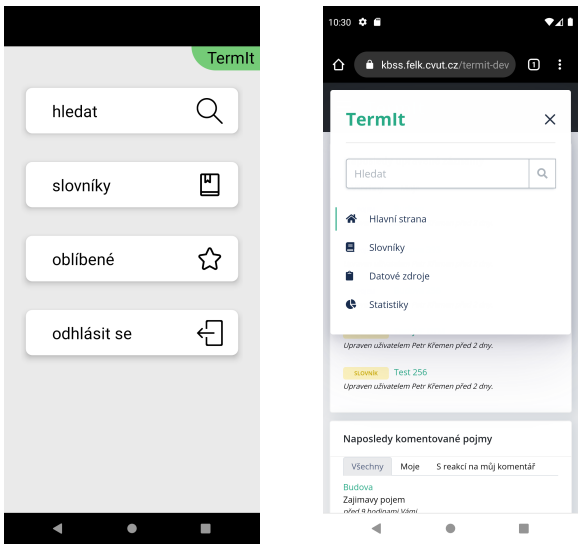
Příloha E

Ukázky aplikace

E.1 Přihlašovací obrazovka

Obrázek E.1: Přihlašovací obrazovka mobilní aplikace

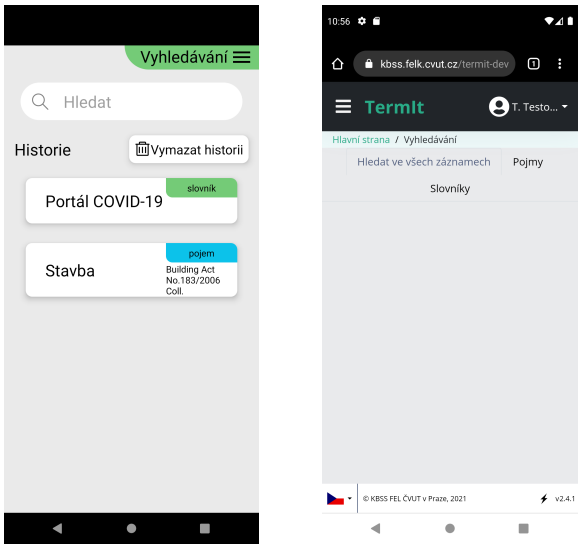
E.2 Srovnání menu



(a) : Mobilní aplikace **(b) :** Webová verze

Obrázek E.2: Srovnání menu

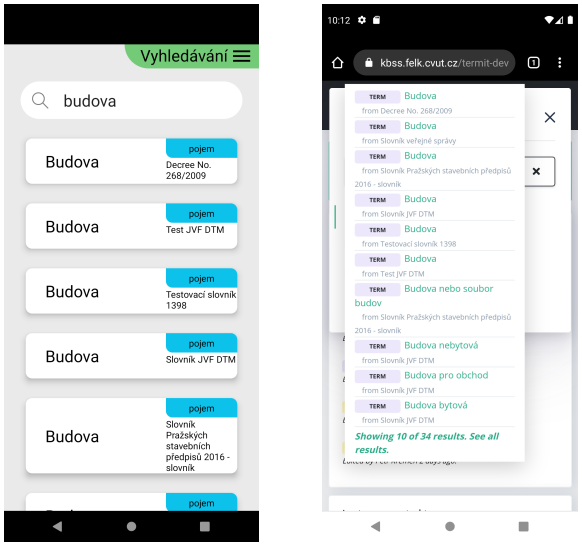
E.3 Srovnání vyhledávání



(a) : Mobilní aplikace **(b) :** Webová verze

Obrázek E.3: Srovnání vyhledávání

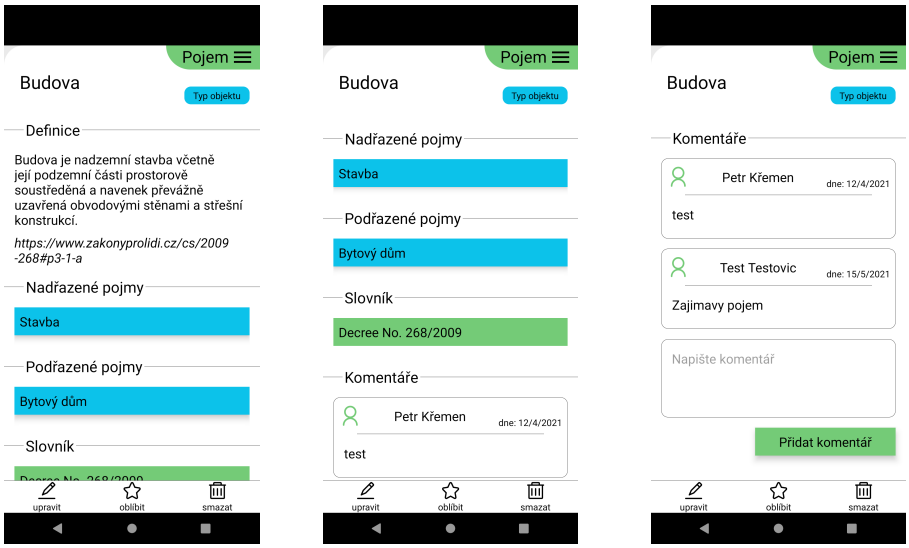
E.4 Srovnání výsledků vyhledávání



(a) : Mobilní aplikace **(b) :** Webová verze

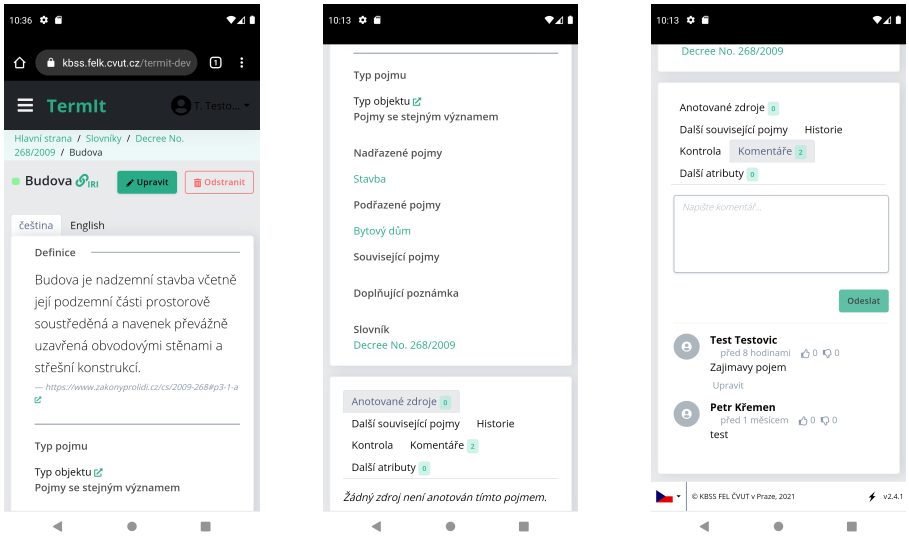
Obrázek E.4: Srovnání výsledků vyhledávání

E.5 Srovnání výpisu pojmu



(a) : Výpis pojmu 1. část **(b) :** Výpis pojmu 2. část **(c) :** Výpis pojmu 3. část

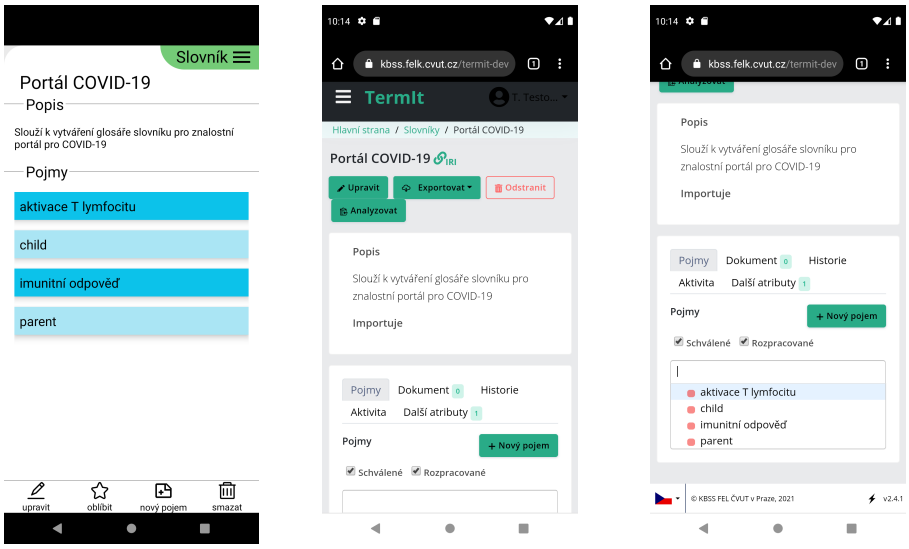
Obrázek E.5: Výpis pojmu mobilní aplikací



(a) : Výpis pojmu 1. část **(b) :** Výpis pojmu 2. část **(c) :** Výpis pojmu 3. část

Obrázek E.6: Výpis pojmu webovou verzí

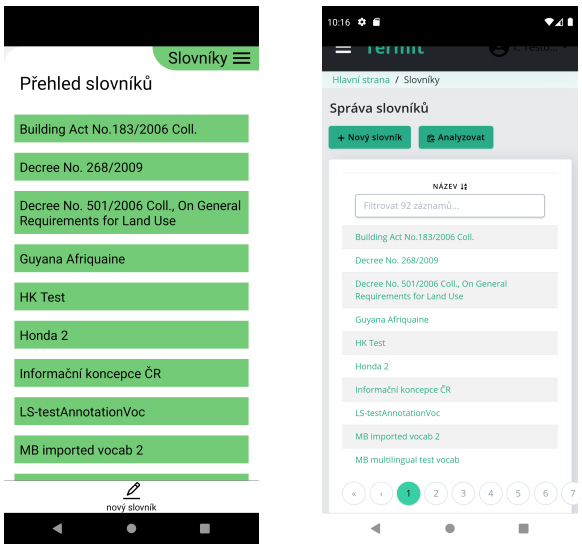
E.6 Srovnání výpisu slovníku



(a) : Mobilní aplikace **(b) :** Webové verze 1. část **(c) :** Webové verze 2. část

Obrázek E.7: Srovnání výpisu slovníku

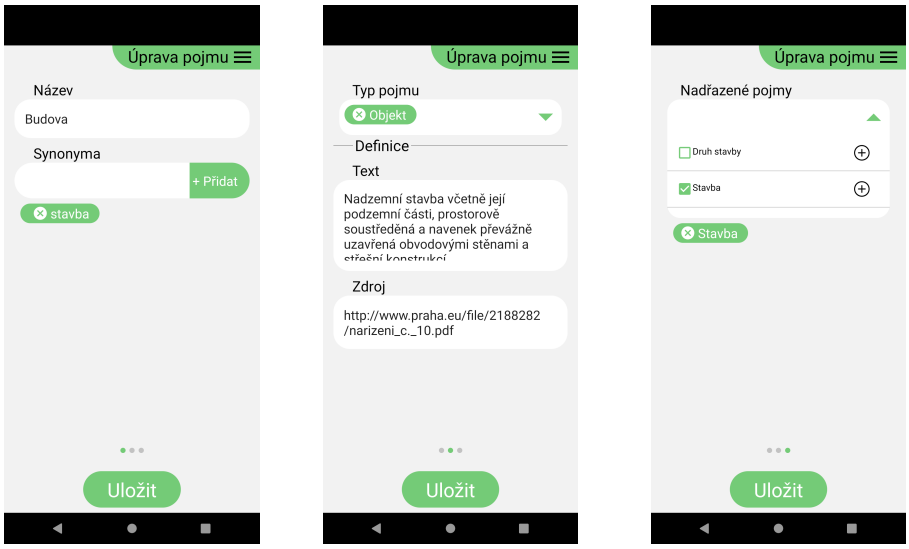
E.7 Srovnání výpisu dostupných slovníků



(a) : Mobilní aplikace **(b) :** Webové verze

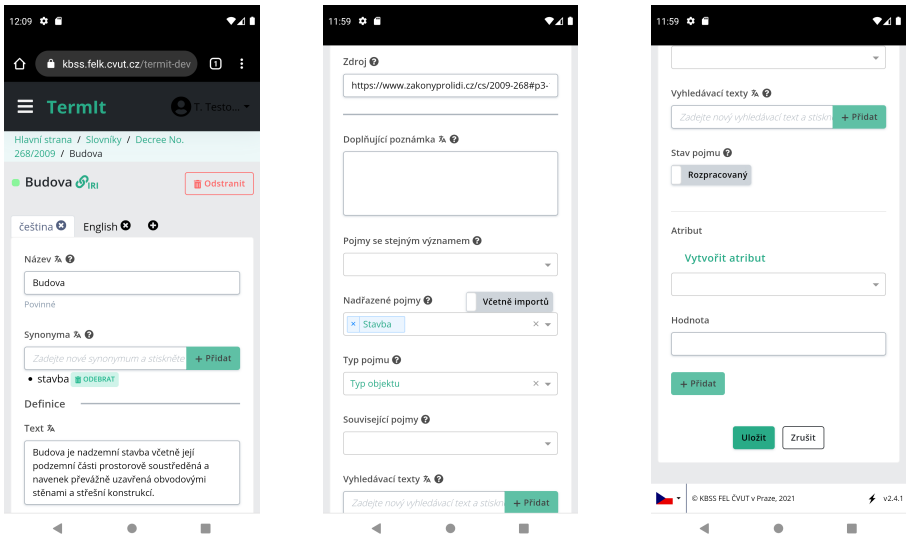
Obrázek E.8: Srovnání výpisu dostupných slovníku

E.8 Srovnání úpravy pojmu



(a) : Úprava pojmu 1. část **(b) :** Úprava pojmu 2. část **(c) :** Úprava pojmu 3. část

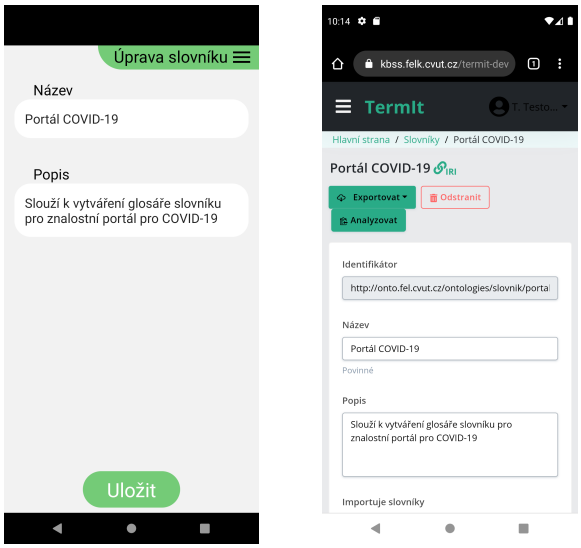
Obrázek E.9: Úprava pojmu mobilní aplikací



(a) : Úprava pojmu 1. část **(b) :** Úprava pojmu 2. část **(c) :** Úprava pojmu 3. část

Obrázek E.10: Úprava pojmu webovou verzí

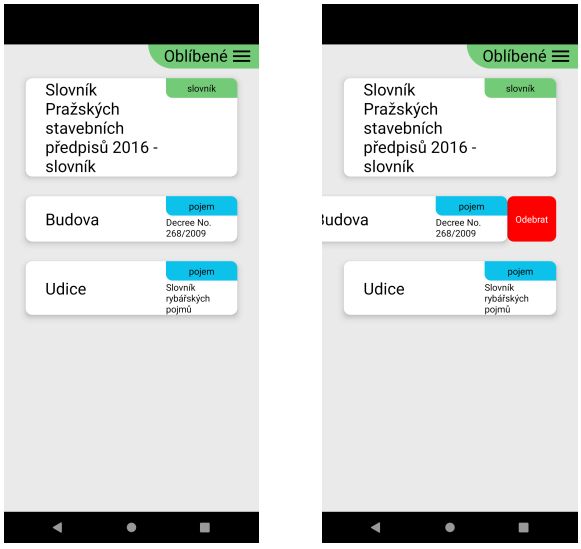
E.9 Srovnání úpravy slovníku



(a) : Mobilní aplikace **(b) :** Webová verze

Obrázek E.11: Srovnání úpravy slovníku

E.10 Oblíbené položky



(a) : Seznam oblíbených položek **(b) :** Možnost odebrání položky

Obrázek E.12: Seznam oblíbených položek

Příloha F

TermIt - serverová část

F.1 API koncové body

Výpis používaných API koncových bodů mobilní aplikací. Texty ve složených závorkách jsou proměnlivé v závislosti na požadovaném zdroji. U koncových bodů nejsou poznamenané dodatečné parametry dotazů (query strings), pro úplný přehled včetně dodatečných parametrů se prosím odkažte na oficiální dokumentaci[1].

Popis	**HTTP metoda**	**API koncový bod**
Přihlášení	POST	/j_spring_security_check
Aktuálně přihlášený uživatel	GET	/rest/users/current
Kořenové pojmy slovníku	GET	/rest/vocabularies/{název slovníku} /terms/roots
Podpojmy pojmu	GET	/rest/vocabularies/{název slovníku} /terms/{název pojmu}/subterms
Typy pojmů	GET	/rest/language/types
Dostupné slovníky	GET	/rest/vocabularies
Detail pojmu	GET	/rest/vocabularies/{název slovníku} /terms/{název pojmu}
Upravit pojem	PUT	/rest/vocabularies/{název slovníku} /terms/{název pojmu}
Vytvořit pojem	POST	/rest/vocabularies/{název slovníku} /terms/{název pojmu}
Smazat pojem	DELETE	/rest/vocabularies/{název slovníku} /terms/{název pojmu}
Detail slovníku	GET	/rest/vocabularies/{název slovníku}
Upravit slovník	PUT	/rest/vocabularies/{název slovníku}
Vytvořit slovník	POST	/rest/vocabularies
Smazat slovník	DELETE	/rest/vocabularies/{název slovníku}
Komentáře k pojmu	GET	/rest/terms/{název pojmu}/comments
Kontrola dostupnosti jména	HEAD	/rest/vocabularies/{název slovníku}/terms
Vygenerování uri pro zdroj	POST	/rest/identifiers
Získání názvu pro zdroj	GET	/rest/data/label
Vyhledávání	GET	/rest/search/fts

[1] Dokumentace celého API: https://app.swaggerhub.com/apis/ledvima1/TermIt/ navštíveno: 18.5.2020

Příloha G

Obsah elektronické přílohy

```
Zdrojove kody aplikace.zip
└── TermItMobileApp ..... Kořenový adresář projektu
    ├── __mocks__ ..... Mocky pro testování
    ├── actions ..... Action creators pro Redux
    ├── assets ..... Doplňkový obsah (multimédia)
    ├── components ..... React komponenty
    ├── models ..... Datové modely
    ├── reducers ..... Reducers pro Redux
    ├── screens ..... Obrazovky
    ├── store ..... Definice typů pro jednotlivé reducers
    ├── utils ..... Nápomocné funkce, typy a konstanty
    ├── App.js ..... Vstupní bod aplikace
    ├── app.json ..... Nastavení aplikace
    ├── babel.config.js ..... Nastavení pro Babel
    ├── jest.config.js ..... Nastavení pro Jest
    ├── package.json ..... Seznam node modulů
    ├── package-lock.json ..... Seznam node modulů
    ├── tsconfig.json ..... Nastavení pro TypeScript
    └── README.md ..... Manuál pro instalaci a spouštění aplikace
```