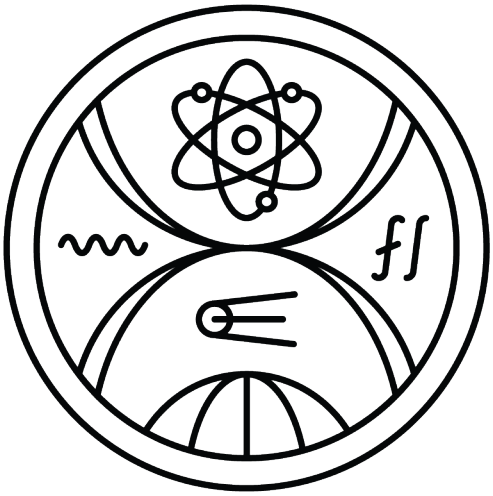
UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

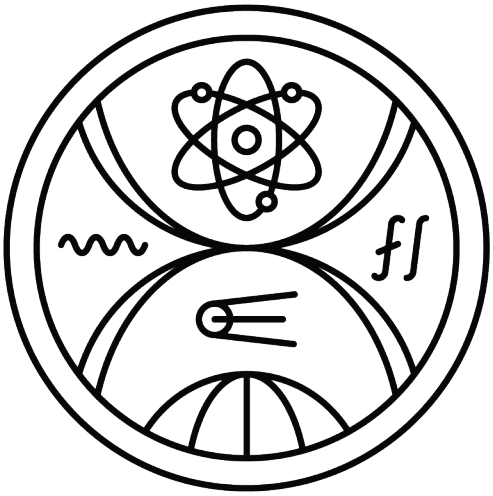


PRACOVNÁ PLOCHA VÝSKUMNÍKA

Diplomová práca

2023 Bc. Matej Dráb

UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY



PRACOVNÁ PLOCHA VÝSKUMNÍKA

Diplomová práca

Študijný program:	Aplikovaná informatika
Študijný odbor:	18 Aplikovaná informatika
Školiace pracovisko:	Katedra aplikovanej informatiky
Školiteľ:	Mgr. Pavol Petrovič, PhD.

Bratislava, 2023 Bc. Matej Dráb

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Bc. Matej Dráb
Študijný program: aplikovaná informatika (Jednoodborové štúdium, magisterský II. st., denná forma)
Študijný odbor: informatika
Typ záverečnej práce: diplomová
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: Pracovná plocha výskumníka
Researcher's Desktop

Anotácia: Cieľom práce je pomocou moderných technológií Java EE a Spring vyvinúť aplikáciu, ktorá bude slúžiť ako knowledge base a zároveň odkladacia skriňa pre výskumníka. Eviduje tam svoje články a prezentácie, posudky, práce, ktoré viedol, správy, pracovné cesty, kontakty, projekty, literatúru, sleduje konferencie a odbornú literatúru, vedie si zoznam svojich citácií a všetko má dostupné kdekoľvek z internetu. Všetky záznamy je možné navzájom previazať, pridávať k nim prílohy, exerné odkazy, klúčové slová a komentáre. Systém má responzívny design. Práca nadväzuje na predchádzajúce bakalárske práce, ktoré boli vyvíjané vo frameworku GWT, resp. jazyku node-js, ktorých výsledkom bolo iba rozpracovanie prototypu, avšak kvôli rozsahu nad rámec bakalárskej práce sa ani raz nepodarilo dospieť k úplnému a nasaditeľnému systému. Applikáciu na základe prototypov je nutné vyvinúť od začiatku, keďže predchádzajúce verzie stavajú na menej aktuálnej technológii. Výsledný systém má zároveň slúžiť ako ukážková aplikácia vo frameworku Spring pre magisterský predmet Java EE. Súčasťou práce bude obsažná prehľadová časť o tejto technológii.

Literatúra:
Miroslav Gregorec: Pracovná plocha výskumníka, diplomová práca, FMFI UK, 2017.
Katarína Matysová: Pracovná plocha výskumníka, bakalárska práca, FMFI UK, 2011.
Dinesh Rajput: Designing Applications with Spring Boot 2.2 and React JS: Step-by-step guide to design and develop intuitive full stack web applications, BPB Publications, 2019.

Kľúčové slová: java ee, spring, web application

Vedúci: Mgr. Pavel Petrovič, PhD.
Katedra: FMFI.KAI - Katedra aplikovanej informatiky
Vedúci katedry: prof. Ing. Igor Farkaš, Dr.

Dátum zadania: 21.11.2020

Dátum schválenia: 26.11.2020 prof. RNDr. Roman Ďurikovič, PhD.
garant študijného programu

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

....

študent vedúci práce

Čestne prehlasujem, že túto diplomovú prácu som vypracoval samostatne len s použitím uvedenej literatúry a za pomoci konzultácií u môjho školiteľa.

. . . . . .

Bratislava, 2023 Bc. Matej Dráb

Poďakovanie

Chcem sa poďakovať svojmu školiteľovi Mgr. Pavlovi Petrovičovi, PhD., za cennú pomoc, rady, konzultácie a čas, ktorý mi venoval počas písania diplomovej práce.

Abstrakt

Táto práca sa venuje vytvoreniu webovej aplikácie, ktorá slúži na ukladanie údajov rôzneho typu. Tieto údaje sú usporiadané do jednotlivých kategórií. Ku každému údaju sa dajú pripojiť súbory, komentáre, url linky a odkazy na iné údaje v aplikácii. Aplikácia obsahuje možnosť fulltextového vyhľadávania a vyhľadávania pomocou kľúčových slov pomocou logických formúl v konjunktívnom normálovom tvare. Práca obsahuje popis funkcionality jednotlivých častí aplikácie. Taktiež sa v nej nachádza porovnanie s už existujúcimi systémami. Jednou z častí práce je aj výskum, ktorý sa zaoberá dynamickým odporúčaním kľúčových slov na základe podobnosti záznamov. Navrhli sme viacero metód na uprednostňovanie a odporúčanie kľúčových slov, tieto metódy vyhodnocujeme na datasete veľkosti 120 záznamov, ktorý poskytuje cieľový používateľ systému. Ich porovnanie je prezentované v grafoch, ktoré ukazujú výhody niektorých metód oproti iným metódam. Výsledkom práce je funkčná webová aplikácia na základe funkčných požiadaviek, ktorá umožňuje jednoducho kategorizovať a vyhľadávať potrebné údaje. Aplikácia je naprogramovaná v jazyku Javascript, konkrétne vo frameworku React a v jazyku Java. Aplikácia je nasadená do reálneho prostredia a uľahčuje tak prácu jej používateľom.

Kľúčové slová: React, Springboot, kľúčove slová, webová aplikácia

Abstract

This thesis is devoted to the development of a web application that is used to store various types of data. This data is organized into individual categories. Files, comments, url links and links to other data in the application can be attached to each data. The application includes the option of full-text search and keyword search using logical formulas in conjunctive normal form. The thesis contains a description of the functionality of individual parts of the application. It also contains a comparison with other existing systems. We study the research that deals with the dynamic recommendation of keywords based on the similarity of records. We design several methods for prioritizing and recommending keywords and evaluate them on a dataset of 120 records provided by the target user of the system. Their comparison is presented in charts that show significant advantage of some methods over others. The result of the thesis is a web application based on functional requirements, which allows easy categorization and search of necessary data. The application is programmed in the Javascript language, specifically in the React framework and in the Java language. The application is deployed in a real environment and thus facilitates the work of its users.

Keywords: React, Springboot, keywords, web application

Obsah

1 Úvod **1**

1.1 Štruktúra práce 2

2 Požiadavky **3**

2.1 Používatelia 3

2.2 Funkčné požiadavky 3

2.2.1 Komentáre 4

2.2.2 Kľúčové slová 4

2.2.3 Odkazy na iné záznamy 4

2.2.4 URL linky 5

2.2.5 Prílohy 5

2.2.6 Zoznam uchovávaných kategórií 5

2.2.7 Publikácie 6

2.2.8 Projekty 6

2.2.9 Posudky 6

2.2.10 Pracovné cesty 7

2.2.11 Konferencie 7

2.2.12 Kontakty 7

2.2.13 Knihy 7

2.2.14 Evidencia majetku 8

2.2.15 Podujatia 8
2.2.16 Softvér 8
2.2.17 Záverečné práce 8
2.2.18 Spolupráca 9
2.2.19 Rozpracované publikácie 9
2.2.20 Weby 9
2.2.21 Dokumenty 9
2.2.22 Ostatné 9
2.3 Základné charakteristiky systému 10
2.3.1 Prístup, jednoduchosť, rozšíriteľnosť 10
2.3.2 Previazanosť záznamov 10
2.3.3 Kalendár 11
2.3.4 Vyhľadávanie 11
2.3.5 Dynamické prideľovanie kľúčových slov 11

3 Predchádzajúce práce **13**

4 Existujúce riešenia **15**
4.1 Evernote 15
4.2 Outlook 16
4.3 Google Drive 17

5 Technológie **19**
5.1 React.js 19
5.2 Node.js 20
5.3 Axios 20
5.4 Spring Boot 21
5.5 Mysql 21
5.6 Spring Data 22

6 Výskum **24**

6.1 Algoritmus dynamického prideľovania kľúčových slov 24

6.1.1 Trénovacia časť 25

6.1.2 Testovacia časť 26

6.1.3 Experiment 27

6.2 Typy algoritmov 29

6.2.1 Algoritmus sumy 29

6.2.2 Algoritmus maxima 30

6.2.3 Algoritmus priemeru 30

6.2.4 Rozšírený algoritmus sumy 30

6.2.5 Rozšírený algoritmus Maximum 34

6.2.6 Rozšírený algoritmus priemer 35

6.3 Porovnanie Algoritmov 35

6.3.1 Porovnanie algoritmov sumy a maxima 36

6.3.2 Porovnanie algoritmov sumy a priemeru 40

6.3.3 Porovnanie algoritmov sumy a rozšírenej sumy 43

6.3.4 Porovnanie algoritmov rozšírenej sumy a rozšíreného maxima 46

6.3.5 Porovnanie algoritmov rozšírenej sumy a rozšíreného priemeru 49

6.4 Algoritmus podobnosti slov JNLA 51

6.5 Zhrnutie 55

7 Návrh **56**

7.1 Používateľské rozhranie 56

7.1.1 Zobrazenie záznamov 58

7.1.2 Pridávanie a editovanie záznamov 58

7.1.3 Pridávanie a editovanie kategórií 61

7.1.4 Detail záznamu 63
7.2 Architektúra 64
7.2.1 Backend 64
7.2.2 Frontend 65
7.3 Návrh Databázy 65

8 Implementácia **68**
8.1 Backend 68
8.2 Frontend 69
8.2.1 Komponent App 69
8.2.2 Komponent ListZaznam 69
8.2.3 Komponent CNFvyhladavanie 70
8.2.4 Komponent PridajZaznam 71
8.2.5 Komponent ZobrazFormular 72
8.2.6 Komponent KlucoveSlova 73
8.2.7 Komponent DetailZaznam 73
8.2.8 Komponent UpravKategorie 74
8.2.9 Spojenie 74

9 Nasadenie do prevádzky **75**
9.1 Produkčný režim 75
9.2 Vývojársky režim 76

10 Záver **77**

1

Úvod

Táto aplikácia je určená najmä pre výskumníkov, ale aj pre ľudí, ktorí vo svojej práci pracujú s veľkým množstvom údajov. Títo ľudia potrebujú tieto údaje vhodne organizovať a to tak, aby v nich mali prehľad a jednoducho sa vedeli dostať k tomu, čo v danej chvíli práve potrebujú. Preto je potrebné, aby mali tieto údaje na jednom mieste a aby boli dostupné kedykoľvek a odkiaľkoľvek.

Cieľom mojej diplomovej práce je vytvoriť webovú aplikáciu, ktorá zjednoduší prácu výskumníka. Táto aplikácia bude ukladať elektronické materiály na jedno miesto, aby sa k ním človek mohol dostať kedykoľvek to bude potrebovať. Ďalej bude uchovávať štrukturované údaje (projekty, knihy, pracovné cesty a pod.), ku ktorým bude možné pridávať rôzne prílohy, linky a komentáre. Na rozdiel od iných programov podporujúcich túto funkcionalitu, bude možné jednotlivé záznamy medzi sebou prepájať. Každý záznam bude možné prepojiť s hocijakým iným záznamom, vďaka čomu dokážeme jednotlivé údaje jednoduchšie nájsť. Taktiež bude naša aplikácia podporovať pridávanie jednotlivých záznamov do rôznych kategórií, pričom zoznam týchto kategórií nie je pevný a používateľ je schopný vytvoriť si vlastné ka-

tegórie. Medzi kľúčovú vlastnosť patrí vyhľadávanie. Daná aplikácia bude poskytovať fulltextové vyhľadávanie a vyhľadávanie podľa kľúčových slov. Taktiež bude automaticky navrhovať kľúčové slová na základe podobnosti záznamov.

1.1 Štruktúra práce

V prvej kapitole sa venujem základnému opisu mojej práce.

Druhá kapitola sa zaoberá základnými požiadavkami, ktoré musí daná aplikácia spĺňať.

Tretia kapitola popisuje predchádzajúce práce, ktoré sa zaoberali touto problematikou.

V štvrtej kapitole sa budem venovať existujúcim riešeniam. Aké sú ich výhody a nevýhody.

Piata kapitola popisuje jednotlivé technológie, ktoré sú použité na vývoj danej webovej aplikácie.

Šiesta kapitola sa venuje teórii a výskumu.

V siedmej kapitole sa budem venovať podrobnému návrhu aplikácie. Rozoberiem sem jednotlivé časti, z ktorých sa aplikácia skladá.

Osma kapitola opisuje postup implementácie. Rozoberám v nej ako fungujú jednotlivé komponenty.

Deviata kapitola sa bude zaoberať nasadením aplikácie do prevádzky.

2

Požiadavky

V tejto kapitole sa budem zaoberať požiadavkami, ktoré musí daná aplikácia spĺňať.

2.1 Používatelia

Cieľovou skupinou používateľov pre túto aplikáciu sú najmä ľudia, ktorí pracujú s veľkým množstvom informácií. Ktorí tieto informácie potrebujú vhodne organizovať, aby sa im s nimi lepšie pracovalo.

2.2 Funkčné požiadavky

Aplikácia eviduje informácie z rôznych kategórií, ktoré obsahujú záznamy. Kategórie nemajú pevne danú štruktúru, čo v praxi znamená, že je možné pridať, prípadne vymazať už z existujúcej kategórie nejaký atribút. Taktiež je možné zadefinovať si vlastnú kategóriu, pre špecifické potreby používateľa. Ku každému záznamu bez ohľadu na kategóriu bude môcť používateľ pridať nasledujúce položky.

- Komentáre
- Kľúčové slová
- Odkazy na iné záznamy
- URL linky
- Prílohy

2.2.1 Komentáre

Ku každému záznamu bude možné pridať ľubovoľne veľa komentárov (textových polí), kde si bude môcť používateľ zaznačiť svoje komentáre, ktoré bude môcť kedykoľvek jednoducho editovať.

2.2.2 Kľúčové slová

Ku každému záznamu bude mať používateľ možnosť priradiť kľúčové slová. Zoznam kľúčových slov si používateľ vytvára sám. Pri pridávaní záznamu si potom používateľ už len vyberie kľúčové slovo z už vytvoreného zoznamu kľúčových slov. Kľúčové slová budú tvoriť stromovú štruktúru, vďaka čomu budú dobre zadeliteľné do skupín a podskupín, čo umožní jednoduchšie vyhľadávanie.

Pri pridávaní záznamu budú kľúčové slová automaticky ponúkané používateľovi, na základe podobnosti jednotlivých záznamov.

2.2.3 Odkazy na iné záznamy

Táto vlastnosť patrí medzi najdôležitejšie vlastnosti tejto aplikácie. Umožňuje používateľovi prepojiť ľubovoľný záznam, s ľubovoľne veľa inými exis-

tujúcimi záznamami v aplikácii. Napríklad je takto možné prepojiť diplomovú prácu s posudkom, ktorý k nej prináleží. Všetky tieto odkazy pre daný záznam sa nám zobrazia pri prezeraní daného záznamu.

2.2.4 URL linky

Ku každému záznamu sa bude dať pripojiť ľubovolne veľa url liniek, na nami požadované webové stránky. Ku každej linke je možné pridať komentár.

2.2.5 Prílohy

Používateľ môže ku každému záznamu pridať ľubovoľný počet súborov. Typ súborov nie je obmedzený, čiže môže nahrávať dokumenty, fotky a rôzne iné súbory.

2.2.6 Zoznam uchovávaných kategórií

Tento zoznam obsahuje základne kategórie údajov, ktoré bude aplikácia uchovávať.

- Publikácie
- Projekty
- Posudky
- Pracovné cesty
- Konferencie
- Kontakty
- Knihy
- Evidencia majetku
- Podujatia
- Softvér
- Záverečné práce
- Spolupráca
- Rozpracované publikácie
- Weby

- Dokumenty
- Ostatné

2.2.7 Publikácie

Do tejto kategórie môžeme uložiť rôzne publikácie, pričom samotný záznam bude pozostávať z práve jednej publikácie. Publikáciu môžeme priložiť k záznamu ako samostatný súbor. Publikácia bude pozostávať z týchto údajov: názov, autori, typ , rok vydania, vydavateľstvo, číslo vydania, strany, doi, vyšlo v. Taktiež bude možné prelinkovať publikáciu napríklad s kontaktami, prípadne inými súvisiacimi záznamami.

2.2.8 Projekty

Táto kategória popisuje rôzne projekty. Základnými typmi údajov, ktoré patria k tejto kategórii sú: názov, hlavný riešiteľ, popis projektu, ostatní riešitelia, dátum začiatku projektu, dátum konca projektu, typ projektu, stav projektu. Termíny týkajúce sa projektu sa budú môcť pridať do kalendára.

2.2.9 Posudky

Do tejto kategórie patria posudky k projektom, diplomovým, bakalárskym prácam, článkom a pod. Medzi údaje patriace do tejto kategórie patrí názov, autor, typ, udalosť. Posudok sa potom môže prepojiť so záznamom, ku ktorému prislúcha. Taktiež je možné nahrať posudok v elektronickej podobe súboru napríklad typu pdf.

2.2.10 Pracovné cesty

Tu bude môcť používateľ pridávať údaje o svojej plánovanej pracovnej ceste. Záznam bude uchovávať: udalosť, miesto, dátum začiatku a dátum konca pracovnej cesty, zdroj financovania, účastníci. K týmto záznamom môže používateľ pridať ľubovoľné súbory súvisiace s danou pracovnou cestou ako cestovné lístky, podklady, fotografie a pod.

2.2.11 Konferencie

V rámci tejto kategórie bude môcť používateľ uchovávať záznamy o plánovaných, ako aj už ukončených konferenciách. Záznamy budú obsahovať názov konferencie, dátum jej začiatku a konca, miesto, dátum poslania abstraktu, dátum akceptácie, organizátora.

2.2.12 Kontakty

Medzi jednu z veľmi dôležitých vecí patria aj kontakty, s ktorými daný používateľ spolupracuje, prípadne inak interaguje. Jednotlivé kontakty budú pozostávať z nasledujúcich údajov: meno, priezvisko, adresa, e-mail, telefón, pracovná pozícia, pôsobisko, vznik kontaktu. Taktiež bude možné pridať k záznamu fotografiu, prípadne rôzne iné prílohy.

2.2.13 Knihy

V tejto kategórii si bude môcť používateľ uchovávať záznamy o knihách. Záznam bude pozostávať z nasledujúcich údajov: názov, autori, vydavateľstvo, číslo vydania, rok vydania, dátum nadobudnutia, požičané od koho komu, umiestenie, stav(moja, požičaná niekomu, požičaná od niekoho, stratená, želaná).

2.2.14 Evidencia majetku

Táto kategória slúži na prehľadné ukladanie majetku. Používateľ si tu môže ukladať veci, ktoré vlastní prípadne by chcel vlastniť. Táto kategória bude uchovávať nasledujúce údaje: názov, cena, mena, cena v eurách, dátum nadobudnutia, účel nadobudnutia, zdroj financovania, umiestnenie, miesto v miestnosti. Zvyšné veci môže používateľ priložiť vo forme prílohy.

2.2.15 Podujatia

V rámci tejto kategórie sa budú uchovávať záznamy o jednotlivých podujatiach, o ktoré by mohol mať používateľ záujem. Záznamy budú obsahovať názov podujatia, popis, typ podujatia, dátum začiatku a konca daného podujatia, skratku, adresu podujatia, zdroj financovania, počet ľudí, meno kontaktnej osoby prípadne kontakt na kontaktnú osobu. Tieto podujatia bude možné taktiež pridať do kalendára, kde bude môcť používateľ prehľadne vidieť všetky podujatia, ktoré bude systém uchovávať.

2.2.16 Softvér

Táto kategória bude uchovávať informácie o softvéri, ktorý daný používateľ používa. Bude obsahovať názov produktu, verziu, licenciu produktu, cenu, výrobcu, popis produktu.

2.2.17 Záverečné práce

Do tejto kategórie budú patriť všetky typy záverečných prác. Kategória bude uchovávať nasledujúce údaje: názov práce, predpokladaný rok ukončenia práce, cieľ práce, študenta, ktorému bola práca pridelená, typ práce, moju rolu v danej práci(školiteľ, oponent).

2.2.18 Spolupráca

V tejto kategórii budeme uchovávať údaje o firmách, organizáciach, s ktorými používateľ spolupracuje, prípadne má s nimi podpísanú nejakú zmluvu. Záznam bude pozostávať z týchto údajov: názov spoločnosti, kontaktná osoba, telefón, e-mail, zmluva, dátum exspirácie zmluvy, adresa sídla, bankový kontakt, ičo.

2.2.19 Rozpracované publikácie

V rámci tejto kategórie bude možné ukladať publikácie, ktoré ešte nie sú dokončené a treba na nich ešte pracovať. Záznam bude tvoriť názov, abstrakt, cieľové podujatie.

2.2.20 Weby

Tu bude môcť používateľ ukladať webové stránky, servery, ktoré používateľ spravuje, prípadne k nim má nejaký iný vzťah. Medzi údaje patriace k tejto kategórii patria: názov, linka na stránku, meno servera, kontakt na používateľa, kontakt na autora, účel, softvérové požiadavky, automatický backup.

2.2.21 Dokumenty

Táto kategória slúži na ukladanie rôznych dokumentov, ktoré sú pre používateľa dôležité. Najmä takých, ku ktorým potrebuje mať rýchly prístup. Záznam bude obsahovať názov dokumentu, popis dokumentu a dátum.

2.2.22 Ostatné

Okrem už spomínaných kategórií si bude môcť používateľ vytvoriť aj vlastné kategórie. Vďaka tomu si bude môcť aplikáciu prispôsobiť podľa seba a ucho-

vávať záznamy podľa svojej potreby.

2.3 Základné charakteristiky systému

2.3.1 Prístup, jednoduchosť, rozšíriteľnosť

Táto aplikácia je určená pre ľudí pracujúcich s veľkým množstvom dát. Slúži najmä nato, aby používateľovi zjednodušila jeho prácu. Preto je dôležité, aby mal používateľ prístup k svojím dátam kedykoľvek a kdekoľvek. Z toho dôvodu je najlepšie riešenie vytvoriť webovú aplikáciu, ktorú budú môcť používatelia používať prostredníctvom internetu. Predíde sa tým komplikáciám, kedy by bolo nutné aplikáciu inštalovať na každé zariadenie, na ktorom by ju chcel používateľ použiť. Taktiež je treba myslieť aj na efektívnosť. V tomto smere je nutné minimalizovať komunikáciu medzi klientom a serverom, aby aplikácia fungovala plynule aj pri pomalšom pripojení na internet.

Medzi kľúčové vlastnosti patrí aj dizajn, ktorý by mal byť čo najintuitívnejší. Aby používateľ vedel pracovať s danou aplikáciou bez potreby vysvetlenia.

Ďalej by mala byť aplikácia rozšíriteľná, teda aby sa do nej dali jednoducho pridať prípadne odobrať niektoré funkcionality.

2.3.2 Previazanosť záznamov

Táto vlasnosť patrí medzi najdôležitejšie vlastnosti aplikácie. Umožňuje používateľovi prepojiť ľubovoľný záznam s inými existujúcimi záznamami v aplikácii. Vďaka tomu bude mať používateľ lepší prehľad o tom, ako jednotlivé záznamy spolu súvisia.

2.3.3 Kalendár

Jednou z podstatných častí našej aplikácie je aj kalendár. Kalendár bude slúžiť pre lepšiu orientáciu medzi jednotlivými udalosťami, termínmi projektov, konferencií a pod. Do kalendára budeme môcť pridávať záznamy z jednotlivých kategórií, ktoré obsahujú dátumy.

2.3.4 Vyhľadávanie

Aby používateľ vedel nájsť v aplikácii konkrétny záznam mu pomôže vyhľadávanie. Aplikácia bude ponúkať dva druhy vyhľadávania: fulltextové vyhľadávanie, vyhľadávanie pomocou kľúčových slov. Fulltextové vyhľadávanie bude vyhľadávať záznamy v celej aplikácii. Vyhľadávanie pomocou kľúčových slov vyhľadá len také záznamy, ktoré obsahujú dané kľúčové slovo.

2.3.5 Dynamické prideľovanie kľúčových slov

Aby používateľ nemusel zakaždým pri pridávaní kľúčového slova k záznamu hľadať, aké kľúčové slovo by sa k danému záznamu najviac hodilo. Bude mať možnosť vybrať si z určitého počtu slov, ktoré mu ponúkne algoritmus. Počet slov, ktoré algoritmus ponúka bude môcť používateľ nastaviť. Algoritmus na základe podobnosti práve pridávaného záznamu so záznamami uloženými v aplikácii odporučí vhodné kľúčové slová, ktoré používateľ môže, ale nemusí k danému záznamu pridať. Algoritmus bude odporúčať kľúčové slová na základe slov, ktoré bude obsahovať práve pridávaný záznam. Tieto slová bude porovnávať so slovami, ktoré sa už nachádzajú v databáze, na základe nastavenia odporúčania. Odporúčanie môžeme nastaviť nasledujúcimi spôsobmi.

- Všetky kategórie, všetky atribúty

Pri tomto nastavení sa bude hľadať zhoda s tým čo používateľ zadal, so všetkými slovami všetkých záznamov bez ohľadu na kategóriu alebo atribúty.

- Konkrétna kategória, všetky atribúty

  V tomto prípade sa budú porovnávať zadané slová len so slovami, ktoré obsahujú záznamy nastavenej kategórie.

- Konkrétna kategória, konkrétny atribút

  Ak si používateľ zvolí takéto nastavenie. Tak algoritmus bude hľadať zhodu len so slovami, ktoré sa nachádzajú v nastavenej kategórii a nastavenom atribúte.

3

Predchádzajúce práce

Predchádzajúce práce venujúce sa tejto téme robili predo mnou aj Bc. Miroslav Gregorec [Gre17] a Katarína Matysová [Mat11].

V práci [Mat11] bol použitý framework Google web Toolkit, ide o framework pre vytváranie webových aplikácii. V tomto frameworku je využívaná technológia Ajax, ktorá slúži na komunikáciu medzi klientom a serverom. Program bol písaný v programovacom jazyku Java. Aplikácia využíva technológiu Mysql pre prácu s databázou. Na prácu s databázou je použitá aj technológia Hibernate.

V tejto práci bol použitý veľmi pekný a intuitívny dizajn aplikácie.

Medzi nevýhody by som radil najmä komplikovanosť databázy. V prípade, že by sme chceli pridať do aplikácie nejakú novú kategóriu museli by sme meniť štruktúru databázy. Teda aplikácia pracuje len s obmedzenými vopred definovanými kategóriami a neumožňuje používateľovi vytvoriť novú kategóriu podľa jeho potrieb.

V tejto práci [Gre17] narozdiel od predchádzajúcej bol použitý framework Express. Tento framework funguje na základe Node.js, ktorý umožňuje spracovávať viacero požiadaviek v jednom vlákne, tým pádom je odolný voči zahlteniu. Node.js taktiež efektívne spravuje pamäť a aj vďaka tomu poskytuje rýchlu odozvu pre viacero používateľov súčasne. Aplikácia bola naprogramovaná v javascriptovom frameworku Angular, ktorý umožňuje jednoduchú spoluprácu s Node.js. V rámci databázovej časti aplikácie, bola použitá taktiež technológia Mysql.

Táto aplikácia má veľmi dobré rozmiestnenie komponentov a má dobre organizovaný dizajn, čo môže byť pre nás inšpiráciou pri tvorbe našej aplikácie. Taktiež má táto práca veľmi dobre navrhnutý databázový model. Databáza obsahuje jednu tabuľku, ktorá obsahuje všetky záznamy spolu so všetkými atribútmi. V našej aplikácii budeme používať podobnú štruktúru databázy ako v tejto práci.

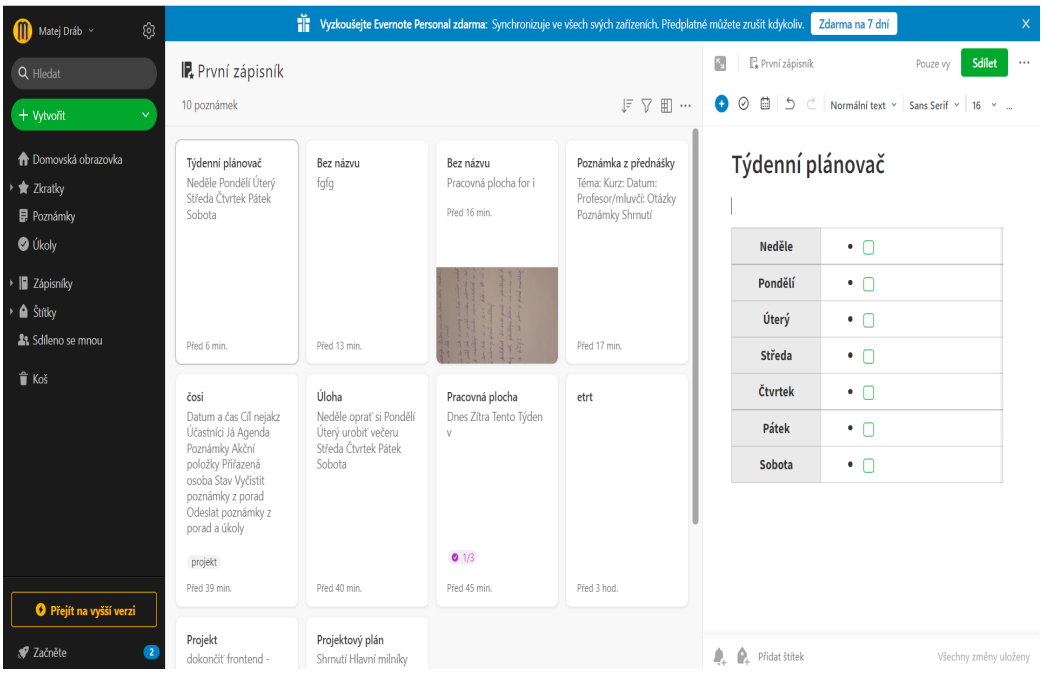
Jednou z nevýhod je to, že fulltextové vyhľadávanie funguje iba pri presnej zhode. Pričom v práci je uvádzané, že by malo fungovať aj pri miernych chybách v slovách.

4

Existujúce riešenia

4.1 Evernote

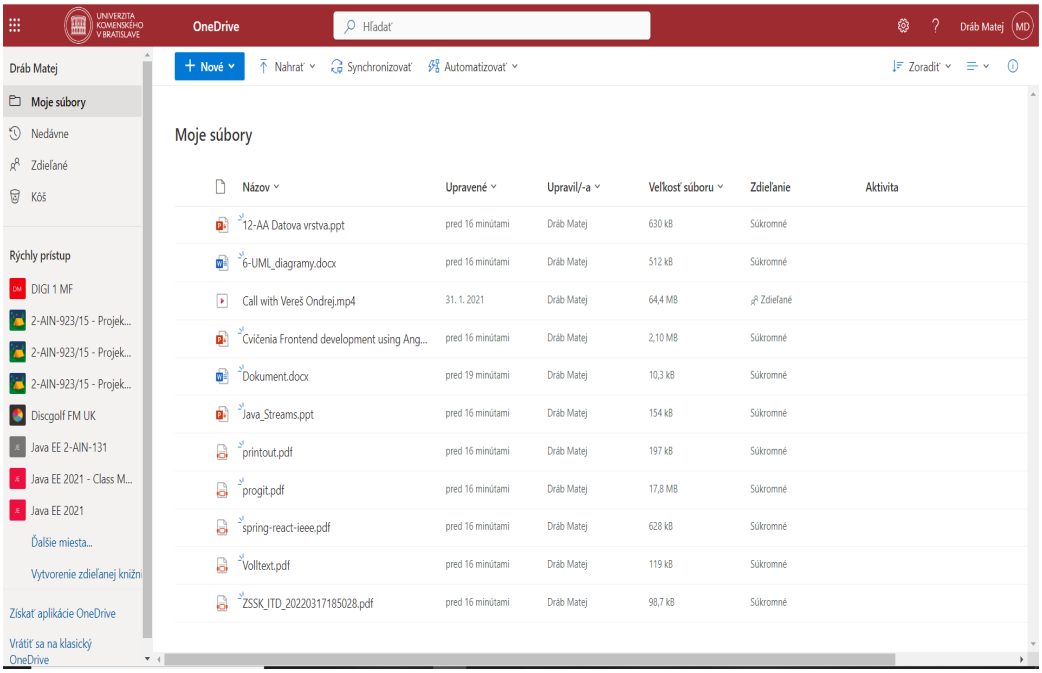
Je webová aplikácia, ktorá umožňuje uchovávať poznámky. K poznámkam môže používateľ pripojiť rôzne prílohy ako sú fotky, videá, webové stránky a pod. Umožňuje taktiež pridávať k jednotlivým poznámkam kľúčové slová a pripomienky. Okrem toho je možné ďalej tieto poznámky zdielať. Takisto ako v našej aplikácii umožňuje aj fulltextové vyhľadávanie aj vyhľadávanie pomocou kľúčových slov. K poznámkam sa dajú prikladať odkazy na iné záznamy. Medzi výhody patrí aj to, že je to webová aplikácia, teda používateľovi stačí len pripojenie k internetu aby mohol pracovať. Čo sa týka kategórií, tie nie sú nijako pevne dané, používateľ si môže vytvárať zápisníky, ktoré budú obsahovať poznámky a je len na ňom akým spôsobom si bude svoje poznámky kategorizovať. Pri vytváraní poznámky je možné vybrať si jednu z množstva šablón, ktoré nám už dopredu naformátujú danú poznámku podľa našich potrieb. Taktiež je možné vytvoriť si vlastnú šablónu. Medzi nevýhody tejto aplikácie patrí najmä to, že je platená a väčšina funkcionalít sa nachádza práve v platenej verzii. Napríklad kalendár a funkcionality s ním spojené



Obrázok. 4.1: Ukážka programu Evernote zdroj: `https://evernote.com/intl/cs`

4.2 Outlook

OneDrive je jedna z mnohým webových služieb, ktoré ponúka microsoft Office. Táto služba ponúka možnosť ukladať si na web rôzne typy súborov, ktoré je možné ďalej zdielať s inými ľuďmi. V rámci inej služby od microsoftu je možné vytvárať poznámky a taktiež existuje služba na správu kalendára. Nevýhodou však je to, že tieto služby spolu nedokážu spolupracovať. Teda k súborom uloženým vo OneDrive nie je možné pridávať poznámky a takisto k poznámkam nie je možné pridávať súbory. Ďalšou nevýhodou je to, že používateľovi nie je umožnené vytvárať kľúčové slová a záznamy nie je možné kategorizovať

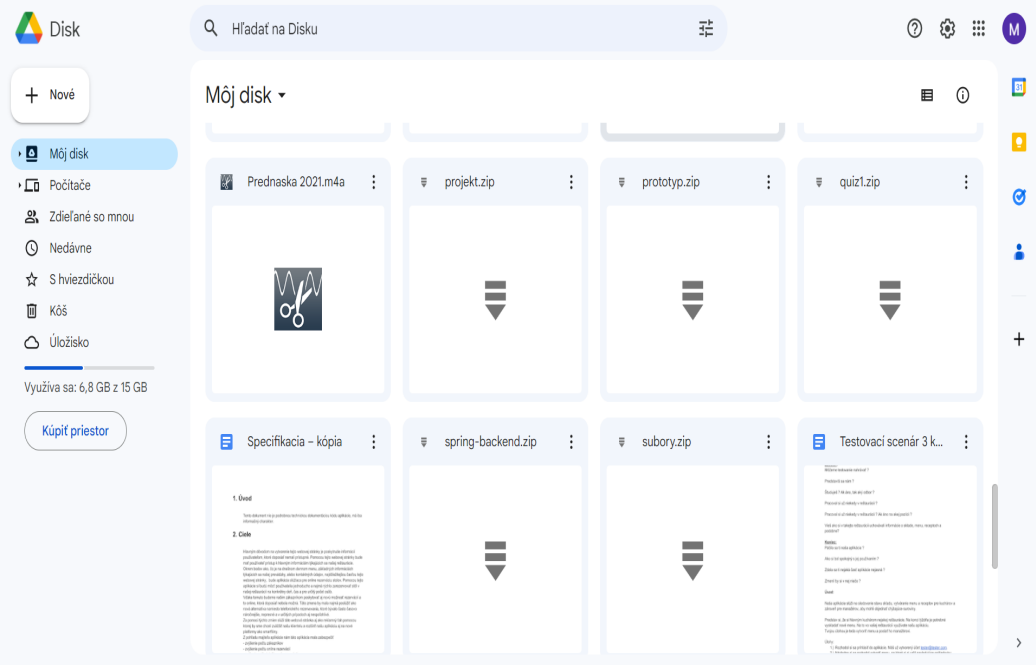


Obrázok. 4.2: Ukážka programu OneDrive zdroj: `https://www.microsoft.com/sk-sk/microsoft-365/onedrive/online-cloud-storage`

4.3 Google Drive

Google Drive je webová služba od spoločnosti Google umožňujúca svojim používateľom ukladať rôzne typy súborov. Tieto súbory je možné ďalej zdieľať s inými ľudmi. Výhodou je to, že ak správne nastavíme skupinu ľudí, s ktorou chceme niečo zdieľať. Viacerí ľudia majú prístup k danému súboru. Teda umožňuje viacerým ľudom naraz editovať zdieľaný súbor. Táto vec je výhodná pri riešení skupinových projektov a podobných vecí. Taktiež Google ponúka službu kalendára, do ktorej je možné pridávať udalosti. Obsahuje aj možnosť fulltextového vyhľadávania Medzi nevýhody patrí najmä to, že sa tu nedajú vytvárať kategórie. Takisto nie je možné zadefinovať kľúčové slová prípadne si pridať poznámku k nejakému súboru. Okrem toho nie je možné

záznamy medzi sebou previazať.



Obrázok. 4.3: Ukážka programu GoogleDrive zdroj: `https://www.google.com/drive`

5

Technológie

V tejto kapitole rozoberiem jednotlivé technológie použité na vytvorenie výslednej aplikácie.

5.1 React.js

React je javascriptový framework pre tvorbu používateľského rozhrania. Používa sa najmä kvôli svojej jednoduchosti a efektívnosti. Efektívny je v tom, že zabezpečuje, aby sa stránka nerenderovala vždy celá. Ale aby sa zmenili len tie časti stránky, pri ktorých došlo k zmene dát. Vďaka tejto vlastnosti sa stránky, na ktorých je táto technológia použitá renderujú bez zbytočného čakania.

Hlavnou ideou Reactu je rozdelenie stránky na jednotlivé komponenty, ktoré sú v určitej hierarchii. Každý komponent má svoj stav takzvaný "state". Nato aby sa tento stav zmenil je potrebné zavolať metódu setState(). Pri každom zavolaní tejto metódy sa zmení stav komponentu a následne sa rerenderuje daný komponent. Komponenty sú v hierarchii, kde komponent, ktorý je na čele hierarchie vie posielať svojím potomkom dáta cez takzvané

"props". To je objekt, ktorý môže obsahovať viacero rôznych dát. Takýmto spôsobom vieme poslať dáta z jedného komponentu do ďalších komponentov, čím zabezpečujeme komunikáciu medzi jednotlivými komponentami.

5.2 Node.js

Node js je open source multiplatformový javascritový framework. Často sa používa, ako serverová časť pri tvorbe webových aplikácií. Medzi jeho hlavné výhody patrí jeho asynchrónna event-driven architektúra, ktorá je neblokujúca. Je vhodný na tvorbu aplikácií v reálnom čase. To znamená, že ak niečo zmeníme v našom kóde, automaticky sa to prejaví aj na frontende, ak využívame node.js.

V tejto práci slúži node js len ako development server pre frontendovú časť našej aplikácie, ktorá je programovaná prostredníctvom Reactu. Teda inými slovami potrebujeme ho len pri vývoji našej aplikácie.

5.3 Axios

Axios je promised based http klient pre prehliadač a node.js. Používa sa najmä na komunikáciu s backendom prostredníctvom REST API. Medzi jeho výhody patrí to, že obsahuje funkcie na spracovanie http požiadaviek. Napríklad ak chceme niečo poslať na server, stačí zadať príkaz axios.post(url,data). V ktorom vyplníme príslušnú url, pridáme dáta a o zvyšné veci sa už nemusíme starať. Okrem toho požiadavky, ktoré obhospodarúva sa automaticky pre konvertujú na JSON.

5.4 Spring Boot

Spring Boot je open source javový framework, ktorý slúži na vytváranie samostatných webových aplikácií. Medzi jeho hlavné výhody patrí to, že zjednodušuje vývoj webových aplikácií. Tým, že poskytuje jednoduchý spôsob automatickej konfigurácie. To v praxi znamená, že tento framework za nás prednastaví nastavenia a nám už stačí len spustiť danú aplikáciu. Tým pádom sa nemusíme zaoberať žiadnym zložitým nastavovaním.

Taktiež má v sebe vstavaný aplikačný server Apache Tomcat, ktorý štandardne beží na porte 8080. Samozrejme toto nastavenie môžeme kedykoľvek zmeniť. Tento aplikačný server sa zapne vždy keď spustíme aplikáciu (ak nenastavíme inak). Vďaka tomuto serveru sa nemusíme spoliehať na externý server. To nám zaručuje, že aplikáciu môžeme bez problémov spustiť na akejkoľvek platforme. Toto nám tiež uľahčuje to, že na mieste, na ktorom chceme spustiť našu aplikáciu nemusíme konfigurovať serverovú časť. Tá je už vďaka tomuto serveru prispôsobená našej aplikácii.

Medzi jednu z výhod tiež patrí to, že celú aplikáciu môžeme jednoducho zabaliť do jedného súboru typu jar. Tento súbor už potom jednoducho spustíme a naša aplikácia bude bežať. Okrem iného Spring-boot podporuje tiež prácu s rôznymi inými rozšíreniami. Napríklad pre prácu s databázou môžeme využiť balíček Spring Data JPA.

5.5 Mysql

Mysql je voľne šíriteľný viacúčelový relačný systém. Medzi jeho hlavné výhody patrí najmä dostupnosť na viacerých platformách. Pre prácu s databázou používa jazyk sql.

5.6 Spring Data

SpringData je framework v jazyku java určený na podporu ORM objektovo relačného mapovania. ORM je metóda, ktorá slúži nato, aby objekty naprogramované v jazyku java boli namapované na tabuľky v databáze. Inak povedané objekty v jave budú predstavovať záznamy v databáze. Na konverziu týchto objektov do databázy sa používa technika ORM. Táto technika sa môže použiť dvoma spôsobmi. Prvým na základe anotácií. Triede, ku ktorej chceme spraviť tabuľku pridáme priamo do kódu anotáciu *@Entity* a jej atribútom anotáciu *@Column*. Druhý spôsob je vytvorenie samostatného XML súboru, kde budú informácie o každej triede ako má byť namapovaná na databázu. Veľkou výhodou používania ORM je to, že pri zmene štruktúry databázy nemusíme meniť samotnú databázu. Taktiež nemusíme meniť kód a to ako k dátam pristupujeme. V prípade, že chceme k nejakej entite pridať nejaký stĺpec. Stačí pridať atribút do triedy a dať mu príslušnú anotáciu. Následne po spustení kódu sa atribút k danej tabuľke pridá, pričom všetky dáta, ktoré už daná tabuľka obsahuje ostanú nezmenené.

Medzi ďalšie výhody patrí to, že ak máme pri niektorých objektoch anotácie typu, *@OneToMany* alebo *@ManyToMany* a podobne. Nemusíme ručne hľadať dané záznamy. Stačí nám vybrať jeden konkrétny záznam a spolu s ním nám budú vrátené aj všetky ostatné, ktoré s ním nejako súvisia. Taktiež veľmi veľkou výhodou je to, že programátor nemusí ručne písať dopyty na databázu. Čím sa znižuje šanca písania chybných dopytov.

Na prácu s dátami si však musíme ku každej triede vytvoriť vlastný interface ktorý bude dediť od triedy JPARepository a pridať mu anotáciu *@Repository*.

Následne pri pridávaní záznamu nám stačí zavolať len metódu tohto interfacu. V prípade, že by sme si chceli urobiť vlastný dopyt na databázu.

Tak musíme do tohto nášho interfacu pridať metódu, ktorá môže obsahovať parametre, ktoré budú dosadené do daného dopytu. Tento dopyt môžeme písať v jazyku JPQL alebo SQL. V prípade jazyka SQL musíme pridať do našej anotácie aj položku nativeQuery nastavenú na true.

```
@Query(value = "SELECT * FROM zaznamy z where" +
        " z.id_kategorie =?1", nativeQuery = true)
List<Zaznam> getZaznamKategoria(Long id);
```

Obrázok. 5.1: Ukážka query

6

Výskum

6.1 Algoritmus dynamického prideľovania kľúčových slov

V tejto práci je jednou z našich výskumným tém navrhnúť vhodný algoritmus, ktorý bude automaticky ponúkať používateľovi kľúčové slová na základe podobnosti jednotlivých záznamov. V nasledujúcej stati budeme popisovať experimenty, v ktorých budeme skúmať jednotlivé algoritmy. Tieto algoritmy budeme medzi sebou porovnávať a zisťovať, ktorý z nich dokáže najlepšie odporúčať kľúčové slová. V jednotlivých experimentoch budeme merať celkové počty úspešne odporúčaných slov pre konkrétnu množinu záznamov a priemerný počet úspešne odporúčaných slov na záznam. Výsledky budeme vyhodnocovať na základe prieniku množiny odporúčaných slov s množinou správnych kľúčových slov, pretože toto najviac zodpovedá zvýšeniu komfortu používateľa.

6.1.1 Trénovacia časť

Na vstupe bude istý počet záznamov, ktoré už budú mať pridelené svoje kľúčové slová. Základ algoritmu spočíva v tom rozdeliť atribúty záznamov, ktoré už sú v databáze na slová. Pre tieto slová urobíme karteziánsky súčin medzi množinou slov, ktoré sme dostali zo záznamu a množinou kľúčových slov, ktoré boli priradené k danému záznamu. Vytvoríme tak dvojicu (slovo, kľúčové) Záznam je teda reprezentovaný ako dvojica (W, KW), kde W je k-tica (w_1, w_2, ... , w_k) označujúca všetky neignorované slová vyskytujúce sa v zázname a KW je n-tica (kw_1, kw_2, ... , $_n$) označujúca všetky kľúčové slová daného záznamu. Pri spracovaní záznamu budeme vždy uvažovať všetky dvojice (w_i, kw_j), pre i = 1..k, j = 1..n.

Následne pridáme dvojicu do frekvenčnej tabuľky. Táto tabuľka bude obsahovať slovo zo záznamu, kľúčové slovo a frekvenciu výskytu. Ak sa daná dvojica ešte nenachádza v tabuľke, tak ju tam pridáme a nastavíme frekvenciu výskytu na 1. V prípade, že už sa daná dvojica v tabuľke nachádza, tak zvýšime frekvenciu o 1.

Nasledujúca tabuľka opisuje príklad kedy máme v databáze dva nasledujúce záznamy, ktorých slová sú uchované v tejto tabuľke:

Záznam: Pracovná plocha výskumníka Kľúčové slová: systém, web
Záznam: Pracovná webová stránka Kľúčové slová: web

Tabuľka 6.1: Tabuľka výskytov dvojíc (slovo, kľúčové slovo)

Slovo	**Kľúčové slovo**	**Frekvencia výskytu**
pracovná	web	2
pracovná	systém	1
plocha	web	1
plocha	systém	1
výskumníka	web	1
výskumníka	systém	1
webová	web	1
stránka	web	1

6.1.2 Testovacia časť

V tejto časti príde na vstup nový záznam, ktorý ešte nemá určené kľúčové slová. Následne rozdelíme obsah záznamu na slová. Potom postupne budeme prechádzať jednotlivými slovami nadpisu a pozrieme sa, či sa už dané slovo nenachádza v tabuľke výskytov. Ak áno, tak sa pozrieme nato, aké kľúčové slová boli k danému slovu priradené. Tieto slová spolu s ich frekvenciou výskytu pridáme do novej tabuľky odporúčaných slov (táto tabuľka bude samostatná pre každý záznam), ktorá bude obsahovať (kľúčové slovo, skóre). Položka skóre popisuje, ako vhodné je dané kľúčové slovo pre konkrétny záznam. Následne prejdeme na ďalšie slovo, ak nastane prípad, kedy sa kľúčové slovo už nachádza v tabuľke odporúčaných slov, tak upravíme skóre. Každý algoritmus bude upravovať skóre iným špecifickým spôsobom. Po prejdení všetkých slov v nadpise vyberieme určitý počet kľúčových slov, ktoré mali najvyššie skóre. Tieto slová následne odporučíme používateľovi.

Nasledujúca tabuľka zobrazuje odporúčané slová pre nasledovný záznam v prípade použitia algoritmu sumy, ktorý je popísaný nižšie.

Záznam: Pracovná webová stránka pre školských učiteľov

Tabuľka 6.2: Tabuľka odporúčaných slov

Kľúčové slovo	**Skóre**
web	4
systém	1

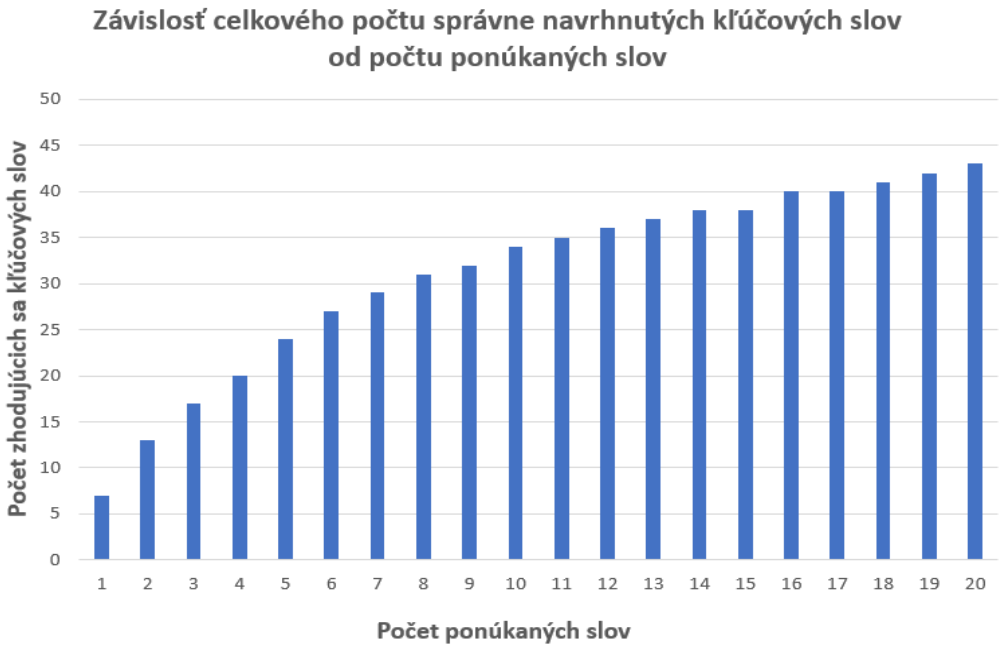
6.1.3 Experiment

V tejto časti popíšeme experiment na datasete, ktorý obsahuje 120 záznamov. Jedná sa o reálny dataset (nebol vytvorený umelo/strojovo), ktorý poskytol cieľový používateľ navrhovaného systému. Obsahuje záznamy o odborných článkoch, ktoré si používateľ v tomto systéme chce evidovať so svojimi poznámkami k týmto článkom. Články pokrývajú niekoľko rôznych oblastí, ku každému je priradených 2-8 kľúčových slov, v priemere 4.53 kľúčových slov/záznam, celkovo je použitých 159 rôznych kľúčových slov. Tento dataset je zverejnený na githube `https://github.com/drab12/diplomovaPraca` a v elektronickej prílohe tejto práce.

Na tomto datasete už priradenými kľúčovými slovami vykonáme nasledujúci experiment. Na začiatku tieto záznamy náhodne zamiešame. Potom pre prvých 100 záznamov spustíme trénovaciu časť algoritmu. Na zvyšných 20 záznamoch spustíme testovaciu časť algoritmu a porovnáme či slová, ktoré náš algoritmus odporúčal, sa zhodujú s kľúčovými slovami priradenými k daným záznamom. Tento proces budeme robiť 200-krát pre určitý počet ponúkaných slov, aby sme získali priemerný celkový počet odporúčaných slov, ktoré sa zhodujú s kľúčovými slovami daných záznamov v testovacej časti, pre daný počet ponúkaných slov. Následne zopakujeme tento proces 30 krát pre rôzny počet ponúkaných slov

Graf na obrázku 6.1 zobrazuje priemerný celkový počet odporúčaných slov pre 20 záznamov, ktoré sa zhodujú s kľúčovými slovami pre rôzny počet

ponúkaných slov.



Obrázok. 6.1: Graf závislosti vhodne odporúčaných slov od počtu ponúkaných slov

Na grafe môžeme pozorovať viacero trendov:

1. Počet slov, ktoré algoritmus vybral a ktoré sa zhodujú s kľúčovými slovami priradenými k záznamom, sa zvyšuje s počtom ponúkaných slov. Tento trend je prirodzene očakávaný, opak by znamenal, že od určitého počtu K by všetky ďalšie ponúkané slová boli celkom nesprávne. To by sa mohlo stať buď vtedy, keby algoritmus dokázal presne vybrať všetky správne kľúčové slová (čo je nereálne splniť), alebo keby nepracoval správne. Čím viac slov teda aplikácia bude používateľovi ponúkať, tým viac mu uľahčí označovanie kľúčovými slovami, lebo ich nebude musieť vyberať z neprehľadného zoznamu všetkých kľúčových slov. Dĺžka tohto zoznamu by ale mala byť limitovaná, pretože od určí-

tej dĺžky bude zoznam priveľmi zaťažovať používateľa a nútiť ho zvažovať aj množstvo takých slov, ktoré nie sú vhodné. Napr. v prípade 11 ponúkaných kľúčových slov bol počet správnych rovný 35 zo všetkých potenciálne až 20 * 11 = 220 slov. Je teda vhodné zvoliť určitý kompromis.

2. Počet správnych slov rastie spočiatku pomerne prudko a takmer lineárne, takže algoritmu sa dobre darí určiť vhodné slová a teda má zmysel, aby aplikácia poskytla okolo 6-8 slov, kedy sa trend láme. Zlom je spôsobený vlastnosťami datasetu: napríklad počtom kľúčových slov, ktoré sú k záznamom priradené a mierou podobnosti priradenia kľúčových slov v datasete - nakoľko sa vyskytujú ojedinelé prípady neužitočných slov, ktoré neboli ignorované a nachádzajú sa v daných záznamoch.

6.2 Typy algoritmov

V tejto časti sa lepšie pozrieme na jednotlivé typy algoritmov ktoré sme testovali.

6.2.1 Algoritmus sumy

Tento algoritmus najskôr v trénovacej časti vytvorí a naplní tabuľku výskytov podľa vyššie spomínaného postupu. Následne prejdeme do testovacej časti. V tejto časti budeme vytvárať tabuľky odporúčaných slov pre jednotlivé záznamy v testovacej časti. Každý záznam bude obsahovať vlastnú tabuľku odporúčaných slov. Tabuľku budeme vytvárať tak, že budeme prechádzať slová novo pridávaného záznamu. Ak sa nejaké z týchto slov nachádza v tabuľke výskytov, tak všetky jeho kľúčové slová pridáme do tabuľky odporúčaných slov nasledovným spôsobom. Ak sa kľúčové slovo ešte nenachádza v tabuľke

odporúčaných slov, tak ho tam pridáme spolu s jeho priradenou frekvenciou. V prípade, že už sa kľúčové slovo v tabuľke nachádza tak k jeho frekvencii pripočítame frekvenciu z tabuľky výskytov.

6.2.2 Algoritmus maxima

Algoritmus maxima je takmer identický ako algoritmus sumy, s tým rozdielom, že v prípade, že sa už kľúčové slovo nachádza v tabuľke odporúčaných slov, tak sa pozrieme nato, či je jeho frekvencia priradená v tabuľke výskytov a nie je väčšia ako frekvencia v tabuľke odporúčaných kľúčových slov. Ak áno, tak potom túto frekvenciu prepíšeme.

6.2.3 Algoritmus priemeru

V tomto algoritme budeme postupovať identicky ako v algoritme sumy, s tým rozdielom, že si budeme v pomocnej premennej uchovávať pre každé kľúčové slovo počet koľkokrát bolo zmenené v tabuľke odporúčaných slov. Následne, keď prejdeme všetky slová, frekvenciu každého slova z tabuľky odporúčaných slov podelíme príslušným počtom. Takýmto spôsobom dostaneme priemernú frekvenciu.

6.2.4 Rozšírený algoritmus sumy

Trénovacia časť

V tomto algoritme budeme mať o trochu inú trénovaciu časť. Rozdiel bude v tom, že tabuľka výskytov bude obsahovať namiesto frekvencie, pozitívnu a negatívnu frekvenciu. Pozitívna frekvencia obsahuje informáciu o tom koľkokrát sa v našom datasete vyskytlo dané slovo spolu s daným kľúčovým

slovom. Negatívna frekvencia vyjadruje informáciu o tom koľkokrát sa vyskytlo dané slovo bez daného kľúčového slova. Tým pádom pri pridávaní nového slova si budeme uchovávať okrem dvojice slovo, kľúčové slovo aj dva ďalšie atribúty pre frekvenciu. Teda pridávanie ďalšieho slova do tabuľky výskytov bude vyzerať nasledovne.

Pre každú dvojicu (w_i, kw_j) môže nastať jeden z týchto prípadov:

1. w_i ani kw_j sa v tabuľke výskytov ešte nenachádzajú
2. w_i sa v tabuľke ešte nenachádza v žiadnej dvojici, ale kw_j áno
3. kw_j sa v tabuľke ešte nenachádza v žiadnej dvojici, ale w_i áno
4. aj w_i aj kw_j sa v tabuľke nachádzajú, ale dvojica (w_i, kw_j) sa tam ešte nenachádza
5. dvojica (w_i, kw_j) sa v tabuľke už nachádza

V jednotlivých prípadoch budeme postupovať takto:

- V prípadoch 1 a 2: do tabuľky pridáme nový riadok (slovo, kľúčové slovo, $F_{positive}$, $F_{negative}$) = (w_i, kw_j, 1, 0)
- Prípady 3, 4: do tabuľky pridáme nový riadok (slovo, kľúčové slovo, $F_{positive}$, $F_{negative}$) = (w_i, kw_j, 1, M) pričom M je súčet $F_{positive}$ + $F_{negative}$ z ľubovoľného už existujúceho riadku v tabuľke, kde sa nachádza w_i, ale berieme do úvahy hodnoty, aké tam boli pred aktualizáciou na základe tohto záznamu.
- Prípad 5. v riadku (w_i, kw_j, $F_{positive}$, $F_{negative}$) zvýšime $F_{positive}$ o 1 a vykonáme krok UPDATE(w_i) opísaný nižšie

- UPDATE(w_i) vo všetkých riadkoch z tabuľky vo formáte (w_i, kw_t, $F_{positive}$, $F_{negative}$): ak kw_t nie je medzi (kw_1, ... , kw_n), tak $F_{negative}$ zvýšime o 1

V prípade, že sa slovo nenachádza v tabuľke výskytov, tak pridáme slovo do tabuľky spolu s kľúčovými slovami, ktoré sa viažu k danému záznamu. Kľúčovým slovám nastavíme frekvenciu, kde pozitívnu frekvenciu nastavíme na jedna a negatívnu frekvenciu na nulu. Ak sa však už dané slovo nachádza v tabuľke výskytov, vtedy budeme postupne prechádzať množinu kľúčových slov, ktoré sa už nachádzajú v tabuľke výskytov. Tieto kľúčové slová budeme porovnávať s tými, ktoré sú priradené k danému záznamu a kontrolovať, či sa dané kľúčové slová z tabuľky výskytov vyskytujú aj pri danom zázname. Ak áno, tak potom zvýšime ich pozitívnu frekvenciu. Ak sa však také kľúčové slová pri danom zázname nenachádzajú, tak zvýšime ich negatívnu frekvenciu. Následne po tomto kroku budeme ešte prechádzať kľúčové slová priradené k danému záznamu. Pri týchto kľúčových slovách budeme kontrolovať, či sa tu nenachádzajú slová, ktoré ešte v našej tabuľke výskytov nie sú. Ak sa tam také nachádzajú, tak pridáme do tabuľky novú dvojicu (slovo, kľúčové slovo) a nastavíme pozitívnu frekvenciu na jedna a negatívnu frekvenciu na súčet pozitívnej a negatívnej frekvencie slov, ktoré sa už predtým v tabuľke nachádzali.

Nasledujúca tabuľka opisuje príklad, kedy máme v databáze dva nasledujúce záznamy:

Záznam: Pracovná plocha výskumníka Kľúčové slová: systém, web
Záznam: Pracovná webová stránka Kľúčové slová: web

Tabuľka 6.3: Tabuľka výskytov dvojíc (slovo, kľúčové slovo)

Slovo	**Kľúčové slovo**	**FrekvenciaP**	**FrekvenciaN**
pracovná	web	2	0
pracovná	systém	1	1
plocha	web	1	0
plocha	systém	1	0
výskumníka	web	1	0
výskumníka	systém	1	0
webová	web	1	0
stránka	web	1	0

Testovacia časť

Aj v tomto prípade chceme pre jeden záznam, pre ktorý algoritmus doteraz nepozná priradenie kľúčových slov, ponúknuť usporiadaný zoznam odporúčaných kľúčových slov vhodných pre charakterizovanie tohto záznamu. Tento raz to urobíme tak, že pre každé potenciálne kľúčové slovo určíme pravdepodobnosť jeho priradenia. Pravdepodobnosť priradenia kľúčového slova bude nulová, ak nebolo priradené k ani jednému záznamu trénovacej množiny, v ktorom by sa vyskytovalo akékoľvek slovo z aktuálneho záznamu.

V tejto časti budeme opäť zisťovať, či sa slovo z práve pridávaného záznamu nachádza v tabuľke výskytov. Ak sa tam slovo nachádza, tak všetky kľúčové slová, s ktorými je slovo spojené pridáme do tabuľky odporúčaných slov. Ku každému kľúčovému slovu priradíme aj skóre a to nasledovným spôsobom. Z tabuľky výskytov vypočítame podmienenú pravdepodobnosť kľúčového slova za podmienky výskytu konkrétneho slova v zázname a to nasledovným spôsobom.

word!=0 znamená, že berieme do úvahy všetky slová, ktoré sa v trénovacej množine vyskytli aspoň raz.

$$Score(keyword) = \sum_{word!=0} P(keyword|word) =$$

$$\sum_{word!=0} \frac{FreqP}{FreqP + FreqN} \tag{6.1}$$

Teda v prípade, že sa kľúčové slovo nenachádza v tabuľke odporúčaných slov, tak mu vypočítame pravdepodobnosť. Ale ak sa už v tabuľke nachádza, tak k tejto pravdepodobnosti pričítame ďalšiu hodnotu.

Nasledujúci príklad ilustruje tabuľku odporúčaných slov pre nasledovný záznam.

Záznam: Pracovná webová stránka

Tabuľka 6.4: Tabuľka odporúčaných slov

Kľúčové slovo	**Skóre**
web	2
systém	0,5

6.2.5 Rozšírený algoritmus Maximum

Tento algoritmus funguje podobne ako rozšírený algoritmus sumy. Jediný rozdiel je v tom, že pravdepodobnosť nebudeme sčítavať, ale porovnávať. Teda, ak sa nám na vstupe objaví slovo, ktoré sa už nachádza v tabuľke výskytov a nejaké z jeho kľúčových slov sa už nachádza v tabuľke odporúčaných slov, tak potom vypočítame pravdepodobnosť pre toto kľúčové slovo a porovnáme ju s pravdepodobnosťou, ktorá sa nachádza v tabuľke odporúčaných slov. Ak je väčšia, tak hodnotu v tabuľke odporúčaných slov prepíšeme novou hodnotou pravdepodobnosti.

$$score(keyword) = Max(P(keyword|word)) \tag{6.2}$$

6.2.6 Rozšírený algoritmus priemer

Nasledujúci algoritmus funguje presne rovnako ako rozšírený algoritmus suma. Rozdiel je len v tom, že po sčítaní všetkých pravdepodobností. Bude súčet vydelený počtom sčítancov.

$$score(keyword) = Avg(P(keyword|word)) \tag{6.3}$$

6.3 Porovnanie Algoritmov

V nasledujúcej časti budeme porovnávať jednotlivé algoritmy. Budeme skúmať tri rôzne veličiny:

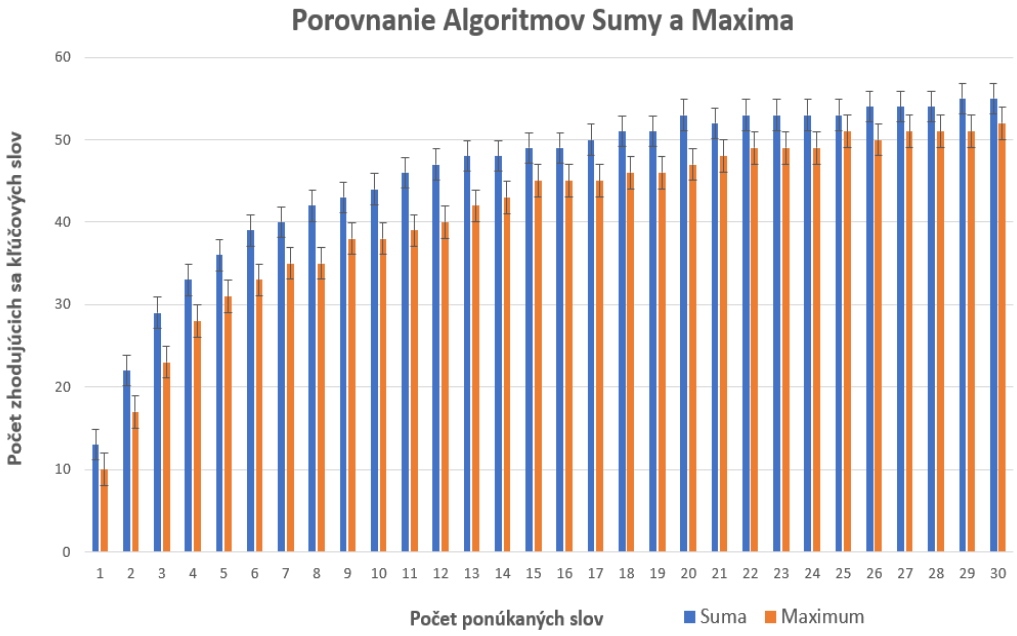
- celkový priemerný počet odporúčaných kľúčových slov pre 20 záznamov
- priemerný počet odporúčaných kľúčových slov pre jeden záznam
- celkový priemerný počet odporúčaných kľúčových slov pre všetky záznamy

Prvé dva grafy pri každom porovnaní algoritmov predstavujú priemerné hodnoty z datasetu o rozmere 120 záznamov s už priradenými kľúčovými slovami. Pričom na trénovaciu časť bolo použitých 100 záznamov a na testovaciu 20 záznamov. V testovacej časti budeme porovnávať, či sa kľúčové slová priradené k záznamom zhodujú s odporúčanými kľúčovými slovami.Celý tento proces budeme robiť tak, že najskôr náhodne zamiešame záznamy v datasete. Potom spustíme trénovaciu časť pre prvých 100 záznamov. Ďalej sa spustí testovacia časť pre zvyšných 20 záznamov. Následne tieto kroky budeme opakovať 200 krát, aby sme získali priemerné hodnoty pre určitý počet ponúkaných slov. Potom toto celé zopakujeme 30 pre rôzny počet ponúkaných slov.

Tretí graf bude predstavovať takýto experiment. V tomto experimente opäť zoberieme dataset o rozmere 120 záznamov a budeme algoritmus trénovať a testovať zároveň. Teda budeme postupne prechádzať všetky záznamy a pri každom pridaní záznamu do tabuľky výskytov budeme odporúčať určitý počet kľúčových slov. Tým pádom sa náš algoritmus bude trénovať za behu. Teda pri N-tom zázname bude odporúčať slová, ktoré sa naučil pri N-1 záznamoch. Postupovať budeme nasledovne. Na začiatku zamiešame záznamy v datasete. Potom budeme testovať a trénovať za behu. Toto zopakujeme 200-krát pre určitý počet ponúkaných slov. Potom celý tento proces zopakujeme 30 krát pričom budeme meniť počet ponúkaných slov.

6.3.1 Porovnanie algoritmov sumy a maxima

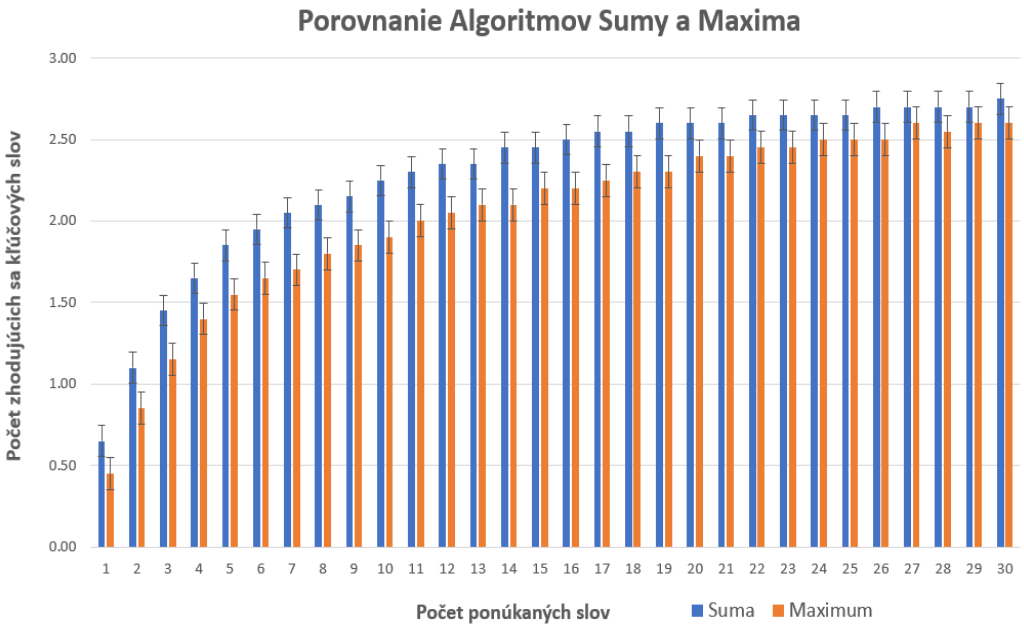
Graf na obrázku 6.2 zobrazuje celkové priemerné počty odporúčaných slov pre 20 záznamov testovacej časti.



Obrázok. 6.2: Graf celkového priemerného počtu úspešne odporúčaných kľúčových sumy a maxima slov pre 20 záznamov

V tomto grafe sme porovnali úspešnosť algoritmov sumy a maxima, ako môžeme vidieť algoritmus sumy dokáže odporúčať slová lepšie, ako algoritmus maxima.

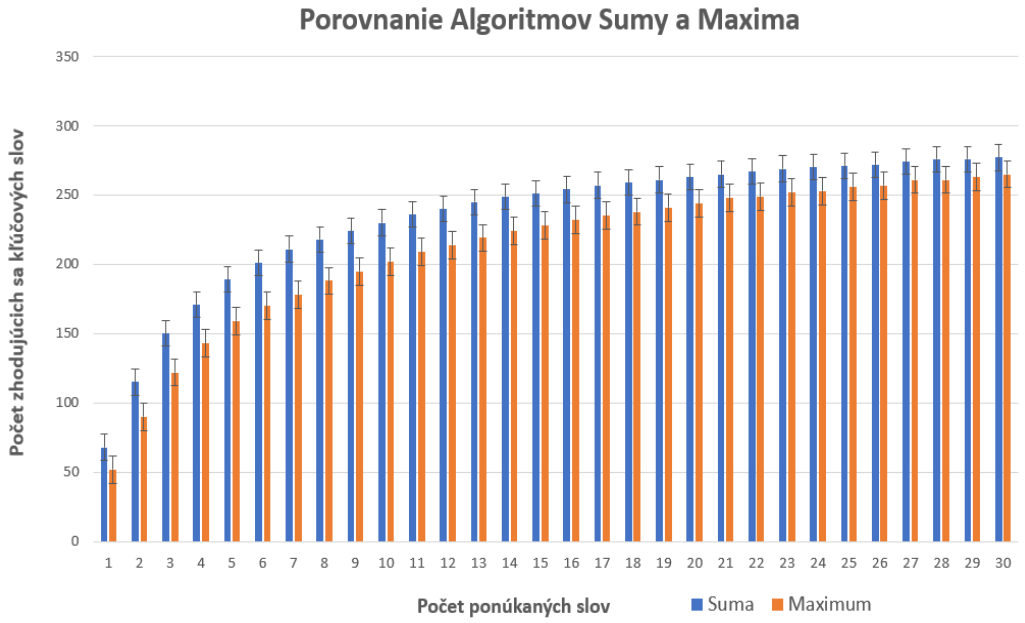
Nasledujúci graf zobrazuje priemerný počet úspešne odporúčaných slov pre jeden záznam



Obrázok. 6.3: Graf priemerného počtu úspešne odporúčaných kľúčových slov sumy a maxima pre jeden záznam

Ako môžeme vidieť i v tomto grafe je algoritmus sumy lepší, ako algoritmus maxima.

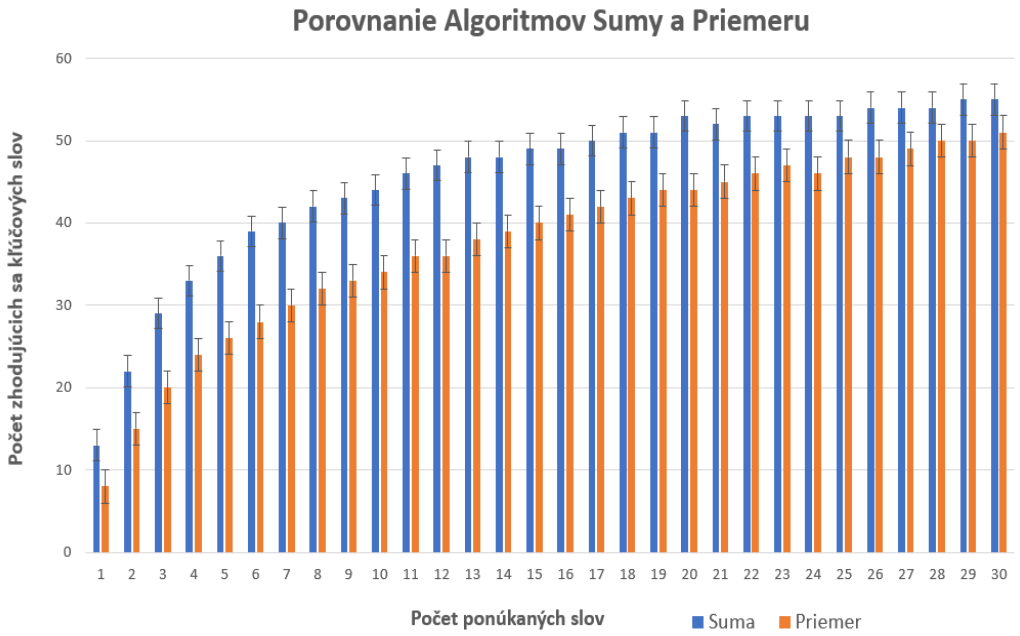
Nasledujúci graf zobrazuje celkový priemerný počet odporúčaných slov pre rôzny počet ponúkaných slov, pri trénovaní a testovaní za behu.



Obrázok. 6.4: Graf celkového priemerného počtu odporúčaných kľúčových slov sumy a maxima pri testovaní a trénovaní naraz

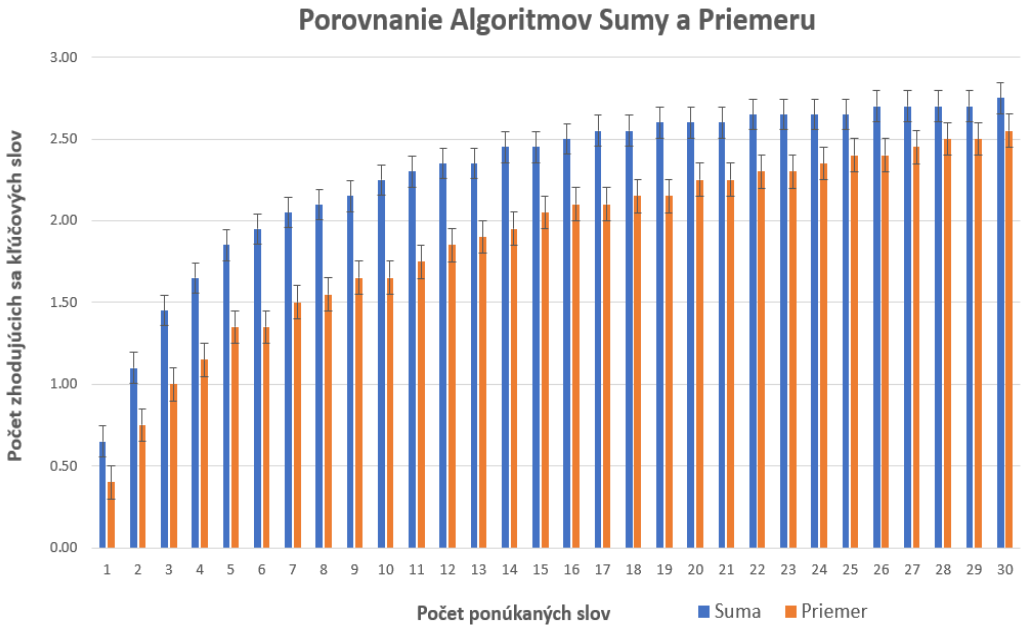
V tomto grafe je možné vidieť ten istý jav ako v predošlých dvoch. Teda v ďalšom porovnaní budeme pracovať už len s algoritmom sumy.

6.3.2 Porovnanie algoritmov sumy a priemeru



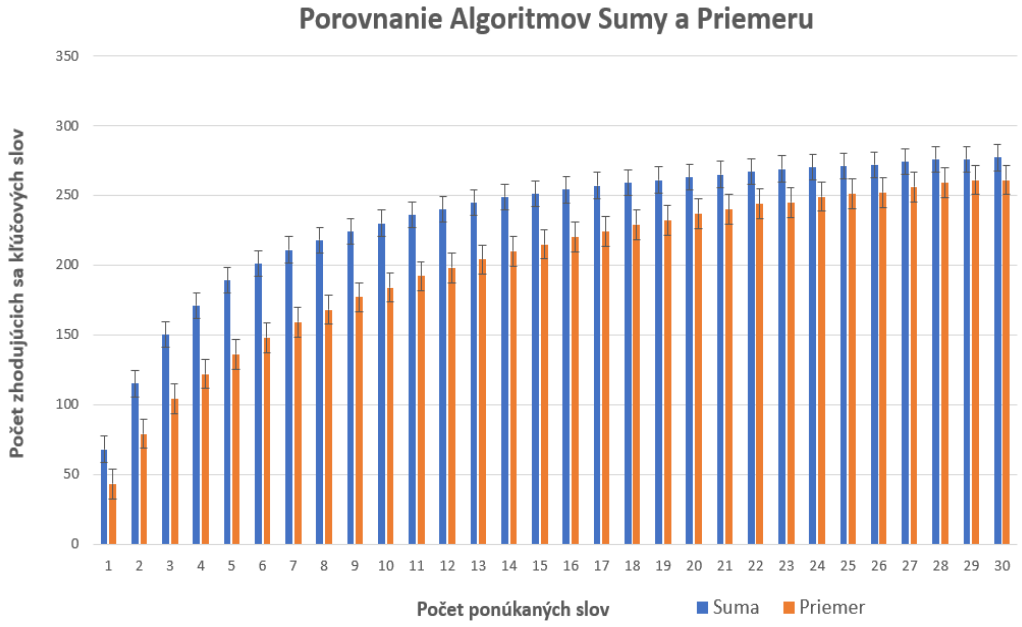
Obrázok. 6.5: Graf celkového priemerného počtu úspešne odporúčaných kľúčových slov sumy a priemeru pre 20 záznamov

Tento graf nám jasne ukazuje, že algoritmus sumy je ďaleko lepší, ako algoritmus priemeru.



Obrázok. 6.6: Graf priemerného počtu úspešne odporúčaných kľúčových slov sumy a priemeru pre jeden záznam

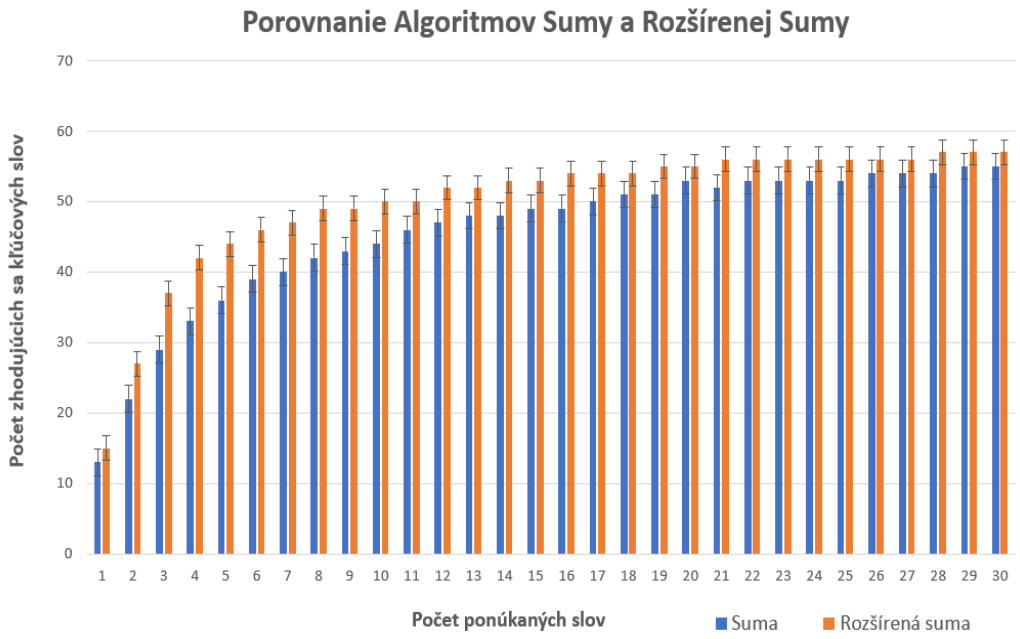
V tomto grafe môžeme vidieť ten istý jav ako v predošlom.



Obrázok. 6.7: Graf celkového priemerného počtu odporúčaných kľúčových slov sumy a priemeru pri testovaní a trénovaní naraz

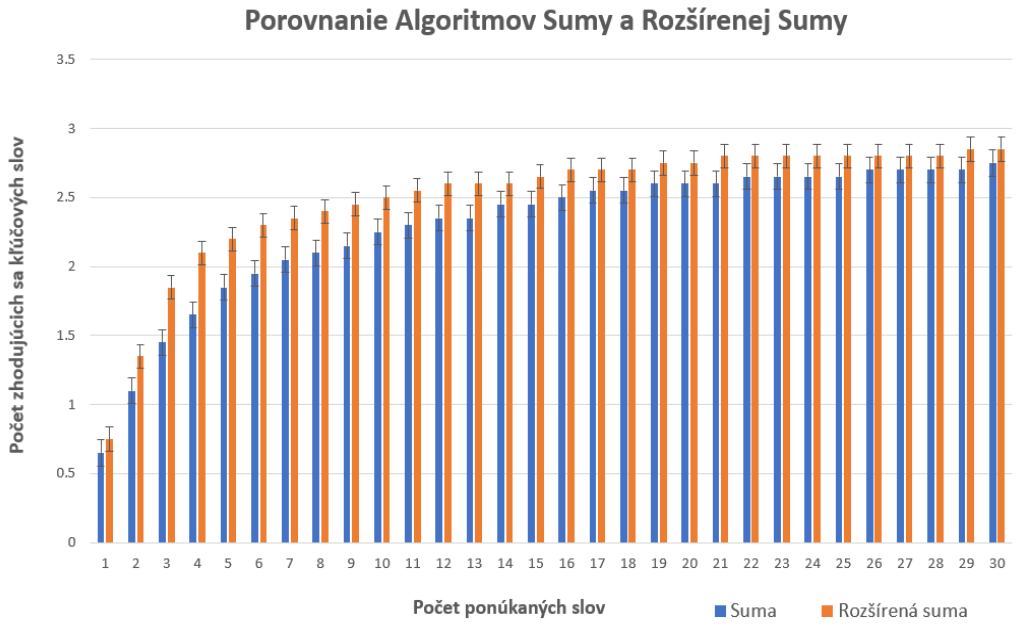
Z tohto ako aj z predchádzajúcich dvoch grafov môžeme usúdiť, že algoritmus sumy je jednoznačne lepší ako algoritmus priemeru pri všetkých meraných veličinách. Teda v ďalšom kroku budeme postupovať už len s algoritmom sumy.

6.3.3 Porovnanie algoritmov sumy a rozšírenej sumy



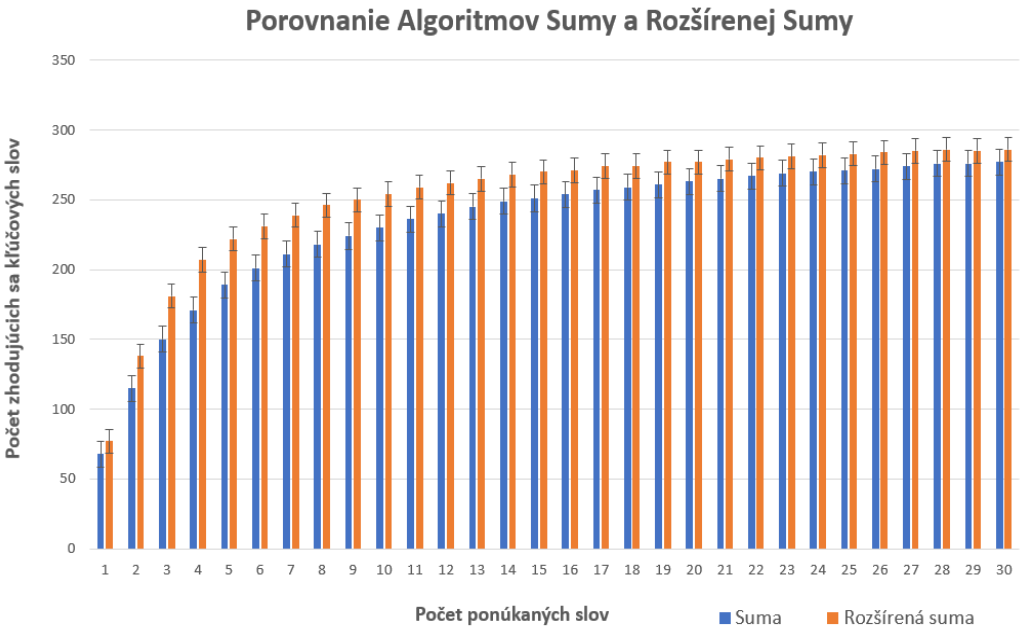
Obrázok. 6.8: Graf celkového priemerného počtu úspešne odporúčaných kľúčových slov sumy a rozšírenej sumy pre 20 záznamov

Predchádzajúci graf ukazuje, že rozšírená suma, ktorá je rozšírená práve o negatívnu frekvenciu. Dokáže odporúčať kľúčové slová lepšie ako naša pôvodná suma. Preto budeme ďalej pracovať s algoritmom rozšírenej sumy.



Obrázok. 6.9: Graf priemerného počtu úspešne odporúčaných kľúčových slov sumy a rozšírenej sumy pre jeden záznam

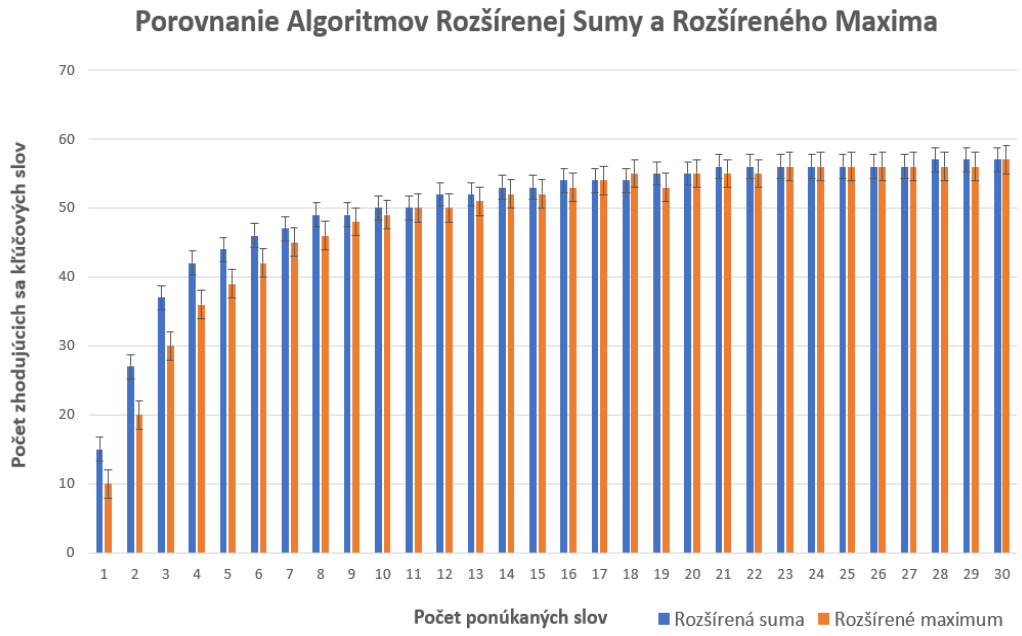
Ako môžeme vidieť na predchádzajúcom grafe priemerný počet úspešne odporúčaných slov je u rozšírenej sumy väčší ako u sumy.



Obrázok. 6.10: Graf celkového priemerného počtu odporúčaných kľúčových slov sumy a rozšírenej sumy pri testovaní a trénovaní naraz

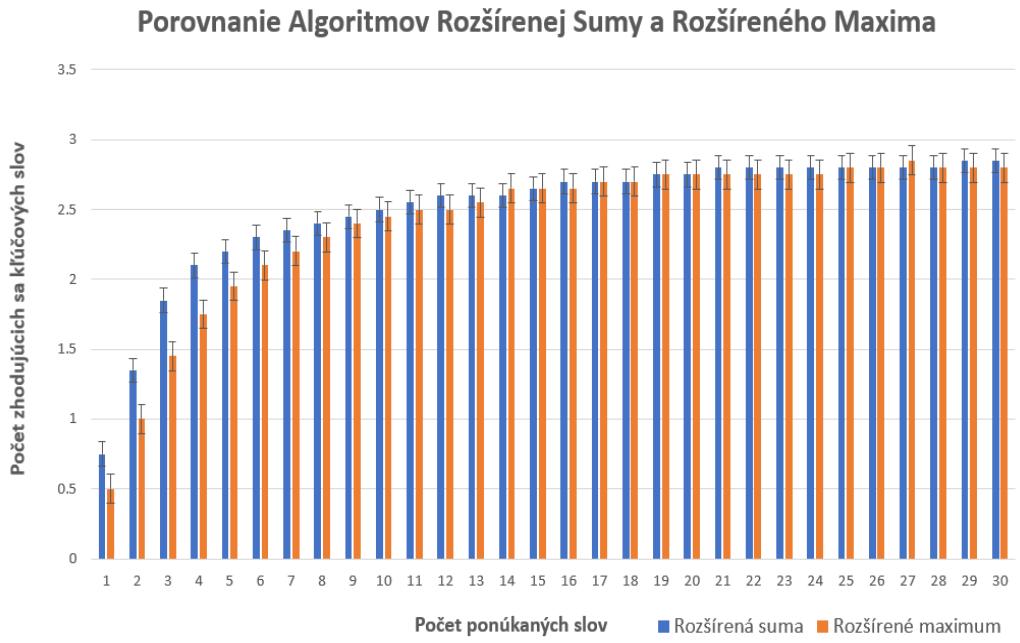
Z tohto i predchádzajúcich grafov môžeme usúdiť, že sme našli lepší algoritmus akým bol doposiaľ algoritmus sumy. V ďalších pokusoch budeme teda pracovať už len s algoritmom rozšírenej sumy.

6.3.4 Porovnanie algoritmov rozšírenej sumy a rozšíreného maxima



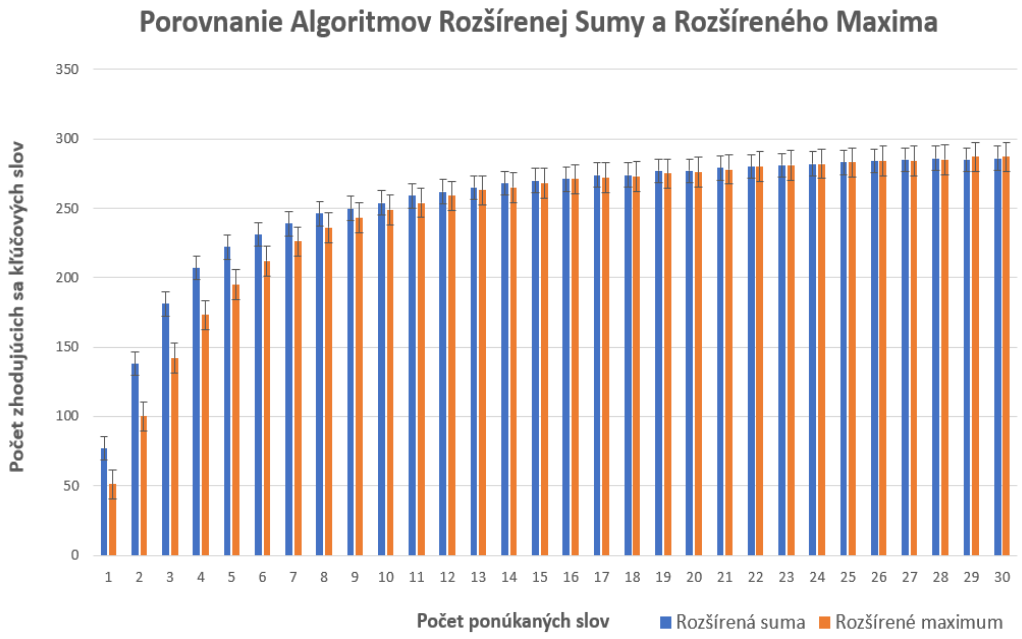
Obrázok. 6.11: Graf celkového priemerného počtu úspešne odporúčaných kľúčových slov rozšírenej sumy a rozšíreného maxima pre 20 záznamov

Na tomto grafe môžeme vidieť, že rozšírená suma je zo začiatku lepšia ako rozšírené maximum. Problémom rozšíreného maxima je to, že nevie správne určiť prioritu slov a správne slová sa objavia s nižšou prioritou ako v prípade rozšírenej sumy. No pri odporúčaní dostatočne veľkého počtu kľúčových slov, dokážu oba algoritmy odporúčať kľúčové slová rovnako vhodne. V prípade, že by naša aplikácia navrhovala do 10 odporúčaných slov, bolo by lepšie použiť algoritmus rozšírenej sumy.



Obrázok. 6.12: Graf priemerného počtu úspešne odporúčaných kľúčových slov rozšírenej sumy a rozšíreného maxima pre jeden záznam

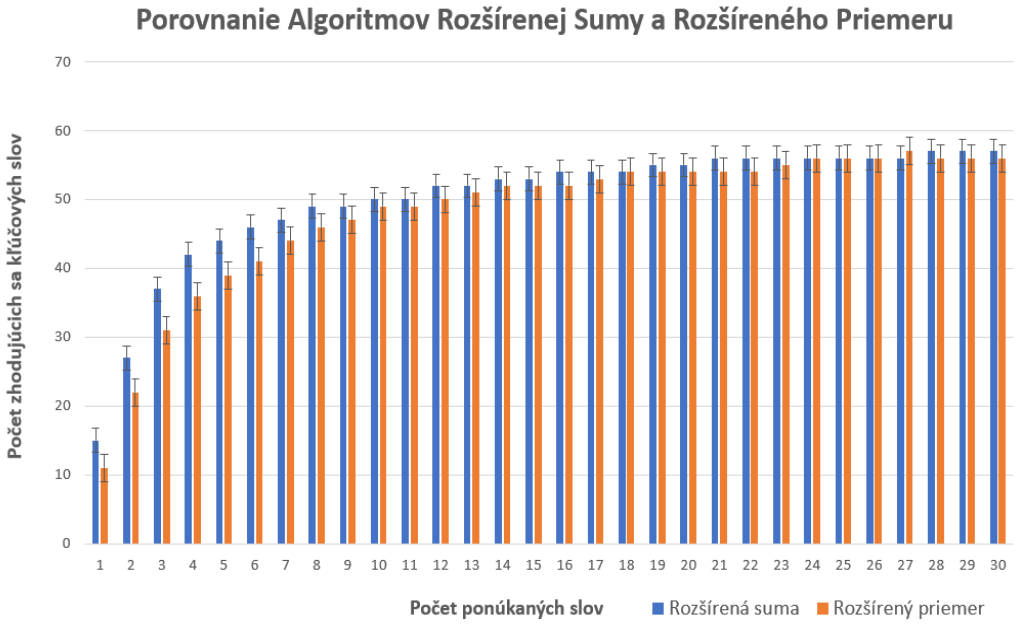
Tento graf nám ukazuje, že pri nižších hodnotách dosahuje rozšírená suma lepšie hodnoty približne o 0.5. Pri vyšších hodnotách sú už algoritmy vyrovnané.



Obrázok. 6.13: Graf celkového priemerného počtu odporúčaných kľúčových slov rozšírenej sumy a rozšíreného maxima pri testovaní a trénovaní naraz

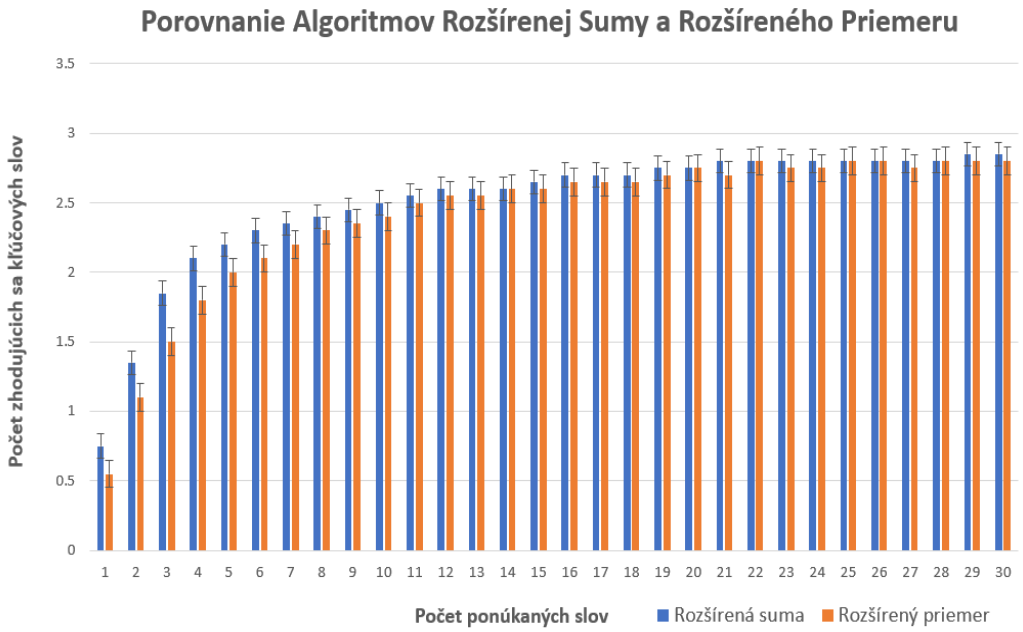
Tak ako v predošlých aj tento graf nám potvrdzuje to, že rozšírená suma je pri nižších počtoch ponúkaných slov lepším algoritmom. Z toho dôvodu budeme v našom výskume pracovať ďalej s rozšírenou sumou.

6.3.5 Porovnanie algoritmov rozšírenej sumy a rozšíreného priemeru



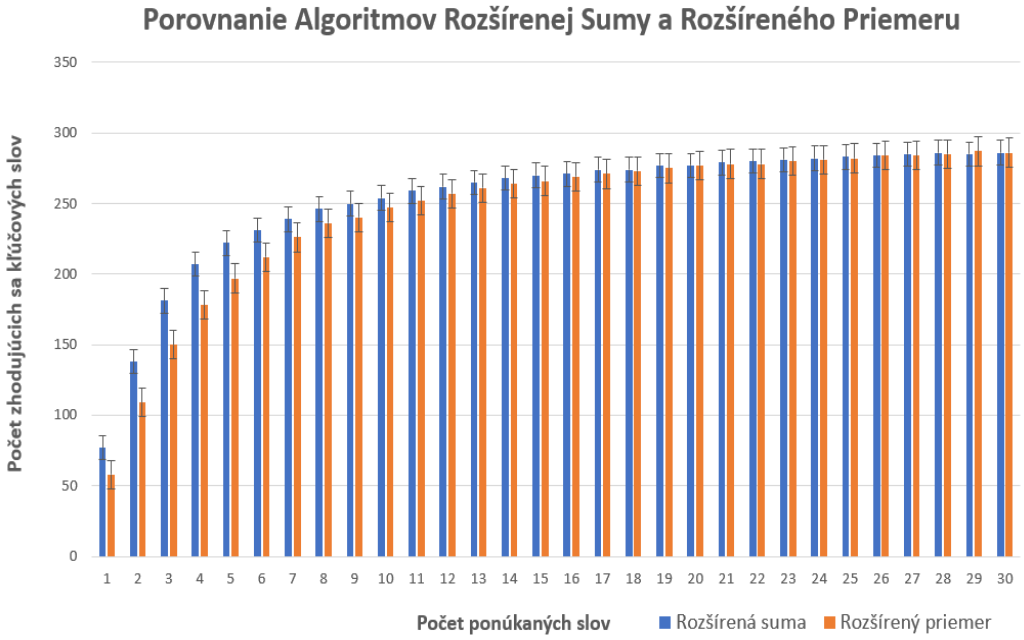
Obrázok. 6.14: Graf celkového priemerného počtu úspešne odporúčaných kľúčových slov rozšírenej sumy a rozšíreného priemeru pre 20 záznamov

Tu môžeme vidieť opäť ten istý jav ako v predchádzajúcom grafe. Algoritmus rozšírenej sumy je lepší v prípade, že chceme odporučiť menší počet slov. V prípade väčšieho počtu slov sú si algoritmy vyrovnané.



Obrázok. 6.15: Graf priemerného počtu úspešne odporúčaných kľúčových slov rozšírenej sumy a rozšíreného priemeru pre jeden záznam

Tá istá informácia, ktorú sme získali v predchádzajúcom grafe platí aj v tomto. Rozšírená suma je lepšia pri nižších hodnotách ponúkaných slov.



Obrázok. 6.16: Graf celkového priemerného počtu odporúčaných kľúčových slov rozšírenej sumy a rozšíreného priemeru pri testovaní a trénovaní naraz

Z výsledkov vyplývajúcich z predošlých grafov sme dospeli k tomu, že lepším algoritmom z tejto dvojice je rozšírená suma.

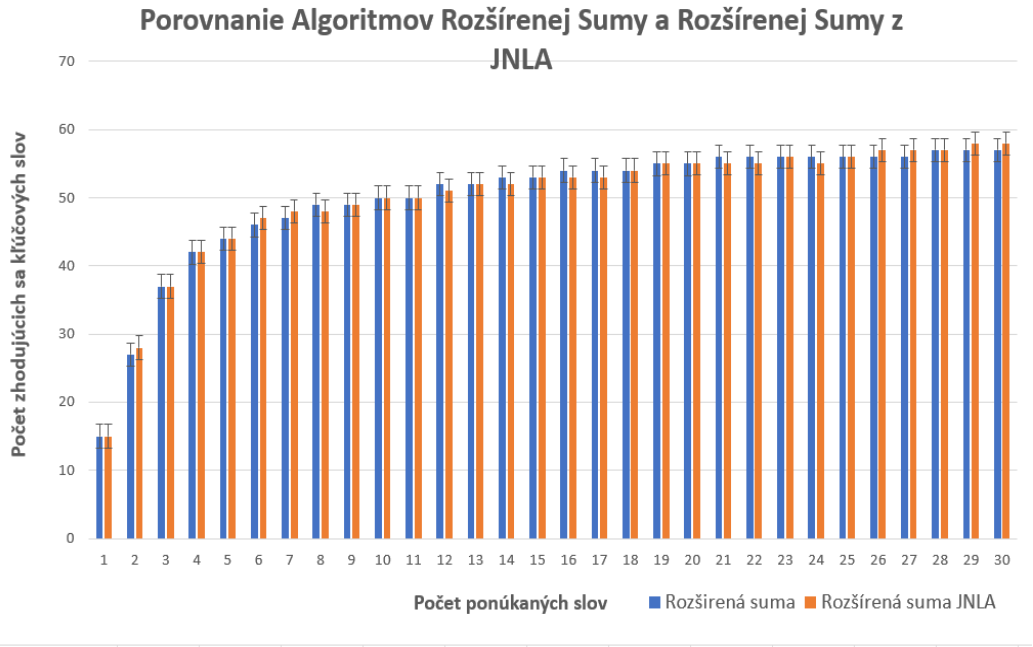
6.4 Algoritmus podobnosti slov JNLA

Problém priraďovania správneho kľúčového slova na základe vstupného, nie nutne správne vyhláskovaného slova, bol študovaný využitím štatistických konštrukcií Jaccard, N-Gram a Vector Space v článku [SN13]. Rozhodli sme sa využiť túto metódu na vylepšenie úspešnosti nášho algoritmu. Pre slová, ktoré sa síce nenachádzajú v databáze, ale sú im veľmi podobné.

Napríklad ak by sa v databáze vyskytovalo slovo: informatika. Ale v práve pridávanom zázname by sa vyskytovali slová ako informatik, informatiky, in-

formatike. Tak by sme pre daný záznam nedostali žiadne kľúčové slová. Preto by bolo rozumné použiť algoritmus, ktorý by zisťoval podobnosť medzi slovami práve pridávaného záznamu, ktoré sa v databáze nenachádzajú. So slovami, ktoré už v databáze sú. V prípade, že by sme našli také slová, ktorých odlišnosť je minimálna. Tak by sme vybrali slovo s najväčšou podobnosťou a pre toto slovo by sme pridali odporúčané kľúčové slová. Samozrejme nepridávali by sme hocijaké slová s najväčšou podobnosťou, ale len také, ktorých podobnosť so slovom v databáze by presiahla určitú hodnotu. Konkrétne pri algoritme JNLA by musela byť hodnota podobnosti daných slov prekročiť hodnotu 0.67.

Nasledujúci graf zobrazuje celkové priemerné počty odporúčaných slov pre 20 záznamov testovacej časti



Obrázok. 6.17: Graf porovnania rozšírenej sumy a rozšírenej sumy s JNLA pre celkové priemerné počty odporúčaných slov pre 20 záznamov

Na tomto grafe môžeme vidieť, že algoritmus Rozšírená suma spolu s algoritmom JNLA dosahuje približne rovnaké hodnoty ako algoritmus rozšírenej sumy. Ale pre komfort používateľa bude lepšie, keď použijeme algoritmus s JNLA. Teda pri pridávaní nového záznamu nemusí nastať presná zhoda s tým, čo sa už nachádza v databáze. Pokiaľ budú slová dostatočne podobné algoritmus ponúkne používateľovi aspoň niečo. Zatiaľ čo v prípade použitia rozšírenej sumy by algoritmus neponúkol používateľovi žiadne kľúčové slová, dokým by nenastala presná zhoda.

Pre tento algoritmus sme spravili ešte jeden experiment, ktorý je popísaný v nasledujúcej tabuľke. Nasledujúca tabuľka obsahuje v riadku i počet kľúčových slov, ktoré algoritmus odporúča. V stĺpci k požadujeme aby algoritmus odporučil aspoň k kľúčových slov správne. Číslo v bunke (i,k) vyjadruje v koľkých z 20 prípadov priemerne došlo k splneniu podmienky zadanej v stĺpcoch.

Tabuľka 6.5: Absolútna úspešnosť algoritmu pri podmienke správneho odporučenia aspoň k kľúčových slov

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
1	15	0	0	0	0	0	0	0	0	0	0	0
2	17	11	0	0	0	0	0	0	0	0	0	0
3	17	13	6	0	0	0	0	0	0	0	0	0
4	18	14	8	1	0	0	0	0	0	0	0	0
5	18	14	8	2	0	0	0	0	0	0	0	0
6	18	14	9	3	0	0	0	0	0	0	0	0
7	18	15	9	3	0	0	0	0	0	0	0	0
8	18	15	10	3	0	0	0	0	0	0	0	0
9	18	15	10	3	0	0	0	0	0	0	0	0
10	18	15	10	3	0	0	0	0	0	0	0	0
11	19	15	10	4	1	0	0	0	0	0	0	0
12	19	16	10	4	0	0	0	0	0	0	0	0
13	19	16	10	4	1	0	0	0	0	0	0	0
14	19	16	11	4	1	0	0	0	0	0	0	0
15	19	16	11	4	0	0	0	0	0	0	0	0
16	19	16	11	4	1	0	0	0	0	0	0	0
17	19	16	11	4	1	0	0	0	0	0	0	0
18	19	16	11	5	1	0	0	0	0	0	0	0
19	19	16	11	5	1	0	0	0	0	0	0	0
20	19	16	11	5	1	0	0	0	0	0	0	0
21	19	16	11	5	1	0	0	0	0	0	0	0
22	19	16	11	5	1	0	0	0	0	0	0	0
23	19	17	12	5	1	0	0	0	0	0	0	0
24	19	17	11	5	1	0	0	0	0	0	0	0
25	19	16	11	5	1	0	0	0	0	0	0	0
26	19	16	11	5	2	0	0	0	0	0	0	0
27	19	17	12	5	2	0	0	0	0	0	0	0
28	19	17	12	5	2	0	0	0	0	0	0	0
29	19	17	12	5	2	0	0	0	0	0	0	0
30	19	17	12	5	2	0	0	0	0	0	0	0

6.5 Zhrnutie

V predchádzajúcej časti sme porovnávali a skúmali 6 rôznych algoritmov na odporúčanie kľúčových slov. Z jednotlivých grafov sme usúdili, že medzi najlepšie algoritmy patria rozšírené algoritmy sumy, maxima a priemeru. Tieto algoritmy sme medzi sebou porovnali. Zistili sme, že pri vyšších počtoch ponúkaných slov odporúčajú kľúčové slová rovnako dobre. Ale pri menšom počte ponúkaných slov bol algoritmus rozšírenej sumy lepší ako algoritmy rozšíreného maxima a rozšíreného priemeru. Najlepší zo všetkých algoritmov je teda algoritmus rozšírenej sumy.

Všetky tieto grafy spolu z údajmi a zdrojovými kódmi sa budú nachádzať v elektronickej prílohe práce.

7

Návrh

7.1 Používateľské rozhranie

Používateľské rozhranie je rozdelené na tri časti.

Prvá časť, ktorá sa nachádza naľavo je zoznam všetkých kategórií, ktoré daná aplikácia obsahuje. Tento zoznam môžeme meniť pridaním vlastnej kategórie, prípadne zmazaním už existujúcej kategórie.

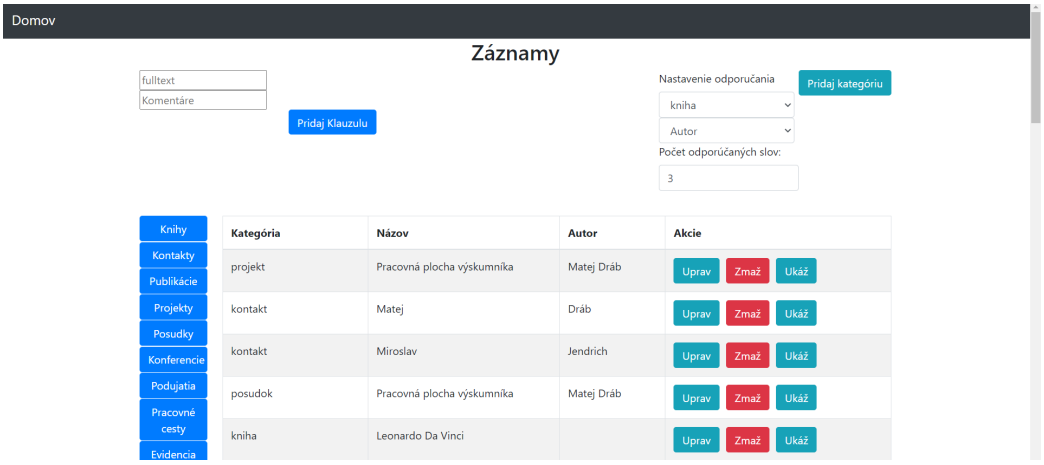
Druhá časť sa nachádza v hornej časti aplikácie. Sem sa nachádza vyhľadávanie, pridanie novej kategórie a nastavenie odporúčania kľúčových slov. Konkrétne tu nastavujeme kritéria, na základe ktorých sa nám budú zobrazovať odporúčané kľúčové slová. Môžeme nastaviť, aby sa nám odporúčali kľúčové slová na základe slov zo všetkých záznamov v databáze alebo len záznamov konkrétnej kategórie. Prípadne len konkrétneho atribútu konkrétnej kategórie. Napríklad ak nastavíme kategóriu kontakty a atribút priezvisko. Tak sa nám budú odporúčať kľúčové slová, len ak sa v novo pridávanom zázname vyskytne slovo, ktoré sa nachádza v zázname s kategóriou kontakty pri atribúte priezvisko, ktorý sa už nachádza v databáze. Všetky kľúčové slová , ktoré má priradený daný záznam sa nám pridajú do množiny odporúča-

ných slov. Okrem tohto nastavenia môžeme nastaviť aj počet odporúčaných kľúčových slov, ktoré nám algoritmus bude ponúkať.

V rámci vyhľadávania aplikácia ponúka tri druhy vyhľadávania.

- fulltextové vyhľadávanie zobrazuje záznamy, v ktorých atribútoch sa nachádza hľadaná hodnota.
- fulltextové vyhľadávanie v komentároch zobrazí záznamy, v ktorých komentároch sa nachádza hľadaná hodnota.
- CNF(Conjunctive normal form) vyhľadávanie vyhľadá záznamy, podľa toho či spĺňajú podmienku zadanú v CNF. Napríklad, či sa v daných záznamoch nachádza kľúčové slovo.

Tretia časť sa nachádza v strede a tvoria ju záznamy, ktoré sú uložené v našej aplikácii.



Obrázok. 7.1: Ukážka hlavnej stránky aplikácie

7.1.1 Zobrazenie záznamov

Ako môžeme vidieť na predchádzajúcom obrázku záznamy sa zobrazujú v strede našej aplikácie. Vždy po kliknutí na konkrétnu kategóriu sa zobrazia záznamy len danej kategórie. Záznamy sa zobrazujú po jednom v tabuľke. Každý riadok tabuľky predstavuje jeden záznam. Na hlavnej stránke sa nám nezobrazujú všetky informácie o záznamoch, zobrazujú sa tu len hodnoty prvých dvoch atribútov zobrazovanej kategórie. S každým záznamom je možné robiť určité druhy operácií a to konkrétne: upravovať, mazať, zobraziť detail záznamu. V prípade, že by sme chceli nejaký záznam upraviť, stačí kliknúť na tlačidlo upraviť a tam už môžeme upraviť konkrétne hodnoty daného záznamu. v prípade, že klikneme na tlačidlo zmazať daný záznam sa zmaže. Pri kliknutí na tlačidlo ukáž sa nám ukážu všetky informácie o danom zázname, vrátane záznamov naň prepojených spolu s komentármi a súbormi, ktoré boli k danému záznamu pridané. V prípade, že by sme chceli pridať nový záznam stačí kliknúť na tlačidlo pridaj záznam, ktoré sa zobrazí hore nad vyhľadávaním pri výbere konkrétnej kategórie.

7.1.2 Pridávanie a editovanie záznamov

V prípade pridávania alebo editovania záznamov sa nám v strede okna zobrazí formulár, do ktorého môžeme vkladať dáta. V prípade editovania sú už dáta vopred vyplnené na základe dát uložených v databáze. Formulár bude obsahovať nasledujúce typy údajov.

- Údaje príslušnej kategórie. Tu sa zobrazia polia pre všetky atribúty konkrétnej kategórie, ktorej záznam sme sa rozhodli pridať (editovať).
- Externé odkazy obsahujú pole pre url adresu stránky a textové pole pre komentár.

- Interné odkazy obsahujúce odkaz na konkrétny záznam v našej aplikácii.Spolu s ním môžeme pridať aj komentár k danému záznamu. Vďaka tomu môžeme vytvoriť vzťah medzi záznamami.

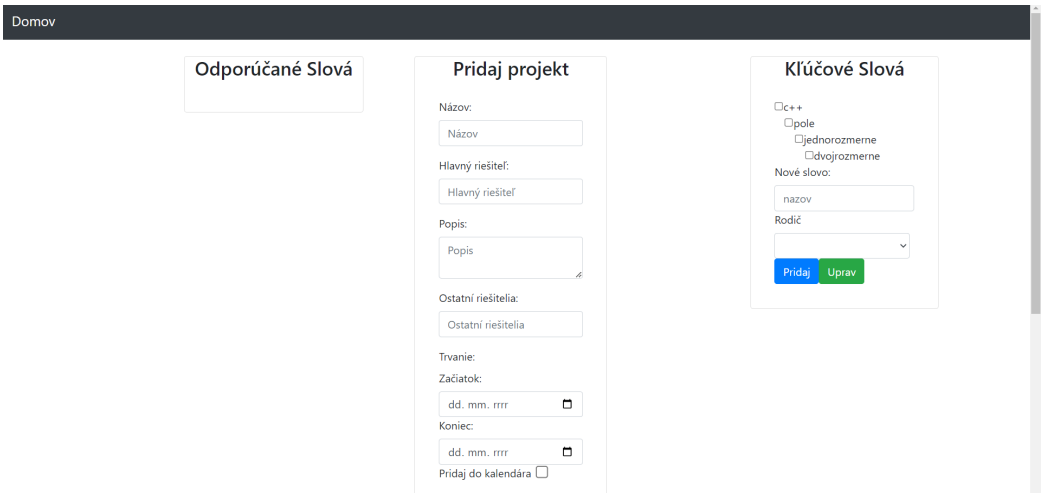
- Komentáre sem môže používateľ zadať ľubovoľný počet komentárov popisujúcich daný záznam.

- Súbory používateľ má možnosť vybrať požadované súbory a nahrať ich do aplikácie.

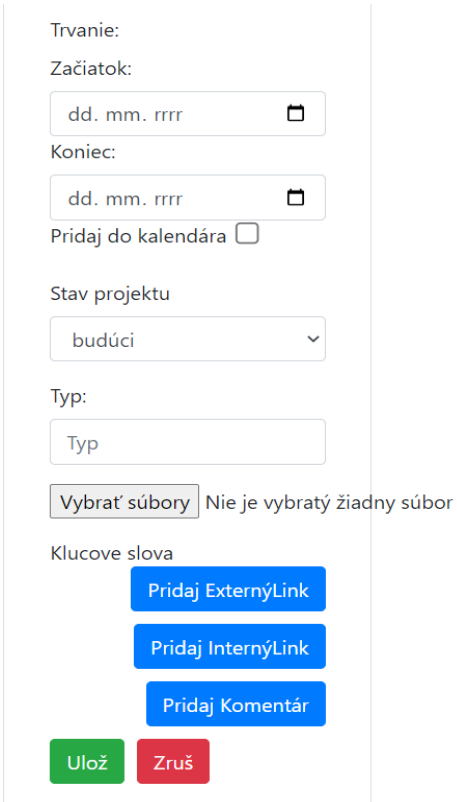
- Kľúčové slová sem sa zobrazia všetky kľúčové slová, ktoré sa používateľ rozhodol pridať k danému záznamu.

Pri pridávaní záznamu sa nám v ľavej časti okna zobrazia odporúčané kľúčové slová, ktoré sú ponúknuté k danému záznamu na základe algoritmu odporúčaných slov. Tento algoritmus odporúča kľúčové slová k pridávanému záznamu na základe dát vyskytujúcich sa v jednotlivých poliach pre atribúty pridávaného záznamu. Používateľ môže zaškrtnúť požadované kľúčové slovo, ktoré sa následne po označení zobrazí v strede okna pri položke kľúčové slová.

Pri pridávaní, ako aj pri editovaní záznamu sa nám v pravej časti okna zobrazia všetky kľúčové slová, ktoré sme už v danej aplikácii vytvorili. Tak isto ako pri odporúčaných slovách môžeme aj tieto slová priradiť k danému záznamu. Taktiež môžeme v tomto okne aj pridať nové kľúčové slovo.



Obrázok. 7.2: Ukážka pridávania záznamov



Obrázok. 7.3: Ukážka pridávania záznamov

7.1.3 Pridávanie a editovanie kategórií

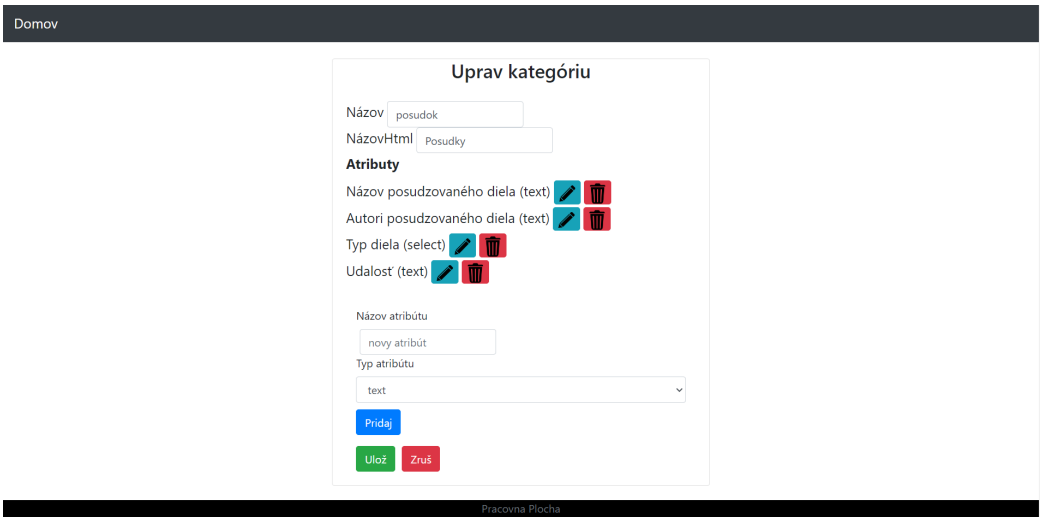
Pridávanie novej kategórie sa uskutočňuje vpísaním novej kategórie do textového poľa vo formáte JSON. Konkrétne formát kategórie bude vyzerať nasledovne:

{"nazov": "Názov kategórie", "nazovHtml": "NázovKategorie", ätributy": [{"nazov": "prvy atribút", "typ": "text"} ,{"nazov": "druhy atribút", "typ": "text"}] }

Kde nazov predstavuje názov, pod ktorým sa bude kategória zobrazovať v jednotnom čísle. Zatiaľ čo nazovHtml predstavuje množné číslo. V položke atribútov sa nachádza atribút typ. Tento atribút určuje typ daného atribútu. Môže nadobúdať iba jednu z nasledujúcich hodnôt: text, textarea, number, date, dateinterval, checkbox, select.

Pri editovaní kategórie sa nám taktiež zobrazí v strede okna formulár, kde sa budú nachádzať už vopred načítané údaje o danej kategórii. Konkrétne sa tu bude nachádzať názov kategórie a jej atribúty. V tomto prípade môžeme zmeniť názov alebo zmeniť názov atribútu. Prípadne i vymazať alebo pridať nový atribút. Pri vymazaní atribútu sa nám však vymažú aj všetky dáta, ktoré obsahovali záznamy v databáze pri danom atribúte.

Taktiež je možné okrem pridania novej kategórie zmazať kategóriu. V prípade, že nejakú kategóriu zmažeme, tak spolu s ňou zmažeme aj všetky záznamy, ktoré daná kategória obsahovala.



Obrázok. 7.4: Ukážka editovania kategórie

Obrázok. 7.5: Ukážka pridávania kategórie

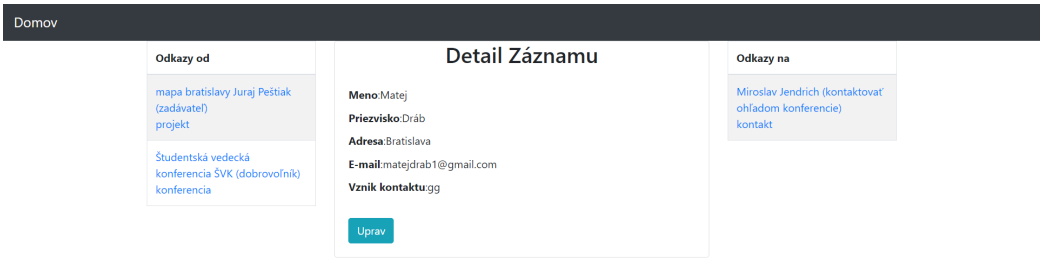
7.1.4 Detail záznamu

Pri detaile záznamu sa nám v strede okna zobrazia podrobné informácie o zázname. Zobrazia sa nám tu všetky vyplnené polia s atribútmi. Polia, ktoré pri pridávaní alebo upravovaní neboli vyplnené sa tu nezobrazia. Okrem toho sa zobrazia aj všetky komentáre, externé linky a súbory.

V ľavej časti okna sa zobrazia všetky záznamy, ktoré obsahujú odkaz na daný záznam. V rámci takéhoto záznamu sa zobrazia iba prvé dva atribúty a v zátvorke sa zobrazí, aký je vzťah medzi týmito záznamami.

V pravej časti okna sa zobrazia záznamy, na ktoré sa odkazujeme z práve

prezeraného záznamu. Zobrazia sa sem tie isté informácie o záznamoch ako v ľavej časti okna.



Obrázok. 7.6: Ukážka detailu záznamu

7.2 Architektúra

Naša aplikácia sa delí na dva väčšie celky, ktoré spolu komunikujú konkrétne sa delí na backend a frontend. Na frontende používame technológiu React a na backende používame technológiu Springboot.

7.2.1 Backend

Backendová časť našej aplikácie pozostáva z nasledujúcich častí.

- Controller

  Modul controller slúži na odchytávanie REST API. V tomto module definujeme, čo sa má stať pri konkrétnej HTTP požiadavke. Inak povedané ak klient pošle na backend požiadavku typu GET. V tomto module je definované ako sa má daná požiadavka spracovať a čo má vrátiť ako odpoveď.

- Repository

  Tento modul slúži na komunikáciu s databázou.

7.2.2 Frontend

Frontendová časť aplikácie pozostáva z nasledujúcich modulov.

- Komponent

  Frontendová časť sa skladá z viacerých komponentov, ktoré spolu interagujú. Každý komponent zabezpečuje istú časť funkcionality aplikácie a tiež zobrazuje určité časti stránky.

- Spojenie Tento modul slúži na komunikáciu s backendom prostredníctvom http požiadaviek. V tomto module využívame knižnicu axios prostredníctvom, ktorej posielame jednotlivé typy požiadaviek.

7.3 Návrh Databázy

Na obrázku nižšie môžeme vidieť databázový model našej aplikácie obsahujúci jednotlivé tabuľky a vzťahy medzi nimi.

Tabuľka **kategorie** uchováva jednotlivé kategórie, obsahuje atribúty id, názov a nazovHTml. V tejto tabuľke uchovávame všetky používateľom vytvorené kategórie. Táto tabuľka je vo vzťahu one-to-many k tabuľke **atributy**, pretože jedna kategória môže obsahovať viacero atribútov. Tabuľka atribútov pozostáva z atribútov: id, nazov, typ, id kategorie. Kde názov môže byť ľubovoľný textový reťazec. Atribút typ môže nadobúdať iba hodnoty: text, textarea, select, checkbox, date, number, dateinterval.

Ďalej môžeme vidieť, že tabuľka atribúty je prepojená s tabuľkou **enum**. Toto prepojenie slúži, v prípade že atribút bude typu select, teda bude mať pridelených viacero hodnôt.

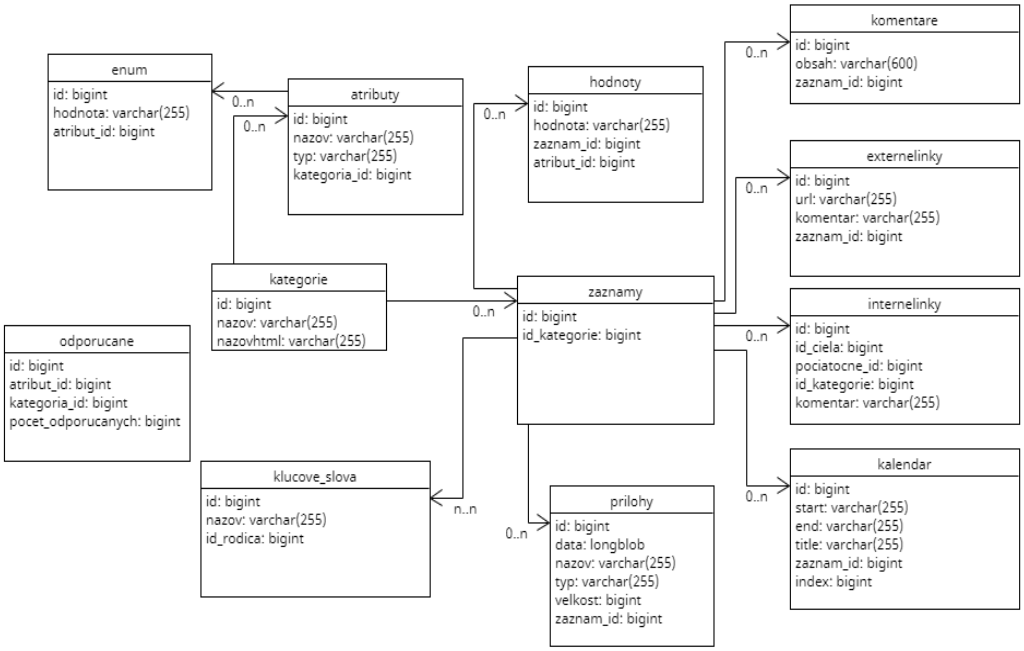
Tabuľka **kategorie** je prepojená aj s tabuľkou **zaznamy**. Tieto tabuľky sú vo vzťahu many-to-one, pretože jeden záznam môže mať len jednu kate-

góriu, ale jedna kategória môže mať viacero záznamov. Tabuľka záznamov je prepojená s tabuľkou **hodnoty**. V tejto tabuľke sa uchovávajú všetky hodnoty atribútov prislúchajúcich ku kategórii daného záznamu. Každý riadok tabuľky obsahuje id atribútu, id záznamu, ku ktorému prislúcha a hodnotu daného atribútu. Záznamy sú ďalej prepojené s tabuľkou **komentare**, kde každý komentár obsahuje svoje id, text komentára a id záznamu, ktorý k nemu patrí. Tabuľka **prilohy**, **externelinky** a **kalendar** vyzerajú podobne ako tabuľka komentárov.

Pokiaľ ide o tabuľku **interneliny**, tak v tejto tabuľke sa ukladajú všetky odkazy smerujúce od záznamu k inému záznamu, ale aj od iných záznamov k danému záznamu. Konkrétne, ak pridáme k nejakému záznamu odkaz na iný záznam ako počiatočné id sa nastaví id pridávaného (upravovaného) záznamu a ako idciela sa nastaví id záznamu, na ktorý chceme pridať odkaz. Ako Id kategórie sa nastaví id kategorie záznamu, na ktorý odkazujeme. Atribút komentár v tomto prípade popisuje vzťah medzi danými záznamami.

Záznamy sú taktiež prepojené s tabuľkou **klucove_slova**. Tabuľky majú medzi sebou vzťah many-to-many, pretože jeden záznam môže obsahovať viacero kľúčových slov. Ale taktiež jedno kľúčové slovo môže obsahovať viacero záznamov.

Tabuľka **odporucane** uchováva nastavenia odporúčania, ktoré používateľ nastavil.



Obrázok. 7.7: Databázový model aplikácie

8

Implementácia

V tejto kapitole sa budem venovať implementácii jednotlivých častí aplikácie.

8.1 Backend

V rámci backendovej časti aplikácie musíme pre každú tabuľku v databázovom modeli vytvoriť triedu v jave, ktorá bude popisovať danú tabuľku. Tejto triede bude pridelená anotácia *@Entity* a jej atribútom, ktoré budú predstavovať stĺpce tabuľky bude pridelená anotácia *@Column*. Pre túto triedu vytvoríme dva konštruktory. Jeden bez parametrov a druhý s parametrami a pre všetky atribúty vytvoríme get a set metódy. Všetky vzťahy medzi jednotlivými entitami popíšeme v týchto atribútoch príslušnými anotáciami typu *@ManyToMany*, *@OneToMany* a podobne, pričom k nim musíme pridať aj anotáciu *@JoinColumn*, v ktorej nastavíme primárny kľúč danej tabuľky a meno cudzieho kľúča v druhej tabuľke. Okrem toho bude nutné každej triede vytvoriť interface, ktorý bude dediť od JPARepository a bude mu pridelená anotácia *@Repository*. Následne je nutné už len vytvoriť triedy pre Controllery, ktorým priradíme anotácie *@RestController*, *@RequestMapping*,

@CrossOrigin. Tieto triedy bude obsahovať metódy, ktoré budú volať jednotlivé interfacy, na základe ktorých budeme vykonávať databázové dopyty. Metódy tejto triedy musia obsahovať anotáciu na základe http požiadavky, na ktorú budú posielať odpoveď.

```
@GetMapping("/kategorie")
public List<Kategoria> getKategorie() { return kategoriaRepozitar.findAll(); }
```

Obrázok. 8.1: Ukážka metódy Controllera

Okrem týchto vecí bude backendová časť obsahovať aj algoritmus dynamického prideľovania kľúčových slov a iné pomocné triedy pre spracovanie požiadaviek prichádzajúcich z frontendu.

8.2 Frontend

Frontendová časť našej aplikácie je tvorená viacerými komponentami. Každý z týchto komponentov zabezpečuje istú časť z funkcionality tejto aplikácie. Nasledovať bude popis jednotlivých komponentov.

8.2.1 Komponent App

V tomto komponente je implementované presmerovanie medzi jednotlivými stránkami a komponentami. Sú tu nastavené url adresy a komponenty, ktoré sa majú zobraziť pri príslušnej adrese v prehliadači.

8.2.2 Komponent ListZaznam

Tento komponent zobrazuje hlavnú stránku našej aplikácie a je zodpovedný za zobrazovanie záznamov v tabuľke, ktoré sa v ňom načítajú z databázy.

Okrem toho načítava z databázy aj všetky kategórie, ktoré zobrazuje v podobe tlačidiel. V prípade kliknutia na jednu z kategórií sa nastaví premenná vybranaKategoria na id konkrétnej kategórie. Potom sa nám budú zobrazovať už len záznamy danej kategórie.

Komponent tiež obsahuje funkcionalitu nastavenia odporúčania kľúčových slov. Konkrétne sú v ňom použité dva elementy typu select, kde je možné nastaviť kategóriu a atribút, na základe ktorých sa budú potom odporúčať kľúčové slová pri pridávaní záznamu. Pri výbere sa dáta z daného elementu automaticky uložia do databázy.

V tomto komponente je tiež implementované vyhľadávanie. Používame tu tri typy vyhľadávania a to fulltextové vyhľadávanie, ktoré prehľadáva všetky atribúty kategórií a hľadá či sa v nich nevyskytuje zadaný reťazec. Druhým typom je vyhľadávanie v komentároch. Toto vyhľadávanie prehľadá všetky komentáre a vráti záznamy, ktoré dané komentáre obsahujú. V prípade, že nemáme vybratú konkrétnu kategóriu, vyhľadávanie funguje v celej aplikácii. Pokiaľ však máme vybratú konkrétnu kategóriu, vyhľadávanie sa aplikuje len na zvolenú kategóriu.

Tretí typ vyhľadávania je vyhľadávanie kľúčových slov pomocou konjunktívnej normálovej formy. Toto vyhľadávanie sa síce používa v tomto komponente, no jeho implementácia sa nachádza v komponente CNFvyhladavanie.

8.2.3 Komponent CNFvyhladavanie

V tomto komponente implementujeme vyhľadávanie kľúčových slov na základe konjuktívnej normálovej formy. Formula v konjunktívnej normálovej forme je konjunkcia postupnosti klauzúl v nasledujúcom tvare, kde symboly x predstavujú klauzuly

$$X_1 \wedge X_2 \wedge ... \wedge X_n \tag{8.1}$$

Klauzula je disjunkcia postupnosti literálov v tvare

$$Y_1 \vee Y_2 \vee ... \vee Y_n \tag{8.2}$$

Literál v našom prípade predstavuje objekt typu *{ slovo, neg, podslova }*, kde hodnota slovo predstavuje id hľadaného slova. Hodnota neg môže nadobúdať hodnoty true, false a hodnota pod slova bude obsahovať idčká všetkých pod slov daného slova.

Toto vyhľadávanie implementujeme nasledovne: Máme jedno pole. Toto pole obsahuje ďalšie polia, ktoré predstavujú jednotlivé klauzuly. Každá klauzula obsahuje prvky poľa, ktoré predstavujú literály.

V prípade, že chceme hľadať záznamy, kde sa dané slovo nevyskytuje, tak priradíme hodnote neg hodnotu true. V našej aplikácii platí, že kľúčové slová majú hierarchickú štruktúru. To znamená, že slová majú stromovú štruktúru s odkazom na svojho rodiča. Teda v prípade, že budeme hľadať výskyt kľúčového slova, ktoré obsahuje ďalšie tri pod slová. Tak budeme hľadať také záznamy, ktoré obsahujú aspoň jedno akékoľvek slovo z danej hierarchie. V prípade negácie sa budú hľadať záznamy, v ktorých sa nevyskytuje ani jedno slovo z danej hierarchie.

Pri každom pridaní kľúčového slova sa vytvorí databázový dopyt na základe poľa klauzúl a zobrazia sa nám záznamy spĺňajúce podmienky, ktoré sme uviedli v konjunktívnej normálovej forme.

8.2.4 Komponent PridajZaznam

Tento komponent je zodpovedný za pridávanie záznamov do databázy. Na začiatku si načítame z databázy kategóriu spolu s jej atribútmi, ku ktorej chceme pridať záznam. Všetky dáta týkajúce sa konkrétneho záznamu bu-

deme mať uložené v premennej *formData*. Tieto dáta budeme získavať z komponentu ZobrazFormular, kde budeme dané dáta zadávať a po zadaní ich budeme posielať tomuto komponentu cez props.

8.2.5 Komponent ZobrazFormular

V tomto komponente na začiatku načítame dáta. Dáta budeme čítať podľa toho, či tento komponent komunikuje s komponentom PridajZaznam alebo UpravZaznam. Túto informáciu máme uloženú v premennej props.update. Všetky dáta obsahujúce informácie o zázname sú uložené v premennej *formData*.

Následne vygenerujeme formulár, ktorý bude pozostávať z atribútov danej kategórie. Okrem atribútov kategórie bude formulár obsahovať aj tlačidlá na pridanie externých linkov, interných linkov, komentárov a súborov. Všetky zmeny ukladáme do premennej *formData*. Následne tieto zmeny posielame svojim rodičovským komponentom, a to buď komponentu PridajZaznam, alebo UpravZaznam. Keď klikneme na tlačidlo *Ulož*, zavolá sa metóda rodičovských komponentov, ktorá dáta uloží do databázy.

Ohľadom interných linkov je treba spomenúť, že majú veľmi dôležitú funkciu. Prostredníctvom nich zabezpečujeme prepojenia medzi záznamami. Táto vlastnosť je jedna z kľúčových vlastností našej aplikácie. Na základe pridania takéhoto odkazu na iný záznam sa vytvorí prepojenie medzi záznamami. V princípe ide o sémantickú sieť pojmov, čiže istú formu reprezentácie poznatkov. Pri pridávaní sa zobrazí možnosť vybrať kategóriu a potom vybrať záznam z danej kategórie. Toto prepojenie bude potom uložené v tabuľke interných linkov.

V tomto komponente využívame taktiež komponent KlucoveSlova na priraďovanie kľúčových slov k záznamu.

8.2.6 Komponent KlucoveSlova

V rámci tohto komponentu zobrazujeme kľúčové slová. Tieto slová zobrazujeme na základe hierarchie. To znamená, že slová, ktoré sú potomkovia iných slov, sú mierne odsadené od iných slov, aby bolo možné jasne vidieť, ktoré slová sú ich rodičia.

Na začiatku si načítame kľúčové slová. Každé slovo sa skladá z nasledujúcich atribútov *id, nazovKlucovehoSlova, Rodic, checked, deti* Túto štruktúru vytvárame na serverovej časti aplikácie. Následne rekurzívne prejdeme tieto slová a zobrazíme ich v stromovej štruktúre. Každé slovo zobrazujeme ako checkbox. V prípade zakliknutia daného slova sa toto slovo priradí k záznamu. Vtedy sa cez props pošle správa komponentu ZobrazFormular, ktorý dané slovo pridá k záznamu, s ktorým aktuálne pracujeme. Okrem toho je možné v tomto komponente aj pridať nové kľúčové slovo do databázy.

V tomto komponente taktiež zobrazujeme aj odporúčané kľúčové slová. Tie sa zobrazujú na základe dát uvedených vo formulári, ktoré sú poslané na backend, kde ich algoritmus vyhodnotí a pošle späť slová, ktoré sa najviac hodia k danému záznamu. Tieto slová sa nám posielajú z komponentu ZobrazFormular, ktorý pracuje s dátami daného záznamu. Opäť platí to, čo pri ostatných kľúčových slovách - v prípade označenia sa slovo priradí k záznamu a pošle sa správa rodičovskému komponentu.

8.2.7 Komponent DetailZaznam

V tomto komponente zobrazujeme všetky atribúty konkrétneho záznamu spolu s jeho externými linkami, komentármi, kľúčovými slovami a súbormi. Každý súbor obsahuje link na stiahnutie daného súboru.

Na začiatku si načítame konkrétny záznam z databázy. Okrem neho si

načítame z tabuľky interných linkov aj záznamy, na ktoré sa tento záznam odkazuje. A následne si ešte načítame záznamy, ktoré sa odkazujú na tento záznam. Potom sa nám v strede okna zobrazia informácie o zázname. Na ľavej strane sa nám zobrazia všetky záznamy, ktoré sa na tento záznam odkazujú. Na pravej strane sa nám zobrazia záznamy, na ktoré sa odkazujeme z tohto záznamu. Medzi všetkými záznamami sa dá jednoducho preklikávať.

8.2.8 Komponent UpravKategorie

Tento komponent slúži na upravovanie konkrétnej kategórie. Používateľ tu môže zmeniť názov kategórie, atribúty. Prípadne pridať alebo vymazať konkrétny atribút.

Z databázy si načítame kategóriu, ktorú chceme upraviť spolu s jej atribútmi. Kategóriu načítame do premennej *kategoria* a jej atribúty do pomocnej premennej *atributy*. V tejto pomocnej premennej budeme mať pôvodné názvy atribútov spolu so všetkými pridanými a vymazanými atribútmi, zatiaľ čo zmenené budeme ukladať do premennej *kategoria*. Následne vygenerujeme formulár, v ktorom bude názov kategórie a všetky jej atribúty. Formulár bude obsahovať aj možnosť pridania nového atribútu.

8.2.9 Spojenie

Táto trieda slúži na komunikáciu medzi frontendom a backendom. Všetko, čo sa posiela medzi týmito vrstvami, je implementované tu. Pomocou modulu axios posielame na backend požiadavky, na základe ktorých potom načítavame dáta z databázy, zapisujeme, upravujeme alebo mažeme dáta. Metódy tejto triedy používame naprieč všetkými komponentami.

9

Nasadenie do prevádzky

Naša aplikácia je pripravená na použitie. Aplikáciu je možné spustiť v dvoch režimoch. Vo vývojárskom, v prípade že by sme chceli niečo zmeniť v zdrojovom kóde. A v produkčnom kde sa už nedá meniť nič v zdrojovom kóde.

9.1 Produkčný režim

Na spustenie aplikácie v produkčnom režime sa treba riadiť nasledujúcimi krokmi.

- Java 17 Na počítači, na ktorom bude aplikácia spustená musí byť nainštalovaná Java s verziou minimálne 17.
- Mysql Treba mať nainštalovanú verziu Mysql 8.0.33, kde je potrebné vytvoriť používateľa s menom user a heslom user. Následne treba danému používateľovi povoliť všetky práva. Potom je potrebné vytvoriť databázu s názvom pracovnaplocha.
- java -jar pracovnaplocha.jar Ďalej je potrebné spustiť tento príkaz v konzole.

- V poslednom kroku je potrebné pridať do tabuľky odporucane riadok s nasledovnými dátami (1, -1, -1, 1)

V tomto súbore pracovnaplocha.jar je naprogramovaná celá aplikácia, spolu s konfiguráciou používateľa aj databázy. Aplikácia bude fungovať na adrese `http://localhost:8072`.

9.2 Vývojársky režim

V prípade, že chceme spustiť aplikáciu vo vývojárskom režime je nutné najskôr splniť prvé dva kroky, ktoré sme opísali nižšie. Okrem nich je nutné mať tiež nainštalovanú najnovšiu verziu Node.js. Následne stačí súbor diplomovaPraca.zip rozbaliť.

V rozbalenom súbore sa nachádzajú priečinky frontend a backend. Priečinok backend otvoríme napríklad v nástroji Intellij alebo v akomkoľvek inom vývojovom prostredí, ktoré podporuje javu a dajú sa v ňom spúšťať aplikácie. Následne spustíme aplikáciu.

Po vykonaní tohto kroku sa nám vytvoria v databáze tabuľky. Teraz tak ako v predchádzajúcom prípade pridáme do tabuľky odporucane riadok (1, -1, -1, 1)

Potom sa v konzole presunieme do priečinku frontend a spustíme príkaz npm install. Počkáme, kým sa príkaz dokončí a spustíme príkaz npm start.

Ak sme vykonali korektne všetky kroky aplikácia by sa mala spustiť v prehliadači na adrese `http://localhost:3000`.

10

Záver

Cieľom tejto diplomovej práce bolo od začiatku vyvinúť aplikáciu, ktorá bude slúžiť na ukladanie informácií rôzneho typu. V ktorej sa budú tieto informácie jednoducho a efektívne vyhľadávať pomocou viacerých spôsobov. Dôležitou súčasťou práce bol aj výskum, v ktorom sme skúmali algoritmy, ktoré by navrhovali používateľovi vhodné kľúčové slová na základe dát, ktoré používateľ zadal.

Vo výskume sme navrhovali a skúšali algoritmy, ktoré by vhodným spôsobom na základe podobností dát v databáze s práve pridávaným záznamom navrhli kľúčové slová k danému záznamu. Tieto algoritmy sme medzi sebou porovnávali a hľadali najvhodnejší z nich.

Pri vývoji sme použili technológiu React pre tvorbu používateľského rozhrania, túto technológiu sme vybrali aj preto, lebo dokáže jednoducho spolupracovať s inými technológiami. Na prácu na serverovej časti aplikácie sme použili technológiu Spring Boot. Výhodou tejto technológie bolo najmä to, že má v sebe vstavaný aplikačný server Apache Tomcat. Vďaka ktorému sme mohli aplikáciu jednoduchšie testovať.

Podarilo sa nám vyvinúť aplikáciu, ktorá spĺňa požiadavky uvedené v špe-

cifikácii. Konkrétne je možné v tejto aplikácii pridávať a editovať záznamy jednotlivých kategórií. Vyhľadávať záznamy fulltextovo alebo pomocou kľúčových slov. Pridávať k záznamom kľúčové slová, komentáre, url linky, prílohy. Okrem toho aplikácia umožňuje ďalej prepájať jednotlivé záznamy medzi sebou a dynamicky prideľovať kľúčové slová na základe podobnosti záznamov. Aplikácia obsahuje aj kalendár do ktorého je možné pridávať udalosti. Okrem toho sa v nej nachádza niekoľko už dopredu vytvorených kategórií. Zoznam týchto kategórií nie je konečný. Používateľ si môže priamo v aplikácii zadefinovať vlastnú kategóriu, ktorej môže jednoducho definovať atribúty podľa svojej potreby. Vďaka tomu môže aplikáciu používať širšie spektrum ľudí na ich vlastné účely. Aplikácia je nainštalovaná na školskom serveri. Aplikácia je open source a jej zdrojový kód sa nachádza na tejto stránke `https://github.com/drab12/diplomovaPraca`

Táto webová aplikácia síce spĺňa požiadavky, ktoré sme si stanovili. Bolo by však vhodné pokračovať v jej vývoji a obohatiť ju tak o novú funkcionalitu. Napríklad prepojiť kalendár s google kalendárom, mať možnosť zdielať záznamy s inými ľuďmi pomocou url a ďalšie iné. Aplikácia by tak mohla slúžiť nielen výskumníkom, ale i študentom a iným ľudom, ktorí pri svojej profesii pracujú s dátami.

Literatúra

[Boo] Modal. `https://getbootstrap.com/docs/4.5/components/modal/`. Accessed mar. 25, 2023 [Online].

[GDK18] Kavya Guntupally, Ranjeet Devarakonda, and Kenneth Kehoe. Spring boot based rest api to improve data quality report generation for big scientific data: Arm data center example. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 5328–5329, 2018.

[Gre17] Bc. Miroslav Gregorec. Pracovná plocha výskumníka. diplomová práca, UNIVERZITA KOMENSKÉHO V BRATISLAVE, 2017.

[GZ19] Michal Gajewski and Wojciech Zabierowski. Analysis and comparison of the spring framework and play framework performance, used to create web applications in java. In *2019 IEEE XVth International Conference on the Perspective Technologies and Methods in MEMS Design (MEMSTECH)*, pages 170–173, 2019.

[Hin19] J. Hinkula. *Hands-On Full Stack Development with Spring Boot 2 and React: Build Modern and Scalable Full Stack Applications*

Using Spring Framework 5 and React with Hooks, 2nd Edition. Packt Publishing, Limited, 2019.

[jav] Spring boot react project - full stack web application tutorial. `https://www.javaguides.net/2021/10/spring-boot-react-full-stack-web-development-tutorial.html`. Accessed apr. 17, 2022 [Online].

[LL19] Samuel Pascal Levin and Michael Levin. Managing ideas, people, and projects: Organizational tools and strategies for researchers. *iScience*, 20:278–291, 2019.

[Mat11] Katarína Matysová. Pracovná plocha výskumníka. bakalárska práca, UNIVERZITA KOMENSKÉHO V BRATISLAVE, 2011.

[rea] React – a javascript library for buildinguser interfaces. `https://reactjs.org/`. Accessed jan. 09, 2023 [Online].

[SN13] Jatsada Singthongchai and Suphakit Niwattanakul. A method for measuring keywords similarity by applying jaccard's, n-gram and vector space. 2013.

[spr] Spring boot. `https://spring.io/projects/spring-boot/`. Accessed jan. 10, 2023 [Online].

[TO21] Cătălin Tudose and Carmen Odubăşteanu. Object-relational mapping using jpa, hibernate and spring data jpa. In *2021 23rd International Conference on Control Systems and Computer Science (CSCS)*, pages 424–431, 2021.

[WGPR+10] Paul Warren, Jose Manuel Gomez-Perez, Carlos Ruiz, John Davies, Ian Thurlow, and Igor Dolinsek. Context as a tool for organizing and sharing knowledge. In *CIAO@ EKAW*, 2010.

Zoznam obrázkov

4.1 Ukážka programu Evernote zdroj: `https://evernote.com/intl/cs` 16
4.2 Ukážka programu OneDrive zdroj: `https://www.microsoft.com/sk-sk/microsoft-365/onedrive/online-cloud-storage` 17
4.3 Ukážka programu GoogleDrive zdroj: `https://www.google.com/drive` 18

5.1 Ukážka query 23

6.1 Graf závislosti vhodne odporúčaných slov od počtu ponúkaných slov 28
6.2 Graf celkového priemerného počtu úspešne odporúčaných kľúčových sumy a maxima slov pre 20 záznamov 37
6.3 Graf priemerného počtu úspešne odporúčaných kľúčových slov sumy a maxima pre jeden záznam 38
6.4 Graf celkového priemerného počtu odporúčaných kľúčových slov sumy a maxima pri testovaní a trénovaní naraz 39
6.5 Graf celkového priemerného počtu úspešne odporúčaných kľúčových slov sumy a priemeru pre 20 záznamov 40
6.6 Graf priemerného počtu úspešne odporúčaných kľúčových slov sumy a priemeru pre jeden záznam 41

6.7 Graf celkového priemerného počtu odporúčaných kľúčových slov sumy a priemeru pri testovaní a trénovaní naraz 42
6.8 Graf celkového priemerného počtu úspešne odporúčaných kľúčových slov sumy a rozšírenej sumy pre 20 záznamov 43
6.9 Graf priemerného počtu úspešne odporúčaných kľúčových slov sumy a rozšírenej sumy pre jeden záznam 44
6.10 Graf celkového priemerného počtu odporúčaných kľúčových slov sumy a rozšírenej sumy pri testovaní a trénovaní naraz . . 45
6.11 Graf celkového priemerného počtu úspešne odporúčaných kľúčových slov rozšírenej sumy a rozšíreného maxima pre 20 záznamov 46
6.12 Graf priemerného počtu úspešne odporúčaných kľúčových slov rozšírenej sumy a rozšíreného maxima pre jeden záznam . . . 47
6.13 Graf celkového priemerného počtu odporúčaných kľúčových slov rozšírenej sumy a rozšíreného maxima pri testovaní a trénovaní naraz 48
6.14 Graf celkového priemerného počtu úspešne odporúčaných kľúčových slov rozšírenej sumy a rozšíreného priemeru pre 20 záznamov 49
6.15 Graf priemerného počtu úspešne odporúčaných kľúčových slov rozšírenej sumy a rozšíreného priemeru pre jeden záznam . . . 50
6.16 Graf celkového priemerného počtu odporúčaných kľúčových slov rozšírenej sumy a rozšíreného priemeru pri testovaní a trénovaní naraz 51
6.17 Graf porovnania rozšírenej sumy a rozšírenej sumy s JNLA pre celkové priemerné počty odporúčaných slov pre 20 záznamov 52

7.1 Ukážka hlavnej stránky aplikácie 57

7.2 Ukážka pridávania záznamov 60
7.3 Ukážka pridávania záznamov 60
7.4 Ukážka editovania kategórie 62
7.5 Ukážka pridávania kategórie 63
7.6 Ukážka detailu záznamu 64
7.7 Databázový model aplikácie 67

8.1 Ukážka metódy Controllera 69