Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií

Projektová dokumentácia

Tím 05: TextMania

Akademický rok 2018/19
Predmet Tímový projekt

Študenti Bc. Dávid Csomor
Bc. Adam Ďuriš
Bc. Júlia Krajčoviechová
Bc. Peter Križan
Bc. Alan Kováč
Bc. Daniel Kováč
Bc. Patrik Melicherík
Bc. Krištof Orlovský

Vedúci tímu Ing. Miroslav Blšták, PhD.

Obsah

Dokumentácia k riadeniu

Úvod

Dokument je výstupom práce na tímovom projekte po zimnom semestri, teda v piatich šprintoch. V úvode dokumentu sú bližšie predstavení členovia tímu spolu s rolami (oblasťami), ktorým sa v rámci tímového projektu venujú. Ďalšia časť dokumentácie k riadeniu je zameraná na opis manažmentu v tíme. Bližšie opisuje ako sú organizované jednotlivé časti manažmentu (komunikácia, dokumentácia, plánovanie, úlohy, výmena dát, kvalita). Taktiež obsahuje vyhodnotenie absolvovaných šprintov spolu s podrobným prehľadom naplánovaných a vykonaných úloh rozdelených do jednotlivých šprintov. Dokument tiež obsahuje globálnu retrospektívu, ktorá opisuje najzásadnejšie identifikované problémy, s ktorými sme sa počas práce na projekte ako tím stretli. Dôležitou súčasťou sú tiež metodiky v rôznych oblastiach, podľa ktorých sa pri vývoji riadime.
S využitím agilného vývoja sme sa ako tím stretli po prvý krát, preto na sebe neustále pracujeme a snažíme sa čo najviac priblížiť štandardom, ktoré by mali byť podľa agilného vývoja dodržané.

1. Predstavenie členov tímu

Ing. Miroslav Blšták, PhD.
Zastupuje rolu product owner(a) a tiež vedúceho tímu. Na jednotlivých stretnutiach diskutujeme predstavy, ktoré zo strany zákazníka má a smerovanie práce na projekte usmerňuje zadávaním úloh, na ktorých počas šprintov pracujeme.

Bc. Dávid Csomor
Zaoberá sa dátovou analýzou a metódami strojového učenia v jazyku python. Vo voľnom čase sa zaoberá vývojom custom ROM pre platformu android. V rámci tímového projektu sa venuje prevažne analýzam a ich dokumentácii, dokumentácii stretnutí, dokumentácii a problematike spojenej s jazykom python.

Bc. Adam Ďuriš
Mimo školských povinností sa primárne zaoberá vývojom backendu v jazyku Java s použitím frameworkov Spring a Hibernate. Okrem toho sa zaujíma o oblasť dátovej analýzy a strojového učenia. V tímovom projekte sa zväčša venuje vývoju frontendovej časti aplikácie, nakoľko by aj v tejto oblasti rád nabral praktické skúsenosti.

Bc. Daniel Kováč
Zaoberá sa vývojom backendu v jazykoch Java a JavaScript s využitím rámcov Spring a Express. Taktiež sa venuje vývoju webových aplikácií. Má znalosti v oblasti administrácie softvéru na operačných systémoch Linux. V tímovom projekte sa venuje administrácii databázy MongoDB aj servera, na ktorom je projekt nasadený. Taktiež vyvíja backend a občas implementuje aj menšie časti frontendu.

Bc. Alan Kováč
V tomto projekte sa venuje najmä vývoju backend časti v programovacom jazyku JavaScript. Jeho úlohou je zabezpečenie fungovania a prístupu do databázy MongoDB v rámci aplikácie, a taktiež tvorba zložitejších dopytov nad dátami. Stará sa o architektúru systému a skladanie komponentov do jedného celku. Mimo školy pracuje ako backend developer v programovacom jazyku Java.

Bc. Júlia Krajčoviechová
Zaujíma sa najmä o oblasť strojového učenia, dátovej analýzy a testovania softvéru. Z programovacích jazykov preferuje najmä Python. V rámci tímového projektu sa zaoberá najmä tvorbou dokumentácie a dohľadom nad potrebnými odovzdaniami spojenými tiež s účasťou v súťaži TP Cup. Zapája sa tiež do implementácie častí projektu v jazyku Python.

Bc. Peter Križan
Venuje sa vývoju aplikácií v NET pričom štúdium smeruje do oblasti umelej inteligencie. Z jazykových znalostí prevažuje C#, Java a Python, pričom disponuje aj základnou znalosťou HTML a CSS. V rámci tímového projektu sa zaoberá backendovou časťou a integráciou Python funkcionalít.

Bc. Patrik Melicherík
Primárne sa venuje hlavne frontendových technológiám a grafickému dizajnu. Má znalosť v HTML, CSS, JavaScript, rámci Angular. Tiež sa venuje backendovým technológiám založeným na jazyku JavaScript. V rámci tímového projektu sa podieľa na tvorbe frontendu a grafického dizajnu.

Bc. Krištof Orlovský
V projekte sa venuje poväčšine backendu v jazyku JavaScript a interakcii s databázou. Medzi ďalšie jazykové znalosti patrí v poslednej dobe C++, Java a iné OO jazyky. Disponuje tiež základnými znalosťami v oblasti tvorby webov. Štúdium má zamerané na problematiku umelej inteligencie.

Tabuľka: Percentuálny podiel autorov pri tvorbe dokumentácie

Časť dokumentácie	Dávid	Adam	Alan	Daniel	Júlia	Peter	Patrik	Krištof
Predstavenie členov tímu	11%	11%	11%	11%	22%	11%	11%	11%
Aplikácie manažmentov				10%	80%		10%	
Prehľad šprintov a retrospektíva	20				80%			
Metodiky	20%			5%	10%	20%	45%	
Zápisnice zo stretnutí	70%	5%	5%	5%			5%	10%
Dokumentácia k inžinierskemu dielu			16%	16%	5%	16%	31%	16%

2. Aplikácie manažmentov

Manažment komunikácie

Zodpovedná: Júlia Krajčoviechová
Metodika: Metodika komunikácie

Komunikáciu je možné rozdeliť na spoločné stretnutia a komunikáciu mimo stretnutí. Na spoločných stretnutiach diskutujeme o úlohách a častiach, na ktorých tím pracoval a o vykonanej práci. V úvode stretnutia je vyhradený priestor pre diskusiu ohľadom podnetov spojených s organizáciou a iných pripomienok. Časť stretnutia je vyhradená na plánovanie. Komunikácia mimo stretnutí je zabezpečená formou online komunikačného nástroja Slack. Pre prehľadnosť komunikácie sú vytvorené komunikačné kanále podľa tematických oblastí. Do online komunikácie je tiež zapojený aj vedúci nášho projektu. Pre formálnu komunikáciu bol zriadený tímový e-mail. Pre neformálnu komunikáciu sa využíva aj skupinový chat na Facebooku.

Manažment dokumentácie

Zodpovedný: Dávid Csomor
Metodika: Metodika dokumentácie

Dokumentácie sú priebežne vytvárané počas jednotlivých stretnutí. Konkrétne sú vytvárané zápisnice zo stretnutí spolu s vedúcim projektu a zápisnice zo šprintov (na konci šprintov). Na každom stretnutí je určená zodpovedná osoba, ktorá sa venuje tvorbe zápisnice podľa dohodnutej formy. Tieto zápisnice sú potom sprístupnené aj na webovej stránke tímu.

Manažment výmeny dát (informácií)

Zodpovedný: Adam Ďuriš
Metodika: Metodika výmeny dát

Dôležitou časťou tímovej práce je výmena (zdieľanie) dát a informácií. Dáta zahŕňajú najmä zdrojové kódy, ktoré sú zdieľané formou Git repozitára . Kód je organizovaný formou vetiev, ktoré sú rozdelené podľa jednotlivých user story. Nad všetkými vetvami je hlavná - master vetva. Ako úložisko pre rôzne dokumenty a materiály spojené s projektom je určené úložisko Google Disk, kde sú dokumenty organizované do priečinkov podľa jednotlivého tematického zamerania.

Manažment plánovania

Zodpovedný: Peter Križan
Metodika: Metodika používania a správy TFS

Úlohy sú plánované na každom spoločnom stretnutí, pričom sú zaraďované do jednotlivých šprintov. Vedúci projektu vytvorí features, na základe ktorých sa odvíja plánovanie úloh. Úlohy pre jednotlivé šprinty sú plánované s určitým časovým odhadom, aby boli v rámci jedného šprintu splniteľné. Taktiež sa kladie dôraz aj na to, aby mal každý člen tímu pridelenú prácu na nasledujúce obdobie. Po naplánovaní šprintu každý člen tímu rekapituluje čomu sa bude venovať a v čom spočíva jeho úloha.

Manažment úloh

Zodpovedný: Krištof Orlovský
Metodika: Definition of Done, Definition Of Ready

Po spoločnej konzultácii a naplánovaní častí, ktoré sa budú implementovať a riešiť počas šprintu, spoločne vytvoríme user stories pre konkrétny šprint. Pre každú user story je určený riešiteľ, resp. zodpovedná osoba v prípade, že na jednej user story bude pracovať viacero členov tímu. Taktiež priradíme story pointy využitím metódy planning poker. Riešitelia si neskôr samostatne vytvoria potrebné tasky pre riešenie danej story.

Manažment prehliadok kódu

Zodpovedný: Daniel Kováč
Metodika: Metodika prehliadok kódu

Pri realizovaní pull requestov sa uskutočňujú prehliadky kódu vždy, v menšej miere sa prehliadky realizujú aj behom samotnej implementácie. Pri prehliadke kódu je najdôležitejšie zvoliť osobu, ktorá bude prehliadku realizovať. Vždy musí mať znalosť v danej technológii, ktorú ide kontrolovať na nadpriemernej úrovni vrámci tímu. Časté prehliadky kódu vedú ku kvalitnejšiemu kódu. Manažovanie prehliadok sa realizuje s pomocou nástroja TFS, ktoré umožňuje vykonať prehliadku kódu a pridávanie komentárov priamo v jeho rozhraní.

Manažment verziovania

Zodpovedný: Patrik Melicherik
Metodika: Metodika manažmentu verzií

Ako verziovací systém bol zvolený Git repozitár nástroja TFS, ktorý využívame na manažment úloh. A to z dôvodu jeho priameho prepojenia s manažmentom úloh. Týmto spôsobom je jednoduchšie vyhľadať jednotlivé vetvy k ich príslušným úlohám. Verziovanie poskytuje niekoľko vetiev, z ktorých hlavnými sú live a master. Na vetve live beží aplikácia s produkčnou databázou, ktorá je dostupná product ownerovi na prehliadanie. Vetva master slúži na development a pre interné vývojárske účely, táto vetva nie je prístupná product ownerovi. Ostatné vetvy sú vytvárané podľa zvoleného štandardu (10293-create-inverted-index-api) a každá vetva sa dá mapovať na User Story. Samotný nástroj na verziovanie je možné zvoliť podľa vlastných preferencií, nutné je však dodržiavať štandard písania commit správ v tvare (#10293 Create inverted index api).

Manažment konfigurácie softvérového systému

Zodpovedný: Alan Kováč
Metodika: Metodika konfigurácie softvérového systému

Tento manažment je zodpovedný za navrhovanie a používanie komponentov (hardvérových a softvérových častí). Zodpovedá za správnosť integrovania špecifickej množiny technológií a nástrojov do jedného celku. V rámci týchto artefaktov pevne definuje vlastnosti a vzťahy, ktoré určujú presný cieľ používania.
Roľa v tomto tíme zabezpečuje správne používanie procesov súvisiacich s konfiguráciou serveru, operačným systémom (server, sieť), kontinuálnou integritou, programovacím jazykom alebo inými vlastnosťami spojené s výkonnosťou (rýchlosť, pamäť), či riešením problémov architektúry. Spolu s rolou vznikla aj metodika pre nasadzovanie aplikácie do produkčného prostredia v časti Nasadenie aplikacie. Táto časť je realizovaná pomocným nástrojom (zabezpečuje automatické nasadenie pomocou pull do *Live* vetvy). Tento nástroj má zjednodušiť, zrýchliť a zdokumentovať proces nasadzovania. Manažment je zodpovedný za stav počas behu servera sledovaním funkčnosti produkčného prostredia, aby aplikácia bola dostupná zákazníkovi.

3. Prehľad šprintov

Počas 12. týždňov semestra sme absolvovali 5 šprintov. Posledný (5.) šprint bol ukončený 12.12.2018.

Úvodný šprint (25.9.2018 - 2.10.2018, dĺžka šprintu: 1 týždeň), ktorý do celkového počtu šprintov nezapočítavame, bol zameraný najmä na oboznámenie sa s problematikou, ktorou sa zaoberáme. Tento šprint tiež zahŕňal úvodné stretnutia, kde sme si dohodli spôsoby komunikácie, plánovania, zvolili sme nástroj na manažment úloh a podobne. Prvotné kroky boli tiež zamerané na vytvorenie tímového webu a plagátu. V neskoršej fáze 0. šprintu sme sa zamerali na analýzu a návrh, ako bol napríklad návrh štruktúry databázy, analýza nástrojov pre spracovanie textov či analýza možností importovania článkov. Rozhodli sme sa v akom jazyku budú implementované jednotlivé časti projektu a aká bude celková architektúra systému.

Šprint č. 1: 3.10. 2018 - 17.10.2018 (dĺžka šprintu: 2 týždne)

V tomto šprinte sme zamerali na finalizáciu analýz, ktoré začali už v úvodnom šprinte. Z administratívnej časti sme sa zamerali na lepšiu organizáciu TFS a tiež na prepojenie Gitu s TFS. Skompletizoval sa návrh architektúry systému, čoho výstupom je aj diagram, ktorý navrhnutú architektúru zobrazuje. Z implementačnej stránky sme sa zamerali na tvorbu rozhrania, proces importu dát a anotáciu tokenov. Časť šprintu bola venovaná štúdiu MongoDB a príprave API pre prístup do databázy. Rozhodli sme sa tiež pre účasť v súťaži TP Cup, čo vyžadovalo vytvorenie dokumentu slúžiaceho ako prihláška.
Náš prvý šprint hodnotíme pozitívne, nakoľko boli splnené všetky naplánované úlohy, pričom sa nám tiež podarilo nasadiť náš projekt na server, na ktorom je prístupný.

Šprint č. 2: 18.10.2018 - 24.10.2018 (dĺžka šprintu: 1 týždeň)

Z organizačných dôvodov bol tento šprint stanovený iba na 1 týždeň. Väčšina úloh bola zameraná na implementáciu ďalších častí projektu. Konkrétne sme sa venovali anotácií článkov, vytvoreniu konkrétnych datasetov naplnenými článkami a formátovaniu článkov, s ktorými neskôr budeme pracovať. Pre dostatočný počet zdrojov sme tiež vytvorili crawler pre sťahovanie článkov z webovej stránky webnoviny.sk. Časť tímu sa venovala tvorbe invertovaného indexu a prototypu rozhrania pre invertovaný index.
Väčšina úloh, aj napriek miernej časovej tiesni, bola splnená.

Šprint č. 3: 25.10.2018 - 13.11.2018 (dĺžka šprintu: 3 týždne)

Tento šprint bol z dôvodu nesplnenia naplánovaných úloh predĺžený z pôvodných 2 týždňov na 3 týždne. Časový sklz vznikol z dôvodu týždňa určeného na samoštúdium, počas ktorého sa nám nepodarilo zorganizovať spoločné stretnutie, na ktorom by sme na úlohách pracovali. Po predĺžení časového rozmedzia sa nám však už úspešne podarilo splniť väčšinu z naplánovaných úloh.
Zameranie úloh bolo opäť skôr na implementáciu ďalších častí projektu. Pokračovali práce na skripte pre invertovaný index. Upravili sme formátovanie článkov a tiež upravili implementáciu crawleru pre sťahovanie článkov z webovej stránky webnoviny.sk, aby bola funkcionalita 100% spoľahlivá. Časť tímu sa opäť venovala frontendovej časti, kde bolo doplnené rozhranie pre invertovaný index, zobrazenie analyzovaných článkov a bola upravená celková vizualizácia rozhrania. V neposlednom rade prebiehali práce spojené s nasadením aktuálnej verzie projektu na server a implementácia iných doplnkových častí pre prepojenie jednotlivých modulov projektu.

Šprint č. 4: 14.11.2018 - 27.11.2018 (dĺžka šprintu: 2 týždne)

V 4. šprinte sa tím venoval práci na kompletizácii dokumentácií potrebných pre odovzdanie ako 1. kontrolný bod po absolvovaní 3. Šprintov. Mimo tvorby dokumentácie však tiež pokračovali aj implementačné práce na projekte. Zamerali sme sa na oddelenie verejnej funkcionality našej aplikácie od vývojárskej časti pre zamedzenie prístupu bežných používateľov k častiam, ktoré sú určené a potrebné výlučne pre vývoj. Časť tímu sa tiež venovala sprehľadneniu používateľského rozhrania, čo spočívalo najmä v pridaní rôznych filtrov, stránkovania a spätných referencií. Taktiež sme sa zamerali na zobrazenie analyzovaných tokenov, aby naše výsledky boli viditeľné a preukázateľné aj pre vedúceho nášho projektu.
S lepšou časovou organizáciou sa nám podarilo tento šprint úspešne ukončiť a dodržali sme všetky stanovené úlohy a termíny (napriek miernej časovej tiesni spôsobenej zlou organizáciou tvorby dokumentácie).

Šprint č. 5: 28.11.2018 - 12.12.2018 (dĺžka šprintu: 2 týždne)

V poslednom šprinte zimného semestra sme sa zamerali najmä na naplnenie databázy dostatočným počtom článkov, ktoré budú neskôr využité pre ďalšiu prácu. Taktiež sme sa venovali odstráneniu vetkých chybných a nefunkčných častí projektu, aby verzia, ktorú odovzdávame ako výstup zo zimného semestra bola plne funkčná a mohli sme na vyvíjaní bez problémov pokračovať v letnom semestri. Časť tímu sa venovala v pokračovaní prác spojených s invertovaným indexom. V neposlednom rade bolo potrebné skompletizovať finálnu verziu dokumentácie pre zimný semester.
V tomto šprinte sme sa snažili skôr o odstránenie nedostatkov a doladenie detailov existujúcich častí projektu, než implementovať nové časti. Väčšina prác bola splnená a dokončená včas.

Prehľad všetkých úloh jednotlivých šprintov je bližšie uvedený formou exportu z nástroja TFS v kapitole 7.

4. Globálna retrospektíva

Pri každom stretnutí tímu je vytvorený priestor na pripomienky a postrehy. Pri väčšine stretnutí nevznikajú rozsiahlejšie diskusie s pripomienkami, čo môže byť spôsobené aj tým, že sa ako tím poznáme už dlhšiu dobu a dokážeme sa efektívne zorganizovať a spolupracovať s rovnakým cieľom - dokončiť všetky úlohy.

Počas jednotlivých šprintov sa však vyskytujú určité nedostatky, na ktorých eliminácii pracujeme (väčšina nedostatkov už bola odstránená).

- Väčšina komunikácie prebieha prostredníctvom FB a nie Slacku, ktorý je na online komunikáciu primárne určený
- Je potrebné lepšie prerozdelenie úloh medzi členov tímu
- Úlohy sa niekedy riešia na poslednú chvíľu (pred spoločným stretnutím/pred ukončením šprintu)
- V TFS sú nekompletné opisy k úlohám, nedoplnený čas strávený riešením úlohy a priebežne neaktualizovaný stav úlohy

Pozitívne hodnotíme schopnosť pracovať ako tím a zapálenosť tímu pre prácu na projekte o čom svedčí aj fakt, že väčšinu úloh sa nám podarí splniť na čas. Po 12 týždňoch semestra môžeme vidieť reálne výsledky, ktoré nie sú zanedbateľné a vidieť za nimi veľký kus odvedenej práce.
Hoci sa v začiatočných týždňoch semestra stávalo, že úlohy boli prerozdelené neprimerane (niektorí členovia tímu mali robiť všetko, niektorí sa nevedeli zapojiť), tento problém bol odstránený a momentálne pri každom plánovaní prebieha zhrnutie úloh, pričom každý člen tímu zrekapituluje na čom bude pracovať, prípadne komu vie s prácou pomôcť. Je však potrebné zlepšiť morálku niektorých členov a predísť tak problémom s časovým nedostatkom na splnenie úlohy.
Postupom času eliminujeme nedostatok v administratívnej stránke projektu, ktorý sa prejavoval v nekompletných opisoch úloh v nástroji na manažment úloh (TFS). Opisy jednotlivých úloh sú podrobnejšie vytvárané už počas plánovania. Je však potrebné zapracovať ešte na priebežnom aktualizovaní stavu úlohy a evidencii času pre riešenie úlohy.

5. Motivačný dokument

Predstavenie tímu

Náš tím sa skladá z 8 členov a dovoľujeme si povedať, že je funkčným celkom. Väčšina z nás sa pozná už od prvého ročníka štúdia na bakalárskom stupni na FIIT STU a nových dvoch členov sme s radosťou prijali tiež. Dokážeme spolupracovať a aj sa ľudsky podporiť. Sme vskutku pracovití a súdržné jadro tímu funguje skupinovo už skoro štyri roky. Najsilnejšou stránkou nášho tímu je naše odhodlanie učiť sa nové technológie a robiť veci najlepšie ako dokážeme.
Ako tím číslo 5 sme našli mnoho tém, ktoré nás zaujali, či už oslovili väčšiu alebo menšiu časť tímu. Po kolektívnej diskusii sme sa rozhodli pre tri témy, ktoré nás, ako tím najviac zaujali, či už z hľadiska zručností, predchádzajúcich skúseností alebo celkových preferencií tematického zamerania.
Jednohlasne sme sa zhodli, že nás veľmi zaujíma koncept strojového učenia. Viacero našich členov (Júlia, Peter, Adam, Dávid) sa s oblasťou strojového učenia a spracovania dát zaoberali vo svojich bakalárskych prácach. Patrik má výrazné skúsenosti s oblasťou vývoja frontendových aplikácií, dlhšiu dobu sa venuje oblasti tvorby samotných webových aplikácií. Čo sa frameworkov týka, Adam má skúsenosti s Angularom, aj keď vo väčšej miere sa venuje backendu a samotnej Jave. Alan počas tvorby svojej bakalárskej práce zase nadobudol skúsenosti s databázovými systémami, presnejšie s MongoDB. Krištof je zdatný, okrem iného, v rôznych technických oblastiach, ktoré tiež nadobudol pri realizovaní bakalárskej práce zameranej na prevod kódu do grafického zobrazenia. S jazykom PHP sa vo väčšej či menšej miere stretli Patrik aj Daniel, ale problém by nám neurobila ani implementácia backendovej časti v inom jazyku. Oblasti backendových technológií sa venuje v podstate väčšina tímu. Napríklad v Jave, C#, JavaScripte alebo Pythone, so všetkými týmito jazykmi máme skúsenosti. Medzi ďalšie zručnosti, ktoré sa oplatí spomenúť platí databáza PostgreSQL, v ktorej sa obzvlášť vyzná Daniel, avšak určité zručnosti má aj zvyšok tímu.
Každý z členov tímu má zapísané aj určité užitočné predmety, ktoré sú vhodné pre nami preferované témy. Všetci absolvujeme povinné predmety Objavovanie znalostí a Architektúra softvérových systémov. Júlia a Dávid majú zapísaný predmet Vyhľadávanie informácií. Väčšina tímu bude absolvovať predmet Objektovo orientovaná analýza a návrh softvéru. Taktiež niektorí členovia absolvujú predmet ako napríklad Neurónové siete či Aspektovo-orientovaný vývoj softvéru. Veríme, že každý člen tímu bude nápomocný a svoje poznatky nadobudnuté absolvovaním vymenovaných predmetov užitočne využije pri práci na tímovom projekte.

E-mailový kontakt na tím: team5fiit@gmail.com

Motivácia

Téma č. 1: Prostredie pre inteligentnú analýzu textov

Hlavnou motiváciou pri výbere tejto témy je orientácia sa viacerých členov v kľúčovej problematike. Viacerí členovia majú reálne skúsenosti s podobným typom práce hlavne z prostredia bakalárskych projektov. Peter má skúsenosti s analýzou a klasifikáciou textu v podobe zdrojových kódov a teoretické poznatky v oblasti lexikálnych diel a ich možnej klasifikácie do užších celkov. Nechýbajú ani hlbšie znalosti z oblasti strojového učenia (Peter, Adam, Dávid, Júlia) za použitia knižníc (scikit-learn, graphviz, plot). Júlia môže tiež prispieť znalosťami so spracovaním objemnejších dátových sád (Pandas, numpy). Za výsledné prostredie bude zodpovedný Patrik, ktorý má bohaté skúsenosti i grafické cítenie v tejto oblasti. Daniel spolu s väčšinou tímu má bohaté backendové zručnosti a Alan má priame skúsenosti s používaním MongoDB. Krištof je zdatný v rôznych technických oblastiach, ale aj v oblasti automatizácie.

Hlavným cieľom je vytvorenie funkčného nástroja pre analýzu textov, ktorý nám pomôže prehĺbiť naše doterajšie znalosti v danej problematike a umožní priniesť jedinečné riešenie pre analýzu textov v slovenskom jazyku. V rámci tímu by sme sa radi bližšie venovali triedeniu článkov do tematických kategórii, prípadne určovanie vhodnosti textu pre publikum, nakoľko filtrácia nevhodného textového obsahu pre slovenský jazyk výrazne absentuje.

Téma č. 2: Analýza správania sa vozidiel v meste

Druhá nami preferovaná téma sa znova dotýka technológií, ktoré nám ani zďaleka nie sú vzdialené. Téma Analýza správania sa vozidiel v meste sa úzko dotýka internetu vecí. IoT bolo predmetom bakalárskej práce Alana a v menšej miere aj Daniela. Ostatné možné technológie nám sú blízke tiež, ako sme už spomenuli pri predstavení tímu. Taktiež máme zapísané predmety, ktoré sú pre riešenie danej témy užitočné.

Problematika IoT je dnes rozšírená a preto nás daná téma prirodzene zaujala, ale hlavne pozitívne hodnotíme snahu o využitie tejto technológie pre zlepšenie stavu spoločnosti. Táto téma nás oslovila tiež najmä preto, lebo zbieranie a spracovanie dát z fyzického prostredia je nesmierne dôležité pre dnešnú automatizáciu sveta ako napr. v doprave. Veríme, že projekt s týmto zámerom má potenciál aj mimo tímového projektu na FIIT a má pomôcť dopravnej situácií na Slovensku. Doposiaľ z bakalárskej práce a predmetu PSI máme skúsenosti len s komunikáciou IoT zariadení cez MQTT (websocket), HTTP (rest služby). Motiváciou je aj zoznámenie a zdokonaľovanie sa v nových veciach. Ak by sme dostali možnosť pracovať na tejto téme, nezabránila by nám ani neznalosť IoT LoRa a knižníc na spracovanie obrazu TensorFlow, či OpenCV.

Téma č. 3: Databanka otázok a úloh

Jednou z tém, ktorá sa dostala do nášho výberu, je Databanka otázok a úloh. V databáze PostgreSQL sa obzvlášť vyzná Daniel. Ostatné technológie potrebné pre túto tému poznáme tiež, ale to sme už

spomínali v predstavení tímu. Z užitočných predmetov majú niektorí členovia vybraný predmet Objektovo orientovaná analýza a návrh systémov. Predmet Pokročilé databázové systémy nemá zapísaný nik z nás, ale v práci s databázami sme nadpriemerne pokročilí, čiže to nepovažujeme za problém.

Táto téma zahŕňa zaujímavé technológie, ktoré obsahujú jednak backendovú, ale aj frontendovú stránku. V oboch týchto oblastiach sú členovia nášho tímu zdatní a skúsení. Myšlienka e-vzdelávania je sama o sebe pre nás zaujímavá a v dnešnej dobe je považovaná za žiadanú. Zaujíma nás jednak myšlienka témy, ale aj samotné technológie. Ďalším bodom je práca na už existujúcom projekte, do ktorého môžeme reálne prispieť. Motiváciou je aj príspevok ku zvýšeniu kvality vzdelávania ďalších generácií. Keďže táto téma hovorí o vývoji webovej aplikácie, s ktorou väčšina z nás má skúsenosti je pre nás ešte zaujímavejšia, keďže v dnešnej dobe sa väčšina aplikácií snaží migrovať práve na web. Preto považujeme túto problematiku za modernú a pre nás ešte príťažlivejšiu.

6. Metodiky

6.1. Metodika komunikácie

Autor: Júlia Krajčoviechová

Táto metodika slúži na usmernenie komunikácie medzi členmi tímu, nakoľko komunikácia je dôležitá súčasť vývoja, ktorá každopádne nemôže byť podcenená alebo vynechaná. Komunikáciu rozdeľujeme na spoločné stretnutia a online komunikáciu.

Spoločné stretnutia

Opakujú sa pravidelne každý týždeň s dĺžkou trvania cca 3 hodiny. V stredu od 12:00 do 15:00 je vyhradený čas na spoločné stretnutie spolu s vedúcim projektu, ktoré prebieha v miestnosti 4.26 na Fakulte informatiky a informačných technológií STU BA. Spoločné stretnutie členov tímu (bez vedúceho projektu) sú organizované v utorok od 11:00 do 14:00.

Online komunikácia

Počas práce na projekte je samozrejme potrebné komunikovať aj mimo dohodnuté časy spoločných stretnutí. Pre online komunikáciu využívame nástroj **Slack**, v ktorom sú vytvorené viaceré komunikačné kanály pre lepšiu organizáciu komunikácie (a predídenie straty dôležitých informácií).

Vytvorené komunikačné kanály sú:

- **General:** určené pre všeobecné informácie a požiadavky, v rámci ktorých taktiež funguje tzv. Slack reminder, ktorý pripomína nutnosť napísať zápisnicu zo šprintu a zapísať hodiny
- **Backend:** vetva určená na komunikáciu ohľadom častí týkajúcich sa backendu systému
- **Frontend:** pre komunikáciu ohľadom frontendovej časti systému
- **Email:** vetva kde sa kopírujú e-maily prijaté na spoločnom e-maily, nakoľko po prečítaní e-mailu jedným členom tímu dochádzalo k tomu, že ostatní členovia tímu si mail nevšimli (nakoľko už bol prečítaný)
- **Documentation:** pre otázky a diskusiu ohľadom dokumentácie k projektu
- **Mongo:** vetva pre diskusiu ohľadom databázy
- **Tp_cup:** pre zdieľanie informácií a pripomienok týkajúcich sa účasti v súťaži Tp Cup
- **Tfs:** vetva pre automatické informovanie o zmenách vykonaných v nástroji TFS
- **Tfs-git:** vetva pre automatické informovanie o zmenách vykonaných v Gite

Pre komunikáciu je tiež vytvorený aj skupinový chat na Facebooku, ktorý však nie je určený ako primárny komunikačný prostriedok!!!

Pre formálnu komunikáciu je zriadený spoločný tímový e-mail: team5fiit@gmail.com.

6.2. Metodika manažmentu úloh

Autor: Krištof Orlovský

Pri inicializácii prvého šprintu je vytvorený zo strany product ownera backlog obsahujúci témy, niekedy tiež epiky, či priamo už user stories, ktoré sú momentálne esenciálne a priamo zrejmé pre vytvorenie cieľového produktu. Nejedná sa o poslednú verziu a stav product backlogu, keďže tento backlog je ownerom rozšírený vždy po ukončení šprintu, v ktorom tím docielil splnenie viac ako polovice epikov.

Na začiatku každého šprintu, sú definované aktuálne požiadavky product ownera na funkcionalitu a zmeny aplikácie. Tieto požiadavky sa častokrát týkajú rôznych častí aplikácie (frontend, backend) a tiež rôznych oblastí. Ak sú niektoré z týchto požiadaviek stále iba vo forme epikov, celý tím v spolupráci s konzultujúcim ownerom rozdelí epik na jednu až viaceré user stories. Vhodnosť user stories určíme, prípadne upravíme kontrolou podľa Definition of Ready (skrátene DoR). Manažment úloh na týchto stretnutiach má na starosti scrum master. Scrum master má zodpovednosť rozdeliť user stories medzi členov tímu na základe oblasti, ktorej sa týkajú, týchto členov označujeme za koordinátorov user story. Po tom ako sú user stories rozdelené medzi zodpovedných koordinátorov, tí definujú jednotlivé úlohy a rozdelia ich ostatným členom, ktorí na konkrétnom user story budú spolupracovať. Za korektné ukončenie user story je zodpovedný koordinátor určený scrum masterom na začiatku šprintu.

Zároveň s vyššie opísaným rozdeľovaním stories a úloh prebieha aj odhad hodnôt story pointov, časového rozsahu, priority story a následne sa doplnia priority a odhadované trvania aj úlohám. O hodnotách story pointov sa hlasuje na stretnutí.

Nasleduje priebeh šprintu, kedy člen tímu vypracováva zadané úlohy, čo zahŕňa už spomenuté samotné vypracovanie úlohy, tvorbu požadovanej dokumentácie, zálohovanie a distribúciu riešenia pomocou metodiky pre verziovanie a v neposlednom rade aktualizáciu stavov úlohy. Od povahy úlohy môžu vyplývať aj iné povinnosti. Stavy úloh/taskov riešiteľ nastavuje nasledovné:

- Pre úlohy
    - New - nová úloha, už je pridelený zodpovedný člen, no na vypracovaní úlohy ešte nezačal pracovať
    - Active - na úlohe sa pracuje
    - Closed - práca na úlohe ukončená, splnené možné podmienky DoD

- Pre user story

- New - nová user story, podobne ako úloha ale na vyššej úrovni abstrakcie. Na jej úlohách sa ešte nezačalo pracovať a zároveň koordinátor ešte nemonitoroval stav úloh
- Active - user story obsahuje aktívne úlohy a prebieha ich monitorovanie
- Resolved - Úlohy v user story ukončené, vykonané možné práce nad DoD
- Closed - uzavreta product ownerom po splnení celého DoD zoznamu.

Na osobnom stretnutí, ktoré sa koná dvakrát počas šprintu, scrum master prejde jednotlivé úlohy a zistí od zodpovedných členov, v akom stave sa úlohy momentálne nachádzajú. Ak nastáva situácia, že niektorá úloha vyzerá že sa nestihne dokončiť, automaticky alarmuje koordinátora user story, aby dohliadol na korektné ukončenie úlohy. Uzatváranie úloh a zmenu ich stavu rieši zodpovedná osoba, ale označenie user story za vyriešenú má na zodpovednosti koordinátor.

Na najbližšom šprinte sa na základe pravidiel Definition of Done (skrátene DoD) rozhodne, či sú úlohy považované za uzatvorené, alebo je potrebné ich ešte upraviť, aby spĺňali DoD.

Zahŕňa procesy:

- Definovanie úlohy
- Analýza úlohy
- Zaznamenanie úlohy do nástroja na to určeného (TFS)
- Definovanie pravidiel DoR (definition of ready) a DoD (definition of done)
- Vyvodenie pod-úloh vyplývajúc z danej úlohy
- Odhad času, náročnosti a priority úlohy
- Voľba zodpovednej osoby za úlohu
- Rozdelenie pod-úloh ostatným členom tímu
- Vypracovanie úlohy
- Umiestnenie výstupu úlohy do nástroja na to určeného (TFS, Git)
- Označenie úlohy za zapracovanú
- Prehliadka a overenie správnosti úlohy
- Označenie úlohy za vyriešenú podľa DoD, alebo znovu otvorenie úlohy

Definition of done na úrovni úlohy/task-u:

- Dodané riešenie (spravidla v podobe kódu) pre požadovaný problém
- Vyjadrenie zodpovedajúceho, že predpokladajú splnenie user story
- Splnené akceptačné kritéria
- Vykonaná code review s následnými opravami a refactoringom
- Realizácia úlohy je dodaná do projektu na testovacom prostredí
- Projekt je možné bezproblémovo „buildnúť“
- Otestovaná funkcionalita buď iným členom/členmi tímu, alebo v prípade komplexnej funkcionality realizátor odprezentoval funkčnosť niekomu v tíme

Definition of done na úrovni epiku, témy, či feature:

- Splnené všetky DoD spojené s prislúchajúcimi user stories

Definition of done na úrovni šprintu:
- Splnené všetky DoD spojené s user stories zahrnutými v šprinte
- Vyhotovené všetky user stories v šprinte
- V prípade potreby je update-nutý product backlog (splnenenie vyše polovice epikov)
- Všetky v minulosti identifikované bugy sú opravené

Definition of done/ready na úrovni user story:
Zvyšné Definition of Done/Ready, boli podstatnejšie a obsahovo plnšie, preto je pre nich vytvorený priestor na úrovni metodík.

6.3. Metodika prehliadok kódu

Autor: Daniel Kováč

Prehliadky kódu prebiehajú pomocou dvoch známych metód. Prvou metódou je bežná prehliadka kódu cez plece, kedy skúsenejší programátor z tímu dohliada nad kódom menej skúseného programátora a upozorňuje ho na chyby a konvencie, ktoré treba dodržiavať. Toto nielen, že šetrí čas, ale zároveň poskytne menej skúsenému programátorovi lepšiu spätnú väzbu a priestor na otázky. Druhou metódou je prehliadka kódu podporená softvérovým nástrojov (konkrétne TFS - Git).
Po ukončení práce na úlohe zodpovedná osoba umiestni kód do Git repozitára. Tu pri commite prebieha kontrola pomocou 2 linterov, ktorá vykoná základnú prehliadku kód z pohľadu syntaktických konvencií. Pokiaľ jeden z linterov (pre JavaScript knižnica StandardJS, pre Python knižnica Autopep8) nájde syntaktické chyby, ktoré dokáže opraviť, tak ich opraví. Na ostatné chyby, ktoré nedokáže opraviť upozorní používateľa a nedovolí uskutočnenie commitu.
Následne, po úspešnom commite programátor požiada o pull request (zlúčenie svojej vetvy s hlavou, master vetvou). Na pull request je označený minimálne jeden posudzovateľ, ktorý nemôže byť zároveň žiadateľom o pull request. Posudzovateľ má za úlohu posúdiť kód a pridať komentáre, či už k chybám, alebo návrhom na úpravu, ak je to potrebné. Pokiaľ nejaké komentáre vzniknú, je žiadateľ povinný opraviť/ upraviť svoj kód a znovu ho umiestniť do repozitára, dovtedy pull request nie je schválený.
Prehliadka kódu cez plece funguje vo forme osobnej konzultácie. Skúsenejší programátor sa pozerá na menej skúseného behom implementácie. Počas tohto procesu zvykne prerušiť menej skúseného programátora ak zbadá nejaký nedostatok v jeho kóde (štylistická chyba ako napr. názov premennej, logická chyba ako napr. nesprávne podmienky alebo iná chyba ako napr. nesprávny HTTP status alebo nedostatočné logovanie). Menej skúsený programátor tak behom takéhoto programovania píše kvalitnejší kód a učí sa na vlastných chybách. Taktiež sa vyvinú kvalitnejšie riešenia, keďže dvaja programátori dokážu intenzívnym konzultovaním dospieť k efektívnejším riešeniam.

Zahŕňa procesy:

- Práca na úlohe
- Dokončenie časti funkcionality
- Commitnutie výslednej funkcionality do nástroja na to určeného (Git)
- Žiadosť o pull-request
- Prehliadka kódu posudzovateľom
- Vyhodnotenie prehliadky kódu
- Zapracovanie pripomienok
- Opätovná prehliadka
- Označenie úlohy za vyriešenú
- Zlúčenie časti funkcionality s hlavnou vetvou

6.4. Metodika verziovania

Autor: Patrik Melicherik

Na verziovanie kódu používame TFS repozitár založený na princípe Git. Hlavnou vetvou je master, kde sa nachádza vždy funkčný kód, ktorý je prezentovaný product ownerovi. Ostatné vedľajšie vetvy sú určené na vývojové účely. Vetvy sú vytvárané na základe user-stories.

Konvencie vytvárania vetiev:

- na začiatku obsahuje "issue number" user-story, ktorej sa týka
- pomocou oddeľovačov "-" nasleduje výstižný názov user-story v anglickom jazyku

príklad: *9987-create-admin-zone*

Nástroj na prístup k repozitáru nebol priamo určený. Odporúčané bolo CLI, ale u väčšiny sa presadil nástroj SourceTree z dôvodu, že poskytoval lepší prehľad nad dianím sa v repozitári a zjednodušoval s ním prácu.

Zakázané operácie:

- zákaz pushovania kódu do master vetvy (master bol označený ako chránený, teda nikto doň nemôže pushovať)
- zákaz schvaľovania vlastných pull requestov (pri schvaľovaní bola nastavená možnosť, aby žiadateľ o pull request nemohol schváliť vlastný pull request)
- zákaz používania prepínača -f , --force

Základné príkazy

git clone <url-na-repozitar> = tu sú dve možnosti, buď Http alebo SSH (na SSH je potrebne mat v Gite pridaný SSH key, ten sa da lahko vygenerovať -> googli 'generate SSH key'; Http vyžaduje autentifikáciu pomocou prihlasovacích údajov do Gitu)

git pull = stiahne zmeny z remote vetvy a dá ich dokopy s lokálnou vetvou (tu je nutné aby vaša lokálna vetva sledovala tú správnu remote vetvu).
git pull origin <nazov-remote-vetvy> = rovnaké ako git pull, ale stiahne zmeny a zlúči ich dokopy s konkrétnou remote vetvou
git add <cesta-k-suborom> = pridá súbory na zadanej ceste do trackovaných súborov, a tým pádom budú commitnuté do remote vetvy
git commit -m "CISLO_ISSUE Add api route for articles" = pridá správu, k vášmu commitu (správa musí byť výstižná a stručna, prvé slovo musí byť ideálne jedno z "Add, Remove, Update, Fix" - prvé písmeno za číslom issue musí byť kapitálka)
git push = umiestni vaše zmeny z lokálnej vetvy do remote vetvy a označí ich commit správou, ktorú ste uviedli v predchádzajúcom kroku

Ako riešiť konflikty

Program VS Code po vykonaní git pull zobrazí v sekcii Git súbory, ktoré boli aktualizované a stiahnuté z remove vetvy. Niekedy sa stane, že sa objavia konflikty, pokiaľ nastali zmeny v rovnakých súboroch a Git to nedokáže rozumne vyriešiť. Konflikty je nutné riešiť manuálne. V prípade, že sa nevyznáte / nie ste si istý, ktorú verziu kódu použiť, odporúčam kontaktovať autora kódu pre vyhnutie sa problémom.

Ako vyzerá pracovný postup

1. Cez Git UI vytvorím vetvu na základe master vetvy,
2. u seba lokálne zavolám príkaz git fetch , ktorý mi zosynchronizuje nové zmeny v gite a teda zobrazí novú branch, ktorú som vytvoril cez Git UI.
3. premiestnim sa na svoju vetvu pomocou príkazu git checkout <nazov-vytvorenej-remote-branche>
4. pracujem na svojej vetve a po dokončení určitej časti funkcionlity robím commit *(pozn.: commit nemusí byť do remote vetvy, stačí aj lokálne, tzn. bez príkazu git push)*
5. Keď dokončím svoju prácu, je čas umiestniť zmeny do remote vetvy
6. Vykonám príkaz git add <cesta-k-suborom> ,ktorý pridá všetky súbory, ktoré sú na danej ceste do sledovaných súborov
7. Následne pridám správu pre môj commit pomocou príkazu git commit -m "CISLO_ISSUE Add api route for articles", ktorá výstižne definuje moju prácu *(pozn.: ideálne aby commit bol tak veľký, aby ho vystihla jedna výstižná správa, SPRÁVA MUSÍ NA ZAČIATKU OBSAHOVAŤ ISSUE NUMBER A NÁSLEDNE MEDZERU A SPRÁVU SO ZAČIATOČNÝM VEĽKÝM PÍSMENOM)*
8. Teraz je nutné stiahnuť si lokálne aktuálne zmeny z master vetvy, tým sa predíde konfliktom pri pull requeste, toto vykonám príkazom git pull origin master
9. Ak pri kroku 8 nastali konflikty, treba ich vyriešiť
10. vykonám príkaz, aby som svoje zmeny dostal do remove vetvy git push *(pozn.: ak chcem vykonať push do inej vetvy je nutné za push uviesť ešte aj názov remote vetvy)*

6.5. Definition of ready na úrovni user story

Autor: Krištof Orlovský

Vzhľadom na nepresnú formuláciu prvotných úloh sme pristúpili k oficiálnej formulácií „definition of ready“, ktoré vyplynuli z retrospektívy niektorých (prevažne menej úspešných úloh). Tieto podmienky musí spĺňať každá novo vytvorená user story. Tieto definície sú bodovo zhrnuté nižšie:

Vzniknutá inšpiráciou z INVEST prístupu[6.2.1], častokrát používaného pri scrum-och kde odstránime písmenko I (independence) nezávislosť, ktoré by popisovalo nezávislosť User Stories, či aj Epikov, o ktorú sa budeme pokúšať, no nemôžeme ju v našom tímovom projekte zabezpečiť. Takže DoR znie nasledovne:

- N (negotiable) – user story by mali poskytovať priestor v správnych postupoch ich plnenia
- V (valuable) – prínos user story zákazníkovi by mal byť jasný
- E (estimable) – Veľkosťou porovnateľná s ostatnými user stories.
- S (small) - parametre pre user story, ako trvanie , priorita a pod. by mali byť určiteľné pred jej vykonaním. Taktiež by sa dané parametre mali zmestiť do rozsahu jedného šprintu.
- T (testable) – má definované akceptačné kritéria
- Ďalšie nami pridané body vykonané po NVEST:
    - S priradeným koordinátorom/reviewerom
    - Ohodnotená story pointmi
    - Priradené potrebné dodatočné výstupy (vizuálne, dokumentačné...)

[6.2.1] https://www.scrum.org/resources/blog/walking-through-definition-ready

6.6. Definition of done na úrovni user story

Autor: Krištof Orlovský

Úvodnými konzultáciami sa definovali rôzne stavy pre vytvárané user story. Pri vytvorení novej user story má stav "new" a po priradení a začatí riešenia danej úlohy sa prepne do stavu "Active". Spoločnou dohodou bola prijatá konvencia, kedy členovia tímu úlohu neuzatvárajú (nemenia stav na "Closed"). Pre označenie hotovej user story zo strany členov tímu bol vytvorený stav "Resolved". Stav "Resolved => Closed / "Resolved => Active" následne mení product owner na základe požiadaviek naň kladených. Pre elimináciu prípadov, kedy product owner musel opätovne otvoriť danú úlohu, boli prijaté nižšie uvedené pravidlá.

User story môže byť označená ako "RESOLVED" iba v prípade:

1. Splnené všetky DoD spojené s prislúchajúcimi úlohami/taskmi
2. Dodané riešenie (spravidla v podobe kódu) pre požadovaný problém
3. Vyjadrenie zodpovedajúcich, že predpokladajú splnenie user story
4. Ukončený refactoring
5. Spĺňa všetky akceptačné kritériá, ktoré boli zadefinované pri tvorbe danej US
6. Obsahuje náležitosti, ktoré boli definované pri tvorbe US (dokumentácie , diagramy, ..)
7. Všetky vytvorené úlohy/chyby pod touto US musia byť v stave "Closed"
8. V prípade implementovanej funkčnej časti systému musí prejsť testami / byť otestovaná nezávislým členom tímu v závislosti od charakteru US
9. V prípade implementovanej funkčnej časti systému, musí byť výsledok zlúčený v "master" vetve => musí byť otestovaná, musí prejsť code review, akceptovaný pull request minimálne dvoma členmi tímu. Funkcionality, ktoré nie sú v master vetve NESMÚ byť označené ako "Resolved".
10. Vykonané QA a docielený dostatočný výstup
11. Stav "Resolved" mení člen tímu zodpovedný za danú US po splnení všetkých predošlých kritérií.

6.7. Metodika dokumentácie

Autor: Dávid Csomor

Nakoľko je dokumentácia k inžinierskemu dielu výsledkom spojenia výstupov rôznych činností, treba každú z nich náležite zdokumentovať. Pod pojmom náležite myslíme tak, aby jednotlivé výstupy tak, ako aj výsledná dokumentácia, neobsahovali príliš málo ani príliš veľa informácií, aby boli náležite štruktúrované, aby mali prevažne jednotnú formu a riadili sa podobnými formátovacími pravidlami. Je taktiež dôležité, aby jednotliví členovia, ktorí sú zodpovední za tvorbu konkrétnych výstupov, ich vedeli aj správne umiestniť tak, aby bol dokument jasne označený a

dohľadateľný. Môžeme povedať, že pri tvorbe akéhokoľvek druhu dokumentácie spojenej s práve vytváraným inžinierskym dielom, riešime nasledujúce problémy:

1. Činnosť pri ktorej bol konkrétny výstup vytvorený (účel dokumentu)
2. Označenie, kategorizácia a umiestnenie výstupov

6.7.1. Dokumentácia z hľadiska účelu:

Z hľadiska účelu môžeme v našom prípade rozdeliť dokumentácie na tri kategórie, ktoré sa môžu aj prelínať, avšak je dobré pokiaľ sú zachytené samostatne:

- zachytávajúcu prevažne ľudské činnosti (správa o činnosti/ výstupoch)
- podrobnejšie opisujúcu určité funkčné celky (technická dokumentácia)
- určenú na prezentáciu, propagáciu ponúkaného riešenia alebo marketingové účely ("biznis dokumentácia")

Dokumentácia zachytávajúca činnosti:

Medzi takéto dokumentácie patria: zápisnice zo stretnutí s product ownerom, zápisnice z konzultácií na predmete MTS (Manažment v Tvorbe Systému) a zápisnice z tímového stretnutia. Takáto dokumentácia má zväčša mierne variabilnú štruktúru, nakoľko sa diskutované problémy môžu líšiť v charaktere, množstve a pošte pripomienok. Na nasledujúcom, nižšie uvedenom obrázku môžeme vidieť ukážku štruktúry dokumentu zo stretnutia tímu z product ownerom. Dokument je štruktúrovaný tak, ako to znázorňuje vpísaný ukážkový text. Hlavička dokumentu v sebe obsahuje nasledujúce štruktúrované údaje:

- Číslo zápisnice
- Dátum, čas začiatku a miesto stretnutia
- Mená zapisovateľa a prítomných, ktorí sa na stretnutí zapájali

Ďalším príkladom na takýto typ dokumentácie je aj výstup z konkrétneho šprintu, ktorý je však už svojím obsahom na hranici medzi dokumentáciou zachytávajúcou činnosti a technickou dokumentáciou.

Zápisnica X

Dátum	DD.MM.RRR – HH:MM
Miesto	BUDOVA, MIESTNOSŤ
Zapisovateľ	MENO PRIEZVISKO
Prítomní	MENO PRIEZVISKO 1
	MENO PRIEZVISKO 2

Úvod - pripomienky

- Úvodný prehovor jednotlivých členov tímu, ktorí sa postupne vyjadrujú k priebehu plnenia úloh, ktoré im boli pridelené
- Pripomienkovanie od product ownera a členov tímu
- Zhodnotenie stavu riešenia jednotlivých časti product ownerom
- Identifikácia relevantných činností pre nasledujúce obdobia

Úlohy

- Voľný opis úloh/ činností, ktoré je potrebné vykonať, a ktorých špecifické znenie je zadávané do vybraného nástroja na sledovanie vývoja

V čom pokračovať

- V prípade, že sa product ownerom alebo členmi tímu identifikuje činnosť prislúchajúca k jednej z týchto troch kategórií, je daná činnosť zaznamenaná. V opačnom prípade sa kategória ani nezapíše.

Čo prestať robiť

Čo začať robiť

Technická dokumentácia:

Takáto dokumentácia sa prevažne venuje opisu a hodnoteniu konkrétnych implementačných rozhodnutí, ich prepojeniu a fungovaniu. Ukážku takejto časti dokumentácie môžeme vidieť na nasledujúcom, nižšie uvedenom obrázku. Takýto dokument je v našom prípade kolaboratívne tvorený takým spôsobom, že jednotliví členovia tímu vložia podrobný opis svojej časti riešenia na miesta vytvorené a určené manažérom dokumentácie. Ďalším príkladom takejto dokumentácie môže byť napríklad inštalačná príručka.

3.Pravidlový lematizátor (Tvaroslovník) [Horváth]

- Založené na pozorovaní, že najväčší vplyv na ohýbanie slovenského slova v slovenskom jazyku ma jeho koncovka (suffix)
- Ten sa použije na výber šablóny, podľa ktorej sa upraví vstupné slovo.
- Nevýhodou tejto služby sú nepresnosti spôsobené prípadným nesprávnym výberom vhodnej šablóny.
- Výhodou je možnosť lematizácie všetkých slov, aj takých, ktoré nie sú v nejakom slovníku (nové slová, cudzie slová, ...).
- Web: http://text.fiit.stuba.sk/lemmatizer/
- 2 metódy
- Fast lemmatization - príklad

```
URL: http://text.fiit.stuba.sk:8080/lematizer/services/lemmatizer/lemmatize/fast
Method: POST
Headers: Content-Type: text/plain
Data: Už niekoľko rokov patrí Slovensko medzi krajiny s najvyšším počtom áut vyrobených na tisíc
obyvateľov.

Response
Data: už niekoľko rok patriť slovensko medzi krajina s vysoký počet auto vyrobený na tisíc obyvateľ
```

- Full lemmatization - príklad

```
URL: http://text.fiit.stuba.sk:8080/lematizer/services/lemmatizer/lemmatize/full
Method: POST
Headers: Content-Type: text/plain
Data: Mám psa.

Response
Data: <?xml version="1.0" encoding="UTF-8" standalone="yes"?><customLemmasHolders>
<customLemmasHolder><form>mám</form><lemmas><lema>mama</lema><rank>0</rank></lemmas><lemmas>
<lema>mať</lema><rank>0</rank></lemmas><lemmas><lema>mámiť</lema><rank>0</rank></lemmas>
</customLemmasHolder><customLemmasHolder><form>psa</form><lemmas><lema>pes</lema><rank>0</rank>
</lemmas><lemmas><lema>pes</lema><rank>0</rank></lemmas></customLemmasHolder></customLemmasHolders>
```

Biznis dokumentácia:

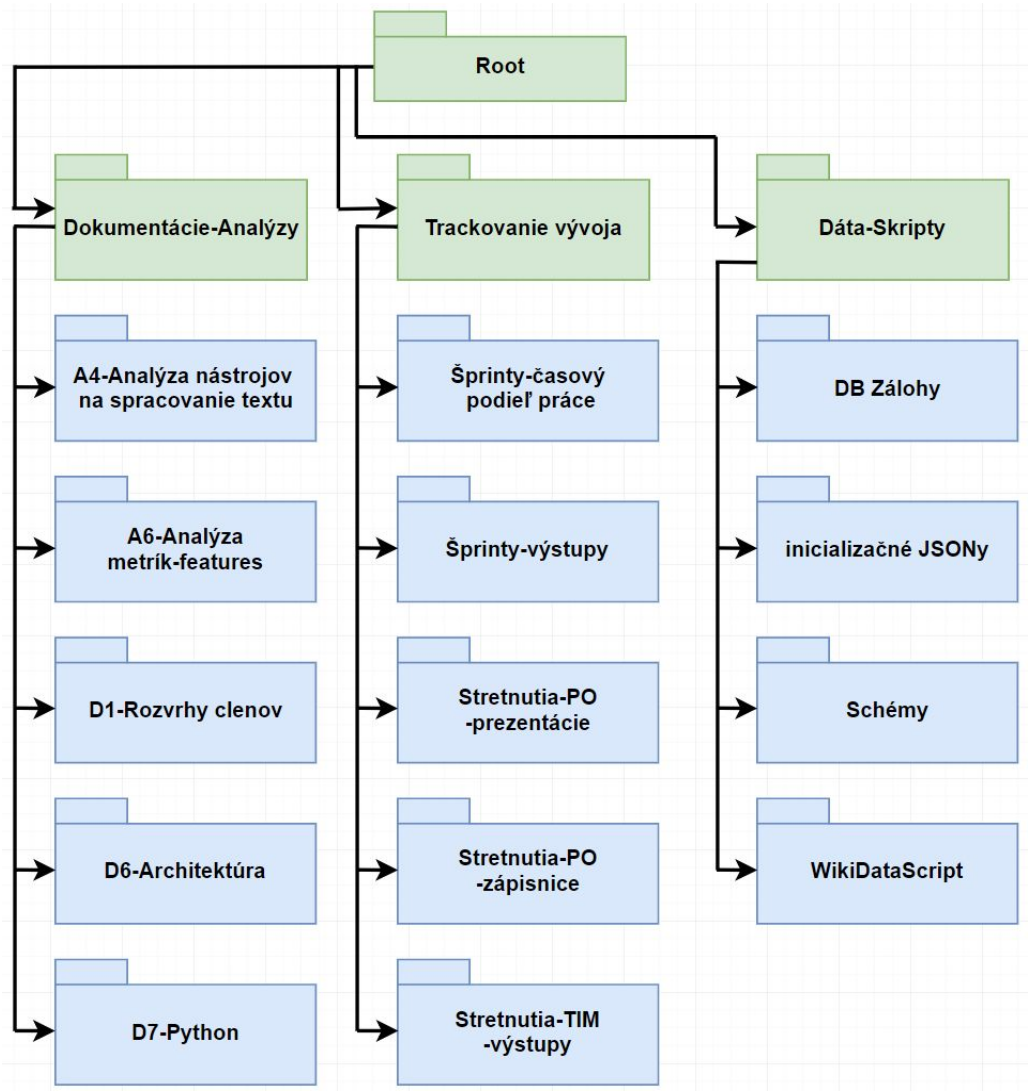
Do biznis dokumentácie môžeme zaradiť také dokumenty, ktoré majú istým spôsobom opisovať /prezentovať navrhované riešenie s marketingovým účelom, napríklad kontrakt medzi zadávateľom a vyhotoviteľom, zoznam požiadaviek, manažment plánovania, prezentácia riešenia a iné, ktoré bude vyhodnocovať potenciálny klient alebo zadávateľ. Súčasťou biznis dokumentácie môže byť v prípade agilného vývoja napríklad aj export z product backlogu (PB). Ukážku PB môžeme vidieť v nasledujúcej sekcii "export úloh z tfs".

6.7.2. Kategorizácia a označovanie dokumentácie

Dokumentácia má slúžiť nielen pre zadávateľa, ale môže byť veľmi užitočný nástroj aj pre samotných programátorov a ľudí podieľajúcich sa na vývoji poskytovaného riešenia. Pri väčších

projektoch sa môžu jednotlivé časti dokumentácie (medzi-výstupy) pomerne jednoducho stratiť a ich doh ľadanie v prípade potreby, môže mať za následok zbytočnú stratu času a frustráciu (v najlepšom prípade). V horšom prípade sa potýkame s chýbajúcou dokumentáciou, ktorú je v s odstupom času ťažšie dopracovať alebo s chýbajúcou dokumentáciou, ktorej absencia brzdí zvyšok tímu ak nastanú okolnosti, ktorých následkom je člen tímu indisponovaný a je potrebné zapracovať zmeny do nezdokumentovaného kódu/modulu, prípadne je potrebný daný celok komplexne použiť. Z toho dôvodu je ideálne, aby manažér dokumentácie vopred určil systém označovania (pomenovávania) a kategorizácie vznikajúcich dokumentov tak, aby boli jednotlivé časti dokumentácie jasne, jednoznačne pomenované a jednoducho doh ľadate ľné v prípade potreby. Dokumentácia a iné náležitosti sú uchovávané v zdie ľanom internetovom adresári od spoločnosti Google - Google Drive.

Postupne sme sa pre účely našej práce rozhodli rozdeliť dokumentáciu do nasledujúcej stromovej štruktúry:



Priečinok "**Dokumentácie-Analýzy**" obsahuje buď dokumentáciu už hotových častí nášho riešenia (čo označujeme prefixom "D"), alebo analýzu už existujúcej oblasti / riešení relevantných pre náš projekt (čo označujeme prefixom "A"). Číslo označuje poradie týždňa v semestri, kedy bol dokument vytvorený.

Priečinok "**Trackovanie vývoja**" je štruktúrovaný vzhľadom na účel dokumentu.

- "Šprinty-časový podiel práce" - obsahuje Google Spreadsheet-y odzrkadľujúce množstvo času a percentuálny podiel na výsledku úloh pre každého člena
- "Šprinty-výstupy" - obsahuje výstupy šprintov, v ktorých každý člen zhodnotí svoje úlohy a svoju prácu za daný šprint
- "Stretnutia-PO-prezentácie" - obsahuje prezentácie pre PO (product ownera) za uplynulý týždeň. Prezentácie konkrétne zoznam user stories a taskov, ktoré boli na daný šprint pridelené
- "Stretnutia-PO-zápisnice" - obsahuje zápisnice zo stretnutí tímu s product ownerom
- "Stretnutia-TIM-výstupy" - obsahuje výstupy za uplynulý týždeň (nie za šprint, ktorý má teoreticky variabilnú dĺžku trvania) spísané na stretnutí iba samotného tímu bez prítomnosti product ownera

Priečinok "**Dáta-Skripty**" obsahuje rôzne dátové štruktúry, zálohy, prvotne implementované skripty, navrhnuté schémy a iné elektronické materiály, ktoré tématicky spadajú pod danú kategóriu (názov priečinku) a už pre nás nie sú bezprostredne dôležité, ale stále užitočné za účelom zálohy, archivácie alebo dočasného úložiska.

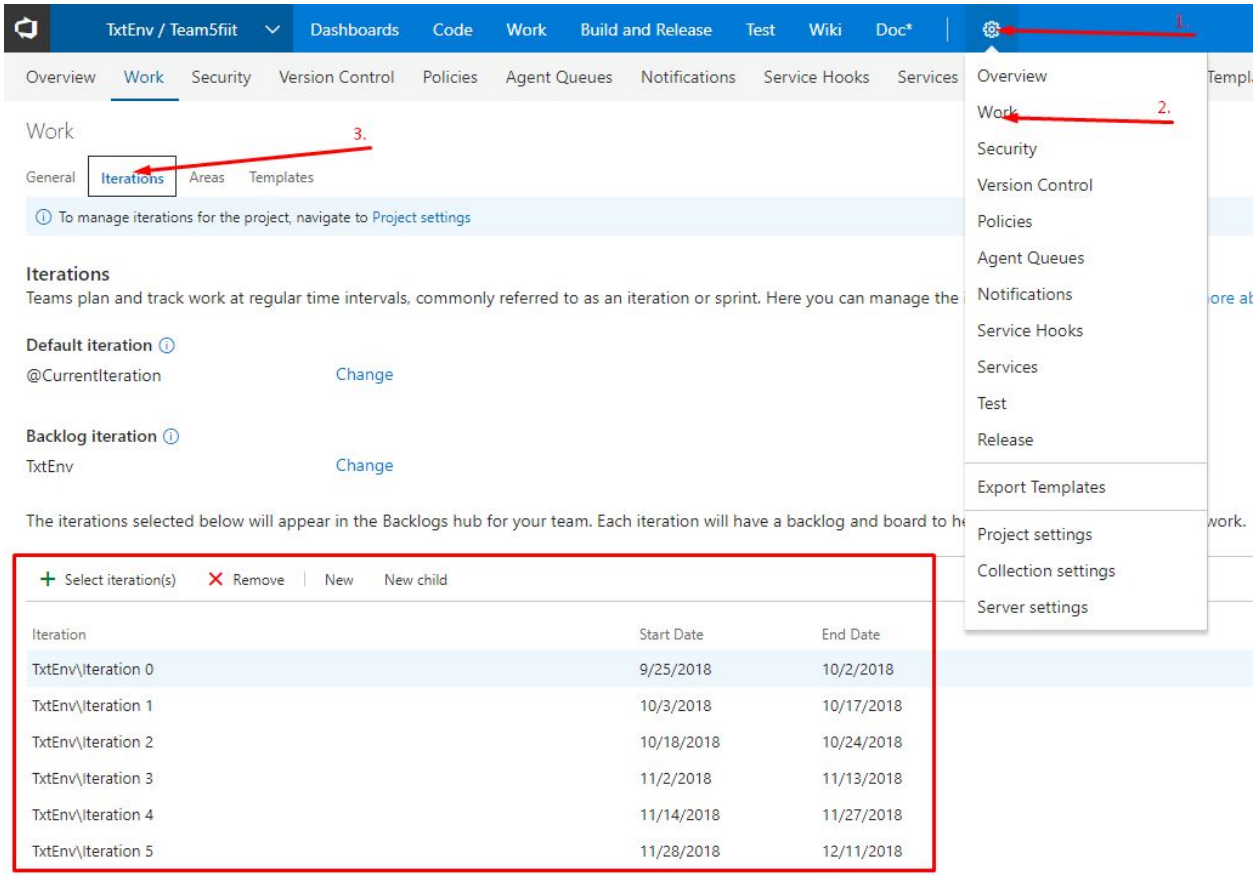
6.8. Metodika používania a správy TFS

Autor: Peter Križan

Táto metodika je určená pre všetkých členov tímu a opisuje spôsob nastavenia (úvodnej inicializácie) pri začiatku šprintu , správy jednotlivých úloh , a dodržiavanie kritérií, ktoré musia byť splnené za predpokladu efektívnej správy a dodržiavania stanoveného plánu. Jednotlivé body sú koncipované na prostredie nášho projektu a niektoré body sú silne závislé od danej domény a školských kritérií.

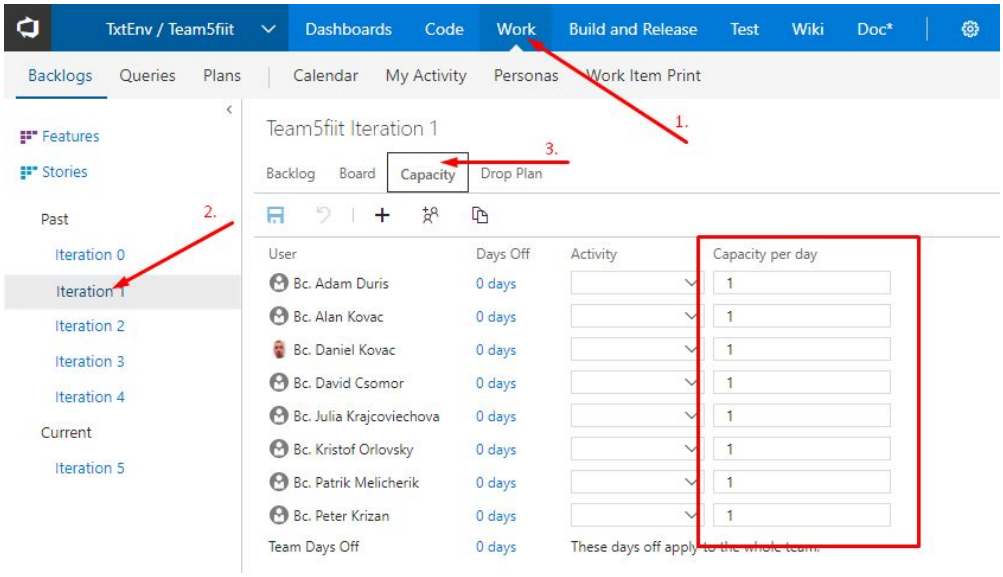
1. Inicializácia iterácií a dĺžka ich trvania

Tento proces môže byť jednorázový v prípade pevnej a striktne naplánovanej dĺžky jednotlivých šprintov na dlhšiu dobu. V našom prípade volíme metódu priebežnej tvorby šprintov na jeden šprint vpred. Proces vytvorenia a správy iterácii je možné nájsť v TFS pod záložkou Settings/Work/Iterations tak ako je znázornené na obrázku 1 kde orámovaná časť predstavuje editačnú oblasť jednotlivých šprintov.



2. **Nastavenie časovej kapacity jednotlivým členom tímu**

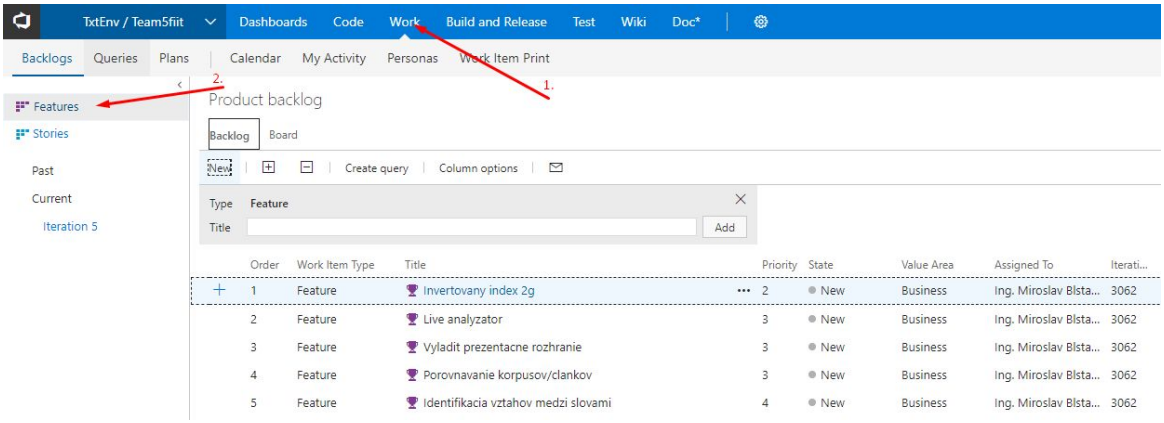
Toto nastavenie je kritické z pohľadu správneho plánovania sledovania priebehu daného šprintu (burndown chart). Nastavujú sa jednotlivé roly pre daný šprint (v našom projekte nezohľadňujeme) a časovú kapacitu jednotlivých členov tímu, ktorá v našom prípade predstavuje kritérium kreditového systému (7 kreditov ~ 14 hodinám za šprint aktívnej samostatnej práce, t.j. Denná kapacita jedného člena tímu je cca 1hodinu). Tieto nastavenia sa nastavujú v časti Work/Iterácia/Capacity obrázok 2.



3. Správa features (product backlog)

Správa tohto prostredia v našom projekte tvorí návrhy product ownera pre väčšie implementačné časti, ktoré si daný tím na začiatku šprintu analyzuje a realizuje výber najvhodnejších, čo predstavuje výber realizovaný s ohľadom na aktuálny stav projektu s požiadavkami product ownera.

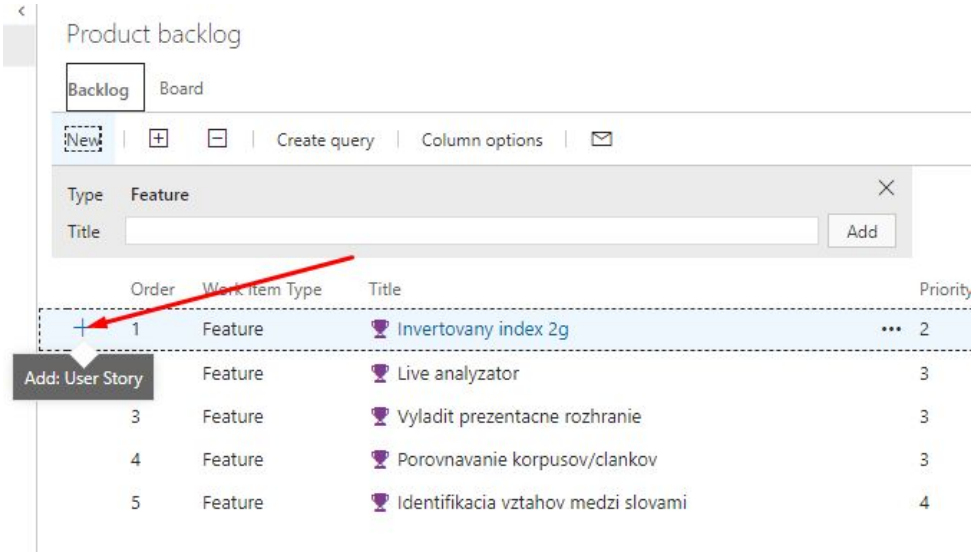
Jednotlivé Features sú prioritizované a obsahujú opis daného implementačného prípadne analyzujúceho celku. Podľa dohody na úvodnom stretnutí sa v našom projekte stará o Epic elementy sám product owner a iba on má výhradné právo uzatvoriť/vyhodnotiť epic ako hotový.



Správa user stories a úloh

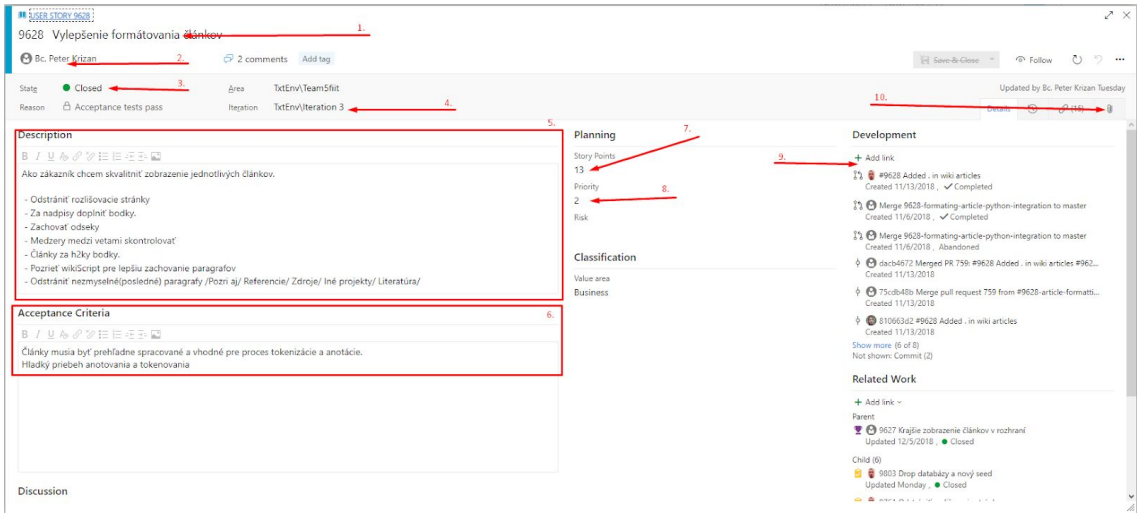
Správa tohto prostredia je zameraná na časový úsek kedy je daný epic vybraný v procese plánovania pre následnú realizáciu v najbližšom/najbližších šprintoch. Tvorba user story spočíva v previazaní daného epicu ako rodičovského elementu s daným konkrétnym user story. Na nasledujúcich obrázkoch je proces vytvorenia user story s previazaním na konkrétny epic. Tento spôsob nie je jediný ale je odporúčaný nako ľko je najmenej náchylný na chyby.

1. Tvorba user story - user story sa automaticky nalinkuje na daný epic



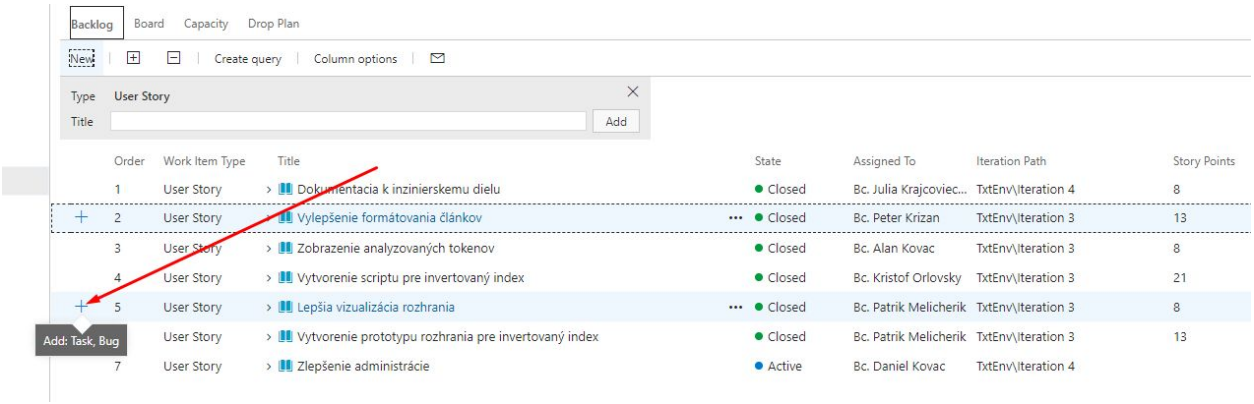
2. Správne vypísaná User story (s vysvetlivkami jednotlivých častí)

Legenda k obrázku: 1. Názov User story, 2. Zodpovedný člen tímu za US, 3. Stav US, 4. Iterácia v ktorej je US vykonávaná, 5. Description(formuluje product owner), 6. Akceptačné kritériá, 7. Story pointy (vyplývajúce z planning pokeru), 8. Priorita US 9 . Nalinkované procesy 10. Prílohy



1. Tvorba Taskov - úloh

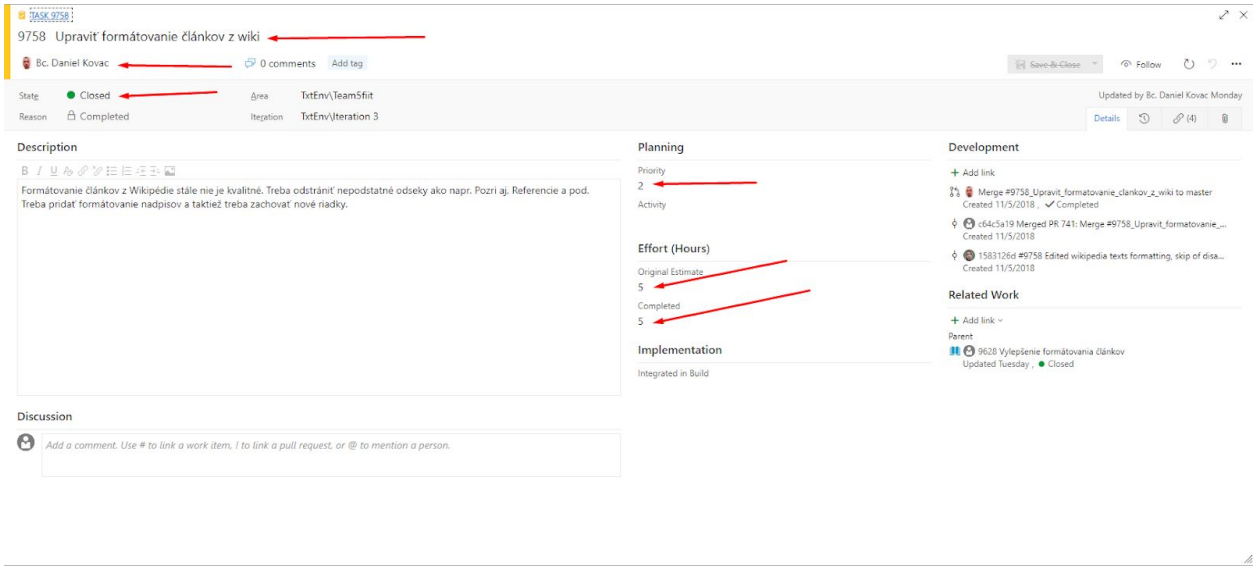
Tvorba úlohy spočíva v rozdelení príslušnej user story do častí, ktorých realizácia by nemala presahovať 1 “man day” (približný pracovný čas pracovníka ~ 8hodín) . Tieto časti sú následne prerozdelené medzi jednotlivých členov a ich manažment spadá priamo pod zodpovednú osobu za danú úlohu. Na obrázku nižšie je zobrazený proces vytvorenia úlohy s nalinkovaním na príslušnú user story.



2. Vyplnenie úlohy

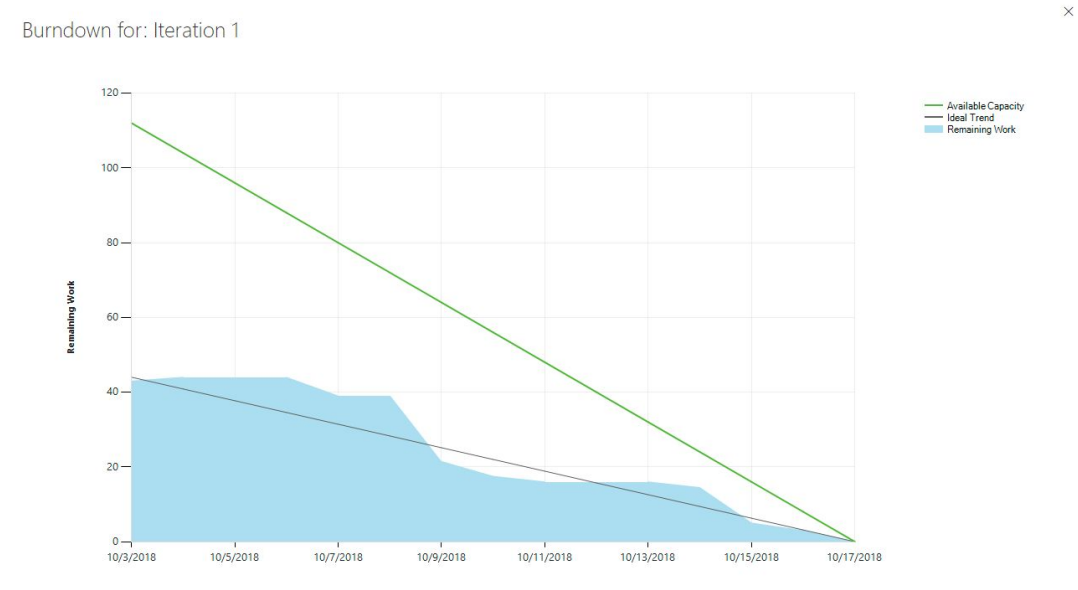
Úloha zahŕňa správne vyplnenie požadovaných polí. Tento proces je **nutný pre správne vyhodnocovanie štatistík za daný šprint,** nakoľko vkladané a editované hodnoty majú výsledný dopad na burndown chart.

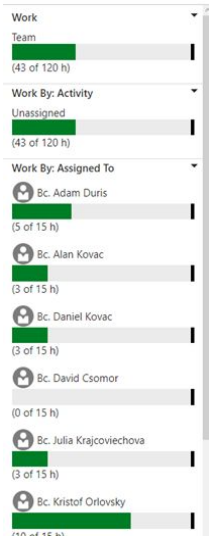
Legenda k obrázku 1. Názov úlohy, 2. Zodpovedný člen tímu, 3. Stav úlohy, 4. Priorita úlohy, 5. Odhadovaný čas vykonania úlohy, 6. Skutočný čas realizácie úlohy



Ciele

Výsledkom tejto metodiky je udržateľný stav manažmentu úloh a ľahší proces plánovania, nakoľko nástroj poskytuje užitočné informácie o efektivite tímu. Nižšie je zobrazený ideálny burndown chart za jednu iteráciu. Z tohto obrázku môžeme vyčítať, že daná iterácia bola naplánovaná dobre. Proces zlepšenia by spočíval v skoršom začatí riešenia jednotlivých úloh by graf mal stálo -klesajúcu tendenciu. Taktiež z uvedeného grafu vyplýva nárazová práca v dvoch etapách (pred stretnutím uprostred iterácie a pred koncom iterácie), celkovo však má graf klesajúcu tendenciu a množstvo prác na konci iterácie je nulové čo zodpovedá splneniu všetkých požadovaných úloh za danú iteráciu. Na druhom obrázku je vyobrazené vyťaženie jednotlivých členov tímu





6.9. Metodika konfigurácie softvérového systému

Autor: Alan Kováč

Procesy v tejto metodike sa vykonávajú vo viacerých štádiách. Prvým štádiom je inicializácia projektu. Na inicializáciu je potrebné vytýčiť, analyzovať a identifikovať komponenty. Integrácia týchto komponentov a návrhu použitia sa predloží vo forme dokumentu ostatným členom tímu. Manažér konfigurácie schváli predložené komponenty na základe odôvodnenia celého tímu. Manažér konfigurácie ohodnotí kvalitu komponentov vzhľadom na projekt. Po schválení komponentov a technológií pre realizovanie integrácie manažér konfigurácie postupuje presnými krokmi tejto metodiky. V prvom kroku nakonfiguruje vývojové prostredia pre členov tímu aby boli schopní pracovať na projekte. Pri konfigurácií sa hľadí na verzie technológií aby kompatibilita medzi vývojovým a produkčným prostredím nebola problémová. Všetky tieto procesy validuje a monitoruje rola manažéra konfigurácie, ktorá dohliada na nezrovnalosti medzi komponentami počas behu systému. Serverový administrátor rieši všetky problémy spojené s konfiguráciou skriptov pre buildovanie a lokálne spúšťanie aplikácie v závislosti od technológií na základe modelu aplikácie.

Inicializačná fáza kontinuálnej integrity :

1. Konfigurácia lokálnej časti, tak aby bola vykonaná na základe vzniknutých dokumentov a artefaktov.
2. Konfigurácia serverovej časti, tak aby bola vykonaná na základe vzniknutých dokumentov a artefaktov.
   Prvý a druhý bod zahŕňa integrovanie vytýčených technológií na zariadenia. Rôzne nastavenia pre produkčné a vývojové prostredie (rôzne nastavenia IP serverov, alebo iných konfiguračných vlastností v config). Vytvoria sa inicializačné skripty pre vytvorenie, naplnenie a otestovanie dát (naplnených do databázy).

3. Funkčný prototyp nakonfigurovaných časti spustíme a vyskúšame v rámci členov v tíme. Členovia poukážu na nedostatky a možné zlepšenia, ktoré boli opravené v rámci rekonfigurácie prvého kroku tejto metodiky.
4. Na základe výsledkov a vyhodnotenia prvých trocha procesov, môžeme pripraviť kontinuálnu integritu pre nasadzovanie softvéru do produkčného prostredia. Manažér dohliada a kontroluje členov tímu pri používaní tohto nástroja. Nástroj má členom uľahčiť nasadzovanie v týchto krokoch :

   Skontrolovaný artefakt (po Pull requeste podľa metodiky prehliadok kódu) je najaktuálnejší zdrojový kód vo vetve *Master.*

   Prebratie *Master* vetvy nasleduje po vykonaní prehliadky kódu 6.3. na základe čoho sa vyhodnotí spustiteľný artefakt manažérom prehliadok kódu.

   Člen tímu, ktorý ide vytvoriť požiadavku na nasadenia, musí pracovať len s vzdialenou *Master* vetvou.

   Ak sa nachádza na takejto *Master* vetve je nutné vytvoriť Pull na vzdialenú *Live* vetvu.

   Vytvorením Pull do vetvy *Live* odošle celý zdrojový kód z nášho repozitára na server.

   Zároveň po predošlom kroku sa automaticky aplikácie buildne a nasadí do produkčného prostredia.

6.10. Metodika pre výmenu dát

Autor: Adam Ďuriš

Táto metodika sa zaoberá procesom výmeny dát medzi jednotlivými členmi tímu. Pod dátami rozumieme ako zdrojový kód vyvíjaného softvérového produktu, tak aj všetku dokumentáciu, zápisnice a výstupy zo šprintov. Za dáta, ktoré sú potrebné zdieľať považujeme aj výstupy jednotlivých user stories a úloh (taskov), ako napr. rôzne analýzy, dokumenty, tabuľky a rôzne iné súbory rôzneho typu. Na základe nátury dát potrebných na zdieľanie boli identifikované tri hlavné platformy, ktoré sú využívané počas vývoja:

1. Git repozitár
2. Nástroj TFS
3. Google disk (Google drive)

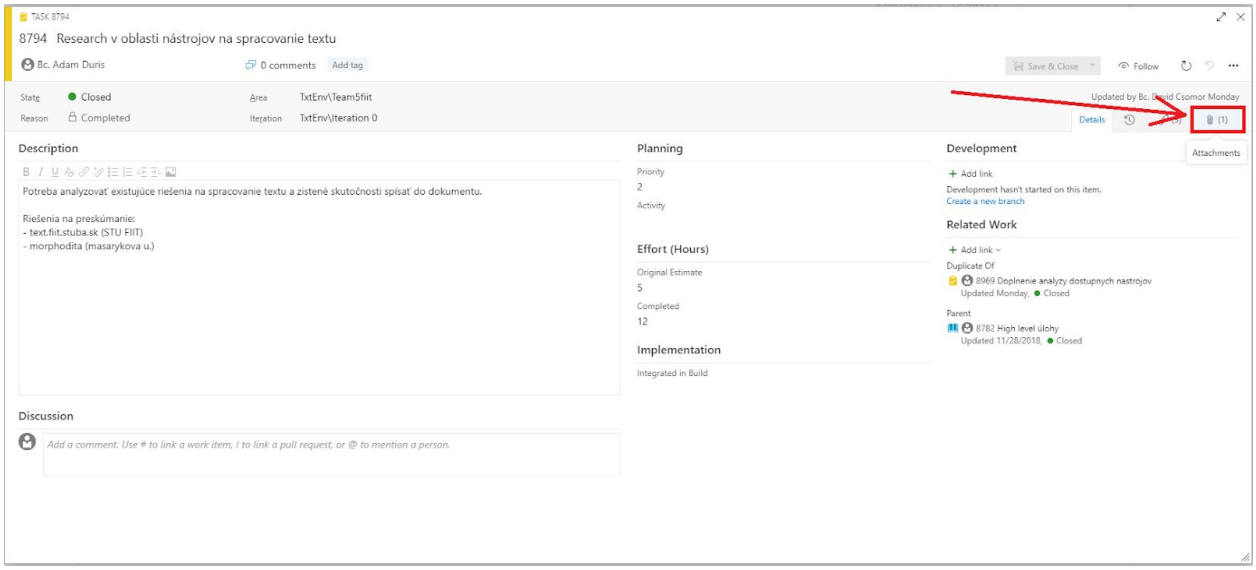
6.10.1. Git repozitár

Git repozitár je v dnešnej dobe štandardom pre zdieľanie zdrojových kódov. Nami využívaný repozitár je poskytovaný nástrojom TFS, ktorý slúži aj na sledovanie vývoja. V repozitári existuje jedna hlavná master vetva a každá ďalšia vetva zodpovedá jednej vytvorenej user story. Viac o manažmente vetiev v git repozitári sa možno dočítať v časti Metodika verziovania.

6.10.2. Nástroj TFS

Keďže pod dátami rozumieme aj výstupy user stories, ktoré sa nezaoberajú priamo implementáciou, ako napríklad analytické a výskumné úlohy a ich výstupom sú rôzne dokumenty, poznámky a pod., tak vzniknuté výstupy sa zdieľajú pomocou nástroja TFS. Zdieľajú sa tak, že vzniknutý dokument, alebo akýkoľvek iný súbor, ktorý je vhodné (a častokrát nutné) zdieľať, sa pridá ako príloha k príslušnej user story, resp. k príslušnej úlohe (tasku).
Postup pre pripojenie výstupného súboru k user story:

- Zvoliť príslušnú user story alebo príslušný task kliknutím na názov v šprint backlogu
- Kliknúť v pravom hornom rohu na záložku s prílohami (ikona spinky)



- Pridať prílohu (dva spôsoby)
    - Jednoduchým spôsobom “drag and drop” presunieme súbor do otvoreného okna
    - Kliknutím na “Add attachment” a následné zvolenie súboru
- Pridaný súbor sa následne zobrazí medzi prílohami

6.10.3. Google Disk

Ako centrálne úložisko dát bol zvolený Google disk, ktorý bol založený na tímovom Google účte a je zdieľaný s každým členom tímu a product ownerom (vedúcim tímu). V tomto úložisku sa uchovávajú všetky dokumentácie, zápisnice, tabuľky, analýzy, ale aj dôležité odkazy na stránky a rôzne iné. Okrem toho je potrebné, aby aj sem členovia tímu pridávali výstupné súbory z úloh, nakoľko priečinková organizácia poskytuje omnoho lepšiu prehľadnosť. Každý súbor je potrebné pridať do správneho priečinka tak, aby boli súbory logicky organizované. Bližšie o organizácií priečinkov sa možno dočítať v 6.7 Metodika dokumentácie.

Pokiaľ je nutné vyhotoviť dokument, tabuľku, alebo prezentáciu, na ktorých sa podieľajú viacerí členovia, tak využívame prostriedky, ktoré poskytuje spoločnosť Google, akými sú Google Docs, Google Sheets, alebo Google Slides, ktoré umožňujú paralelnú editáciu a zdieľanie medzi všetkými členmi tímu, pričom každý takto vytvorený súbor musí byť takisto vytvorený v príslušnom priečinku.

7. Export úloh z TFS

Prehľad úloh 1. šprintu

ID	Title	Work Item Type	State	Assigned To	Story Points
8982	Práce spojené s TPcupom	User Story	Closed	Bc. Julia Krajcoviechova	3
8984	Spísanie dokumentu (prihláška)	Task	Closed	Bc. Julia Krajcoviechova	
9271	Zlepšenie administrácie	User Story	Active	Bc. Daniel Kovac	
9272	Pridanie logovania	Task	Closed	Bc. Adam Duris	
8996	Finalizácia a dokumentácia návrhu architektúry systému	User Story	Closed	Bc. Alan Kovac	3
8999	Modelácia a spísanie doku k návrhu	Task	Closed	Bc. Alan Kovac	
8964	Finálne nastavenie TFS	User Story	Closed	Bc. Peter Krizan	5
8965	Prepojenie git-u s TFS	Task	Closed	Bc. Peter Krizan	
8966	Inicializácia taskov + plánovanie Features	Task	Closed	Bc. Peter Krizan	
8998	Filtrovanie nad mongoDB setom	User Story	Closed	Bc. Daniel Kovac	5
9000	Štúdium MongoDB	Task	Closed	Bc. Daniel Kovac	
9047	Príprava API na prístup do DB	Task	Closed	Bc. Daniel Kovac	
8993	Kompletizácia analýzy nástrojov	User Story	Closed	Bc. Adam Duris	8
8969	Doplnenie analyzy dostupnych nastrojov	Task	Closed	Bc. David Csomor	
8977	Zjednotiť proces importu dát	User Story	Closed	Bc. Kristof Orlovsky	21
8978	Zjednotiť Pepe_script a Kiko_script	Task	Closed	Bc. Kristof Orlovsky	
8979	Testovanie a nasadenie výsledného scriptu pre použitie na serveri	Task	Closed	Bc. Peter Krizan	
9044	Úprava schémy a výstupu importérov podľa nej	Task	Closed	Bc. Alan Kovac	
9269	Implementácia uloženia nového korpusu	Task	Closed	Bc. Kristof Orlovsky	
9270	Úprava vkladania článku	Task	Closed	Bc. Daniel Kovac	
9273	Pridanie rozhrania na pridanie korpusu	Task	Closed	Bc. Daniel Kovac	
9274	Pridanie selectu na výber korpusu pri importe článku	Task	Closed	Bc. Patrik Melicherik	
9198	Anotácia tokenov	User Story	Closed	Bc. Daniel Kovac	5
9222	Implementácia API na spracovanie tokenov k textom	Task	Closed	Bc. Alan Kovac	
9223	Implementácia volania API na pridanie tokenov na frontende	Task	Closed	Bc. Patrik Melicherik	
8994	Návrh + prototyp rozhrania	User Story	Closed	Bc. Patrik Melicherik	13
8995	Vytvorenie wireframe-ov	Task	Closed	Bc. Patrik Melicherik	
9202	Vytvorenie klikatelneho prototypu	Task	Closed	Bc. Patrik Melicherik	

Prehľad úloh 2. šprintu

ID	Title	Work Item Type	State	Assigned To	Story Points
9271	Zlepšenie administrácie	User Story	Active	Bc. Daniel Kovac	
9385	Spojiť logger so slackom	Task	Closed	Bc. Adam Duris	
9402	Vytiahnutie URL v angulari do config suboru	Task	Closed	Bc. Patrik Melicherik	
9411	Crawler na články z webnoviny.sk	User Story	Closed	Bc. Julia Krajcoviechova	5
9603	Vytvorenie crawleru	Task	Closed	Bc. Julia Krajcoviechova	
9604	Sťahovanie dát použitím Crawleru	Task	Closed	Bc. Julia Krajcoviechova	
9611	Pridanie článkov do DB	Task	Closed	Bc. Daniel Kovac	
9201	Analýza features z textu	User Story	Active	Bc. Peter Krizan	8
9512	Analýza rôznych sád čŕt pre texty	Task	Active	Bc. Peter Krizan	
9410	Anotácia pridaných článkov	User Story	Closed	Bc. Alan Kovac	13
9501	Pridanie označovania paragrafu v texte	Task	Closed	Bc. Kristof Orlovsky	
9507	Zavolanie tokenizatora pre davku clankov podla korpusu	Task	Closed	Bc. Daniel Kovac	
9508	Rozdelenie textu v volani externeho API tokenizatora	Task	Closed	Bc. Alan Kovac	
9510	Rozšírenie schemy o datumove znacky	Task	Closed	Bc. Alan Kovac	
9513	Pridanie buttonu na tokenovanie celeho korpusu	Task	Closed	Bc. Daniel Kovac	
9405	Pridat viacero kolekcii	User Story	Closed	Bc. Kristof Orlovsky	5
9502	Úprava formátu zoznamu článkov	Task	Closed	Bc. Kristof Orlovsky	
9504	Vytvorenie korpusu mestá a následné naplnenie článkami o mestách	Task	Closed	Bc. Kristof Orlovsky	
9505	Vytvorenie korpusu Vrchy a jeho zaplnenie	Task	Closed	Bc. Kristof Orlovsky	
9509	Vytvoriť dataset z ľubovoľných známych osobností na Wikipedii	Task	Closed	Bc. Kristof Orlovsky	
9511	Pridat korpus a články o osobnostiach	Task	Closed	Bc. Kristof Orlovsky	
9414	Vytvorenie scriptu pre invertovaný index	User Story	Closed	Bc. Kristof Orlovsky	21
9594	Vytvorenie struktury pre invertovany index	Task	Closed	Bc. Kristof Orlovsky	
9415	Vytvorenie prototypu rozhrania pre invertovaný index	User Story	Closed	Bc. Patrik Melicherik	13
9590	Refactoring	Task	Closed	Bc. Patrik Melicherik	
9605	Vytvorenie notifikacii	Task	Closed	Bc. Patrik Melicherik	

Prehľad úloh 3. šprintu

ID	Title	Work Item Type	State	Assigned To	Story Points
9822	Dokumentacia k inzinierskemu dielu	User Story	Active	Bc. Julia Krajcoviechova	
9824	Analyza vsetkych casti, ktore maju byt dokumentovane	Task	Closed	Bc. Julia Krajcoviechova	
9628	Vylepšenie formátovania článkov	User Story	Closed	Bc. Peter Krizan	13
9758	Upraviť formátovanie článkov z wiki	Task	Closed	Bc. Daniel Kovac	
9759	Upraviť Crawler webnovín, aby vyhovoval formátovaniu	Task	Closed	Bc. Julia Krajcoviechova	
9761	Odstrániť rozlišovacie stránky	Task	Closed	Bc. Daniel Kovac	
9762	Integrácia pythonu s nodejs	Task	Closed	Bc. Peter Krizan	
9803	Drop databázy a nový seed	Task	Closed	Bc. Daniel Kovac	
9816	Formátovanie článkov z webnovín	Task	Closed	Bc. Peter Krizan	
9271	Zlepšenie administrácie	User Story	Active	Bc. Daniel Kovac	
9809	Logovanie v pythone	Task	Closed	Bc. David Csomor	
9810	Kontrola coding style v Pythone	Task	Closed	Bc. Daniel Kovac	
9818	Nasadenie aktuálnej verzie	Task	Closed	Bc. Daniel Kovac	
9819	Spúštanie Python časti	Task	Closed	Bc. Daniel Kovac	
9821	Vedenie evidencie Python závislosti	Task	Closed	Bc. Julia Krajcoviechova	
9631	Zobrazenie analyzovaných tokenov	User Story	Active	Bc. Alan Kovac	8
9711	Vytvorenie API na ziskanie tokenov pre konkretny clanok	Task	Closed	Bc. Alan Kovac	
9764	Vytvoriť rozhranie pre tokeny	Task	Closed	Bc. Patrik Melicherik	
9814	Pridat tab histogramu	Task	Closed	Bc. Patrik Melicherik	
9414	Vytvorenie scriptu pre invertovaný index	User Story	Closed	Bc. Kristof Orlovsky	21
9555	Implementacia pre konkretny clanok	Task	Closed	Bc. Kristof Orlovsky	
9788	Implementácia pre otokenizovaný korpus	Task	Closed	Bc. Kristof Orlovsky	
9820	Analyzovat blizsie TFIDF	Task	Closed	Bc. Kristof Orlovsky	
9629	Lepšia vizualizácia rozhrania	User Story	Closed	Bc. Patrik Melicherik	8
9713	Pridat angular material	Task	Closed	Bc. Patrik Melicherik	
9811	Úprava webu	Task	Closed	Bc. Daniel Kovac	
9813	Pridat datumy pre jednotlive clanky	Task	Closed	Bc. Adam Duris	
9815	Pridat rozhranie pre import z Webnovin	Task	Closed	Bc. Peter Krizan	
9967	Ziskat dlzku clanku	Task	Closed	Bc. Daniel Kovac	
9415	Vytvorenie prototypu rozhrania pre invertovaný index	User Story	Closed	Bc. Patrik Melicherik	13
9556	Vytvorenie rozhrania pre invertovany index	Task	Closed	Bc. Patrik Melicherik	
9557	Napojenie na backend	Task	Closed	Bc. Patrik Melicherik	
9558	Filtrovanie indexov	Task	Closed	Bc. Patrik Melicherik	
9807	Uprava rozhrania pre invertovany index	Task	Closed	Bc. Adam Duris	

Prehľad úloh 4. šprintu

ID	Title	Work Item Type	State	Assigned To	Story Points
9822	Dokumentacia k inzinierskemu dielu	User Story	Closed	Bc. Julia Krajcoviechova	8
9823	Vytvorenie instalacnej prirucky	Task	Closed	Bc. Julia Krajcoviechova	
9979	Dokumentácia k API službám	Task	Closed	Bc. Patrik Melicherik	
9980	Dokumentácia k odovzdaniu k Ing. Diela	Task	Closed	Bc. Julia Krajcoviechova	
9981	Dokumentácia k výstupu šprintov	Task	Closed	Bc. David Csomor	
9631	Zobrazenie analyzovaných tokenov	User Story	Closed	Bc. Alan Kovac	8
9817	API pre histogram článku	Task	Closed	Bc. Alan Kovac	
10120	Uprava frontendu z dôvodu úpravy API pre histogramy	Task	Closed	Bc. Patrik Melicherik	
9414	Vytvorenie scriptu pre invertovaný index	User Story	Closed	Bc. Kristof Orlovsky	21
10222	V oindexovanom článku chýba dátum indexedAt	Bug	Closed	Bc. Peter Krizan	
9972	Sprehľadnenie rozhrania	User Story	Closed	Bc. Patrik Melicherik	8
9982	API pre invertovaný index jedného článku	Task	Closed	Bc. Kristof Orlovsky	
9983	Prekliky z článku do invertovaného indexu a späť	Task	Closed	Bc. Adam Duris	
9984	Filtrovanie kliknutím na slovo alebo lému	Task	Closed	Bc. Patrik Melicherik	
9985	Úprava formátovania tabuľky	Task	Closed	Bc. Adam Duris	
9986	POS zobrazovať iba nultý index	Task	Closed	Bc. Adam Duris	
10066	Pridanie filtrov pre POS	Task	Closed	Bc. Patrik Melicherik	
9271	Zlepšenie administrácie	User Story	Active	Bc. Daniel Kovac	
9995	Nasadenie Python časti na server	Task	Closed	Bc. Daniel Kovac	
9997	Vytvorenie deploy skriptu pre celý SW	Task	Closed	Bc. Daniel Kovac	
10129	Integrácia Angularu do Expressu	Task	Closed	Bc. Daniel Kovac	
9974	Oddeliť verejné funkcionality od správy webstránky.	User Story	Closed	Bc. Daniel Kovac	21
9987	Vytvorenie loginu na frontende	Task	Closed	Bc. Patrik Melicherik	
9988	Vytvorenie tokenu na serveri	Task	Closed	Bc. Daniel Kovac	
9989	Zamknutie API na token	Task	Closed	Bc. Daniel Kovac	
9992	Vytvorenie kolekcia používateľov	Task	Closed	Bc. Daniel Kovac	
10121	Zamknutie neverejnej funkcionality ak user nie je prihlásený	Task	Closed	Bc. Patrik Melicherik	
9975	Vytvoriť prepínanie prostredí	User Story	Closed	Bc. Alan Kovac	3
9990	Vytvorenie enviromentu pre js (production/develop)	Task	Closed	Bc. Alan Kovac	
9991	Vytvorenie novej DB s celým seedom	Task	Closed	Bc. Alan Kovac	
9977	Tvorba dokumentu k mentoringu TPcup	User Story	Closed	Bc. Julia Krajcoviechova	2
9993	Vytvorenie dokumentu + kontrola otázok	Task	Closed	Bc. Julia Krajcoviechova	
9978	Zobraziť štatistiky o aktuálnych dátach	User Story	Closed	Bc. Peter Krizan	8
9994	api na štatistiky o článkoch	Task	Closed	Bc. Peter Krizan	
9996	Vytvorenie landing page pre štatistiky	Task	Closed	Bc. Kristof Orlovsky	

Prehľad úloh 5. Šprintu

ID	Title	Work Item Type	State	Assigned To	Story Points
10299	Vyhotovenie finálnej dokumentácie	User Story	Active	Bc. Julia Krajcoviechova	13
10460	Kompletizácia dokumentácie	Task	Closed	Bc. Julia Krajcoviechova	
10461	Odovzdanie dokumentácie	Task	Active	Bc. David Csomor	
10294	Naplnenie databázy článkami z webnovín do finálnych korpusov	User Story	Closed	Bc. Peter Krizan	21
10300	Napojenie django na db	Task	Closed	Bc. Daniel Kovac	
10301	Úprava crawleru z webnovín	Task	Closed	Bc. Peter Krizan	
10302	Metódy na získanie a vloženie do cache	Task	Closed	Bc. Daniel Kovac	
10303	Naplnenie db článkami z webnovín	Task	Closed	Bc. Peter Krizan	
10312	Pridať obmedzenie /page	Task	Closed	Bc. Peter Krizan	
10295	Naplniť databázu článkami z wikipédie	User Story	Closed	Bc. Kristof Orlovsky	8
10304	Research nových osobností z wiki	Task	Closed	Bc. Kristof Orlovsky	
10305	Naplniť db wiki článkami	Task	Closed	Bc. Kristof Orlovsky	
10489	Volanie invertovaného indexu sa zahlti pri veľkých volaniach	Bug	Closed	Bc. Kristof Orlovsky	
10296	Anotovať novo pridané články v korpusoch	User Story	Closed	Bc. Kristof Orlovsky	5
10306	Spustenie procesu anotácie a indexácie	Task	Closed	Bc. Daniel Kovac	
10358	Tokenizacia/Indexovanie podla korpusu z admin sekcie	Bug	Closed	Bc. Alan Kovac	
10297	Zlepšenie filtrovania v sekcii invertovaný index	User Story	Closed	Bc. Adam Duris	8
10307	Pridať nové filtre	Task	Closed	Bc. Adam Duris	
10308	Upraviť existujúce filtre	Task	Closed	Bc. Adam Duris	
10298	Vytvorenie zoskupenia v sekcii invertovaný index	User Story	Closed	Bc. Alan Kovac	13
10309	Vytvorenie API pre group by podľa korpusu	Task	Active	Bc. Alan Kovac	
10310	Vytvorenie API pre celok	Task	Active	Bc. Alan Kovac	
10311	Upraviť tabuľku invertovaného indexu	Task	Active	Bc. Patrik Melicherik	
10357	Filtorvanie v inverted index po prechod cez article	Bug	Active	Bc. Patrik Melicherik	
10313	Prototyp odtlačku článku	User Story	New	Bc. David Csomor	8
10398	Spočítanie POS podľa slovného druhu	Task	Active	Bc. David Csomor	

Prehľad úloh 6. Šprintu

ID	Title	Work Item Type	Assigned To	State	Story Points
10662	Abstrakt pre IT SRC	User Story	Bc. David Csomor	Resolved	8
10663	Abstrakt	Task	Bc. David Csomor	Closed	
10664	Článok	Task	Bc. David Csomor	Closed	
10313	Prototyp odtlačku článku	User Story	Bc. David Csomor	Closed	13
10646	Optimalizácia výpisov	User Story	Bc. Patrik Melicherik	Closed	13
10666	Vytvorenie nového selectu s celkovým počtom záznamov	Task	Bc. Alan Kovac	Closed	
10648	Optimalizácia anotácií	User Story	Bc. Kristof Orlovsky	Closed	8
10669	Návrh novej schémy	Task	Bc. Kristof Orlovsky	Closed	
10670	Implementácia/nasadenie novej schémy	Task	Bc. Kristof Orlovsky	Closed	
10650	Doplniť články do DB	User Story	Bc. Peter Krizan	Closed	13
10651	Doplniť články z wiki	Task	Bc. Kristof Orlovsky	Closed	
10652	Doplniť články z webnovín	Task	Bc. Peter Krizan	Closed	
10653	Úprava frontendovej časti +DB	User Story	Bc. Patrik Melicherik	Closed	29
10654	Zjednotenie jazyku	Task	Bc. Patrik Melicherik	Closed	
10657	Uprava názvov korpusov z webnovín	Task	Bc. Daniel Kovac	Closed	
10658	Úprava farieb grafov + Jquery countTo + cache štatistík	Task	Bc. Daniel Kovac	Closed	

Prehľad úloh 7. Šprintu

ID	Title	Work Item Type	Assigned To	State	Story Points
10891	Príprava live analyzátora pre používateľa a inej zaujímavosti	User Story	Bc. Daniel Kovac	Closed	
10892	Porozmýšlať nad aktivitou pre verejnosť	Task	Bc. Peter Krizan	Closed	
10907	Pripraviť rozhranie na live analýzu textu	Task	Bc. Daniel Kovac	Closed	
11001	Pripraviť rozhranie pre Otdlačok analyzovaného článku	Task	Bc. Peter Krizan	Closed	
10313	Prototyp odtlačku článku	User Story	Bc. David Csomor	Closed	13
10398	Spočítanie POS podľa slovného druhu	Task	Bc. David Csomor	Closed	
10915	Vytvoriť rozhranie pre odtlačok článku	Task	Bc. Adam Duris	Closed	
11051	Spočítanie NER podľa kategórie	Task	Bc. David Csomor	Closed	
10646	Optimalizácia výpisov	User Story	Bc. Patrik Melicherik	Closed	13
10665	Úprava selectov do DB	Task	Bc. Alan Kovac	Closed	
10667	Úprava API pre stránkovanie	Task	Bc. Alan Kovac	Closed	
10648	Optimalizácia anotácií	User Story	Bc. Kristof Orlovsky	Closed	8
10671	Úprava dotknutých častí projektu	Task	Bc. Kristof Orlovsky	Closed	

Prehľad úloh 8. Šprintu

ID	Title	Work Item Type	Assigned To	State	Story Points
11083	Prototyp pre interaktávnu hru za účelom získania informácií od používateľa	User Story	Bc. Peter Krizan	Closed	13
11091	Vymyslenie samotného konceptu hry	Task	Bc. Peter Krizan	Closed	
11092	Implementácia serverovej časti hry	Task	Bc. Daniel Kovac	Closed	
11093	Klientská časť hry	Task	Bc. Adam Duris	Closed	
11172	FE template pre interaktívnu hru	Task	Bc. Peter Krizan	Closed	
11305	Herna logika	Task	Bc. Peter Krizan	Closed	
11082	Prototyp rozhrania pre porovnanie článku s korpusmi	User Story	Bc. Patrik Melicherik	Closed	13
11114	Pripravenie prototypu	Task	Bc. Patrik Melicherik	Closed	
11246	Príprava prototypu porovnania článku z live analýzy	Task	Bc. Daniel Kovac	Closed	
11080	Doplnenie poradia metriky TFIDF do invertovaného indexu	User Story	Bc. Kristof Orlovsky	Closed	13
11115	Pridanie metriky TFIDF	Task	Bc. Alan Kovac	Closed	
11306	Výpočet TFIDF nad viacerými úrovňami abstrakcie	Task	Bc. Kristof Orlovsky	Closed	
9271	Zlepšenie administrácie	User Story	Bc. Daniel Kovac	Active	
11247	Kompletná záloha produkčnej databázy	Task	Bc. Kristof Orlovsky	Active	
10646	Optimalizácia výpisov	User Story	Bc. Patrik Melicherik	Closed	13
10668	Úprava stránkovania na frontende	Task	Bc. Patrik Melicherik	Closed	
11201	Optimalizácia dopytu pre získanie článkov	Task	Bc. Daniel Kovac	Closed	
10653	Úprava frontendovej časti +DB	User Story	Bc. Patrik Melicherik	Closed	29
10655	Oprava prelinkovania z článku na invertovaný index	Task	Bc. Adam Duris	Closed	
11073	Premenovať korpus "štáty a mestá" na "geografia"	Task	Bc. Kristof Orlovsky	Closed	

Prehľad úloh 9. Šprintu

ID	Title	Work Item Type	Assigned To	State	Story Points
11462	Testovanie softvéru	User Story	Bc. Daniel Kovac	Active	
11463	Testovanie API	Task	Bc. Daniel Kovac	New	
11464	Testovanie 1. časti frontendu	Task	Bc. Patrik Melicherik	New	
11465	Testovanie 2. časti frontendu	Task	Bc. Adam Duris	New	
11466	Testovanie Python časti	Task	Bc. David Csomor	Closed	
11315	Priradenie percentualneho podielu uploadnutého článku k existujúcim korpusom	User Story	Bc. Patrik Melicherik	Resolved	13
11512	Backend vypocet percentualnej hodnoty slov z clanku voci korpusu	Task	Bc. Alan Kovac	Closed	
11313	FInalizácia komparátora	User Story	Bc. Patrik Melicherik	Closed	13
11340	Pridanie výpočtu TF a IDF pre jeden článok zo živej analýzy	Task	Bc. Kristof Orlovsky	Closed	
11506	Dokončenie komparátora pre živú analýzu	Task	Bc. Daniel Kovac	Closed	
11308	Plagát na ITSRC	User Story	Bc. David Csomor	Closed	8
11447	Vytvorenie plagátu	Task	Bc. David Csomor	Closed	
11448	Vytvorenie biznis karty/vizitky	Task	Bc. David Csomor	Closed	
11309	Dokončenie dokumentov na ITSRC	User Story	Bc. David Csomor	Closed	1
11422	Pár viet k videu	Task	Bc. Kristof Orlovsky	Closed	
11445	Vytvorenie videa	Task	Bc. Daniel Kovac	Closed	
11446	Napísanie a odovzdanie dokumentov na IIT-SRC	Task	Bc. David Csomor	Closed	
11080	Doplnenie poradia metriky TFIDF do invertovaného indexu	User Story	Bc. Kristof Orlovsky	Closed	13
11557	Vytvorenie tabuľky na frontende a jej zaplnenie	Task	Bc. Kristof Orlovsky	Closed	
11314	Pridanie hodnoty poradia do invertovaného indexu	Task	Bc. Alan Kovac	Closed	
11507	Pridanie tabuľky najčastejšie vyskytujúcich sa slov korpusov	User Story	Bc. Kristof Orlovsky	Resolved	
11556	Prepojenie volaní na backendovej strane spolu s tvorbou cacheovania	Task	Bc. Kristof Orlovsky	Closed	
10653	Úprava frontendovej časti +DB	User Story	Bc. Patrik Melicherik	Closed	29
11310	Úprava filtrov, strankovania a pridanie poradia	Task	Bc. David Csomor	Closed	
11343	Odstránenie prebytočných prelinkovaní	Task	Bc. Daniel Kovac	Closed	
11385	Odstránenie desatinných čiarok na domovskej stránke	Task	Bc. Daniel Kovac	Closed	
11481	Úprava obrazovky pre živú analýzu	Task	Bc. Adam Duris	Closed	
11311	Dokončenie funkčnej a hrateľnej verzie hry	User Story	Bc. Peter Krizan	Closed	13
11560	Vyriešenie responzivity pre nižšie rozlíšenia	Task	Bc. Peter Krizan	Closed	
11561	Úprava API pre uloženie výsledku hry	Task	Bc. Daniel Kovac	Closed	
11562	Implementácia uloženia hry	Task	Bc. Peter Krizan	Closed	

Prehľad úloh 10. Šprintu

ID	Title	Work Item Type	Assigned To	State	Story Points
11569	Hladanie chýb	User Story	Bc. Daniel Kovac	Active	8
11588	Hľadanie chýb	Task	Bc. Kristof Orlovsky	New	
11589	Hľadanie chýb	Task	Bc. Patrik Melicherik	Closed	
11590	Hľadanie chýb	Task	Bc. Daniel Kovac	Closed	
11591	Hľadanie chýb	Task	Bc. Peter Krizan	New	
11592	Hľadanie chýb	Task	Bc. David Csomor	Closed	
11593	Hľadanie chýb	Task	Bc. Adam Duris	Closed	
11594	Hľadanie chýb	Task	Bc. Alan Kovac	Active	
11708	Fix pre fungovanie webu v prehliadačoch Firefox a Edge	Task	Bc. Daniel Kovac	Closed	
11568	Vytvoriť rozhranie pre zobrazenie štatistík z hry	User Story	Bc. Peter Krizan	Closed	13
11595	Uloženie vety do DB o štatistikách z hry	Task	Bc. Peter Krizan	Closed	
11596	Vytvorenie rebríčku na frontende	Task	Bc. Peter Krizan	New	
11597	Vytvorenie API pre rebríček	Task	Bc. Peter Krizan	Active	
11567	Vyladenie tabulky relevantných slov	User Story	Bc. Kristof Orlovsky	Closed	8
11598	Potrebné úpravy	Task	Bc. Kristof Orlovsky	Closed	
11308	Plagát na ITSRC	User Story	Bc. David Csomor	Closed	8
11563	Vytlačenie prezentačných meteriálov	Task	Bc. David Csomor	Closed	
11565	Vylepšenie živej analýzy pre prezentačné účely	User Story	Bc. Adam Duris	Closed	8
11600	Fixnutie API pre porovnanie v živej analýze	Task	Bc. Alan Kovac	Closed	
11731	Zahrnutie percentuálneho porovnania	Task	Bc. Daniel Kovac	Closed	
11733	Reorganizácia tabov	Task	Bc. Adam Duris	Closed	
10653	Úprava frontendovej časti +DB	User Story	Bc. Patrik Melicherik	Closed	29
11566	Rozsirenie zaznamov invertovaneho indexu	Task	Bc. Patrik Melicherik	Closed	
11766	Oprava chýb	User Story	Bc. Adam Duris	Active	
11767	Opraviť zoradenie prvkov v histograme článku	Task	Bc. Adam Duris	Closed	

Prehľad úloh 11. Šprintu

ID	Title	Work Item Type	Assigned To	State	Story Points
11803	Opravy tabuľky najrelevantnejších slov korpusov	User Story	Bc. Kristof Orlovsky	Resolved	
11804	Zosortovať názvy korpusov v tabuľke	Task	Bc. Kristof Orlovsky	Closed	
11805	Nastavenie textu vo filtrovaní invertovaného indexu	Task	Bc. Kristof Orlovsky	Closed	
11773	Vytvorenie kolekcie viet	User Story	Bc. Alan Kovac	Closed	13
11895	Vytvorenie schemy	Task	Bc. Alan Kovac	Closed	
11896	Select a zgroupenie viet do poli	Task	Bc. Alan Kovac	Closed	
11897	Vlozenie davky viet do DB	Task	Bc. Alan Kovac	Closed	
11774	Vytvorenie kolekcie pre n-gramy	User Story	Bc. Alan Kovac	Closed	8
11898	vytvorenie schemy pre ngramy	Task	Bc. Alan Kovac	Closed	
11899	vlozenie davky ngramov do DB	Task	Bc. Alan Kovac	Closed	
11900	Select a redukovanie vysledkov do Ngramov	Task	Bc. Alan Kovac	Closed	
11776	Používateľská príručka	User Story	Bc. Adam Duris	Resolved	5
11889	Napísanie dokumentu s používateľskou príručkou	Task	Bc. Adam Duris	Closed	
11777	Inštalačná príručka	User Story	Bc. David Csomor	New	2
11778	Dokumentácia k inžinierskemu dielu	User Story	Bc. David Csomor	New	13
11915	Popísať nahratie článku z wikipédie	Task	Bc. Kristof Orlovsky	Closed	
11916	Popísať tvorbu invertovaného indexu	Task	Bc. Kristof Orlovsky	Closed	
11917	popísať API a dopyty spojene s pridaním Wiki článkou, tvorbou invertovaných indexu a počítanim tfidf	Task	Bc. Kristof Orlovsky	Closed	
11918	Proces nahratia článkov z web novín	Task	Bc. Peter Krizan	Closed	

8. Zápisnice zo stretnutí

Zápisnica 1

Dátum	25.09.2018 – 12:00
Miesto	FIIT STU, 4.26
Zapisovateľ	Dávid Csomor

Analýza nástrojov

- Košická katedra nie je veľmi dobra (nástroje majú nedostatky - pozrieť)
- Podrobne analyzovať nástroje na spracovanie textu na fakulte FIIT - http://text.fiit.stuba.sk/

Nefunkcionálne požiadavky na produkt

- Navrhnúť riešenie tak, aby nebol zaťažovaný server product ownera nárazovo (cache-ovanie)

Funkcionálne požiadavky na produkt

- Databáza plná textov (ktoré sa budú značkovať a kategorizovať)
  - príklad: export z wiki (cez náš crawler)
- Keď sa pridá novy text, tak by sme ho program mal byť schopný automaticky za radiť do kategórie a označkovať
- Vlastná no-SQL "custom made" databáza na uchovanie textu
- Premyslieť si účasť v TP CUP
- Budeme robiť len prototyp, podstatné je, aby fungovalo jadro (žiadne login-y, ošetrovanie pre používateľov a podobne)
- Definition of done – treba si presne definovať na začiatku, že kedy bude úloha hotová, resp. určiť si čo presne musí robiť a potom pri code review to musí splniť počiatočné ciele
- Vybrať nástroj na manažment úloh: youtrack, scrumdesk, TFS
- Dohodnúť sa na spoločnom nástroji na „trackovanie“

TP cup

- Ak sa neprihlásime do TP CUPu, tak na konci semestra bude prezentácia pred iným tímom (treba mať opísané aj to, že ako sme sa sledovali jednotlivé oblasti a vôbec ako bol vyriešený management)

Prezentácia tímu

- Treba mať riešene nasadené na serveri, ktoré bude po semestri premiestnená a zväčšnená na serveroch FIIT

Zápisnica 2

Dátum	26.09.2018 – 12:00
Miesto	FIIT STU, 4.26
Zapisovateľ	Dávid Csomor

Analýza nástrojov na sledovanie vývoja:

- Na logovanie počtu hodín: toggl, clock
- Na vyjadrenie percentuálneho hodnotenie zo 100% pre každého: excel
- Na hodnotenie user stories používať „scrum poker cards" (odhaľovať naraz)
- TP CUP (hlasovanie: 8 za, 0 proti)

MongoDB overview:

- Možné alternatívy: mongoVUE, RockMongo (na Rock treba PHP)
- Treba vytvoriť štruktúru mongo databázy
- Corpus bude zhluk/kolekcia článkov (mal by byť ako cudzí kľúč, mongo štandardne nemá join)
- Články z jedného zdroja (získať články zo stránok wiki a tie následne vyfiltrovať)
- Pridávať tagy (budeme ich tam písať manuálne vo forme zoznamu slov oddelených čiarkou)
- Dokument bude jedna inštancia (článok) už spracovaný text (rozdelený na slová, lemy)ň
- Slovo (uchovávanie duplicít je v poriadku aj v prípade, že majú iné kľúče)
- POS kategórie (part of speech) – gramatické kategórie/ slovné druhy
- Slovo ako NER → NER (named entity recognition) – názvoslovná entita

Zdroje:

- wikipédia, neskôr aj žurnály , SITA/TASR (zdroj správ zo zahraničia, možno poskytujú RSS, ktoré by sme mohli parsovať)

Úlohy vyplývajúce zo stretnutia:

- 1. možnosť: wikipédia umožňuje vrátiť stránku v JSONE ak pridáme nejaký špeciálny tag k URL
- 2. možnosť: vlastný parser a parsovať rovno z URL
- Na úvod napr. stiahnuť a spracovať štáty z wiki
- Spraviť rozhranie na vizualizáciu dát z databázy
- Bude obsahovať:
    - zoznam článkov

    - priemerná dĺžka článkov
    - počet znakov, slov, plnovýznamových slov, slovies, ...
    - robiť štatistiky nad článkami s vybraným tagom
    - histogram (slov, lem, slovných druhov, názvoslovných entít, tagov, N-gramy (možno v budúcnosti))
    - KWIC (keyword in content) – N-gramy
    - tag-cloud
    - frekvencia slov
    - TFIDF – metrika, ktorá počíta s frekvenciou slova (používa sa napr., aby sa nezvýhodňovali dlhšie články)
    - prideľovanie váhy slovám podľa toho ako často sa používajú
- Treba vedieť odpovedať na dotazy: histogram mužských rodov, ...
- Treba vedieť ku každému článku pridať tag, aby sa potom dali hľadať podobné

Rozhranie pre import:

- Môže byť CLI (script bez GUI)
- Musí obsahovať:
    - import: názov článku, text
    - tokenizáciu (rozsekať a označiť)
    - export

Do budúceho stretnutia:

- Pripraviť štruktúru aplikácie
- Pripraviť štruktúru DB
- Pripraviť plagát
- Low fidelity návrh pre UI
- Premyslieť ako stiahnuť články z wikipédie
- Analyzovať nástroje na spracovanie textu a spraviť podrobný report (čo tam je, či to funguje a či to vieme použiť):
    - text.fiit.stuba.sk – STU FIIT
    - morphodita – Masarykova univerzita

MISC:

- TFIDF odfiltruje často používané slová a ostanú nám slová špecifické pre danú tématiku
- "word2cec" – prevádza slová do vektoru - algoritmus na predpovedanie postupnosti slov (vieme pomocou neho hľadať synonymá)

Odkazy:

- https://korpus.sk/morpho.html → SAV morfológia
- http://slovniky.juls.savba.sk/ → SAV slovník
- https://korpus.sk/morpho.html → SK jazyk

- https://dumps.wikimedia.org/skwiki/latest/ → wiki db dump SK
- http://ufal.mff.cuni.cz/morphodita --> morphoDiTa

Zoznam high-level úloh (product backlog)

- Vytvorenie webu tímu + administrácia: Paťo 8 SP (story-points)
- Vytvorenie plagátu: Júlia 3 SP
- Navrhnutie štruktúry mongoDB: Danko, Alan, Krištof 3-5 SP
- Porovnať existujúce klientske aplikácie pre mongoDB: Alan 3 SP
- Wireframe-y pre web: Paťo 5 SP
- Zistiť možnosti importu článkov z wiki: Peťo, Dávid 5-8 SP
- Vytvoriť iniciálny insert pre vzorové články (v JSON štruktúre): Danko, Alan, Peto 2-11 SP
- Nájsť nástroje na spracovanie textu (text.fiit.stuba.sk -> report / name tag + morphodita -> Masarykova univerzita): Adam, David 5 SP
- Navrhnúť architektúru systému: Danko, Krištof 3 SP
- Navrhnúť rozhranie pre web. aplikáciu (zobrazenie dát z DB do user-friendly formy) 5 SP
- Vytvoriť crawler na získavanie žurnálových článkov (sme, aktuality, sita, tasr) 3 SP

mongoDB

- Vytvoriť štruktúru
- Collection: články (názov, text článku, tagy o článku (šport, veda, atď.), tokeny -> pole slov, ktoré uchováva pole informácií)
- Korpus (zhluk článkov) – napr. všetky články z wiki
- Dokument: článok
- Uchovávanie spracovaného textu (rozdelený na slová, pre každé slovo uchovať jeho lemu (peknej -> pekný))
- Uchovať gramatické kategórie (POS → part of speech (rod, číslo, pád, vzor))
- Uchovať názvoslovné entity (NER → name entity recognition (podstatne etc.))
- Kategórie NER (osoba, (person from locality → bratislavčan (variable)), lokalita, organizácia, dátum, čas, mena (currency))
- Môžeme uchovať duplicity (ak to ma význam → selecty etc.)
- Compass, mongovue, mongo rock (alebo iný client)

Rozhranie pre vizualizaciu

- Bude obsahovať zoznam korpusov, ktorý bude obsahovať:
    - zoznam článkov
    - informácie o dĺžke článku / počet slov/ počet plnovýznamových slov / počet slovies/ atď.
    - umožniť pridať k článkom tagy
    - zobrazenie N-gramov, najčastejšie vyskytujúce sa N-tice slov
    - pozrieť WORD 2 VEC
    - funkcie and korpusom:
        - histogram (podľa ľubovoľnej kategórie) + v budúcnosti histogram N-gramov

- KWIC – keyword in content
- tagcloud
- frekvencia slov (therm frequency) – vzorec na to, ako často sa používa slovo a aká je jeho priorita
- TF-IDF

Rozhranie pre import

- Môže byť comand line skript
- Spraviť iniciálny upload do DB (názov článku + text)
- Spraviť tokenizáciu (otagovať článok, rozdeliť na slova etc.)
- Spraviť export (do rozumného formátu - JSON)

Zdrojové dáta

- Zdroje: Wikipedia , žurnálové články (na wiki robiť export článkov)
- Pre wiki spraviť prieskum/analýzu, že ako chceme získavať údaje (či exportom, pomocou URL získať JSON), crawlerom, wiki dump-om, ...)

Zápisnica 3

Dátum	03.10.2018 – 12:00
Miesto	FIIT STU, 4.26
Zapisovateľ	Dávid Csomor

Úvod - pripomienky

- Dávid, Adam, Krištof – doplniť výstup o ARL6
- Paťo – upraviť textový wireframe, resp. pridať TABy
- Krištof, Peťo, Paťo – integrovať existujúce skripty do jedného celku
- Dano, Alan – doplniť schému (premiestniť celú kolekciu „tokens“do konkrétnych článkov, vyhnúť sa odkazovaniu F-key)
- Prešli sme a zhodnotili sme si úlohy, ukončené aj rozrobené
- Analyzovali sme možnosti MS TFS – pokúšali sme sa nájsť iterácie

Úlohy vyplývajúce zo stretnutia

- Doplnenie analýzy dostupných nástrojov. Pridanie nástroja ARL6
- Proces importu
    - návrh
    - implementácia
- Prototyp rozhrania by mal obsahovať:
    - výpis korpusov
    - výpis článkov korpusu
    - zobrazenie detailu článku s tab-mi
    - TAB1 – celý článok
    - TAB2 - zoznam tokenov a ich slovný druh
    - TAB3 - NER
    - TAB4 - usporiadaný zoznam slov
    - TAB5 - histogram (koľko podstatných mien, prídavných mien, atď.)
- Crawler na extrakciu článkov z redakčných webov (Future task)
- Možnosti filtrovania v MongoDB
    - Výstupom je krátky jednoduchý tutoriál
    - Vybrať články z daného korpusu
    - Vybrať články, kde je aspoň jedna entita typu „lokality“
    - Vybrať články podľa nejakého tag-u
    - Zoradiť články podľa počtu token-ov
    - Zobraziť články podľa počtu slovies – console form
- Celková architektúra (dokončiť, prerobiť do virtuálnej formy a uložiť)
- TP Cup vyplniť dokumenty spojené s prihláškou

TP Cup

- Zodpovedná: Júlia
- Dokončiť dokumenty týkajúce sa prihlášky na TP Cup
- Počet SP (story points): 3
- Priorita: 1

Dplnenie analýzy dostupných nástrojov.

- Zodpovedný: Dávid
- Pridanie nástroja ARL6 do analýzy
- Porovnanie výstupov všetkých nástrojov
- Počet SP: 5
- Priorita: 2

Upratať TFS

- Zodpovedný: Peťo
- Zjednotiť pravidlá týkajúce sa systému TFS a publikovať ich – spraviť manuál
- Počet SP: 5
- Priorita: 1

Proces importu

- Zodpovedný: Krištofovi
- Počet SP: 8
- Priorita: 2

Prototyp rozhrania

- Priradené Patrikovi
- Počet SP:13
- Priorita: 2

Crawler

- Priradené: -
- Priorita: 4
- Estimated: -

Moznosti filtrovania z MongoDB

- Zodpovední: Daniel a Alan
- Počet SP:8
- Priorita: 2

Architektúra systému

- Zodpovední: Alan
- Počet SP: 3
- Priorita: 2

Zápisnica 4

Dátum	09.10.2018 – 12:00
Miesto	FIIT STU, 4.26
Zapisovateľ	Dávid Csomor

Úvod - pripomienky

- Alan s Danielom spravili API pre pridávanie/ update-ovanie korpusov/ článkov a naštudovali si prácu s MongoDB
- Peťo spravil manuál pre TFS, upravil celkové rozloženie práce v TFS a v TFS sa vďaka tomu už dá pekne orientovať. Taktiež bol vyriešený problém s prepojením GITu a TFS.
- Júlia pokračovala v práci s TP-CUPom
- Paťo zdokonaľoval frontend
- Krištof riešil úpravu skriptov na ťahanie dát
- Adam s Dávidom robili analýzu a prehľad o projekte

Úlohy vyplývajúce zo stretnutia

- Vytvoriť kolekciu synoným (synonymické množiny/ sety)
- Z textu potrebujeme vyťahovať features, na základe ktorých vieme text klasifikovať
- Zamyslieť sa, že aké features by sme používali:
    - distribúcia podstatných mien
    - subj. obj. slov
    - percentuálny podiel interpunkcií
    - medián (pre vylúčenie outlierov)
    - priemerná dĺžka vety
    - podiel slov, ktoré sú lokácia
    - podiel slov, ktoré nevieme identifikovať
- Zamyslieť sa nad uchovávaním slov pomocou invertovaného indexu a pre naše účely aj naopak (treba mať „inverzný" skript ku inverznej indexácií)
- Prípadne spracovať inú sekciu Wiki (zoznam miest, zoznam vrchov nad 400m +-, rieky nám vie poskytnúť product owner)
- Synonymá hľadať až po lematizácií
- Pozrieť sa na slovník „wordnet na JULS", ktorý reprezentuje synonymické sety pomocou ID
- STOP slová by sme si mali vybudovať samy (z invertovaného indexu sa to dá zistiť)

Zápisnica 5

Dátum	17.10.2018 – 12:00
Miesto	FIIT STU, 4.26
Zapisovateľ	Dávid Csomor

Úvod - pripomienky

- Import je vyriešený
- Anotácia je vybavená (ale je problém s obmedzeným počtom znakov zo strany API product ownera)

Organizácia + návrhy na zlepšenie

- Bude sa treba vážne venovať validácií
- My ako tím sme mali zistiť koľko story pointov vieme dodať za týždeň
- Môžeme to robiť benevolentnejšie v zmysle, že ak bude "to" hlavné vybavené, tak môžeme robiť aj veci z/do budúcnosti (nejaké veci , ktorým sa budeme venovať v budúcnosti)
- Mali by sme robiť 9h/ týždenne na člena
- V rámci tímového projektu, sa robí viacero parserov (anotácia, invertované indexy)
    - vieme si to rozdeliť a robiť ako keby dopredu
- Zváženie zapojenia sa do "blbec dňa":
    - Product owner sa stretne s BD a pokúsi sa vybaviť nacrawlované dáta
- Streda 31.10 nebude stretnutie → rozdelíme šprinty na 1T a 2T

Diskusia k minulým úlohám

- Velká relačná tabuľka, kde sa bude nachádzať:
    - slovo
    - lema
    - POS
    - NER (pole)
    - počet
    - tfidf-document (frekvencia výskytu slova z hľadiska dokumentu)
    - tfid-corpus
    - Ngrams (pole ngramov)
- Pre každý spracovaný článok treba vytvoriť dátum
- Zatiaľ netreba riešiť tfidf
- Drilldown je slovo ←→ článok tak, že budeme môcť postupne konkretizovať filter a zmenšovať zobrazené výsledky
- Pri 2 článkoch rozdelíme obrazovku na 2 časti a štatistiky budú pod tým
- Najjednoduchšie na parsovanie sú webnoviny (neexistuje rozumná alternatíva)

- Reportovať API product ownera cez slack ak by dobre nefungovalo
- Product owner má slovník, ktorý nám vie poslúžiť na hľadanie synoným + Azet slovník
- Slovník sa bude budovať iba na kolekciu top 1000 po tom, čo spravíme invertovaný index

Úlohy vyplývajúce zo stretnutia

- Pridať do datasetu mestá, vrchy, osobnosti (osobnosti z wiki)
    - tie osobnosti bude treba manuálne naklikať
- Začať robiť invertovaný index
- **Dokumenty:**
    - dokumentácia k riadeniu (-) - (vždy musí byť po šprinte)
    - dokumentácia k inžinierskemu dielu (architektúra a pod.) - (na konci semestra)
- Od 2. šprintu treba lepšie písať user stories
    - pridávať aj akceptačné kritériá
    - pridávať opis
    - user stories robíme my ako tím/členovia
    - vedúci sa stará a FEATURES nie U.S.
- Pridať dátum „kedy sme článok pridali“, „kedy sme ho oanotovali“, „kedy sme vytvorili invertovaný index“
- Články by sa mali dať reanotovať
- Byť schopný pridať dátum a anotáciu automaticky napr. pre články v korpuse Slovensko, sa pridá dátum a tag Slovensko pre všetko v corpuse Slovensko automaticky
- Webcrawler pre Webnoviny – téma: šport
    - treba aj dokument ku crawleru
    - treba, aby bola zachovaná hierarchia
    - dátum, že kedy sa to stiahlo
- Vyriešiť taxonómiu (ako budovať slovník)
    - ísť na stránku s cudzím jazykom a odtiaľ si stiahnuť/ inšpirovať sa „slová + vzťah“
- Pozrieť fast-text word2vec
- Spraviť kolekciu pre vety

Zápisnica 6

Dátum	24.10.2018 – 12:00
Miesto	FIIT STU, 4.26
Zapisovateľ	Dávid Csomor

Úvod – pripomienky

- Crawler je hotový ako samostatný skript
- Peťo si pripravil analýzu features:
  - skúšal niekoľko indexov na sledovanie náročnosti textu
- Metrika od product ownera:
  - koľko z našich slov je v TOP 500
- 300 pridaných osobností postačí
- Invertovaný index má spravený prototyp rozhrania a základnú schému
- Keď už budeme mať invertovaný index, tak až potom by sme spravili slovník

Úlohy

- Poslať najdlhší článok product ownerovi, nech môže nastaviť limit
- Dokážeme identifikovať rozlišovaciu stránku?
  - také spoznáme tak, že má na začiatku „môže byť" (iba ak je to na začiatku)
  - zrušiť rozlišovacie stránky
- Zachovať odseky ak sa nám to podarí
- Pridávať bodky:
  - za nadpisy vo wiki
  - za nadpisy v článkoch
- Tlačidlá typu „tokenizovať", by sme mali vedieť v budúcnosti skryť
- Odstraňovať z článkov z web-novín pojem „Bratislava..."
- Pridať TAB: „Analyzovaný text"
- V TABe „Zoznam článkov" pridať počet znakov/slov článku
- Pridať filter na korpus (SELECT)
- Pouvažovať nad aktivitou, ktorú by sme vložili do nášho inteligentného analyzátora
  - zoberieme článok, nejaké typické slova a kto prvý uhádne kategóriu
  - napr. slovo „futbalista" – uhádnuť, že do akej kategórie patrí
  - Prípadne pouvažovať nad niečím ako „Akinátor"

Invertovaný index

- Vytvoriť IE podľa predpisu v TFS – Features - „9321 Vytvoriť invertovaný index pre tokeny z článkov"

Zápisnica 7 - MTS

Dátum	07.11.2018 – 17:00
Miesto	FIIT STU, 4.26
Zapisovateľ	Dávid Csomor

Krátke zhrnutie

- Product backlog by mal byť „určitý zoznam“, ktorý by už mal obsahovať v nejakej miere všetky featury/funkcionalitu, ktorá sa od nás bude požadovať po dvoch semestroch
- Spokojnosť s backlogom by mala byť zaevidovaná do nejakého dokumentu
- Backlog musí určite obsahovať úlohy na viaceré šprinty (väčší scope než len na jeden šprint)
- Backlog by mal mať úlohy jednoznačne zoradené podľa priority
- Treba si určiť definition of done
  Velocity by sa mala eventuálne ustáliť
- Treba si dobre spísať retrospektívu a pomenovať veci (good, bad idea) (začať, prestať, pokračovať robiť)
- Malo by existovať jedno miesto, kde je určená metodika, teda kto sa bude starať o aké veci
- Cvičenie ohľadom rizík, ich identifikácia a eliminácia

Zápisnica 7

Dátum	07.11.2018 – 12:00
Miesto	FIIT STU, 4.26
Zapisovateľ	Dávid Csomor

Úvod – pripomienky

- Rekapitulácia šprintu
- Stačí aby sa weby (články z nich) zatiaľ importovali cez Postman, netreba mať rozhranie
- Pri invertovanom indexe prípadne pridať zoraďovanie podľa atribútov
- Neposielať request na analýzu textu nad viacerými (100+) článkami paralelne
- Treba zmeniť prerozdeľovanie úloh

Úlohy

- Otestovať invertovaný index
- Text pri nespracovaných článkoch odstrániť (zatiaľ stačí vložiť 1-2 kategórie)
- Prípadná úloha: „hľadať chyby vo webovom riešení“
- Premenovať frontend na niečo viac používateľsky prívetivé
- Pridať filter na POS a na NER pri invertovanom indexe
- Pridať logovanie a kontrolu syntaxu do pythonu
- Začať spisovať návody, že ako použiť časti nášho riešenia

Zápisnica 8

Dátum	14.11.2018 – 12:00
Miesto	FIIT STU, 4.26
Zapisovateľ	Dávid Csomor

Úvod - pripomienky

- Preberanie súvislostí medzi MTS a TP
- pokiaľ nejaké funkcie prechádzajú evolúciou, tak majú testy zmysel
- prezentácia jednotlivých členov tímu za posledný šprint
- Pre histogram spraviť jednu veľkú tabuľku alebo to rozdeliť na taby?
- ísť podľa značiek na úrovni znakov
- najprv si spraviť mapu (POS → POČET), kde kľúč by bol POS
- spraviť prototyp podľa uváženia a potom budeme na základe toho pracovať
- Invertovaný index prepojiť s článkami
- V súčasnej verzii zatiaľ netreba riešiť pády, čísla a pod.
- Treba eventuálne pridať indexy
- Z webnovín zatiaľ ťaháme 3 kategórie
- zvážiť rozdelenie databázy na testovaciu a produkčnú

Úlohy

- Do budúceho Štvrtku treba pripraviť veci na TP-CUP
- keď bude viac článkov, zmeniť filtrovanie tak, aby to hľadalo/dávalo radšej menej výsledkov
- TFIDF má byť nad korpusom aj nad článkom
- Pri tom najglobálnejšom TFIDF by bolo ideálne postupovať tak, že počet výskytov celkovo / počet výskytov v článkoch
- Počet desatinných miest orezať v tabuľke

Čo začať robiť:

- Retrospektívy
- Online prezentácie na úrovni Features
- Začať robiť skorej na šprinte
- Skúsiť wiki/ spraviť analýzu

Zápisnica 9

Dátum	21.11.2018 – 12:00
Miesto	FIIT STU, 4.26
Zapisovateľ	Dávid Csomor

Úvod – pripomienky

- Prezentácia týždňového pokroku podľa prezentácie „9T- Prehľad úloh"
- Spoločná prehliadka a konzultácia predbežnej verzie dokumentu (PVD)
    - treba sa rozhodnúť či chceme pridať inštalačnú príručku už v tejto verzií
- Zápisnice nechať v predčasnej verzií dokumentu (prípadne to dať ako prílohu)
- Sprístupniť API k službám v dokumente aj keď ten dokument bude verejný (?)
- Zlepšenie Administrácie
    - fungujú URL pri Pythone
    - integrácia Angularu do Expressu je takmer hotová
- Zobrazenie analyzovaných tokenov
    - API pre histogram článku funguje podľa požiadaviek
- Sprehľadnenie rozhrania
    - boli doplnené potrebné API pre jednotlivé komponenty
    - treba vyriešiť triedenie pri invertovanom indexe, nakoľko sa to zoraďuje nevyhovujúcim spôsobom (nie AaBb-Zz, ale ABC-xyz)
- Oddeliť verejné funkcionality od správy webstránky
    - používateľ sa už vie prihlásiť na frontende
    - na backende je taktiež API, ktorá používateľovi vygeneruje token na 24 hodín, s ktorým nemá žiadne obmedzenia
    - registrácia na frontende nebude potrebná/ vyžadovaná
- Vytvoriť prepínanie prostredí DB
    - bola vytvorená kópia databázy, teda produkčná/ vývojová databáza
    - prostredie pre prepínanie spomínaných databáz ešte nie je kompletne hotové
    - prípadne pridať používateľovi možnosť prepínať sa medzi produkčnou a testovacou databázou
        - to by bolo veľmi náročné, nakoľko momentálne rozlišujeme vo všeobecnosti dva stavy: prihlásený a neprihlásený
- Tvorba dokumentu pre mentoring CP CUP
    - prídu odborníci z praxe na 2 stretnutia, ktorí nám budú radiť ako naše riešenie zlepšiť
- Zobraziť štatistiky o aktuálnych dátach
    - nakoľko ešte API na poskytovanie štatistík nefungujú úplne dobre,"landing page" pre štatistiky bude hotový až na konci šprintu
    - product owner odporúča cache-ovať obsah DB raz za 24 hodín a potom nad tým vykonávať štatistiky

    - (ale) invertovaný index by mal byť aktualizovaný častejšie než raz za 24 hodín
-

Úlohy

- Dokončenie predbežnej verzie dokumentu má najvyššiu prioritu!
- Pokračovať na rozpracovaných úlohách

Zápisnica 10

Dátum	28.11.2018 – 12:00
Miesto	FIIT STU, 4.26
Zapisovateľ	Dávid Csomor

Úvod – pripomienky

- Riešenie bug-u zoraďovania v indexe, prepojenosti úloh v TFS
  - stačí zoraďovať podľa tfidf, count
- Vytvoriť user-stories pre chyby (?)
- Features v TFS zoradil product owner podľa dôležitosti
- Treba sa dohodnúť v akom „stave", v akej „forme" a aké dáta budeme mať
  - Spojiť korpusy „štáty" a „mestá" pod názov „GEO" (+- 500)
- Potreba prepracovať crawler nakoľko už začína byť blokovaný
- Treba zapracovať do skriptu, aby sťahovalo po častiach, nech môžeme sťahovať z viacerých strojov
- Odporúčanie:
  - najprv sa pozrieť, či aktuálna URL nebola crawlovaná posledný týždeň
    - Ak áno: zobrať HTML z cache
    - Ak nie: stiahnuť
- Vybrať kategórie, ktoré pridať do korpusov
- Treba používať relatívne metriky (relatívny počet výskytu nejakého slova vs absolútnemu výskytu)
- Odtlačok článku je spredmetnený v zistení:
  - Počte podstatných mien
  - Koľko slov článku je patrí do TOP 1000
  - ...

Úlohy

- Vybrať kategórie, ktoré pridať do korpusov
- Spraviť „exact match" pri vyhľadávaní v invertovanom indexe
- Prepracovať crawler
  - aby nebol blokovaný stránkami, ktoré crawluje
- Pridať na úvodnú stránku nejakú zaujímavú štatistiku

Zápisnica 11

Dátum	05.12.2018 – 12:00
Miesto	FIIT STU, 4.26
Zapisovateľ	Dávid Csomor

Úvod – pripomienky

- Prehľad progresu za posledný týždeň
  - Peťo sa venoval crawleru a integrácii pythonu s DAO
- Krištof
  - vyhľadal osobnosti a dal do DB
- Adam
  - porobil úlohy na tento šprint, ale ešte to potrebuje skontrolovať a pushnúť
- Alan
  - Spravil API-ny podľa špecifikácie
- Prehľad o náležitostiach TP-CUPu
  - sústrediť sa viac na biznis stránku/prezentáciu
- Prípadne použitie funkcií LIMIT a SKIP kvôli veľkému množstvu dát
- Posielať 2 requesty (1. request/filter, 2. limit/skip)
- Diskusia k manažérskym úlohám v tíme a príprava na prezentáciu

Úlohy

- Presunúť filtrovanie z frontendu na backend (eventuálne – ďalší semester)
- Notifikovať vedúceho ak sa spraví nejaká zmena
- Určiť si niekoho na kontrolu kvality, aby sa našou stránkou preklikával a testoval ju
- Treba spraviť dokumentáciu k mongo selectom a API od Alana

Zápisnica 12

Dátum	12.12.2018 – 12:00
Miesto	FIIT STU, 4.26
Zapisovateľ	Dávid Csomor

Úvod – pripomienky

- Product owner spravil aj vlastný crawler v prípade potreby
- Server sa začína s pribúdajúcimi článkami spomaľovať
- Uvažovať nad založením anonymného docu na zhromaždenie názorov
- Zváženie skoršieho šprintu na začiatku ďalšieho semestra

Úlohy

- Doriešiť odtlačok článku cez prázdniny

Retrospektíva

- Potreba rýchlejšie aktualizovať úlohy v TFS nech má každý lepší prehľad o svojich úlohách

Zápisnica 13 – pred-semestrálne stretnutie

Dátum	07.02.2019 – 14:00
Miesto	FIIT STU, 4.26
Zapisovateľ	Dávid Csomor

Úvod – Motivácia - Stratégia

- Tímové stretnutia môžeme potenciálne mať v Utorok/Štvrtok 8-11:00
- Vybrať si akým smerom sa chceme uberať (biznis/marketing/programovanie)
- Pristupovať k tímovému projektu tak akoby jeho výsledok mohol slúžiť ako dlhodobá referencia
- Mali by sme si určiť roly v tíme tak, aby bolo jasné, že čo sme robili (kvôli spätnej väzbe)
- Pozrieť si pojem „Hedonická adaptácia
- 1. šprint naplánujeme v 1. týždni semestra

Marketing

- Dobre si pripraviť niekoľko viet (nie veľa)
- Video sa robí v rámci TP-CUPu (iba finalisti a semifinalisti)

ToDo

- Dohodnúť si stretnutie na refaktoring/upratanie kódu
- Zvážiť testovanie
- Treba doplniť:
    - štruktúru databázy
    - opísať parser
    - problémy, ktoré sme riešili napr. pri implementácii parseru
    - upraviť/vylepšiť querries/requesty/stránkovanie
- Premyslieť API na sortovanie
- Pridať kategórie: Krimi, Noviny, Šport

Requesty (návrh implementácie)

- V prvom requeste sa vykoná filtrovanie + count (NEsortovať)
- V druhom requeste filtrovanie, sortovanie, potom pagging

Zápisnica 14

Dátum	14.02.2019 – 08:00
Miesto	FIIT STU, 4.26
Zapisovateľ	Dávid Csomor

Úvod – začiatok LS

- Prebranie refaktoringu (počítať s tým, že pre front-end bude treba prerobiť spôsob selektovania a posielania dát)

ToDo

- Pridať k článkom „politika“ a „krimi“
- Ak má článok pod nejakú hranicu znakov, tak zvážiť jeho preskočenie
- Pozrieť linky na stránke, ktoré sa odkazujú na invertovaný index
- Z názvov datasetov odstrániť „_webnoviny“
- Pridať z jQuery plugin „countto“

Zápisnica 15

Dátum	19.02.2019 – 08:00
Miesto	FIIT STU, 4.26
Zapisovateľ	Dávid Csomor

Rekapitulácia

- Prebratie postupu ohľadne crawlovania a doplnenie a doplnenie stiahnutých vecí
- Zjednotenie jazykov rozhrania sa podarilo
- Oprava schémy a invertovaného indexu
- O-indexovanie a otokenovanie 150 článkov trvá tak 5 minút
- Krištof navrhuje porozmýšľať nad presunom indexovania a tokenizácie z Node.js do Python časti
- Danko premenoval corpusy s web novinami, upravil výzor frontendu podľa požiadaviek zadávateľa, na backende pridal cache pre urýchlenie činnosti systému
- Treba kontaktovať Peťa Lacka
- TP-CUP:
    - Nepísať o technológiách v článku
    - Vždy to bude o kategorizácii článkov – klasifikovanie článkov
    - Prípadne vložiť „progress bary" ohľadne náročnosti, tématiky
    - Vizualizácia informácii o článku je náš primárny cieľ
        - Na koľko percent je taký a na koľko percent je taký
        - Vložiť niekoľko slov a vizualizácia ich zaradenia
- Používať iba 2 NERY (nepoužívať dictionary 13)
- Odtlačok článku
    - Mať stĺpce ako článok, kategória, Podstatné mená, Prídavné mená, ...
    - Riadok bude mať teda článok, kategóriu a % podiel jednotlivých slov
-

Ďalší týždeň

- Spraviť histogram slov a podľa pokynov v TFS
- Poslať priebežnú správu do Piatku product ownerovi
- Pripraviť „elevator pitch" pre iných ľudí a zjednotiť sa ohľadne toho
    - Každý vytvorí dokument, v ktorom opíšeme náš tímový projekt tak, aby to to bolo:
        - jednoduché na vysvetlenie
        - výstižné
        - dalo sa to vysvetliť za čo najkratší čas

Zápisnica 16

Dátum	26.02.2019 – 08:00
Miesto	FIIT STU, 4.26
Zapisovateľ	Dávid Csomor

Úvod - pripomienky

- Invertovaný index potrebuje optimalizáciu
    - Invertovaný index robiť z hľadiska článku
    - Ak je slovo viackrát v článku, stačí spraviť 1 insert
    - Ak je slovo v rôznych článkoch, spraviť toľko insertov pre slovo v koľkých je článkoch
    - Kombinácia: slovo-léma-článok musí byť v invertovanom indexe unikátna
    - Z NER spraviť string namiesto poľa (zabrániť tmu, aby sa stalo A-B, B-A)
    - Zoradiť NER podľa abecedy
    - Merge-núť NER záznamy
- Import článkov od vedúceho
    - Bol pustený lokálne na produkčnú databázu
- Ešte nie je dokončený refaktoring front endu
- Odtlačok článku – kolekciu nazvať: Imprint

Ďalší šprint

- Zmena schémy
- Upload vlastného textu do používateľa 500 znakov
    - Umožniť analyzovať text pre nezaregistrovaných používateľov ako ukážku
    - Umiestniť do vlastnej kolekcie user-upload a potom to na základe textu priradiť do kategórie
    - Uvažovať nad tým, že by sa ten text nikam neukladal
- Porozmýšľať nad nejakou drobnou aktivitou pre používateľov na našej stránke
    - Ukázať náhodnú vetu z náhodného článku a nechať ľudí hádať, že do akej kategórie patrí ten článok
    - Pridaná hodnota by bola zistenie či naše algoritmy pracujú správne a keď tak ich vylepšiť

Zápisnica 17

Dátum	05.03.2019 – 08:00
Miesto	FIIT STU, 4.26
Zapisovateľ	Dávid Csomor

Úvod – pripomienky

- Danko
  - Začal robiť živý analyzátor
  - Zatiaľ je upload ako mockup
- Adam
  - Spravil rozhranie (TAB) pre odtlačok článku
- David
  - Robil odtlačok článku
  - Vie sa vytvoriť histogram, ale je problém s push-ovaním do databázy
- Krištof
  - Pracoval primárne na invertovanom indexe
  - Prerobil to a doladil to
  - Groupby zaberá asi najviac času – celkovo to trvá relatívne dlho
- Patrik
  - Spravil optimalizáciu vizualizácie
- Alan
  - Prerábal selecty za účelom zrýchlenia/optimalizácie
- Znížiť prioritu vytvorenia aktivít pre verejnosť na minimum (voliteľné)
- Preberal sa úvod do strojového učenia
  - Pre zaradenie článku sa nájde jeho podobnosť s priemerom z článkov z korpusov
  - Entropia – „bordel" – ak sa zoberie nejaký atribút a podľa neho sa strom rozdelí, tak ako veľmi sa dáta v strome „upracú"
  - Úloha odtlačkov je nájsť také stĺpce, ktorých hodnoty budú výpovedné a budú sa dať podľa nich aj zaraďovať neznáme články do korpusov
- Návrh pre aktivitu pre verejnosť
  - Dať používateľom vetu a má ju zaradiť do kategórie
  - Prípadne dať používateľovi prečítať článok a má určiť percentuálne zaradenie, napr. „70% šport, 30% politika"
- Predstavenie verejnosti
  - Treba povedať v 2-3 vetách, že prečo by to mal ten človek chcieť a ako mu to pomôže v jeho živote
  - Nespomínať architektúru systému
  - Držme sa témy: „či sú články vhodné/nevhodné pre istú kategóriu poslucháčov"
  - Potenciálny biznis „pitch/case":

Zápisnica 18

Dátum	12.03.2019 – 08:00
Miesto	FIIT STU, 4.26
Zapisovateľ	Dávid Csomor

Úvod – pripomienky

- Niektoré slová sú zle otokenované a označené
- Odtlačok článku:
  - Plnovýznamové slová
  - Neplnovýznamové slová
  - STOP slova – budú také, ktoré sa budú nachádzať vo veľa článkov – z tfidf
  - TOP.global 1000 – zatiaľ nie
  - TOP.$corpus 1000 – zatiaľ nie
  - Pozitívne, negatívne, neutrálne – zatiaľ nie

Ďalší šprint

- **Odtlačok/TFIDF korpusu vs článku**
- Každý článok by mal mať odtlačok
- Každý korpus by mal mať odtlačok
- Porovnanie odtlačku korpusu vs článku
- Vyberieme nejaký token článku a porovnáme pravdepodobnosť výskytu tokenu v iných článkoch (prípadne korpusov)
- "Článok X, sa podobá na 20% článku Y"
- „Článok X má takéto slová a na základe toho bol priradený do korpusu Z"
- Pri porovnaniach, porovnávať články s článkami a korpusy s korpusmi
- Vybrať 150 slov z každého článku a počítať
- 1 tabuľka pre článok a 7 tabuliek pre korpus

- **Miniaplikácia**
- Pouvažovať nad interaktívnou miniaplikáciu za účelom zbierania dát od používateľov

Zápisnica 19

Dátum	19.03.2019 – 08:00
Miesto	FIIT STU, 4.26
Zapisovateľ	Dávid Csomor

Úvod – pripomienky

- Do dokumentácie treba podrobne opísať:
  - Postup ako crawler spracováva text
  - Všetky mongo SELECTY
- Pridať do kategórie články SELECT na korpus ešte pred načítavaním
- Do invertovaného indexu dať jeden článok natvrdo ako ukážku pre používateľa

Ďalší šprint

- **Miniaplikácia – Hra**
  - Hra už má hotová frontend
  - Treba ukladať: vetu, corpus, typ v % z celkového rozpočtu
  - Minimum 2 slová a sloveso
  - Veta sa začína {S} a končí {/S}

Zápisnica 20

Dátum	26.03.2019 – 08:00
Miesto	FIIT STU, 4.26
Zapisovateľ	Dávid Csomor

Úvod – pripomienky

- V invertovanom indexe pri niektorých slovách chýbalo TFIDF v develop DB
- IT-SRC
    - chýba plagát
    - treba dopracovať pripomienky
- Pri vyhľadávaní článku cez web nefunguje ku dnešnému dňu „exact match“ – treba prerobiť
- V power pointe nastaviť viac DPI pri vytvárania plagátu

Ďalší šprint

- Opraviť pozadie/text v interaktívnej hre
- Dorobiť insertovanie do databázy
- Prerobiť „exact match“ vo vyhľadávaní článkov
- Opraviť padding v komparátore

Zápisnica 21

Dátum	02.04.2019 – 08:00
Miesto	FIIT STU, 4.26
Zapisovateľ	Dávid Csomor

Úvod – pripomienky

- Prebranie interaktívnej hry – pripomienky
- Preberanie dokumentov na IIT.SRC

Prebrané skutočnosti

- Spoločné vypracovanie textu k videu na robime.it
- Prebranie štruktúry plagátu na IIT.SRC a potrebných náležitostí
- Prebranie videí konkurencie na robime.it
- Možnosti oživenia Live Analyzátora

Ďalší šprint

- Plagát:
    - Nadpis, mená, formality
    - Spraviť strom zo slov
    - Pridať TOP 5 vecí, ktoré riešime
- Leták:
    - Názov tímu, URL, 2-3 vety
    - Cca A5 – A6

Zápisnica 22

Dátum	09.04.2019 – 08:00
Miesto	FIIT STU, 4.26
Zapisovateľ	Dávid Csomor

Úvod – pripomienky

- Pri TP CUP môžeme dať požiadavku na väčší monitor

Prebrané skutočnosti

- Plagát na IIT.SRC

Zápisnica 23

Dátum	16.04.2019 – 08:00
Miesto	FIIT STU, 4.26
Zapisovateľ	Dávid Csomor

Úvod – pripomienky

-

Prebrané skutočnosti

- Interaktívna hra
- Oprava webu pre prehliadače iné ako Google Chrome
- Prebranie harmonogramu a priebehu IIT.SRC/TP CUPu
- Rýchlosť MongoDB vs klasickej relačnej DB

IIT.RSC - skutočnosti

- Postupuje 6 tímov z 11

Zápisnica 24

Dátum	25.04.2019 – 08:00
Miesto	FIIT STU, 4.26
Zapisovateľ	Dávid Csomor

Úvod – pripomienky

- Prebranie účasti na IIT.SRC
- Prebranie skutočností ohľadne posledného šprintu a organizácie
- V dokumentácií opísať najmä inžinierske procesy + ako robíme s článkami
    - Ako presne robiť import článkov
    - Ako presne pracovať s invertovnaým indexom
    - Ako pracovať s Mongo databázou
- Môžeme opísať proces živej analýzy (ale asi to bude duplicita)

Ďalší šprint

- Pripraviť naše riešenie na odovzdanie
- Finalizácia dokumentu

Zápisnica 25

Dátum	07.05.2019 – 08:00
Miesto	FIIT STU, 4.26
Zapisovateľ	Dávid Csomor

Témy prebrané na stretnutí:

- Prebranie náležitostí potrebných k odovzdaniu finálnej verzie riešenia
- Kontrola finálnej dokumentácie

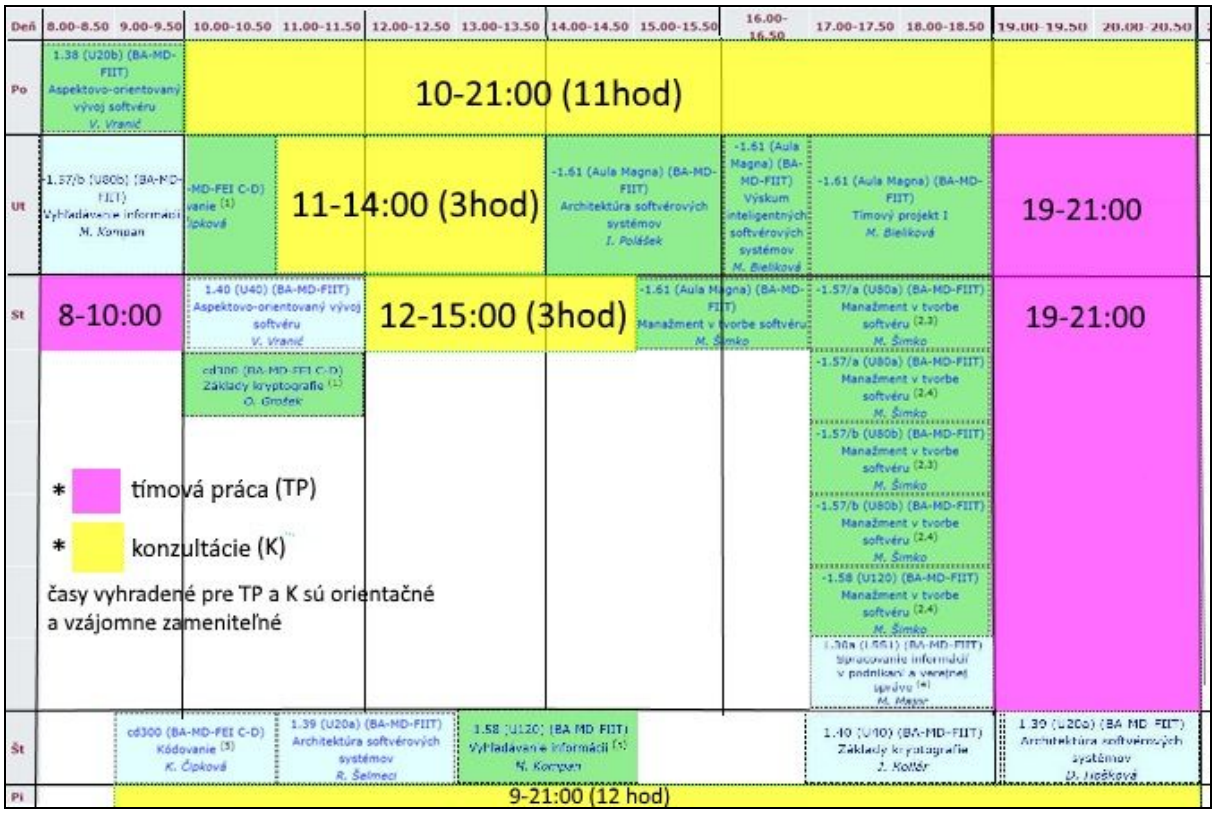
Príloha A-1: Preferencia tém

1. Prostredie pre inteligentnú analýzu textov [TxtEnv]
17. Analýza správania sa vozidiel v meste [SmartMobility]
11. Databanka otázok a úloh [FIIT - DU]
5. Podpora výskumu behaviorálnej biometrie [behametrics-learn]
10. IoT systém monitorovania osôb [Breyslet]
2. Inteligentný importér verejných dát [Importer]
15. Monitoring antisociálneho správania [MonAnt]
3. Vyhľadávanie pomocou obrázkov [ImageSearch]
18. Škola hrou vo virtuálnej realite [VREducation]
12. In-memory databáza s využitím GPU [In-memory-DB]
16. Prostredie na vizualizáciu mikrogridu [GridBox]
14. Identifikácia entít – spracovanie textu [SK-CZ-TEXT]
19. Automatické testovanie v prostredí internetu vecí [IoTesting]
13. 3D simulovaný robotický futbal [3D futbal]
7. Vizualizácia softvéru vo virtuálnej a rozšírenej realite (Remake) [VizReal]
9. Vnímanie neviditeľného [Holographic Eyes]
8. Generátor 3D priestoru [3DSpaceGen]
6. Monitorovanie a vyhodnocovanie fyziologických procesov človeka [BioMonitor]
20. WiFi Funtoro [WFuntoro]
4. 3D UML, improved version [3D-UML]
21. Webový prehliadač pre slabozrakých a nevidiacich [Webable]

Príloha A-2: Rozvrhy členov tímu

Nižšie uvedený rozvrh predstavuje kombináciu všetkých rozvrhov členov tímu z časového hľadiska tak, že ak majú dvaja rôzni členovia tímu cvičenie/prednášku z odlišných predmetov v ten istý čas, tak je uvedené iba cvičenie/prednáška jedného z členov tak, aby rozvrh demonštroval časovú nedostupnosť.

Časové úseky vyznačené žltou farbou predstavujú predbežne dostupný priestor na konzultácie s vedúcim projektu a budú neskôr zredukované tak, aby vyhovovali vedúcemu projektu.
Časové úseky vyznačené fialovou farbou predstavujú predbežne dostupný priestor na stretnutie členov tímu a spoločnú prácu na projekte.



Píloha A-3: Prihláška do súťaže TP-CUP

1. Predstavenie tímu

Náš tím sa skladá z 8 členov a dovoľujeme si povedať, že je funkčným celkom. Väčšina z nás sa pozná už od prvého ročníka štúdia na bakalárskom stupni na FIIT STU a nových dvoch členov sme s radosťou prijali tiež. Dokážeme spolupracovať a aj sa ľudsky podporiť. Sme v skutku pracovití a súdržné jadro tímu funguje skupinovo už skoro štyri roky. Najsilnejšou stránkou nášho tímu je naše odhodlanie učiť sa nové technológie a robiť veci najlepšie ako dokážeme.

Každý člen tímu disponuje inými znalosťami, ktoré nadobudol či už počas štúdia, tvorby bakalárskej práce alebo z pracovných skúseností. Viacero členov tímu (Júlia, Peter, Adam, Dávid) sa zaujíma o koncept strojového učenia, na čo sme zamerali aj výber témy. Patrik má výrazné skúsenosti s oblasťou vývoja frontendových aplikácií, dlhšiu dobu sa venuje oblasti tvorby samotných webových aplikácií. Adam má skúsenosti s Angularom, aj keď vo väčšej miere sa venuje backendu a samotnej Jave. Alan disponuje skúsenosťami s databázovými systémami, presnejšie s MongoDB. Krištof je zdatný, okrem iného, v rôznych technických oblastiach. S jazykom PHP sa vo väčšej či menšej miere stretli Patrik aj Daniel, avšak nebránime sa implementácii backendovej časti ani v inom jazyku. Máme skúsenosti s viacerými programovacími jazykmi, napríklad Java, C#, JavaScript alebo Python. Medzi ďalšie zručnosti, ktoré sa oplatí spomenúť platí databáza PostgreSQL, v ktorej sa obzvlášť vyzná Daniel, avšak určité zručnosti má aj zvyšok tímu.

E-mailový kontakt na tím: team5fiit@gmail.com

2. Motivácia

Hlavnou motiváciou pri výbere tejto témy je orientácia sa viacerých členov v kľúčovej problematike. Viacerí členovia tímu majú reálne skúsenosti s podobným typom práce hlavne z prostredia bakalárskych projektov. Peter má skúsenosti s analýzou a klasifikáciou textu v podobe zdrojových kódov a teoretické poznatky v oblasti lexikálnych diel a ich možnej klasifikácie do užších celkov. Nechýbajú ani hlbšie znalosti z oblasti strojového učenia (Peter, Adam, Dávid, Júlia) za použitia knižníc (scikit-learn, graphviz, plot). Júlia môže tiež prispieť znalosťami so spracovaním objemnejších dátových sád (Pandas, numpy). Za výsledné prostredie bude zodpovedný Patrik, ktorý má bohaté skúsenosti i grafické cítenie v tejto oblasti. Daniel spolu s väčšinou tímu má bohaté backendové zručnosti a Alan má priame skúsenosti s používaním MongoDB. Krištof je zdatný v rôznych technických oblastiach, ale aj v oblasti automatizácie.

Hlavným cieľom je vytvorenie funkčného nástroja pre analýzu textov, ktorý nám pomôže prehĺbiť naše doterajšie znalosti v danej problematike a umožní priniesť jedinečné riešenie pre analýzu textov v slovenskom jazyku. V rámci tímu sa chceme bližšie venovať triedeniu článkov do tematických kategórii, prípadne určovaniu vhodnosti textu pre publikum, nakoľko filtrácia nevhodného textového obsahu pre slovenský jazyk výrazne absentuje.

3. Opis projektu

Projekt je zameraný na vytvorenie prostredia pre analýzu textov v slovenčine, nakoľko podobný nástroj zatiaľ pre slovenský jazyk neexistuje. Dôležité tiež je, aby s prostredím dokázal pracovať aj bežný používateľ, čím by sa uľahčil a urýchlil rozvoj tejto oblasti.

Kontext projektu sa opiera o problematiku veľkého množstva textu, ktorým sme obklopení, a nedostatku času, ktorý je potrebný pre poctivé prečítanie týchto článkov a ručné poznámkovanie, prípadne kategorizovanie. Práve k uľahčeniu tejto práce by mali byť nápomocné nové prístupy z oblasti umelej inteligencie, strojového učenia a automatizovaného spracovania textu. V spojitosti s bežným používateľom je vhodné až žiadúce pre rozvoj tejto oblasti, aby sa aj samotný používateľ bez odborných programátorských znalostí mohol podieľať pri učení strojov spracovávať text. Pri zapojení používateľa do procesu učenia strojov pracovať s textom, by sa vývoj algoritmov, porovnávanie a ich vylepšovanie výrazne urýchlilo.

4. Ciele projektu

Vyvíjané prostredie má byť v prvom rade prispôsobené na prácu s kolekciami textov (napr. články z rôznych webových stránok). Tieto texty musia byť efektívne uchovávané spolu s ich metaúdajmi. Metaúdaje sa budú získavať napríklad kategorizovaním vložených dokumentov, identifikáciou charakteristík vybranej skupiny textov, prípadne identifikáciou použitých slov. Všetky údaje je potrebné prehľadne vizualizovať, aby boli interpretovateľné aj menej zdatnými používateľmi.

Pre ukladanie dát použijeme databázu, konkrétne MongoDB, pričom články budú ukladané formou kolekcií. Po manuálnom otagovaní článkov z jedného zdroja bude táto kolekcia článkov tvoriť takzvaný korpus. Pod pojmom dokument budeme rozlišovať už spracovaný text, t. j. rozdelený na slová, lémy, určené gramatické kategórie (POS) a názvoslovná entita (NER).

V rámci nášho tímového projektu budeme vytvárať dve rozhrania. Rozhranie pre import článkov do databázy, ktoré bude obsahovať názov článku, text, tokenizáciu a tiež možnosť exportu tohto článku. Druhé rozhranie bude slúžiť na vizualizáciu dát z databázy, teda ide o vizualizáciu už konkrétnych článkov spolu s ich priemernou dĺžkou, počtom znakov, slov (rôzneho druhu), ... Toto rozhranie bude tiež podporovať funkcionalitu štatistiky nad článkami, ktoré obsahujú zvolené tagy, vytvorenie histogramu (slov, lém, slovných druhov, názvoslovných entít, tagov), vytvorenie takzvaného tag – cloud(u). Pri spracovaní jednotlivých článkov budeme využívať metódu TFIDF, ktorá prideľuje váhy slovám, podľa toho ako často sa používajú, čím zamedzíme zvýhodňovaniu dlhších článkov.

Finálny produkt poskytne používateľovi možnosť pridania nového textu do nami vytvorenej aplikácie, kde prebehne spracovanie tohto článku formou automatickej kategorizácie a otagovania. Ako sme uviedli už v predchádzajúcich kapitolách tohto dokumentu, náš nástroj bude pracovať s textami v slovenskom jazyku, ktoré v súčasnosti medzi dostupnými nástrojmi vysoko absentuje. Pri implementácii sme sa zamerali najmä na využitie technológií NodeJS, Python, MongoDB.

Píloha A-4: Podklad pre TP-CUP mentoring

Tím 05: Prostredie pre inteligentnú analýzu textov [TxtEnv]

Vedúci projektu: Miroslav Blšták

Členovia tímu: Dávid Csomor, Adam Ďuriš, Alan Kováč, Daniel Kováč, Júlia Krajčoviechová, Peter Križan, Patrik Melicherík, Krištof Orlovský

1. Opis projektu

Projekt je zameraný na vytvorenie prostredia pre analýzu textov v slovenčine, nakoľko podobný nástroj zatiaľ pre slovenský jazyk neexistuje. Dôležité tiež je, aby s prostredím dokázal pracovať aj bežný používateľ, čím by sa uľahčil a urýchlil rozvoj tejto oblasti.

Kontext projektu sa opiera o problematiku veľkého množstva textu, ktorým sme obklopení, a nedostatku času, ktorý je potrebný pre poctivé prečítanie týchto článkov a ručné poznámkovanie, prípadne kategorizovanie. Práve k uľahčeniu tejto práce by mali byť nápomocné nové prístupy z oblasti umelej inteligencie, strojového učenia a automatizovaného spracovania textu. V spojitosti s bežným používateľom je vhodné až žiadúce pre rozvoj tejto oblasti, aby sa aj samotný používateľ bez odborných programátorských znalostí mohol podieľať pri učení strojov spracovávať text. Pri zapojení používateľa do procesu učenia strojov pracovať s textom, by sa vývoj algoritmov, porovnávanie a ich vylepšovanie výrazne urýchlilo.

V rámci tímu sa chceme bližšie venovať triedeniu článkov do tematických kategórii, prípadne určovaniu vhodnosti textu pre publikum, nakoľko filtrácia nevhodného textového obsahu pre slovenský jazyk výrazne absentuje.

2. Ciele projektu

Vyvíjané prostredie má byť v prvom rade prispôsobené na prácu s kolekciami textov (napr. články z rôznych webových stránok). Tieto texty musia byť efektívne uchovávané spolu s ich metaúdajmi. Metaúdaje sa budú získavať napríklad kategorizovaním vložených dokumentov, identifikáciou charakteristík vybranej skupiny textov, prípadne identifikáciou použitých slov. Všetky údaje je potrebné prehľadne vizualizovať, aby boli interpretovateľné aj menej zdatnými používateľmi.

Pre ukladanie dát použijeme databázu, konkrétne MongoDB, pričom články budú ukladané formou kolekcií. Po manuálnom otagovaní článkov z jedného zdroja bude táto kolekcia článkov tvoriť takzvaný korpus. Pod pojmom dokument budeme rozlišovať už spracovaný text, t. j. rozdelený na slová, lémy, určené gramatické kategórie (POS) a názvoslovná entita (NER).

V rámci nášho tímového projektu budeme vytvárať dve rozhrania. Rozhranie pre import článkov do databázy, ktoré bude obsahovať názov článku, text, tokenizáciu a tiež možnosť exportu tohto článku. Druhé rozhranie bude slúžiť na vizualizáciu dát z databázy, teda ide o vizualizáciu už konkrétnych článkov spolu s ich priemernou dĺžkou, počtom znakov, slov (rôzneho druhu), ... Toto rozhranie bude tiež podporovať funkcionalitu štatistiky nad článkami, ktoré obsahujú zvolené tagy, vytvorenie histogramu (slov, lém, slovných druhov, názvoslovných entít, tagov), vytvorenie takzvaného tag – cloud(u). Pri spracovaní jednotlivých článkov budeme využívať metódu TFIDF, ktorá prideľuje váhy slovám, podľa toho ako často sa používajú, čím zamedzíme zvýhodňovaniu dlhších článkov.

Finálny produkt poskytne používateľovi možnosť pridania nového textu do nami vytvorenej aplikácie, kde prebehne spracovanie tohto článku formou automatickej kategorizácie a otagovania. Ako sme uviedli už v predchádzajúcich kapitolách tohto dokumentu, náš nástroj bude pracovať s textami v slovenskom jazyku, ktoré v súčasnosti medzi dostupnými nástrojmi vysoko absentuje. Pri implementácií sa zameriavame najmä na využitie technológií NodeJS, Python, MongoDB.

Otázky na agilný vývoj

1. Ako predchádzať časovému sklzu a pracovať na úlohách priebežne? (Ako zlepšiť časovú organizáciu práce na úlohách?)

2. Ako veľmi je nutné dbať na správu plánovacieho SW (TFS, Jira)? Resp. pokiaľ vývoj ide dobre je problém, že sa nedbá na stav TFS?

3. Ako vhodne manažovať prácu, za ktorú som zodpovedný a svojich spolupracovníkov, ktorí so mnou pracujú na úlohe?

4. Aké metriky brať do úvahy pri určovaní odhadovaného času pre konkrétnu user-story / task? (Ako vhodne určiť odhadovaný čas?)

5. Ako riešiť to, keď niektorí členovia tímu nemajú také bohaté skúsenosti ako ostatní, ale chcú byť aj napriek tomu dôležitou súčasťou tímu?

6. Ako zjemniť prístup (princíp agile) v prípade, ak sa členovia tímu veľmi dobre poznajú?

7. Ako správne robiť retrospektívu a ako ju vyhodnocovať?

8. Aké sú efektívne postupy pri testovaní produktu, v rámci funkcionálneho testovania?

Dokumentácia k inžinierkemu dielu

Úvod

Cieľom tohto dokumentu je priblížiť inžinierske dielo, ktoré vytvárame v tíme skladajúcom sa z 8 členov v rámci predmetu Tímový projekt. Dokument približuje všeobecný opis a ciele projektu, časť projektu slúži ako technická dokumentácia.
Dokument je tematicky rozdelený do viacerých častí, čím je zabezpečené, že čitateľ postupným čítaním nadobudne prehľad o architektúre vytváraného systému a tiež o funkcionalitách, ktoré boli implementované a tvoria takzvané moduly systému. Celkový pohľad na systém je rozdelený do troch častí: architektúra systému, backendová a frontendová časť. Opis modulov systému je tiež doplnený o náhľady kódov a rôzne diagramy, čím sa zdokonaľuje predstavivosť čitateľa o vytvorenom systéme. V závere je uvedený tiež zoznam API spolu s popisom, URL, metódou a vstupnými a návratovými hodnotami.

1. Globálne ciele projektu

Cieľom tohto projektu je vytvoriť prostredie pre inteligentnú analýzu textov. Vyvíjané prostredie má byť v prvom rade prispôsobené na prácu s kolekciami textov (napr. články z rôznych webových stránok). Tieto texty musia byť efektívne uchovávané spolu s ich metaúdajmi. Metaúdaje sa budú získavať napríklad kategorizovaním vložených dokumentov, identifikáciou charakteristík vybranej skupiny textov, prípadne identifikáciou použitých slov. Všetky údaje je potrebné prehľadne vizualizovať, aby boli interpretovateľné aj menej zdatnými používateľmi.

Pre ukladanie dát použijeme databázu, konkrétne MongoDB, pričom články budú ukladané formou kolekcií. Po manuálnom otagovaní článkov z jedného zdroja bude táto kolekcia článkov tvoriť takzvaný korpus. Pod pojmom dokument budeme rozlišovať už spracovaný text, t. j. rozdelený na slová, lémy, určené gramatické katégorie (POS) a názvoslovná entita (NER).

V rámci nášho tímového projektu budeme vytvárať dve rozhrania. Rozhranie pre import článkov do databázy, ktoré bude obsahovať názov článku, text, tokenizáciu a tiež možnosť exportu tohto článku. Druhé rozhranie bude slúžiť na vizualizáciu dát z databázy, teda ide o vizualizáciu už konkrétnych článkov spolu s ich priemernou dĺžkou, počtom znakov, slov (rôzneho druhu), ... Toto rozhranie bude tiež podporovať funkcionalitu štatistiky nad článkami, ktoré obsahujú zvolené tagy, vytvorenie histogramu (slov, lém, slovných druhov, názvoslovných entít, tagov), vytvorenie takzvaného tag – cloud(u). Pri spracovaní jednotlivých článkov budeme využívať metódu TFIDF, ktorá pridelľuje váhy slovám, podľa toho ako často sa používajú, čím zamedzíme zvýhodňovaniu dlhších článkov.

Finálny produkt poskytne používateľovi možnosť pridania nového textu do nami vytvorenej aplikácie, kde prebehne spracovanie tohto článku formou automatickej kategorizácie a otagovania. Ako sme uviedli už v predchádzajúcich kapitolách tohto dokumentu, náš nástroj bude pracovať s textami v slovenskom jazyku, ktoré v súčasnosti medzi dostupnými nástrojmi vysoko absentuje. Pri implementácií sme sa zamerali najmä na využitie technológií NodeJS, Python, MongoDB.

1. Ciele pre zimný semester

Počas zimného semestra sa zameriavame na úvodnú analýzu dostupných nástrojov a možností, ktoré máme pri implementácií systému. Dôležitou súčasťou je implementačná časť, v ktorej je cieľom vytvoriť použiteľné a funkčné rozhranie, ktoré bude dostupné pre používateľov. Pričom sa snažíme riadiť spôsobom agilného vývoja.

Pri návrhu a tvorbe rozhrania je dôležité klásť dôraz najmä na použiteľnosť, aby bolo vhodné pre používateľov s rôznou úrovňou schopností. Cieľom je teda vytvoriť intuitívne použiteľné rozhranie. Rozhranie bude zobrazovať jednotlivé články zaradené do korpusov, bude poskytovať možnosť rôznych filtrácií podľa zvolených parametrov. Cieľom je tiež implementácia

modulov predstavujúcich hlavnú funkcionalitu (import článkov, tokenizácia, invertovaný index, tf-idf).

V neposlednom rade je cieľom zabezpečiť oddelenie sekcie pre používateľov od sekcie pre administráciu, čiže vytvoriť prostredie určené pre vývoj systému, čím budú tiež obmedzené verejné funkcionality. Týmto krokom tiež zvýšime bezpečnosť systému, nakoľko používateľ bude mať obmedzený prístup k funkcionalitám, ktoré pre neho nie sú určené a potrebné.

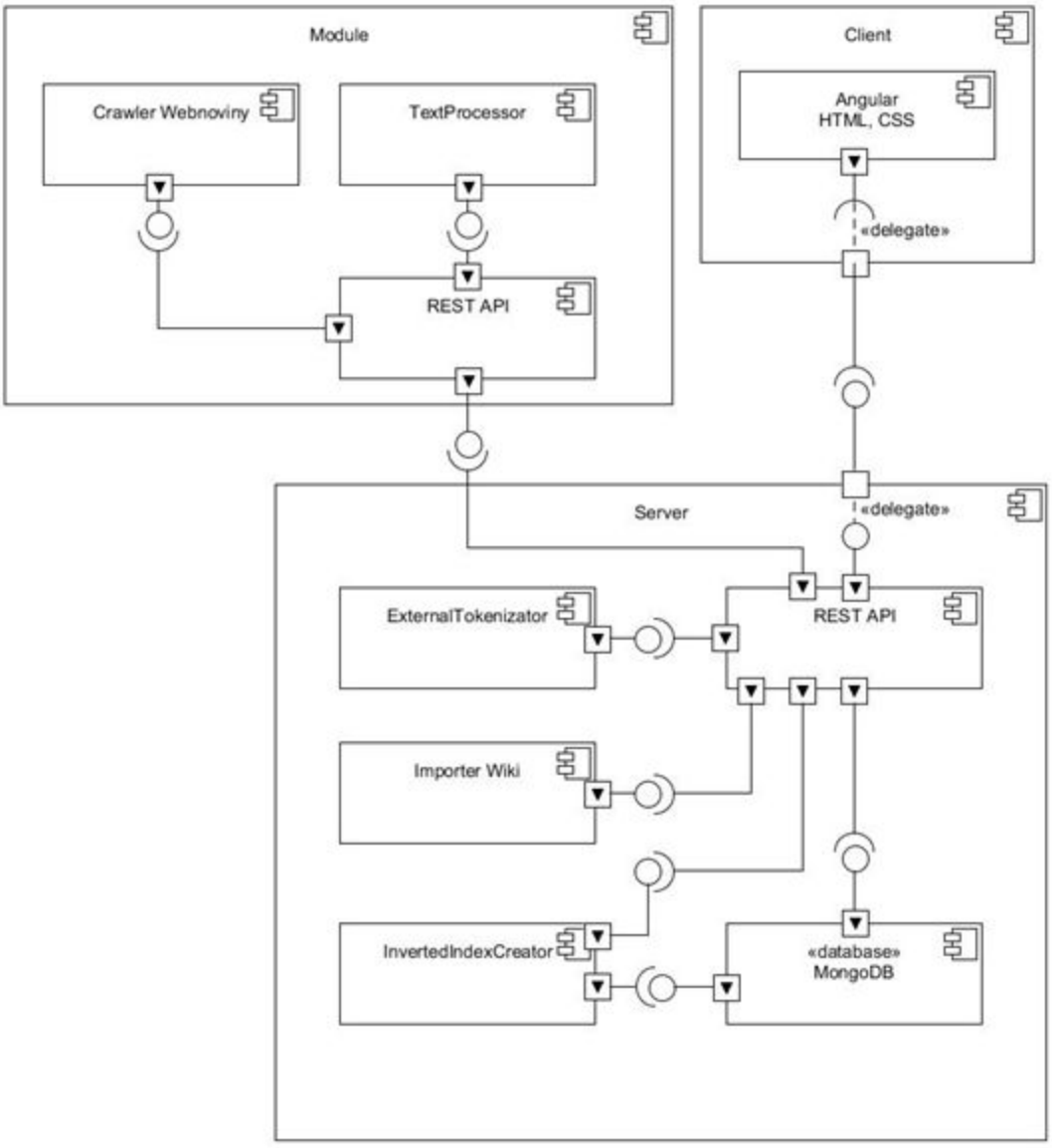
2. Celkový pohľad na systém

Vytváraný systém rozdeľujeme na backedovú a frontendovú časť.

2.1. Architektúra systému

Architektúra systému bude pozostávať z dvoch hlavných častí: klient a server. Z toho vyplýva, že v architektúre bude použitý architektonický štýl klient-server. Klientska časť je poskytovaná webovým klientom v prezenčnej forme. Tento komponent môžeme jednoducho zameniť, slúži len na zobrazovanie a získavanie dát zo servera cez REST API pre prezentačnú vrstvu webového klienta.
Server disponuje službami REST API v rámci ktorých pristupuje pomocou dátového mapovania a schém k databáze alebo k ďalším samostatným komponentom, ktorými vykonávame nad dátami čiastočnú aplikačnú logiku. Hlavná časť logiky na spracovanie textu bude implementovaná v prídavnom module. Tento modul bude samostatný článok systému oddeliteľný od serverového stroja vlastnými REST API. Tieto REST API budú vyvolané nejakým plánovačom alebo presmerovaním zo serverovej API. Do budúcna predpokladáme možnosť rozdelenia výpočtovej sily (na viac serverov), vzhľadom k náročnosti operácií nad textom. V prípade potreby môžeme operácie nad textom vykonávať na výkonnejšom stroji.

Obrázok X popisuje architektúru nášho systému. Všetky požiadavky z Angular klientskej časti smerujú na REST API servera. Vďaka integrácií Angularu do node.js sa nám podarilo vyriešiť problém klienta s prístupom cez sieťové porty k API alebo samotnej Angular aplikácií. Angular dopytuje a vykonáva operácie nad API CRUD metódami a odpoveď získava v formáte JSON. Všetky implementované komponenty ako *Importer, Crawler, Tokenizator* sú opísane v ďalších častiach tohto dokumentu.



Obr X. Architektúra softvéru

Vytvárame webovú aplikáciu na prácu s textom, čiže našou doménou budú *bigdata*. Preto sme sa rozhodli použiť databázu mongo, ktorá dokáže spracovať veľa požiadaviek nad textom v relatívne krátkom čase. Knižnica mongoose ponúka ODM modelovanie a mapovanie objektov, ktoré sa používa na ukladanie korpusov a komplexných článkov z 2.2 Architektúra databázy. Ako bolo spomenuté, k databáze pristupujeme cez DAO v REST API. REST API implementujeme webovým rámcom Express.js, ktorý beží na JavaScript run-time platforme node.js serveru. Server bude zároveň prostredník, ktorý volá ďalšie externé služby.

Používané externé služby na spracovanie textu z nášho modulu *processer* alebo *arl6* knižnice budú volané cez REST API. V pythone bude bežať nás samostatný modul, v ktorom budú skripty na

spracovanie textu a strojové učenie. Python sme zvolili vzhľadom k veľkému množstvu knižníc v oblasti a predošlým skúsenostiam tímu.
Klientsku časť front-endu zastrešuje rámec Angular. Hlavným dôvodom tejto voľby bolo zníženie požiadaviek na server, ale aj zrýchlenie zobrazovania prezenčnej vrstvy používateľovi, bez načítavania celej stránky.

2.2. Architektúra databázy

Architektúra NoSQL databázy a organizácia dokumentov do kolekcií je pre nás nesmierne dôležitá. Navrhli a použili sme dve kolekcie, konkrétne korpus a články. Tieto kolekcie majú medzi sebou referenčný vzťah.

Korpus

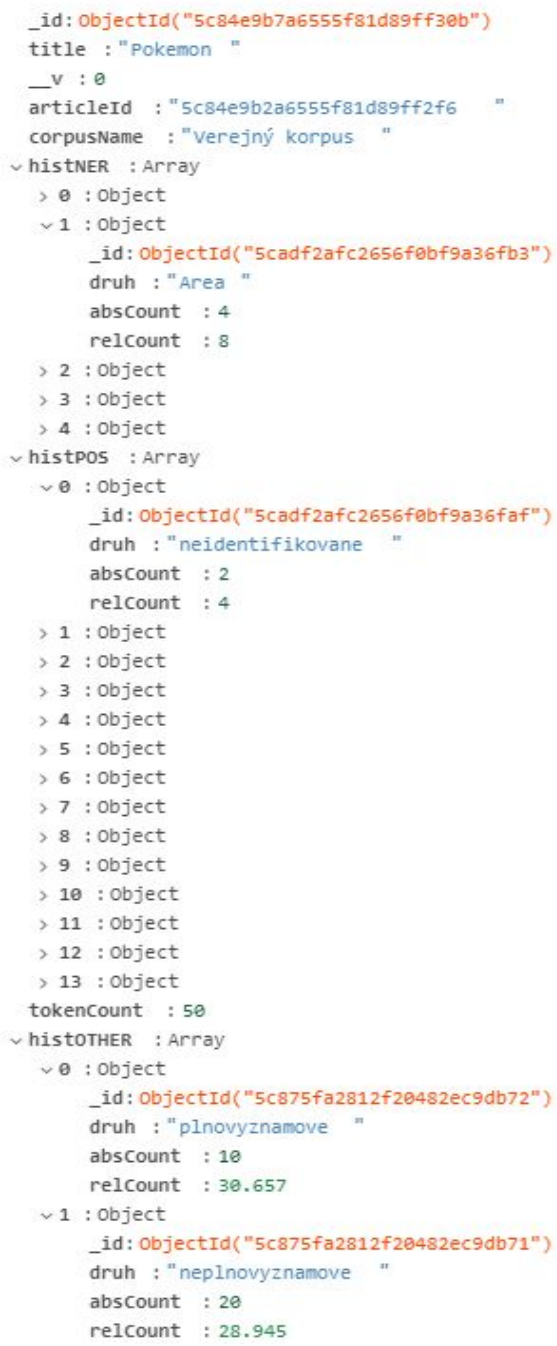
Kolekcia korpus slúži na kategorizovanie článkov podľa obsahu. Obsahuje názov korpusu, popis čoho sa korpus týka a pole kľúčových slov, ktoré sa najviac vyskytujú v článkoch patriacich do daného korpusu. Náhľad existujúceho korpusu zobrazuje nasledujúci obrázok.

Články

Kolekcia články obsahuje všetky články vložené do databázy pod špecifickým korpusom. Táto kolekcia obsahuje okrem názvu, nespracovaného textu, orientačných dátumov, značiek a zdroja aj rozsiahle informácie o článku. Tieto informácie nazývame tokeny, patrí do nich štatistika článku (počet znakov, slov, plnovýznamových slov), a spracovaný text. Spracovaný text je vnorené pole slov článku a obsahuje pôvodné slovo, jeho základný tvar, POS a NER formáty slova. Prepojenie článku na korpusy zabezpečujeme kľúčom na názov korpusu. Náhľad existujúceho článku zobrazuje nasledujúci obrázok.

Odlačok článku

Kolekcia odtlačok článku obsahuje dáta ako histogram POS, NER formátov nad všetkými článkami. Atributy tohto dokumentu sú title (názov článku), corpusName (názov korpusu), histNER, (histogram všetkých NER druhov v článku), histPOS, (histogram všetkých POS druhov, plnovýznamových, neplnovýznamových, neidentifikovaných v článku), tokensCount (počet tokenov v článku),



Invertovaný Index

Kolekciu s invertovaným indexom používame na zoskupenie a vytvorenie skupín pre slová používané vo všetkých článkoch. Táto kolekcia obsahuje slovo a názov článku (identifikátor), názov korpusu, ktoré sú kľúčové pre invertovaný index. Ďalej obsahuje lemu slova, počet slova v článku, frekvenciu používania výrazu a jeho NER, či POS značky. Do invertovaného indexu sa vypočítavajú ďalšie hodnotné atribúty ako tf, idfCorpus, tfidfCorpus, tfidfArticle, ktoré slúžia na určenie frekvencií používania týchto slov v článku alebo korpuse. Pre jednoduchšie a rýchlejšie

dopytovania poradia sa aktualizujú záznamy o atribút articleRelevance/corpusRelevance. Na nasledujúcom obrázku je zachytený príklad dokument invertovaného indexu na produkčnej DB.

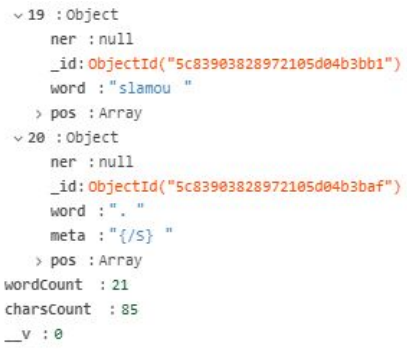

```
_id: ObjectId("5c882021cb74e620a44ac31b")
word : "ak "
lema : "ak "
pos : "O "
ner : null
tf : 0.0031446540880503146
idfCorpus : 1.1187946084179672
corpusName : "Politika "
title : "Ak by uznesenie v prípade Poľska a Maďarska bolo o dialógu, Pellegrini by ho nevnímal ako podporu.
articleId : "5c6c5c681d380327ac9138d0 "
count : 2
articleRelevance : 122
corpusRelevance : 111
collectionRelevance : null
__v : 0
tfidfArticle : 3.5182220390502117
tfCorpus : 0.0014037813319277026
tfidfCorpus : 1.5705429855585065
tfidfCollection : 0.802145907627457
tfCollection : 0.0003901600264906482
```


Ngramy (5)

Kolekcia sa volá ngram obsahuje dokumenty n-tic vytvorených z textu článkov. Identifikátory dokumentu sú ID článku, názov článku, názov korpusu. Každý dokument ma pole 5-tic slov s okolím slova 2 zľava a 2 zprava. Analogicky to platí aj pre 5-ticu POS formy.

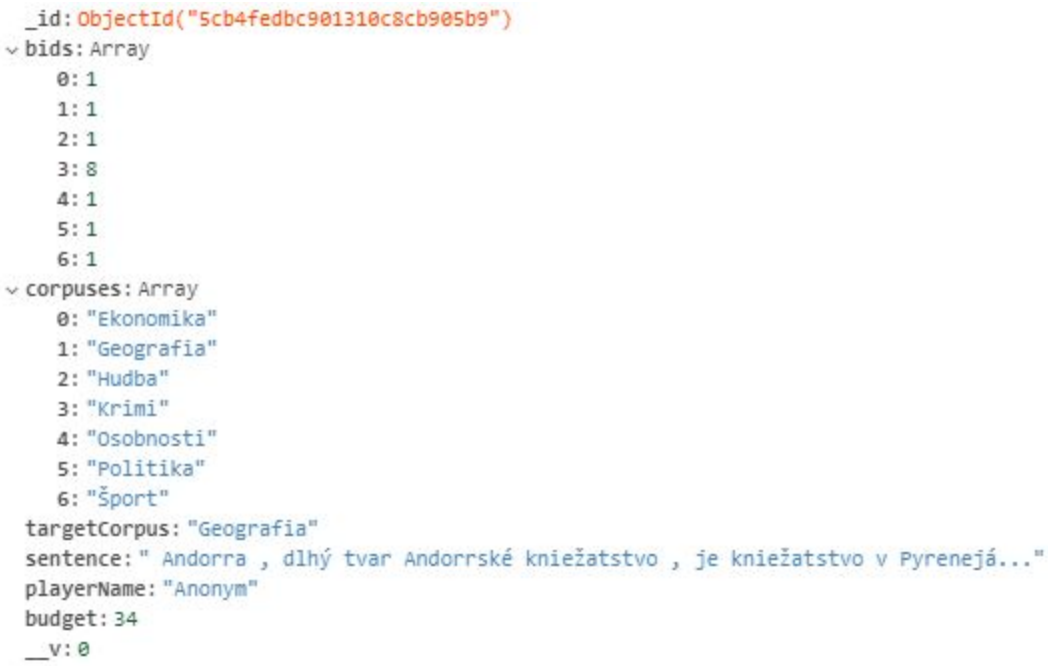
Vety

Kolekcia sa volá sentence obsahuje dokumenty viet. Vytvára sa z textu a tokenov článkov. Obsahuje samotný text vety, POS textový reťazec podľa postupnosti slov z vety. Tak isto obsahuje aj základné identifkátory ako ID článku, názov korpusu, článku. Obsahuje pole všetkých tokenov pre vetu, počet tokenov/slov, počet znakov vety.



Výsledky hry

Kolekcia sa volá gameResults. Do kolekcie sa ukladajú záznamy z odohranej hry neprihláseným užívateľom (reprezentovaným ako playerName a jeho budget peniazmi na stávkovanie). V hre vystupuje veta ako prvok, ktorý sa snaží používateľ zaradiť do najbližšieho korpusu z ponuky. Stávky a hodnota sa mapuju na názov korpusu medzi atributmi bids a corpuses, ktoré sú typu poľa.

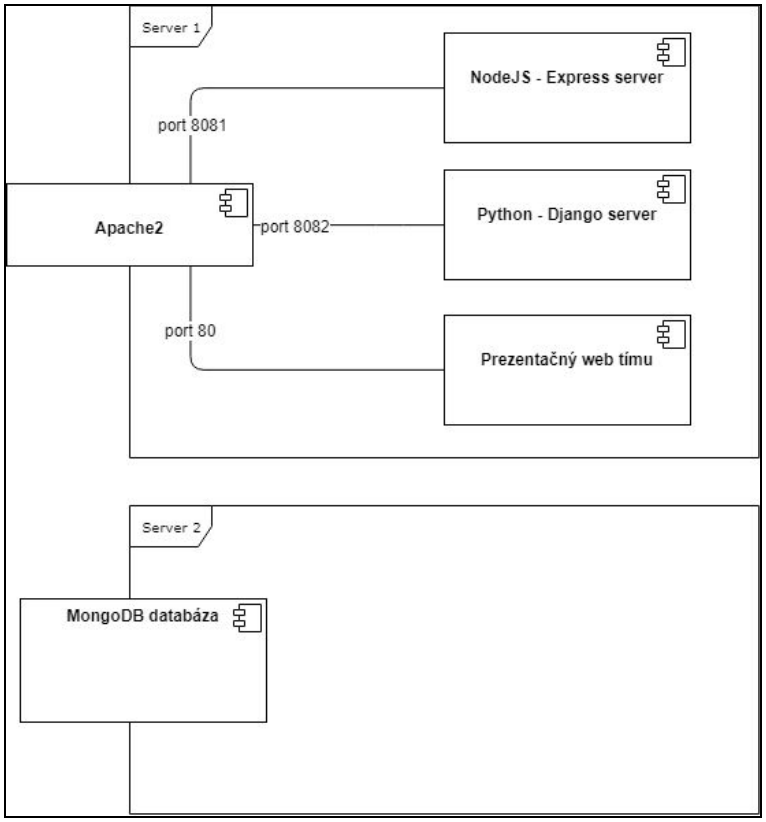


Používatelia

Kolekcia používateľov bola zavedená vzhľadom na zabezpečenie volaní REST API. Pri autentifikácii používateľa sa v aplikácií kontrolujú vstupné prihlasovacie údaje. Preto v dokumente ako identifikátor používateľa používame email a zahashované heslo. Po správnom autentifikovaní používateľovi je sprístupnený token na prístup k uzamknutým API službám. Tento token je logicky generovaný v časových úsekoch podľa používania. Náhľad dokumentu s používateľom je zobrazený na obrázku.

```
_id: ObjectId("5bf04e5327500d6b9a3521b1")
email: "tester@test.sk"
hashedPassword: "ff8ce2ea1d618f76c6ad4b2d6cfed1a61ed4898d"
createdAt: 2018-11-17 18:22:27.863
accessToken: "0170b47ad999291b11ded432b933b8efad75f02ee50beb9e2ef59a41eede7d2174a1f2..."
accessTokenCreatedAt: 2018-11-20 12:22:34.803
```

2.3. Backend systém



Jednotlivé súčasti aplikácie sú nasadené na dvoch serveroch. Hlavná serverová časť je nasadená na školskom serveri s IP adresou 147.175.149.189. Na tomto serveri je nasadený aj prezentačný web tímu, ktorý je dostupný na štandardnom HTTP porte 80.

Na porte 8081 je dostupný samotný projekt, resp. jednak frontend rozhranie dostupné pre klientov (root portu 8081), ale aj API rozhranie dostupné na podadresách začínajúcich prefixom /api. Celý port smeruje na NodeJS server implementovaný prostredníctvom rámca Express. Tento rámec zgrupuje jednotlivé funkcionality tohto projektu, tj. zabezpečuje verejné API ako aj distribúciu klientskej časti implementovanej v rámci Angular ku klientom. API je zdokumentovaná nižšie v tomto dokumente.

Na porte 8082 je dostupná Python časť projektu implementovaná prostredníctvom rámca Django, ktorá tiež poskytuje funkcionalitu vo forme API. V budúcnosti nebude táto časť dostupná verejne, zatiaľ je dostupná, z dôvodu prebiehajúceho vývoja.

Databázová časť je nasadená na inom serveri s IP adresou 165.227.163.222 pod štandardným MongoDB portom číslo 27017. V budúcnosti bude nasadená táto časť na rovnakom serveri ako zvyšok a nebude k dispozícii verejne.

Prvá technológia, ktorá bol zvolená pre tento projekt bola práve databáza MongoDB, keďže je vhodná na prácu s textom. NodeJS, resp. rámec Express bol zvolený z dôvodu jeho peknej súdržnosti s MongoDB a aj z dôvodu poznania tohto rámca viacerými členmi tímu. Python na implementáciu určitých častí projektu bol zvolený z dôvodu jednoduchšej implementácie strojového učenia v budúcnosti a taktiež z dôvodu, že bol známy viacerým členom tímu. Rámec Django pre Python časť bol zvolený z dôvodu, že sa jedná o asi najznámejší a najlepší rámec umožňujúci tvorbu API v Pythone.

2.4. Frontend systému

Aplikácia z väčšej časti využíva webové služby, na základe tohto faktu sme sa rozhodli umiestniť aplikáciu online. Čo zabezpečuje aj jej dostupnosť pre širšiu verejnosť. Pri výbere nástrojov pre zobrazovanie dát sme sa rozhodovali medzi niekoľkými variantami, z ktorých niektoré využívali server-side rendering, nakoniec sme sa však rozhodli použiť klient-side rendering, keďže product owner vyžadoval filtrovanie dát v reálnom čase.

Rámec Angular

Používateľské rozhranie je realizované pomocou rámca Angular. Tento rámec bol použitý z dôvodu pokrytia požiadaviek product ownera a tiež z dôvodu, že aplikácia využíva veľa rozhraní, ktoré poskytujú filtrovanie dát v reálnom čase.
Základným stavebným elementom sú komponenty, z ktorých je vystavaná celá aplikácia na klientskej strane. Samotné komponenty sú samostatne stojace elementy stránky, ktoré sú vyvolávané na základe URL adresy. Tieto komponenty medzi sebou komunikujú pomocou angular služieb (angular services), cez ktoré si vymieňajú dáta.

Služby

Služby sú tiež použité na komunikáciu s backendovou časťou, ktorá je implementovaná cez NodeJS a poskytuje Rest API pre frontendovú časť aplikácie. Služby komunikujúce s Rest API možno rozdeliť do dvoch skupín:

- služby na získanie dát (korpusy, články, inv. indexy)
- služby na posielanie dát (tokenizácia, indexácia, prihlásenie)

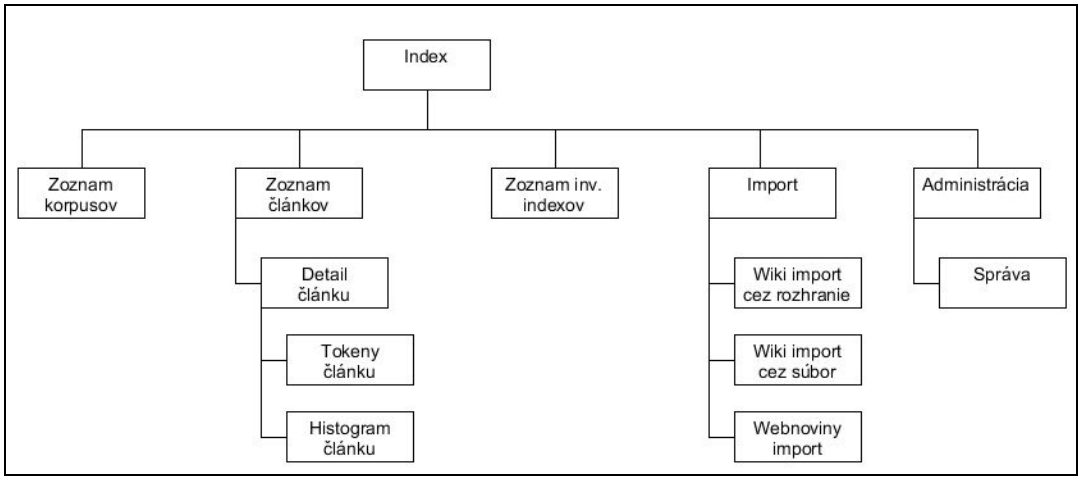
Výstupom z volaných služieb sú JSON objekty, ktoré obsahujú dáta, ktoré rámec Angular ďalej spracuje a robí nad nimi rôzne operácie. Primárnou funkciou Angularu je však dynamické a efektívne zobrazovanie dát.

Niektoré služby volajú funkcionalitu, ktorá by nemala byť voľne dostupná pre používateľov aplikácie a preto je jej použitie obmedzené. Pre použitie týchto aplikácií je potrebné mať prístupový token, ktorý sa posiela na backend pre overenie prístupových práv k CRUD funkcionalite. Token je možné získať prihlásením sa do používateľského konta pomocou emailu a hesla, ktoré sa posiela na overenie na backend, v prípade, že údaje sú správne vracia sa na frontend email a vygenerovaný token, na základe ktorého je následne možné používať služby vyžadujúce verifikovanie.

Poskytnuté rozhrania

Aktuálne rozhranie disponuje obrazovkami pre nižšie uvedené časti, jednotlivé prepojenia týchto rozhraní popisuje diagram nižšie:

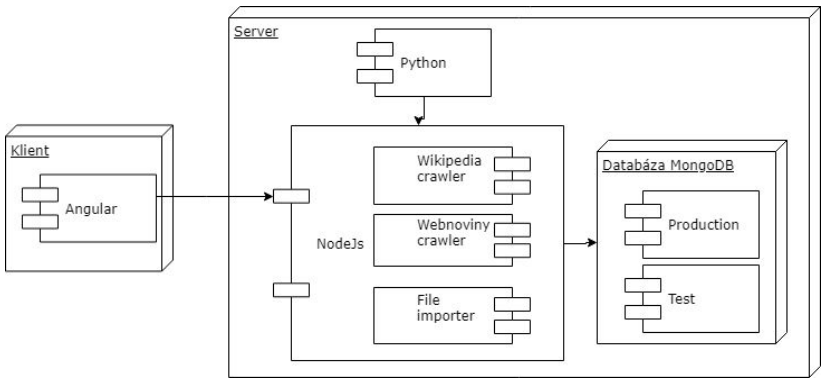
- zoznam korpusov a ich detail, cez ktoré je možné dostať sa ku konkrétnym článkom v korpuse,
- zoznam invertovaných indexov s detailom o každom inv. indexe,
- zoznam článkov s preklikom na detail článku
- detail článku so záložkami: index, tokeny, histogram
- rozhranie pre import článkov do databázy cez scrapper (tu je možné zvoliť viacero možností importu: import z wikipedie priamo cez rozhranie, import z wikipedie cez súbor, import z webnovín cez rozhranie)
- admin rozhranie poskytujúce funkcionalitu prístupnú len oprávneným používateľom



3. Moduly systému

Celý proces implementácie je realizovaný v šprintoch, ktorých výsledkom sú fungujúce celky. Tieto celky obsahujú súbor funkcionalít, ktoré navyšujú hodnotu systému pre product ownera. Tieto celky reprezentujú nami nižšie opísané moduly, ktoré sú zdokumentované čiastkovým diagramom pre lepšie pochopenie fungovania danej časti, stručným opisom.

3.1. Modul importu dát



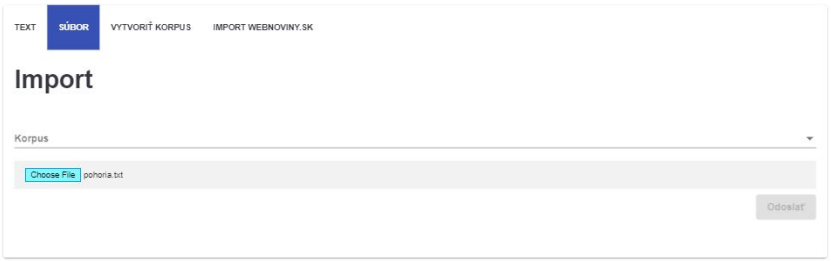
Ako jeden z primárnych modulov, ktoré boli v našom projekte implementované je modul importu dát. Nakoľko dáta s ktorými narábame sú textového charakteru a pre následnú prácu so systémom bolo kritické zabezpečiť dostatok týchto dát. Preto sme sa zamerali na extrakciu slovenských textov z dostatočne veľkých domén, ktoré uchovávajú rozsiahle lexikálne zdroje. Intuitívny výber tvoril portál slovenskej wikipedie a pre všeobecnejší set dát sme zvolili aj portál webnoviny.sk.

Na vyššie spomenuté zdroje boli navrhnuté a implementované tri nezávislé extraktory, ktorých výsledkom je získanie daného článku a uloženie obsahu do našej databázy. Pri preberaní textov sme sa snažili zachovať čo najvernejšie formát a štýl písaného textu pre uchovanie členenia a logického významu. Texty po prevzatí taktiež prešli korekciou a úpravou, kde boli odstránené pre nás nevýznamné vety, sekcie či dokonca celé články(napríklad rozlišovacie stránky z wikipedia.sk). Vyššie spomenuté tri nezávislé extraktory zahŕňajú:

- Možnosť importovať konkrétny článok z wikipedie

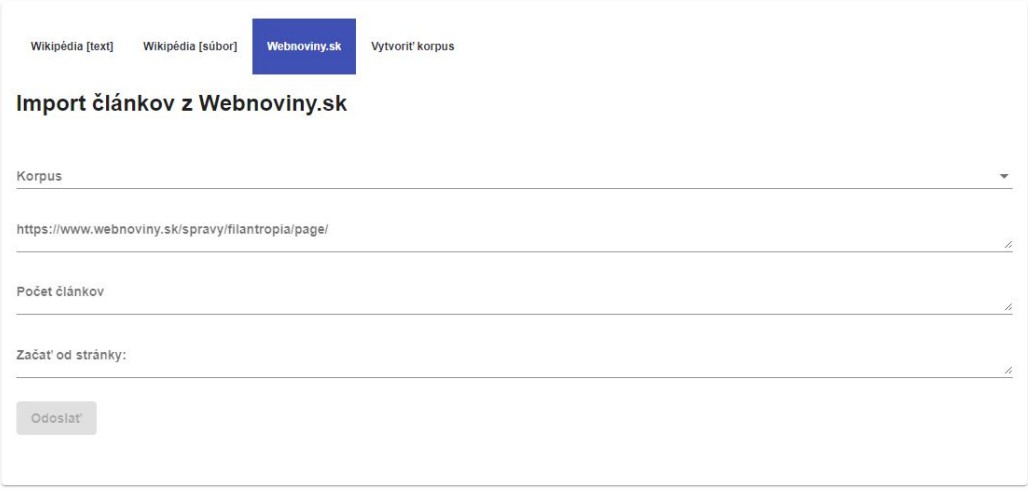
Táto časť modulu je implementovaná priamo v backendovej časti systému(nodejs). Výsledkom vykonaného procesu je uložený článok v databáze v danom korpuse článkov.

- Možnosť importovať viacero článkov z wikipedie z titulov spísaných v .txt



Rozširujúca funkcionalita pre vyššie spomenutú časť. Funkcionálne rovnako založený proces, ktorý je schopný spracovať dávku titulov načítanú z externého súboru. Výsledkom sú dané články uložené v databáze v danom korpuse.

- Možnosť importovať stanovený počet článkov z určitej sekcie webnoviny.sk



Posledná časť reprezentuje crawler implementovaný v jazyku Python. Volanie tejto funkcionality je zabezpečené cez REST Django, ktoré zabezpečuje komunikáciu medzi Node.js a Python. Nakoľko sa jedná o slovenský portál, ktorý má zabezpečenia proti mnohonásobným požiadavkam, crawler je implementovaný čo najmenej ofenzívne. Preto aj výsledkom nemusí byť požadovaný počet článkov nakoľko sa proces ukončí v momente, keď portál webnoviny.sk trikrát po sebe zablokuje našu žiadosť a uloží všetky dovtedy získané články do databázy.

3.1.1. Proces vkladania článkov zo zdroja Wikipedia.org

Samotná logika nahrávania článkov z Wikipédia je centralizovaná v triede WikiExtractor. Aby sme za reálnych okolností tuto funkcionalitu mohli momentálne volať musíme využiť nasledujúci postup. Potrebné je využitie nášho webového rozhrania, ako môžeme vidieť v texte odstavci 3.1, pri ktorom je nutné sa prihlásiť do admin zony rozhrania a zvoliť tab s názvom *Import*. V danej sekcii *Import* už podla svojich potrieb zvolíme možnosť importu dát. Rozhranie pomocou POST volaní služieb kontaktuje server pomocou cesty "*<názov-stránky>:<port>/api/articles/import*", o tejto API je viac napísané tiež v sekcii 4 s podnadpisom *Import článkov z Wikipédie*. Daný POST request obsahuje vo svojom tele okrem iných dáta *titles* a *corpusName*. Tieto dáta predstavujú názvy Wikipédia stránok, jedná sa o *titles* a názov korpusu, ktorého ideme vkladať, čo je asi zrejmé, že ide o *corpusName*. Keďže sa jedná o POST volanie služby, je jasné, že teoreticky by zručným používateľom stačilo poznať spomenuté atribúty tela a využiť aplikáciu ako Postman a výsledky replikovať, nie je tomu však tak. Z bezpečnostných dôvodov frontend obaľuje telo requestu ešte ďalšími autorizačnými informáciami, ktoré sú overované na serverovej strane, preto takéto priamočiare volanie requestov nie je vykonateľné bez úprav. V našom projekte sú na spracovavanie volaní a volaných ciest vyhradené špeciálne triedy obsahujúce vo svojom názve koncovku "...router.js". V tomto konkrétnom prípade sa bavíme o triede *articleRouter*. Pri volaní requestu práve ona vyvoláva funkcionalitu cielovej triedy WikiExtractor *getWikiArticles()*, ktorej poskytne ako parametre spomínané *titles* a *corpusName*.

Funkcia triedy Wikiextractor, getWikiArticles(), následne prechádza v cykle všetky poskytnuté názvy Wiki stránok a pomocou api volaní na URL: "https://sk.wikipedia.org/w/api.php?format=json&action=query&prop=extracts|pageprops&explaintext=1&titles=<Názov Wiki stránky>" Získava obsah daných stránok v Json formáte. Na vykonanie samotných volaní api je použitá knižnica Axios. Navrátený Json objekt obsahuje mnohé pre nás redundantné informácie, preto sú v prvom rade potrebné dáta (text samotnej Wiki stránky) pomocou funkcie *getObjects()* vyextrahované a navrátené. Po získaní údajov čisto stránky skontrolujeme, či sa nejedná len o rozhodovaciu (ang. disambiguation) stránku a taktiež či nejde o príliš krátku stránku (text s menej ako 200 slovami). Ukážku prijatého a očisteného Json objektu môžeme vidieť na obrázku pod týmto odstavcom. Ak boli tieto vlastnosti stránky v poriadku tak vytvoríme nový Json objekt dodržujúci schému totožnú s našou databázovou schémou a naplníme ho údajmi o novom článku. Tieto objekty už do veľkej miery identické môžete vidieť v databáze

pod kolekciou *article*, jediný rozdiel je v stave samotného textu stránok, ktorý ešte podíde očisteniu a oprave.


```
{
        "batchcomplete": "",
        "warnings": {
            "extracts": {
                "*": "\"exlimit\" was too large for a whole article
extracts request, lowered to 1."
            }
        },
        "query": {
            "pages": {
                "278495": {
                    "pageid": 278495,
                    "ns": 0,
                    "title": "1 cent",
                    "extract": "1 cent môže byť:\n\nminca:\n1 cent
(americký dolár)\n1 cent (austrálsky dolár)\n1 cent (euro)\n1 cent
(kanadský dolár)\n\n\n== Pozri aj ==\ncent, hmotnostná, fyzikálna alebo
hudobná jednotka\ncent (menová jednotka), čiastková menová jednotka",
                    "pageprops": {
                        "disambiguation": "",
                        "wikibase_item": "Q161594"
                    }
                }
            }
        }
},
```


Ukážka prijatého JSON objektu článku. Jedná sa zrovna o disambiguation stránku, ktorú vyfiltrujeme.

Keď máme prejdené všetky názvy stránok, získané všetky stránky, vyfiltrované nevhodné a uložené vyhovujúce nastane už spomenuté prečistenie textu stránok. Vhodné stránky sme uložili do poľa, ktoré preiterujeme a pre každú stránku zavolame funkciu *cleanText()*, ktorej parametrom je text samotnej stránky. V krátkosti v tejto funkcii pomocou regulárnych výrazov odstraňujeme nepotrebné časti textu, ktoré vieme, že sa budú nachádzať pod určitými podnadpismi, odstraňujeme textové ikony, ktoré boli zdeformované, nastavujeme správne odriadkovanie a vo všeobecnosti formátujeme text tak, aby vyzeral dobre na frontende a v neposlednom rade ho upravujeme tak aby bol čitateľný nástrojom arl6, napríklad pridávame bodky za nadpisy.
Pre jednoduchú replikáciu a zkonkrétnenie riešenia prikladáme konkrétne regexy:

- text.replace(/(\[\d+\])/gm, '')
- text.replace(/== Referencie ==.*/gms, '')
- text.replace(/== Iné projekty ==.*/gms, '')
- text.replace(/== Externé odkazy ==.*/gms, '')
- text.replace(/== Literatúra ==.*/gms, '')
- text.replace(/== Pozri aj ==.*/gms, '')
- text.replace(/== (.*) ==/gm, '== $1. ==')
- text.replace(/( )/gm, '')

- text.replace(/[\n\r]+/gms, '\n')
- text.replace(/([a-zA-Z0-9])[.]?\s?[\n]/gms, '$1.\n')
- while (text.slice(-1) === '\n' || text.slice(-1) === ' ')
  { text = text.slice(0, -1) }
- if (/[a-zA-Z0-9]/.test(text.slice(-1))) { text += '.' }
- text.replace(/={2,}/gm, ' ==== ')

Stránky s vyčistenými textami su navrátené ako výstup funkcie *getWikiArticles()* do triedy articleRouter a hneď uložené pomocou metody *save()* triedy *articleDao*, ktorá má ako vstupné parametre spomínane pole Json objektov predstavujúcich články.

3.1.2. Proces vkladania článkov zo zdroja webnoviny.sk

Proces dolovania článkov z webnovín je prevádzaný samostatným modulom implementovaným v jazyku python. Knižnice, ktoré sú nevyhnutné pre fungovanie tohto modulu sú *request, Beautifulsoup, urlopen, json.* Proces dolovania začína vyvolaním príslušnej akcie cez API *<názov-stránky>/api/articles/importWebNoviny* ktorá následne šíri požiadavku cez api *<django-rest>*/import-web-news pre django rámec zodpovedného za python súčasti a spustí proces dolovania článkov.

Proces dolovania je v Pyton častiach implementovaný v triede WebNewsCrawler. Ako parametre pre tento proces sú korpus do ktorého sa dané články uložia, url na ktorej sa nachádzajú požadované články (napr. https://www.webnoviny.sk/kl/ekonomika/slovensko/) a začiatočná stránka ktorá reprezentuje na ktorej "záložke" má metóda začať a počet článkov ktoré chceme dolovať. Celkovo sa týmto nastavením dá dosiahnuť akákoľvek variabilita dolovania článkov z portálu webnoviny.sk. Samotný proces spracovania spočíval v nasledujúcich krokoch:

- Získanie zoznamu url zo zoznamu článkov
- Otvorenie konkrétneho článku
- V prípade novej url sa daný článok serializuje do databázy pre budúce použitie
- Parsovanie obsahu daného článku konkrétne:
  - Názov článku
  - Z jednotlivých paragrafov a headrov obsah článku
  - Vynechané sú videá, fotky a iné elementy s nasledujúcimi tagmi:

```
if "Aktualizované:" in title:
    title = title.replace("Aktualizované: ", "")
if "Video" in title:
    title = title.replace("Video: ", "")
if "Foto" in title:
    title = title.replace("Foto: ", "")
```

Články sú ukladané do pola objektov typu Product ktorý je po ukončení procesu dolovania (splnení počtu požadovaných článkov) serializovaný do .json formátu, ktorý je vrátený do node.js modulu. Následne je .json postupne deserializovaný do typu Article a uložený do databázy. Taktiež je samotný obsah(odstavce, kapitly) označený pre lepšie spracovanie na rozhraní.

```
for element in headersAndParagraphs:
    if "<h2>" in str(element):
        articleText = articleText + " ====  " + element.text + ".  ==== \n"
```

3.2. Modul tokenizátora

Vyvolanie tokenizátora je zapríčinené žiadosťou otokenizovať text článku pre články z korpusu aj samotný článok. Použitím tokenizátora sa vytvorí nová inštancia tohto modulu.

Vstupom pre tento modul je ucelený text článku. Tento text článku sa spracuje po sekciách a nad jednotlivými sekciami textu sa volá API nlp6 tokenizátor. Parametre na používanie tokenizátora sú pevne definované v konfigu. Konkrétne sa jedná o prístupový API kľúč, modul POS značkovania (ProbabilisticPOSTagger) a NER značkovače (DictionaryNERTagger, RuleNERTagger), ktoré používame.

Metódou *callExternal()* vykonáme dávku sekcií nad, ktorou voláme externé služby. Návratovou hodnotou týchto služieb je pole otokenizovaných slov z textu. V priebehu pomocou *reduce*, redukujeme a zoradíme výsledky do nášho jednotného formátu. Celkovým výstupom tejto metódy je článok, ktorý obsahuje otokenizované všetky slová v texte. Návratový objekt článku je spätne aktualizovaný v databáze.

3.3. Modul invertovaného indexu

Texty z Wikipédie a aj portálu Webnovín sú po uložení do databázy a otokenizovaní pripravené na ďalšie spracovanie. Ďalším krokom v ich analýze je vytvorenie kolekcie tzv. Invertovaného indexu. Invertovaný index predstavuje indexovú dátovú štruktúru uchovávajúcu mapovanie zo zdroja, ktorý je v našom prípade slovo z článku, či skupiny článkov z databázy, vhodným spôsobom späť do databázy. Účelom invertovaného indexu je umožnenie rýchlejšieho vyhľadávania vo full-textových dátach za trade off spojený s náročnejším spracovávaním pri ukladaní článku a taktiež priestorovú náročnosť. Invertovaný index je vo svojej kolekcii samotným prvkom v databáze. Je to najobľúbenejšia dátová štruktúra používaná v systémoch na vyhľadávanie dokumentov, ktoré sa používajú vo veľkom meradle napríklad vo vyhľadávačoch.

3.3.1. Proces tvorby invertovaného indexu

Ako sme už spomenuly v odseku vyššie pre tvorbu invertovaného indexu potrebujeme mať naseedovanú kolkeciu article (článkov) a taktiež musia byť dané články už otokenizované (anotované / preposlané nástroju arl6). Funkcionalita tvorby invertovaného indexu je centralizovaná v triede *CreateInvertedIndex*. Ak chceme komunikovať so spomínanou triedou a vyvolávať jej funkcionalitu môžeme využiť API volania s URL "*<názov-stránky>:<port>/api/inverted-indices/corpus/<názov korpusu>"*. Viac o tejto službe nájdete v sekcii 4 s podnadpisom *Vytvorenie invertovaných indexov pre konkrétny korpus*. Volanie je možné vykonať z nášho webového rozhrania z admin sekcie, ale keďže nie je obalené autorizačným tokenom je možné funkcionalitu vykonávať aj pomocou obyčajných POST volaní. V našom projekte sú na spracovavanie volaní a volaných ciest vyhradené špeciálne triedy obsahujúce vo svojom názve koncovku "...router.js". V tomto konkrétnom prípade sa bavíme o triede *invertedIndexRouter*. Pri volaní requestu práve ona vyvoláva funkcionalitu cielovej triedy *CreateInvertedIndex createInvertedIndexOfArticle()*, ktorej poskytne ako parameter článok, ktorý cheme oindexovať. *invertedIndexRouter* však vykonáva niekoľké predpoklady a dôsledky pred a po volaní tvorby invertovaného indexu.

Keďže indexujeme celý korpus a ako ste si mohli všimnúť do funkcie *createInvertedIndexOfArticle()* vstupuje len jeden parameter a to jeden článok, ktorý chceme oindexovať tak práve toto predspracovanie vykonáva *invertedIndexRouter*. Routrovacia trieda použije dopyt nad databázou pomocou volania *articleDao.getByCorpus()*, kde je vstupným parametrom názov cieľového korpusu. Tento dopyt navráti triede zoznam všetkých článkov spadajúcich pod vybraný korpus. S dostupnými všetkými článkami cieľového korpusu môžeme daným zoznamom preiterovať a volať funkcionalitu triedy *CreateInvertedIndex*.

V triede *CreateInvertedIndex* v samotnej funkcii tvorby indexu *createInvertedIndexOfArticle* začíname tým že si pomocou triedy spracúvajúcej databázove dopyty nat tokenmi článku vypýtame všetky tokeny (spracované slová) konkrétneho článku využitím metódy tokenOfArticleDao.getAllTokensOfArticle(article._id). Pomocou tohto volania dostávame zoznam všetkých slov článku aj s ich potrebnými doplňujúcimi dátami. Týmto zoznamom iterujeme a vytvárame nové Json objekty, ktorých schéma je vopred definovaná v priečinku a súbore root/schema/invertedIndex.js práve využitím informácii zo zoznamu.

```
const indexSchema = new Schema({
  word: String,
  lema: String,
  pos: String,
  ner: String,
  tf: Number,
  tfCorpus: Number,
  tfCollection: Number,
  idfCorpus: Number,
  tfidfArticle: Number,
  tfidfCorpus: Number,
  tfidfCollection: Number,
  corpusName: String,
  title: String,
  articleId: String,
  count: Number,
  articleRelevance: Number,
  corpusRelevance: Number,
  collectionRelevance: Number
}, { collection: 'invertedIndex' })
```

Táto schéma nám umožňuje udržiavať všetky dôležité údaje o slovách samotných, ale taktiež o ich príslušnosti k článku, korpusu a k celej kolekcii. V tomto bode vyplníme hodnoty:

- Word - samotné slovo
- Lema - koreň slova
- Pos - Part of speech slova
- Ner - Rozpoznané entity popsiujúce slovo
- corpusName názov príslušného korpusu, v ktorom bolo slovo
- Title - názov článku v ktorom bolo slovo
- articleId - ID článku v ktorom bolo slovo (duplicita o článku slova lebo pri ďalšom spracovaní frontend používal oba parametre)
- Count - počet výskytov presne tohto slova v konkrétnom článku korpusu

Vytvorené objekty zhromažďujeme do ďalšieho poľa predstavujúceho celkový výsledný invertovaný index jedného článku z cieľového korpusu. Toto pole je návratovou hodnotou funkcie *createInvertedIndexOfArticle()*, čo znamená že invertovaný index pre článok je vrátený do triedy *invertedIndexRouter* a môžeme vykonať vloženie poľa do databázy.

Vloženie samotných indexov do databázy prebehne znova vďaka volaniu príslušnej správcovskej triedy *invertedIndexDao* a jej metódy *createAndSave()*, ktorej parametrom je práve pole objektov invertovaných indexov. Táto metóda je popísaná aj v sekcii 5.3 *Dopyty v triede invertedIndexDao*, ale aj tu uvedieme, že bol použitý Mongoose model *insertMany*, ktorý na vstupe očakáva pole objektov spĺňajúcich schému kolekcie a sám si ošetruje efektívne vloženie do databázy. Jedná sa o fakt rýchle vykonanie, ktoré tvorbu indexov poznateľné urýchlilo.

Posledným bodom dôsledkovej činnosti triedy *invertedIndexRouter* je naznačenie v konkrétnom indexovanom článku, že bol oindexovaný. Táto trieda už pracuje s objektom identickým s tým v našej databáze predstavujúci článok, takže len priamo upravíme náš lokálny objekt a vykonáme opreráciu update nad kolekciou článkov pomocou volania *articleDao.updateOne(article, 'indexedAt')*. Všetky spomenuté podprocesy sa vykonávaju v cykle triedy *invertedIndexRouter*, kým nie sú oindexované všetky články korpusu.

3.4. Modul funkcionality tf-idf

Už v predošlej sekcii sme na obrázku popisujúcom schému invertovaného indexu mohli spozorovať hodnoty s názvom tfidf. Tfidf je číselná štatistika, ktorá má vyjadrovať aké dôležité je slovo pre dokument v korpuse, samotný korpus alebo kolekciu. Často sa používa ako váhový faktor pri vyhľadávaní informácií, pri vyhľadávaní textov a pri modelovaní používateľov. Hodnota tf-idf sa zvyšuje proporcionálne k počtu zobrazení slova v dokumente/kolekcii a je kompenzovaná počtom dokumentov v korpuse/kolekcii, ktoré obsahujú slovo, čo pomáha prispôsobiť sa skutočnosti, že niektoré slová sa vo všeobecnosti vyskytujú častejšie. Tf-idf je dnes jednou z najpopulárnejších schém na váženie termínov. Keďže využitie funkcionality invertovaného indexu a kalkulácie tf-idf sú veľmi úzko späté, preto schéma invertovaného indexu obsahuje taktiež informácie o hodnotách tf-idf konkrétnych výrazov.

Výpočet hodnôt tf-idf prebiehal nasledovne:

- Pojem tf (term frequency) nad konkrétnym článkom: **tf = počet vyskytov slova X v článku Y/počet slov článku Y**, kde **X** je hľadané slovo, pre ktoré rátame tf-idf a **Y** je konkrétny článok, v ktorom práve rátame frekvenciu.
- Pojem idf (inverse document frequency) nad konkrétnym korpusom: **IDFcorpus = log(všetky články v korpuse/články v korpuse obsahujúce pojem X)**, kde **X** je znova pojem, pre ktorý rátame hodnotu tf-idf.
- Následne hodnota popisujúca váhu pojmu v rozmedzí korpusu:
  TFIDFarticle = tf *IDFcorpus
- S malou úpravou vieme vyrátať danú hodnotu aj pre samotný korpus kde pozmeníme výpočet hodnoty tf na **TFcorpus = počet výskytov slova v korpuse/počet všetkých slov korpusu**.
- Potom **TFIDFcorpus = TFcorpus * IDFcorpus**. Jedná sa však skôr o experimentálny výpočet, ktorého správanie sa chceme sledovať.
- My však rátame význam/váhu slova aj pre celú kolekciu korpusov, teda na vyššej úrovni abstrakcie, kde si musíme výpočty trochu upraviť.
- IDF nad kolekciou rátame nasledovne: **IDFcollection = log(všetky články v kolekcii/články v kolekcii obsahujúce pojem X)**
- Podobnými postupmi ako doteraz vytvoríme aj **TFcollection = počet výskytov slova v kolekcii/počet všetkých slov kolekcie**.

- finálny výpočet je nasledovný:
  TFIDFcollection = TFcollection*IDFcollection, kde sa znovaraz jedná hlavne o experimentálny výpočet.

3.4.1. Proces výpočtu hodnôt tf-idf

Samotný proces doplnenia invertovaného indexu o tieto hodnoty je v našej implementácii vykonaný v triede *invertedIndexRouter* z dôvodu, že celé výpočty spočívajú z databázových volaní, ktoré daná trieda spravuje. Začnime výpočtom hodnôt TFIDF z pohľadu článkov koprusu a korpusu samotného. Tieto hodnoty vieme rátať vždy pre zadaný cieľový korpus. Funkcionalitu vieme vyvolať pomocou URL cesty *"<názov-stránky>:<port>/api/inverted-indices/tfidf-corpus/<názov korpusu>"*. Jedná sa o API bez využitia bezpečnostného tokenu a taktiež ani nie je dostupné z webovej admin sekcie, takže funkcionalita je dostupná len cez GET volania vyššie uvedenej URL. Táto istá informácia o volaniach bude platiť aj pre výpočet hodnôt nad kolekciou.

Pri vyvolaní výpočtu TFIDF nad korpusom metóda v prvom rade získa všetky články korpusu, pre ktoré budeme rátať hodnoty. To docielime volaním funkcionality databázovej správcovskej triedy *articleDao.getByCorpus(corpusName)*. Toto volanie nám navrátilo pole všetkých článkov korpusu. Následne , ešte pred preiterovaním článkov, si získame počet všetkých slov korpusu, keďže sa jedná o náročnejší dopyt a stačí nám ho zavolať raz za celý korpus a to volaním *invertedIndexDao.getCorpusTokensCount(corpusName)*. Začneme iterovať článkami korpusu. Získavame si dĺžku konkrétneho článku *invertedIndexDao.getArticleTextLength(corpusName, article.title)*. V tomto bode máme všetko potrebné pre výpočet hodnôt TF článku, ale potrebujeme zistiť výskyty slov v dokumentoch pre výpočet IDF a TF-korpusu. Preto iterujeme cez každý index/slovo/token článku a získame jeho výskyty v článkoch korpusu *invertedIndexDao.getWordinCorpusAppearances(resultIndex.word, resultIndex.lema, resultIndex.pos, resultIndex.ner, corpusName)*. Aby sme dostali čo najexaktnejšie výsledky posielame ako zhodovací parameter nie len slovo samotné ale aj jeho lému, pos a ner. Výstupom volania sú počty: celkový súčet výstkytov slova v korpuse a počet dokumentov obsahujúcich dané slovo. Následne na základe vyššie uvedených pravidiel dorátame hodnoty indexu schémy *tfidfArticle* a *tfidfCorpus*. Výsledky indexu/tokenu/slova uložíme do databázy volaním metódy *invertedIndexDao.appendTfIdfForArticleInCorpus(tfidf, tfidfCorpus, idf, tf, tfCorpus, resultIndex._id)*. Na záver zopakujem, že toto bola činnosť pre jedno slovo jedného článku korpusu, takže tieto činnosti sa ešte v cykloch opakujú.

Čo sa týka výpočtu hodnôt TFIDF pre celú kolekciu postup bol v mnohých bodoch veľmi podobný, preto spomenieme len odlišnosti a zhodné veci len preletíme. Výkon tejto činnosti spustíme volaním URL cesty *"<názov-stránky>:<port>/api/inverted-indices/tfidf-collection/<názov korpusu>"*. Pre urýchlenejšie vykonanie a následné skontrolovanie sa funkcionalita znova vykonáva po korpusoch. Rozdielom je že tentokrát si na začiatku získame počet všetkých slov kolekcie a nie len počet slov korpusu a to zavolaním dopytu *invertedIndexDao.getAllTokensCount()*. Tak ako aj predtým získavame si všetky články korpusu a prechádzame všetky indexy článkov. Znova

vykonáme poobný dopyt ako pri korpusoch *invertedIndexDao.getAllContainigArticlesCountAndAvg(resultIndex.word, resultIndex.lema, resultIndex.pos, resultIndex.ner)* ale v tomto prípade nám dopyt vráti súčet výskytov slova v celej kolekcii a súčet článkov obsahujúcich slovo znova v cele kolekcii. Keď máme všetky informácie, tak podľa vyššie uvedeného pravidla vyrátame hodnoty tfidfCollection pre index a výsledky ukladáme pomocou *invertedIndexDao.appendTfIdfForCollection(tfidfGlobal, tfCollection, resultIndex._id)*. Znova na záver jednalo sa o jeden index jedného článku, takže táto funkcionalita sa ešte viackrát vykoná.

3.4.2. Vzťah medzi TFIDF a relevanciou indexu

Na úplný záver sekcie len spomenieme, že hodnoty TFIDF a relevance v schéme invertovaného indexu sú veľmi previazané. Obidve hodnoty hovoria o tom istom len iným spôsobom. TFIDF vyjadruje váhu slova pre článok, korpus či kolekciu, čo znamená že čím je hodnota väčšia tak tím je slovo pre nich podstatnejšie. Z toho vyplývajú vlastnosti, ako nie priamo a rýchlo ľudsky čitateľné, exaktnejšie a hodnoty sa môžu opakovať ak boli výpočty pre slovo rovnaké. Hodnoty relevancie rátame priamo z hodnôt TFIDF takže vyjadrujú to isté, ale iným spôsobom. Jedná sa o zoradenie hodnôt TFIDF a v prípade rovnakých hodnôt dochádza k abecednému usporiadaniu, takže neexistujú duplicity. Dostávame teda poradové hodnoty dôležitosti slova pre článok, korpus a kolekciu, ktoré sú čitateľnejšie ľuďmi, ale menej exaktné, ako hodnoty TFIDF.

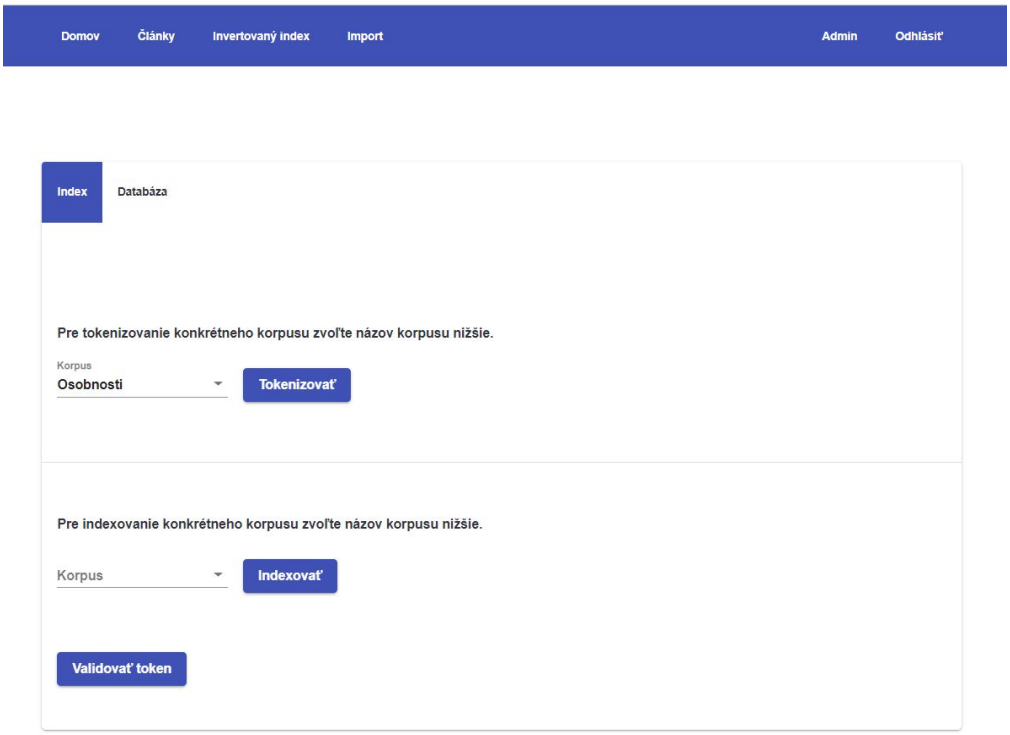
3.5. Modul článkov

Pre zobrazenie zoznamu článok, detailu článku a iných informácií bol vytvorený samostatný modul. Tento modul obsahuje zoznam článkov, detail článku vrátane tokenov, odtlačok článku a komparátor článku, v ktorom je možné porovnať samotný článok a jeho top 20 slov voči ostatným korpusov. Táto informácia má výpovednú hodnotu o tom na koľko percent článok patrí do daného korpusu. Modul článkov, presnejšie detail článku poskytuje tlačidlo na presmerovanie do modulu invertovaného indexu s už predvoleným filtrom pre daný článok, kde sa automaticky vyfiltrujú všetky invertované indexy pre konkrétny článok. Modul článku získava dáta z backendu pomocou frontendovej angular služby na základe webových služieb typu REST, tak ako ostatné služby na frontnedovej časti.

3.6. Modul správy cez Admin sekciu

Pre umožnenie práce oprávnených používateľov s databázovým prostredím bola vytvorená sekcia pre administrátora. Hlavným cieľom tejto sekcie je oddelenie funkcionality, ktorá by mohla poškodiť databázový systém od verejnej funkcionality, ktorá je považovaná za dostupnú pre všetkých. Do sekcie pre administráciu je nutné sa prihlásiť pomocou emailovej adresy a hesla, ktoré musí byť vygenerované správcom systému. Neexistuje žiadny formulár, ktorý oprávňuje bežných používateľov registrovať sa ako administrátor.

Po prihlásení sa používateľovi na základe prihlasovacích údajov vygeneruje token, ktorý je odoslaný na backendovú časť aplikácie za účelom autentifikácie pri niektorých funkcionalitách.
Samotná sekcia administrácie poskytuje funkcionalitu importovania korpusov a článkov do databázy priamo z prostredia administrácie. Dočasne je tiež vyriešený spôsob akým umožniť používateľovi anotovanie a indexovanie korpusov / článkov. Prihlásený používateľ má tiež možnosť anotovať jednotlivé články a to tak, že sa dostane do detailu konkrétneho článku, kde mu je prístupné tlačidlo na anotovanie. Tlačidlo "Validovať token" posiela požiadavku na backendovú časť a slúži na overenie platnosti tokenu, ktorý po 24 hodinách zaniká a je nutné sa znovu prihlásiť do svojho účtu. Informácia o tom, či je token platný alebo nie sa zobrazuje v konzole prehliadača po odoslaní požiadavky na backend.



4. Dokumentácia k API

Kapitola bližšie popisuje jednotlivé funkcie, ktoré API poskytuje. API je založená na metóde REST, kde sa pomocou URL adresy cez klientske rozhranie (Angular) dopytujeme na backend (NodeJS). Formát návratových hodnôt (správ) z backendu je JSON. Aplikácia poskytuje funkcie týkajúce sa jednak rôznych modelov schémy, ale aj iných služieb, ktoré sú ďalej volané.

Získanie zoznamu článkov

Popis: Získanie všetkých článkov, telom volania API je vstupný JSON. Atribúr count označuje typ odpovede s vypnutým počítadlom sú na výstupe dáta článku, text, tokeny. S prepínačom sa spočítajú všetky články s aplikovaním filtrovania.
URL: *<názov-stránky>/api/articles*
Vstupná hodnota: stránkovanie, filtrovanie, zoraďovanie
Príklad:

```
paging: {
    limit: 10,
    skip: 0
  },
  filter: {
    title: null,
    corpusName: null
  },
  sort: {
      title: 1
  },
  count : false
```

Prístup: bez tokenu
Metóda: POST
Návratová hodnota: pole článkov

Získanie konkrétneho článku

Popis: Získanie konkrétneho článku na základe identifikátora článku
URL: *<názov-stránky>/api/articles/:id*
Vstupná hodnota: identifikátor článku *(napr.: 5bebfee81d380327acf9ac40)*
Prístup: bez tokenu
Metóda: GET
Návratová hodnota: konkrétny článkov

Získanie náhodného článku

Popis: Získanie náhodného článku
URL: *<názov-stránky>/api/articles/random*
Vstupná hodnota: -
Prístup: bez tokenu
Metóda: GET
Návratová hodnota: náhodný článkov

Tokenizovanie konkrétneho článku

Popis: Tokenizovanie konkrétneho článku na základe identifikátora článku a následné uloženie do databázy.
URL: *<názov-stránky>/api/articles/tokenization/:id*
Vstupná hodnota: identifikátor článku *(napr.: 5bebfee81d380327acf9ac40)*
Prístup: s tokenom
Metóda: POST
Návratová hodnota: status *(napr.: 200 - OK , 500 - chyba)*

Import článkov z Wikipedie

Popis: Importovanie dávky článkov určitej kategórie z Wikipedie a následné uloženie do databázy.
URL: *<názov-stránky>/api/articles/import*
Vstupná hodnota: JSON objekt obsahujúci názov kategórie a zoznam článkov v kategórii
Prístup: s tokenom
Metóda: POST
Návratová hodnota: status *(napr.: 200 - OK , 500 - chyba)*
Použité databázové dopyty: articleDao.save(), articleDao.saveOne(), userDao.verifyAccess TokenAndExecuteFunction()

Import článkov z Webnoviny.sk

Popis: Importovanie dávky článkov určitej kategórie z portálu Webnoviny.sk a následné uloženie do databázy. Prebieha pomocou volania scrapperu z Python API.
URL: *<názov-stránky>/api/articles/importWebNoviny*
Vstupná hodnota: JSON objekt obsahujúci názov kategórie a zoznam článkov v kategórii
Prístup: s tokenom
Metóda: POST
Návratová hodnota: status *(napr.: 200 - OK , 500 - chyba)*

Vytvorenie tokenov a odtlačku článku

Popis: Vytvorenie tokenov a odtlačku článku pre živú analýzu
URL: *<názov-stránky>/api/articles/insert*
Vstupná hodnota: JSON object obsahujúci názov a text článku
Príklad:
{
  title: 'Slovensko opäť vyhralo ďalší zápas',
  text: 'Slovenským hokejistom sa opäť podarilo zabodovať a vyhrali nám ďalší zápas'
},
Prístup: bez tokenom
Metóda: POST
Návratová hodnota: status *(napr.: 200 - OK , 500 - chyba)*

Získanie tokenov konkrétneho článku

Popis: Získanie tokenov pre konkrétny článok
URL: *<názov-stránky>/api/article-token/:id*
Vstupná hodnota: identifikátor článku *(napr.: 5bebfee81d380327acf9ac40)*
Prístup: bez tokenu
Metóda: GET
Návratová hodnota: zoznam tokenov pre konkrétny článok

Získanie histogramu konkrétneho článku

Popis: Získanie histogramu pre konkrétny článok
URL: *<názov-stránky>/api/article-token/histogram/:id*
Vstupná hodnota: identifikátor článku *(napr.: 5bebfee81d380327acf9ac40)*
Prístup: bez tokenu
Metóda: GET
Návratová hodnota: histogram podľa POS pre konkrétny článok

Výpočet odtlačku pre tokenizovaný článok

Popis: Výpočet odtlačku článku pre daný tokenizovaný článok
URL: *<názov-stránky>/api/article-token/imprintToDB/:id*
Vstupná hodnota: identifikátor článku *(napr.: 5bebfee81d380327acf9ac40)*
Prístup: bez tokenu
Metóda: GET
Návratová hodnota: status *(napr.: 200 - OK , 500 - chyba)*

Výpočet a získanie odtlačku pre tokenizovaný článok

Popis: Získanie tokenizovaného článku z databázy, následný výpočet odtlačku článku a vrátenie JSON objektu s údajmi o odtlačku.
URL: *<názov-stránky>/api/article-token/imprint/:id*
Vstupná hodnota: identifikátor článku *(napr.: 5bebfee81d380327acf9ac40)*
Prístup: bez tokenu
Metóda: GET
Návratová hodnota: JSON objekt obsahujúci informácie o odtlačku konkrétneho článku

Získanie zoznamu korpusov

Popis: Získanie všetkých korpusov
URL: *<názov-stránky>/api/corpuses*
Vstupná hodnota: -
Prístup: bez tokenu
Metóda: GET
Návratová hodnota: pole korpusov

Získanie zoznamu článkov konkrétneho korpusu

Popis: Získanie všetkých korpusov
URL: *<názov-stránky>/api/corpuses/:corpusName*
Vstupná hodnota: názov korpusu *(napr.: Ekonomika)*
Prístup: bez tokenu
Metóda: GET
Návratová hodnota: JSON objekt obsahujúci pole článkov pre konkrétny korpus

Vytvorenie korpusu

Popis: Importovanie vytvoreného korpusu do databázy pre účely importovania článkov do novovytvoreného korpusu
URL: *<názov-stránky>/api/corpuses/import*
Vstupná hodnota: JSON objekt obsahujúci názov, popis a kľúčové slová korpusu
Prístup: s tokenom
Metóda: POST
Návratová hodnota: status *(napr.: 200 - OK , 500 - chyba)*

Tokenizovanie korpusu

Popis: Tokenizovanie všetkých článkov umiestnených v konkrétnom korpuse a následné uloženie do databázy
URL: *<názov-stránky>/api/corpuses/tokenization/:corpusName*
Vstupná hodnota: názov korpusu *(napr.: Automobily)*
Prístup: s tokenom
Metóda: POST
Návratová hodnota: status *(napr.: 200 - OK , 500 - chyba)*

Získanie zoznamu invertovaných indexu do tabuľky

Popis: Získanie všetkých invertovaných indexov podľa požadovaného formátu a stránkovaním, zoradením alebo rôznych nadhľadov inv. index : 1 článok, 2 korpus, 3 celok (sa zabezpečuje pomocou group by)
URL: *<názov-stránky>/api/inverted-indices/table/<1-3>*
Vstupná hodnota: Príklad vstupného tela pre API : stránkovanie, filter (null = bez filtrovania, zadaním textového reťazca sa v selekte vytvára match atribút), zoradenie (vstupom sú atribúty podľa, ktorých chceme sortovať asc = 1 / desc = -1)
Príklad:

```
paging: {
    limit: 10,
    skip: 0
  },
  filter: {
    corpusName: null,
    title: null,
    word: null,
    lema: null,
    pos: null,
    ner: null
  },
  sort: {
      articleRelevance : 1
  }
```

Prístup: bez tokenu

Metóda: POST
Návratová hodnota: zoznam invertovaných indexov v požadovanom group by rozhľade.

Získanie zoznamu invertovaných indexov pre konkrétny článok

Popis: Získanie všetkých invertovaných indexov pre konkrétny článok
URL: *<názov-stránky>/api/inverted-indices/get-by-article/:id*
Vstupná hodnota: identifikátor článku *(napr.: 5bebfee81d380327acf9ac40)*
Prístup: bez tokenu
Metóda: GET
Návratová hodnota: zoznam invertovaných indexov pre konkrétny článok

Vytvorenie invertovaných indexov pre konkrétny článok

Popis: Vytvorenie invertovaných indexov pre všetky články v danom korpuse a následné uloženie do databázy
URL: *<názov-stránky>/api/inverted-indices/article/:id*
Vstupná hodnota: názov článku *(napr.: "Slovensko opäť vyhralo")*
Prístup: s tokenom
Metóda: POST
Návratová hodnota: status *(napr.: 200 - OK , 500 - chyba)*

Vytvorenie invertovaných indexov pre konkrétny korpus

Popis: Vytvorenie invertovaných indexov pre všetky články v danom korpuse a následné uloženie do databázy
URL: *<názov-stránky>/api/inverted-indices/corpus/:corpusName*
Vstupná hodnota: názov korpusu *(napr.: Automobily)*
Prístup: s tokenom
Metóda: GET
Návratová hodnota: status *(napr.: 200 - OK , 500 - chyba)*

Výpočet TFIDF pre invertované indexy konkrétneho korpusu

Popis: Výpočet hodnoty TFIDF pre invertované indexy vo všetkých článkoch patriach do konkrétneho korpusu a následné uloženie do databázy.
URL: *<názov-stránky>/api/inverted-indices/import/:corpusName*
Vstupná hodnota: názov korpusu *(napr.: Automobily)*
Prístup: s tokenom
Metóda: POST
Návratová hodnota: status *(napr.: 200 - OK , 500 - chyba)*

Výpočet TFIDF pre invertované indexy všetkých korpusov

Popis: Výpočet hodnoty TFIDF pre invertované indexy pre články vo všetkých korpusoch a následné uloženie do databázy.

URL: *<názov-stránky>/api/inverted-indices/tfidf-collection*
Vstupná hodnota: -
Prístup: s tokenom
Metóda: POST
Návratová hodnota: status *(napr.: 200 - OK , 500 - chyba)*

Vytvorenie a uloženie Ngramy(5) z článkov podľa názvu korpusu

Popis: Vytvorí 5 gram z textu článkov. Samotný Ngram záznam obsahuje údaje o každom slove nachádzajúcom sa v n-tici, poradie slov a POS tvar. Do dávky sa vkladajú n-tice vytvorené z jedneho článku ukladanie sa vykonáva nad kolekciou *ngram-5.* Aplikuje sa na všetky články s názvom korpusu.
URL: *<názov-stránky>/api/article-token/createNgrams/<corpusName>*
Vstupná hodnota: názov korpusu *(napr.: Geografia)*
Prístup: s tokenom
Metóda: GET
Návratová hodnota: status *(napr.: 200 - OK , 500 - chyba)*

Vytvorenie a uloženie viet z článkov podľa názvu korpusu

Popis: Vytvorí sadu viet z textu a tokenov článku. Záznam obsahuje údaje o poradí vety, sekvenciu textového reťazca POS tvarov. Dávka obsahuje dokumenty viet vytvorených z jedného článku, ktorá sa ukladá do kolekcie *sentences.* Aplikuje sa na všetky články s názvom korpusu.
URL: *<názov-stránky>/api/article-token/createSentences/<corpusName>*
Vstupná hodnota: názov korpusu *(napr.: Geografia)*
Prístup: s tokenom
Metóda: GET
Návratová hodnota: status *(napr.: 200 - OK , 500 - chyba)*

Výpočet a zoradenie slov podľa frekvencie používania v článku, korpuse a v celku

Popis: Vykonáva sa sekvenčne nad všetkými článkami a samotným korpusom (ako posledným v poradí) s názvom z parametra API. V rámci iterácií voláme API na selekt zoradeného invertovaného indexu, ktorého záznamy následne updatneme v kolekcií *invertedIndex* o pole *articleRelevance/corpusRelevance/collectionRelevance*, ktoré určuje poradie celočíselné podľa frekvencie používania.
URL: *<názov-stránky>/api/order-word/calculate-order-for-corpus/<corpusName>*
Vstupná hodnota: názov korpusu *(napr.: Geografia)*
Prístup: s tokenom
Metóda: GET
Návratová hodnota: status *(napr.: 200 - OK , 500 - chyba)*

Získanie ekvivalentných slov z korpusu voči vstupu (zoradených slov)

Popis: V DB selekte vytvoríme filtre podľa vstupného slova a pre zadaný názov korpusu. Výsledok upravime, tak aby poradie od povedalo k poradiu vstupných slov. Formát je zjednodušený pre

výpis na front-end, obsahuje pole slov, číslo relevancie, POS formátov, NER formátov, ak sa ekvivalentné slovo nenachádza v korpuse výsledok je doplnený reťazcom ‘neexistuje’. Ďalším globálnym výstupom je názov korpusu a percentuálnu hodnotu na koľko percent slova zapadajú do korpusu.
Používanie: Live analýza textu, komparátor pre článok.
URL: *<názov-stránky>/api/order-word/calculate-order-for-corpus/<corpusName>*
Vstupná hodnota: názov korpusu v parametri *(napr.: Geografia), telo API obsahuje pole zoradenych slov vo formáte dokumentu z kolekcie invertovaný index (získaný napr. Cez API inverted-indices/table/<1-3>).*
Prístup: s tokenom
Metóda: POST
Návratová hodnota: status *(napr.: 200 - OK , 500 - chyba)*

Získanie štatistík o datasete

Popis: Získanie štatistík o počte korpusov a ich naplnenosti.
URL: *<názov-stránky>/api/statistics/index*
Vstupná hodnota: -
Prístup: bez tokenu
Metóda: GET
Návratová hodnota: JSON objekt obsahujúci štatistiky o každom korpuse v datasete

Získanie štatistík o top slovách

Popis: Získanie štatistík o top slovách pre všetky korpusy.
URL: *<názov-stránky>/api/statistics/tableData*
Vstupná hodnota: -
Prístup: bez tokenu
Metóda: GET
Návratová hodnota: JSON objekt obsahujúci zoznam korpusov a pre každý korpus top 10 slov podľa ich relevantnosti.

5. Dokumentácia k databázovým dopytom

5.1. Dopyty v triede articleDao

Metódy save(articles)/saveOne(article)

Použitie: Pri importovaní článkov zo stránok Wikipédie. Metódy v nadpise umožňujú ukladanie článkov do databázy. Metóda save viacnásobné volá vykonanie metódy saveOne, ktorej dopyt je popísaný nižšie.
Popis: Táto query je nadstavbou pomocou Mongoose modelu nad základnými dopytmi mongodb databázy, podľa mongoose dokumentácie sa vydá príkaz aktualizácie mongodb *findAndModify*. Model nájde zodpovedajúci dokument, aktualizuje ho podľa aktualizačných argumentov, vykoná všetky options a vráti nájdený dokument (ak existuje) spätnému volaniu. Dotaz sa vykoná, ak sa vráti spätné volanie, inak sa vráti objekt typu Query.
Odkaz: https://mongoosejs.com/docs/api.html#model_Model.findOneAndUpdate
Znenie:

```
Article.findOneAndUpdate(
{ 'title': article.title },
{ title:
        article.title,
        tokens: article.tokens,
        corpusName: article.corpusName,
        text: article.text,
        tags: article.tags,
        source: article.source,
        createdAt: article.createdAt,
        annotatedAt: article.annotatedAt,
        indexedAt: article.indexedAt },
{ upsert: true },
function (err, doc, res) { if (err) logger.error('Failed to save article', err) })
```

updateOne(article, ...props)

Použitie: Používa sa na aktualizovanie propertie zo vstupného parametru funkcie podľa názvu v dokumente.
Popis: Vstupom je článok s novými properties, ktoré sa aktualizujú na článku s ID.
Odkaz: -
Znenie:

```
Article.findAndUpdate(
{ 'title': article.title },
{ '..props': article.props.value },
{ upsert: true },)
```

getAllFilterSortPagging(filters, sort, paging)

Použitie: Používa sa na získanie zoznamu článkov bez textu a tokenov. Táto query sa používa, keď doťahujeme veľké zoznamy článkov. V reálnom čase sa dopočita dĺžka textu
Popis: Vstupom je článok s novými properties, ktoré sa aktualizujú na článku s ID.
Odkaz: Query hľada všetky zhody v článkov podľa vstupných filtrov, následne prejektuje, zoraďuje a zabezpečuje stránkovanie.
Znenie:

```
Article.aggregate([
{$match: {corpusName: corpusName}},
{$project: {
        _id: 1,
        title: 1,
        annotatedAt: 1,
        createdAt: 1,
        indexedAt: 1,
        source: 1,
        tags: 1,
        textLength: {$strLenCP: $text}
}},
{$sort:  {title: 1}},
{$skip:  {10 * 1}},
{$limit:  {1}}
])
```

5.2. Dopyty v triede tokenOfArticleDao

Metóda getCountGrammarCategoryOfarticle(articleId)

Použitie: Používa sa pri získaní histogramu pre článok z POS foriem.
Popis: Query zráta všetky zhodne POS formy pomocou group by klauzuli a funkciou sum zráta zhodné skupiny.
Odkaz: -
Znenie:

```
Article.aggregate([
{$match:{_id: ObjectID(XXXX)}},
{$unwind: '$tokens.processedText'},
{$group:{_id:
        pos: {
                $arrayElemAt:[{$arrayElemAt: [$tokens.processedText.pos.forms, 0]}, 0]
        } },
        count: {$sum : 1}
}])
```

Metóda getAllTokensOfArticle (article._id)

Použitie: Používa sa pri tvorbe invertovaného indexu.
Popis: Jedná sa znova o použitie Mongoose modelu findOne. Táto query prehľadáva kolekciu článkov a hľadá zhodu na základe ich id. V prípade zhody vráti zoradené tokeny článku.
Odkaz: https://mongoosejs.com/docs/api.html#model_Model.findOne
Znenie:

```
Article.findOne(
{ '_id': articleId },
{ 'tokens': 1 }).sort({ 'title': 1 }
).collation({ locale: 'sk' }).limit(1) .exec() }
```

5.3. Dopyty v triede invertedIndexDao

Metóda createAndSave(index)

Použitie: Pri uložení vytvorených invertovaných indexov jedného článku.
Popis: Použitie príkazu insertMany pre umožnenie efektívneho vkladu viacerých elementov naraz. Po vykonaní sa uloží celé pole indexov do kolekcie.
Odkaz: https://docs.mongodb.com/manual/reference/method/db.collection.insertMany/
Znenie:

```
InvertedIndex.insertMany(indicesArray, function (err, mongooseDocuments) {
if (err) {
        LOG.error('Failed to save indices' + indicesArray[0].title, err)
}
else {
        LOG.debug('indices of ' + indicesArray[0].title + 'were succefully inserted')
}})
```

Metóda getCorpusTokensCount(corpusName)

Použitie: Pri tvorbe hodnôt TFIDF pre zistenie počtu slov korpusu.

Popis: Prehľadávame kolekciu invertedIndex a hľadáme zhodu so zadaným názvom korpusu. Nad všetkými výsledkami vykonáme group by znova nad názvom korpusu (čo nám spojí uplne všetky predtým matchnuté výsledky) a sčítame hodnoty atribútov $count každého jedného indexu. Count predstavuje početnosť indexu/slova/tokenu v článku.
Znenie:

```
InvertedIndex.aggregate([
{ $match: { 'corpusName': corpusName } },
{ $group: { _id: '$corpusName', wordCount: { $sum: '$count' } } }
])
```

Metóda getArticletextLength(corpusName, article.title)

Použitie: Pri rátaní hodnôt TFIDF pre vyrátanie hodnoty TF pre článok.
Popis: V kolekcii invertedIndex hľadáme indexy, ktoré nám spadajú do konkrétného článku, konkrétneho korpusu. Výsledky group by-neme podľa článku a sčítame hodnoty atribútov $count. Count predstavuje početnosť indexu/slova/tokenu v článku, preto sčítavame count a nie word. Ak bolo slovo v článku 5x count nám prišíta 5 a word by pričítalo 1.
Znenie:

```
InvertedIndex.aggregate( [
{ $match: { title: givenTitle, corpusName: givenCorpus } },
{ $group: { _id: '$title', length: { $sum: '$count' } } }
] )
```

Metóda getByCorpusAndArticle(corpusName, article.title)

Použitie: Pri výpočte TFIDF pre získanie správnych indexov, ktorých hodnoty TFIDF ideme rátať a neskôr editovať.
Popis: Len jednoduché nájdenie indexov v kolekcii invertedIndex na základe korpusu a článku a ich navrátenie programu.
Znenie:

```
InvertedIndex.find(
{ 'corpusName': corpusName, 'title': title }
).exec()
```

Metóda getWordinCorpusAppearances(word, lema, pos, ner, corpusName)

Použitie: Pri počítaní TFIDF hlavne pre hodnotu IDF korpusu a taktiež TF v rámci celého korpusu.
Popis: Prehľadávame kolekciu invertedIndex. Indexy vyhľadávame v rámci korpusu a chceme získať nielen rovnaké indexy/tokeny/slová, ale aj indexy čo majú rovnaké pos, ner, lému. Preto nad danými štyrmi hodnotami vykonáme group by. Vraciame výskyt v dokumente,pre ktorý stačí sum: 1 lebo len chceme vedieť koľko dokumentov slovo vôbec obsahuje pred IDF a početnosť slov je súčet hodnôt $count keďže chceme vedieť aj reálny počet výsktov slov v korpuse.

Znenie:

```
InvertedIndex.aggregate( [
{ $match: {
        word: givenWord,
        corpusName: givenCorpus,
        lema: givenLema,
        pos: givenPos,
        ner: givenNer } },
{ $group: {
        _id: {
                word: '$word',
                lema: '$lema',
                pos: '$pos',
                ner: '$ner'
        },
         appearances: { $sum: 1 },
        wordCount: { $sum: '$count' } } }
] )
```

Metóda invertedIndexDao.appendTfIdfForArticleInCorpus(tfidf, tfidfCorpus, idf, tf, tfCorpus, _id)

Použitie: Pri ukladaní novovyrátaných hodnôt TFIDF korpusu pre jeden index.
Popis: Vykonanie jednoduchého updateu už existujúceho indexu o novovyrátané hodnoty. Vyh ľadáme si cie ľový index na základe jeho ID a updateneme hodnoty TFIDF korpusu.
Znenie:

```
InvertedIndex.updateOne(
{ '_id': indexId },
{ $set: {
        'tf': tf,
        'tfCorpus': tfCorpus,
        'idfCorpus': corpusIdf,
        'tfidfArticle': articleTfIdf,
        'tfidfCorpus': corpusTfIdf }
}, function (err, doc, res) { if (err) LOG.error('Failed to update tfidf for given corpus ' + indexId, err)
})
```

Metóda invertedIndexDao.getAllTokensCount()
Použitie: Pri rátaní TFIDF na úrovni celej kolekcie.
Popis: Získame si počet všetkých indexov existujúcich v kolekcii. Využijeme agregáciu nasledovanú $group: null. Tento výraz nam group by -ne všetky výsledky, v tomto prípade celú kolekciu.
Znenie:

InvertedIndex.aggregate([{ $group: { _id: null, wordCount: { $sum: '$count' } } }])

Metóda appendTfIdfForCollection(tfidfGlobal, tfCollection, resultIndex. id)

Použitie: Pri ukladaní novovyrátaných hodnôt TFIDF kolekcie pre jeden index.

Popis: Vykonanie jednoduchého updateu už existujúceho indexu o novovyrátané hodnoty. Vyh ľadáme si cie ľový index na základe jeho ID a updateneme hodnoty TFIDF kolekcie.

Znenie:

```
InvertedIndex.updateOne(
{ '_id': indexId },
{ $set: {
        'tfCollection': tfCollection,
        'tfidfCollection': tfidf
        }
}, function (err, doc, res) { if (err) LOG.error('Failed to update tfidf for collection for index: ')
})
```

Metóda getFiltredSortedGroupBySchemeWithPaging(group, sort, filters, paging)

Použitie: Používa sa na komplexné selektovanie dát z invertovaného indexu. Selekt dokáže dáta rôzne zoskupovať v použití rozh ľadov (článok, korpus, celok) nad kolekciou invertovaný index. Selekt je volaný z front-endu pri rozh ľadovej tabu ľke, komparátore pri získaní N zoradenych slov a v ďalších interných volaniach.

Popis: Selekt pipeline sa vyskladáva dynamicky. Základom je agregácia, ktorrú tvorí $match pod ľa vstupnej filter pipeline (corpusName: ''Osobnosti''), ďalším dátovodom je group pipeline

Odkaz: -

Príklad:

```
Article.aggregate([
{$match: {corpusName: corpusName}},
{$group: {
        _id: {,
                word: $word,
                lema: $lema,
                ner: $ner,
                corpusName: $corpusName,
        },
        tfCorpus: $tfCorpus,
        count: {$sum: $count}
}},
{$sort:  {corpusName: 1}},
{$skip:  {10 * 1}},
{$limit:  {1}}
])
```

Metóda updateOrderWordInBulk(updateFilter)

Použitie: Dávkové aktualizovanie invertovaného indexu sa používa pri priradovaní celočíselného hodnoty poradia podľa slova, lemy v článku. Analogicky sa používa aj pre corpus, kedy je selekt zložený z iných pipeline pre filter a update.

Popis: Query sa skladá dynamicky podľa požadovanej aktualizácií záznamu. Pri aktualizácií relevancie slova v korpuse by dátovod filtru z príkladu znenia obsahoval corpusName namiesto articleId a aktualizované pole by bolo corpusRelevance namiesto articleRelevance. Výhodou je, že v rámci jednej požiadavky na DB dokáže aktualizovať niekoľko záznamov naraz.

Odkaz: -

Znenie:

```
InvertedIndex.bulkWrite(
{$updateMany:
        {articleId: ObjectID(5c0fd1518dsaf4rg48fg),
        word: 'Ahoj',
        lema: 'ahoj'
        },
        {articleRelevance: 1},
        {articleId: ObjectID(5c0fd1518dsaf4rg48fg),
        word: 'hlina',
        lema: 'hlina'
        },
        {articleRelevance: 2}}
)
```

Metóda findOrderWordInBulk(filter,Array corpusName, filterKeys)

Použitie: Používa sa ako druhý krok v porovnávači poradia vstupného poľa slov s slovami v zvolenom korpuse.

Popis: Najpodstatnejšou časťou je tvorba pipeline pre získanie ekvivalentných slov k vstupnému polu. Query berie ohľad nato, že v kolekcií invertovaného indexu sa slova nachádzajú duplicitne a musíme použiť pipeline group by podľa filtrovatelných kľúčov (word, pos, ner), ktoré zabezpečia zjednotenie výsledkov a získanie príslušného corpusRelevance, tfidfCorpus.

Odkaz: -

Znenie: Príklad pre vyhľadanie ekvivalentných slov k YYY a XXX.

```
InvertedIndex.aggregate([
{$match: {
        $or: [   {word: "YYY", pos: "YYY", lema "YYY"},
                 {word: "XXX", pos: "XXX", lema: "XXX"}
        ]
}},
{$group: {
        _id: {
                 word: word,
                 pos: pos,
                 lema: lema
        },
        corpusRelevance: {$first : $corpusRelevance},
        corpusRelevance: {$first : $tfidfCorpus}
}}
])
```

6. Záver

V zimnom semestri sme sa venovali globálnej analýze oblasti problematiky. Veľmi rýchlo sme začali s implementáciou systému, na ktorej každý týždeň usilovne pracujeme a napredujeme. Podarilo sa nám navrhnúť, implementovať a nasadiť rozhranie s vybranými funkcionalitami. Zamerali sme sa najmä na zber dostatočných počtov dát (článkov) z viacerých zdrojov, nad ktorými sa vykonávajú rôzne úpravy a analýzy (tokenizácia, invertovaný index, filtrovanie, …).
Absolvované šprinty hodnotíme pozitívne, odviedli sme podstatný kus práce, ktorý náš projekt vyžaduje. Nadobudli sme množstvo nových znalostí a poznatkov s použitím (pre väčšinu členov tímu) nových technológií.

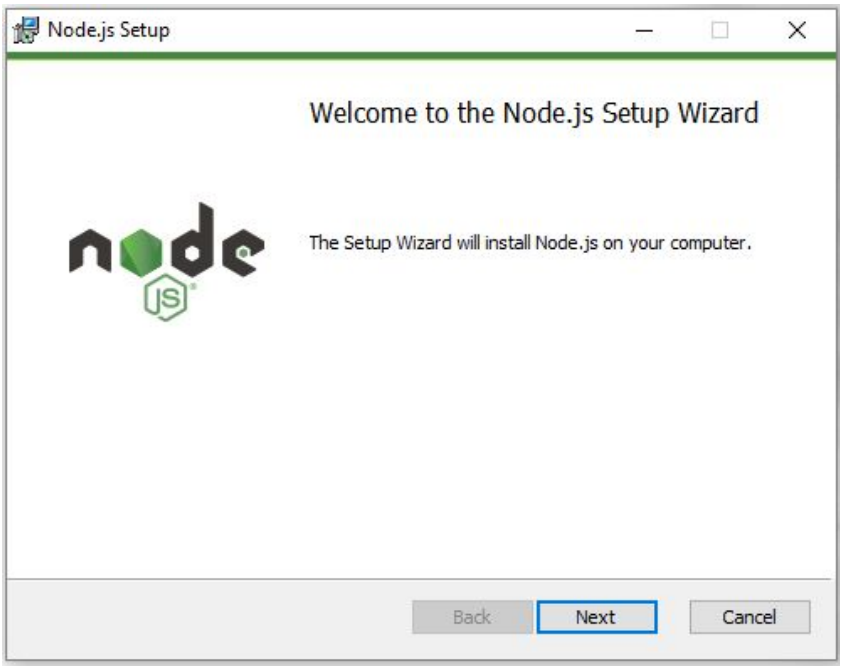
Príloha: B-1 Inštalačná príručka

Pre lokálnu inštaláciu a používanie aplikácie TxtEnv je potrebné mať na svojom lokálnom počítači nainštalovaný NodeJS a Python. Taktiež je potrebné stiahnuť repozitár so všetkými zdrojovými súbormi (pomocou príkazu git clone), ktorý je dostupný na GitHube, url: https://tfs.fiit.stuba.sk:8443/tfs/StudentsProjects/TxtEnv/_git/TxtEnv%20TFS%20Repozit%C3%A1r

- **Inštalácia NodeJS**

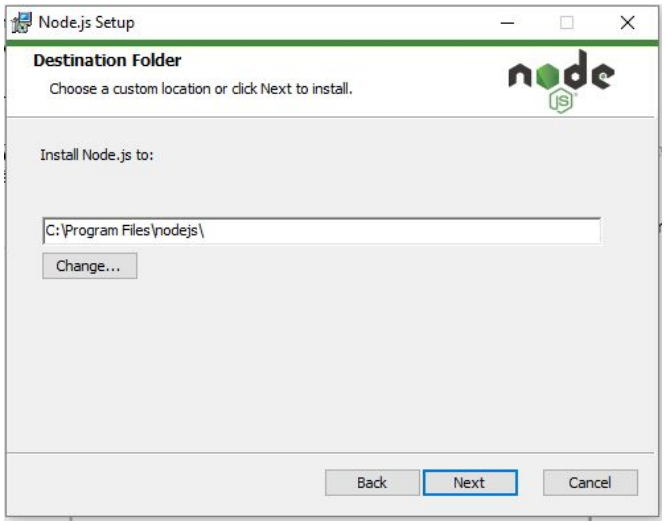
Inštalačný súbor je možné stiahnuť z webovej stránky: nodejs.org/en/download/. Dostupné sú inštalačné súbory na všetky platformy (WIN, MAC, Linux) v 32 a 64 -bitovej verzií.

Po stiahnutí súboru spustíme inštaláciu jednoduchých dvojklikom na súbor a nasledujeme inštrukcie podľa sprievodcu inštaláciou.

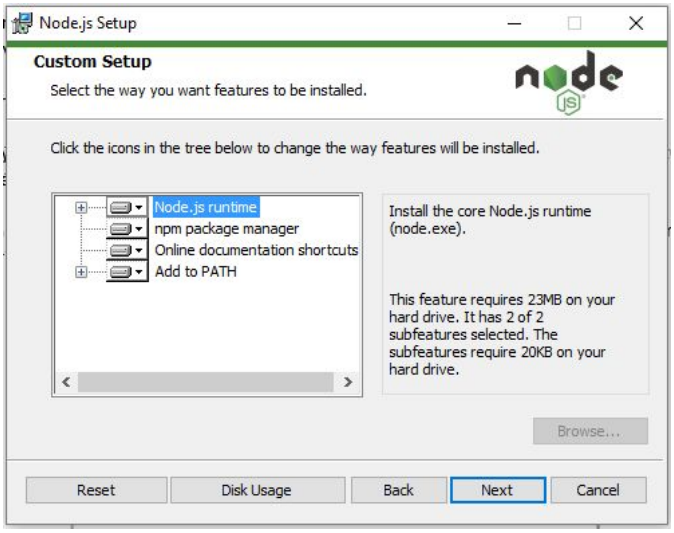


Začatie inštalácie. Ďalší krok (Next).

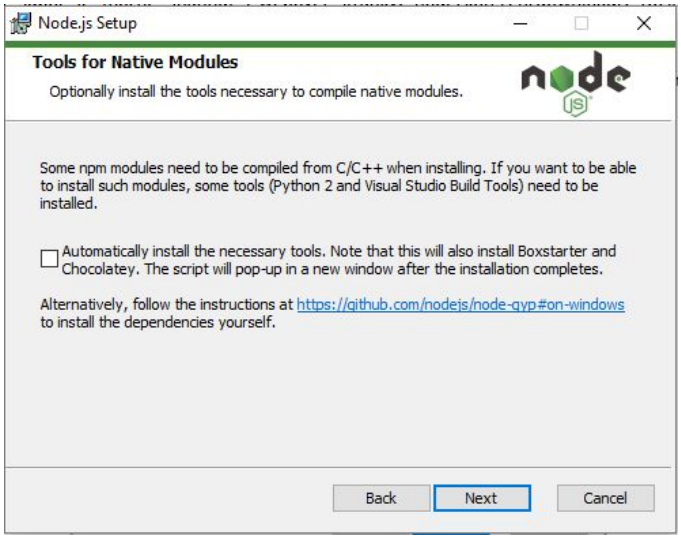
Súhlas s licenčnými podmienkami. Ďalší krok (Next).



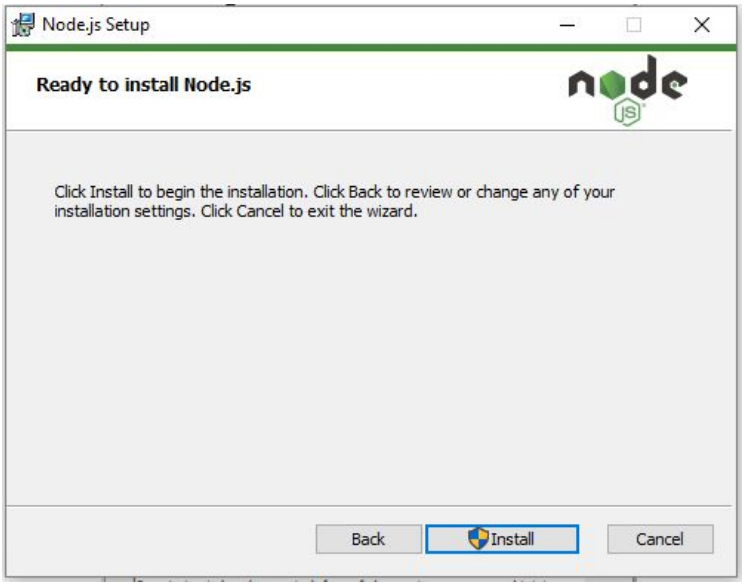
Výber adresára. Ďalší krok (Next).



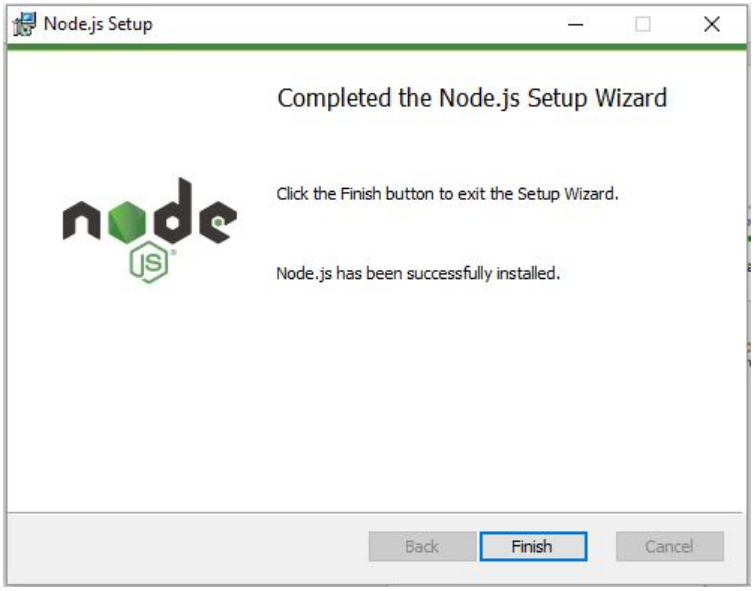
V tomto kroku nemeníme žiadne z nastavení. Pokračujeme na ďalší krok (Next).



Vo ľba tejto možnosti je na používateľovi, nie je povinná. Pokračujeme ďalším krokom (Next).

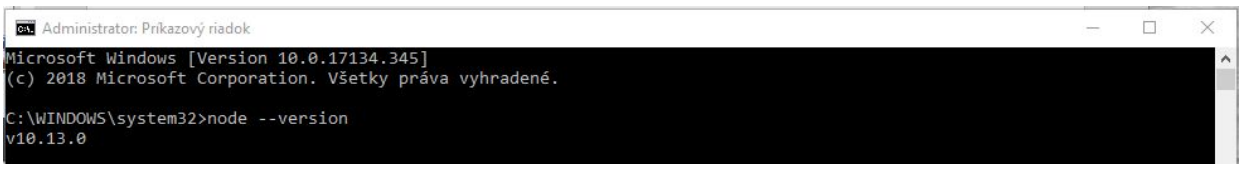


Inštaláciu zahájime kliknutím na tlačidlo Install.



Dokončenie inštalácie potvrdíme kliknutím na tlačidlo Finish.

Po úspešnom nainštalovaní overíme správnosť nainštalovania prostredníctvom príkazového riadka (konzoly). Spustíme si konzolu (ako správca) a zadáme príkaz node –version



Pokiaľ bol NodeJS správne nainštalovaný, odpoveďou na nami zadaný príkaz je verzia NodeJS, ktorá je aktuálne nainštalovaná na počítači.

V ďalšom kroku je potrebné nainštalovať závislosti potrebné pre server. Opäť spustíme konzolu, presunieme sa do základného priečinku git repozitára s projektom a zadáme príkaz npm install.

Úspešnú inštaláciu overíme spustením servera a teda zadaním príkazu npm start v príkazovom riadku. Spustenie servera je oznámené odpoveďou: Server running at http://localhost:8080/.

Ďalej nainštalujeme potrebné závislosti pre framework Angular. V konzole sa presunieme do priečinka
Angular a zadáme príkaz npm install.

Úspešnú inštaláciu frameworku Angular opäť overíme zadaním príkazu npm run start-angular (spustenie). Aktívnosť servera je oznámená odpoveďou: Angular Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200/.

Po zadaní uvedenej adresy do prehliadača je dostupná webová aplikácia TxtEnv. Pre správne fungovanie je však nutné nainštalovať ešte Python časť, nakoľko niektoré časti funkcionality sú implementované v tomto jazyku.

- **Inštalácia Python**

Z oficiálnej webovej stránky Python si stiahneme inštalačný súbor podľa platformy a verzie (32/64-bit). Odporúčame stiahnúť verziu „executable installer“, ktorá je doplnená o rozhranie s intuitívnym sprievodcom inštaláciou. Po stiahnutí súboru jednoducho otvoríme inštalačný súbor dvojklikom.

Je potrebné označiť možnosť „Add Python 3.6 to PATH“. Inštaláciu spustíme kliknutím na možnosť
Install now.

Po nainštalovaní je ešte potrebné doinštalovať všetky potrebné balíky (knižnice), ktoré sú využívané v rámci projektu. Tieto balíky sú v súbore DEPENDENCIES. Balíky nainštalujeme pomocou príkazu npm run install-python.

Samotný python zapneme nastavením sa do základného priečinka repozitára a zadaním príkazu npm run start-python.

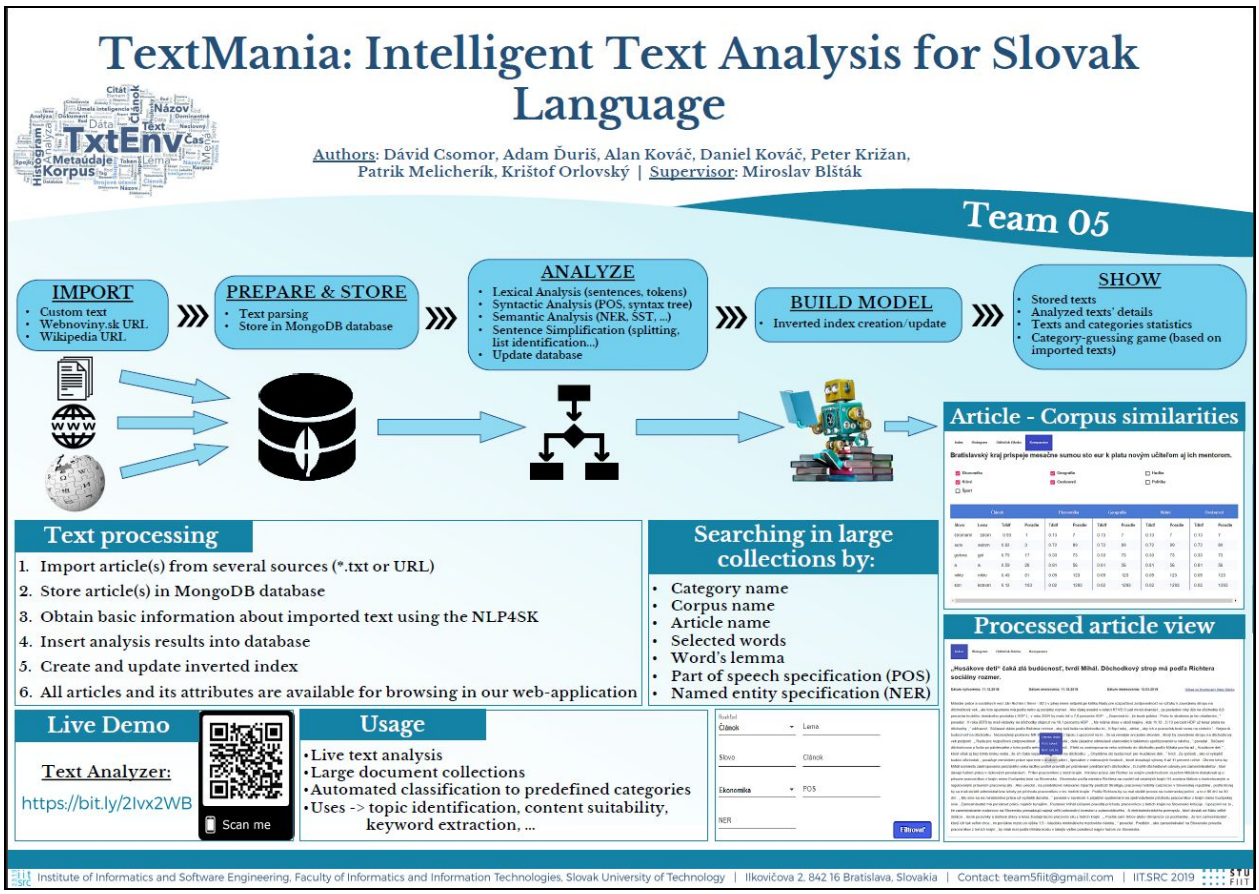
Prostredia sú rozdelené na vývojárske a produkčné. Server s produkčnou databázou spustíme pomocou skriptu npm run start:prod, vývojarske prostredie spustíme príkazom npm run start:dev.

Porty, na ktorých sú spustené jednotlivé časti projektu:

- Server: localhost 8080
- Angular: localhost 4200
- Python: localhost 8000

Príloha: B-2 Materiály z konferencie IIT.SRC 2019

Tímový plagát



Tímová prezentačná brožúrka

TextMania

Intelligent Text Analysis for Slovak Language

Team 05

- Live text analysis
- Large document collections
- Automated classification to predefined categories
- Uses -> topic identification, content suitability, keyword extraction, ...

Live Demo

Text Analyzer:

https://bit.ly/2Ivx2WB

Scan me

Institute of Informatics and Software Engineering, Faculty of Informatics and Information Technologies,
Slovak University of Technology | Contact: team5fiit@gmail.com | IIT.SRC 2019

Článok na konferenciu IIT.SRC

TextMania: Intelligent Text Analytics

Bc. Dávid CSOMOR*, Bc. Adam ĎURIŠ*, Bc. Alan KOVÁČ*, Bc. Daniel KOVÁČ*,
Bc. Peter KRIŽAN*, Bc. Patrik MELICHERÍK*, Bc. Krištof ORLOVSKÝ*

Slovak University of Technology in Bratislava
Faculty of Informatics and Information Technologies
Ilkovičova 2, 842 16 Bratislava, Slovakia
team5fiit@gmail.com

Nowadays, there is a huge amount of digitized written information, which surrounds us and is growing every minute. Our inability and lack of time to carefully read and organize every new article and categorize it, requires a tool for automated text analysis and tag recommendation. This tool should be able to import new texts, categorize them based on provided tags train the machine learning model and allow future texts to be categorized based on that pre-trained model. There are already several tools like this, but none of them is useful for Slovak language.

With this purpose in mind, we are developing a web-based environment with Node.js backend for analysis of text documents in Slovak language. We use technologies and advancements from the field of machine learning and automated text processing to achieve automated tagging and categorization.

In this paper, we focus on importing, analysing and automated categorization of new articles based on their content. Using our solution, we can easily import and categorize new articles based on our pretrained machine learning model.

Our proposed text analysis environment is designed with intent to be user friendly above all, so it can be used not only by data scientists, but also by regular users (without programming or scientific background). This should lead to improvements of our machine learning model, because we should have more input data. It is very important for us to make the process as easy and accessible as possible, so users can help us grow our datasets and improve our algorithms.

Our environment is focused on working with large collections of texts in Slovak language (e.g. various news articles, encyclopedia articles or any other text). For initial tracking, we firstly obtained articles from Slovak instance of Wikipedia[1] and Slovak popular news site Webnoviny[2]. Then we tagged them with keywords from the source page. Texts of these articles are stored in MongoDB, which is a flexible and scalable NoSQL database useful for textual data.

The first text processing task in our pipeline is word-by-word analysis. For this we use a third-party web-application NLP4SK[3], which is a set of tools for natural language processing. NLP4SK provides tools for text tokenization, sentence identification, lemmatization, part-of-speech (POS) tagging, named entity recognition (NER) and some other features.

After processing, all data are available for browsing and visualization in our web-application. Our main use case of text analysis is:

1. Import article(s) from several sources (*.txt or URL).
2. Store article(s) in MongoDB database.
3. Obtain basic information about imported text using the NLP4SK.
4. Insert analysis results into database.
5. Create and update inverted index.
6. Retrain machine learning model used for automatic categorization.
7. All articles and its attributes are available for browsing in our web-application.

Articles added during the development were manually tagged by our team, which provided a starting point for our machine learning process. We analyzed the inserted text and stored the retrieved data such as grammatical categories (*POS* – part-of-speech), identified entities (*NER* – named-entity-recognition) such as location and person, lemmas and N-Grams.

---

* Master degree study programme in field: Software Engineering
Supervisor: Dr. Miroslav Blšták, Institute of Informatics, Information Systems and Software Engineering, Faculty of Informatics and Information Technologies STU in Bratislava

[1] https://sk.wikipedia.org

[2] https://www.webnoviny.sk

[3] Link to NLP4SK: http://arl6.library.sk/nlp4sk/

IIT.SRC 2019, Bratislava, April 17, 2019, pp. 1–2.

Our solution is based on two main parts. Figure 1 shows an overview of the system architecture. Client side of the application is built on Angular[4] framework, which provides flexibility for the frontend. User interface currently allows user to:

- Add new articles from supported sources.
- Show visualized statistical data based on stored articles.
- Search and browse articles based on specified criteria.

Inverted index section, shown on Figure 2, allows user to search our database based on:

- Category name
- Selected words
- Article names
- Part of speech specification
- Named entity specification
- Word's lemma
-

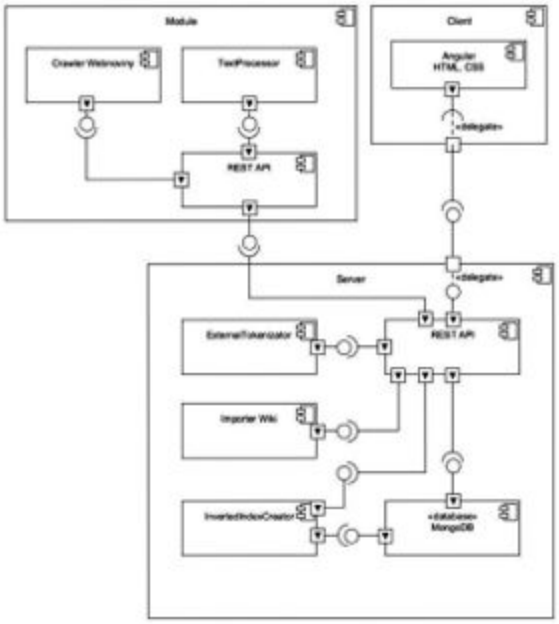


Figure 1.System architecture.

Server side of the application is built on Express[5] framework and running in Node.js environment. This component provides database access and routing. In this part are implemented "Wikipedia importing", "inverted index creation" and all other features mandatory for the client side.

There is also component powered by Django[6] framework which is the backbone of our machine learning processes. In the Python module, there is also implemented "articles importing from "Webnoviny".

All these components are communicating with each other through the HTTP and are independent. This allows you to deploy the Python component to another server and thus facilitate processing on the JavaScript component (the client-side searching through articles will be still smooth).

Figure 2. Application preview – inverted index view.

Final product will allow users to add new articles to our solution through our web page interface, where it will be processed, tagged and stored for later analysis by the user. Regular users could use it for categorization and researchers can use it for data analysis purposes, since they will have detailed information about this process and so they should be able improve their algorithms.

[4] Link to angular framework: https://angular.io
[5] Link to express framework: https://expressjs.com
[6] Link to Django: https://www.djangoproject.com

Príloha: B-3 Používateľská príručka

Používateľská príručka

Obsah používateľskej príručky

Úvod

Táto používateľská príručka má za cieľ priblížiť a objasniť jednotlivé funkcionality nástroja na inteligentnú analýzu slovenských textov s názvom TxtEnv. Tento nástroj je vyvíjaný v rámci predmetu Tímový projekt na Fakulte informatiky a informačných technológií Slovenskej technickej univerzity v Bratislave a vyvíja ho tím číslo 5 - TextMania.

Adresa, na ktorej je spomínaný nástroj dostupný, je http://bit.ly/2U6YFYa.

Štruktúra používateľskej príručky je rozdelená do častí podľa jednotlivých častí (modulov) nástroja.

Jednotlivé časti (moduly) aplikácie sú sprístupnené pomocou navigačného menu umiestneného na vrchnej časti stránky a do žiadanej časti aplikácie sa možno dostať jednoduchým kliknutím na názov časti.
Dostupné časti aplikácie sú:

- Domovská obrazovka
- Živá analýza
- Články
- Invertovaný index
- Hra s vetami

Domovská obrazovka

Domovská obrazovka sa zobrazí ako hlavná obrazovka pri spustení webovej aplikácie a zobrazuje štatistické údaje o nástroji.

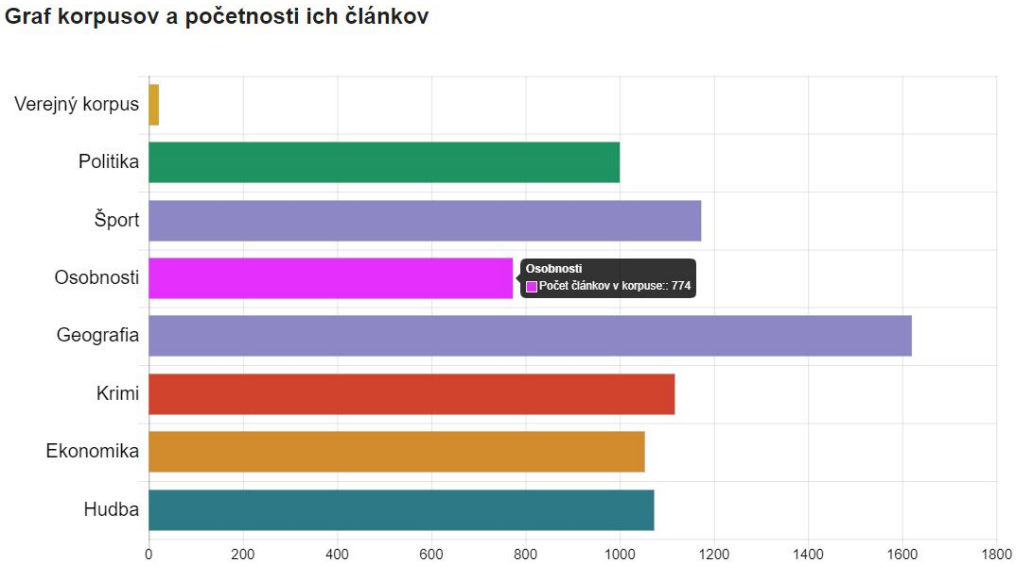
Všeobecné textové informácie

Na vrchnej časti obrazovky sú textovou formou zobrazené nasledovné údaje:

- Počet dostupných korpusov s článkami
- Celkový počet dostupných článkov
- Počet otokenovaných článkov
- Počet oindexovaných článkov

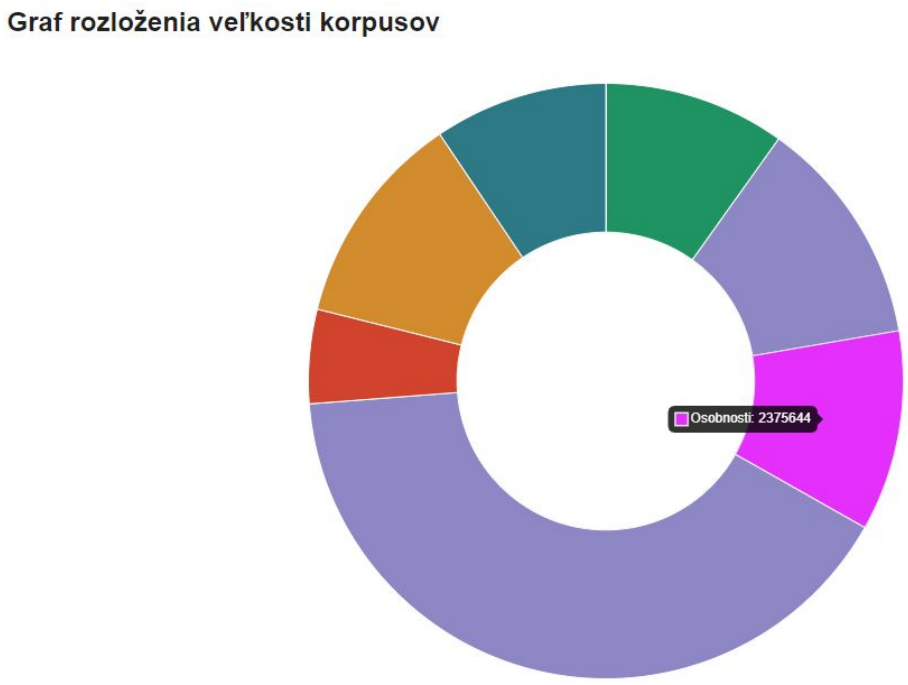
Graf korpusov a početnosti ich článkov

Pod textovými informáciami o počte dostupných korpusov a článkov sa nachádza stĺpcový graf znázorňujúci počty článkov v jednotlivých korpusoch. Pri ukázaní myšou na jednotlivé korpusy sa v textovej forme zobrazí informácia o presnom počte článkov v danom korpuse.



Graf rozloženia veľkosti korpusov

Ďalším zobrazeným grafom je koláčový graf, ktorý vyjadruje veľkosť jednotlivých korpusov s ohľadom na celkový počet znakov, ktoré sa v korpuse nachádzajú. Vyjadruje teda súčet dĺžok jednotlivých článkov v danom korpusov s ohľadom na počet znakov. Podobne, ako v predchádzajúcom grafe, aj v tomto grafe je možné zobraziť presnejšiu informáciu o počte znakov ukázaním na daný korpus pomocou myši.



Tabuľka najrelevantnejších slov podľa korpusov

V spodnej časti domovskej obrazovky je vyobrazená tabuľka s najrelevantnejšími slovami vo všetkých dostupných korpusoch. Riadky v spomínanej tabuľke prislúchajú k jednotlivým korpusom a stĺpce reprezentujú poradie relevantnosti slov (prvý stĺpec zobrazuje najčastejšie vyskytujúce sa slovo v korpuse atď.). Nad tabuľkou sa nachádza prepínač, pomocou ktorého je možné si z tabuľky odfiltrovať tzv. stopslová (*stopwords*) tak, aby sa v tabuľke zobrazovali čo najrelevantnejšie výrazy. Kliknutím na názov korpusu v tabuľke je možné presmerovať sa na invertovaný index slov pre daný korpus.

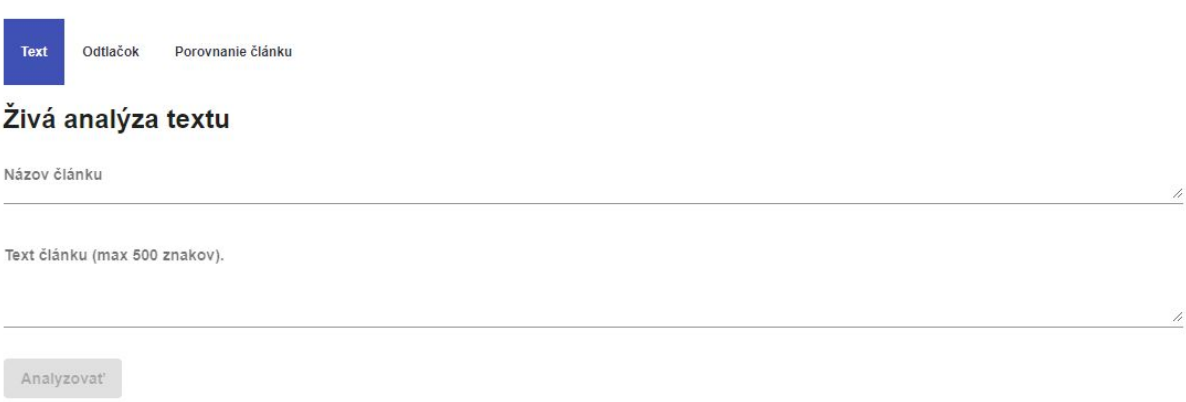
Živá analýza

Časť aplikácie s názvom “Živá analýza” ponúka používateľovi možnosť analyzovať ľubovoľný vlastný text, zobraziť jeho tokeny, porovnať ho s článkami v ostatnom korpuse a tým ho percentuálne zaradiť do jednotlivých korpusov. Samotná časť so živou analýzou obsahuje 3 záložky, v ktorých sa možno navigovať obdobným spôsobom ako v hlavnom navigačnou menu aplikácie vo vrchnej časti obrazovky:

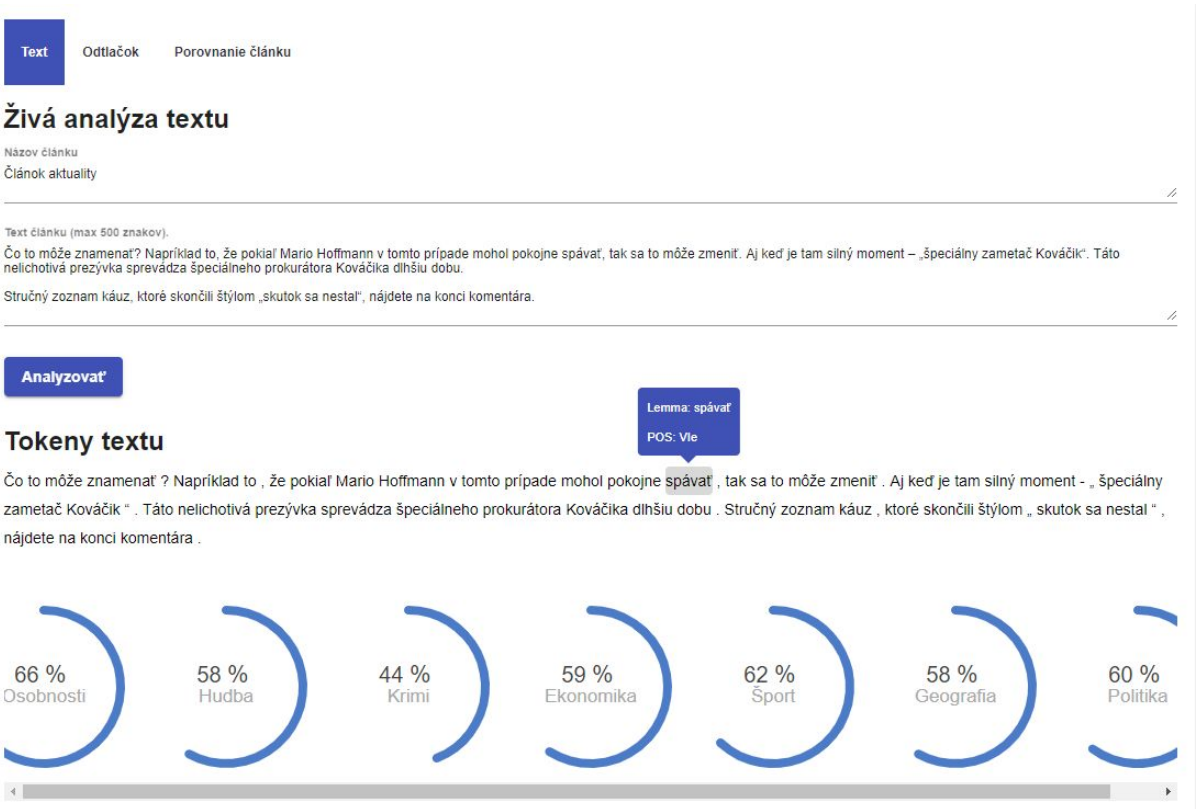
- Text
- Odtlačok
- Porovnanie článku

Záložka Text

V tejto záložke sa nachádzajú dve textové polia. Menšie textové pole slúži pre zadanie nadpisu analyzovaného článku a väčšie textové pole slúži na samotný text článku. Dĺžka textu článku je obmedzená, a to na dĺžku 500 znakov. Na spustenie analýzy článku zadaného používateľom je potrebné, aby obe zmienené polia obsahovali nejaký text a neboli prázdne.

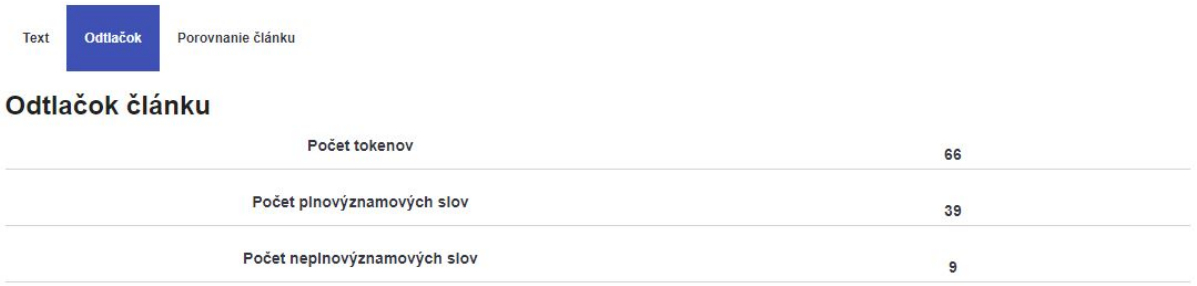


Po vyplnení oboch textových polí je umožnené používateľovi kliknúť na tlačidlo s textom “Analyzovať” a tým spustiť analýzu vloženého článku. Po úspešnej analýze článku sa pod dvoma textovými poliami zobrazia tokeny článku. Používateľ má možnosť ukázať myšou na jednotlivé tokeny, čím sa mu zobrazia dodatočné informácie o danom tokene (tagy, POS, léma, NER atď.). Pod tokenmi článku sú zobrazené percentuálne podobnosti článku s dostupnými korpusmi, resp. pravdepodobnosti príslušnosti analyzovaného článku do korpusov.



Záložka Odtlačok

V záložke “Odtlačok” sa zobrazuje tzv. odtlačok článku a nachádzajú sa tu celkovo 3 tabuľky, ktoré budú opísané v takom poradí, v akom sa zobrazujú na obrazovke (v smere zvrchu nadol). Prvá tabuľka zobrazuje celkový počet tokenov v analyzovanom článku spolu s počtom plnovýznamových a počtom neplnovýznamových slov.



Druhá tabuľka reprezentuje histogram POS (part of speech) pre jednotlivé tokeny. V prvom stĺpci tabuľky sa zobrazuje slovný druh, v druhom stĺpci počet slov s daným slovným druhom a v treťom stĺpci sa zobrazuje percentuálne zastúpenie daného slovného druhu v článku.

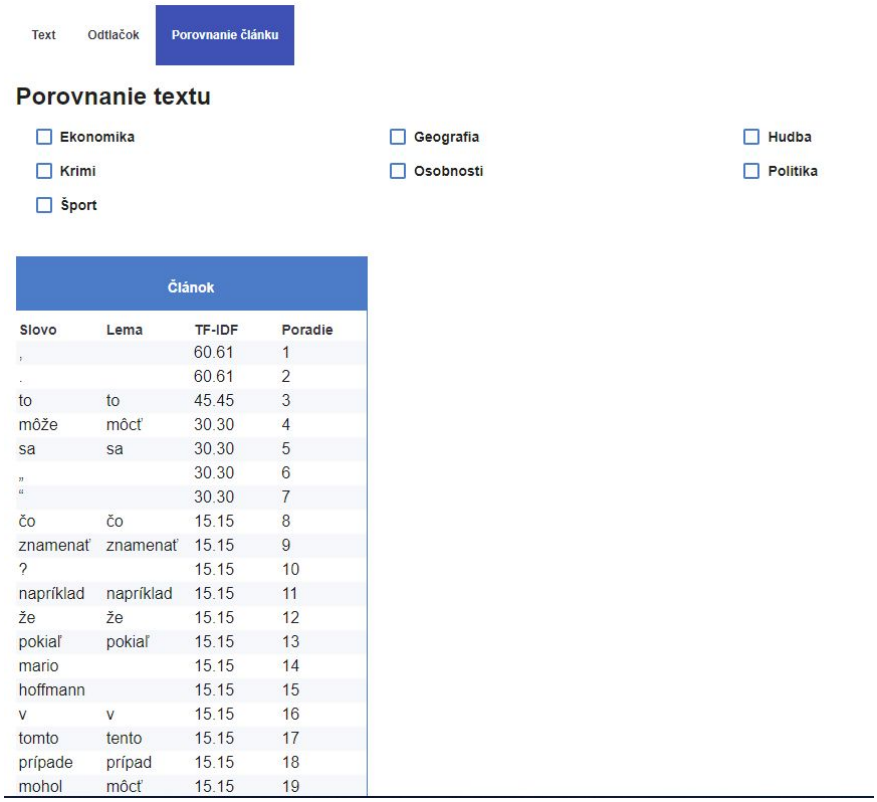
POS histogram

Slovný druh	Počet	%
neidentifikovane	2	3.03
Spojka O	6	9.09
Zámená P	7	10.61
Slovesá V	11	16.67
Interpunkcia Z	14	21.21
Častice T	1	1.52
Predložky E	2	3.03
Podstatné mená S	14	21.21
Príslovky D	1	1.52
R	2	3.03
Prídavné mená A	6	9.09

Tretia tabuľka má obdobný výzor ako predchádzajúca tabuľka zobrazujúca slovné druhy. Rozdielom však je, že táto tabuľka zobrazuje počty jednotlivých typov identifikovaných pomenovaných entít v texte. V prvom stĺpci tabuľky sa nachádza typ identifikovanej pomenovaj entity a nasledujúce dva stĺpce sú rovnaké ako v predchádzajúcej tabuľke.

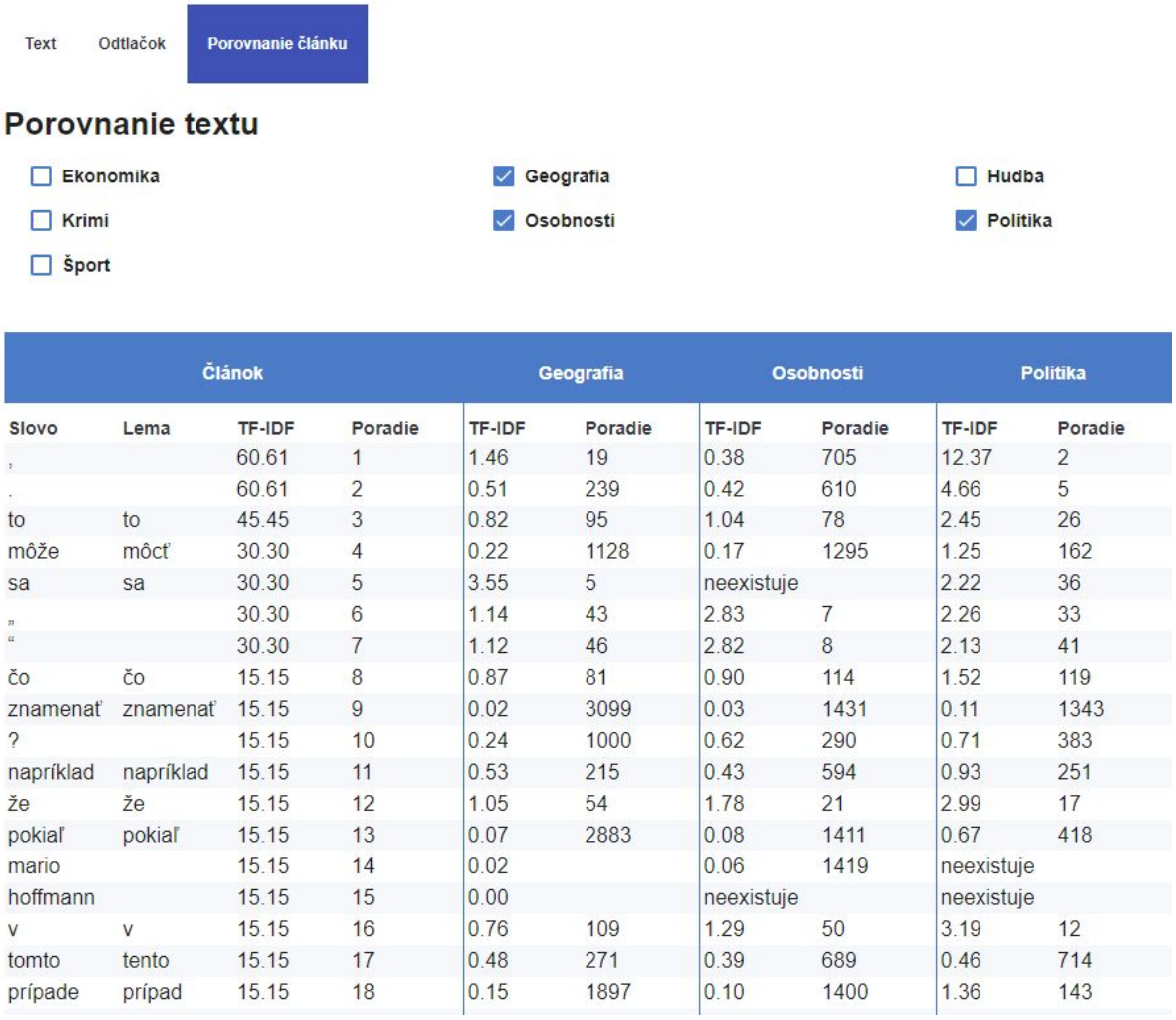
Záložka Porovnanie článku

Záložka Porovnanie článku slúži na porovnanie článku s dostupnými korpusmi pri iniciálnom zobrazení obsahuje stĺpce so slovom v článku, jeho lemou, hodnotou TF-IDF a poradovým číslom slova podľa hodnoty TF-IDF.



Článok			
Slovo	Lema	TF-IDF	Poradie
,		60.61	1
.		60.61	2
to	to	45.45	3
môže	môcť	30.30	4
sa	sa	30.30	5
„		30.30	6
“		30.30	7
čo	čo	15.15	8
znamenať	znamenať	15.15	9
?		15.15	10
napríklad	napríklad	15.15	11
že	že	15.15	12
pokiaľ	pokiaľ	15.15	13
mario		15.15	14
hoffmann		15.15	15
v	v	15.15	16
tomto	tento	15.15	17
prípade	prípad	15.15	18
mohol	môcť	15.15	19

Nad tabuľkou sa nachádzajú zaškrtávacie políčka s názvami korpusov. Pri zaškrtnutí korpusu sa do tabuľky pridajú dva stĺpce pre každý zaškrtnutý korpus - hodnota TF-IDF daného slova v príslušnou korpuse spolu s jeho poradím na základe hodnoty TF-IDF vrámci zvoleného korpusu.



Článok				Geografia		Osobnosti		Politika	
Slovo	Lema	TF-IDF	Poradie	TF-IDF	Poradie	TF-IDF	Poradie	TF-IDF	Poradie
,		60.61	1	1.46	19	0.38	705	12.37	2
.		60.61	2	0.51	239	0.42	610	4.66	5
to	to	45.45	3	0.82	95	1.04	78	2.45	26
môže	môcť	30.30	4	0.22	1128	0.17	1295	1.25	162
sa	sa	30.30	5	3.55	5	neexistuje		2.22	36
„		30.30	6	1.14	43	2.83	7	2.26	33
“		30.30	7	1.12	46	2.82	8	2.13	41
čo	čo	15.15	8	0.87	81	0.90	114	1.52	119
znamenať	znamenať	15.15	9	0.02	3099	0.03	1431	0.11	1343
?		15.15	10	0.24	1000	0.62	290	0.71	383
napríklad	napríklad	15.15	11	0.53	215	0.43	594	0.93	251
že	že	15.15	12	1.05	54	1.78	21	2.99	17
pokiaľ	pokiaľ	15.15	13	0.07	2883	0.08	1411	0.67	418
mario		15.15	14	0.02		0.06	1419	neexistuje	
hoffmann		15.15	15	0.00		neexistuje		neexistuje	
v	v	15.15	16	0.76	109	1.29	50	3.19	12
tomto	tento	15.15	17	0.48	271	0.39	689	0.46	714
prípade	prípad	15.15	18	0.15	1897	0.10	1400	1.36	143

Pod tabuľkou sa taktiež zobrazia percentuálne podobnosti analyzovaného textu so zvolenými korpusmi. Tieto percentuálne podobnosti majú rovnaký význam ako v časti Záložka Text.

stylom	styl	15.15	48	0.05		0.06	1421	neexistuje	
skutok	skutok	15.15	49	neexistuje		neexistuje		0.03	1365
nestal	nestať	15.15	50	neexistuje		neexistuje		0.02	1366
nájdete	nájsť	15.15	51	0.05	2996	0.02	1432	0.02	1366
na	na	15.15	52	0.58	176	1.56	33	1.87	71
konci	koniec	15.15	53	0.33	545	0.18	1274	0.16	1304
komentára	komentár	15.15	54	neexistuje		neexistuje		neexistuje	

Články

Modul aplikácie s názvom Články možno rozdeliť do dvoch pomyselných častí, ktorými sú Zoznam článkov a Detail článku, ktorý obsahuje niekoľko záložiek. Po kliknutí na záložku "Články" v hornom navigačnom menu sa používateľovi zobrazí už spomínaný zoznam článkov.

Zoznam článkov

V tejto časti sa nachádza zoznam všetkých dostupných článkov v systéme. Každý riadok v zozname zodpovedá jednému článku a nachádzajú sa v ňom nadpis článku a korpus, v ktorom sa článok nachádza. Na konci riadku sa nachádza tlačidlo "Detail", ktoré umožňuje presmerovanie na Detail zvoleného článku.

Zoznam článkov je možné abecedne zoradiť, a to kliknutím na hlavičku príslušného stĺpca v tabuľke, podľa ktorého chceme zoznam zoradiť. Opakovaným kliknutím je možné prepínať medzi zoradením vzostupne a zostupne.

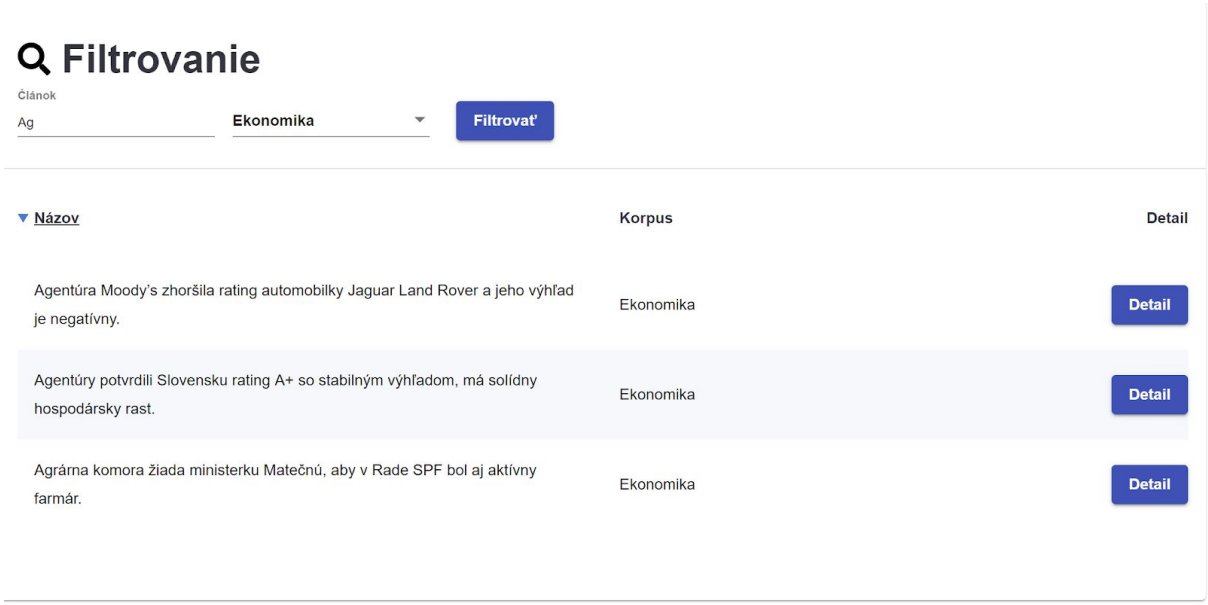
Nad zoznamom sa nachádzajú dve filtračné polia:

- **Názov** - Textové pole pre filtrovanie článkov podľa nadpisu. Na filtrovanie podľa názvu nie je potrebné zadať celý nadpis článku, keďže filter filtruje podľa začiatočných znakov (napr. pri vložení písmena "A" sa zobrazia články, ktorých názov sa začína na písmeno "A".)
  Je potrebné uviesť, že filter je citlivý na veľkosť písmen (tzv. case-sensitive), to znamená že rozlišuje malé a veľké písmená v názvoch.

- **Korpus** - filtrovať na základe korpusu je možné rozkliknutím rozbaľovacieho menu a následným výberom žiadaného korpusu.

Alternatívne je možné kliknúť na názov korpusu v akomkoľvek riadku z tabuľky a tým filter nastaviť na daný korpus.

Filtrovanie zoznamu sa spustí kliknutím na tlačidlo “Filtrovať”, alebo stlačením klávesy Enter.

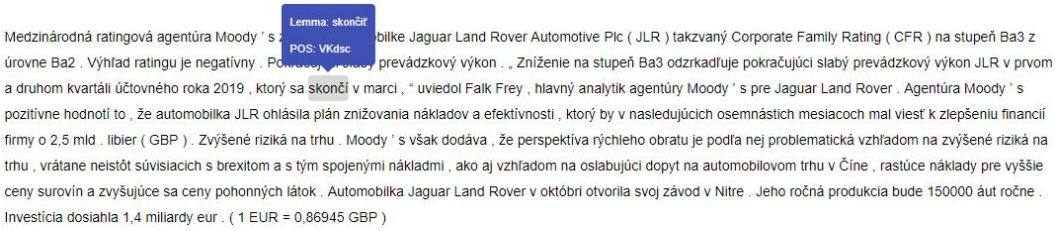


Detail článku

Záložka Index

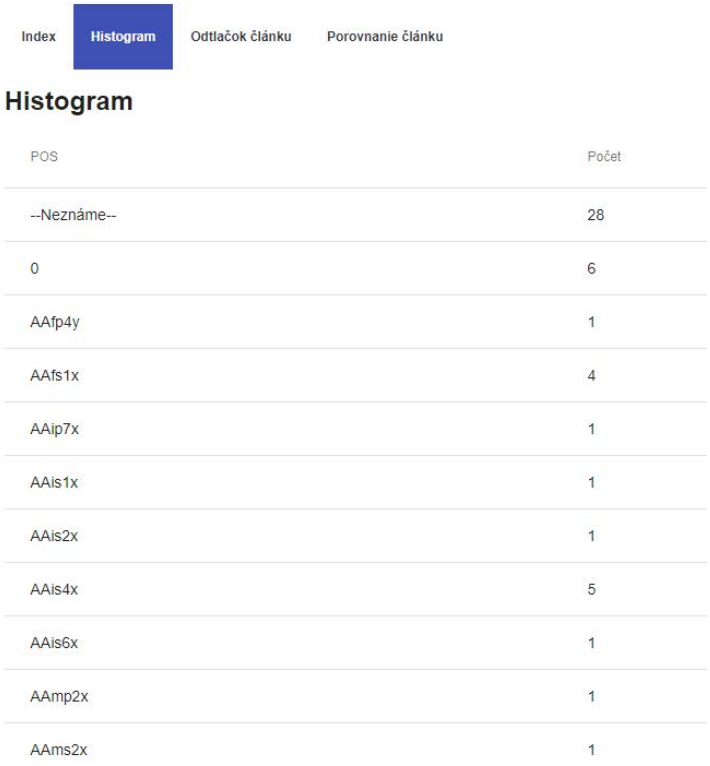
V záložke Index sa vo vrchnej časti zobrazuje nadpis vybraného článku. Pod nadpisom sa zobrazujú informácie o dátume vytvorenia, anotovania a indexovania článku. Na pravej strane pod nadpisom sa nachádza odkaz na Invertovaný index slov pre tento článok. Pod nadpisom a dátumami sa nachádza samotný text vybraného článku.

Po ukázaní myšou na ľubovoľné slovo sa nad ním zobrazia dodatočné informácie, akými sú POS tagy, léma, NER atď (ak sú tieto informácie dostupné).



Záložka Histogram

V záložke histogram má možnosť vidieť tabuľku početností jednotlivých POS tagov všetkých slov v článku. Tabuľka obsahuje iba dva stĺpce, a to konkrétny POS tag a počet, koľkokrát sa určitý POS tag v článku vyskytuje.



Záložka Odtlačok článku

Záložka Odtlačok článku v článkovom module funguje rovnako ako Záložka Odtlačok v časti Živá analýza.

Záložka Porovnanie článku

Záložka Porovnanie článku v článkovom module funguje rovnako ako Záložka Porovnanie článku v časti Živá analýza.

Invertovaný index

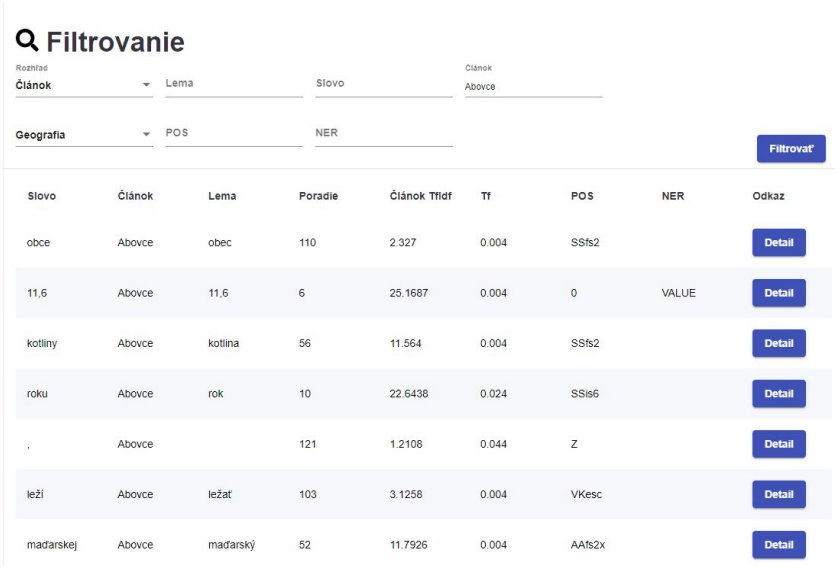
Obrazovka Invertovaný index zobrazuje tabuľku so všetkými dostupnými relevantnými informáciami o slovách, akými sú:

- Léma
- Korpus/Článok, v ktorom sa slovo nachádza - *zobrazenie závislé od zvoleného nastavenia rozhľadu*
- NER
- POS
- Hodnota TF (term frequency)
- Celok/Korpus/Článok TF-IDF - *zobrazenie závislé od zvoleného nastavenia rozhľadu*
- Poradie - poradie slova na základe hodnoty TF-IDF - *zobrazenie závislé od zvoleného nastavenia rozhľadu*
- Odkaz na článok, v ktorom sa slovo nachádza - *zobrazenie závislé od zvoleného nastavenia rozhľadu*

Záznamy v tabuľke je možné filtrovať pomocou filtračných polí umiestnených nad tabuľkou. Patria medzi ne:

- **Rozhľad** - rozhoduje o tom, podľa akého kritéria sú záznamy v tabuľke zoskupené, pri čom používateľ má v rozbaľovacom menu na výber 3 možnosti: článok, korpus alebo celý dataset. Nastavenie filtra tiež ovplyvňuje zobrazenie relevantných stĺpcov v tabuľke.
- **Léma** - textové pole slúžiace na filtrovanie na základe lémy slova, pri čom je potrebné zadať celú lému slova (nie iba časť)
  Tento filter je možné nastaviť aj kliknutím na lému jedného zo záznamov v tabuľke.
- **Slovo** - textové pole slúžiace na filtrovanie na základe slova, pri čom je potrebné zadať celé slovo (nie iba časť)
- **Článok** - textové pole slúžiace na filtrovanie na základe nadpisu článku. V tomto filtri je postačujúce zadať iba začiatok nadpisu a filter následne zobrazí záznamy pre všetky články, ktoré sa začínajú zadaným textovým reťazcom.
  Tento filter je možné nastaviť aj kliknutím na nadpis článku jedného zo záznamov v tabuľke.
- **Korpus** - rozbaľovacie menu slúžiace na filtrovanie podľa korpusu
- **POS** - textové pole slúžiace na filtrovanie podľa POS tagu slova. V tomto filtri je postačujúce zadať iba začiatok POS tagu a filter následne zobrazí všetky slová, ktorých hodnota POS sa začína zadaným textovým reťazcom.
- **NER** - textové pole slúžiace na filtrovanie podľa NER daného slova, pričom je potrebné zadať celé a presné znenie hodnoty NER (nie iba časť slova)

Filtrovanie zoznamu sa spustí kliknutím na tlačidlo “Filtrovať”, alebo stlačením klávesy Enter.



Hra s vetami

Aplikácia obsahuje aj interaktívnu hru, v ktorej má používateľ za úlohu identifikovať do ktorého z korpusov patrí zobrazená veta. Veta je vybraná náhodne zo všetkých článkov. Hra prebieha tak, že používateľ začína s iniciálnym rozpočtom 100 virtuálnych peňazí. Z jeho rozpočtu môže následne rozdeliť medzi korpusy jednotlivé čiastky podľa svojho uváženia tak, aby trafil korpus, v ktorom sa zobrazená veta nachádza. V prípade úspešného tipu sa používateľovi priráta čiastka do rozpočtu a zobrazí sa nová veta. Pri neúspešnom pokuse sa naopak čiastka odráta. Hra sa končí, keď hráč už nemá dostupné žiadne prostriedky.

Na vrchnej časti pomyselného hracieho stola je zobrazený aktuálny rozpočet hráča a pod ním je zobrazená aktuálna veta. Čiastky môže používateľ prerozdeľovať pomocou znakov plus (+) a mínus (-), ktoré sú umiestnené pod názvom jednotlivých dostupných korpusov. Svoj tip a stávku používateľ potvrdí stlačením tlačidla s textom “Vyhodnoť !”.