Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií

Projektová dokumentácia

Tím 05: TxTEnv

Akademický rok 2018/2019
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 prvých 3. šprintoch. V úvode dokumentu sú bližšie predstavení členovia tímu spolu s rolami (oblasťami), ktorým sa v rámci tímového projektu venujú. Ďalšia časti 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 práci 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 dokumentáciou, dokumentácií stretnutí, dokumentácií 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 o 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 1: 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					100%			
Metodiky	20%			5%	10%	20%	45%	
Zápisnice zo stretnutí	65%	5%	5%	5%			5%	15%
Dokumentácia k inžinierskemu dielu			16%	16%	5%	16%	31%	16%

2. Aplikácie manažmentov

Manažment 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 vlákna 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

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)

Dôležitou časťou tímovej práce je výmena (zdieľanie) dát a informácií. Dáta zahŕňajú najmä zdrojové kódy, ktoré sú zdieľané formou GitHub-u. 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

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

Po spoločnej konzultácií 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

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 kvality

Manažment kvality je zameraný najmä na kvalitu zdrojového kódu a jednotlivých výstupných dokumentov.. Pre písanie kódu boli vytvorené metodiky, ktoré sú dodržiavané počas implementácie. Po dokončení implementácie je tiež potrebný Pull request, kde určitý člen tímu skontroľuje prácu svojho kolegu (viď Manažment prehliadok kódu). Až po úprave zmien a pripomienok je pull request schválený a funkcionalita je prenesená do hlavnej vetvy programu, čím je tiež úloha považovaná za ukončenú. Pre zachovanie konvencie písania zdrojového kódu sú implementované nástroje pre kontrolu zdrojového kódu, ktorý nabáda vývojárov k dodržiavaniu určitých pravidiel, aby kód zachovával určitú štruktúru. Pri výstupných dokumentoch si vždy dokument prečíta ďalšia osoba, ktorá dokument upraví, či už po obsahovej alebo formátovej stránke. Až po takejto kontrole sa dokument považuje za dokončený.

3. Prehľad šprintov

Počas 9. týždňov semestra sme zatiaľ absolvovali 3 šprinty. V aktuálnom období momentálne prebieha 4. šprint, ktorý bude ukončený 28.11.2018.

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 obnášalo 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.

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 9 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 evidencií 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 neznalosti 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.

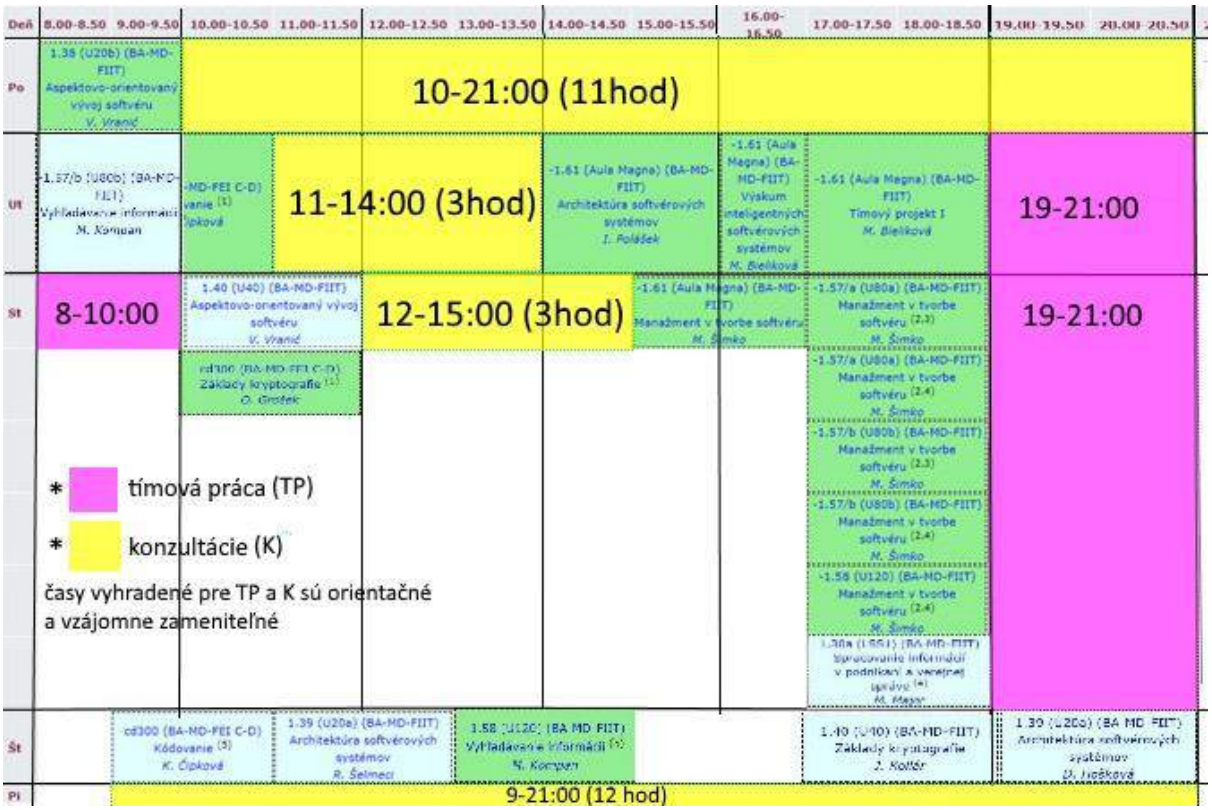
Príloha A: Preferencia tém

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 B: 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.



Obrázok č. 1: Rozvrh členov tímu s možnosťou

6. Metodiky

6.1. Metodika komunikácie

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

Sa opakujú pravidelne každý týždeň a 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 vetvy sú:

- **General:** určené pre všeobecné informácie a požiadavky, v rámci ktorého 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

Manažment úloh má na starosti Scrum-master. Na začiatku každého šprintu, sú definované požiadavky product ownera na funkcionalitu a zmeny aplikácie. Tieto požiadavky sa častokrát týkajú rôznych častí aplikácie (frontend, backend) a tiež rôznych oblastí. 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.
Odhad hodnôt časového rozsahu, náročnosť úlohy a jej priority prebieha pri definovaní požiadavky Scrum-masterom, kedy sa určujú aj samotní koordinátori. O týchto hodnotách sa hlasuje na stretnutí. Po tom ako je definovaný časový rozsah pre konkrétnu user-story, definujú sa základné úlohy, ktoré z požiadavky vyplývajú.
Nasleduje definovanie pravidiel pre DoR (definition of ready) a DoD (definition of done). DoR zadefinujú členovia tímu spolu s product ownerom.
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 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 podúloh vyplývajúc z danej úlohy
- Odhad času, náročnosti a priority úlohy
- Voľba zodpovednej osoby za úlohu
- Rozdelenie podúhol 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

6.3. Metodika prehliadok kódu

Prehliadky kódu prebiehajú pomocou dvoch znamych metód. Prvou metódou je bežná prehliadka kódu cez plece, kedy seniornejší programátor z tímu dohliada nad kódom juniornejšieho programátora a upozorňuje ho na chyby a konvencie, ktoré treba dodržiavať. Toto nie len že šetrí čas, ale zároveň poskytne juniornejšiemu 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)
- Požiadavanie o pull-request
- Prehliadka kódu posudzovateľom
- Vyhodnotenie prehliadky kódu
- Zapracovanie pripomienok
- Opätnová prehliadka
- Označenie úlohy za vyriešenú
- Zlúčenie časti funkcionality s hlavnou vetvou

6.4. Metodika verziovania

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 vrstvou (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 vrstvou
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. Metodika Definition of ready

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:

1. User story musí byť malá (malá v zmysle realizácie úlohy v danom šprinte)
2. User story musí byť nezávislá od iných User stories
3. User story musí byť spísaná a sformulovaná product ownerom. Musí byť rozvinutá natoľko, aby aj s odstupom času vykazovala vysokú informačnú hodnotu o náplni danej User story pre každého člena tímu.
4. User story musí mať zadefinované akceptačné kritériá (pokiaľ nie sú jednoznačne zrejmé z opisu danej úlohy)
5. User story musí byť ohodnotená Story points každým členom tímu, priemer prípadne diskusia stanoví výslednú hodnotu User story
6. Pre každú úlohu sa zadefinujú výstupné dokumenty, bez ktorých nebude môcť byť User story akceptovaná
7. Určí sa zodpovedná osoba, ktorá dohliada na vykonanie všetkých potrebných úkonov pre vyriešenie danej User story
8. Určí sa osoba, ktorá bude zodpovedná za code review po implementácií (Nesmie to byť zodpovedná osoba za User story)
9. Pre implementovanú funkcionalitu sa navrhnú testy/testovacie scenáre, prípadne sa vyhodnotí nutnosť či spôsob ich realizácie

6.6. Metodika Definition of done

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. Spĺňa všetky kritériá, ktoré boli zadefinované pri tvorbe danej US
2. Obsahuje náležitosti, ktoré boli definované pri tvorbe US (dokumentácie , diagramy, ..)
3. Všetky vytvorené úlohy/chyby pod touto US musia byť v stave “Closed”
4. V prípade implementovanej funkčnej časti systému musí prejsť testami / byť otestovaná nezávislým členom tímu v závislosti od charakteru US
5. 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”.
6. Stav “Resolved” mení člen tímu zodpovedný za danú US po splnení všetkých predošlých kritérií.

6.7. Metodika dokumentácia

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 vytáraným 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 zípisnice 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 zapájal

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>
```

Obrázok č. 2: Pravidlá pre písanie dokumentácie

Biznis dokumentácia:

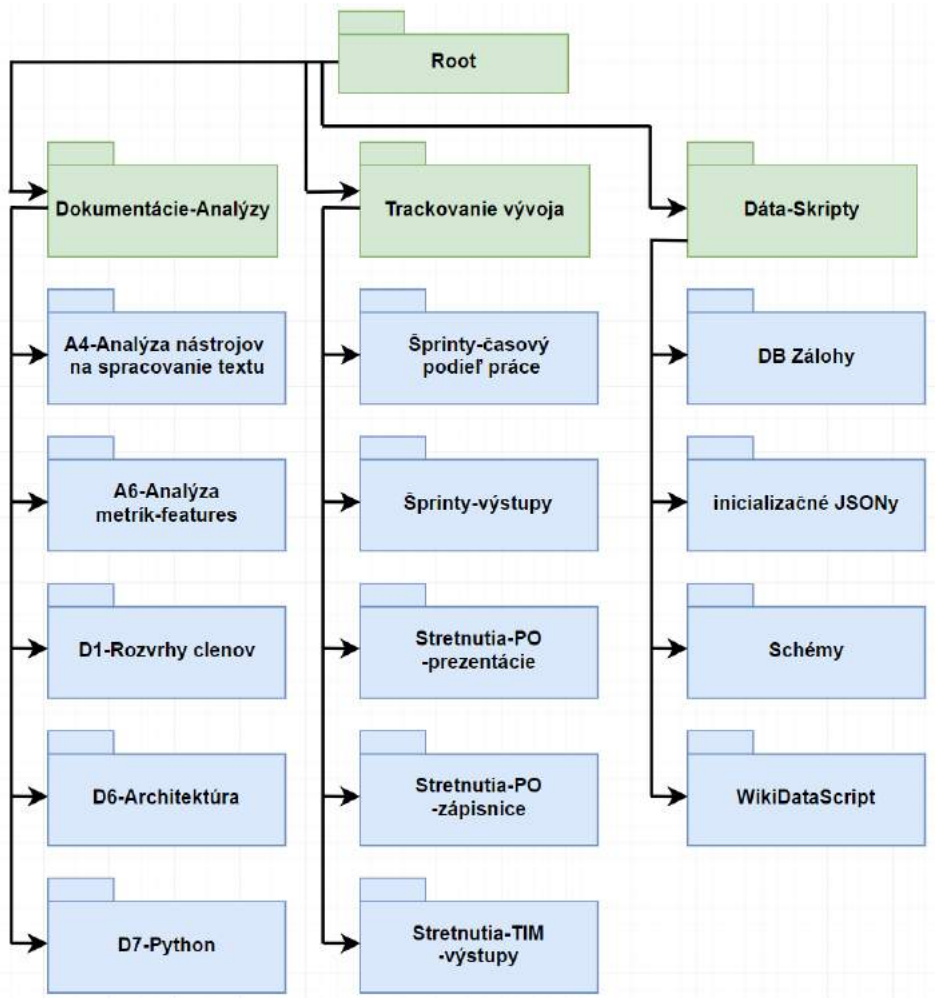
Do biznis dokumentácie môžeme zaradiť také dokumenty, ktoré majú istým spôsobom opisovať /prezentovať navrhované riešenie s marketingovým účelom, napríklad kontrakt medzi zadávateľom a vyhotoviteľom, zoznam požiadaviek, manažment plánovania, prezentácia riešenia a iné, ktoré bude vyhodnocovať potenciálny klient alebo zadávateľ. Súčasťou biznis dokumentácie môže byť v prípade agilného vývoja napríklad aj export z product backlogu (PB). Ukážku PB môžeme viedieť v nasledujúcej sekcií "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 dohľadanie 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 dohľadateľné v prípade potreby. Dokumentácia a iné náležitosti sú uchovávané v zdieľanom internetovom adresári od spoločnosti Google - Google Drive.

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



Obrázok č. 3: Organizácia dokumentov

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

Priečinok "**Trackovanie vývoja**" je štrukturovaný 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 strenutí 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.

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	

Obrázok č. 4: Prehľad úloh 1. šprintu

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	

Obrázok č. 5: Prehľad úloh 2. šprintu

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	

Obrázok č. 5: Prehľad úloh 3. šprintu

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 verzií 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

Dokumentácia k inžinierskemu 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 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 sme sa zamerali najmä na využitie technológií NodeJS, Python, MongoDB.

1.1. 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.

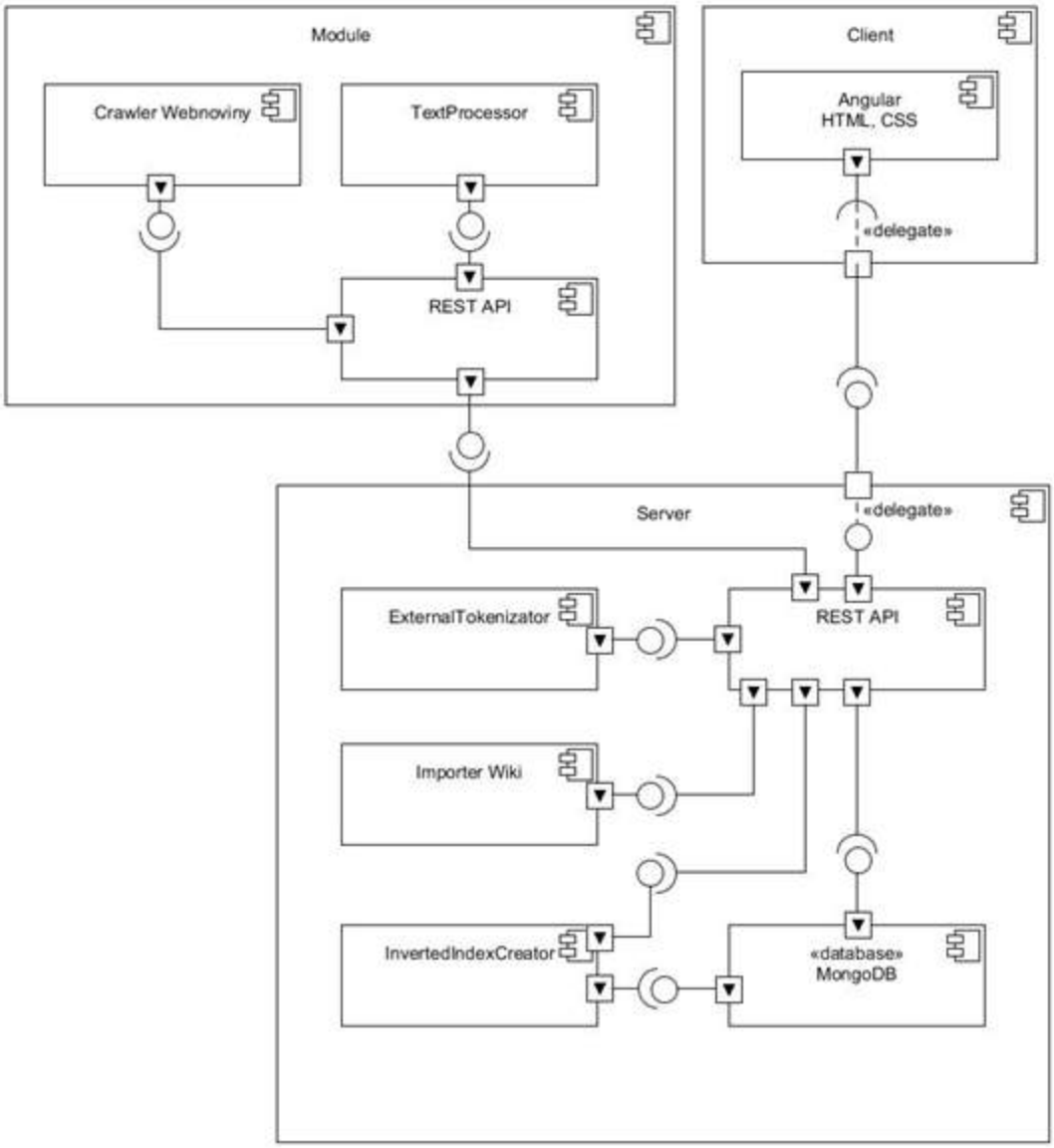
2. Celkový pohľad na systém

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.



Obrázok č. 1: 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

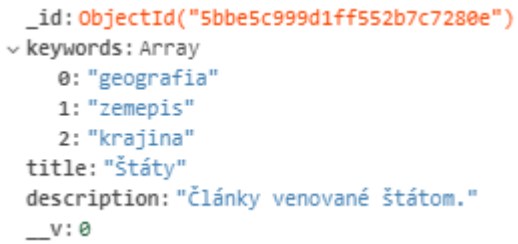
spracovanie textu a strojové učenie. Python sme zvolili vzhľadom k veľkému množstvu knižníc v oblasti a predošlým skúsenostiam tímu.
Klientsku časť front-endu zastrešuje rámec Angular. Hlavným dôvodom tejto voľby bolo zníženie požiadaviek na server, ale aj zrýchlenie zobrazovania prezenčnej vrstvy používateľovi,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



Obrázok č. 2: Náhľad korpusu

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 ja 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.

Obrázok č. 3: Náhšľd existujúceho článku

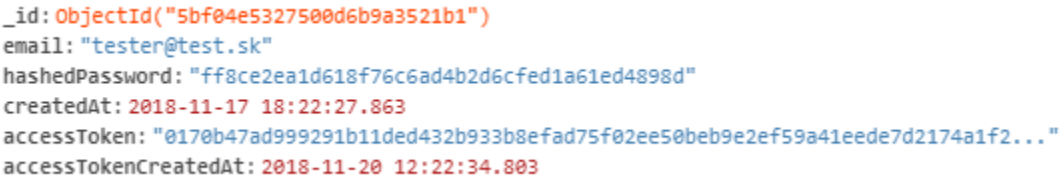
Invertovaný Index

Kolekciu s invertovaným indexom používame na zoskupenie a vytvorenie skupín pre slová používané vo všetkých článkoch. Táto kolekcia obsahuje slovo, ktoré je kľúčom invertovaného indexu. Ďalej obsahuje lemu slova, frekvenciu používania výrazu a jeho NER, či POS značky. Pole článkov v tomto dokumente nesie informácie o jednotlivých článkoch pre toto slovo. Zachytáva n-gramy, pozície a počet výskytov slova v článku, tiež ale aj názov a identifikátor článku. Na nasledujúcom obrázku je zachytený dokument invertovaného indexu.

Obrázok č. 3: Dokument invertovaného indexu

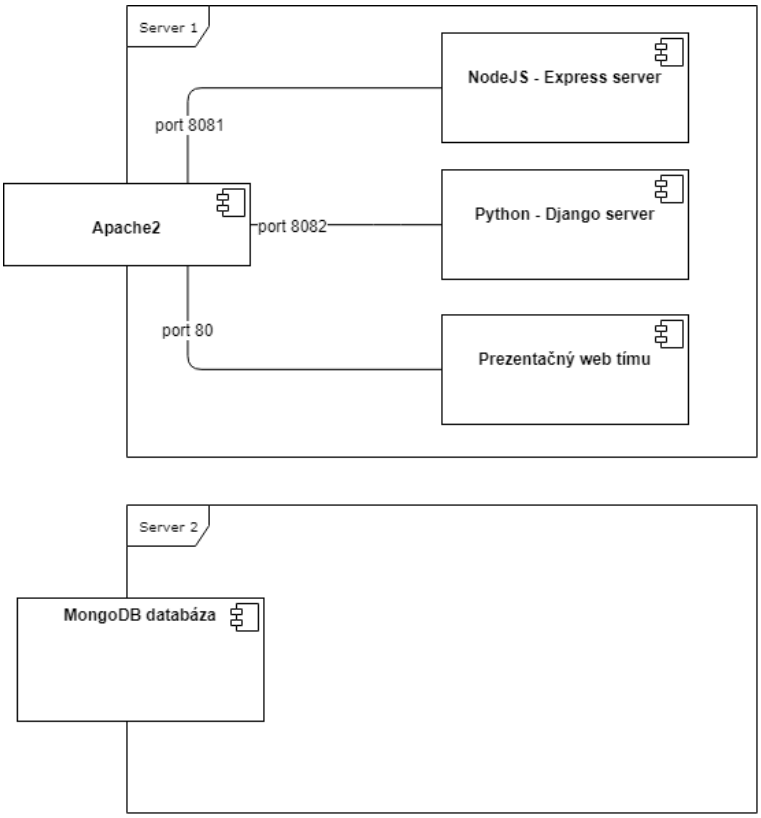
Používatelia

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



Obrázok č. 4: Dokument s používateľom

2.3. Backend systému



Obrázok č. 5: Schéma backendu systému

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.

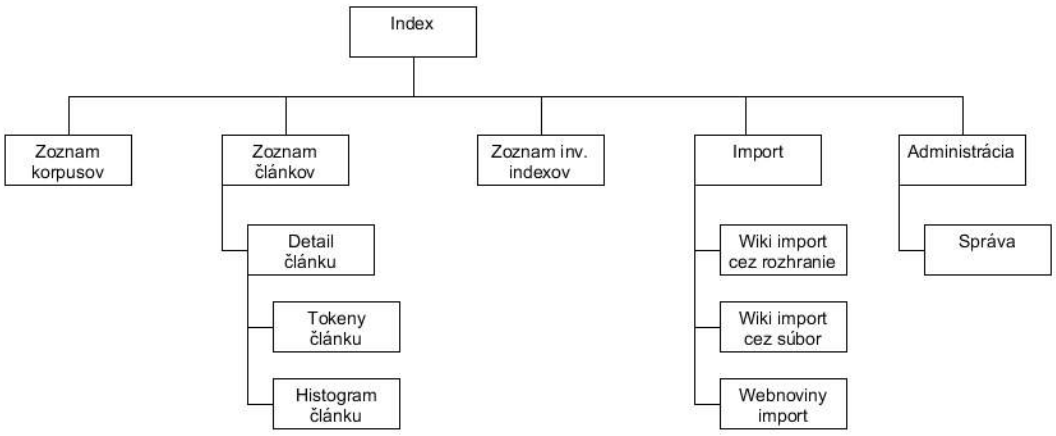
Niektoré služby volajú funkcionalitu, ktorá by nemala byť voľne dostupná pre používateľov aplikácie a preto je jej použitie obmedzené. Pre použitie týchto aplikácií je potrebné mať prístupový

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 oprávnených používateľom

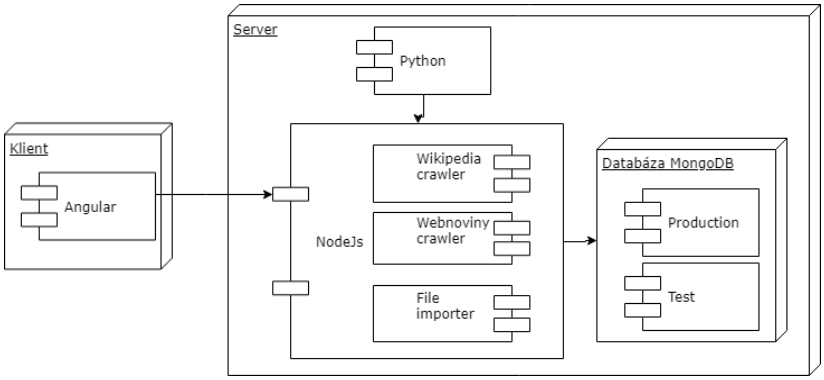


Obrázok č. 6: Rozhrania systému

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



Obrázok č. 7: Schéma 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

Obrázok č. 8: Ukážka rozhrania importu článku

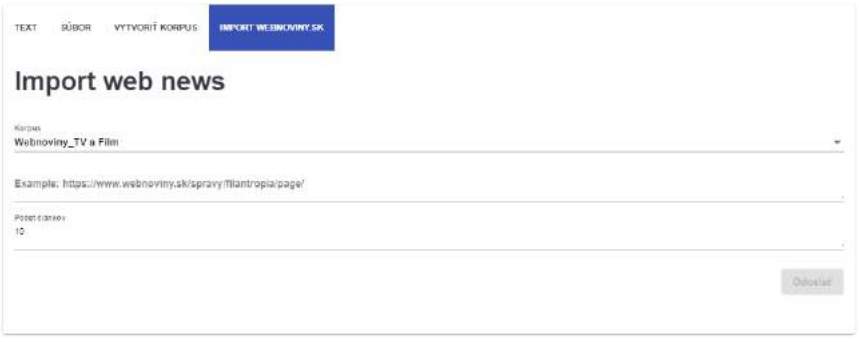
Táto časť modulu je implementovaná priamo v backendovej časti systému(nodejs). Výsledkom vykonaného procesu je uložený článok v databáze v danom korpuse článkov.

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

Obrázok č. 9: Ukážka rozhrania importu zo súboru

Rozširujúce 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



Obrázok č. 10: Ukážka rozhrania importu obmedzeného počtu článkov

Posledná časť reprezentuje crawler implementovaný v jazyku Python. Volanie tejto funkcionality je zebezpečené cez REST Django, ktoré zabezpečuje komunikáciu medzi Node.js a Python. Nakoľko sa jedná o slovenský portál, ktorý má zebezpečenia 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.

3.2. 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.

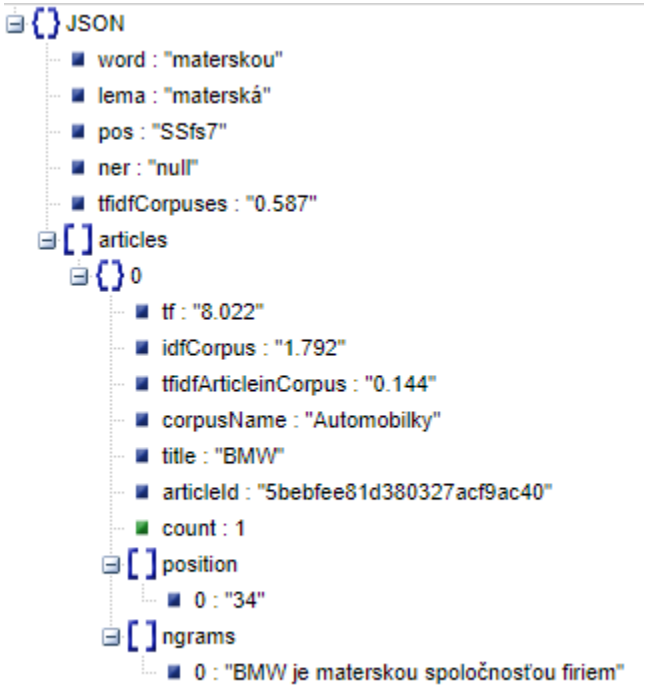
Vstupom pre tento modul je ucelený text článku. Tento text článku sa spracuje po sekciách a nad jednotlivými sekciami textu sa volá API nlp6 tokenizátor. Parametre na používanie tokenizátora sú pevne definované v konfigu. Konkrétne sa jedná o prístupový API kľúč, modul POS značkovania (ProbabilisticPOSTagger)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

V našej implementácii sa dá vytvoriť invertovaný index nad konkrétnym korpusom článkov už uložených v databáze. Pomocou volania služby, ktorá dostane ako parameter názov korpusu vykonáme tvorbu invertovaného indexu nad daným korpusom. Na nasledujúcom obrázku môžeme vidieť konkrétnu schému použitú pre uchovávanie invertovaného indexu v našej databáze.



Obrázok č. 11: Invertovaný index

Konkrétna implementácia je nasledovná. Volanie API spustí dopyt po všetkých článkoch daného korpusu. Pre dané články si musíme získať tiež ich text. Postupne prechádzame text článkov po slovách. Prechádzané slovo pre nás predstavuje potenciálny prvok do kolekcie databázy invertovaný index. Pre každé slovo vytvoríme JSON objekt identický so schémou samotnej databázy. Prechádzané slovo zaznamenávame nasledovne: pre nové slovo (ešte neexistujúce lokálne) vytvoríme úplne nový objekt, pre slovo existujúce ale v inom článku priradime nový podobjekt "výskyt v článku", pre existujúce slovo aj z rovnakého článku vykonáme update nad podobjektom "výskyt v článku", kde upravíme početnosť, ngram-y a pozície.

Po prejdení všetkých slov a textov máme lokálny JSON objekt obsahujúci invertovaný index pre konkrétny korpus. Online v databáze sa nachádza dovtedy vytvorený invertovaný index. Vykonáme dopyt pre kompletný online invertovaný index. Lokálne porovnávame stavy našej schémy a tej remote. Insert, alebo update nad databázou vykonáme len v prípade odlišností. Takýto prístup spája výhody batch prístupu a tiež odľahčuje dopyty nad databázou.

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 alebo kolekcii. 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 = vyskyt 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(všetký 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: **TFIDFcorpus = tf *IDFcorpus**
- 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)**
- finálny výpočet je nasledovný: **TFIDFcollection = avg(všetkých tf hodnôt pojmu X)*IDFcollection**, kde využitie priemeru **avg()** možná nevracia najpresnejšie hodnoty, ale hodnoty sú pomerne presné a pre náš účel vhodné.

4. Dokumentácia k API

Kapitola bližšie popisuje jednotlivé funkcie, ktoré API poskytuje. API je založená na metóde REST, kde sa pomocou URL adresy cez klientské 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
URL: *<názov-stránky>/api/articles*
Vstupná hodnota: -
Prístup: bez tokenu
Metóda: GET
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

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)*

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)*

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 pre konkrétny článok

Získanie zoznamu korpusov

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

Získanie konkrétneho korpusu

Popis: Získanie konkrétneho korpusu
URL: *<názov-stránky>/api/corpuses*
Vstupná hodnota: názov korpusu *(napr.: Automobily)*
Prístup: bez tokenu
Metóda: GET
Návratová hodnota: 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 indexov v1

Popis: Získanie všetkých invertovaných indexov vo formáte 1 inv.index : n článok
URL: *<názov-stránky>/api/inverted-indices*
Vstupná hodnota: -
Prístup: bez tokenu
Metóda: GET
Návratová hodnota: zoznam invertovaných indexov

Získanie zoznamu invertovaných indexov v2

Popis: Získanie všetkých invertovaných indexov vo formáte 1 inv. index : 1 článok
URL: *<názov-stránky>/api/inverted-indices/get-unwinded*
Vstupná hodnota: -
Prístup: bez tokenu
Metóda: GET
Návratová hodnota: zoznam invertovaných indexov

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 korpus

Popis: Vytvorenie invertovaných indexov pre všetky články v danom korpuse a následné uloženie do databázy
URL: *<názov-stránky>/api/inverted-indices/tfidf=corpus/:corpusName*
Vstupná hodnota: názov korpusu *(napr.: Automobily)*
Prístup: bez tokenu
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)*

5. Záver

V prvých 3. šprintoch semestra sme sa venovali globálnej analýze oblasti problemtiky. 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, …).
Úvodné 3 špritny hodnotíme pozitívne, odviedli sme podstatný kus práce, ktorý náš projekt vyžaduje. Nadobudli sme množstvo nových znalstí a poznatkov s použitím (pre väčšinu členov tímu) nových technológií.