Slovenská technická univerzita

Fakulta informatiky a informačných technológií

Ilkovičova 3, 812 19 Bratislava

Analýza výsledkov výskumu

Dokumentácia k inžinierskemu dielu

Tím 10 ResearchRank
Vedúci projektu: Ing. Nadežda Andrejčíková, PhD.
Členovia tímu: Bc. Michael Gloger
Bc. Rastislav Kostrab
Bc. Šimon Kompas
Bc. Tomáš Jánošík
Bc. Daniel Kĺč
Bc. Stanislav Kubica
Bc. Michal Walder

1 Úvod

Tento dokument slúži ako dokumentácia k tímovému projektu s názvom Analýza výsledkov výskumu. Ide o systém, ktorý slúži na správu publikačnej činnosti v oblasti vedy a výskumu. Systém spracúva, analyzuje a uchováva záznamy o vedeckej publikačnej činnosti. Pomáha vedeckým pracovníkom sledovať svoje publikácie a okrem ohlasov, odhaľuje aj iné zaujímavé vzťahy medzi nimi.

1.1 Štruktúra dokumentu

Dokument je štruktúrovaný nasledovne:

Prvá kapitola Analýza problematiky obsahuje analýzu problémovej domény. V druhej kapitole Špecifikácia požiadaviek sme zhrnuli požiadavky na prvotný systém. Tretia kapitola Návrh obsahuje opis technológií, ktoré sme sa rozhodli použiť na riešenie jednotlivých problémov, spojených s našim projektom. Štvrtá kapitola Implementácia obsahuje popis riešenia jednotlivých problémov a v závere dokumentu sú opísané naše šprinty.

Vývoj v tíme je riadený agilnou metódou SCRUM a tomu bol prispôsobený aj vývoj dokumentu. Dokument bude aktualizovaný postupne ako bude prebiehať naša práca.

1.2 Popis problému

Vedecké pracoviská fungujú z veľkej časti na základe príjmov z grantov, na ktoré prispieva ministerstvo školstva SR. Ak pracoviská chcú získavať tieto granty, musia sa usilovať o výskumnú činnosť a výsledky z nej publikovať. Výsledky výskumnej činnosti sa okrem vedeckých digitálnych knižníc evidujú v našej krajine do Centrálneho registra publikačnej činnosti (CREPČ). Jedným zo základných kvalitatívnych meradiel výsledkov výskumnej činnosti je počet ohlasov na publikované diela, čo znamená, že časť daného diela bola citovaná v inom novom diele, ktoré bolo publikované v rámci rovnakej výskumnej domény.

Získavanie prehľadu o ohlasoch na autorove vlastné diela je veľmi ťažkou a zdĺhavou manuálnou činnosťou, ktorá by sa mala dať zautomatizovať vytvorením systému, ktorý bude informácie o týchto dielach spracovávať automaticky a bude tak šetriť čas autorom publikovaných diel. Pri manuálnom vyhľadávaní autori diel taktiež robia chyby, ktoré vznikajú napríklad nechcenom prehliadnutí iného diela a podobne. Pre obohatenie databázy týchto ohlasov je potrebné ju naplniť dátami z nových zdrojov a identifikovať nové prepojenia za pomoci porovnávacích algoritmov. Tomuto kroku však predchádza ešte jeden ďalší fakt, že tieto diela môžu byť v CREPČ zadané chybne alebo v rôznych variantoch. Informácie o týchto dielach sa do CREPČ zadávajú manuálne a dôvodom občasnej chyby pri zadávaní je samozrejme ľudský faktor.

Získavanie informácií o výskumnej činnosti v našej krajine je teda veľmi zdĺhavý a zložitý proces.

1.3 Cieľ projektu

Cieľom nášho projektu je vytvoriť systém, ktorý bude automatizovane spracovávať informácie o vedeckej publikačnej činnosti. Dáta budú čerpané z rôznych zdrojov, medzi ktoré patrí napríklad: CREPČ, Google Scholar a iné. Výstupom z našej práce bude webový informačný systém rozšírený o webové služby, pomocou ktorých je možné dopytovať sa na služby, ktoré bude náš systém ponúkať

1.4 Použité skratky a pojmy

Pojem / skratka	Popis

CREPČ	Centrálny register evidencie publikačnej činnosti
GWT	Google Web Toolkit
ORM	Objektovo - relačný mapovač

2 Analýza problematiky

2.1 Problematika párovania publikácií

Na párovanie publikácií existuje množstvo článkov a odborných prác. Existujú prístupy, ktoré sa snažia vyvinúť unsupervised metódu bez nutnosti refaktorizácie celej databázy [1], prístupy, ktoré sa snažia párovať publikácie na základe vyššej štatistiky [2], na základe strojového učenia [3], na základe heuristík a rozšírenia existujúcich metód [4] a mnohé iné. Dodnes neexistuje algoritmus, ktorý by s určitosťou vedel spárovať všetky duplicitné publikácie iba podľa údajov uvedených v zdroji informácií.

2.1.1 Špecifikácia problému

Problém ako taký pozostáva z viacerých pod-problémov. Akýkoľvek algoritmus párovania musí počítať s neúplnými alebo nesprávnymi údajmi, polymorfiou mien, viac-jazyčnými čiastočnými prekladmi, podobnosťou nadväzujúcich prác. Následkom týchto faktorov sa pri párovaní používajú váhy na jednotlivé dostupné informácie.

Taktiež je mnoho problémov implicitne naviazaných na jazyk, v ktorom sú situované. Napríklad mená slovenských autorov majú iné chyby a skomoleniny, špecifické problémy a synonymá ako mená anglických autorov a podobne.

2.1.2 Používané prístupy

Vo všeobecnosti sa páruje podľa prvého autora a potom sa používajú iné dostupné informácie a určitou váhou na verifikáciu výsledku porovnania prvých autorov. Používajú sa co-autori, názov práce, abstrakty, ISBN, ISSN a iné informácie, avšak väčšinou iba niektoré z týchto sú dostupné a v prípade autorov je výsledok tiež podmienený nejakou pravdepodobnosťou zhody. Kvôli tomuto vznikli aj alternatívne metriky, ktoré hodnotia kontext práce, jej zameranie, zhodu zamerania co-autorov a pod. Je možné logicky rozdeliť techniky, ktoré sa používajú na techniky párovania mien, techniky porovnávania údajov a techniky porovnania kontextu.

2.1.2.1 Párovanie mien

Existuje množstvo prístupov avšak študované boli predovšetkým tie najpoužívanejšie. Menovite algoritmu Jaro-Winkler, Levensteinova vzdialenosť a Damerau-Levensteinova vzdialenosť, tieto porovnávajú v podstate aké veľké úsilie je potrebné na to, aby sa prvý člen porovnania zmenil na druhý člen. Ďalej sa preštudovali algoritmy, ktoré porovnávajú priamo podiel rovnakých sekvencií v reťazci. Tu boli preštudované aspekty 2-gramov a 3-gramov.

2.1.2.2 Techniky porovnávania údajov

Porovnávania údajov je vo všeobecnosti dobre preskúmaná oblasť, ktorou sa zaoberá množstvo publikácií. Pre nás boli relevantné predovšetkým techniky, ktoré boli priamo už niekým použité pri porovnávaní publikácií, alebo techniky, ktoré by mohli určité čiastkové problémy porovnávania vyriešiť. Predovšetkým problematický je nedostatok informácií.

Väčšina porovnávaných atribútov je tradične otázka identity alebo dostatočného prieniku dvoch množín údajov. Z tohto hľadiska sú zaujímavé predovšetkým porovnania plných textov a abstraktov, poprípade titulov prác. Tu sa často aplikuje heuristika, ktorá hovorí, že texty, resp. abstrakty alebo tituly musia byť identické na to, aby to bola rovnaká práca. Aj napriek tomu, že sa s tým vo všeobecnosti odborná komunita stotožňuje, problémy do tejto problematiky vnášajú rôzne súčasti, ktoré môžu byť k textu pridané zo zdroja. V tomto prípade i keď ide len malú súčasť, ktorá nemá s obsahom alebo formátom samotného textu nič spoločné, tak rozvráti túto vo všeobecnosti zaužívanú heuristiku. Existuje množstvo metód, ktoré sa na

obídenie tohto problému zameriavajú. Naše štúdium sa zameralo predovšetkým na metódy s de Bruijnovými grafmi a String grafmi. Tieto okrem porovnania samotných textov môžu podľa potreby poskytnúť veľa iných užitočných informácií.

2.1.2.3 Techniky porovnávania kontextu

Na porovnávanie kontextu človek potrebuje množstvo metadát. Na určenie kontextu práce treba zistiť zameranie práce, množinu autorov, s ktorými prvý autor bežne spolupracuje a samotný zdroj práce a jeho zameranie. Napríklad sa používa budovanie odborných slovníkov a porovnávanie s množinami iných autorov na určenie zamerania konkrétnej práce. Na budovanie odborného slovníka sa používajú napríklad spomínané de Bruijnové grafy alebo String grafy (a rôzne ich adaptácie). V nich po určení prieniku a vyčiarknutí najpoužívanejších slov sa vytvoria implicitne prepojené skupiny spojení špecifické pre danú oblasť, skupinu alebo indivíduum. Touto technikou, výberom a porovnaním situovania jednotlivých entít sa zistí kontext práce. Na tento proces však je potrebné množstvo informácií predovšetkým v oblasti kontextu práce, a preto je to prakticky nevyužiteľné pre autora, ktorý je úplne nový pre databázu, pretože o ňom nie je dosť informácií. Taktiež existujú pokusy na určenie kontextu zo samotných záznamov zdroja publikácie, pracoviska autorov a podobne. Vo všeobecnosti, aj keď je tento údaj veľmi užitočný, nie vždy je možné ho získať z dostupných informácií a preto je väčšinou chápaný existujúcimi prístupmi ako menej dôležitý ako predchádzajúce dva komponenty párovania.

2.1.2.4 Techniky výberu kandidátov na porovnanie

V zásade je nemožné porovnávať záznam označený na párovanie s celou databázou záznamov, pretože takéto porovnávanie by viedlo k zahlteniu systému a zbytočnému porovnávaniu takých záznamov, ktoré majú len málo spoločné. Oblasť vyberania kandidátov na porovnávanie je vcelku dobre zmapovaná, existujú dva hlavné prístupy, ktoré potom rôznymi obmenami dosahujú lepšie výsledky v určitých špecifických oblastiach. Jedným prístupom je tzv. blokový prístup (napríklad [6]), ktorý rozdelí celú množinu záznamov na bloky, v rámci ktorých potom záznamy porovnáva. Druhým prístupom je Sorted Neighbourhood Method [5], kde ide o to, že máme záznamy zoradené podľa niektorého poľa(často je to primárny kľúč alebo najdôležitejší záznam, ktorý je čo najviac jedinečný) a porovnávame len v rámci určitého okna záznamov.

2.1.3 Supervised vs. unsupervised metódy

Všetky techniky párovania podliehajú ešte jednému atribútu a to je supervised alebo unsupervised. Rozdiel je v tom, že prvé sú pred nasadením kalibrované na známej správnej množine údajov, aby sa určili ich parametre. Je to kvôli mnohým faktorom, ale predovšetkým pre to, že v rôznych jazykoch sa grafy efektivity hodnoty parametrov môžu líšiť. Naproti tomu unsupervised metóda je univerzálne navrhnutá tak, aby fungovala na ľubovoľných údajoch. Vo všeobecnosti však prevládajú supervised metódy a to čiastočne kvôli tomu, že konštrukcia unsupervised metódy môže byť oveľa ťažšia a čiastočne preto, že unsupervised metódy musia počítať s oveľa viacej faktormi, čo má vplyv na ich výkon.

2.2 Zdroje metadát publikácií

Základnú množinu metadát o vedeckých prácach získavame z portálu Centrálneho registra evidencie publikačnej činnosti (ďalej CREPČ). Na tomto portáli je možné vyhľadávať vedecké publikácie na základe názvov, mien autorov, pracovísk, hesiel, roku vydania publikácie, roku citovania, ISBN, ISSN a vydavateľstiev. CREPČ tiež poskytuje ročný export metadát publikácií daných vedeckých pracovísk vo formáte XML. Na vytvorenie úvodnej množiny údajov používame práve tieto exporty, z ktorých získavame informácie o konkrétnom diele, jeho autoroch a iné užitočné metadáta (rok vydania, vydavateľstvo, pracovisko a pod.).

Vzhľadom na neúplnosť a občasnú nepresnosť záznamov v xml exporte CREPČ, bude neskôr potrebné tieto dáta obohatiť použitím webových služieb citačných indexov ako sú WOS a SCOPUS.

Nasledujúca tabuľka obsahuje jednotlivé elementy CREPČ xml exportu, dátové typy, ktoré obsahuje, početnosti a samotné namapovanie na relačnú databázu.

Názov atribútu	Dátový typ	Formát	Početnosť	Názov stĺpca v DB	Tabuľka v DB
Kolekcia_hlavicka			1		
sigla	String	Enum	1	sigla	library
kolekcia_data			1		
kolekcia_data/data_zaznam/			0..*		
data_zaznam /zaznam_identifikacia			1		
identifikacia_orgID(attr: id_typ)	String		1..*	original_id	publication
identifikacia_datumvytvorenia	String	datum _typ	1	date_make	publication
identifikacia_datumzmeny	String	datum _typ	1	date_edit	publication
identifikacia_druhdokumentu	String		0..1	code_of_typ e	document_ty pe
identifikacia_ISBN	String		0..*	code	identification _code
identifikacia_ISSN	String		0..*	code	identification _code
identifikacia_MDT	String		0..*	code	identification _code
data_zaznam /zaznam_nazov(attr: format)			1		
nazov_hlavny	String		1..*	name	name

nazov_subezny(attr: jazyk)	String		0..*	name	name
nazov_podnazov	String		0..*	name	name
nazov_preklad(attr: jazyk, doplnky, cislo_casti, nazov_casti)	String		0..*	name	name_translation
nazov_casti	String		0..*	name	name_of_part
nazov_cislocasti	String		0..*	number	name_number_of_part
nazov_autorskezahlavie(attr: typ)	String		0..*	heading	name_author_heading
data_zaznam /zaznam_autori (typ=primarna, sekundarna)			0..1		
autori_osoba(attr: typ)			0..*		
autori_osoba/osoba_priezvisko	String		1	surname	author_person
autori_osoba/osoba_meno	String		1	name	author_person
autori_osoba/osoba_pracovisko(attr:pracovisko_typ)	String	Enum	1..*	code	person_workplace
autori_osoba/osoba_rola alebo ezp_rola (attr: format = UNIMARC,MARC21)	String		1..*	role	person_role
autori_osoba/osoba_podiel	Integer		1	contribution	author_person
autori_osoba/osoba_orgID(attr:id_typ)	String		0..*	original_id	author_person
autori_osoba/osoba_titul(attr: titul_typ)	String		0..*	title	title

autori_korporacia			0..*		
autori_korporacia/korporacia_nazov	String		1	name	author_corporation
autori_korporacia/korporacia_podcastnazvu	String		0..*	subname	corporation_subname
autori_korporacia/korporacia_privlastok	String		0..*	attribute	corporation_attribute
autori_korporacia/korporacia_rola	String		1..*	role	corporation_role
autori_korporacia/korporacia_orgID	String		0..1	original_id	corporation
autori_korporacia/korporacia_zhromazdenie_cislo	String		0..1	number	convention
autori_korporacia/korporacia_zhromazdenie_miesto	String		0..1	place	convention
autori_korporacia/korporacia_zhromazdenie_datum	String		0..1	date	convention
data_zaznam /zaznam_pracoviska			1		
pracoviska_pracovisko			1..*		
pracoviska_pracovisko/pracovisko_fakulta	String		1	faculty	publication_workplace
pracoviska_pracovisko/pracovisko_katedra	String		1	desk	publication_workplace
data_zaznam/zaznam_jazyk (attr: format)			1		

jazyk_textu	String		1..*	language_text	publication
jazyk_originalu	String		0..*	language_original	publication
jazyk_resume	String		0..*	language_resume	publication
data_zaznam /zaznam_ohlasy			0..1		
ohlasy_ohlas			1..*		
ohlasy_ohlas/ohlas_kategoria	String		1	category	response
ohlasy_ohlas/ohlas_rok	String		1	year	response
ohlasy_ohlas/ohlas_zapis	String		0..1	entry	response
ohlasy_ohlas/ohlas_index	String		0..1	index	response
data_zaznam /zaznam_vydanie			0..1		
vydanie_krajina(attr: format)	String		0..*	name	publication_country
vydanie_miesto	String		0..*	name	publication_place
vydanie_vydavatel	String		0..*	name	publisher
vydanie_rok	String		0..*	year	publication_year
vydanie_cislo	String		0..*	number	publication_number

data_zaznam /zaznam_ine			0..1		
ine_rozsah	String		0..1	range	publication
ine_edicia_nazov	String		0..1	name	edition
ine_edicia_zvazok	String		0..1	edition_volume	publication
ine_url(attr: url_typ)	String		0..*	url	url
data_zaznam /zaznam_hesla			0..1		
heslo_osoba			0..*		
heslo_osoba/osoba_priezvisko	String		1	surname	keyword_person
heslo_osoba/osoba_meno	String		1	name	keyword_person
heslo_osoba/osoba_orgID	String		0..1	original_id	keyword_person
heslo_osoba/heslo_casti	heslo_casti		0..1	keyword_of_part_id	keyword_person
heslo_korporacia			0..*		
heslo_korporacia/korporacia_nazov	String		1	name	keyword_corporation
heslo_korporacia/korporacia_podcastnazvu	String		0..*	subname	keyword_corporation
heslo_korporacia/korporacia_privlastok	String		0..*	attribute	keyword_corporation
heslo_korporacia/korporacia_orgID	String		0..1	original_id	keyword_corporation

heslo_korporacia/heslo_casti	heslo_casti		0..1	keyword_of_part_id	keyword_corporation
heslo_predmet(attr: typ)			0..*		
heslo_predmet/predmet_nazov	String		1..*	name	keyword_subject
heslo_predmet/predmet_orgID	String		0..1	original_id	keyword_subject
heslo_predmet/heslo_casti	heslo_casti		0..1	keyword_of_part_id	keyword_subject
heslo_casti			0..1		
cast_tematicka	String		0..1	part_theme	keyword_of_part
cast_geograficka	String		0..1	part_geographic	keyword_of_part
cast_chronologicka	String		0..1	part_chronological	keyword_of_part
data_zaznam /zaznam_vazby			0..1		
vazby_zaznam(attr: typ)			1..*		
zaznam_identifikacia_viazaneho			0..1		
zaznam_identifikacia_viazaneho/ identifikacia_orgID	String		0..1	original_id	publication
zaznam_identifikacia_viazaneho/ identifikacia_ISBN	String		0..*	code	identification_code
zaznam_identifikacia_viazaneho/ identifikacia_ISSN	String		0..*	code	identification_code

zaznam_nazov	zaznam_nazov		1	POPIS VYŠŠIE	
zaznam_oznacenie_zvazku	String		0..1	POPIS VYŠŠIE	
zaznam_vydanie	zaznam_vydanie		0..1	POPIS VYŠŠIE	
zaznam_autori	zaznam_autori		0..1	POPIS VYŠŠIE	
ohlasy_ohlas	ohlasy_ohlas		0..1	POPIS VYŠŠIE	

3 Špecifikácia požiadaviek

3.1 Nefunkcionálne požiadavky

3.1.1 Párovanie publikácií

Algoritmy porovnávania a párovania musia mať dostatočne malú časovú zložitosť, aby mohli byť nasadené v "real time" prostredí a aby mohli analýzy a párovania prebiehať online. V rámci požiadaviek na produkt zákazník požaduje, aby bol systém schopný dohľadať informácie k špecifickému autorovi. Táto úloha však potrebuje prebehnúť v konečnom čase tak, aby sme nového používateľa neodradili dlhotrvajúcimi výpočtami a neskorými dobami odpovede. Ďalej je potrebné výpočtový výkon rozdeliť medzi získavanie nových údajov a spracovávanie týchto nových údajov, preto musia algoritmy párovania bežať čo najrýchlejšie.

3.1.2 Import dát do databázy z exportu XML CREPČ

Algoritmus pre import dát z exportu z XML z Centrálneho registra evidencie publikačnej činnosti (CREPČ) musí byť efektívny. Nie je určené žiadne obmedzenie pre rýchlosť algoritmu. Za základe faktu, o aké veľké dáta pri spracovaní ide, tento algoritmus musí pracovať tak, aby mal čo najnižšie nároky na pamäť.

3.2 Funkcionálne požiadavky

3.2.1 Párovanie publikácií

Je dôležité pracovníkom poskytnúť čo možno najspoľahlivejšie informácie, a preto je dôležité, aby algoritmy párovania publikácií boli čo možno najúčinnejšie. Okrem toho z hľadiska budúceho rozšírenia programu musia poskytovať možnosť rozšírenia o zisťovanie ďalších relevantných informácií. Ďalej musia podporovať manuálne spárovanie publikácii, ak overený a zaregistrovaný výskumník, teda používateľ našej aplikácie označí dve publikácie ako totožné.

3.2.2 Import dát do databázy z exportu XML CREPČ

Pre fungovanie párovania publikácii je potrebné pracovať s nejakými dátami. Tieto dáta sa budú získavať z Centrálneho registra evidencie publikačnej činnosti (CREPČ) v istých dlhodobých intervaloch vo forme XML exportu. Dané XML sa bude spracovávať na základe zadanej požiadavky do systému cez webové rozhranie. XML sa bude parsovat a importovat do existujúcej databázy. Pre účel perzistencie údajov sparsovaných údajov z XML CREPČ sa vytvorí PostgreSQL databáza (relačného charakteru), ktorá bude tieto údaje uchovávať a poskytovať pre funkciu Párovania publikácií. Pri zadaní požiadavky do systému sa funkcia nespustí ihneď. Bude sa spúšťať v nočných hodinách, kedy je na server smerovaných čo najmenej požiadaviek.

3.2.3 Použitá databázová technológia

Pre perzistenciu spracovávaných údajov bude potrebné použitie relačnej databázy, ktorá by bola dostatočne výkonná, pretože sa jedná o spracovanie veľkého množstva údajov. Jednou z požiadaviek je možnosť pripojenia na danú databázu pomocou Java JDBC. Použitie dokumentovej databázy sa nejavilo vhodné z dôvodu potreby relačných vzťahov medzi záznamami. Požiadavkou, ktorú musí spĺňať uvedená databáza, tiež bola možnosť vytvárania databázových funkcií (v jazyku SQL, PL/SQL alebo C), ktoré by mohli zjednodušiť niektoré opakované databázové operácie.

3.2.4 Webové rozhranie

Webové rozhranie systému musí ponúkať jednoduchý spôsob zobrazovania nahromadených dát v databáze. Medzi hlavné entity patria autori, publikácie a vzťahy medzi nimi. Používateľské rozhranie musí ponúkať prehľadný spôsob filtrovania veľkého počtu dát na základe všetkých atribútov, ktorými dané entity disponujú.

3.2.5 Webové služby

Systém bude okrem webového rozhrania ponúkať tiež webové služby, cez ktoré bude možné využívať implementované algoritmy porovnávania a dopytovanie sa nad dátami v našej databáze.

3.2.6 Zdroje znalostí

Systém bude využívať viaceré zdroje znalostí a kombinovať údaje v nich nájdené, aby sa tak zabezpečila väčšia presnosť párovania publikácií a viac údajov o ohlasoch. Týmito zdrojmi budú CREPČ - Centrálny register evidencie publikačnej činnosti, Scopus, Web of Science a Google Scholar.

Prvý menovaný zdroj zabezpečuje údaje o publikáciách na Slovensku. Nie všetky vedecké inštitúcie využívajú tento systém. Zvyšné tri sú citačné indexy, v ktorých budeme vyhľadávať ohlasy na diela získané z CREPČ a na získavanie ďalších diel.

Preto musí systém vedieť komunikovať s týmito službami, musí vedieť spracovávať výstupy získané z týchto služieb a mapovať tieto výstupy na seba. Pod mapovaním rozumieme jednak spracovanie údajov do spoločného formátu, a potom aj párovanie publikácií.

4 Návrh

4.1 Technológie

4.1.1 C++

Na návrh a implementáciu párovacích algoritmov bol vybraný programovací jazyk C++ a to najmä pre svoju rýchlosť, ktorá je v tomto procese veľmi dôležitá a vzhľadom na to, že sme používame adaptáciu návrhového vzoru repository, tak nemusí byť tento komponent v Jave. Vzhľadom na to, že aj MongoDB je v naprogramované v C++, používanie C++ na párovanie je taktiež výhodné.

4.1.2 PostgreSQL

Na uloženie metadát získaných z exportu CREPČ a neskôr aj metadát získaných z citačných indexov sme použili databázu PostgreSQL. Riešenie minuloročného tímu s použitím MongoDB sa javilo ako nevhodné z dôvodu potreby ukladania vzťahov medzi jednotlivými publikáciami a autormi, a medzi publikáciami navzájom (v prípade že sa niektoré dielo odkazuje na iné formou citácie).

4.1.3 Java

Modul importovania CREPČ do PostgreSQL, modul crawlovania Google Scholar a webový frontend aplikácie bol vytvorený v programovacom jazyku Java a to hlavne kvôli objektovo orientovanému prístupu a podpory mnohých knižníc, ktoré zjednodušujú prácu s databázou (Hibernate), xml súbormi (dom4j), a umožňujú vytvárať webové aplikácie v prostredí java (GWT). Výhodou takéhoto riešenia je vytvorenie systému, ktorý bude pripravený na ďalšie rozširovanie a reálnu prevádzku.

4.1.4 Hibernate

Na zabezpečenie objektového prístupu k PostgreSQL databáze bola použitá technológia Hibernate. Tá predstavuje objektovo - relačný mapovač (ORM), ktorý zjednodušuje a sprehľadňuje prácu s databázou, keďže odbremeňuje programátora od písania SQL dotazov.

4.1.5 JSP, Servlet

Frontend aplikácie je naprogramovaný v jazyku Java a využíva Tomcat server. Vďaka tejto skutočnosti vieme jednoduchým spôsobom pristupovať k našim dátam pomocou Hibernate ORM rozhrania a nemusíme implementovať túto komunikáciu nanovo (ako by bolo nutné napríklad pri využití technológie PHP).

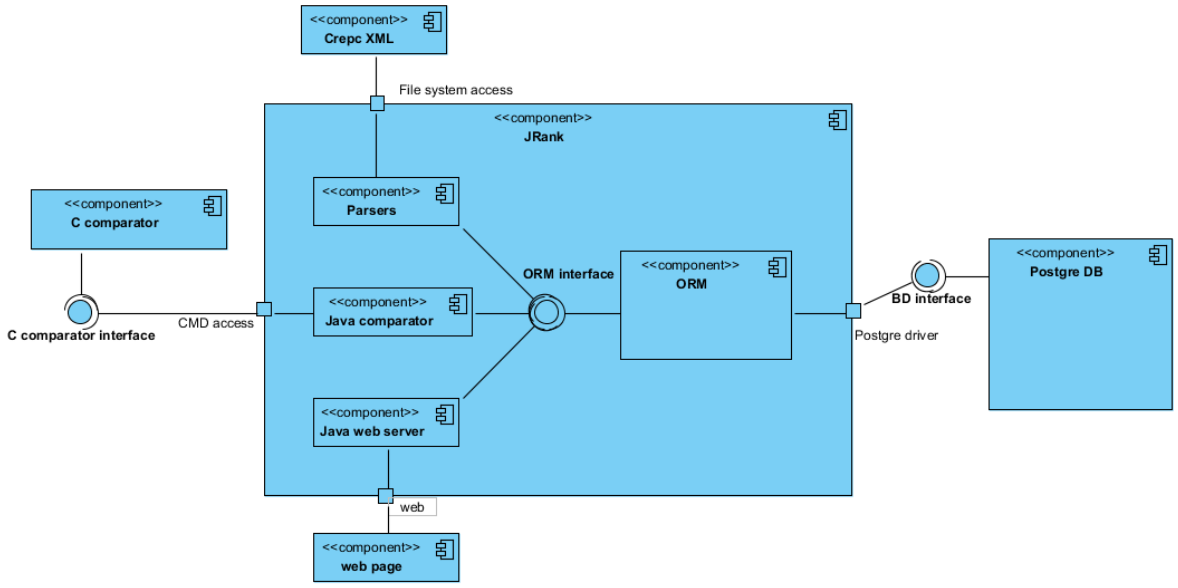
Využívame Servlety ako jednoduchý a priamočiari spôsob komunikácie typu klient-server v prostredí Java.

5 [1]Implementácia

Koncepcia systému vznikla predovšetkým s prihliadnutím na jasnú, jednoduchú organizáciu, slabo previazané komponenty a znovupoužiteľnosť. Taktiež jedna z najdôležitejších požiadaviek bolo zabezpečiť rozšíriteľnosť systému. Na zabezpečenie týchto požiadaviek vznikol systém opísaný na diagrame nižšie.

Research rank systém dokáže flexibilne, asynchrónne spracovávať údaje z daných zdrojov vedomostí. Nad údajmi sa ďalej asynchrónne spúšťa proces deduplikácie a proces čistenia údajov, kde sa snaží systém automaticky párovať spracované entity. Tieto procesy skvalitnia, zjednotia a sprehľadnia údaje v databáze, čím sa skvalitnia poskytované informácie. Údaje potom zobrazuje pomocou webového rozhrania.

Zatiaľ je zapojený v tomto prototype len XML Crepč. Prebieha príprava zdrojov Web of Science (WoS), Scopus a Google Scholar.



Obrázok 1: Diagram komponentov systému Research Rank.

Opis jednotlivých komponentov ďalej v podkapitolách.

5.1 JRank

Obsahuje komponenty systému napísane v jazyku Java.

5.1.1 ORM

5.1.1.1 Plnenie databázy ResearchRank dátami z databázy CREPČ

Pre potreby párovania vedeckých publikovaných diel vznikla potreba vytvorenia relačnej databázy zahrnutej v rámci systému ResearchRank, ktorá by tvorila základ dát potrených pre párovanie iných diel. Údaje, ktorými

[1] Bolo zmenené formátovanie textu oproti prvému odovzdaniu v zimnom semestri, keďže sa pôvodné formátovanie vzhľadom k rozšírenému obsahu stalo neprehľadným.

je túto databázu potrebné naplniť, sú údaje z Centrálneho registra evidencie publikačnej činnosti (ďalej CREPČ). Údaje z tejto databázy sú poskytované vo forme exportu vo formáte XML. Všetky potrebné údaje o štruktúre XML súboru sú dostupné v dokumente o výsledkoch analýzy XML CREPČ.

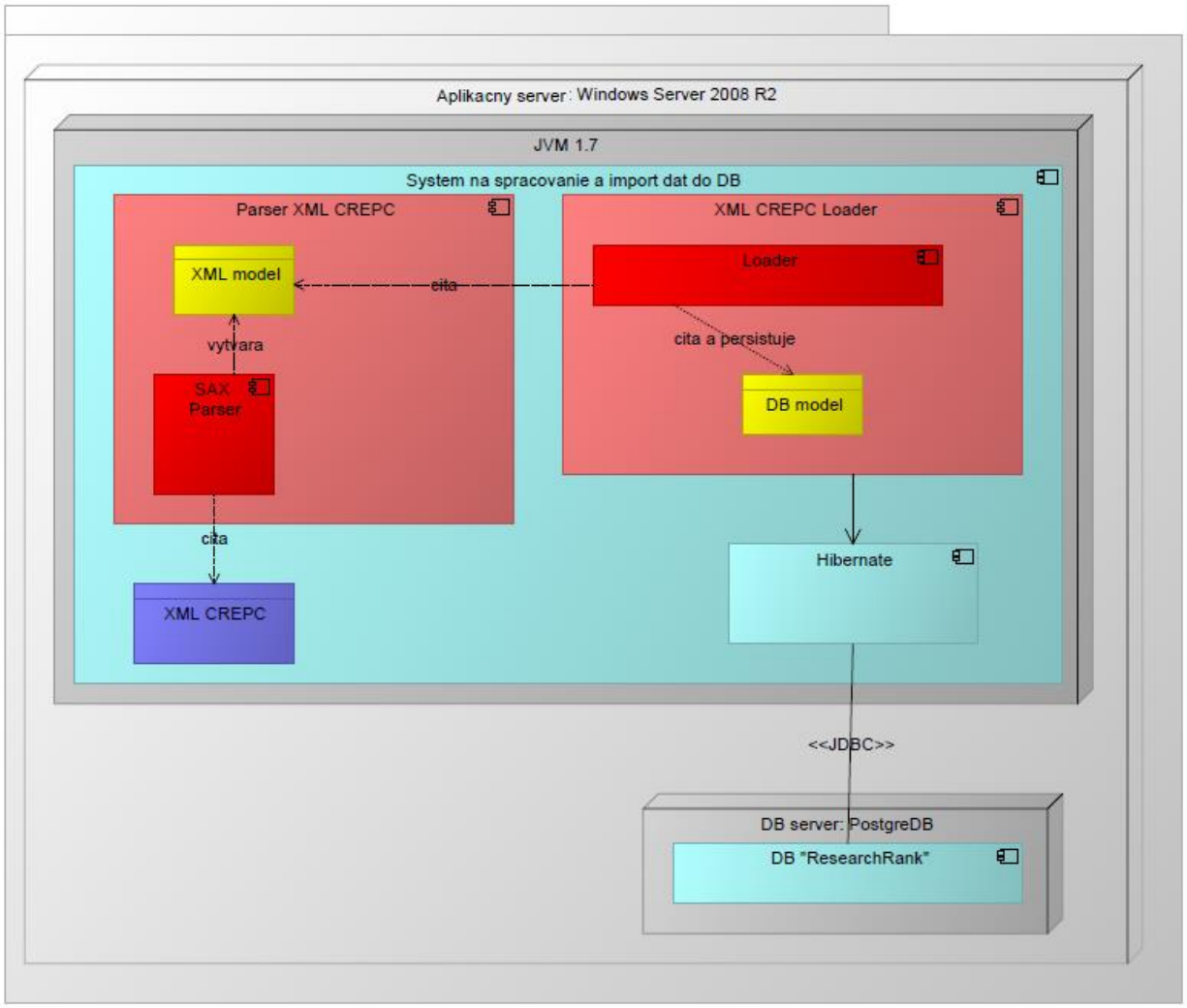
5.1.1.1.1 Systém pre plnenie databázy ResearchRank relevantnými dátami

Pre vytvorenie takejto databázy je nevyhnutné vytvoriť systém, ktorý exportovaný XML súbor prvé spracuje a potom naimportuje do relačnej databázy. Celý systém je implementovaný v programovacom jazyku Java. Systém je implementovaný v rámci prostredia JRE (Java Runtime Environment) verzie 1.7 bežiaceho v operačnom systéme Windows 7 Professional (64-bit).

Systém som rozdelil na 2 časti:

- Parser XML CREPČ
- Importer do PostgreSQL databázy

Prvá časť bude spracovávať (ďalej parsovať) XML CREPČ. Spracovávať ho bude do štruktúry objektov implementovaných podľa štruktúry XML CREPČ. Celá zaplnená štruktúra objektov XML CREPČ je postupne prepisovaná do štruktúry modelu databázy ResearchRank a počas prepisovania sa postupne ukladá do databázy. K dispozícii je dokument javadoc celého systému a dokumentácia schém DB a Hibernate.



Obrázok 2: Systém pre plnenie databázy ResearchRank relevantnými dátami

5.1.1.1.1.1 Parser XML CREPC

Komponent Parser XML CREP funguje na základe metódy SAX parsovania. Čítajú sa postupne riadok po riadku a pri prečítaní xml tagu sa spustí funkcia závislá od toho, či ide o začiatočný tag alebo o koncový tag. Pri takomto postupnom čítaní tagov prebieha načítavanie textových reťazcov do XML modelu objektov.

Po skončení parsovania sa model objektov pošle do komponentu XML CREPC Loader.

5.1.1.1.1.2 XML CREPC Loader

Tento komponent využíva technológiu Hibernate. Základom používania tejto technológie bolo vygenerovanie DB modelu, do ktorého sa postupne vkladali dáta z XML modelu a perzistovali priamo popri ukladaní. Implementácia systému sa niesla vo verziovaní.

Verzie systému sú:

- 1 - implementácia základnej fungujúcej verzie,
- 2 - optimalizácia - pri ktorej sa dopytom do DB kontrolovala existencia danej entity v DB namiesto vopred načítaných entít z DB, ktoré boli pamäťovo náročné.
- 3 - optimalizácia - zrušenie transakcií pri každej operácii perzistovania, ktoré boli časovo náročné,
- 4 - implementácia bezpečnostného prvku - vytvorenie jednej transakcie v rámci celého importu, čím sa zamedzilo uloženiu nekompletného importu pri nečakanom prerušení procesu,
- 5 - implementácia bezpečnostného prvku - zrušenie jednej transakcie v rámci celého importu a vytvorenie transakcie pre perzistovanie každého jedného záznamu a implementácia kontrolného zapisovania indexu naposledy uloženého záznamu, ktorý sa použije pre pokračovanie procesu importu po nečakanom zlyhaní.

5.1.2 Frontend

5.1.2.1 Funkcia:

Sprístupňuje dáta z databázy pomocou používateľského rozhrania. Ponúka možnosti filtrovania dát a ich správu po prihlásení používateľa.

5.1.2.2 Vstup:

Dáta z Postgre databázy sprístupňované pomocou modulu ORM.

5.1.2.3 Výstup:

Zobrazovanie dát, ich filtrovanie a vizualizácia vzťahov.

5.1.2.4 Popis:

ponúka jednoduché používateľské prostredie pre získavanie požadovaných informácií ohľadom publikácií, ich autorov a vzťahoch medzi nimi.

5.1.3 Java Comparator

5.1.3.1 Funkcia:

Zabezpečuje rozhranie medzi samotnými algoritmami párovania a databázou, zahŕňa algoritmus na výber kandidátov. Taktiež zabezpečuje spracovanie výstupov z C komparátora.

5.1.3.2 Vstup:

Nové záznamy v databáze (prípadne iné), ktoré je potrebné spárovať s existujúcimi záznamami v databáze. Konkrétne sú to tie záznamy, ktoré majú nastavenú hodnotu checked na false.

5.1.3.3 Výstup:

Spárované záznamy v databáze

5.1.3.4 Popis:

Java komparátor je komponent, ktorý je spúšťaný v pravidelných intervaloch, kedy sú pripravené nové údaje. Taktiež ďalší scenár počíta s pravidelným spúšťaním tak, aby sme skontrolovali záznamy v databáze a podľa novo získaných údajov sa ich opätovne pokúsili spárovať s ďalšími záznamami.

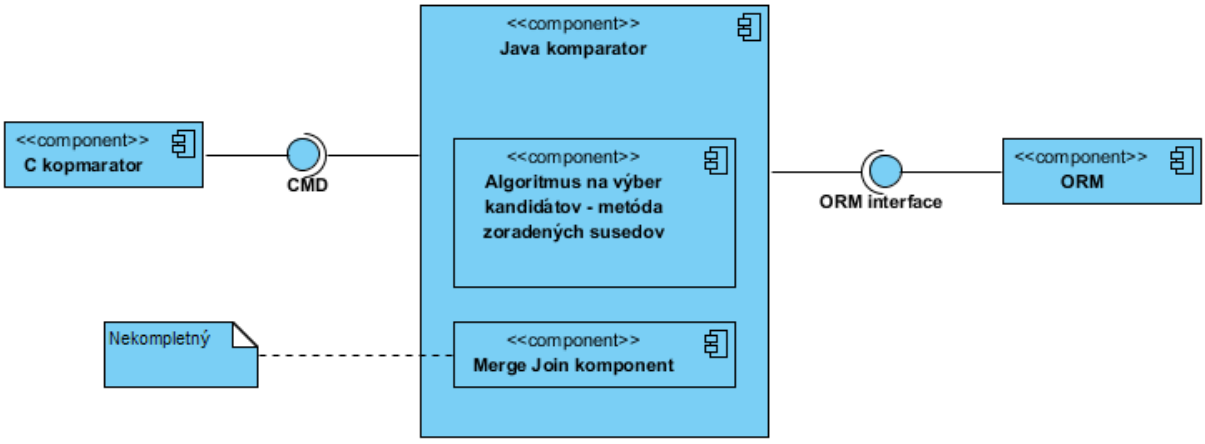
Java komparátor využíva ORM na získanie záznamov, s ktorými potom ďalej pracuje. Iteruje cez záznamy, ktoré sú označené na párovanie a ku každému získa zoznam objektov, ktoré majú najväčšiu pravdepodobnosť zhody alebo podobnosti s príslušným záznamom. Na to sa používajú algoritmy výberu kandidátov.

V tomto komponente je implementovaný algoritmus používajúci metódu zoradených susedov. Záznamy zoraďuje podľa ich názvov a na základe stanovenej hranice zoberie určitý počet záznamov nachádzajúci sa pod a nad hľadaným záznamom. Na zoradenie používa databázovú funkciu row_count.

Aby sme dokázali využiť funkcionalitu, ktorú ponúka Hibernate, vytvorili sme view, ktorý má jedinú funkciu priradiť k záznamu jeho poradie v rámci zoradeného zoznamu záznamov.

Do budúcnosti je plánované rozšíriť zoraďovanie podľa iných atribútov, napríklad podľa mien autorov alebo zdrojovej publikácie, aby sme tak zabezpečili väčšiu šancu takých záznamov, kde sa vyskytuje chyba na prvých písmenách názvu publikácie.

Podľa špecifikovaného formátu vyskladá Java komparátor argumenty C komparátora a odošle ich na samotné párovanie. Potom spracuje výsledky párovania, ktoré prijme od C komparátora. Tieto výsledky sú vo forme poradového čísla záznamu, v zozname argumentov ktorý má byť spárovaný s hľadaným záznamom. Ďalej tieto výsledky pošle Java komparátor na vykonanie merge a join operácií. Táto časť funkcionality však ešte nie je implementovaná.



Obrázok 3: Java komparátor - komponent

5.2 CRank

Obsahuje popis komponentov napísaných v C a C++.

5.2.1 C Comparator

5.2.1.1 Funkcia:

Komponent zabezpečuje službu, ktorá určí, či daná publikácia ma v príslušnej množine publikácií duplikát.

5.2.1.2 Vstup:

Vstupom komponentu sú argumenty z príkazového riadku. Prvý argument je počet publikácií, ktoré predávame komponentu. Druhý argument je publikácia, ktorej duplikáty hľadáme. V ostatných argumentoch predávame komparátoru množinu publikácií, kde má hľadať duplikáty. Formát publikácie je definovaný nasledujúcou tabuľkou:

	First author	Count titles	Titles []	count keywords	keywords[]	count co-authors	co-authors[]	Abstract_present	abstract	Year present	year	Source presents	source_name	ISBN count	ISBN[]	ISSN count	ISSN[]	workplace count	workplaces[]	publisher count	publishers[]
Input string 1	"FA T_C T1 .. Tn K_C K1 .. Kn CA_C CA1 .. CAn AP Abst. YP Year SP SN ISBN_C ISBN1 .. ISBNn ISSN_C ISSN1 .. ISSNn WP_C WP1 .. WPn P_C P1 .. Pn"																				
:	..																				
Input string n	..																				

Vzhľadom na množstvo možných informácií je vstupný odtlačok publikácie značne zložitý, čo ilustruje tabuľka.

5.2.1.3 Výstup:

Výstup komparátora je vektor čísiel. Prvé číslo určuje počet nájdených duplikátov a ostatné čísla určujú indexy duplikátov v relatívne ku vstupnej množine publikácií.

Príklad: Vo vstupnej množine je 3. publikácia duplikátom.

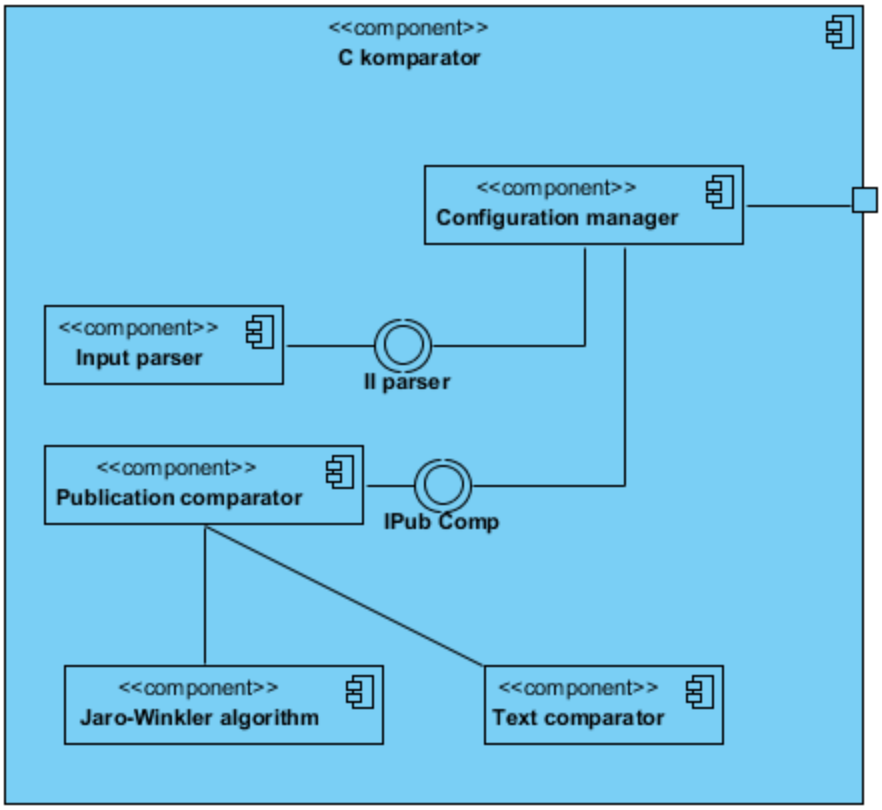
Výstup potom bude: “1 3”.

5.2.1.4 Popis:

C komparátor má čo možno najspoľahlivejšie určiť podobnosť publikácií. Na to má v sebe implementovaných niekoľko komponentov. S vonkajšieho sveta ho volajú aplikácie na komparáciu pomocou príkazového riadku. V rámci architektúry programu by mal interagovať iba s Java komparátorom, ktorý zabezpečuje výber kandidátov a spúšťanie C komparátora.

Komparátor bol implementovaný vo Visual Studiu 2012 v jazyku C++. Neobsahuje žiadne “third party” komponenty.

Konkrétna implementácia C komparátora bola určená na základe porovnania niekoľkých metód párovania, z nich bol vybraný najlepší algoritmus na základe testov nad menami autorov zo slovenského prostredia. Architektúra bola koncipovaná predovšetkým s prihliadnutím na možnosť neskoršej zmeny komponentov ako aj rozšírenia funkcionality.



Obrázok 4: Vnútorná štruktúra C komparátora.

Komparátor obsahuje 5 základných komponentov. Manažér konfigurácie verifikuje správnosť vstupu, manažuje volanie parsera na vstup, volanie samotného komparátora a nakoniec výstup. Parser vstupu dostane verifikovaný vstup od manažéra konfigurácie a transformuje odtlačky vstupných publikácií do reprezentácie v pamäti kompatibilnej s komparátorom publikácii. Komparátor publikácií dostane od manažéra konfigurácie záznamy publikácií, ktoré má porovnať. Následne aplikuje na jednotlivé polia publikácie porovnanie. Podľa typu ide o porovnania pomocou Jaro-Winklerovho algoritmu na porovnávanie, porovnávanie textov (ako sú tituly, abstrakty a pod.) a nakoniec porovnania vyžadujúce presnú zhodu atribútov. Komponent Jaro-Winkler algoritmus obsahuje implementáciu rovnomenného algoritmu. Textový komparátor hľadá zhodu sekvencií tokenov v rámci stokenizovaného textu.

Ak sa komparátoru správne nastavia koeficienty, podľa ktorých určí podobnosť za zhodu, tak výkon komparátora z hľadiska odhaľovania duplicity je uspokojivý.

5.3 DBRank

Obsahuje popis databázy, ktorá je v systéme použitá.

5.3.1 Postgre DB

5.3.1.1 Funkcia:

Uchovávanie dát získaných z CREPČ, a neskôr aj iných zdrojov. Slúži tiež na ukladanie informácií o používateľoch frontedu.

5.3.1.2 Vstup:

Vstupné dáta sú do databázy vkladané prostredníctvom CREPČ parsera. Ten využíva ORM na prístup a úpravu databázy. Neskôr bude možné rozšíriť uvedený systém o ďalšie zdroje, akými sú WoS, Scopus a Google Scholar.

5.3.1.3 Výstup:

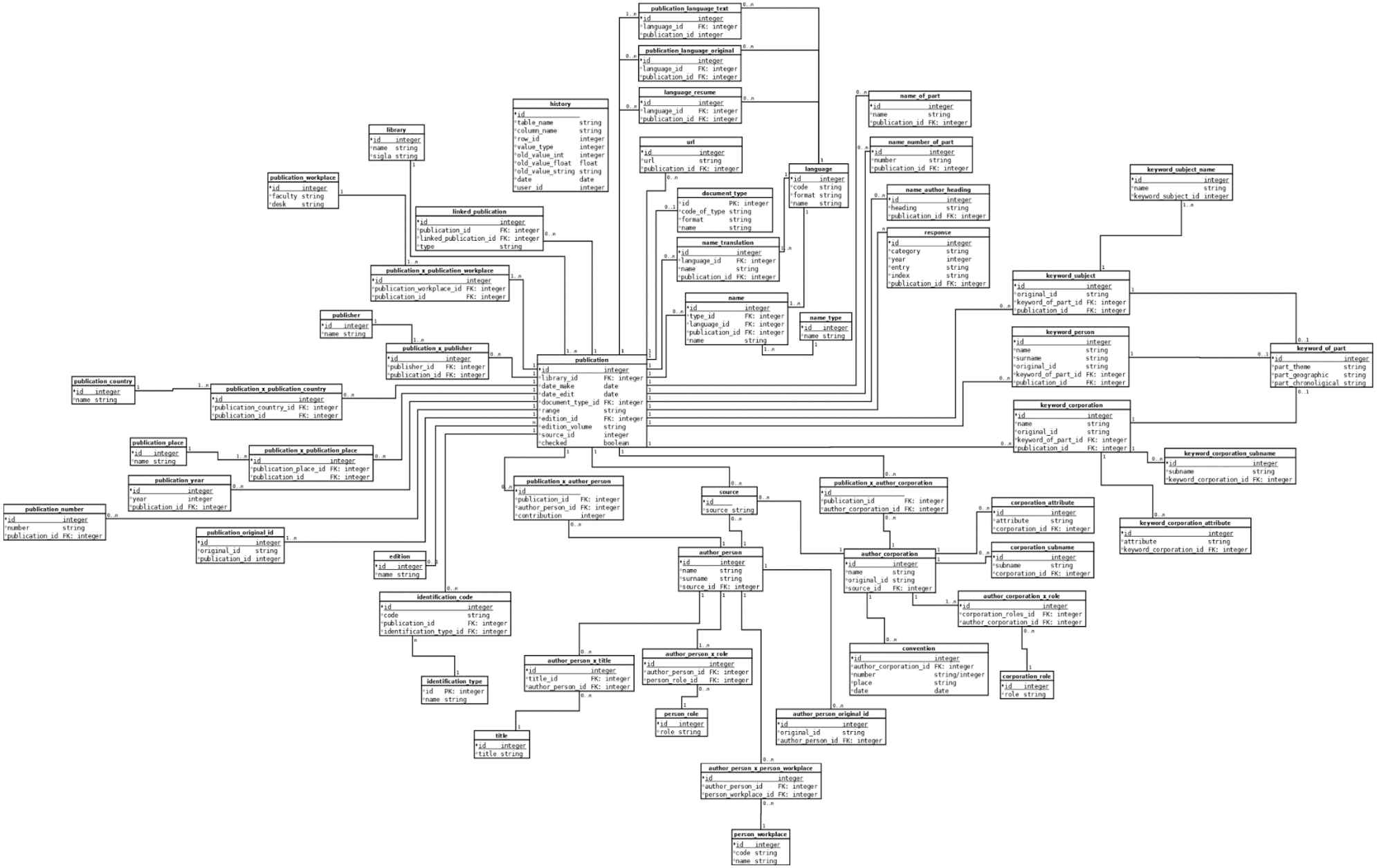
Výstupom sú uchované dáta získané v predošlých krokoch.

5.3.1.4 Popis:

Technológiu Postgre sme zvolili kvôli rýchlosti, ktorou disponuje a faktu, že sa jedná o open source riešenie a pre jej použitie nie je teda potrebná platená licencia. Jednotlivé komponenty systému pracujú s databázou prostredníctvom objektovo-relačného mapovača Hibernate.

Pri návrhu dátového modelu sme primárne vychádzali z dátového modelu CREPČ-u, ktorý bol pre naše požiadavky najvhodnejší. Samotný dátový model určený pre ukladanie údajov publikácií je možné vidieť na nasledujúcom obrázku.

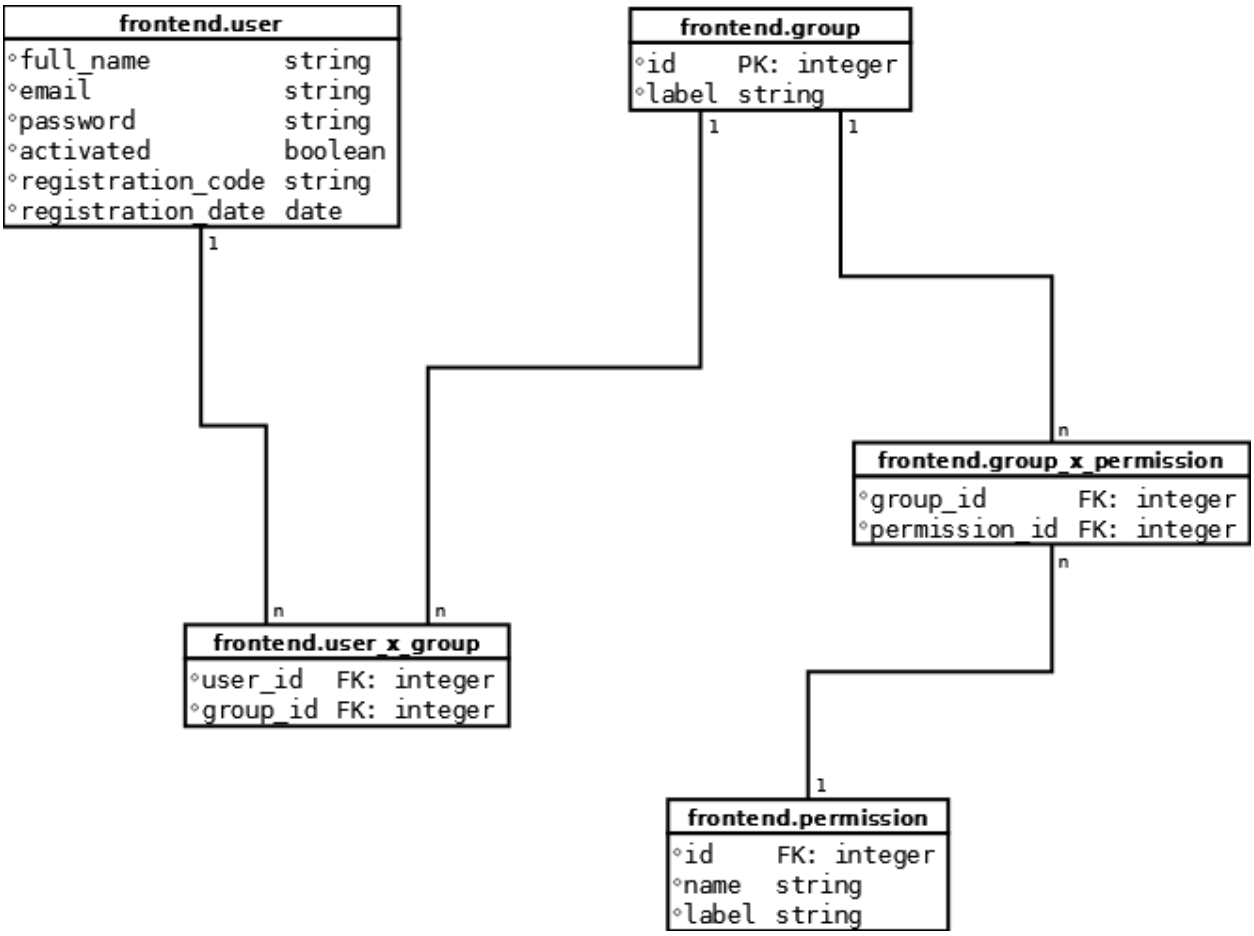
5.3.1.5 Schéma databázy


publication_language_text
*id integer
*language_id FK: integer
*publication_id integer
publication_language_original
*id integer
*language_id FK: integer
*publication_id FK: integer
language_resume
*id integer
*language_id FK: integer
*publication_id FK: integer
history
*id
*table_name string
*column_name string
*row_id integer
*value_type integer
*old_value_int integer
*old_value_float float
*old_value_string string
*date date
*user_id integer
library
*id integer
*name string
*sigla string
publication_workplace
*id integer
*faculty string
*desk string
linked_publication
*id integer
*publication_id FK: integer
*linked_publication_id FK: integer
*type string
publication_x_publication_workplace
*id integer
*publication_workplace_id FK: integer
*publication_id FK: integer
publisher
*id integer
*name string
publication_x_publisher
*id integer
*publisher_id FK: integer
*publication_id FK: integer
publication_country
*id integer
*name string
publication_x_publication_country
*id integer
*publication_country_id FK: integer
*publication_id FK: integer
publication_place
*id integer
*name string
publication_x_publication_place
*id integer
*publication_place_id FK: integer
*publication_id FK: integer
publication_year
*id integer
*year integer
*publication_id FK: integer
publication_number
*id integer
*number string
*publication_id FK: integer
publication_original_id
*id integer
*original_id string
*publication_id integer
edition
*id integer
*name string
identification_code
*id integer
*code string
*publication_id FK: integer
*identification_type_id FK: integer
identification_type
*id PK: integer
*name string
publication
*id integer
*library_id FK: integer
*date_make date
*date_edit date
*document_type_id FK: integer
*range string
*edition_id FK: integer
*edition_volume string
*source_id integer
*checked boolean
publication_x_author_person
*id
*publication_id FK: integer
*author_person_id FK: integer
*contribution integer
source
*id
*source_string
author_person
*id integer
*name string
*surname string
*source_id FK: integer
author_person_x_title
*id integer
*title_id FK: integer
*author_person_id FK: integer
author_person_x_role
*id integer
*author_person_id FK: integer
*person_role_id FK: integer
person_role
*id integer
*role string
title
*id integer
*title string
author_person_original_id
*id integer
*original_id string
*author_person_id FK: integer
author_person_x_person_workplace
*id integer
*author_person_id FK: integer
*person_workplace_id FK: integer
person_workplace
*id integer
*code string
*name string
publication_x_author_corporation
*id
*publication_id FK: integer
*author_corporation_id FK: integer
author_corporation
*id integer
*name string
*original_id string
*source_id FK: integer
corporation_attribute
*id integer
*attribute string
*corporation_id FK: integer
corporation_subname
*id integer
*subname string
*corporation_id FK: integer
author_corporation_x_role
*id integer
*corporation_roles_id FK: integer
*author_corporation_id FK: integer
convention
*id integer
*author_corporation_id FK: integer
*number string/integer
*place string
*date date
corporation_role
*id integer
*role string
name_of_part
*id integer
*name string
*publication_id FK: integer
name_number_of_part
*id integer
*number string
*publication_id FK: integer
name_author_heading
*id integer
*heading string
*publication_id FK: integer
response
*id integer
*category string
*year integer
*entry string
*index string
*publication_id FK: integer
url
*id integer
*url string
*publication_id FK: integer
document_type
*id PK: integer
*code_of_type string
*format string
*name string
name_translation
*id integer
*language_id FK: integer
*name string
*publication_id FK: integer
name
*id integer
*type_id FK: integer
*language_id FK: integer
*publication_id FK: integer
*name string
language
*id integer
*code string
*format string
*name string
name_type
*id integer
*name string
keyword_subject_name
*id integer
*name string
*keyword_subject_id integer
keyword_subject
*id integer
*original_id string
*keyword_of_part_id FK: integer
*publication_id FK: integer
keyword_person
*id integer
*name string
*surname string
*original_id string
*keyword_of_part_id FK: integer
*publication_id FK: integer
keyword_of_part
*id integer
*part_theme string
*part_geographic string
*part_chronoligical string
keyword_corporation
*id integer
*name string
*original_id string
*keyword_of_part_id FK: integer
*publication_id FK: integer
keyword_corporation_subname
*id integer
*subname string
*keyword_corporation_id FK: integer
keyword_corporation_attribute
*id integer
*attribute string
*keyword_corporation_id FK: integer

Funkcie jednotlivých tabuliek sú popísané v tabuľke analýzy CREPČ uvedenej v kapitole Analýza problematiky.

Nasledujúci obrázok znázorňuje schému dátového modelu pre ukladanie informácii používateľov frontendu. Uvedený model má oproti minuloročnému riešeniu výhodu jednoduchej zmeny práv každej skupiny bez potreby zásahu do zdrojového kódu. Podobne aj v prípade pridania novej skupiny bude stačiť nastaviť jej práva v administrácii systému bez programátorského zásahu.

Registrácia používateľov je ošetrená voči spamu a automatickým registráciám pomocou emailovej aktivácie účtu, bez ktorej nie je možné účet využívať.



5.3.1.5.1 Popis tabuliek frontendu

user

Obsahuje informácie o používateľovi.

Názov stĺpca	Význam
full_name	Celé meno používateľa (aby nebolo nutné nútiť používateľa vypisovať meno, priezvisko, tituly a pod.)

email	Emailová adresa používateľa nutná k registrácii a aktivácii konta
password	Zahashované heslo používateľa
activated	Informácia, či bol daný účet aktivovaný prostredníctvom aktivačného mailu
registration_code	Registračný kód vygenerovaný počas registrácie. Je potrebný na emailovú aktiváciu konta
registration_date	Dátum registrácie používateľa. Môže byť potrebné ak sa budú vymazávať neaktivované účty po určitej dobe.

group

Zoznam skupín používateľov.

Názov stĺpca	Význam
label	Názov skupiny, ktorý sa zobrazuje na frontende (napr. Administrátor).

permission

Tabuľka jednotlivých práv, ktoré môžu používatelia nadobúdať.

Názov stĺpca	Význam
name	Názov práva používaný v systéme (napr. showUserList). Význam jednotlivých práv je implementovaný v zdrojovom kóde frontendu.
label	Označenie práva, ktoré sa zobrazuje vo frontende (napr. Zobraziť zoznam používateľov).

6 Šprinty zimného semestra

Táto kapitola obsahuje popis jednotlivých šprintov, ktoré sme v rámci našej práce absolvovali cez zimný semester. Šprinty sú očíslované a každý obsahuje zoznam a popis úloh, ktoré sme do neho zaradili.

6.1 Šprint 1 - Amstel

6.1.1 Analýza algoritmov párovania minuloročného riešenia, návrhy na zlepšenie

Jedna z dôležitých súčiastok projektu je párovanie publikácií. Táto téma je obsiahla a venuje sa jej veľa výskumníkov. Vzhľadom na to, že zameranie práce, na ktorú nadväzuje je do určitej miery iné, tak bolo treba zistiť, do akej miery je návrh komparátora publikácií v minuloročnom riešení dotiahnutý. V rámci tejto úlohy boli analyzované rôzne prístupy k porovnávaniu bibliografických záznamov a algoritmov na určenie podobnosti záznamov.

6.1.2 Analýza schémy XML CrepČ

Ako zdroj primárnych údajov, ktoré budeme obohacovať o dodatočné informácie, sme zvolili údaje z Centrálneho registra evidencie publikačnej činnosti (CREPČ). Pre použitie týchto údajov bola potrebná analýza ich formátu a obsahu. Informácie z CREPČ sú dostupné ako XML súbory so štruktúrou popísanou v oficiálnej dokumentácii [7]. Analýza formátu a obsahu nám neskôr bola užitočná pri návrhu relačnej databázy určenej na ukladanie týchto údajov. Táto databáza bude tiež slúžiť na ukladanie dodatočných informácií z iných zdrojov.

6.1.3 Analýza Google Scholar

Jednou z primárnych požiadaviek zadávateľa je rozšíriť dáta v systéme o ďalší zdroj - Google Scholar. Úlohou bolo zanalyzovať Google Scholar, dáta ktoré nám ponúka a možnosť ich získania a uloženia do dátového modelu pre ďalšiu prácu s nimi. V tomto šprinte bol navrhnutý spôsob ako budeme dáta získavať a spracovávať a navrhnutý základný prototyp parsera dát.

6.1.4 Vytvorenie webovej prezentácie tímu, nainštalovanie virtuálneho stroja

Pre potreby publikácie našej tímovej webovej stránky, sme potrebovali nainštalovať pridelený virtuálny Linux server. Po nespokojnosti s priloženou inštaláciou Fedory sme si vyžiadali distribúciu Ubuntu servera. Tento server sme úspešne nainštalovali a nakonfigurovali. Ďalej sme na neho nainštalovali webový server, kde sme uložili vytvorenú webovú prezentáciu nášho tímu. Webová stránka tímu bola aktualizovaná o informácie o projekte, ktorý riešime a o členoch nášho tímu. Stránka je pravidelne aktualizovaná a postupne do nej pribúdajú zápisnice zo stretnutí, plány projektu a iné potrebné dokumenty.

6.1.5 Inštalácia JIRA, Confluence, Stash, Crucible

Pre potreby tímovej komunikácie a kolaborácie sme sa rozhodli nainštalovať si Attlasian produkty. Na tieto produkty sme získali akademickú licenciu a vďaka ručnej inštaláciami sme mali plnú kontrolu nad ich konfiguráciou. Výhoda využívania týchto služieb je taká, že sú integrované, takže máme v jednom balíku nástroj na projektový manažment (JIRA), wiki stránky (Confluence), git (Stash) a review kódu (Crucible).

Inštalácia a konfigurácia týchto nástrojov prebieha iteratívne podľa potreby, čo si vyžiadalo istý čas navyše, ktorý sme do toho museli investovať, avšak po zvážení výhod, ktoré tieto nástroje ponúkajú sme sa zhodli, že to bude pre našu tímovú prácu prínosom.

6.2 Šprint 2 - Budweiser

6.2.1 Prvá implementácia a testovanie nových párovacích algoritmov

Párovanie publikácií vo všeobecnosti závisí od porovnania autorov a spoluautorov. Na základe analýzy z predchádzajúceho šprintu boli vybrané dva algoritmy existujúce prístupy na párovanie publikácií (Jaro-Winkler a Damerau-Levenstein) a jeden bol navrhnutý naším tímom. Algoritmy boli implementované, otestované a čiastočne prebehlo ich porovnanie.

6.2.2 Návrh dátového modelu

Na základe predchádzajúcej analýzy CREPČ sme vytvorili návrh dátového modelu pre vytváraný projekt. S použitím tohoto modelu sme následne vytvorili databázu v PostgreSQL, do ktorej sa mal v niektorom z nasledujúcich šprintov vykonať import prvých dát z CREPČ-u. Vzhľadom na rozsiahlosť a komplikovanosť navrhovaného dátového modelu sme sa niekoľkokrát stretli s komplikáciami, ktoré sa však podarilo rýchlo vyriešiť a neohrozilo to splnenie danej úlohy.

6.2.3 Vytvorenie parsera pre XML CrepČ

Pre potrebu vytvorenia databázy, ktorú je potrebné mať na párovanie publikácii, sa údaje získavajú z Centrálneho registra evidencie publikačnej činnosti (CREPČ). Spôsob, akým sa dáta z CREPČ získavajú je export z daného registra do súboru, ktorý je vo forme XML. Pred tým, ako sa do databázy vkladajú tieto dáta z exportu, je potrebné vytvoriť parser, ktorý bude súbor tohto typu spracovávať. Vytvorili sme teda parser, ktorý je schopný export v tvare XML spracovať a uložiť do štruktúry objektov tried, ktoré sú definované presne podľa štruktúry XML CREPČ. Parser funguje na základe parsovacieho algoritmu SAX. Implementácia tohto parsera bola prácna, avšak prebehla bez akýchkoľvek problémov.

6.2.4 Vytvorenie parsera pre Google Scholar

Na základe predchádzajúcej analýzy a základného návrhu parsera bola jeho funkcionalita doplnená. Parser bol schopný získať z Google Scholar o zadanom výskumníkovi názvy jeho publikácií spolu so spoluautormi, rokom vydania a zdrojového dokumentu. V rámci tohto šprintu bol následne parser doplnený o možnosť získavania informácií a full-textu článkov a publikácií, ktoré dané dielo zadaného výskumníka citovali. Tu nastal problém, že nie všetky články majú priamy odkaz na PDF. Preto nie každé dielo v databáze bude mať svoj pridružený dokument v dokumentovej databáze.

6.2.5 Analýza frontendu minuloročného riešenia

V rámci potreby nadviazania na minuloročný tímový projekt *Odhaľovanie a hodnotenie vzťahov v oblasti vedy a výskumu* sme potrebovali analyzovať frontend, ktorý vytvorili a museli sme určiť mieru znovupoužiteľnosti tohto riešenia. Po stretnutí s členom minuloročného tímu, ktorý sa podieľal na riešení tohto problému, sme zhodnotili, že ich riešenie by bolo znovupoužiteľné, ale len do určitej miery. Vzhľadom na to, že sme sa rozhodli nahradiť ich Mongo databázu relačnou databázou PostgreSQL, bolo by nutné ich riešenie do značnej miery refaktorovať. Rozhodli sme sa preto využiť ich riešenie len ako podklad pre používateľské rozhranie, ktoré bude implementované pomocou GWT a logika aplikácie bude na Java serveri.

6.3 Šprint 3 - Carlsberg

6.3.1 Vyriešiť blokovanie parsovania Google Scholar

Táto úloha zostala počas šprintu 3 otvorená a nevyriešená. Jej riešenie sa presúva do 4. šprintu.

6.3.2 Analýza citačných štýlov

Z dôvodu potreby parsovania referencií z článkov a publikácií, ktoré dané dielo citovali bolo potrebné zanalyzovať súčasný stav citačných štýlov a ich používaní výskumníkmi v rámci Google Scholar. Boli

zanalyzované citačné štýly ISO 690, ISO 690-2, APA 6th, CBE 6th a MLA. Následne bolo zanalyzované, ktoré citačné štýly implicitne Google Scholar ponúka a aké je ich zastúpenie v rôznych publikáciách. Bolo však zistené, že tieto formálne citačné štýly dodržiava len veľmi malé percento autorov. Na výsledkoch tejto analýzy bude stavať vývoj parsera pre získavanie referencií z diel.

6.3.3 ORM na import z XML CrepČ

Po tom, ako sme vytvorili parser pre export súboru XML z databázy CREPČ, bolo potrebné vytvoriť systém, ktorý s týmito dátami vedel pracovať. V tomto bode projektu už bola hotová databáza a pripravená pre vytvorenie objektovo-relačného mapovania. ORM sme vytvorili v programovacom jazyku Java a XML aby bol kompatibilný s Parserom XML CREPČ. Generovanie ORM sme použili technológiu Hibernate. Pri vytváraní ORM sme narazili na problémy, ktoré mali formu chýb v databáze. Kvôli tejto skutočnosti sme ORM museli viackrát generovať a v niektorých prípadoch aj manuálne modifikovať generovaný kód.

6.3.4 Experiment na výber algoritmu na porovnávanie

Pre dobré párovanie je potrebný správny algoritmus. Neexistuje univerzálne použiteľne porovnanie existujúcich prístupov, už len preto, že ich výkon sa môže líšiť od množiny autorov k množine. Bol uskutočnený experiment, podľa ktorého sa vybral Jaro-Winkler párovací algoritmus. Tento experiment prebehol nad syntetizovanými dátami, kde sa vybrali určité mená zo záznamov z CREPČ a boli syntetizované rôzne preklepy týchto mien. Bol implementovaný prototyp algoritmu na párovanie zahŕňajúci porovnanie viacerých atribútov. K otestovaniu tohto algoritmu už nedošlo kvôli komplikáciám s akvizíciou reálnych záznamov. Ďalej bolo riešené získavanie záznamov zo služieb WoS a Scopus, kde boli uskutočnené experimenty a analýzy existujúceho riešenia, aby sme vedeli použiť existujúce kódy na akvizíciu dát.

6.3.5 Vytvorenie ORM pre ukladanie získaných dát z Google Scholar do databázy.

Po úspešnom získaní dát bolo úlohou navrhnúť objektovo - relačné mapovanie do pred pripravenej databázy. Samotný Google Scholar však ponúka len malé percento údajov v porovnaní napríklad z CREPČ. Preto sa budú líšiť záznamy z rôznych zdrojov. Táto úloha však nebola v 3. šprinte úplne dokončená a jej dokončenie bude pokračovať na začiatku nasledujúceho šprintu.

6.3.6 Návrh frontend-u

Pre potreby naštýlovania GWT sme potrebovali v prvom rade vytvoriť návrh používateľského rozhrania v statickej forme HTML + CSS. Z tohto návrhu je hotová prihlasovacia obrazovka, zoznam autorov diel a ich filtrovanie. Vytvorenie kompletného návrhu je závislé od pripravenia prostredia pre GWT. Súčasné riešenie je postačujúce pre otestovanie GWT a ostatné časti používateľského rozhrania sa dokončia keď bude GWT nakonfigurované.

6.3.7 Realizácia základnej funkcionality frontend-u

Pri analýze riešenia minuloročného tímu zameraného na rovnakú problematiku sme narazili na viacero nedostatkov a jedným z nich bol frontend aplikácie. Identifikovaným nedostatkom bolo rozdelenie logiky systému do rôznych databáz, beckendu a frontendu zároveň, z čoho priamo vyplývali nedostatky pri použití návrhového vzoru MVC a to na všetkých úrovniach.

Preto sme sa rozhodli vytvoriť graficky podobný prototyp s ohľadom na odstránenie analyzovaných nedostatkov použitím technológie GWT.

Táto úloha počas šprintu nebola realizovaná a presúva sa do nasledovného šprintu.

6.3.8 Návrh normalizácie dát

Na normalizáciu dát existuje mnoho prístupov, ich správna selekcia pre dané prostredie je kľúčová pre spoľahlivosť párovacích rutín. Cieľom bolo vytvoriť prototyp, ktorý by bol schopný párovania záznamov aspoň v základnej funkcionalite názvu diela a autorov diela. Prototyp bol úspešne implementovaný. Zahŕňa algoritmus párovania záznamov, algoritmus porovnávania reťazcov a algoritmus výberu kandidátov. K jeho testovaniu však zatiaľ kvôli komplikáciám s akvizíciou reálnych údajov nedošlo. Problematické je aj začlenenie prototypu do architektúry systému kvôli nekompletnosti ostatných prvkov a problémov s rôznymi používanými databázami.

6.3.9 Správa používateľov (prihlásenia, registrácie)

Pre potrebu autorizácie používateľov, ktorí budú používať vytváraný systém, bolo potrebné analyzovať možné spôsoby ich reprezentácie v dátovom modeli. Medzi hlavnými požiadavkami správy používateľov bola ich registrácia, podpora skupín používateľov a možnosť priraďovania používateľských práv, či už používateľom alebo skupinám. Tieto požiadavky sa s použitím navrhnutého dátového modelu podarilo splniť spolu s pridaním funkcionality ľubovoľného vytvárania a upravovania skupín používateľov, ako aj priraďovaním používateľov do jednotlivých skupín administrátorom (administrátorom systému alebo konkrétnej organizácie). Použitie takéhoto modelu zjednoduší zmenu používateľských práv ako aj významu jednotlivých skupín používateľov, ak to bude v budúcnosti potrebné.

6.4 Šprint 4 - Duff

6.4.1 Pridanie stĺpca pre indikáciu nespracovaných záznamov

Na zabezpečenie možnosti asynchrónneho spracovania údajov v databáze komparátormi je nutné rozlišovať nové záznamy od už spracovaných. Táto modifikácia databázy túto možnosť zabezpečí. Úloha bola dokončená tak, že bol pridaný nový stĺpec do tabuľky “publication”, ktorý zabezpečuje túto funkcionalitu.

6.4.2 Pridanie tabuľky histórie zmien v DB

Na zabezpečenie vrátiteľnosti zmien v databáze je potrebné pridať tabuľku, ktorá túto možnosť zabezpečí. Úloha bola vyriešená. Táto funkcionalita je súčasťou zmien v databáze na podporu spájania publikácií následkom procesu komparácie.

6.4.3 Pridanie stĺpca zdroja a tabuľky obsahujúcej zdroje

Kvôli úplnosti záznamu a komparácií publikácií je potrebné evidovať aj zdroj publikácie. V databáze je potrebné cez ID odkazovať na zdroj údajov. Táto úloha bola vyriešená. Táto funkcionalita je súčasťou zmien v databáze na podporu spájania publikácií následkom procesu komparácie.

6.4.4 Získanie licencie k Bamboo

Bamboo je ďalší produkt od spoločnosti Atlassian. Je určený na buildovanie a testovanie, spájanie úloh, commitov, výsledkov testov, ale najmä deployovanie na server. Táto funkcionalita pre nás bude užitočná najmä v letnom semestri, kedy budeme potrebovať publikovať výsledok svojej práce veľmi často.

Úlohou v tomto šprinte bolo kontaktovanie Atlassian a získanie akademickej licencie pre tento produkt. Odpoveď nám prišla o týždeň po odoslaní požiadavky a máme k dispozícii licenciu, vďaka ktorej si nainštalujeme tento nástroj.

6.4.5 Vytvorenie prototypu frontendu

Pre účely vytvorenia spustiteľného prototypu frontendu bolo nutné vytvoriť prostredie, kde budú môcť prispievať všetci členovia tímu. Rozhodli sme sa pre technológiu GWT, ktorá ponúka framework pre vývoj

webových rozhraní v prostredí Java a využíva prekladanie do natívneho JavaScriptu a používa asynchrónne neblokujúce volania.

Po vytvorení prvého prototypu sme zistili, že táto technológia nie úplne vhodná pre náš typ projektu a rozhodli sme sa zameniť GWT technológiu za JSP a Servlety s použitím Bootstrap knižnice, ktorá ponúka obohatenú paletu pre návrh dizajnu.

Vytvorením prototypu frontendu pomocou JSP sme odstránili nefunkčné časti GWT prototypu a dosiahli sme jednoduchšiu možnosť vývoja a správy frontendu.

Pomocou spomínaných technológiu sme dokázali vytvoriť jednoduchý prototyp v ktorom bolo umožnená funkcionalita registrácie a prihlasovania používateľov. V prototype bol využitý aj ORM mapovač pomocou, ktorého sa vytvárajú dopyty do k databázovej schéme používateľov a ich práv.

6.4.6 Inštalácia Tomcat servera na arl

Pre umožnenie deployovania iterácii nášho produktu (frontendu) sme na náš server nainštalovali a nakonfigurovali Tomcat server. V súčasnosti naň publikujeme vytvorenú aplikáciu ručne, ale v budúcnosti máme v pláne na túto úlohu využívať Bamboo.

6.4.7 Vytvorenie serverovej časti v GWT package pre správu prihlásenia používateľov

Prvá časť prototypu frontendu odnášala funkcionalitu prihlásenia používateľov, kde sa frontend pomocou ORM prihlásil do Postgre databázy a sprostredkoval prihlásenie a registráciu používateľov.

6.4.8 Realizácia základnej funkcionality frontendu

Medzi základnú funkcionalitu frontendu sme zaradili zobrazenie zoznamu autorov a publikácií. Pre potreby prvého prototypu aplikácie sme túto funkcionalitu implementovali a otestovali. Po niektorých zmenách je frontend pripravený obohatiť minuloročné riešenie o dodatočnú funkcionalitu.

6.4.9 Zistenie možnosti volania C programov z javy

Kvôli rýchlosti boli komparačné algoritmy implementované v C++. Aby sa spojil hlavný program, ktorý je v Jave s komparátormi, je nutné zistiť najvýhodnejšiu metódu volanie C++ programov z javy. Úloha bola vykonaná. Bolo vybrané volanie pomocou CMD a to predovšetkým kvôli udržaniu striktnej samostatnosti komparátorov, veľmi dobrej regulácie používania prostriedkov a kvôli možnosti asynchrónneho spracovania (ktorá v tomto modely je implicitná).s

6.4.10 Definícia komunikačného rozhrania medzi komparátormi a javou

Aby časť riešenia napísaná v Jave mohla komunikovať s komparátormi je potrebné určiť komunikačný protokol, ktorý budú obe strany používať. Analýzou potrieb komparátora sa určil protokol, ktorý pracuje na úrovni publikácií a ich atribútov. Jeho bližší popis je uvedený v dokumentácií C komparátora.

6.4.11 Vytvoriť package v jave pre Scheduler

Súvisí to s plánovaním spúšťania párovania. Scheduler zatiaľ nebol vyriešený. Predovšetkým preto, lebo detailné testovanie komunikácie medzi Javou a Komparátormi malo vyššiu prioritu.

6.4.12 Návrh normalizácie údajov

Úloha bola definovaná v predchádzajúcom šprinte. Výsledkom je pár komparátorov. Prvý je definovaný v jazyku C++ a pracuje s algoritmami porovnávania. Druhý je definovaný v Jave a jeho úlohou je zabezpečiť prístup do databázy, aplikovať algoritmy na výber kandidátov a volať C++ komparátor. Úloha nebola dokončená a to hlavne z dôvodu pokračujúceho testovania, keďže najdôležitejšia vlastnosť tohto systému má byť spoľahlivosť.

7 Použitá literatúra

[1] CARVALHO, A. P. DE et al. 2011. Incremental Unsupervised Name Disambiguation in Cleaned Digital Libraries. In *Journal of Information and Data Management.* ISSN 2178-7107, 2011, vol. 2, no. 3, p. 289–304.

[2] BHATTACHARYA, I. et al. 2006. A Latent Dirichlet Model for Unsupervised Entity Resolution. In *Proceedings of the Sixth SIAM International Conference on Data Mining*. Bethesada: SIAM, 2006. ISBN 978-089871611-5, p. 47-58.

[3] CULOTTA, A. et al. 2007. Author Disambiguation using Error-driven Machine Learning with a Ranking Loss Function. In *AAAI Workshop - Technical Report.* Vancouver: AAAI, 2007. ISBN 978-157735341-6, p. 32-37.

[4] FERREIRA, A. A. et al. 2010. Effective Self-Training Author Name Disambiguation in Scholarly Digital Libraries. In *Proceedings of the 10th annual joint conference on Digital libraries.* New York: ACM, 2010. ISBN 978-1-4503-0085-8, p. 39-48.

[5] HERNANDEZ, M. A. 1995. The Merge/Purge Problem for Large Databases. In *Proceedings of the 1995 ACM SIGMOD international conference on Management of data*. New York: ACM, 1995. ISBN 0-89791-731-6, p. 127-138.

[6] VRIES, T. DE. 2011. Robust Record Linkage Blocking Using Suffix Arrays

and Bloom Filters. In *Transactions on Knowledge Discovery from Data (TKDD).* ISSN 1556-4681, 2011, roč. 5, č. 2, čl. 9

[7] Portál CREPČ - Centrálny register publikačnej činnosti a Centrálny register umeleckej činnosti.[Online] [Dátum: 5. 9 2013.]

http://cms.crepc.sk/Data/Sites/1/pdfsubory/docword2.5.pdf.