Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Igor Vinojčić

Analiza in vizualizacija karierne poti diplomantov FRI

diplomsko delo

univerzitetni študij računalništva in informatike

Mentor: doc. dr. Dejan Lavbič

Ljubljana 2013

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Univerza *v Ljubljani*
Fakulteta *za računalništvo in informatiko*

Tržaška 25
1001 Ljubljana, Slovenija
telefon: 01 476 84 11
01 476 83 87
faks: 01 426 46 47
01 476 87 11
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 01946/2013
Datum: 02.09.2013

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **IGOR VINOJČIĆ**

Naslov: **ANALIZA IN VIZUALIZACIJA KARIERNE POTI DIPLOMANTOV FRI**

ANALYSIS AND VIZUALIZATION OF FRI GRADUATES CARRIER PATH

Vrsta naloge: Diplomsko delo univerzitetnega študija

Tematika naloge:

V okviru diplomske naloge naj študent z integracijo podatkov iz digitalne knjižnice ePrints.FRI in podatki iz profesionalnega družabnega omrežja LinkedIn izvede analizo karierne poti diplomantov FRI in njihove vpetosti v slovenski gospodarski prostor. Podatki iz vira ePrints vključujejo podatke o raziskovalnih nalogah, diplomskih nalogah, magistrskih delih in doktorskih disertacijah. Na voljo so podatki o vsebini, ključnih besedah, avtorju, mentorju in področju. Po drugi strani pa so na omrežju LinkedIn na voljo podatki o karierni poti uporabnika - njegove delovne izkušnje, projektno delo in trenutni status. V okviru diplomske naloge bi bilo tako potrebno pregledati kako uspešni so diplomanti FRI po svojem zaključku študija in kakšno vlogo imajo v podjetjih. Odpirajo se tudi številne druge možnosti analize - pregled po mentorjih, pregled po vsebinskih področjih diplomskih del, povprečen čas po diplomiranju za dosego napredovanja ipd.

Mentor:

doc. dr. Dejan Lavbič

Fakulteta za računalništvo in informatiko
2
UNIVERZA V LJUBLJANI

Dekan:

prof. dr. Nikolaj Zimic

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Igor Vinojčić, z vpisno številko **63080157**, sem avtor diplomskega dela z naslovom:

Analiza in vizualizacija karierne poti diplomantov FRI

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom doc. dr. Dejana Lavbiča,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki "Dela FRI".

V Ljubljani, dne 22. septembra 2013 Podpis avtorja:

Zahvaljujem se svojim staršem za podporo in motivacijo v celotnem obdobju študija.

Doc. dr. Dejanu Lavbiču, za strokovno pomoč in koristne nasvete med izdelavo diplomskega dela.

Anji, za razumevanje in pozitivno energijo s katero sem lažje prišel do želenega cilja.

Prijateljem, ki so obdobje študija popestrili in Macotu za tiskarske napake.

Kazalo

Seznam uporabljenih kratic in simbolov

API	(angl. Application Programming Interface) programski vmesnik
URL	(angl. Uniform resource locator) naslov strani na svetovnem spletu
DOM	(angl. Document Object Model) konvencija za predstavitev in interakcijo z objekti HTML, XHTML in XML dokumentov
JSON	(JavaScript Object Notation) standard za izmenjavo podatkov
SQL	(angl. Structured Query Language) povpraševalni jezik za delo s podatkovnimi zbirkami
HTML	(angl. HyperText Markup Language) označevalni jezik namenjen izdelavi spletnih strani
XML	(angl. Extensible Markup Language) označevalni jezik namenjen opisovanju strukturiranih podatkov
WHATWG	(angl. Web Hypertext Application Technology Working Group) skupnost ljudi z interesom razvijanja HTML tehnologij

Povzetek

Posledica hitrega napredka tehnologije in svetovnega spleta je razvoj številnih spletnih socialnih omrežjih, ki so v relativno kratkem času postala zelo priljubljena. Omrežje LinkedIn je eno izmed spletnih socialnih omrežjih, ki je namenjeno poslovnim uporabnikom, katerim omogoča vzdrževanje poslovnih stikov in predstavitev informacij o kariernih poteh, izobrazbi, projektnem delu, veščinah ipd. Fakulteta za računalništvo in informatiko hrani podatke o diplomskih delih diplomantov v digitalni knjižnici ePrints.FRI. V sklopu diplomskega dela smo razvili program za pridobivanje podatkov o diplomskih delih diplomantov FRI in njihovih kariernih poteh ter zgradili podatkovno bazo, ki bo omenjene podatke hranila in omogočila analizo, s katero smo želeli določiti vpetost diplomantov FRI v gospodarski prostor. Program in podatkovna baza sta na voljo na BitBucket strežniku na naslovu *https://bitbucket.org/igorithm/diploma*.

Ključne besede

karierna pot, LinkedIn, eprint, analiza, diplomant

Abstract

Due to fast development of information technology and World Wide Web in the past years, many social networking websites were developed. In relative short period of time social networks gained a lot of popularity. LinkedIn is one of the social networking websites for people in professional occupations wanting to manage their business contacts and share information about their jobs, education, skills, project work etc. On the other hand, Faculty of information and computer science keeps data about their graduate's bachelor theses in digital library ePrints.FRI. We developed program for retrieving graduate's bachelor theses data and their career path data. Further, we designed relational database for storing retrieved data and enabling analysis, in which we determine graduate's integration in economic area. Program and database are located on BitBucket server *https://bitbucket.org/igorithm/diploma*.

Keywords

career path, LinkedIn, eprint, analysis, graduate

Poglavje 1

Uvod

Napredek na področju tehnološke opreme in svetovnega spleta je omogočil razvoj različnih načinov komunikacije, zabave in povezljivosti uporabnikov ne glede na geografsko oddaljenost. Z večanjem števila uporabnikov svetovnega spleta se je večala potreba po storitvah, ki gradijo socialne odnose med uporabniki, ki imajo skupne interese, ideje, se udeležujejo skupnih dogodkov in aktivnosti. Tako so nastala spletna socialna omrežja, ki zagotavljajo sredstva za interakcijo med uporabniki. Socialna omrežja so v razmeroma kratkem času postala med uporabniki izjemno popularna. V Sloveniji je leta 2011 imelo 60% uporabnikov oblikovan profil na vsaj enem od spletnih socialnih omrežjih[1]. Posledica tega je, da je na spletu na voljo ogromna količina podatkov o uporabnikovem zasebnem in profesionalnem življenju.

LinkedIn[5] je poslovno usmerjeno socialno omrežje z več kot 200 milijonov uporabnikov. Namenjeno je uporabnikom, ki želijo vzdrževati poslovne stike in predstaviti informacije o lastni izobrazbi, spretnostih, izkušnjah, projektnemu delu ipd. Na drugi strani Fakulteta za računalništvo in informatiko (FRI) ohranja prosto dostopno digitalno knjižnico ePrints.FRI[4], ki hrani podatke o diplomskih delih diplomantov, doktorskih disertacijah, publikacijah in magisterskih delih. V interesu Fakultete za računalništvo in informatiko je, da bi njeni diplomanti v gospodarskem prostoru bili čim bolj uspešni in učinkoviti, zato bomo v diplomskem delu na podlagi podatkov

iz omrežja LinkedIn in knjižnice ePrints.FRI skušali analizirati karierne poti diplomantov FRI in določiti njihovo vpetost v gospodarski prostor.

Za rešitev problema določitve vpetosti diplomantov FRI v gospodarski prostor smo razvili program v programskem jeziku Java, ki skrbi za pridobivanje potrebnih podatkov, in relacijsko podatkovno bazo MySQL, ki bo pridobljene podatke hranila in s poizvedovanjem omogočila analizo podatkov.

V poglavju 2 so opisane uporabljene tehnologije in razvojna orodja s pomočjo katerih smo izluščili in analizirali podatke. V nadaljevanju (poglavje 3) sledi podrobnejši opis razvoja programa za pridobivanje podatkov in načrtovanja relacijske podatkovne baze MySQL. V tem poglavju so prav tako opisane težave, s katerimi smo se srečali tekom razvoja programa.

Sledi poglavje 4, ki opisuje in vizualizira dobljene rezultate analize podatkov.

Na koncu, v poglavju 5, so v zaključku zapisane sklepne ugotovitve diplomskega dela in smernice za nadaljne delo.

Poglavje 2

Raziskovalna področja in uporabljene tehnologije

Z analizo karierne poti diplomantov FRI želimo določiti njihovo vpetost v gospodarski prostor, kar je tudi glavni problem diplomske naloge. Rešitev tega problema zahteva implementacijo programa za pridobivanje podatkov o diplomskih nalogah in kariernih poteh diplomantov ter izvedbo analize podatkov.

V tem poglavju bomo opisali katere tehnologije bomo uporabili za rešitev poglavitega problema in iz katerih podatkovnih virov bomo črpali potrebne podatke.

2.1 Seznanitev z ePrints.FRI in LinkedIn

2.1.1 ePrints.FRI

Eprint[3] je izraz za digitalni zapis strokovno pregledanega raziskovalnega dela, pred ali po zagovoru. Po uspešnem strokovnem pregledu se delo pretvori v digitalno obliko in shrani v prosto dostopno digitalno knjižnico. V nadaljevanju bomo z izrazom *eprint* označevali digitalni zapis diplomske naloge. Kadar se bomo z izrazom *eprint* sklicevali na raziskovalno delo, ki ni diplomsko delo, bo to eksplicitno poudarjeno.

Prosto dostopna digitalna knjižnica je zgrajena s pomočjo paketa programske opreme imenovanega *EPrints*. Ne tem mestu je potrebno poudariti, da ne smemo zamenjevati *eprints*, ki je množica *eprintov*, z *EPrints*, ki je paket programske opreme.

EPrints.FRI[4] je prosto dostopna digitalna knjižnica Fakultete za računalništvo in informatiko, ki vključuje podatke o raziskovalnih nalogah, diplomskih nalogah, magisterskih delih in doktorskih disertacijah. Slika 2.1 prikazuje domačo spletno stran ePrints.FRI, kjer so na voljo podatki o vsebini, ključnih besedah, avtorju, datumu, mentorju in področju. Po vseh omenjenih podatkih je omogočeno iskanje kakor tudi pregledovanje seznamov vseh eprintov določenega sklopa.

Slika 2.1: ePrints.FRI

Na sliki 2.2, ki prikazuje del seznama vseh eprintov diplomskih nalog,

je z rdečo barvo označena funkcija *izvoz*. Funkcija omogoča izvoz podatkov eprintov iz seznama v mnogo različnih formatov, kot sta format XML in JSON, kar je zelo priročno v našem primeru, saj nas zanimajo podatki o eprintih vseh FRI diplomantov. Funkcija za izvoz je naslov URL, kjer se nahajajo strukturirani podatki.

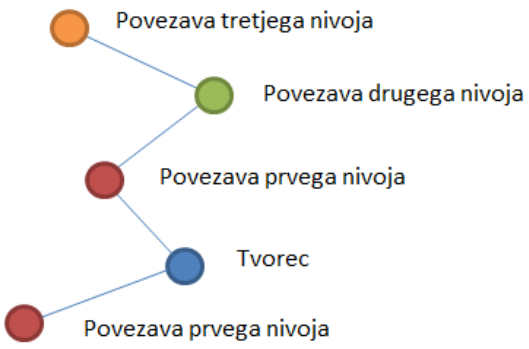
Slika 2.2: Del seznama vseh eprintov diplomskih nalog in funkcija izvoz.

2.1.2 LinkedIn

LinkedIn[5] je poslovno usmerjeno spletno družbeno omrežje z več kot 200 miljoni uporabnikov. Namenjeno je profesionalcem, ki se želijo na spletu predstaviti, povezovati, vzdrževati poslovne stike, iskati službe, primeren kader ipd. Uporabnikom omogoča odprtje računa in predstavitev informacij v že predefinirane kategorije kot so izobrazba, spretnosti, izkušnje, projektno delo itd. Informacije o uporabniku so predstavljene na pripadajoči spletni strani LinkedIn omrežja imenovani *profil uporabnika*.

Ena glavnih funkcionalnosti omrežja je že omenjena možnost medsebojne povezljivosti uporabnikov. Vsak uporabnik lahko vzdržuje lasten seznam uporabnikov, s katerimi je povezan, in tvori svoje lastno omrežje. Iz vidika tvorca omrežja je uporabnik, ki je del tvorjenega omrežja, imenovan *povezava (ang. connection)* in je lahko treh vrst:

- *povezava prvega nivoja (ang. first degree connection)* je uporabnik, ki ima s tvorcem omrežja direktno povezavo,
- *povezava drugega nivoja (ang. second degree connection)* je uporabnik, ki je povezan z vsaj eno tvorčevo povezavo prvega nivoja,
- *povezava tretjega nivoja (ang. third degree connection))* je uporabnik, ki je povezan z vsaj eno tvorčevo povezavo drugega nivoja.



Slika 2.3: Omrežje in vrste povezav.

LinkedIn na podlagi vrste povezave določa, kateri podatki profila so vidni. Na tem mestu velja omeniti, da LinkedIn ponuja iskalnik, s pomočjo katerega lahko iščemo povezave, uporabnike, podjetja itd. Pravila[6] velevajo, da se pri iskanju po imenu in priimku prikaže polni profil uporabnika, ne glede na vrsto povezave . To velja tudi, če uporabnik ni povezava.

Do podatkov uporabnika lahko dostopamo tudi preko aplikacijskega vmesnika LinkedIn API, ki je namenjen vsem razvijalcem. Podroben opis vmesnika LinkedIn API je izven okvirov diplomske naloge. Za nas sta bistvena dva podatka, in sicer:

- Pridobivanje podatkov o *vseh* zaposlitvah uporabnika je težavno ne glede na vrsto povezave. V dokumentaciji[17] vmesnika API ni jasno določeno ali je mogoče pridobiti vse, torej tudi pretekle, zaposlitve. Glede te težave je prav tako veliko pritožb s strani uporabnikov.

- Aplikacija, ki želi dostopati do vmesnika API, mora biti avtorizirana v okvirih standarda *OAuth*. To pomeni, da mora razvijalec aplikacije poskrbeti za implementacijo *OAuth 1.0* ali *OAuth 2.0* protokola.

Zaradi teh dveh podatkov se bomo uporabi vmesnika LinkedIn API skušali izogniti, saj so podatki o zaposlitvah ključni za določitev vpetosti FRI diplomantov v gospodarski prostor, medtem ko lahko implementacija OAuth protokola izrazito poveča kompleksnost rešitve.

2.2 Programsko okolje

Za implementacijo programa, s pomočjo katerega bomo skušali rešiti problem določitve vpetosti diplomantov FRI v gospodarski prostor, bomo uporabili programski jezik Java[21]. Gre za programski jezik, ki je zelo razširjen med razvijalci in ima dobro napisano dokumentacijo. Prav tako je zanj napisanih veliko knjižnic, ki bistveno olajšajo rešitve določenih problemov.

Uporaba programskega jezika Java zahteva prenos knjižnic in njihovo vključevanje v projekt, kar zna biti zamudno. S tehnologijo *Maven*[22] bomo upravljali z Java knjižnicami bolj učinkovito in pregledno.

Program bomo razvijali v razvojnem orodju Eclipse. Prednost razvojnega orodja Eclipse je v tem, da je namenjeno razvoju integriranih tehnologij, kar v našem primeru občutno poenostavlja vključitev tehnologije Maven.

2.2.1 Maven

Maven[22] uporablja datoteko v formatu XML za opis projekta, njegovih odvisnosti in ostalih zunanjih modulov ter komponent. V našem primeru ga bomo uporabljali za prenos potrebnih knjižnic. Maven uporablja centralni repozitorij, iz katerega sname knjižnice in zgradi potrebne odvisnosti znotraj projekta, tako je knjižnica takoj na voljo. Proces vključitve potrebnih knjižnic je poenostavljen, potrebno je le na spletni strani repozitorija Maven[23] poiskati pripadajoči opis knjižnice v formatu XML in ga vključiti v datoteko *pom.xml* glavnega projekta. Primer opisa knjižnice *jsoup* v formatu XML:

```
<dependency>
        <groupId>org.jsoup</groupId>
        <artifactId>jsoup</artifactId>
        <version>0.2.2</version>
</dependency>
```

Podrobnejša razlaga uporabe tehnologije Maven presega okvirje diplomske naloge, ker njena vključitev ne vpliva neposredno na rešitev problema in bi jo brez posledic lahko izpustili.

2.3 Luščenje podatkov iz strukturiranih spletnih virov

V podpoglavju 2.1.1 smo ugotovili, da lahko iz digitalne knjižnice ePrints.FRI izvozimo strukturirane podatke o diplomskih nalogah. Podatke je mogoče izvoziti v formate, kot sta EP3-XML in JSON. Podatke bomo izvozili v format EP3-XML, ki je v bistvu format XML, saj med njima ni očitnih in za nas pomembnih razlik. Nato bomo podatke izluščili s pomočjo procesov *serializacije* in *deserializacije*.

2.3.1 Proces serializacije in deserializacije

Serializacija[7] je proces, s katerim prevedemo objekte in podatkovne strukture v format, ki ga lahko shranimo in ponovno uporabimo. S tem procesom torej zapišemo objekte v podatkovni tok. *Deserializacija* je nasproten proces od serializacije, saj nam omogoča pretvorbo toka podatkov v objekte. V našem primeru bomo vsebino, zapisano v formatu XML in JSON, deserializirali v objekte programskega jezika JAVA.

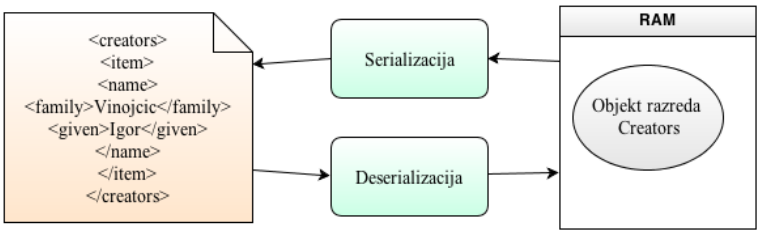
Serializacija in deserializacija sta v programskem jeziku JAVA podprti z večimi knjižnicami. V našem primeru bomo uporabili knjižnico *JAXB*[18] za deserializacijo formata XML in knjižnico *Gson*[19] za deserializacijo formata JSON.

2.3.2 JAXB

JAXB[18] je kratica za ***J**ava **A**rchitecture for **X**ML **B**inding* in omogoča preslikovanje med Java objekti in strukturo XML (v obe smeri). Knjižnica JAXB uradno za preslikovanje med objekti in strukturo XML uporablja procese *zbiranja (ang. marshal).* Čeprav dokument RFC 2713[13] ločuje med serializacijo in zbiranjem, naša definicija serializacije dobro opisuje proces preslikave in jo bomo v nadaljevanju uporabljali kot sinonim za zbiranje.

S pomočjo tehnike anotacije[20] označimo Java razrede v skladu s shemo XML in tako omogočimo preslikavo elementov strukture XML v prave objekte razredov ter njegove atribute. Med starševskim elementom in otrokom mora biti direktna povezava. Povezava je vzpostavljena tako, da anotirani razred starša vsebuje referenco na anotirani razred otroka. Tako proces deserializacije ustvari množico med seboj povezanih Java objektov.

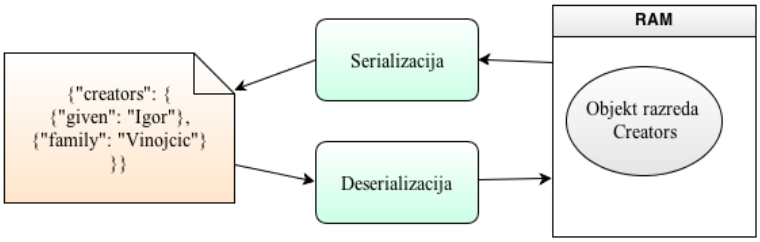
Knjižnica JAXB prav tako omogoča, da preslikamo samo tiste elemente formata XML, ki jih potrebujemo.



Slika 2.4: Primer deserializacije vsebine XML v Java objekte.

2.3.3 Gson

Gson[19] je odprto kodna Java knjižnica, ki omogoča deserializacijo in serializacijo formata JSON. Gson omogoča, da preslikamo samo tista polja strukture JSON, ki jih potrebujemo. Princip preslikovanja je podoben kot pri knjižnici JAXB, le da knjižnica Gson ne uporablja anotiranja, ampak je preslikovanje določeno z imeni razredov in njegovih atributov. Prav tako je potrebno poskrbeti, da med starševskim poljem in otrokom obstaja direktna povezava, če želimo v procesu deserializacije dobiti med seboj povezane objekte.



Slika 2.5: Primer deserializacije vsebine v formatur JSON v Java objekte.

2.4 Luščenje podatkov iz delno strukturiranih spletnih virov

V podpoglavju 2.1 smo ugotovili, da na omrežju LinkedIn lahko najdemo podatke o kariernih poteh uporabnikov. LinkedIn omogoča dostop do struk-

turiranih podatkov preko vmesnika LinkedIn API, vendar se bomo njegovi uporabi izognili zaradi omenjenih razlogov v podpoglavju 2.1. Edina možnost je torej luščenje delno strukturiranih podatkov.

Svetovni splet danes vsebuje ogromno delno strukturiranih podatkov, posledica tega je tudi veliko razvitih knjižnic, ki omogočajo učinkovito luščenje podatkov. Uporabili bomo prosto dostopno Java knjižnico *jsoup*[8].

Knjižnica jsoup zadostuje, kadar je podatkovni vir neposredno določljiv. Preden podatke izluščimo, jih je seveda potrebno najti. Potrebno se je zavedati, da LinkedIn lahko uporabljamo le ob uspešni prijavi v sistem, šele takrat lahko dostopamo do profilov povezav in drugih uporabnikov. To pomeni, da problem zahteva interakcijo s spletnimi stranmi omrežja LinkedIn. Knjižnica jsoup[8] v osnovi ne podpira izvajanja funkcij za potrebe interakcije, zato bomo v kombinaciji z jsoup uporabili Java knjižnico *HtmlUnit*[9].

2.4.1 Jsoup

Jsoup[8] je prosto dostopna Java knjižnica za delo s HTML dokumenti. Jsoup omogoča razčlenitev dokumenta HTML v DOM strukturo in deluje po WHATWG HTML5 specifikaciji. Z uporabo metod, ki upoštevajo DOM, CSS in jquery principe programiranja, lahko luščimo in manipuliramo podatke.

Dokument HTML lahko razčlenimo iz URL, datoteke ali niza podatkov. Jsoup prav tako omogoča manipuliranje z elementi HTML, atributi in teksti.

2.4.2 HtmlUnit

HtmlUnit[9] je prosto dostopna Java knjižnica za delo s HTML dokumenti, ki preko vmesnika API nudi metode za simulacijo funkcij brskalnikov. HtmlUnit si lahko predstavljamo kot brskalnik brez uporabniškega vmesnika, ki podpira osnovne funkcije, kot so kliki gumbov, izpoljevanje form ipd. Nudi tudi JavaScript podporo, ki pa ni popolna. HtmlUnit je običajno uporabljen v namene testiranja ali pridobivanja informacij iz spletnih strani.

2.5 Shranjevanje podatkov

Za potrebe analize je potrebno pridobljene podatke o diplomskih delih in kariernih poti diplomantov trajno shraniti. Podatke bomo shranili v relacijsko bazo MySQL.

2.5.1 Relacijska podatkovna baza

Relacijska podatkovna baza[14] je množica tabel, ki so formalno opisane in organizirane po relacijskem modelu.

Relacijski podatkovni model temelji na teoriji množic in predikatni logiki. Osnovni gradnik modela je relacija, ki je predstavljena s tabelo. Vsaka relacija je sestavljena iz neprazne množice stolpcev in množice podatkovnih vrstic, ki je lahko tudi prazna. Stolpci so opisani z relacijsko shemo in se imenujejo *atributi*.

Relacijska shema določa imena atributov, pripadajočo domeno in identificira primarne ter tuje ključe. Domena predstavlja zalogo vrednosti, ki jih atribut lahko zavzame. Formula 2.1 predstavlja relacijsko shemo R z atributoma A1 in A2 ter domenama D1 in D2:

$$Sh(r) = R(A1 : D1, A2 : D2) \tag{2.1}$$

Primarni ključ je neprazna množica atributov, ki enolično določa vsako podatkovno vrstico. Primer primarnega ključa je številka EMŠO, ki enolično določa državljana Republike Slovenije. Podatkovne vrstice neke relacije so lahko v razmerju s podatkovnimi vrsticami druge relacije preko *tujega ključa*. Tuji ključ je neprazna množica atributov v neki relaciji, ki je primarni ključ v drugi relaciji.

Razmerij med relacijami je več vrst, ker so razmerja lahko različne številnosti. Značilne številnosti razmerij so (relacija A je v razmerju z relacijo B):

- $1 : 1$ (ena proti ena): Podatkovna vrstica relacije A je v razmerju z eno podatkovno vrstico relacije B in obratno. Primer je razmerje med

državljanom in potnim listom. Državljan lahko ima samo en veljaven potni list in en potni list predstavlja natanko enega državljana.

- $1 : N$ (ena proti mnogo): Podatkovna vrstica relacije A je v razmerju z eno ali več podatkovnimi vrsticami v relaciji B, medtem ko je podatkovna vrstica relacije B v razmerju z eno podatkovno vrstico relacije A. Primer je razmerje med nogometnim moštvom in igralcem, kjer je moštvo sestavljeno iz večih igralcev, igralec pa je del samo enega moštva.

- $M : N$ (mnogo proti mnogo): Podatkovna vrstica relacije A je v relaciji z eno ali več podatkovnimi vrsticami relacije B in obratno. Primer je razmerje med študentom in predmetom. Študent ima v času študija lahko enega ali več predmetov in eden ali več študentov se udeležuje enega predmeta.

Pri načrtovanju podatkovne baze je potrebno biti pozoren, da baza ne vsebuje preveč redundantnih podatkov in nima odvečnih razmerij, kar lahko povzroči performančne težav. Redundanco in število razmerij lahko minimiziramo s tehniko *normalizacije*[15]. Normalizacija procesira in organizira relacije ter atribute relacij. Ponavadi proces normalizacije vključuje razbitje relacije na več manjših relacij in definicijo razmerij med novo nastalimi relacijami. Poznamo več soodvisnih oblik normalizacije, in sicer:

- Prva normalna oblika: Relacija je v prvi normalni obliki, če ne vsebuje ponavljajočih se atributov ali skupin atributov.

- Druga normalna oblika: Relacija je v drugi normalni obliki, če je v prvi normalni obliki in so vsi atributi relacije odvisni od celotnega ključa. V relaciji ni nobenega atributa, ki je odvisen samo od dela ključa.

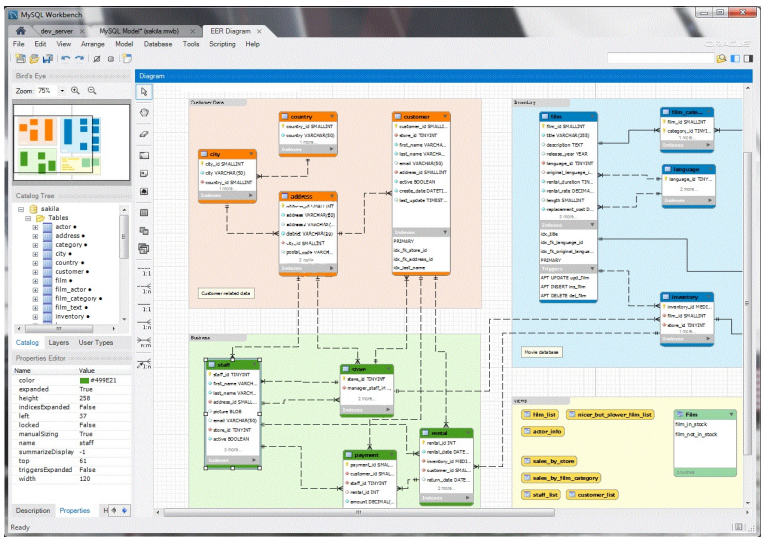
- Tretja normalna oblika: Relacija je v tretji normalni obliki, če je v drugi normalni obliki in noben atribut ni tranzitivno odvisen od ključa. V relaciji ne nastopajo atributi, ki so odvisni od drugega atributa v relaciji, ki ni del ključa.

Obstaja še več normalnih oblik, vendar za rešitev našega problema niso relevantne.

2.5.2 MySQL in MySQL Workbench

MySQL[10] je zelo razširjen odprto kodni sistem za upravljanje z relacijskimi podatkovnimi bazami. Predvsem je priljubljen na področju razvoja spletnih aplikacij. MySQL za upravljanje s podatki uporablja jezik 4. generacije SQL[12]. Funkcije sistema MySQL so uporabnikom dostopne preko ukazne vrstice ali grafičnih vmesnikov, kot sta Toad[2] in MySQL Workbench[11].

Uporabili bomo grafični vmesnik MySQL Workbench[11], saj v primerjavi z ukazno vrstico olajša uporabo funkcij za administracijo in upravljanje s podatkovno bazo. Grafični vmesnik MySQL Workbench omogoča izdelavo logičnega modela podatkovne baze s pomočjo diagramske tehnike imenovane *entitetno-relacijski diagram* (diagram ER), kar se v našem primeru izkaže za zelo priročno funkcijo, saj lahko iz logičnega modela direktno zgradimo relacijsko podatkovno bazo. Slika 2.6 prikazuje grafični vmesnik MySQL Workbench v modulu za načrtovanje podatkovne baze.



Slika 2.6: MySQL Workbench načrtovanje relacijske podatkovne baze.

2.5.3 Java connector

Mysql-connector-java[25] je Java knjižnica, ki omogoča vzpostavitev povezave z MySQL strežnikom. Po vzpostavljeni povezavi je s pomočjo knjižnice mogoče izvajanje ukazov SQL za ustvarjanje in spreminjanje podatkovnega modela. Knjižnica omogoča tudi poizvedovanje po podatkih baze. Rezultate poizvedbe preslika v Java objekte, s čimer je omogočeno enostavno manipuliranje z rezultati poizvedb.

Poglavje 3

Razvoj programa za pridobivanje podatkov

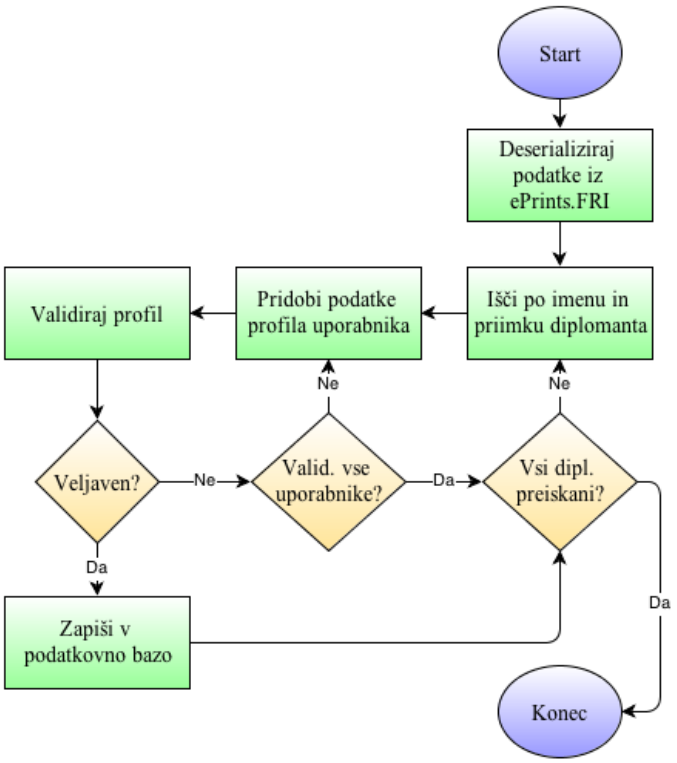
3.1 Zasnova programa

V podpoglavju 2.1 smo ugotovili, da bomo podatke o eprintih diplomantov izluščili iz digitalne knjižnice ePrints.FRI, podatke o kariernih poteh pa iz omrežja LinkedIn.

Strukturirane podatke o eprintih diplomantov lahko pridobimo v eni zahtevi. Podatke bomo s pomočjo deserializacije (glej podpoglavje 2.3.1) preslikali v Java objekte.

Po izvedbi deserializacije bo vsak eprint imel pripadajoči Java objekt. V iteraciji se bomo sprehodili skozi množico deserializiranih eprintov in za vsakega poiskali pripadajoči LinkedIn profil, če ta obstaja. Vsak deserializiran eprint vsebuje podatke o imenu in priimku kreatorja diplomskega dela. Program bo prebral ime in priimek in na omrežju LinkedIn izvedel iskanje.

LinkedIn iskalnik[16] vrača množico rezultatov, katerih relevantnost določa na podlagi mnogih pogojev. V osnovi to pomeni, da od iskalnika ne moremo pričakovati, da bo vedno vrnil samo pravi rezultat. Prav tako ne moremo pričakovati, da bo vedno vrnil prazno množico rezultatov, če iskani uporabnik ne obstaja. Potrebno se je zavedati, da lahko imajo ljudje enake ali podobne



Slika 3.1: Osnovna programska logika pridobivanja podatkov.

priimke in imena. Ker od iskalnika lahko pričakujemo množico večih rezultatov moramo med prejetimi rezultati izluščiti pravega. Problem *validacije rezultatov* od nas torej zahteva implementacijo modula, ki bo skrbel za iskanje veljavnega profila. Pred validacijo je potreben še en dostop do omrežja LinkedIn, saj rezultati ne vsebujejo profila uporabnikov, temveč samo naslov URL do profila.

Če bo program našel veljaven profil diplomanta bo sledilo luščenje potrebnih podatkov iz profila. Na koncu bo program v podatkovno bazo zapisal podatke eprinta in pripadajočega profila. Po vnosu v podatkovno bazo bo program nadaljeval z iteracijo, dokler nebo izvedel iskanje za vsak deserializiran eprint.

Slika 3.1 povzema programsko logiko za rešitev problema pridobivanja podatkov.

3.2 Razvoj podatkovne baze

3.2.1 Zahteve

Podatkovna baza mora podpirati naslednje poizvedbe:

1. Koliko diplomantov FRI je svojo karierno pot začelo na delovnem mestu x?
2. Koliko diplomantov FRI, ki so imeli mentorja y, je svojo karierno pot začelo na delovnem mestu x?
3. Koliko diplomantov FRI, ki je diplomiralo leta z, je svojo karierno pot začelo na delovnem mestu x?
4. Koliko diplomantov FRI je svojo karierno pot začelo v podjetju k?
5. Koliko diplomantov FRI, ki je diplomiralo leta z, je svojo karierno pot začelo v podjetju p?
6. Koliko diplomantov FRI, ki so imeli mentorja y, je svojo karierno pot začelo v podjetju p?
7. Koliko diplomantov FRI je v svoji karierni poti bilo zaposlenih na delovnem mestu x?
8. Koliko diplomantov FRI, ki so imeli mentorja y, je v svoji karierni poti bilo zaposlenih na delovnem mestu x?
9. Kakšen je povprečen čas potreben za napredovanje diplomantov FRI?
10. Kakšen je povprečen čas potreben za napredovanje diplomantov FRI, ki so imeli mentorja y?
11. Kakšen je povprečen čas potreben za napredovanje diplomantov FRI, ki so diplomirali leta z?
12. Koliko diplomantov FRI je imelo karierno pot enako karierni poti e?

Da bi podprli zgoraj omenjene poizvedbe moramo imeti podatke o mentorjih, diplomskih delih (leto diplomiranja), diplomantih, delovnih mestih diplomantov ter o podjetjih, kjer so diplomanti zaposleni.

3.2.2 Entitetno-relacijski diagram

Na ePrints.FRI lahko najdemo podatke o imenu in priimku mentorja, imenu in priimku diplomanta, naslovu diplomskega dela, letnice ustnega zagovora ter podatke o ključnih besedah diplomskega dela. Podatke o diplomskih delih ne želimo imeti shranjene v eni tabeli. S pomočjo normalizacije želimo podatke o diplomskih delih porazdeliti v več medsebojno povezanih tabel. Tako dobimo tabelo *Mentor*, ki predstavlja mentorja, tabelo *Graduate*, ki predstavlja diplomanta in tabelo *Degree*, ki predstavlja diplomsko delo.

V našem primeru lahko ima en primerek diplomskega dela samo enega mentorja. Zaradi enostavnosti ne bomo upoštevali somentorja. Po drugi strani je lahko mentor v relaciji z večimi diplomskimi deli, zato je povezava med tabelo Mentor in Degree 1:M (1 proti mnogo). Vsako diplomsko delo ima natanko enega diplomanta. Prav tako ima vsak diplomant samo eno diplomsko delo. Čeprav bi v realnosti lahko imel diplomant dve diplomi ene Fakultete, so ti primeri izjemno redki. Povezava med tabelo Graduate in tabelo Degree je tako 1:1 (ena proti ena). Na voljo so še podatki o ključnih besedah. Vsaka diplomska naloga lahko ima več ključnih besed, vsaka ključna beseda pa lahko pripada večim diplomskim nalogam. Ker bomo v prihodnosti verjetno želeli poizvedovati po specifičnih ključnih besedah, je smiselno ključne besede shranjevati v posebno tabelo imenovano *Keywords*. Tako je povezava med tabelo Degree in Keywords N:M (mnogo proti mnogo). Ker s pomočjo orodja MySQL Workbench načrtujemo *logični* podatkovni model, moramo med tabelo Degree in Keywords ustvariti tabelo *DegreeKeyword*, ki predstavlja povezavo med tabelama Degree in Keywords.

Na omrežju LinkedIn lahko najdemo podatke o delovnem mestu, kakšen je naziv delovnega mesta in v katerem podjetju je bil diplomant zaposlen. Tako dobimo 3 tabele: tabelo *Work*, ki predstavlja delovno mesto, tabelo

Company, ki predstavlja podjetje delovnega mesta in tabelo *Position*, ki predstavlja naziv delovnega mesta. Vsak diplomant lahko ima enega ali več delovnih mest, vsako delovno mesto pa natanko enega diplomanta. To sicer zveni čudno, vendar v našem primeru želimo obravnavati vsako delovno mesto diplomanta posebej. Tako je povezava med tabelo Graduate in Work 1:M (ena proti mnogo).

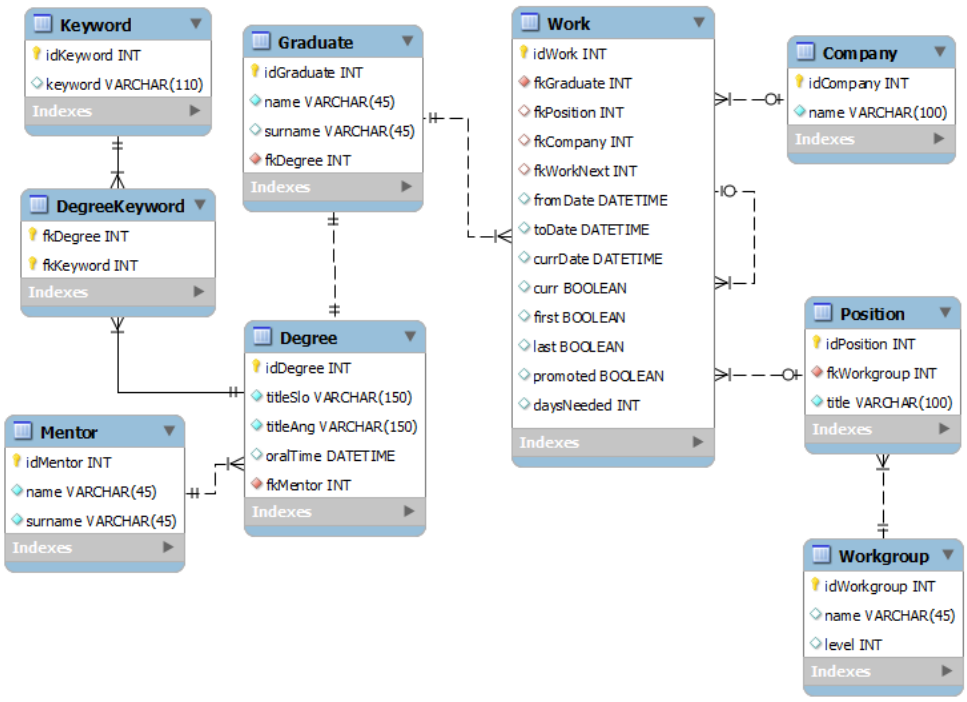
Tabela Work ima nekaj posebnih atributov. To so:

- fromDate: datum začetka zaposlitve.
- toDate: datum konca zaposlitve. Ta datum je prazen, če gre za sedanjo zaposlitev.
- currDate: trenutni datum. Ta datum je prazen, če gre za preteklo zaposlitev.
- curr: če gre za sedanjo zaposlitev diplomanta, ima ta atribut vrednost 1, sicer 0.
- promoted: če zaposlitev pomeni napredovanje, ima ta atribut vrednost 1, sicer 0. Ta atribut omogoča enostavnejše poizvedovanje o zaposlitvah, ki pomenijo napredovanje.
- daysNeeded: če je zaposlitev napredovanje, je tukaj zapisano število potrebnih dni za napredovanje. Ta atribut omogoča enostavnejše poizvedovanje po povprečnih časih potrebnih za napredovanje.
- fkWorkNext: tuji ključ tabele Work, ki kaže na naslednjo zaposlitev v karirerni poti diplomanta. Zato je tabela Work povezana sama s seboj.

Delovno mesto ima en naziv, isti naziv pa lahko ima več delovnih mest. Zato je povezava med tabelo Work in Position M:1 (mnogo proti ena). Za analizo podatkov nam pridejo prav tudi zaposlitve, ki nimajo naziva zaposlitve. Delovno mesto ima eno podjetje, medtem ko je v podjetju lahko več delovnih mest, zato je povezava med tabelo Work in Company M:1 (mnogo

proti ena). Za analizo podatkov nam pridejo prav tudi zaposlitve, ki nimajo podanega podjetja. Program bo poskrbel, da v podatkovni bazi nebo zaposlitev, ki so hkrati brez naziva in brez podjetja. Torej, delovno mesto je lahko brez naziva, če ima podano podjetje in obratno. Zaradi tega smo morali določiti poseben numerični atribut *idWork*, ki bo predstavljal primarni ključ delovnega mesta.

V podpoglavju 3.6.1 smo opisali hierarhično ureditev. Na podlagi hierarhične ureditve bo program vsakemu nazivu zaposlitve priredil skupino zaposlitve (tabela *Workgroup*). Vsak naziv lahko pripada natanko eni skupini zaposlitev, medtem ko skupina zaposlitev lahko pokriva več nazivov. Zato je povezava med tabelo *Workgroup* in *Position* 1:M (ena proti mnogo). Vsaka skupina je v natanko enem hierarhičnem nivoju, vendar nove tabele ne bomo ustvarjali, saj so hierarhični nivoji samo štirje.



Slika 3.2: Entitetno-relacijski diagram. Logični model podatkovne baze.

3.3 Luščenje podatkov iz ePrints.FRI

Program mora prebrati podatke strukturirane v formatu XML in jih s pomočjo knjižnice JAXB deserializirati. Knjižnice JAXB je podrobneje opisana v podpoglavju 2.3.2. Za izvedbno deserializacijo je potrebujemo:

- Podatkovni vir: V našem primeru je to naslov URL do katerega pridemo s klikom funkcije izvoz na spletni strani ePrints.FRI. Program bo prebral vsebino eprintov zapisano v formatu XML.
- Množico anotiranih Java razredov.

Izvedba deserializacije torej zahteva definicijo množice anotiranih razredov. Če preučimo dano strukturo XML, vidimo, da je korenski element *eprints*. Znotraj tega elementa je eden ali več elementov *eprint*, ki predstavlja eprint definiran v podpoglavju 2.1.1. Znotraj elementa eprint so vsi potrebni podatki o diplomski nalogi. Na voljo so podatki o enoličnem identifikatorju eprinta (element *eprint_id*), imenu in priimku kreatorja (element *creators*), imenu in priimku mentorja (element *mentors*), datum ustnega zagovora (element *oral_time*), datum oddaje (element *submitted*) in ključne besede diplomske naloge (element *keywords*). Za vsak omenjeni element in njihove podelemente bomo ustvarili razred in ga smiselno anotirali. Tako bo program z izvedbo deserializacije ustvaril množico med seboj povezanih Java objektov, ki bodo vsebovali vse potrebne podatke.

3.4 Iskanje

Preden program prične z iskanjem, je potrebno izvesti prijavo v sistem, saj LinkedIn ne omogoča uporabe funkcij iskanja in pregledovanja uporabnikom, ki niso prijavljeni. Pogoj za uspešno prijavo je ustvarjen uporabniški račun.

Prijavo bomo izvedli s pomočjo knjižnice *HtmlUnit*. HtmlUnit omogoča izpolnjevanje form programskega jezika HTML in proženje akcij s simuliranjem klikov gumbov. LinkedIn za prijavo uporablja formo z dvema poljema.

Prvo polje je namenjeno vnosu naslova e-mail, drugo pa vnosu gesla. HtmlUnit omogoča identifikacijo polj in vnos vrednosti v polja. Po vnosu vrednosti izvedemo metodo za prožitev akcije, ki simulira klik na gumb *Sign In*. Gumb *Sign In* proži akcijo za prijavo.

Ob uspešni prijavi strežnik vrne kodo HTML domače strani uporabnika. Dobra lastnost HtmlUnit knjižnice je, da vzdržuje sejo med uporabnikom in strežnikom, zato lahko v nadaljevanju prosto navigiramo po spletnih straneh brez potrebe ponovne prijave. Prejeta spletna stran vsebuje iskalnik omrežja LinkedIn, ki je tudi realiziran s HTML formo. Po prejetju HTML vsebine je program pripravljen za proženje akcije iskanja. Program se bo v zanki sprehodil skozi vse objekte, ki predstavljajo eprint. Za vsak objekt eprint bo prebral ime in priimek diplomanta ter s pomočjo knjižnice HtmlUnit vnesel podatke v formo. Zatem bo simulirali klik na gumb, ki bo sprožil akcijo iskanja.

Strežnik vrne brskalniku kodo HTML, ki v elementu *code* vsebuje HTML komentar. To nebi bilo nič posebnega, če le komentar nebi vseboval rezultatov iskanja v strukturirani obliki **JSON**. Primer vsebine v komentarju:

```
<code><!--
{
"url": "url",
"connections": {"_total": 500},
"firstName": "Igor"
}
--></code>
```

Program mora izluščiti vsebino komentarja in strukturo JSON deserializirati. Deserializacijo strukture JSON izvedemo s knjižnico *Gson*. Vsak rezultat predstavlja uporabnika omrežja LinkedIn in vsebuje naslov URL do profila. Po deserializaciji bo program imel na voljo podatke o imenu in priimku uporabnika ter naslovu URL do uporabnikovega profila. Zatem se mora program sprehoditi skozi vse rezultate in vsak rezultat *validirati*.

3.5 Validacija rezultatov

Na omrežju LinkedIn je več kot 200 miljonov uporabnikov, prav tako se število iz dneva v dan povečuje. Zaradi visokega števila uporabnikov obstaja možnost, da imajo uporabniki podobno ali celo enako ime in priimek. Nekateri uporabniki pri vpisovanju osebnih podatkov delajo napake, pišejo imena in priimke brez šumnikov ali celo ustvarjajo lažne in nepopolne profile. Zaradi tega je potreben mehanizem, ki bo med danimi profili našel veljavnega.

Proces validacije za danega diplomanta identificira veljaven profil med dano množico profilov. S tem validacija določi pripadnost veljavnega profila danemu diplomantu. Profil je veljaven, če izpolnjuje vse pogoje validacije (zadetek). Proces validacije se ustavi, če:

- So vsi profili dane množice validirani.
- Najde veljaven profil. To pomeni, da ob zadetku ne bodo validirani profili, ki so v seznamu za veljavnim profilom.

Validacija je eden najpomembnejših procesov, saj ima neposreden vpliv na relevantnost rezultatov diplomskega dela. Vsak profil je veljaven ali neveljaven in ga lahko klasificiramo v eno izmed štirih skupin:

- True positive (TP): pravilno identificiran veljaven profil.
- False positive (FP): Napačno identificiran veljaven profil.
- True negative (TN): Pravilno identificiran neveljaven profil.
- False negative (FN): Napačno identificiran neveljaven profil.

Natančnost in priklic sta določena s formulami:

$$Natancnost = TP/(TP + FP) \tag{3.1}$$

$$Priklic = TP/(TP + FN) \tag{3.2}$$

Želja je, da bi proces validacije *natančnost* in *priklic* približal vrednosti 1. To je mogoče, če storimo čim manjši delež napak FP in FN.

Proces validacije bo *trinivojski*. Na prvem nivoju bo program validiral ime in priimek, na drugem izobraževalno ustanovo in na tretjem leto diplomiranja.

Prvi nivo: validacija imena in priimka

Program bo primerjal ime in priimek diplomanta z imenom in priimkom zapisanim na profilu uporabnika. Program bo zamenjal šumnike s sorodnimi črkami brez šumnikov (npr., č bo preslikan v c), prav tako bo ignoriral velikost črk ter izvedel primerjavo. Na primer, niz *Igor Vinojčić* je ekvivalenten nizu *igor vinojcic*. Na omrežju LinkedIn veliko uporabnikov izpušča uporabo šumnikov. Veliko je tudi takšnih, ki namesto črke *ć* pišejo črko *č*. Z zamenjavo šumnikov rešimo te težave in zmanjšamo napako FN. Po drugi strani povečamo delež napake FP, vendar to napako izničijo naslednji nivoji validacije.

Ob uspešni validaciji imena in priimka se proces nadaljuje na naslednjem nivoju, sicer se proces validacije ustavi in profil je označen kot neveljaven.

Drugi nivo: validacija izobraževalne ustanove

Na profilu LinkedIn lahko uporabnik zapiše, katere izobraževalne institucije je obiskoval. Ker nas zanimajo samo diplomanti Fakultete za računalništvo in informatiko, mora biti na profilu zapisano, da je uporabnik obiskoval Fakulteto za računalništvo in informatiko. Uporabnik lahko zapiše naziv izobraževalne ustanove v veliko inačicah. Primeri:

- Zapisi s kratico: FRI, UL FRI.
- Zapisi v angleščini: Faculty for computer and information science, computer science.
- Zapisi v slovenščini: Fakulteta za računalništvo in informatiko, Univerza v Ljubljani.

Omrežje LinkedIn je razširjeno po celem svetu, zato uporabniki veliko uporabljajo angleški jezik. Brez upoštevanja različnih kombinacij nizov bi naredili preveč napak FN. Program bo za vsako kombinacijo preveril ali je vsebovana v nazivu institucije. Če je kombinacija vsebovana v nazivu, je izobraževalna ustanova uspešno validirana. Kombinacije nizov so sledeče:

- *computer* in *science*,
- *information* in *science*,
- *information* in *technology*,
- *racunalnistv*,
- *informatik*,
- *informacij* in *tehnolog*,
- *engineer's degree, cs*,
- *computer* in *engineering*,
- *fri*.

Zaradi števila kombinacij in njihove posplošenosti se poveča delež napak FP. Če je validacija izobraževalne ustanove uspešna, sledi validacija letnice diplomiranja, sicer je profil označen kot neveljaven.

Tretji nivo: validacija letnice

Uporabnik lahko poleg izobraževalne ustanove zapiše datum pričetka in konca izobraževanja. Za nas je relevanten podatek konca izobraževanja. Letnica konca izobraževanja mora biti enaka letnici ustnega zagovora (element *oral_time* eprinta v formatu XML) ali letnici ustnega zagovora pomanjšani za 1. Po izvedbi testa je bilo mogoče opaziti, da je veliko diplomantov (prb. 10%) na profilu zapisalo letnice diplomiranja za eno leto manjše. Razlogi za to so različni; nekateri uporabniki niso šteli dodatnega leta, nekateri so zapisali

samo prvo letnico šolskega leta, nekateri so skrili leto ponavljanja oziroma pavziranja ipd. Na tem nivoju je dodana kontrola, ki preveri, če obstaja še kakšen diplomant (podatki deserializiranih eprintov) z istim imenom in letnico, ki bi bila enaka letnici zmanjšani za 1. Če je validacija uspešna, je profil označen kot veljaven, sicer kot neveljaven.

Z opisanimi nivoji uspešno zmanjšamo napako FN. Na drugem nivoju odpadejo profili, ki nimajo zapisanih nazivov izobraževalnih ustanov. Na tretjem nivoju odpadejo profili, ki nimajo zapisanih letnic konca diplomiranja. To so kandidati za napake FN, ki se jim žal nemoremo izogniti, sicer bi validacija bila *premehka*.

Na prvi pogled bi lahko rekli, da zmanjšanje napake FN ni imelo smisla, saj smo povečali napako FP, vendar to ni res. Po trinivojski validaciji je napaka FP tisti profil uporabnika, ki ima zelo podobno ime in priimek kot dani diplomant in je obiskoval fakulteto z zelo podobnim nazivom ter diplomiral istega leta ali leto prej. To je, ne glede na 200 miljonov uporabnikov, malo verjetno. Iz tega sledi, da je trinivojska validacija zelo uspešna, saj uspe bistveno zmanjšati delež napak FN in zelo malo povečati delež napak FP.

Pred validacijo je potreben še dostop do profila uporabnika, ker rezultati iskanja ne vsebujejo dovolj podatkov o uporabnikih, da bi jih brez vsebine profila lahko validirali. S pomočjo knjižnice HtmlUnit dostopamo do profila uporabnika. Strežnik vrne profil uporabnika v formatu HTML. Program potem prejeto vsebino posreduje knjižnici *jsoup*, ki nam omogoča luščenje podatkov.

3.6 Luščenje podatkov iz omrežja LinkedIn

Dostop do profila je narejen že v fazi validacije. Prav tako je že inicializirana knjižnica *jsoup*, saj smo podatke iz profila morali luščiti za potrebe procesa validacije.

V tej fazi je potrebno izluščiti podatke o karierni poti. Na profilu se ti podatki nahajajo v razdelku *experiences*, kar v prevodu pomeni *izkušnje*.

Uporabnik lahko za vsako zaposlitev, pretekle in sedanje, zapiše kakšen je bil naziv zaposlitve, v katerem podjetju je bil zaposlen in datum začetka ter konca zaposlitve.

S pomočjo knjižnice *jsoup* preberemo vse omenjene podatke. V vsebini HTML profila se razdelek o izkušnjah nahaja v elementu z identifikatorjem *profile-experience*. Za vsako zaposlitev zgradimo Java objekt razreda *LinkedInExperienceModel* in ga vnesemo v Java seznam (*ArrayList*). To nam namreč omogoča manipulacijo podatkov o zaposlitvah pred vnosom v podatkovno bazo.

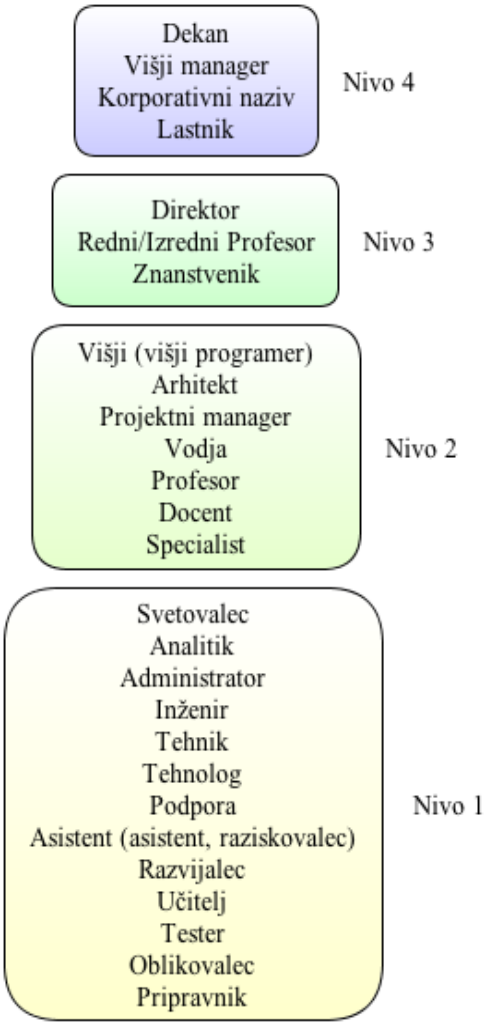
3.6.1 Hierarhična ureditev in napredovanje

V fazi analize bomo preučevali potreben čas za napredovanje diplomantov FRI. Računalnik je deterministično sistem, zato moramo *napredovanje* natančno definirati, kar se izkaže za problematičen proces. Razlogi so sledeči:

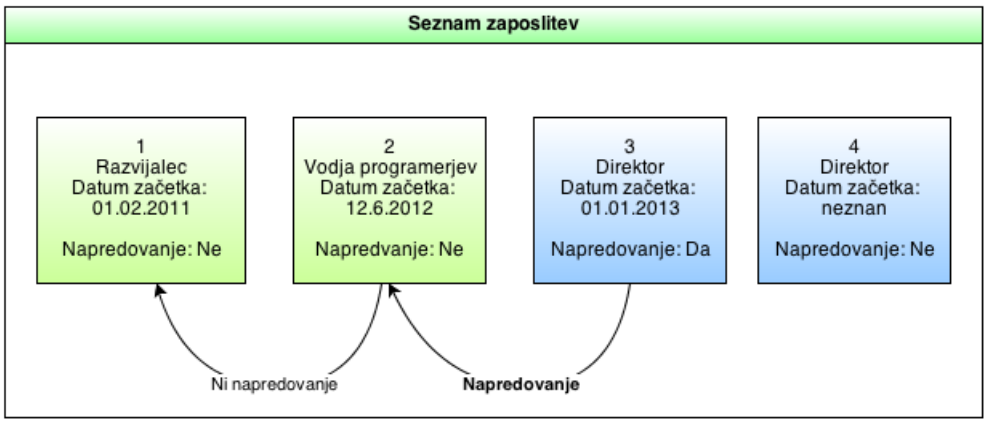
- Trg dela ponuja zelo širok spekter poklicev. Diplomant, ki je dokončal študij iz področja računalništva, nebo nujno zaposlen na področju računalništva.
- Veliko podjetij ima svojo lastno hierarhično ureditev.
- Pomen nazivov se iz države do države razlikujejo.
- Uporabniki omrežja LinkedIn različno imenujejo zelo podobna, če ne celo enaka, delovna mesta. Recimo, nekateri uporabniki so zaposleni kot *razvijalci programov*, drugi kot *programerji*.
- Zaradi različnih izkušenj in percepcije ljudje različno dojemamo napredovanje.
- V velikih korporacijah so večinoma lastniki nadrejeni direktorjem. Kaj pa lastniki manjših podjetij? So ti na hierarhični lestvici višje od direktorjev korporacij? To je težko določljivo.

Zaradi omenjenih razlogov je definicija splošne hierarhične ureditve zelo težavna in kompleksna ter skoraj nemogoča.

Za potrebe rešitve našega problema smo sestavili hierarhično ureditev, ki je večinoma upoštevala zaposlitve na področju računalništva[24]. Vsak nivo hierarhije je sestavljen iz ene ali večih skupin. *Napredovanje* je prehod iz nižjega nivoja hierarhije v višjega. Slika 3.3 prikazuje sestavljeno hierarhijo.



Slika 3.3: Prikaz hierarhije. Spodaj je najnižji nivo, zgoraj najvišji. V oklepaju so primeri nazivov zaposlitev, ki spadajo pod dano skupino.



Slika 3.4: Ugotavljanje napredovanja nad urejenim seznamom.

Naloga programa v tej fazi je, da vsako zaposlitev uvrsti v eno izmed hierahičnih skupin in ugotovi, če je dana zaposlitev *napredovanje*. Omenili smo, da so vse zaposlitve danega diplomanta shranjene v seznamu. Program bo seznam uredil naraščajoče po datumu začetka zaposlitve. Če neka zaposlitev nima pripadajočega začetka zaposlitve, uvrsti zaposlitev na konec seznama. Potem se program sprehodi skozi seznam od konca proti začetku in primerja sosednje zaposlitve. Če neka zaposlitev ima levega soseda v seznamu in njen datum začetka ni prazno polje, potem program primerja hierarhični nivo zaposlitev. Če je hierarhični nivo zaposlitve na višjem nivoju kot nivo zaposlitve levega soseda, je to napredovanje. Vsaki zaposlitvi program priredi spremenljivko, ki pove ali gre za napredovanje. Slika 3.4 prikazuje primer ugotavljanja napredovanja.

V fazi analize nas bo zanimal potreben čas za napredovanje. Računanje potrebnega časa za napredovanje je lažje izvedljivo v programskem jeziku Java kot v programskem jeziku SQL, zato bo za računanje poskrbel program. Algoritem je zelo podoben tistemu, ki ugotavlja napredovanje. Program izračuna število pretečenih dnevov med zaposlitvijo, ki je napredovanje, in prvo zaposlitvijo, ki je na nivoju za 1 nižje. Na primer, program bi izračunal število pretečenih dni med zaposlitvijo 1 (1. 2. 2011) in 3 (1. 1. 2013) na sliki 3.4

3.7 Vnos podatkov v podatkovno bazo

V podpoglavju 2.5.2 smo omenili, da bomo za načrtovanje in ustvarjanje podatkovne baze MySQL uporabili orodje MySQL Workbench, medtem ko bo polnjenje baze izvedel program.

Povezavo med Java programom in MySQL strežnikom omogoča knjižnica *mysql-connector-java*. Knjižnica omogoča povezovanje s podatkovno bazo in izvajanje ukazov SQL.

V tej fazi ima program na voljo vse potrebne podatke. Potrebno je biti pozoren na vrstni red vnašanja podatkov. Zaradi omejitev (ang. *constraints*) podatkovna baza ne dovoljuje vnosa tujega ključa, ki ne obstaja v referenčni tabeli kot primarni ključ. Na primer podatkov o diplomskem delu ne moremo vnesti v tabelo *Degree*, če v tabeli *Mentor* ne obstaja mentor s primarnim ključem, ki je enak tujemu ključu željenega vnosa. To pomeni, da moramo pred vnosom v tabelo *Degree* vnesti podatke mentorja v tabelo *Mentor*.

3.8 Težave

Omrežje LinkedIn se z omejitvami dostopov[26] skuša odbraniti napadov *zavrnitve storitve* in prekomernemu luščenju podatkov. V bazi eprints je več kot 1300 eprintov, kar zahteva proženje vsaj 1300ih iskanj in vsaj 400ih dostopov do profila (v podatkovni bazi je namreč več kot 400 diplomantov) v manj kot pol ure. LinkedIn tega ne dopušča in ob prekomerni uporabi njihovih storitev blokira uporabnika, ki storitev uporablja. Uporabnik je blokiran vsaj 12 ur, natančne številke ne poznamo. Po določenemu številu testiranj smo ugotovili, da lahko program pregleda približno 600 diplomantov. To pa pomeni, da za pregled celotne baze eprints potrebujemo vsaj 24ur, kar je zelo veliko in povzroča zelo zamudno testiranje programa.

Drugo težavo predstavlja nekonsistentna baza knjižnice ePrints.FRI. Namreč, podatki o starejših diplomantih so nepopolni, pri nekaterih manjkajo podatki o času ustnega zagovora, ključnih besedah, mentorjih ipd. Prav tako je v bazi nekaj napak. Na primer, nekateri diplomanti imajo v elementu

ključnih besed zapisane celotne povzetke diplom.

Poglavje 4

Analiza podatkov

4.1 Poizvedbe

4.1.1 Veljavna in prva zaposlitev

Na profilu omrežja LinkedIn ima veliko diplomantov zapisane zaposlitve, ki so jih opravljali v času študija. Za nas so zaposlitve pred datumom ustnega zagovora nepomembne in jih v analizi ne želimo upoštevat. Težava se pojavi, če je diplomant po diplomiranju obdržal zaposlitev še iz časa študija. To težavo rešimo tako, da v poizvedbah upoštevamo samo tiste zaposlitve, pri katerih je *toDate* ali *currDate* (glej podpoglavje 3.2.2) večji od datuma ustnega zagovora (oralTime). Tako dobimo vse zaposlitve, ki jih je diplomant opravljal po diplomi. Problem je predstavljala tudi ugotovitev, da je kar nekaj diplomantov po zaključku diplome hitro zamenjalo zaposlitev. Zaradi tega smo dodali še pogoj, da je zaposlitev veljavna, če jo je diplomant po datumu ustnega zagovora opravljal še vsaj 30 dni. Torej, *veljavne* so tiste zaposlitve, ki imajo datum toDate ali currDate večji (kasnejši) od datuma ustnega zagovora (oral time) in so po datumu ustnega zagovora bile opravljane še vsaj 30 dni.

Kar nekaj diplomantov je imelo v nekem trenutku dve zaposlitvi. S tem so nastale težave pri določitvi, katera je *prva zaposlitev*. Težavo smo

rešili tako, da smo vse veljavne zaposlitve prvo sortirali po datumu *fromDate* naraščajoče, potem pa po času opravljanje zaposlitve po datumu ustnega zagovora padajoče. Tako smo na prvem mestu dobili veljavno zaposlitev, ki jo je diplomant časovno prvo opravljal. Če obstajata zaposlitvi, ki imata isti čas začetka se kot prva vzame tista, ki jo je diplomant po datumu ustnega zagovora dlje opravljal.

4.1.2 Karierna pot in pivotiranje

Za določitev najpogostejših kariernih poti je potrebno implementirati SQL poizvedbo, ki bo grupirala vse karierne poti diplomantov in za vsako grupo prikazala število diplomantov, ki tej karierni poti (grupi) pipada.

V podatkovni bazi imamo za vsakega diplomanta na voljo vse zaposlitve. Slika 4.1 prikazuje primer izpisa zaposlitev diplomanta FRI *Igorja Vinojcica*.

Igor	Vinojcic	Web programmer
Igor	Vinojcic	Computer engineer
Igor	Vinojcic	CTO

Slika 4.1: Zaposlitve diplomanta Igorja Vinojcica

Če predpostavimo, da so zaposlitve na sliki 4.1 urejene po datumu začetka (atribut fromDate tabele Work) naraščajoče, potem je karierna pot Igorja Vinojcica sestavljena iz treh zaposlitev. Začel je kot *web programmer*, potem se je zaposlil kot *computer engineer* in karierno pot zaključil kot *CTO*. Težave se pojavijo, ko želimo grupirati po kariernih poteh. Namreč, tabela na sliki 4.1 ne omogoča grupiranja po kariernih poteh, saj je karierna pot sestavljena iz večih vrstic in ne večih stolpcev. Če želimo grupirati po kariernih poteh, moramo nazive zaposlitev premakniti v stolpce. Premik vrstic v stolpce imenujemo *pivotiranje*. V MySQL je to omogočeno z ukazom *group_concat*. Slika 4.2 prikazuje pivotiranje tabele.

Igor	vinojcic	group_concat(title)
Igor	vinojcic	Web programmer,Computer engineer,CTO

Slika 4.2: Rezultat pivotiranja tabele iz slike 4.1

4.2 Vzorec

V trenutku zagona programa je bilo na digitalni knjižnici 1399 zapisov o diplomskih delih (eprint). Program je za vsak eprint skušal najti veljaven profil na omrežju LinkedIn.

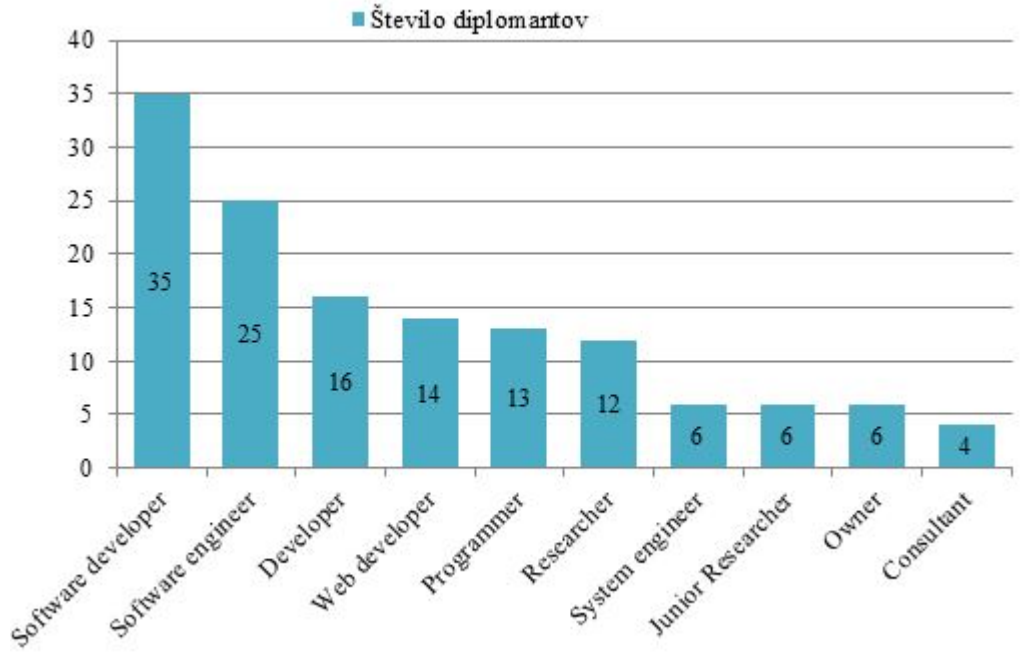
S procesom validacije, opisanim v podpoglavju 3.5, je program našel 405 veljavnih profilov, kar predstavlja *28,95%* delež. V osnovi ima še več diplomantov ustvarjene profile na omrežju LinkedIn, vendar profili niso vsebovali veljavnih oziroma potrebnih podatkov. Nekateri niso imeli zapisanih datumov o zaključkih izobraževanja, drugi niso imeli navedenih zaposlitev ipd.

Za našo analizo so veljavne samo zaposlitve od *prve zaposlitve* dalje. Zaradi tega pogoja se vzorec zmanjša na 386 diplomantov, saj jih 19 nima prve zaposlitve. V večini gre tukaj za diplomante, ki so diplomirali najkasneje in so bili v času študija zaposleni preko študentskih servisov. Torej, dejanski vzorec je velik ***386*** diplomantov, kar predstavlja ***27,59%*** delež.

4.3 Rezultati in interpretacija

4.3.1 Prve zaposlitve diplomantov FRI

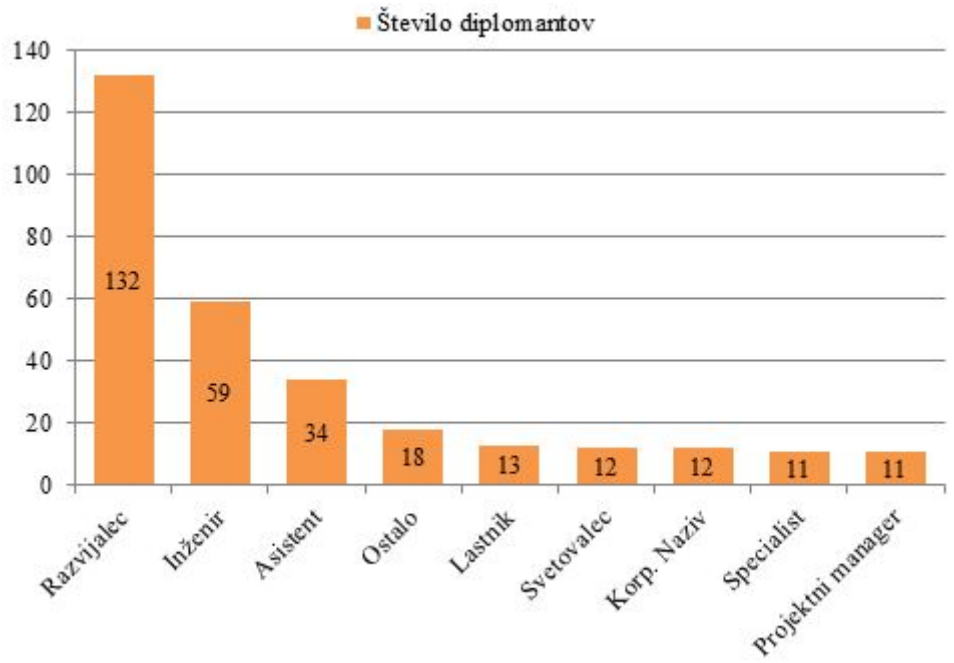
Vzorec predstavlja 386 diplomantov. To so vsi diplomanti shranjeni v podatkovni bazi, ki imajo prvo zaposlitev. Slika 4.3 prikazuje graf desetih najpogostejših nazivov prvih zaposlitev. Vidimo lahko, da je najpogostejša prva zaposlitev *software developer (razvijalec programske opreme). Software developer* predstavlja *9,07%* delež, kar je na prvi pogled presenetljivo malo. Težava je v tem, da so diplomanti na svojih profilih omrežja LinkedIn različno zapisali nazive zelo podobnih poklicev. Na prvih treh mestih lahko vidimo, da



Slika 4.3: Porazdelitev števila diplomantov po nazivih prvih zaposlitev.

se nahajo nazivi software developer (razvijalec programske opreme), software engineer (inženir programske opreme) in developer (razvijalec). V osnovi so ti trije poklici zelo podobni, zato bi za natančnejše rezultate bilo potrebno deleže sešteti. V podpoglavju 3.6.1 smo za rešitev omenjenih težav uporabili *hierarhično ureditev*, s pomočjo katere smo združili podobne zaposlitve v *skupine zaposlitev*.

Slika 4.4 prikazuje graf devetih najpogostejših skupin prvih zaposlitev. Vidimo lahko, da je najpogostejša skupina *razvijalec*. 135 diplomantov se je po diplomi zaposlilo na službenem mestu, ki spada v skupino zaposlitev *razvijalec*. Skupina razvijalec predstavlja *34,91%* delež. Na drugem mestu se nahaja skupina *inženir*, ki predstavlja *15,28%* delež. Če oba deleža seštejemo dobimo *50,19%*. To pomeni, da se je približno 50% diplomantov zaposlilo na službenem mestu, ki spada bodisi v skupino razvijalec bodisi v skupino inženir, kar je pričakovano.



Slika 4.4: Porazdelitev števila diplomantov po skupinah zaposlitev.

Presenetljivo sta se med prvih devet uvrstili skupini *lastnik* in *korporativni naziv*. Ti dve skupini spadata v vrh hierarhične lestvice, definirane v podpoglavju 3.6.1. Skupaj predstavljata *6,47%* (seštevek deleža obeh skupin). Razlog za tako visok delež se skriva predvsem v podjetništvu. Veliko študentov se odloči za samostojno pot, predvsem je občutiti porast *zagonskih* podjetij, ki so posledica razvoja pametnih telefonov. Drugi razlog je ta, da so si nekateri diplomanti nadeli lastniške in korporativne nazive, ker so upravljali z lastnimi spletnimi stranmi.

Rezultati kažejo na očitno pomanjkanje kadra na področju informacijske varnosti. Na omenjenem področju sta se zaposlila le dva diplomanta, in sicer z nazivom *Cisco Certified Academy Instructor for CCNA and CCNA-Security* in *Authentication and authorization infrastructure*. To predstavlja le *0,5%* delež. Veliko delovnih mest na področju računalništva in informatike že v osnovi zahteva znanje in zavedanje informacijske varnosti, vendar smo pričakovali večji delež. Ni potrebno posebej poudarjati, da je danes na spletu

	Povprečje (%)	Standardna deviacija (%)
1 nivo	79,40	15,52
2 nivo	16,18	6,18
3 nivo	10,51	4,60
4 nivo	14,86	9,37

Tabela 4.1: Povprečja in standardne deviacije po hierarhičnih nivojih.

ogromna količina podatkov, ki mora biti primerno zaščitena, to pa zahteva primeren in specializiran kader. Na tem mestu se je smiselno vprašati ali Fakulteta za računalništvo in informatiko Univerze v Ljubljani daje dovolj pozornosti informacijski varnosti.

Prve zaposlitve po mentorjih

V podatkovni bazi je 54 različnih mentorjev, 150 različnih nazivov prvih zaposlitev in 28 različnih skupin prvih zaposlitev. Če upoštevamo, da so samo trije mentorji z več kot dvajsetimi diplomanti, vidimo, da bo analiza zelo otežena. Namreč, večje število kombinacij pomeni manjše število diplomantov na kombinacijo. Zaradi tega bomo analizo izvedli nad nivoji prvih zaposlitev. Nivo predstavlja nivo hierarhične lestvice opisane v podpoglavju 3.6.1. Nivoji so samo štirje, zato izrazito zmanjšamo število kombinacij in tako povečamo število diplomantov na kombinacijo.

Za vsakega mentorja smo izračunali deleže prvih zaposlitev po nivojih in nad dobljenimi deleži za vsak nivo izračunali povprečje in standardno deviacijo.

Po izračunu povprečja smo se vrnili k deležem prvih zaposlitev po nivojih in za vsakega mentorja izračunali odstopanje od povprečja. Če je bila absolutna vrednost odstopanja večja od standardne deviacije, smo mentorja posebej označili. Slika 4.5 prikazuje primer izpisa poizvedbe.

Na sliki 4.6 lahko vidimo, da nobeden od označenih mentorjev izrazito ne izstopa. Zanimivo je, da se je vseh 12 diplomantov mentorja *Mentor 6* prvič

Mentor	Mentor 9	1	15	88.2353	17	15.525663363242412	8.82991824	No
Mentor	Mentor 9	2	1	5.8824	17	6.187143849945413	10.29883801	Yes

Slika 4.5: Primer deleža prvih zaposlitev po nivojih in mentorjih

zaposlilo na nivoju 1. Podobno velja za mentorja *Mentor* 1 in *Mentor 2*, kjer se je 11 diplomantov zaposlilo na 1. nivoju in eden na 2. nivoju. Glede na rezultate ne moremo trditi, da so prve zaposlitve odvisne mentorja.

Prve zaposlitve po letih diplomiranja

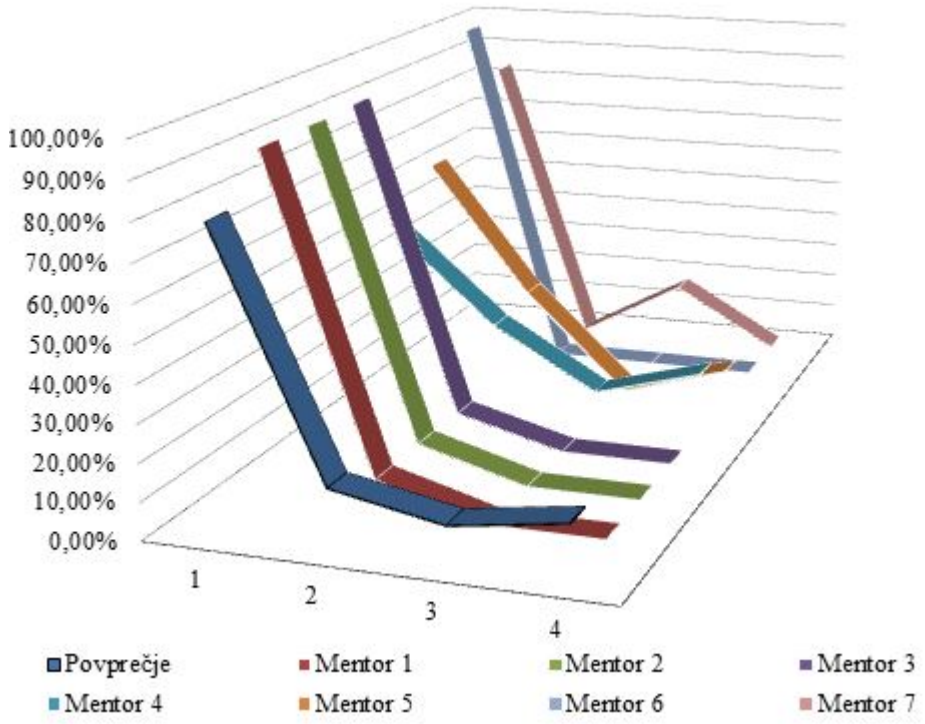
Od leta 2006 se prve zaposlitve niso bistveno spreminjale. Prednjačita skupini *razvijalec* in *inženir*. Zanimivo je gibanje deleža zaposlitev v skupini *asistent* in zaposlitev na področju *mobilne tehnologije*.

Glavni akterji na področju mobilne tehnologije so Microsoft, Apple in Google s svojimi operacijskimi sistemi. Leta 2007 je na trg prišel operacijski sistem iOS podjetja Apple, leta 2008 operacijski sistem Android podjetja Google in leta 2010 operacijski sistem Windows Phone podjetja Microsoft. Kot lahko vidimo na sliki 4.7, se je delež prvih zaposlitev na področju mobilne tehnologije začel bistveno večati leta 2011. Veliko delovnih mest razvijalcev zajema razvijanje mobilnih aplikacij in jih diplomanti kot take niso izpostavljali na svojih profilih, zato je dejanski delež verjetno drugačen. Je pa zanimivo, kdaj se je delež začel večati. V kolikor bi imeli večji vzorec, bi s tem podatkom lahko ugotavljali prilagodljivost fakultete na zahteve trga.

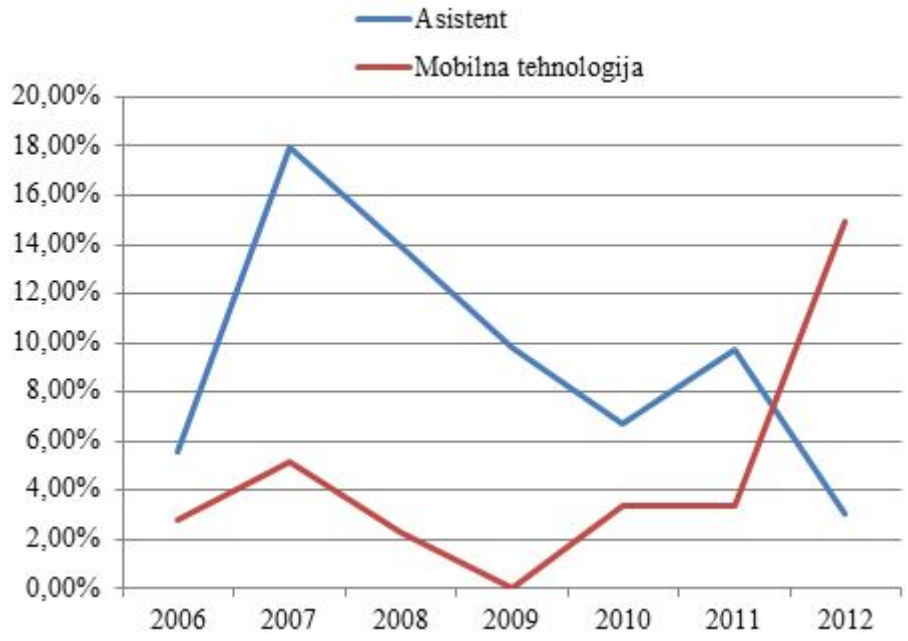
Na sliki 4.7 je z modro črto prikazan delež skupine *assistent*. Ta skupina zajema predvsem diplomante, ki se odločijo nadaljevati svojo pot kot asistenti in raziskovalci na fakultetah ali inštitutih. Zanimivo je, kako je delež leta 2007 strmo narasel in do leta 2012 strmo padel.

Podjetja prvih zaposlitev diplomantov FRI

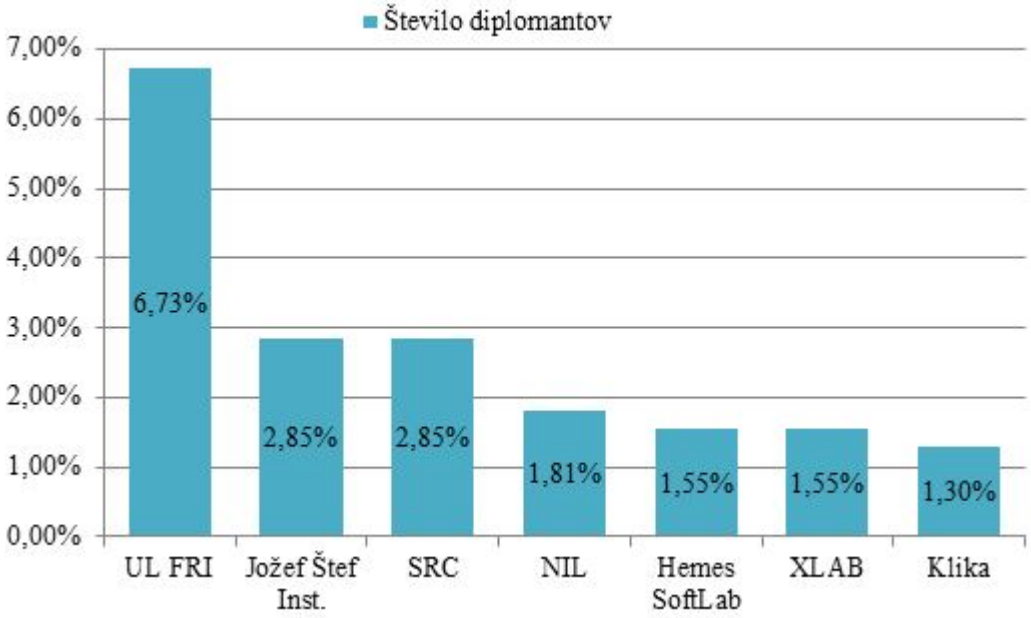
Stolpični graf na sliki 4.8 prikazuje 7 najpogostejših podjetij, kjer diplomanti dobijo prvo zaposlitev. Vidimo lahko, da so na prvih treh mestih



Slika 4.6: Porazdelitev prvih zaposlitev po mentorjih in nivojih.



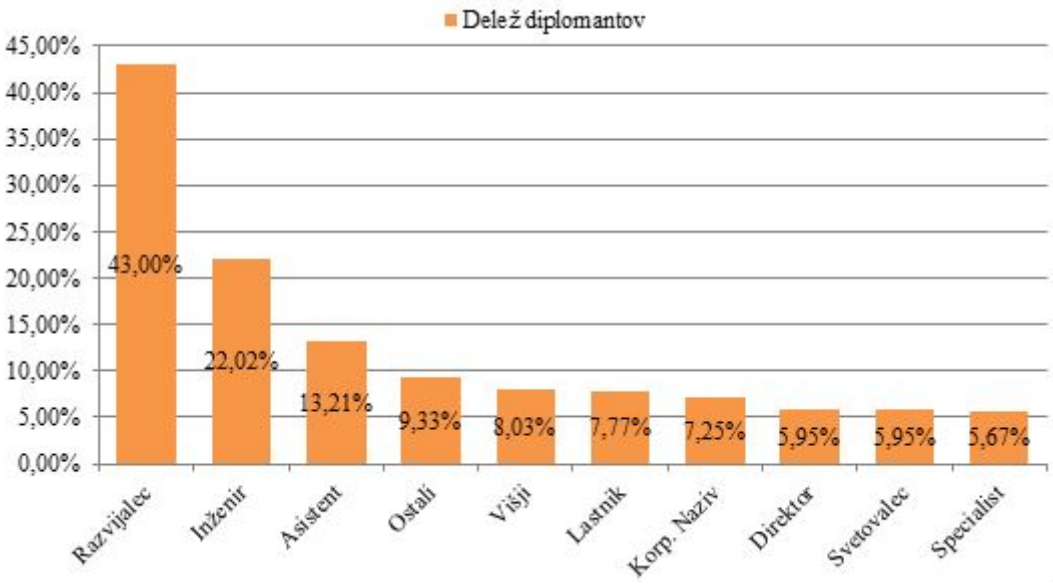
Slika 4.7: Delež prvih zaposlitev po letu diplomiranja. Modra črta predstavlja deleže prvih zaposlitev v skupini asistent, rdeča na področju mobilne tehnologije.



Slika 4.8: Delež diplomantov po podjetjih prvih zaposlitev.

Jožef Štefan Inštitut, *Fakulteta za računalništvo in informatiko Univerze v ljubljani* in *SRC*. Jožef Štefan Inštitut in Fakulteta za računalništvo in informatiko skupaj sodelujeta ter predstavljata 9,58% delež. To pomeni, da se je skoraj 10% diplomantov odločilo, da bo karierno pot začelo kot raziskovalec, asistent ipd. SRC je pričakovano uvrščeno visoko, saj je dolga leta študentom Fakultete računalništva in informatike ponujalo štipendije. Na 4. mestu je podjetje NIL z 1,81% deležem. Podjetja SRC in NIL sta velikokrat sodelovala s Fakulteto za računalništvo in informatiko. Iz tega sledi, da je večja verjetnost, da se bo diplomant prvič zaposlil v podjetju, ki sodeluje s fakulteto, kot v podjetju, ki ne sodeluje s fakulteto.

Podatki grupirani po mentorju ali letnici diplomiranja so zelo razpršeni. Največje število diplomantov kombinacije mentorja in podjetja je 2, kombinacije leta diplomiranja in podjetja pa 6, kar je prenizko za podrobnejšo analizo.



Slika 4.9: Delež diplomantov vsaj enkrat zaposlenih v dani skupini zaposlitev.

4.3.2 Vse zaposlitve diplomantov FRI

V tem delu analize bomo upoštevali vse zaposlitve diplomantov FRI od vključno njihove prve zaposlitve. *Število diplomantov* predstavlja vse diplomante, ki so v svoji karieri (od vključno prve zaposlitve) bili vsaj enkrat zaposleni na danem delovnem mestu. *Delež* predstavlja razmerje med številom diplomantov in vsemi diplomanti, ki imajo svojo prvo zaposlitev (teh je 386).

Na sliki 4.9 je prikazanih 10 najpogostejših skupin zaposlitev. Pričakovano izstopa skupina *razvijalec*, saj je bilo 43% diplomantov od 386 je v svoji karierni poti zaposlenih kot razvijalec. Rezultati se skladajo z rezultati iz podpoglavja 4.3.1. Zanimivo je, da je skupen delež skupin *lastnik*, *korporativni naziv* in *direktor* enak 20,97%. To pomeni, da je približno vsak peti diplomant svoji karieri bil vsaj enkrat v vrhnjem delu hierarhične lestvice, kar je nepričakovano visok delež.

Vse zaposlitve po mentorjih

Veliko zaposlenih si v svoji karierni poti želi biti na vrhu hierarhične lestvice, zato smo izračunali deleže vseh zaposlitev na tretjem in četrtem nivoju po mentorju. Upoštevali smo samo mentorje, ki imajo več kot 3 diplomante FRI kadarkoli zaposlene v tretjem ali četrtem nivoju. Tako smo dobili 6 mentorjev, vendar izstopa samo mentor eden z deležem ***58,33%***. Namreč, izmed dvanajstih diplomantov FRI jih je kar 7 v svoji karierni poti bilo zaposlenih na nivoju 3 ali 4.

4.3.3 Povprečen čas potreben za napredovanje diplomantov FRI

Pri izračunu povprečnega časa smo upoštevali vsa napredovanja diplomantov FRI od vključno prve zaposlitve. Tako smo ignorirali morebitna napredovanja pred diplomiranjem. Napredovanje je definirano v podpoglavju 3.6.1 in predstavlja prehod iz nižjega nivoja hierarhične lestvice v višjega.

V povprečju so diplomanti FRI potrebovali *1756.73 dni* za napredovanje. To je *4,81 neprestopnih let*. Če diplomant začne karierno pot na najnižjem nivoju hierarhične ureditve in želi priti do najvišjega nivoja mora napredovati 3-krat, kar je 5270,19 dni oziroma 14,44 neprestopnih let.

Povprečen čas potreben za napredovanje diplomantov FRI po mentorju

Za vsakega mentorja smo izračunali povprečne čase potrebne za napredovanje. Nad dobljenimi časi smo izračunali povprečje in standardno deviacijo. Povprečje je v tem primeru 1807,82 dni, standardna deviacija 946.06 dni. Če je bila absolutna razlika potrebnega časa za napredovanje po mentorju in povprečnega časa večja od standardne deviacije, smo mentorja izpisali. Tako smo dobili samo tiste mentorje, ki izstopajo iz povprečja glede na potreben čas za napredovanje. Izločili smo tudi mentorje, ki imajo manj kot 10 diplomantov in tiste mentorje, pri katerih so diplomanti skupno napredovali

manj kot dvakrat. Po tej metodi sta izstopala dva mentorja. Diplomanti prvega mentorja (12 diplomantov in 3 napredovanja) so v povprečju potrebovali za napredovanje 4474,67 dni, kar predstavlja *12,25 neprestopnih let*, kar je ogromno. Diplomanti drugega mentorja (19 diplomantov in 6 napredovanj) v povprečju potrebujejo 2927,50, kar je *8,02 neprestopnih let*. Seveda je vzorec v obeh primerih premajhen, da bi lahko karkoli sklepali.

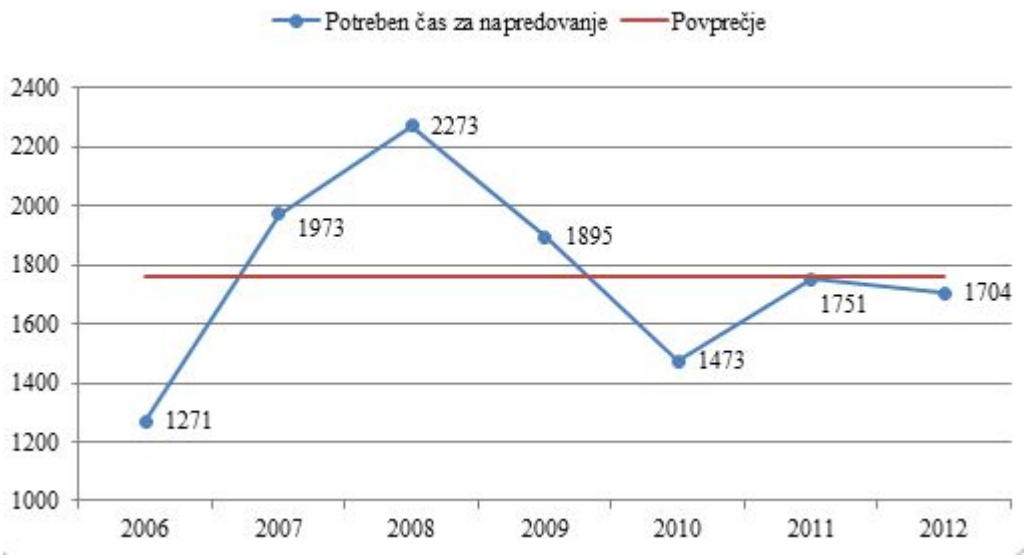
Povprečen čas potreben za napredovanje diplomantov FRI po letu diplomiranja

Diplomanti, ki so diplomirali leta 2008 v povprečju potrebujejo največ časa za napredovanje, in sicer 2273 dni, kar je 6,23 neprestopnih let. To lahko vidimo na sliki 4.10. V povprečju potrebujejo diplomanti, ki so diplomirali leta 2006, najmanj dni za napredovanje, in sicer 1271, kar je 3,48 neprestopnih let. Za podrobnejšo analizo bi bilo potebno imeti večji vzorec z daljšim časovnim razponom. Težava je v tem, da so diplomanti, ki so diplomirali leta 2006 verjetno zaposleni dlje (večje število napredovanj), kot diplomanti, ki so diplomirali leta 2011. Zato je podatek za leto 2006 bolj natančen, kot za leto 2011.

Karierne poti

Za ugotavljanje najpogostejših poti, smo uporabili vse poti diplomantov. Karierna pot diplomanta se začne s prvo zaposlitvijo. 5 najpogostejših zaposlitev je:

1. Developer (razvijalec)
2. Engineer (inženir)
3. Assistant (asistent)
4. Developer (razvijalec) - Engineer (inženir)
5. Specialist (specialist)



Slika 4.10: Rdeča črta predstavlja povprečje in modra povprečno število dni potrebnih za napredovanje po letu diplomiranja.

Z rezultatom smo kar malo razočarani, saj smo pričakovali bolj razvejane in daljše karierne poti. Razlog je ta, da smo karierne poti analizirali prekmalu. Namreč, v bazi imamo največ diplomantov, ki so diplomirali od leta 2006 naprej. Karierna pot se je za večino diplomantov šele dobro začela in traja večinoma manj kot 7 let. Če se vrnemo v podpoglavje 4.3.3 lahko vidimo, da je v povprečju za napredovanje potrebno čakati približno 4 leta.

Poglavje 5

Zaključek in nadgradnje

Cilj diplomske naloge je bil določiti vpetost diplomantov FRI v gospodarski prostor, kar nam je delno uspelo. Ugotovili smo, da svojo karierno pot polovica diplomantov začne kot razvijalec ali inženir. Presenetilo nas je dejstvo, da je zelo malo prvih zaposlitev na področju informacijske varnosti. Na podlagi rezultatov ne moremo trditi, da je prva zaposlitev odvisna od mentorja. Ugotovili smo lahko, da obstaja večja verjetnost, da se diplomant prvič zaposli v podjetju, ki s fakulteto sodeluje, kot v podjetju, ki s fakulteto na takšen ali drugačen način ne sodeluje. Zanimiva je ugotovitev, da je v svoji karierni poti vsak peti diplomant bil vsaj enkrat zaposlen kot direktor ali lastnik ali pa je imel korporativni naziv. Prav tako smo uspeli izračunati povprečen čas za napredovanje, ki je znašal 1756,73 dni, kar je 4,81 neprestopnih let. Zaradi majhnosti vzorca ne moremo trditi, da je povprečen čas napredovanja odvisen od mentorja.

Rezultati bi z določenimi nadgradnjami lahko bili natančnejši in relevantnejši. Omrežje LinkedIn ponuja veliko drugih podatkov, ki jih v analizi nismo upoštevali. Na profilih uporabnikov so zapisi o uporabnikovih veščinah, projektnih delih, priporočilih in povezavah. Program bi lahko nadgradili tako, da izčrpa vse omenjene podatke in nad podatki izvedli analizo. Zanimivo bi bilo videti, kakšne veščine imajo diplomanti FRI. Pri validaciji bi lahko uporabili podatek ali je uporabnik povezan z mentorjem ipd. Prav

tako je mogoča analiza na podlagi ključnih besed diplomskih del in analiza, ki ugotavlja trend skozi čas.

Analizo bi bilo dobro izvesti enkrat v prihodnosti z naprednejšimi metodami za rudarjenje podatkov. Namreč, več podatkov je na voljo o kariernih poteh diplomantov, ki so diplomirali prej, saj so ti večinoma dalj časa prisotni na trgu dela.

Karierne poti bi bilo smiselno analizirati iz vidika omrežja. Ideja je, da bi zaposlitev (naziv ali skupina) predstavljala vozlišče v omrežju. S posebnimi metodami za analizo omrežja bi potem lahko analizirali različne vidike. Morda tisti, ki začnejo karierno pot kot svetovalci pogosteje napredujejo na najvišja delovna mesta, razvijalci stagnirajo ipd.

Prav tako se odpirajo možnosti primerjave področij. Na primer, lahko bi primerjali karierne poti diplomantov FRI z diplomanti drugih fakultet ipd.

Literatura

[1] Spletna stran rabe interneta v Sloveniji, Spletna socialna omrežja, 18.september 2013.

Dostopno na:
http://www.ris.org/index.php?fl=2&lact=1&bid=9805&par

[2] Uradna spletna stran razvojnega orodja Toad.

Dostopno na:
https://www.toadworld.com/ (2013)

[3] Wikipedia (Eprint).

Dostopno na:
http://en.wikipedia.org/wiki/Eprint (2013)

[4] Spletna stran digitalne knjižnice ePrints.FRI.

Dostopno na:
http://eprints.fri.uni-lj.si/ (2013)

[5] Uradna spletna stran LinkedIn.

Dostopno na:
http://www.linkedin.com/ (2013)

[6] Pomoč uporabnikov omrežja LinkedIn (Visibility), 15.september 2013.

Dostopno na:
http://help.linkedin.com/app/answers/detail/a_id/77/~/information-viewable-on-your-profile

[7] Wikipedia (Serialization).

Dostopno na:
http://en.wikipedia.org/wiki/Serialization (2013)

[8] Uradna spletna stran knjižnice jsoup.

Dostopno na:
http://jsoup.org/ (2013)

[9] Uradna spletna stran knjižnice HtmlUnit.

Dostopno na:
http://htmlunit.sourceforge.net/ (2013)

[10] Uradna spletna stran sistema MySQL.

Dostopno na:
http://www.mysql.com/ (2013)

[11] Uradna spletna stran vmesnika MySQL Workbench.

Dostopno na:
http://www.mysql.com/products/workbench/ (2013)

[12] IETF zahteva za mnenja 6922, 15.september 2013.

Dostopno na:
http://tools.ietf.org/html/rfc6922

[13] IETF zahteva za mnenja 2713, 15. september 2013.

Dostopno na:
http://tools.ietf.org/html/rfc2713

[14] Wikipedia (Relation database).

Dostopno na:
http://en.wikipedia.org/wiki/Relational_database (2013)

[15] Wikipedia (Database normalization).

Dostopno na:
http://en.wikipedia.org/wiki/Database_normalization (2013)

[16] Pomoč uporabnikov omrežja LinkedIn (LinkedIn search).

Dostopno na:
http://help.linkedin.com/ (2013)

[17] Uradna spletna stran vmesnika LinkedIn API (Full profile fields).

Dostopno na:
http://developer.linkedin.com/documents/profile-fields#fullprofile
(2013)

[18] Uradna spletna stran knjižnice JAXB.

Dostopno na:
https://jaxb.java.net/ (2013)

[19] Uradna spletna stran knjižnice Gson.

Dostopno na:
https://code.google.com/p/google-gson/ (2013)

[20] Dokumentacija tehnike anotiranja v programskem jeziku Java.

Dostopno na:
http://docs.oracle.com/javase/1.5.0/docs/guide/language/annotations.html
(2013)

[21] Uradna spletna stran tehnične dokumentacije programskega jezika Java.

Dostopno na:
http://docs.oracle.com/javase/ (2013)

[22] Uradna spletna stran tehnologije Maven.

Dostopno na:
http://maven.apache.org/ (2013)

[23] Uradna spletna stran repozitorija Maven.

Dostopno na:
http://mvnrepository.com/ (2013)

[24] Spletna stran hierarhičnih struktur.

Dostopno na:
http://www.hierarchystructure.com/corporate-jobs-hierarchy/ (2013)

[25] Spletna stran java knjižnice mysql-java-connector.

Dostopno na:
http://dev.mysql.com/downloads/connector/j// (2013)

[26] Uradna spletna stran omrežja LinkedIn o omejitvah.

Dostopno na:
http://developer.linkedin.com/documents/throttlelimits (2013)

Slike