A Prototype for an e-Recruitment Platform using Semantic Web Technologies

Mihaela-Irina ENĂCHESCU
Bucharest University of Economic Studies, Bucharest, Romania
irina.enachescu13@gmail.com

Nowadays, the continuous demand for qualified candidates in the IT domain has empowered the use of e-Recruitment tools, which are becoming more and more exploited at the expense of the traditional methods. This study focuses on the use of ontologies in developing a job recommender system, which helps to automatically match job offers with the candidates' profiles and in reverse. In order to design the IT e-Recruitment ontology we have gathered a list with all the features a platform of this kind should provide, for both the job seeker and the recruiter. Based on the selected requirements, we developed the ontology that offers all the necessary means to implement such a job recommender system, designed to connect people with job opportunities and backwards. The second part of the paper proposes a Java based architecture to implement the e-Recruitment platform. To convert the users' input into RDF description, RDF2Go and RDFBeans APIs are employed. Storing and retrieving the data make use of the Jena Framework, which provides dedicated interfaces for accessing the Fuseki2 Server over HTTP. Finally, the prototype of the e-Recruitment platform is presented, together with its core functionalities.

Keywords: *Fuseki2 Server, IT Domain Ontology, Jena Framework, Job Recommender System, Mobile Application, RDF Representation*

1 Introduction

Since the late 1990s when the labour market faced important economy changes and had a high demand for qualified candidates, online web recruitment domain has grown steadily, surpassing the traditional methods. An e-Recruitment solution offers a significant gain in terms of efficiency by speeding up the recruitment process and saving time to all those involved. A job recommender system is supposed to help the recruiters, by recommending the most eligible candidates for a particular job and, on the other hand, to propose jobs to the aspiring candidates, matching their profiles with the existing job offers.

It is difficult for a candidate to analyse all the open positions from one or more companies and then select from them those that really fit his profile. In many cases it can occur that, although the candidate possesses all the required abilities for a job, he may not be aware of the job existence. Automatic matching between the skills and abilities that a company requires for a specific job and those an applicant has, helps decrease the errors that can occur when the process is done manually (having in mind the huge amount of CVs that need to pass by a screening process before they are deep checked). What is important to note is that these e-Recruitment platforms are assisting and not replacing the recruiters in their decision-making process.

Moreover, using an online recruitment system to connect people with job opportunities has also other benefits. For example, it provides all the needed information to do a comprehensive analysis about the labour market, including what are the skills most of the employers are seeking, what is the level of training and expertise of the candidates, what are the salary expectations of the applicants and so on.

The rest of the paper is organized as follows. The next section presents a literature review about the different approaches used to implement a job recommender system. In section 3 we propose an ontology for the IT recruitment domain that helps to automatically match job offers with the candidates' profiles and in reverse, designed to simplify and streamline the recruitment process. Section 4 presents an architecture that uses the defined ontology to implement an e-Recruitment platform, based

DOI: 10.12948/issn14531305/20.4.2016.06

on Java technologies. In section 5 we illustrate the main features of the implemented prototype of the job recommender system. Finally, section 6 presents the conclusions to our work and the future work remarks, which can improve and extend the presented solution.

2 Related Work

In the existing literature [1] [2] there are four main categories used to classify recommender systems:

- Collaborative Filtering (CF) - uses the user-to-user correlation method to predict the unknown preferences of a new user based on the preferences of similar users;
- Content-Based Filtering (CBF) - recommends items having similar content with the one the user has previously viewed or selected;
- Knowledge-Based Approach - makes suggestions based on inferences about the needs and preferences of the user;
- Hybrid Approach - combines one or more of the already mentioned methods to obtain better performance.

Developing a strong job recommender system is definitely not an easy task. There have always been different problems that needed to be solved. The first encountered problem that matching systems had was the amount of semi-structured data which contained many free text fields remaining unfilled. [3] proposes using Structured Relevance Models (SRM) as a solution for the fields that are missing when a candidate fills an online CV. The authors also emphasize that a blank field value can be inferred based on the context the other fields in the record provide. The SRM is a first example of a Collaborative Filtering approach.

Further studies in the e-Recruitment domain suggested developing an automated recommendation system based on supervised machine learning [4], technique that uses the Content-Based Filtering approach. The main idea was to recommend jobs to users using information about their past work experiences, in order to facilitate the process of selecting a new job. Job transition patterns were used to extract features from the publicly available candidates profiles found on the web. The prediction algorithm was based on matching the extracted features of the applicants with the characteristics of an employee from the company.

Other researches, such as [5] and [6], highlight the benefits of a hybrid recommender system. In [5], a system for job seeking and recruiting websites is presented and it is based on collecting information from different sources. It uses a directed, weighted, and multi-relational graph for modelling the collected data and the 3A ranking algorithm to rank features according to their relevance from the perspective of the target user. The collected data model is based on entities connections which are created using the content-based and interaction-based relations. The content-based relations suppose building the relation between applicants and job seekers in two ways such as: profile match (identifies the applicants with the same features like the ones mentioned in the job description) and profile similarity (connects the job seekers with all the users from a group that already have a relation with an applicant from that group). The interaction-based relations imply collecting from the system information about the past interaction of the applicant with a recruiter. Similarly, [6] describes a hybrid recommender system based on collaborative filtering principle which dynamically combines different algorithms to extract the features of applicants and match them with characteristics of the job description provided by the employer. The algorithms used in the previously mentioned paper are divided into the following categories based on the exploited data: algorithms which use rating behaviour of the community, content-based algorithms to extract the features and knowledge-based algorithms to research the general knowledge available in the system. The authors chose to use the hybrid recommendation approach because they considered that every user in the system is unique and different algorithms may be suitable for different users.

More recent studies, such as [7] and [8], focus more on the applicant personality in an online

DOI: 10.12948/issn14531305/20.4.2016.06

recruitment system and claim that already existing e-Recruitment platforms are not assessing personality traits. In [7] is presented a position-to-applicant matching system that performs semantic matching techniques and uses machine learning algorithms in order to rank the candidates. To take into consideration the personality of the applicant in his evaluation, he is required to provide a blog URL upon registration. Then linguistic analysis is applied to the blog posts to derive features of personality traits. [8] proposes applying techniques that use web mining to extract features of the candidate personality depending on their social media activity in microblogging platforms, like Facebook, Instagram and Twitter.

Other studies, like [9] and [10] treat the subject from a different perspective, presenting the ontology researches carried out in the e-Recruitment domain. As stated in [11] "An ontology is an explicit, formal specification of a shared conceptualization" (p. 95). Also in [11], ontologies in the field of Computer Science are presented as "a model used to describe the world that consists of establishing a set of topics, properties, and relationships. A resemblance is established between the real world and the model created by ontologies" (p. 95). In [9], M. Fazel-Zarandi and M. Fox propose using a skill ontology for defining job seekers and job advertisements, apply a deductive model to determine the kind of match between these two defined entities, and, finally, rank the applicants based on their similarity degree. The job seeker is equivalent with a set of skill statements and the job advertisement with a set of requirements (which are divided in must-have and nice-to-have), all represented using Description Logics. In [10], a reference ontology for human resource management is proposed, which is used to describe the CV of a job seeker or the details of a job offer, based on the existing HR standards (NACE, FOET etc.). This reference ontology consists of thirteen modular ontologies: Competence, Compensation, Driving License, Economic Activity, Education, Geography, Job Offer, Job Seeker, Labour Regulatory, Language, Occupation, Skill and Time. Similarly, there have been also national researches, like [12], that manifested interest in the field of ontologies and proposed using them for information technology companies.

3 The Design of an IT e-Recruitment Ontology

Before starting to design an e-Recruitment ontology, we thought about what would be the use of it and what should a job recommender system do for both the job seeker and the recruiter. The first step was to build a list with all the requirements an e-Recruitment solution should meet, which are presented in Table 1.

Table 1: The functionalities provided by the e-Recruitment solution

Functionality	Description
Create the company profile	The company/recruiter needs to provide a piece of information in order to identify itself from the other job advertisers.
Posting a job offer	The recruiter should have the possibility to post at any time a job advertisement, containing the set of requirements the aspiring candidate needs to possess.
Create the candidate's profile	Every candidate who intends to apply for a job should provide a set of personal details (to ensure his uniqueness) and a set of professional information.
List all the candidates that match a job offer	The recruiter should be able to request for any of the posted job offers a list with the most eligible candidates for that particular job description.
List all the job advertisements that match the candidate's profile	After filling his profile, the job seeker should be able to access a list with all the posted offers, ranked by their compatibility with his skills and abilities.

DOI: 10.12948/issn14531305/20.4.2016.06

List all the job offers posted by a recruiter	The company should view at any time the list containing all the job offers posted by it.
Update the applicant's profile	The job seeker has the possibility to add constantly new information in his profile or change the existing details. After the update is done, the suggested job offers that match his profile should also be updated correspondingly.
Update the company's job advertisements	The e-Recruitment solution should provide the company with the possibility to maintain the posted job advertisements. After performing the changes, the list with the eligible candidates should take into consideration the latest version of the job description.
Update the recruiter's profile	The company can update its details or add new information about its activity.

Based on all the presented requirements, an ontology was developed in order to provide the necessary means to implement such a job recommender system that connects people with job opportunities and in reverse. Figure 1 presents a visual representation of the proposed ontology, encompassing only the entities and object properties, using WebVOWL standalone application[1]. The datatype properties were excluded from the visualization, for better readability.

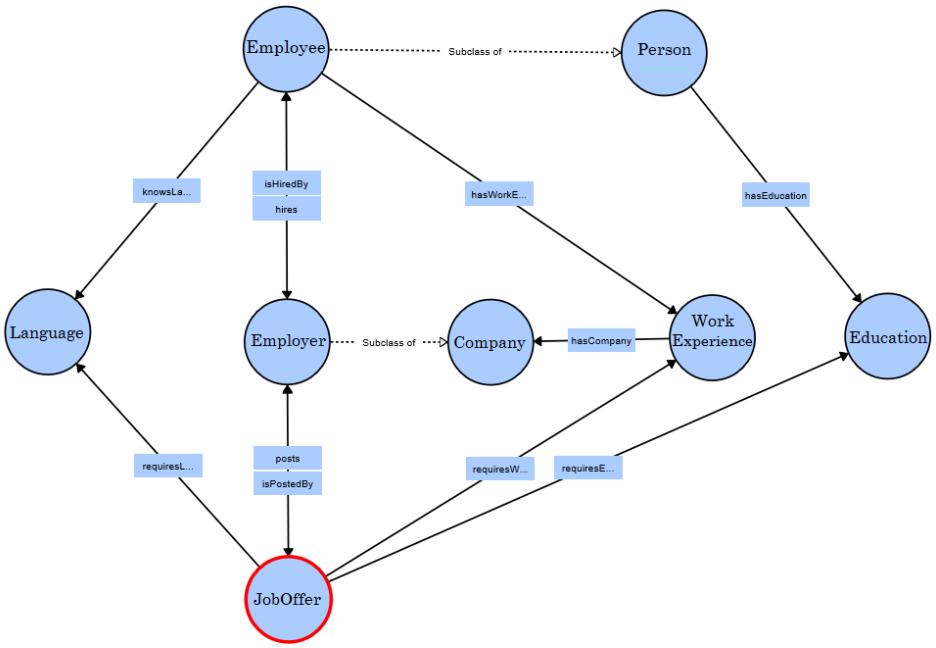

Fig. 1. Graphical representation of the e-Recruitment ontology

The ontology is composed of eight entities: *Language*, *Education*, *WorkExperience*, *Person*, *Employee*, *Company*, *Employer* and *JobOffer*. Each of these entities has its own properties, described below.

Language has three properties: name, level (to distinguish between different levels of proficiency: elementary, intermediate and advanced) and isNative to point out if we speak about a native proficiency or a foreign language.

Education is defined using the datatype properties: institution type (Elementary School, Professional School, High School, Under-

[1] http://vowl.visualdataweb.org/webvowl/index.html

DOI: 10.12948/issn14531305/20.4.2016.06

graduate, College/Bachelor's Degree, Master's Degree, MBA and Doctorate), institution name, profile, start date, end date, city and country.

To define a *Company* the following attributes were selected: name, address, TIN (Taxpayer Identification Number), website and number of employees.

WorkExperience is described using an object property: has Company (because every work experience must be linked with a company where the individual worked for) and a set of datatype properties: position, start date, end date, city, country and skill (more skills can be assigned to a particular work experience).

A *Person* is linked with the Education entity, because every person can have an education and is identified using the first name, last name, address, birth date and genre.

Employee entity is a subclass of *Person*, is hired by an *Employer*, knows *Language*, has *WorkExperience*, email, skill, expected salary (the salary value is represented in tiers, every tier having a lower and an upper limit) and a password. Both, the email and the password (stored as the hash digest) are used to provide access to the recruitment platform.

The *Employer* is a subclass of *Company*, hires *Employee*, posts *JobOffer* and has the datatype properties: contact person, username and a password (used to login on the platform). The password will be stored after applying a cryptographic hash function (for example: SHA256).

The last entity – *JobOffer* – is posted by an *Employer*, requires skill, *Language*, *Education* and *WorkExperience* and has a paid salary (also in tiers).

As it can be noticed, no knowledge level was attached to a skill, because in most of the cases this is more or less a subjective decision, the job seeker being prone to overrate his skills. To obtain a better accuracy when trying to match a job description with the candidate's profile, the applicant's position can be chosen instead of the knowledge level. For example: an employee that has Java as a skill and a work experience position of 'Senior Software Engineer' is more likely to have advanced Java knowledge, than another candidate that evaluates himself with advanced Java skill, but has a position of 'Junior Java Developer'.

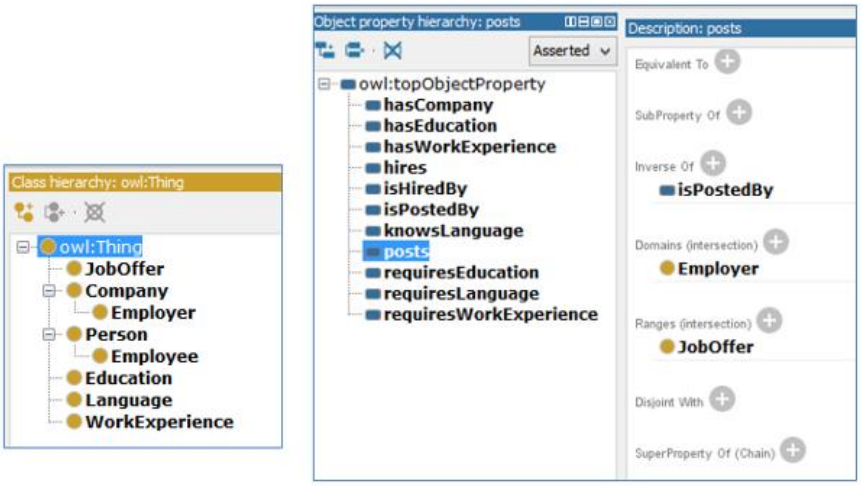

Fig. 2. The e-Recruitment ontology entities represented in Protégé

In Figure 2 the entities that compose the ontology are presented together with some characteristics of an object property. Protégé, an open-source ontology editor, was used to design the developed ontology. For the object property "posts" is emphasized the fact that represents the inverse of "isPostedBy" property, and taking a look at the domain and range, we can conclude that this property is used to describe triplets like an Employer posts a JobOffer.

DOI: 10.12948/issn14531305/20.4.2016.06

All the entities and their properties, presented as a part of the designed ontology, are used to represent the data about the candidates, the employers and job offers in a RDF format in order to store it on the server. The ontology represents the data layer of the application.

4 Using the Proposed Ontology in a Job Recommender System

In this section, we will present how the proposed ontology can be used for developing an e-Recruitment platform. The functionalities of the job recommender system were described in Table 1. What should be remembered is that the purpose of the application is to go along with the manual selection of the candidates for a particular job and not to replace the human factor involved in the selection process. Its goal is to provide a valuable support in the first stage of the selection (the screening phase), by decreasing the number of errors that can occur when thousands of CVs are manually filtered to enter in a detailed analysis or an interview.

We propose a Java based implementation for the e-Recruitment platform. The architecture of the proposed job recommender system is presented in Figure 3.

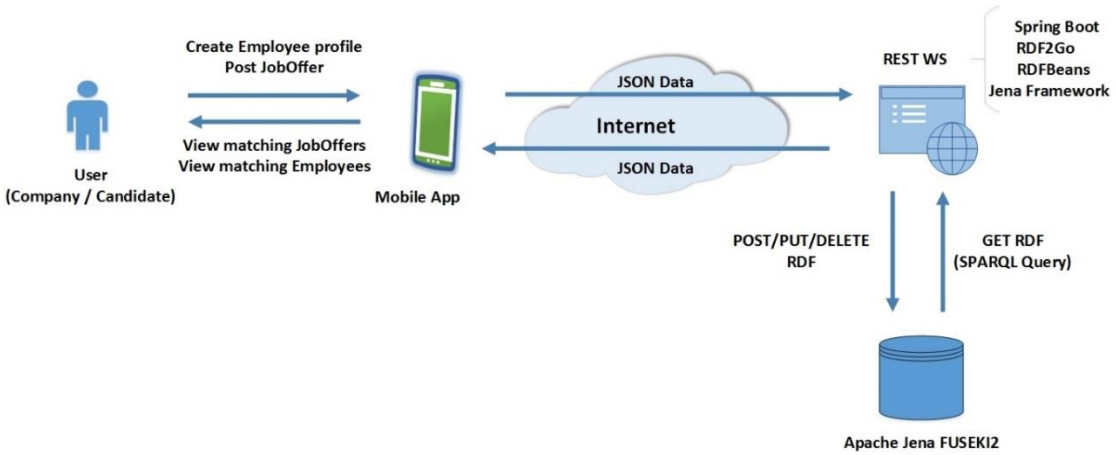

Fig. 3. The architecture of the proposed job recommender system

The e-Recruitment platform is a mobile application developed in Android, mostly because in 2015 about 82% of the sold smartphones used Android as its operating system[2]. The user, either the job seeker or the company, has the possibility to create its own profile on the platform and provide professional information or post a job advertisement, by interacting with the mobile application. Also, through the mobile application, the candidate can update its profile, adding constantly new acquired skills or gained work experience. The companies can post new job offers or delete the ones that are no longer available. Their input will be passed as a JSON request over HTTP to a RESTful web service implemented using Spring Boot Framework.

The web service (WS) should transform the received JSON input in a proper Java object that reflects an entity of the ontology (for example: an Employee or a JobOffer). The next step contains in marshalling the Java object into a RDF resource description using RDF2Go and RDFBeans APIs. These libraries provide a framework for mapping an object-oriented domain model to RDF representation, making use of some specific annotations, such as: @RDFNamespaces, @RDFBean, @RDFSubject and @RDF.

An example of a POJO class together with its annotations is presented in Listing 1.

[2] http://www.statista.com/statistics/216420/global-market-share-forecast-of-smartphone-operating-systems/

DOI: 10.12948/issn14531305/20.4.2016.06

Listing 1. POJO class Education with its specific annotations

```
@RDFNamespaces({
    "recruitment = http://www.semanticweb.org/irina-mihaela/
                  ontologies/2016/Recruitment/",
    "education = http://erecruitment.com/education/"
})
@RDFBean(value = "recruitment:Education")
public class Education implements Serializable  {
      private static final long serialVersionUID = 32838958385848872421;
      private String id;
      private String institutionType;
      private String institutionName;
      private String profile;
      private Date startDate;
      private Date endDate;
      private String city;
      private String country;
      @RDFSubject(prefix = "education:")
      public String getId() {
            return Integer.toString(hashCode());
      }
      public void setId(String id) {
            this.id = id;
      }
      @RDF("recruitment:institutionType")
      public String getInstitutionType() {
            return institutionType;
      }
      public void setInstitutionType(String institutionType) {
            this.institutionType = institutionType;
      }
      @RDF("recruitment:startDate")
      public Date getStartDate() {
            return startDate;
      }
      public void setStartDate(Date startDate) {
            this.startDate = startDate;
      }
      @RDF("recruitment:endDate")
      public Date getEndDate() {
            return endDate;
      }
      public void setEndDate(Date endDate) {
            this.endDate = endDate;
      }
      @RDF("recruitment:city")
      public String getCity() {
            return city;
      }
      public void setCity(String city) {
            this.city = city;
      }
      …
}
```

Furthermore we will continue to introduce the used annotations, defined either at class or method level, in order to facilitate the marshalling of a Java bean in one of the RDF formats (XML/RDF, Ntriples, Turtle, N3):

- @RDFNamespaces – it is used for a class or interface to specify one or more namespace prefixes using the following syntax <*prefix*> = <*uri*>. They are useful

DOI: 10.12948/issn14531305/20.4.2016.06

to simplify the way in which the triplets' uri are accessed.

- @RDFBean – it is used for a class or interface to emphasize that it is a RDFBean. Also it has attached a qualified name or an uri that is used when accessing a subject/property/object of the triplet.
- @RDFSubject – it is applied on a getter method to highlight the fact that this method will be used to return the unique identifier of the RDF bean. Beside the annotation, a prefix parameter can be added (uri of a namespace or a reference to an already defined one in the @RDFNamespaces annotation), that will be used in the identifier generation. If the prefix is not explicit given then the getter method is considered accountable to generate the name completely qualified.
- @RDF – it is applied on a getter method to declare the uri for a RDF property.

For the Education class, presented in Listing 1, is shown below how a Java object is marshalled in a triplets representation (in RDF every statement is represented as a triplet containing a subject, a predicate and an object) based on the defined annotations.

```
@prefix j.1:   <http://www.semanticweb.org/irina-mihaela/ontologies/2016/Recruitment/> .
<http://erecruitment.com/education/-93846990>
        a                    j.1:Education ;
        j.1:institutionName  "The Bucharest University of Economic Studies" ;
        j.1:institutionType  "Master Degree" ;
        j.1:profile          "Economic Informatics" .
```

The <http://erecruitment.com/education/-93846990> uri is generated using the prefix specified together with the @RDFSubject annotation, concatenated with the result of the hash method applied on the Java object that has Education class. It is emphasized that the uri identifies an entity of j.1:Education type, declared using the @RDFBean annotation. Forwards, for every property an uri is created using the information provided together with the @RDF annotation.

After the RDF conversion is done, further step is to persist (remove or replace, depending on the initial request of the user) the triplets on the Apache Jena Fuseki2 Server using Jena Framework. Apache Jena Fuseki2 is a SPARQL Server that provides the SPARQL 1.1 protocols for querying and updating over HTTP. In order to persist a RDF model, DatasetAccessor interface can be used, which provides methods for adding, removing or replacing a model.

Listing 2. The method that persists a RDF model on the Apache Jena Fuseki2 Server

```
public void persistModel(org.ontoware.rdf2go.model.Model model, String
service) {
      Model jenaModel = ModelFactory.createDefaultModel();
      ByteArrayOutputStream out = new ByteArrayOutputStream();
      InputStream in = null;
      try {
            model.writeTo(out, Syntax.RdfXml);
            in = new ByteArrayInputStream(out.toByteArray());
            jenaModel.read(in, "RDF/XML");
            DatasetAccessor accessor =
                  DatasetAccessorFactory.createHTTP(service);
            accessor.add(jenaModel);
      } catch (ModelRuntimeException | IOException e) {
                  …
      } finally {
            …
            jenaModel.close();
      }
}
```

DOI: 10.12948/issn14531305/20.4.2016.06

In Listing 2 is extracted a code snipped from the persistModel method. This function is responsible to convert a RDF2GO model into a Jena model, in order to have it persisted on the Fuseki server, after a HTTP call. In order to exchange the information between the two models, first the content of the first model is written in an output stream, then an input stream is added on top of it. The input stream will represent the source used when creating the Jena model. After the Jena model is loaded, a connection with the dataset from the Fuseki server is established, using the DatasetAccessor interface, and the triplets are persisted.

Furthermore, when the user tries to access the list with the matching employees or job offers (depending on the user type: company/candidate), he will send a request that has its details and the WS is responsible to build a SPARQL query that should retrieve from the Fuseki2 Server all the RDFs that match with the provided input. QueryExecution interface offers the necessary means to execute a SPARQL query, accessing the Fuseki2 Server over HTTP.

Listing 3. The method used to get all the job offers that match an employee profile

```
public List<JobOffer> getJobOffersThatMatchEmployee(Employee employee,
                                                    String service) {
      List<JobOffer> offers = new ArrayList<>();
      StringBuilder query = new StringBuilder(MatchingQueries.
                  matchEmployeeWithJobOffers);

      ...
      QueryExecution qe = QueryExecutionFactory.sparqlService(service,
                                    query.toString());
      ResultSet results = qe.execSelect();
      while (results.hasNext()) {
            JobOffer offer = new JobOffer();
            QuerySolution soln = results.nextSolution();
            String offerId = String.valueOf(soln.getResource("job"));
            offer.setId(StringUtils.substringAfter(offerId,
                  "http://erecruitment.com/jobOffer/"));
            offers.add(offer);
      }
      qe.close();
      return offers;
}
```

Listing 3 illustrates an extract from the method used to obtain all the job offers whose requirements may appeal to a candidate, taking in consideration his profile. Inside the function the connection with the Fuseki server is set and the constructed query is sent (the algorithm to construct the query is explained in Figure 4), in order to be executed. The response after the query is applied on the dataset consists in a ResultSet, from which the offer' details are accessed one by one, based on their properties name. Finally the server connection is closed and the list with all the found jobs is returned.

DOI: 10.12948/issn14531305/20.4.2016.06

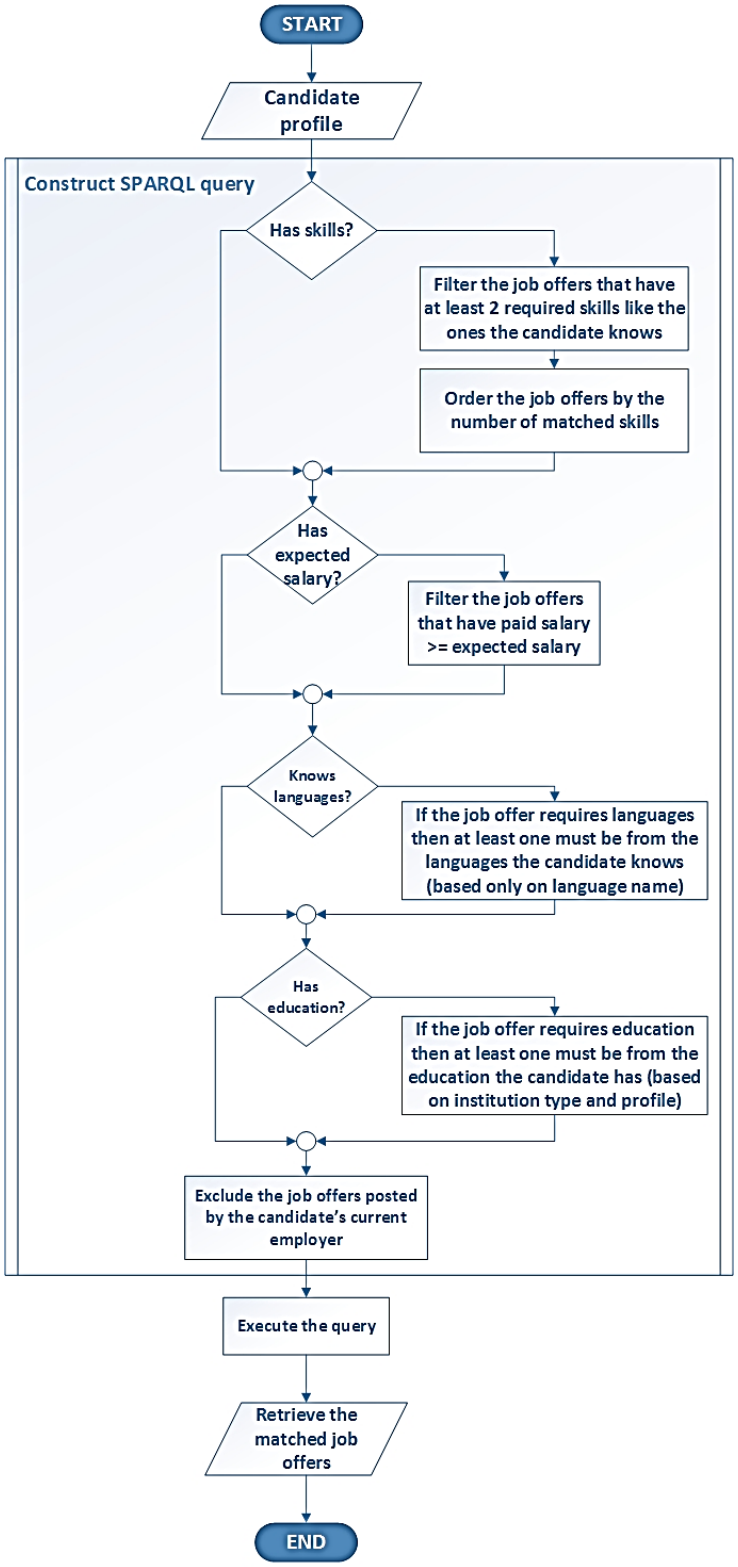

Fig. 4. The algorithm used to construct the SPARQL query in order to find all the job offers that match an employee profile

Figure 4 illustrates the algorithm behind the logic of finding all the existing job offers in the application that are suitable to a particular candidate, based on his current experience. As it can be noticed the offers are selected to match at least two of the candidates' skills (and ordered by the number of matching skills), to have the paid salary between the expected limits set by the candidate and if the offers requires a particular language to be

DOI: 10.12948/issn14531305/20.4.2016.06

known or an education, in case the profile has those details filled in, then it looks for at least one match for each of the criteria. Finally, the offers posted by the candidate current employer are excluded.

5 Results

Aiming to provide the user with all the flexibility he needs, the e-Recruitment platform was implemented as a mobile application. The graphical interface of the application is composed of multiple android activities/fragments, linked together in a logical sequence. Forwards, some of the main screens of the application will be presented.

Any person that seeks a new job or simply wants to be updated with the latest trends in the IT job market has the possibility to create an account in the recruitment platform. After the user registration and successfully logging in the application, the candidate is directed to the Home page. In this section the user can add several basic details about his profile (first name, last name, address, birthdate, expected salary and so on, characteristics for the job seeker) and can also access the application menu, that permits navigation through other particular screens for viewing/adding/updating the rest of the profile sections (education, skills, known languages and gained work experience).

Every time the candidate acquires a new skill or the data he provided on the platform becomes outdated, he has the possibility to update his profile (by changing the existing information or adding a new one), so that his account reflects closely the reality. It is highly recommended for the user to keep his profile up to date, because this will help him to reach job offers that best suit his abilities.

As expected, when a change occurs in the candidate's account, accordingly, the list will all the job offers that match his profile is automatically updated to consider the new information.

Starting with the moment when the job seeker has created a profile on the e-Recruitment platform, he will permanently have access to a list with all the job offers (presented in Figure 5), which are available in the application and have similar requirements to his background. At any time he can view that list and look for details about every job description.

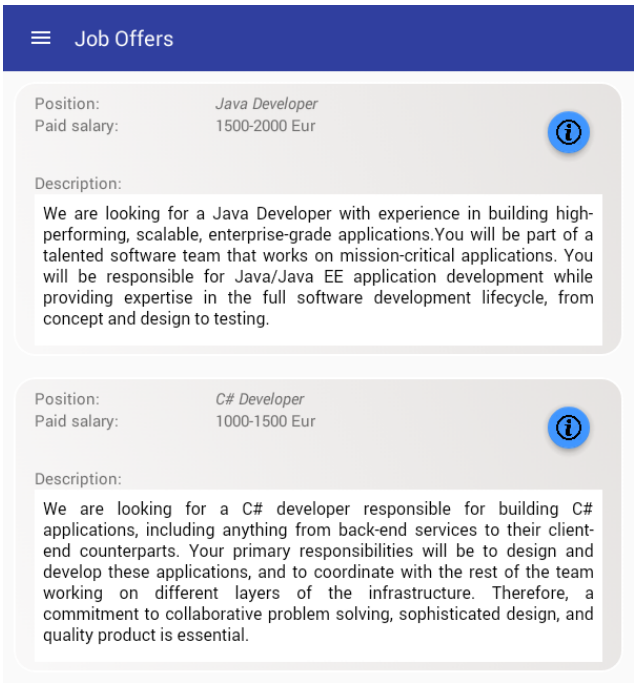

Fig. 5. The screen that displays all the job offers that match the candidates' profile

DOI: 10.12948/issn14531305/20.4.2016.06

Taking in consideration the algorithm presented in Figure 4 that is used to determine all the jobs that match an employee profile we can see that the linkage between what the candidate knows and the job requirements is not a perfect match. Instead, the algorithm is focused on finding offers that the candidate will find attractive, based on his professional background, but not limited only to his skills. In this way, the application aims to help the job seeker to understand what the job market trends are and what are the other skills that the companies are looking, in addition to what he already knows. The purpose is to encourage the candidate to learn new foreign languages or technologies, in order to keep up with the market evolution in the IT sector.

On the other part, once the company registers on the platform (has created a valid profile that can be used to login), it has the possibility to post one or multiple job offers. For every job offer the employer can add the position, short description that includes the given benefits, paid salary in a range, and the requirements that are expected (including skills, languages, education, previous work experience).

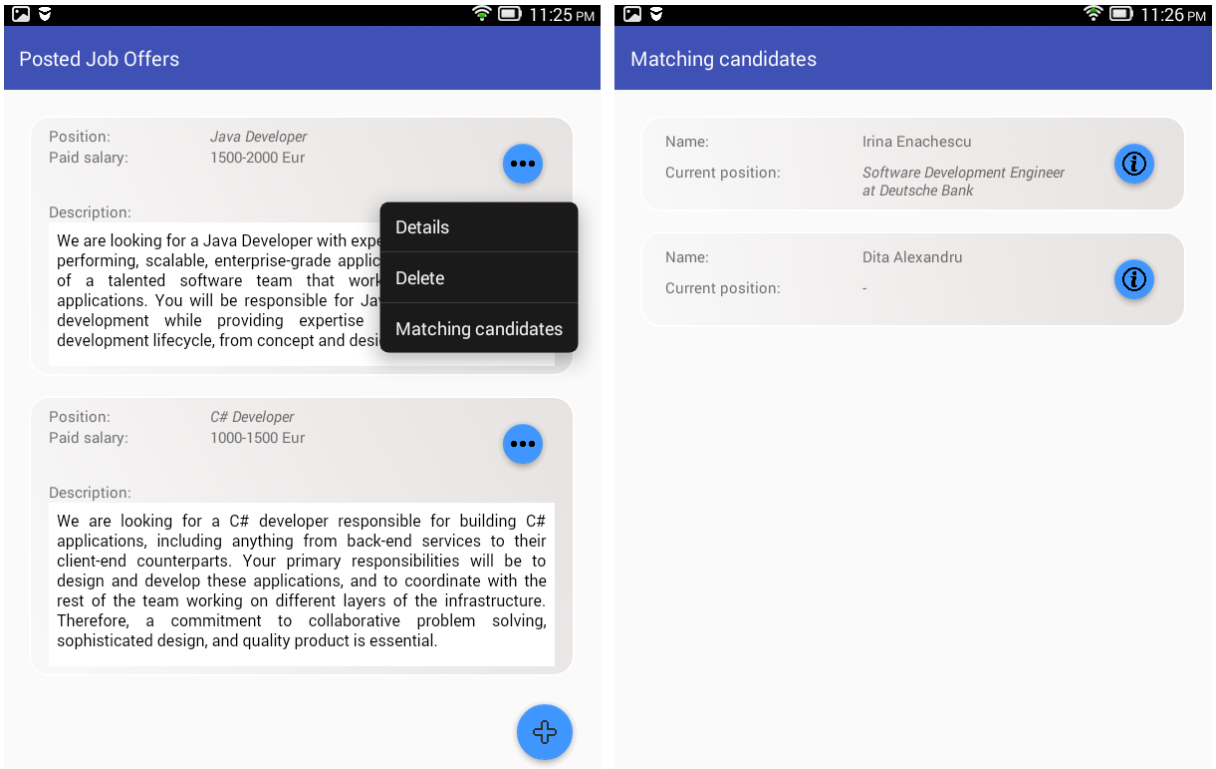

Fig. 6. The screen that displays all the job offers posted by a company and the matching candidates for a particular job

When for a job offer a suitable candidate is found and the job becomes unavailable, the company may consider deleting it from the platform. As shown in Figure 6, at any time the employer can access the complete list with all the available job offers, posted by him. The list consists in a summary presentation of every offer (position, paid salary and description), giving the possibility to view more details for every particular job.

Through the created e-Recruitment platform, once the company has posted a new job offer, it can also access the list with all the registered candidates in the system, that fulfil all the requirements posted in the job description. Moreover, if the job seeker has allowed to be contacted from the recruiters, then the employer may sent him a notification (via email/telephone) to show that it is interested in a collaboration. If the previous matching (a candidate profile with a job) was not looking

DOI: 10.12948/issn14531305/20.4.2016.06

to find a perfect correspondence, the algorithm based on how a job is linked with the suitable candidates targets to find someone that completely match all the requirements.

6 Conclusions and Future Work

In this paper, we proposed an ontology that is used for developing an e-Recruitment platform in the IT field and designed the architecture of it, together with the technologies that are used in the implementation. Afterwards, a prototype of the job recommender system was developed, based on the Semantic Web technologies described in the recommended system architecture.

Directions for future research include matching the job advertisements with the applicants profiles based on a compatibility ratio, instead of trying to find the perfect match. The case when the job seeker profile fulfils exactly the requirements specified in the job offer seldom happens. Therefore, instead of receiving an empty list with the best ranked candidates, it would be useful to access one with them ranked by their compatibility ratio. The lower limit of the matching level should be defined by the user, to avoid displaying results that slightly match the requirements. Another step that will be made in order to improve the overall performance of the e-Recruitment platform is to include the personality traits of the candidate in the matching process. Furthermore, the proposed ontology should be extended in order to broaden its area of use in other domains apart from the IT field.

Acknowledgment

Parts of this research have been published in the Proceedings of the 15th International Conference on Informatics in Economy, IE 2016 [13].

References

[1] S. Al-Otaibi and M. Ykhlef, "Job Recommendation Systems for Enhancing E-recruitment Process" in Proceedings of the International Conference on Information and Knowledge Engineering (IKE), Las Vegas Nevada, USA, 2012, pp. 433-439.

[2] W. Hong, S. Zheng, H. Wang and J. Shi, "A Job Recommender System Based on User Clustering", JCP, vol. 8, no. 8, pp. 1960-1967, August 2013.

[3] X. Yi, J. Allan and W. Croft, "Matching resumes and jobs based on relevance models", in Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval - SIGIR '07, Amsterdam, The Netherlands, 2007, pp. 809-810.

[4] I. Paparrizos, B. Cambazoglu and A. Gionis, "Machine learned job recommendation", Proceedings of the fifth ACM conference on Recommender systems - RecSys '11, Chicago, Illinois, SUA, 2011, pp. 325-328.

[5] Y. Lu, S. El Helou and D. Gillet, "A recommender system for job seeking and recruiting website", Proceedings of the 22nd International Conference on World Wide Web - WWW '13 Companion, Rio de Janeiro, Brazil, 2013, pp. 963-966.

[6] S. Dooms, "Dynamic generation of personalized hybrid recommender systems", Proceedings of the 7th ACM conference on Recommender systems - RecSys '13, Hong Kong, China, 2013, pp. 443-446.

[7] E. Faliagka, L. Iliadis, I. Karydis, M. Rigou, S. Sioutas, A. Tsakalidis and G. Tzimas, "On-line consistent ranking on e-recruitment: seeking the truth behind a well-formed CV", Artificial Intelligence Review, vol. 42, no. 3, pp. 515-528, July 2013.

[8] E. Faliagka, M. Rigou and S. Sirmakessis, "An e-Recruitment System Exploiting Candidates' Social Presence", Current Trends in Web Engineering, vol. 9396, pp. 153-162, September 2015.

[9] M. Fazel-Zarandi and M. Fox, "Semantic Matchmaking for Job Recruitment: An Ontology-Based Hybrid Approach", Proceedings of the 3rd International Workshop on Service Matchmaking and Resource Retrieval, Washington D.C., USA, 2009.

[10] A. Gómez-Pérez, J. Ramírez and B. Villazón-Terrazas, "An Ontology for Modelling Human Resources Management

DOI: 10.12948/issn14531305/20.4.2016.06

based on Standards", in *11th International Conference on Knowledge-Based Intelligent Information & Engineering Systems*, Vietri sul Mare, Italy, 2007, pp. 534-541.

[11] M. Sicilia, *Handbook of metadata, semantics and ontologies*. Singapore: World Scientific Pub. Co., 2014.

[12] C. Niculescu and S. Trausan-Matu, "An Ontology-centered Approach for Designing an Interactive Competence Management System for IT Companies", *Informatica Economică*, vol. 13, no. 4, pp. 159-167, 2009.

[13] M.I. Enăchescu, "Using Semantic Web Technologies in e-Recruitment: A Java Based Architecture", in Proceedings of the 15th International Conference on Informatics in Economy (IE 2016), Cluj-Napoca, Romania, 2016, pp. 146-152

Mihaela-Irina ENĂCHESCU has graduated the Faculty of Cybernetics, Statistics and Economic Informatics in 2014. In 2016 she has graduated the Economic Informatics Master program and she currently pursues a PhD research in Economic Informatics at the Bucharest University of Economic Studies. She is working as a Java Software Developer and is also a teaching assistant in the Department of Economic Informatics and Cybernetics. Her research interests are: Ontologies, Semantic Web and Data Mining.

DOI: 10.12948/issn14531305/20.4.2016.06