INTELLIGENT E-JOB MARKETPLACE

Nuno Nobre Marques
Department of Electrical and Computer Engineering, Instituto Superior Técnico, Av. Rovisco Pais 1, 1049-001 Lisboa, Portugal

In this article, one is concerned with the steps towards the development of a prototype of a job web site that relies on intelligent computing to obtain the matching degree of candidate profiles and job offers compared to a model. It's analyzed the state of the art followed by some of its theoretical basis. Then, the architectural structure of the application is analyzed to ensure that the prototype is fault tolerant and scalable. It's also mentioned the data model that supports all of the data provided by the users, and detailed all of the implemented features, allowing candidates to manage their profiles and view offers, and employers to manage their job offers and view candidate profiles. It's also explained the developed algorithm and discussed the obtained results.

keywords – candidate, employer, job, matching, site

I. INTRODUCTION

The process of recruiting a candidate with specific professional characteristics is a major concern by the employing companies. Most of the companies possess specialized departments to manage their human resources and to find and hire new workers for their needs.

With, the greater use of the internet, information became faster to be accessed causing a major impact in the recruiting process due to the increased amount of available information.

This allowed the development of some web applications concerning this theme, centralizing that information and providing it to candidates and employers. As job sites were being developed, the amount of information available and the features provided were increased. However, not only job sites, but in a general way, most of the web sites feature simple searches. This means that the users (candidates and employers) have some difficulty to analyze the results obtained while searching in a job site, becoming this analysis a slow and dull task. That's because the result set of the search can have a lot of entries and sometimes it's necessary to look at them one by one to get the required data. This means that, most of the times, the sub-set of candidate profiles to be analyzed is chosen in a somehow random way.

The purpose of this work is to present the steps towards the development of a job site that provides the main features found in existing job sites, but including a new type of search that allows users to apply most of the information provided by the other users to create a search criteria and to order the list of results on that criteria.

The steps followed to develop the application are classified in two major categories: the analysis and the implementation. The analysis consists of the description of the architecture of the application, the description of the entities created to structure all the information, the developed features and the presentation of the algorithm, developed and included in the application.

II. STATE OF THE ART

With the purpose of acquiring information regarding the features intended to be developed in the application, some job sites were visited and therefore the main features were analyzed, mostly in terms of the searches provided.

In general, all of the visited sites provided the candidate profile and employer offer insertion, as well as searches by job area or category and by job. Keyword search is also provided in most of the analyzed, as well as city search and in international job sites, the search by country. Other search features found only in some job sites, included candidate experience and academic degree.

Some other features provided, but just in some of the sites, included the submission of candidate profiles to job offers, newsletter sending, mailbox to allow candidates to exchange mails with employers and email sending when a candidate inserts a profile classified in a job category or when an employer inserts a job offer in the same category. Other interesting feature is the insertion of old jobs, found only in one of the analyzed sites.

There wasn't any job site found that provided a more advanced search than the search for one or several criteria, being this criteria somehow limited, and therefore we conclude that the searches provided by the existing sites are not as advanced as the search featured by the developed application.

III. USED TECHNOLOGY AND CONCEPTS

As the developed application was intended to be a job site, and therefore a web application, among the several possible choices, Django framework [1] was the chosen one to develop
the application. **Django** provides middleware intended to ease web development for applications that use a database to persist data and need to generate dynamic web pages to be accessed with a web browser. Other analyzed choices were Ruby on Rails, a rather similar framework, and Java EE. **Django** and Rails provide a faster development than Java and **Django** was chosen because it's written in **Python** [2], a better known language by the author.

**Django** follows the DRY (Don't Repeat Yourself) principle and so it's structured in a way that allows developed code to be written once and used many times. It uses the MVC software pattern, that consists in a three layer architecture. The three layers are the database access layer (Model), that handles the communication with the database, the logic layer (Controller), that contains the logic flow of the application and the presentation layer (View), where the presentation and generation of dynamic pages is handled. **Django** also provides support for general and customized configuration, URL mapping, a unit test environment, form handling, interactive shell, database synchronization as well as a test server to help developing the application with automatic reloading of the developed code. It also provides an advanced templating system, that allows inheritance and inclusion between templates to improve the reuse of the dynamically generated web pages.

**Django** is entirely developed in **Python**. **Python** is an open-source programming language used for application development and for scripting. It's portable across a variety of operating systems, has a simple syntax and is interpreted, so the time compiling the code is spared. It is also an Object Oriented and a dynamic language, not requiring variable types to be explicitly defined. **Python** also includes several libraries that allow the usage in many different types of application for many domains. In this project it's used version 2.7 because the new version 3.0 isn't yet supported by **Django**.

The application was developed using the Test Driven Development (TDD) software development process [3]. TDD was introduced as a component of the programming concept Extreme Programming (XP) and defines a set of simple rules to improve software development.

As shown in Fig. 1, the TDD is an iterative process. The purpose of this process is to add, in each iteration, one single feature to the application, by adding one failing test and then code the feature in the application to allow the written test to pass. Therefore, the application is developed with tests for all of its features. When an application, after some development, starts to be a complex application, the risk of failure increases, while changing an existent feature or developing a new one, because of the existing code dependencies. Using TDD, if a feature is changed, the existing tests allow the programmer to know exactly where the points of failure are located and can easily and securely modify the code accordingly. The TDD process allows also programmers to be sure that the developed features are the ones necessary for the application to have the expected behavior.

This project was developed using TDD and therefore shares the benefits of its usage. As said before, **Django** provides the features to use TDD.

Being a web application, the data transfer between the server where the application is being executed and the client that makes requests to the server and retrieves data from the server, should be also analyzed. The request-response cycle is a data exchange pattern often used by client-server architectures. The client-server architecture is a distributed structure, where the requester (client) and provider (server) have different roles concerning their purpose, and where clients start the communication and retrieve data from servers.

In Fig. 2. is shown the client-server architecture with a server being accessed by several clients.

The main protocol used to transfer data between client and server is the HTTP (Hyper Text Transfer Protocol) [4], which is an application layer protocol in the OSI (Open Systems Interconnection) model requiring a reliable host-to-host transport layer protocol provided by TCP (Transmission Control Protocol). The HTTP protocol was developed to transfer hypertext documents that reference other documents. This referenced documents, that can be accessed only by a click or a key sequence, are identified by a URI (Uniform Resource Identifier), or specifically by a URL (Uniform Resource Locator). URIs and HTML (Hyper Text Markup Language) form a system of interconnected resources that allowed the birth of the World Wide Web in 1990.
The HTTP protocol in its current version HTTP 1.1, provides a standard way to request data using methods (GET, POST, HEAD, PUT, DELETE, OPTIONS and CONNECT), allowing connection reuse, the use of cached resources and providing two authentication methods: basic and digest. The protocol also defines response codes to allow clients (browsers) to interpret the response, acting accordingly, and provides the use of cookies allowing the management of state.

The HTTPS (HTTP over SSL – Secure Socket Layer) [5] is a protocol similar to HTTP, but establishing a secure connection host-to-host. The data transfer in HTTPS is encrypted unlike HTTP. However by using encrypted data and by establishing a secure connection, which can delay the start of the communication process, HTTPS also requires much more data to be transferred. Therefore, in a request that doesn't contain private data, like passwords, HTTP should be used to achieve minor data transfer and faster responses. In a request containing critical data, HTTPS should be used, despite the overhead of data.

The HTTP 1.1 protocol was developed fulfilling several specification introduced by the REST (Representational State Transfer) concept [6]. REST concept is a distributed system architecture for hypermedia content. It introduces a set of restrictions which define the web architecture. Those restrictions are the use of a client-server architecture, the stateless interaction between client and server, the use of cache, the use of an uniform interface between components, a layered system and code-on-demand (optional). REST has three types of elements: information elements, connectors and components. The information elements are the resource, resource name, resource identifier, resource representation, resource metadata, representation metadata and control data. Connectors are elements that can establish connections, like the client, server, a DNS or a tunnel. Components are the origin server, the gateway, the proxy and the user-agent.

In Fig. 3 are shown the components, connectors and the use of cache. REST allows the use of many of specifications from HTTP 1.1 protocol. HTTP 1.1 protocol can be used to manage the resources defined in a REST interface. For instance, if a resource “book”, named “books”, has the resource identifier “http://myserver.com/books”. Requesting the resource using HTTP method GET, the list of all books is retrieved. Instead, if POST is used with book information in the body of the request, one new book is created. Likewise using PUT method, the information of a book can be edited and using DELETE method one book can be deleted.

This allows resources to be managed using HTTP methods, which translates into cleaner URLs, hierarchical mapping of resources and a systematic way of retrieving and managing the resources. The resources of the developed application use REST interfaces and shares the benefits of its use.

Many web applications respond to requests using HTML (Hypertext Markup Language). This language is interpreted by the clients of this applications, mostly browsers, in order to process the response from the server and present the received data to the user. The elements of this language are defined by a starting tag and end tag and are allowed to have attributes. The language is recursive so one can have elements inside other elements. One of the features of HTML is the data type definition. HTML is used to transfer data, although allows the definition of other components like CSS (Cascade Style Sheets) that can be used to format the data presented to the user, as well as the use of a scripting language for client-side scripting like Javascript. HTML can also be used to ask the user to introduce some data throughout forms. They are used in order to allow users to insert data to be sent to the server in a request. Therefore it's achieved a bidirectional flow of information from user to application and from application to user, required for a web application to achieve its purpose if the application is more complex than just an institutional site, in which the flow of data is strictly from application to user.

In order to persist the data sent by the users, the application should have a DBMS (Database Management System), to ensure that that data is correctly and effectively saved and can easily, but not insecurely, be accessed. The purpose of a DBMS is to store large amounts of data and its management and retrieval in an effective way [7]. The most commonly used DBMS require data to be stored using a relational data model.

This model has its own algebra (relational algebra) in order to store data in a structured way, allowing the use of SQL (Structured Query Language) for data management. The relational model consists in a set of tables in which a table maps an entity. Tables can contain sets of data with the same characteristics or attributes. Tables can also be related and its relations are defined by the relational algebra. The reason for naming this model as relational model, is that there is a match between table relations and the mathematical concept of relation defined by the algebra. As said before, Django provides middleware in order to manage the used relational DBMS and allowing database abstraction and this feature takes a key part in the developed application.

IV. DEVELOPED APPLICATION

A. The server architecture

After the choice of framework and the analysis of some required concepts, it will be detailed the most significant features of the developed prototype. The application was designed to be scalable, fault-tolerant and to allow load balancing and is structured in a layered architecture. Each layer has its own role and may have several elements in order to be fault-tolerant and allow load-balancing between each
element of the layer. More elements of that layer can easily be
added with the selected structure, which allows the application
to scale as wanted.

As shown in Fig. 4, the structure of the developed application has
three layers, each one with its specific role. The first layer consists in
web servers, in which, one is chosen to receive the request as it arrives
from the cloud representing the internet. Before the request is processed,
the first server decides which server the request is handled or if handles it
himself. Still in the first layer, the request is processed by the
chosen server. If it consists of static content, like static HTML
files, CSS files, images, Javascript or other static files, the
request is processed and the response is sent to the client.

Otherwise, the request is handled to the second layer that
features application servers. Each application server is also
a web server, but implementing the WSGI (Web Server
Gateway Interface), which allows Python code to be executed
and contains the code developed using the Django framework.
This instances will serve exclusively the non static resources of the
application. If it had been used Jython instead of Python
as the Python interpreter, one could use a Java servlet
container instead of a web server with WSGI, because Jython
is the Java implementation of Python.

The processing of the request in the second layer, often
requires database access and manipulation. This is provided by
the third layer. A DBMS with clustering was chosen to keep
the initially discussed characteristics of fault-tolerance, load-
balancing and scalability also in this third layer.

In order lighten database overloading, one instance of a
distributed cache was placed by each server on the second
layer. This allows fast communication between the respective
server and the cache. The cache is also used to keep the
sessions of the logged users, allowing the servers to be
stateless. This way the sessions are distributed among all the
cache instances allowing each server of the second layer to use
its own cache to get the user sessions as needed. The used
cache was a distributed cache that handles transparently the
replication among the several instances. Also Django provides
middleware in order to easily use a cache mechanism.

B. Entities

In order to structure the information provided by the users,
some entities were created to represent that information in the
application (Models) as well as in the DBMS (Tables).

As this is a web application, it’s accessed by a set of users.
The first entity defined is the User. The User entity contains
the attributes that allow each one of the users to be
authenticated and authorized, as well as register date and if is
active or not. The User is typified as candidate or employer.
For that purpose Candidate and Company have been defined.
These two entities contain attributes that hold the personal
data of candidates and the general data of the employers. A
candidate can have profiles, model offers, bookmarked offers,
submitted offers and old jobs. This implied the definition of
CandidateProfile to hold the information of the profiles
inserted by the candidates, CandidateModelOffer to hold the
information regarding model offers to search offers with the
classification algorithm, CandidateBookmarkerOffer to link
candidates to selected offers, and CandidateAppliedOffer to
link candidates with its applied offers and allow the employer
owner of that offer to know that it has been applied. Also was
defined the entity CandidateOldJob to hold the data regarding
previous jobs. The candidate profile can have professional
courses attended, technologies used and functions performed
by the candidate. This implied respectively the definition of
the following entities: CandidateProfessionalCourse,
CandidateAreaMain Technology and
CandidateAreaMainFunction.

On the other hand, employers can place offers, have
bookmarked candidate profiles, applied candidate profiles and
model profiles. This implied the definition of the entities
CompanyOffer to hold the data of an offer placed by the
employer, CompanyBookmarkedProfile to link employers to
selected profiles, CompanyAppliedProfile to link employers
with their applied profiles and allow the respective candidate
to know that that profile had been applied, and
CompanyModelProfile to hold the data regarding model
profiles in order to search profiles with the classification
algorithm.

Apart from this entities that hold the information regarding
candidates and employers, there was the need to specify other
tentities in order to hold other relevant information for the
application. As so, there have been defined the Country to
hold the several countries, City to hold the several cities
belonging to the countries, Job to hold all the possible jobs,
JobArea to hold the job categories, University to hold the
several universities, AcademicCourse to hold the courses of
the universities and AcademicDegree to hold the data
regarding the different academic degrees. This entities where
defined to standardize user choices of respective attributes and
to minimize insertion ambiguities.
In some of this attributes there can be a degree of similarity between instances. For instance, a university can have several courses similar to the courses of other university. Thus, it would be interesting to define the degree of similarity between some of those instances belonging to the same entity. For that fact, some other entities were defined: UniversitySimilarity, AcademicDegreeSimilarity, UniversityCourseSimilarity, JobAreaSimilarity and JobSimilarity. If the similarity between two instances of the same entity isn't defined, its degree of similarity defaults to not existent or zero.

Beyond the above entities and for the purpose of user registration, the entity AuthenticationTicket has also been defined to hold the created tokens and token data like state, creation date, expiration date and usage date.

C. Resources

Applying the REST concept, the developed application provides resources for users to access. The initial page, allowed to be accessed by all the users registered or not, contains a list of the latest job offers inserted by employers and statistics of the application usage such as registered candidates, registered employers, candidate profiles and job offers and is mapped on the root. This means that if the application URL is “http://myserver.com”, anyone accessing this URL will retrieve the initial page. As defined in a REST interface, the candidate resource is mapped as “http://myserver.com/candidate” and the candidate profile resource is mapped as “http://myserver.com/candidate/profile”.

From now on, every resource referred has a REST interface, but its URL will not be mentioned. The initial page also contains the links for a new candidate and employer registry and the login form.

After a user logs on to the application, depending on the type, he will be redirected to the respective home page. If instead the login process fails, a message will be shown explaining why the login process could not be completed and he will remain in the initial page. If a user, for instance a candidate, tries to access another user's resource for which he has no authorization, the action is blocked and he will be redirected to the initial page.

When arriving at his home page, a candidate will have a menu with several options and will be able see a short list with the latest inserted offers. In this list, the candidate can see each offer detail, and bookmark each offer as well. The options in the menu are the return to this home page, check and edit personal data, change password, insert new profile, list inserted profiles, list notifications by the employers of own profiles, list the detail of latest offers, list bookmarked offers, search offers, insert model offers, list model offers and list searched offers sub-sets. By checking personal data, the candidate can list, insert, edit and delete his old jobs. The candidate can also edit his personal data, change his password, insert a new profile, where he has to fill a form with some information to be displayed to the employers, selecting which fields are visible. The candidate can also list his inserted profiles, and in this list he can edit or delete each one of the listed profiles. On each one of them the candidate can also add, edit and delete professional courses, technologies and functions attached to the respective profile. In the menu, the candidate has an option to list the profiles applied by employers and in this list he can see which employers applied the profile, and can list the offers belonging to each one of them. The candidate also has an option in the menu to list his bookmarked offers, with an option for detail, appliance and unmark in each one, and for the applied offers where he can also see the detail and cancel the appliance in each one.

The candidate can search offers using two distinct methods. The first one consists in a search by an attribute or a set of attributes filled in the search page where he can select which attributes he wants to use and execute the search. Then it will be displayed a list of offers that match the exact criteria used for the search, as well as as the number of total offers found and displayed. In this list he can see the details of the displayed offers as well as bookmark and apply each one of them. To search by the classification algorithm, the candidate must insert model offers. Throughout this model offers, that will be detailed further ahead, he can use this search to list the offers available that are more similar to the model offer inserted. The list of offers and the options available have an equivalent display to the search by criteria. Finally the candidate can apply this search method to a sub-set of offers obtained with a previous search using also this method. To exit the application the candidate should use the logout link from the menu.

Employers have also a home page where they arrive after the login. In this page en employer can see a short list of the latest inserted candidate profiles with options to see the detail and bookmark each one of the profiles, as well as a menu with options. Like the candidate, the employer can see and edit the company data, as well as change his password. The employer can insert new offers, list, edit and delete inserted offers as well as list bookmarked and applied candidate profiles. In the list of bookmarked profiles, he can see the detail, apply and unmark each one of the profiles and in the applied profile list he can see the detail and remove appliance for each one of the profiles. Other option available is the list of own offers applied by candidates where there is an option to display the candidate detail and profiles.

As the candidates, employers can search candidate profiles using two methods: the search by an attribute or a set of attributes filled in the search page where he can select which attributes he wants to use and execute the search, and the search by the classification algorithm. These searches are very similar to the candidate searches although with different attributes and based on profiles instead of offers. They have also the option to use the classification algorithm to search in a sub-set of results obtained with a previous search with the same method. To exit the application, the employer should click on the logout link from the menu.

Each appliance, whether on a profile or on an offer sends an email to the respective candidate or employer to warn that a profile or an offer has been applied.
D. User registry and authentication

User registry is a sensible process because private information from the user must be sent to the server. For this purpose, it's used the HTTPS protocol for the registry an to log users in the application. The user registry process starts by asking the user to fill a form containing some data. If the user is a candidate, he must fill his personal data and if the user is an employer, he must fill the company data. After the submission of the respective form, a message is displayed to inform the user that an email was sent to his mailbox containing instruction for completing the registry. Meanwhile, the user had already been created but in an inactive state and a token created and sent to the user's email. The email contains a link with the token as parameter, and if the user presses the link, activates his account and is allowed to log on to the application the next time. This process assures that users' emails are valid and that that email can be used for further notification of the user if necessary. To spare the users of another login, when the link on the email is pressed and the token is valid, the user is logged on directly. Tokens can be used only once, and if not used, expire after 30 days. To get a new token the user must fill the register form again. A user is not activated if he tries to validate an invalid token.

As previously said, to use the application, users must log on. This information is sent over the HTTPS protocol to enhance privacy. The process chosen for user authentication is the login/password authentication. After submitting the login form, if the credentials are valid, a user session is created. This session is an object placed in the cache to be distributed among the servers, containing some information of the user. This session is identified by the cookie value sent to the user in the response of the login request, if credentials are correct. Therefore, as long as the cookie and the session remains valid the user can access his allowed resources. The cookie is created as a browser session cookie, so its erased when the browser is closed. When the user presses the logout option, the cookie is invalidated and the respective cache is erased, so a new login process is required to access again the application resources.

E. Other implemented features

Other implemented features include some other features relevant the application, but not as important as the ones mentioned previously. One of the features concerns the generated HTML pages. Using the Django templating system, all pages share the same structure and look. The pages have five areas. The header, footer, left, center and right blocks. Header and footer usually contains institutional data. The right block usually contains advertisement, the left block contains the respective menu, and the data is displayed in the center block. However, sometimes it's necessary to display non static content in header, footer and right block. If this is the case, the application is already prepared to populate whatever necessary data to display dynamic content in those components.

Also have been used time rotating logging [8] to register possible application bugs and user behavior. For this purpose, log files contain the cookie value of a user, so one can easily track the resources requested by each user. The application was developed for possible internationalization. Therefore all displayed text blocks and messages are centralized in one single file. If necessary, that file can be translated to a new file containing texts in another language. To use this new file and to completely change the application language, it's only required a change in settings to specify the new file.

Another developed feature consist in allowing users to define the visibility of attributes of their data to be seen or not by other users. When a candidate inserts or edits its personal data or a profile, or when an employer insert or edits its data or an offer, the visibility of most of the attributes to the other users can be defined, so that the candidate or employer can hide sensible data. However this hidden data is used when a search is made. Also some attributes depend on the selected value of others. For instance, cities depend on the country. Instead of displaying a list containing all the cities, after the country is chosen this list is shortened to contain only the cities belonging to that country. This is fulfilled using an AJAX request for the retrieval of the subset of options to populate that input. Sometimes users insert values that are not valid values to fill an attribute like, for instance, the insertion of characters in a numeric field. Therefore all the values inserted by users are validated by the server using the validation features provided by Django, although the validation is only syntactical and not semantic.

V. Classification algorithm

A. Aggregation

The purpose of this application is to have searches using a classification algorithm, so that a score can be found for the resemblance between a candidate profile and a model profile or an employer offer and a model offer. This algorithm is now explained in detail.

Aggregation refers to the process of combining values (numerical or non numerical) into a single one, so that the final result reflects all individual aggregated values [9]. In decision making, aggregated values are preference or satisfaction degrees. Preference quantifies how an alternative A is preferred to an alternative B and thus is a relative appraisal. Satisfaction quantifies how an alternative is satisfactory with respect to a given criterion and thus is an absolute appraisal. Values can belong to a numerical scale or a non numerical scale {small, medium, large}. A non numerical scale is often used in fuzzy set theory. Once defined the values, they can be aggregated and a new value in the same scale is calculated. Properties of aggregation depend on the nature of the aggregated values. In multiple criteria aggregation, the purpose is to find an absolute score for an object in a set of partial scores concerning different criteria.

B. Comprehensive criterion

The comprehensive criterion will describe the operation of aggregation.

It's defined a set $X$ of alternatives $x_1, \ldots, x_n$ and a set of criteria $h_1, \ldots, h_n$. The criteria $h_i$ maps a set of
alternatives in a scale $L$. The value $h_i(x_i)$ is defined as
the partial score of the criteria $i$ for alternative $x_i$. It's
used the same scale for all the criteria and it is the interval
$[0,1]$. Consider also $a_i = h_i(x_i)$. This way, the
alternative $x_i$ can be represented as a vector of partial
scores $[a_1...a_n]$ in $[0,1]^n$.

An aggregation operator $\varphi$ is a mapping of $L^n$ to
$L$ which computes from the vector of partials scores the
total score of alternative $a$.

Weight definition allows some criteria to have more
influence than other concerning aggregation. On the other
hand, the decision maker might want one or several criteria to
suffice (favor) or one or several criteria to suffice (veto).

Most aggregation operations require weights of importance
of criteria. The used aggregation is the weighted mean
operation. A normalized weight vector $[w_1,...,w_n]$ is
defined such as:

$$w_i \in [0,1]$$

$$\sum_i (w_i) = 1$$

The weighted mean operation in the above condition is
defined as:

$$WM_{w_1,...,w_n}(a_1,...,a_n) = \sum_{i=1}^{n} w_i a_i$$

Veto effect can be defined as:
1. $\varphi$ is the final aggregation operation
2. $\phi$ is the chosen aggregation operation
3. $a_i$ is a veto of $\varphi$
4. $\land$ is the minimum operation

$$a_i$$ is a veto of $\varphi$ if $\varphi$ can be decomposed as:

$$\varphi(a_1,...,a_n) = a_i \land \phi(a_1,...,a_n)$$

(4)

Similarity, favor effect can be defined as:
1. $\varphi$ is the final aggregation operation
2. $\phi$ is the chosen aggregation operation
3. $a_i$ is a favor of $\varphi$
4. $\lor$ is the maximum operation

$a_i$ is a favor of $\varphi$ if $\varphi$ can be decomposed as:

$$\varphi(a_1,...,a_n) = a_i \lor \phi(a_1,...,a_n)$$

(5)

The final aggregation operation used can now be defined as:
1. $\varphi$ is the final aggregation operation
2. $WM_{w_1,...,w_n}(a_1,...,a_n)$ is the weighted mean
aggregation operation
3. $a_{x1},...,a_{xn}$ are vetos of $\varphi$
4. $a_{y1},...,a_{yn}$ are favors of $\varphi$
5. $a_{xi} \neq a_{yi}$
6. $\lor$ is the maximum operation
7. $\land$ is the minimum operation

and so it's obtained:

$$\varphi(a_1,...,a_n) = a_{xn} \land WM_{w_1,...,w_n}(a_1,...,a_n) \lor a_{yn}$$

(6)

The used aggregation operation was the weighted mean
with veto and favor effects defined by (6), although each
partial score $a_i$ can’t be veto and favor at the same time.

C. Search methods

In this application there have been developed two kinds of
searches: the criteria search and the classification algorithm
search. The criteria search allows the user to fill and select
each one of the criteria to perform a search based on that set of
chosen criteria. Some of the criteria require the exact value to
be matched while other criteria require only a condition to be
fulfilled. As an example, the city must have the same exact
value, but course average, when searching an offer, can be
lower or equal compared to the defined value. If the criteria
set is too loose and the search returns a list with many offers
or candidate profiles, the user is asked to strick the criteria,
although the total number of records found is shown.

The search using the classification algorithm is very
different from the above. The user inserts a model offer if he is
an employer or a model profile if he is a candidate. The detail
page of these models allow users to search using this method,
specifying weights, vetos and favors for the set of attributes.

All records are retrieved and the most resemblant offers or
candidate profiles, the user is asked to strick the criteria,
though the total number of records found is shown.

The search using the classification algorithm is very

VI. RESULTS

Beyond the unit tests that assure the reliability and
effectiveness of the application, some other tests were
performed to assure the correctness of the used classification
algorithm. For a set of three profiles and three offers the
obtained results were the following:
mechanism is still undeveloped for the sites analyzed. This growth of the response time was linear. For the test with one hundred entities, it was observed that the time for one thousand entities and about two seconds. The classification algorithm search took about twenty seconds, maintaining the same time for one hundred instances. For the searches, the criteria search remained below one second, showing full similarity.

As of model offer 3, there is a favor effect which is fulfilled, although in other criteria than the veto, other offers are not very similar to the model. Test results for model profile 1 show that the existing profiles or offers that satisfy some criteria in order to be analyzed. Therefore this article enumerates the main concerning issues towards the creation of a job site containing the major features found in the actual sites, but defining and implementing a classification algorithm that allows the computing of the similarity between a model profile or a model offer and the existing profiles or offers, displaying only the most similar, ordered by similarity. It initiated with the state of the art, followed by the technology used and some necessary theoretical concepts. Then the prototype build was described as well as the developed and implemented classification algorithm. Finally the obtained results are analyzed.

B. Future work

The developed application is intended to be a prototype and for that fact, some features could be added in the future. One of the features concerns the development of a back-office allowing some users with specific privileges to update the database. The cache could be used more often, specially to minimize the time spent in classification algorithm searches. One other feature that can be developed is the candidate picture and curriculum upload.

## Table 1: Profiles scores with Classification Algorithm search

<table>
<thead>
<tr>
<th>Model Profile</th>
<th>Model Profile</th>
<th>Model Profile</th>
</tr>
</thead>
<tbody>
<tr>
<td>Profile 1</td>
<td>0.857</td>
<td>0.591</td>
</tr>
<tr>
<td>Profile 2</td>
<td>0.581</td>
<td>0.813</td>
</tr>
<tr>
<td>Profile 3</td>
<td>0.000</td>
<td>0.630</td>
</tr>
</tbody>
</table>

## Table 2: Offer scores with Classification Algorithm search

<table>
<thead>
<tr>
<th>Model Offer</th>
<th>Model Offer</th>
<th>Model Offer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Offer 1</td>
<td>0.638</td>
<td>0.547</td>
</tr>
<tr>
<td>Offer 2</td>
<td>0.912</td>
<td>0.605</td>
</tr>
<tr>
<td>Offer 3</td>
<td>0.872</td>
<td>0.712</td>
</tr>
</tbody>
</table>

The application has been tested also with load tests to ensure that sustains high loads well. As so, the application has been tested with over one thousand entity instances in all entities and the behavior as well as response time remained the same. For the searches, the criteria search remained below one second, maintaining the same time for one hundred instances. The classification algorithm search took about twenty seconds for the test with one thousand entities and about two seconds for the test with one hundred entities. It was observed that the growth of the response time was linear.

### VII. Conclusion

#### A. Conclusions

One of the major features of job sites concerning the search mechanism is still undeveloped for the sites analyzed. This influences the analysis method for profiles and offers by the employers and candidates. This method is still individual search for profiles and offers to form a sub-set of candidate profiles or offers that satisfy some criteria in order to be analyzed. Therefore this article enumerates the main concerning issues towards the creation of a job site containing the major features found in the actual sites, but defining and implementing a classification algorithm that allows the computing of the similarity between a model profile or a model offer and the existing profiles or offers, displaying only the most similar, ordered by similarity. It initiated with the state of the art, followed by the technology used and some necessary theoretical concepts. Then the prototype build was described as well as the developed and implemented classification algorithm. Finally the obtained results are analyzed.

#### REFERENCES