NOVA SCHOOL OF SCIENCE & TECHNOLOGY

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

CATARINA MACHADO MOURA

Master in Electrical and Computer Engineering

ALGORITHM TO SUPPORT THE DECISION TO PROPOSE EXPERT PEERS FOR DOCUMENT REVIEW

MASTER IN ELECTRICAL AND COMPUTER ENGINEERING

NOVA University Lisbon
March, 2023

NOVA SCHOOL OF SCIENCE & TECHNOLOGY

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

ALGORITHM TO SUPPORT THE DECISION TO PROPOSE EXPERT PEERS FOR DOCUMENT REVIEW

CATARINA MACHADO MOURA

Master in Electrical and Computer Engineering

Advisers: Ricardo Luís Rosa Jardim Gonçalves
Full Professor, NOVA University Lisbon
José Alexandre Pires Ferreira
Senior Researcher, UNINOVA

MASTER IN ELECTRICAL AND COMPUTER ENGINEERING

NOVA University Lisbon
March, 2023

Algorithm to support the decision to propose expert peers for document review

À minha família.

Acknowledgements

I would like to thank my adviser José Ferreira for his guidance throughout the process. I would also like to recognize the help of my adviser Ricardo Gonçalves. I would like to express my gratitude to my institution, NOVA School of Science and Technology, for this five wonderful years surrounded with amazing people.

I could not have undertaken this journey without Laura and her contagious motivation.

I am very grateful to my family who were a big support for me: my mother, Isabel, who is always there with her love and care; my father, Carlos, who motivated me with his interest in my work; my sister, Mafalda, who made sure I got here; and my brother, Tomás, who only doubted me sometimes.

I am extremely grateful to all my friends for their patience and care.

Special thanks to RYSE that inspired me along the way, and to all the friends I met there, a huge thanks.

I feel privileged for having had so much support.

"Se não fora a possibilidade de sentir os estados do corpo, que estão inerentemente destinado a serem dolorosos ou aprazíveis, não haveria sofrimento ou felicidade, desejo ou misericórdia, tragédia ou glória na condição humana." (António Damásio)

Abstract

Reviewing documents is a necessary task in all professional areas and is often neglected. A very common method these days is to share documents via email, to then be forwarded to a competent person. Assigning a document to someone with the expertise to evaluate it is part of the process. To make document review an efficient mechanism, it is necessary to have a platform that facilitates communication between the candidate and the reviewer. If this mechanism is important for reviewing documents in their entirety, it is even more relevant when it comes to tender applications and their evaluation. The submission of an application must be an easy process for the candidate, without ever forgetting the reviewer's side, who must have the information to be evaluated in a clear and organized manner. Assigning a proposal to a reviewer who has the skills to evaluate it is a crucial aspect for a reliable evaluation. It is important to understand what are the common characteristics between a proposal and a reviewer, so that there is a mechanism to study their compatibility. In this dissertation, a study is made on these characteristics to make the attribution of reviewers to certain proposals automatic. An analysis is also made of the review process so that it becomes quick, accurate, and easy.

Keywords: Proposals revision, Algorithms, Data Analyses, Data Structures, Classification Methods, Portal for Proposal Evaluation, Expert Selection

Resumo

A revisão de documentos é uma tarefa necessária em todas as áreas profissionais e muitas vezes descurada. Um método muito comum nos dias de hoje é partilhar documentos através do email, para depois serem reencaminhados para uma pessoa competente. Atribuir um documento a alguém com os conhecimentos necessários para o avaliar, faz parte do processo. Para tornar a revisão de documentos um mecanismo eficiente, é necessário ter uma plataforma que facilite a comunicação entre o candidato e o revisor. Se este mecanismo é importante para a revisão de documentos na no seu todo, é ainda mais relevante quando se trata de candidaturas a concursos e avaliação das mesmas. A submissão de uma candidatura deve ser um processo fácil para o candidato, sem nunca esquecer o lado do revisor, que deve ter as informações a avaliar de forma clara e organizada. Atribuir uma proposta a um revisor que tenha competências para a avaliar, é um aspeto crucial para uma avaliação fidedigna. É importante perceber quais são as características em comum entre uma proposta e um revisor, de forma que haja um mecanismo de estudar as suas compatibilidades. Nesta dissertação, é feito um estudo sobre estas características para a tornar automática a atribuição de revisores a determinadas propostas. É também feita uma análise ao processo de revisão para que este se torne rápido, preciso e fácil.

Palavras-chave: Revisão de propostas, Algoritmo, Análise de Dados, Estruturas de Dados, Métodos de Classificação, Portal de Avaliação de Propostas, Escolha de peritos para a Revisão

Contents

List of Figures

Acronyms

ANN Artificial Neural Network 14, 15

CPV Common Procurement Vocabulary 9
CRUD Create, Read, Update and Delete 42

DT Decision Tree 15

HTML Hypertext Markup Language 8

kNN k-Nearest Neighbours 15

NB Naive Bayes 14, 15

PHP Hypertext Preprocessor 9
PIC Participant Identification Code 22

SVM Support Vector Machine 14, 15

VAT Value-Added Tax 22

1

Introduction

Technology evolution is exponentially growing and is fundamental for the economic development of a society. Every business depends on technology to succeed. On the other hand, the human being as an individual uses technology not only to succeed but also for comfort. Businesses must have tools that fulfil their needs, improving their work. This also applies to an individual, many platforms were built just to make things easier. With this dissertation, that's exactly what is expected: build a platform completely designed to facilitate a specific matter.

Nowadays, most companies have recruitment calls, funding calls or any kind of open calls. Once a candidate applies to a call, someone needs to review that application/proposal. It would be ideal for the reviewer to have some background about the topic of the proposal being evaluated. It would be even better if the reviewer had already been part of the evaluation process of similar calls. Trying to fit the right proposal to the right evaluator could take days or maybe even weeks if there were too many applications. For that reason, it would be interesting to find an easier and more computerized way to optimize the process of proposal evaluation. This is what this dissertation focuses on.

Other researchers tried a two-stage method to solve the problem of recommending experts for different candidates: "An information filtering method is first offered to identify proper experts as a candidate set. Then, an aggregation model with various constraints is suggested to recommend appropriate experts for each applicant."[1] To better understand this method, these researchers architectured the expert recommendation framework (Figure 1.1).

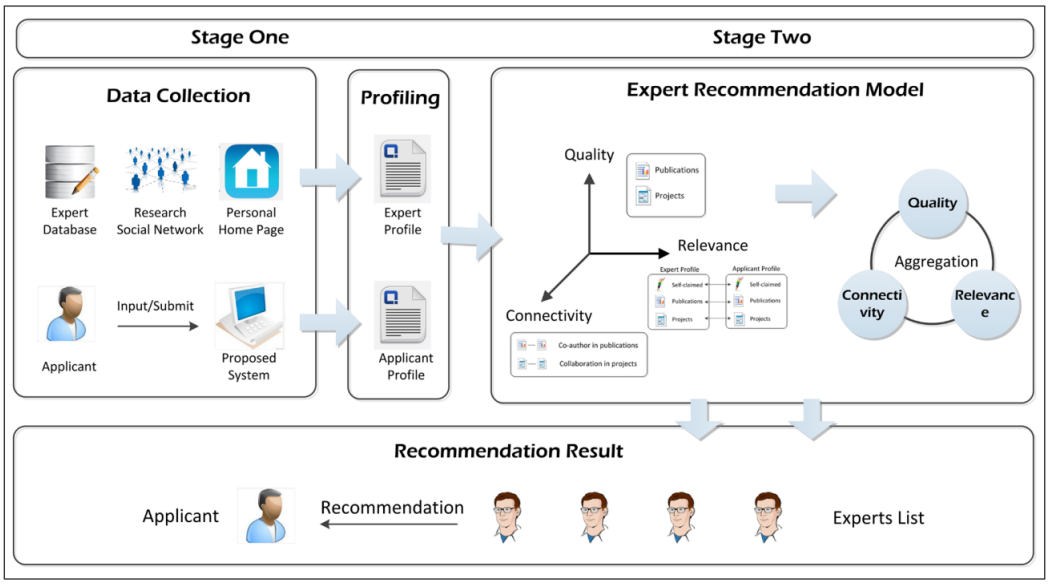

Figure 1.1: Architecture of expert recommendation framework[1]

It is possible to see the two separated stages and what is expected in them. The first stage is focused on profiling not only the candidate but also the expert. The second stage is where the aggregation is made taking into consideration[1]:

- Connectivity - for excluding conflicts of interest;
- Relevance - for calculating the similarity between expert and applicant;
- Quality - for evaluating the experts' level of expertise;

Although this is more focused on the candidate profile, it could be easily adapted to center attention on the proposal characteristics, by changing the way relevance is analyzed: pairing the expert's knowledge to the proposal's topic, instead of calculating the similarity between expert and applicant.

In order to proceed with the evaluation, the admin must choose the reviewer that should evaluate a certain proposal. He has to take into account that the reviewer must have the knowledge to review that. The main purposes for this dissertation is to help the admin rapidly match proposals to reviewers, in order to make the evaluations more reliable. This match is made by taking into consideration some criteria that will be described in the following chapters.

1.1 Motivation

Every selection process, for example for a job or project, starts with a candidate submitting an application. For many companies and entities if a candidate wants to apply for a call they need to send their application by email. This process has some problems:

- the candidate has to use two platforms - the website, where the requirements are found, and the email, for the submission;
- the reviewer gets applications in different formats and often without some necessary details;
- someone has to organize the email and put together all the applications for the same end.

There is an urgent need to digitize and simplify the process of submission and proposal revision. As said in the previous chapter, it is also important to optimize the process of proposal evaluation, which means:

- Having proposals in a single format, to be easier to compare;
- Having all the information organized in separate fields.

For this to happen, the submission platform must be a well-designed tool made specifically for this problem.

In the current market, there are a lot of platforms designed for business, but they are either paid or not that customizable.

The proposal submission and review portal for open calls is a platform for the candidate and the evaluator to communicate. It is a space with multiple open calls accessible to anyone who wants to apply. The app is also a space where it is easy to analyze and evaluate all submitted proposals.

The idea for developing this portal emerged because the DIH4CPS platform[2] needs a better way for proposal submission. It is lacking an easy and fast communication channel. These kind of platforms have a very inefficient method for document submission. If someone wants to apply for a call is redirected to another platform or has to submit it by email. As stated previously, this is a bad experience not only for the user but also for the reviewer that gets the information from all candidates in different formats. This method is complex for every single user of this platform. The idea is to use/build a tool that simplifies the proposal's submission and evaluation process and also to match reviewers to proposals, which is a solution that didn't exist in that context.

Another factor that should be taken into consideration other than the reviewer experience is the review workload, an important detail for a reviewer to decide if they will accept a review invitation. To do so, an empirical investigation was done to develop a method that automatically recommends reviewers while considering the review workload among other factors[3]. This method consists of a multi-objective meta-heuristic algorithm to search for reviewers, guided by two objectives:

- maximizing the chance of participating in a review;
- minimizing the skewness of the review workload distribution among reviewers.

This is a different point of view. If the solution built with this dissertation only took into consideration the similarities between the proposals and the reviewer or the number of proposals reviewed in the same field it could be an initial solution. This need to have in consideration that the proposals are for Open Calls and in the technological fields having impact on the reviewers selection

1.2 Work Methodology

For any work to succeed the best approach is to develop a work plan. For this dissertation, the co-adviser recommended the classical scientific method [4]. The following phases were followed to develop this dissertation:

1. Research questions/Problem - This phase refers to defining the questions or problems in hand (chapter 1.3);
2. Background/Observation - For this phase, the goal is to make observations and gather background information about the problem (chapter 2);
3. Formulate a hypothesis - From this point it is possible to make an educated guess and form a statement that proposes an answer to the problem found in the first phase (chapter 1.4);
4. Design experiment - Once in this phase, the goal is to think of a way to put to the test the hypothesis created in the previous phase, and design tests that can verify the viability of the proposal(chapter 4);
5. Test hypothesis - This phase is where everything is tested, using the tests/experiments that were created in the previous phase to put the hypothesis from the hypothesis to the test (chapter 4.1);
6. Interpret/Analyze results - Does the results from the previous phase prove or disprove the hypothesis? This is the question that should be answered in this phase. It doesn't matter if the result is positive or negative everything could be helpful for the scientific community, and it could be a starting point to reconsider the problem with a different hypothesis or a different approach (such as changing the research question) (chapter 4.2);
7. Publish findings - The last phase is all about putting all this information together and sharing it with other researchers, in this case with a dissertation.

1.3 Research Questions

To achieve some solution that responds to the problem described above, it is important to define what the real problem is. The research questions fulfil that purpose. Defining

the research questions is an important step to focus the attention on the real problem at hand.

For that reason, two questions guide the work developed along this dissertation:

RQ1: "Is there a way to optimize and digitize the process of proposal evaluation?"

RQ2: "Is it possible to create an algorithm that proposes experts to evaluate a proposal based on their areas of expertise?"

The main goal of this dissertation is to study the two problems above, search for similar solutions or work developed in this area, and design/build a solution that fits and solves the problems.

1.4 Hypothesis and approach

According to all the researched information presented in the previous chapters and the research questions from the first chapter, the following hypothesis was formulated:

"Propose experts for proposals evaluations using the meta-information from the reviewers experience and the proposal's details."

This hypothesis will be put to the test. In order to verify its viability, it will be implemented and tested.

2

State of the art

This chapter is dedicated to demonstrating what kind of work has already done by other researchers in this field and what can be used as a guideline to build the platform described previously. First, a description of the platform that can be used as a reference and then an extensive description of algorithms and data structures that can be used to improve the user's experience with the website will be made.

As stated in Chapter 1.1, it is crucial to have a well built and accurately designed portal for proposal submission. To make a better experience for the reviewer, the submission process should be well thought out. This dissertation is about building this tool and also matching the proposals to the right reviewer. Chapter 2.1 resumes the existing applications that are similar to the one that will be created, a website for proposal submissions that groups the information to be easily reviewed. Chapter 2.2 describes the problem with expert selection and presents solutions that can be used to solve the problem such as classification methods, algorithms and data structures.

2.1 Similar platforms

In the following chapters, different kinds of proposal submission websites are described. To improve the evaluation process it is necessary to build a good platform not only for the reviewers but also for the candidates.

There are many of solutions to proposal submissions and this chapter describes existing applications for this purpose. One common characteristic in these apps, which is also the problem described in this dissertation, is automation. The goal of these apps are to make all the interactions easier, faster and better when compared to a manual process. The platforms that will be described in the next chapters are:

- F6S;
- European Commission - Funding & tender opportunities;
- Getsitecontrol.

These platforms have different applications but they all serve the same purpose, as an open calls portal. After describing all of the features that every website has, a comparison will be made between them to analyse what kind of functionalities are important for the portal.

2.1.1 F6S

F6S is a platform designed for founders to find Startup programs such as hackathons, accelerators and competitions[5]. Anyone can create an account and join the platform. Once on the home page, as represented in Figure 2.1, it is possible to see the menu on the top with the functionalities that the website has: Apply, Events, Jobs and Deals.

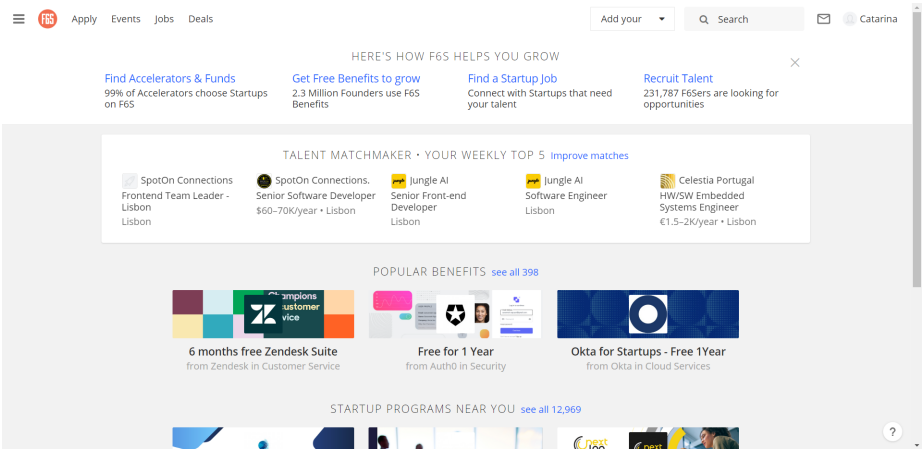

Figure 2.1: F6S - Home page[5]

This platform has implemented a good and efficient search mechanism that works with filters. For example in the Jobs section, in Figure 2.2, it is possible to look for a job taking into consideration:

- Type - Co-founder, Employment, Contract or Internship;
- Location;
- Remote/Office;
- Compensation - Where it is possible to select a minimum and maximum value for the salary and also to choose the minimum equity;
- Role - This filter exists for the candidate to choose the role that they want to apply to;
- Skills - This filter allows to find a job that is looking for skills that the candidate has selected.

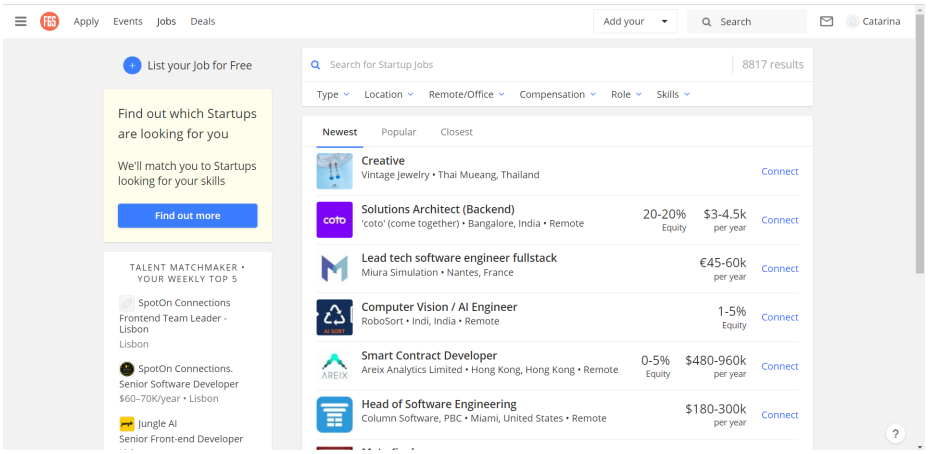

Figure 2.2: F6S - Jobs page

Filtered search is a very useful tool that makes the platform easier to work with and gives a better user experience. F6S has this search mechanism throughout their website, so it doesn't matter if the user is looking for a job or an event, they will have the same experience.

This platform has an area dedicated to the user and their profile, represented in Figure 2.3. F6S encourages the user to improve the profile strength by giving more information such as location, photo, experience, education, skills, tagline, links and more.

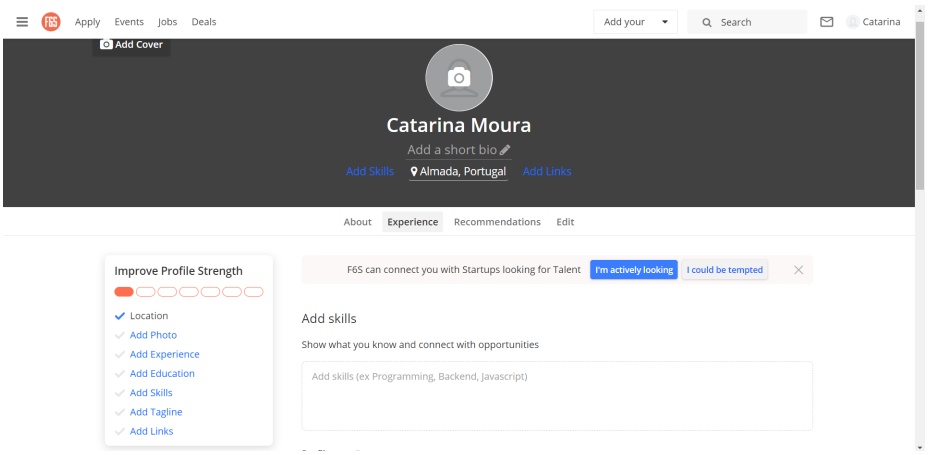

Figure 2.3: F6S - Profile page[5]

The user can apply to all job offers, events and deals, as well as create them. Once a user has signed it, they can interact with the platform as a candidate or as a publisher. It is possible to create: job listings, start-ups, accelerators, benefits/deals, events, contests and funds.

The tools used to create this website were:

- jQuery, a JavaScript library that makes the process of using Hypertext Markup Language (HTML) easier;

- the programming language Hypertext Preprocessor (PHP);
- Apache as the web server;
- CloudFare and Google Cloud Platform as system resources.

2.1.2 European Commission - Funding & tender opportunities

The European Commission – Funding & Tender Opportunities platform is built for users to find funding that suits their project. As it is possible to see in Figure 2.4, there are different filters for the user to narrow the options and find suitable funding, such as programming period, programme, call, Common Procurement Vocabulary (CPV) code, places of delivery or performance and submission status.

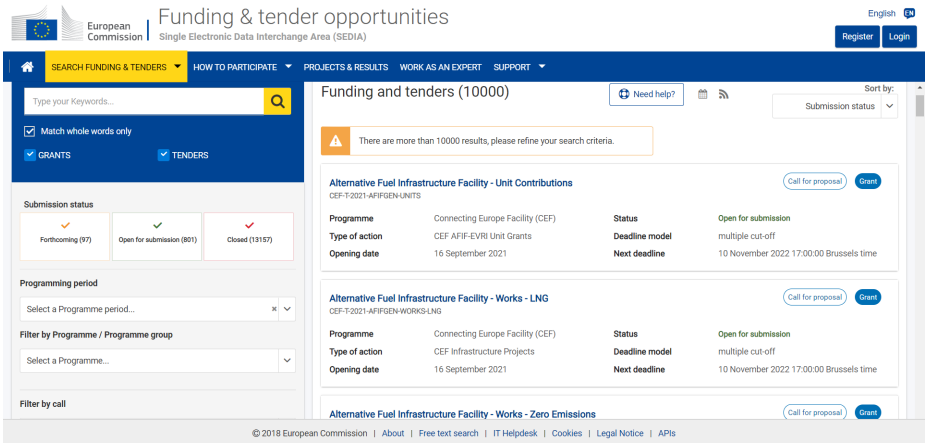

Figure 2.4: Funding & tender opportunities Portal - Calls list [6]

In each call (Figure 2.5) it is possible to check documents about it and also a brief topic description, conditions, budget overview, and general information. On the same page it is also possible to apply for the call, which redirects to a log-in page.

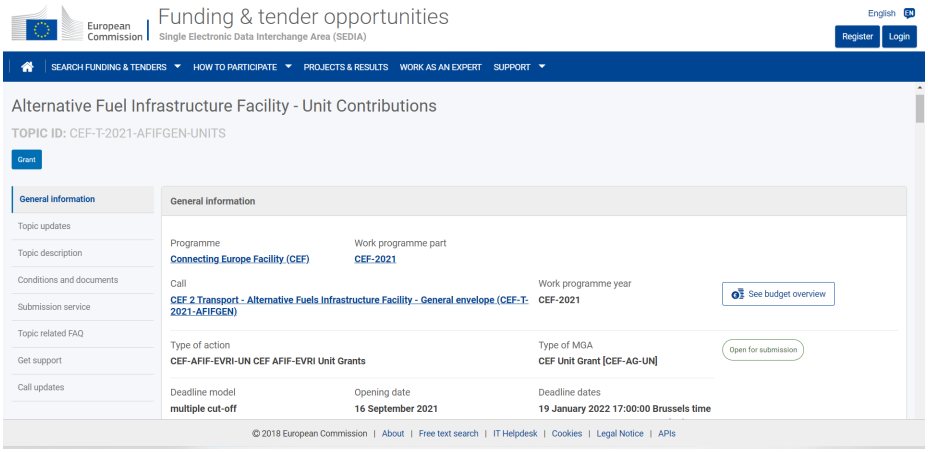

Figure 2.5: Funding & tender opportunities Portal - Call [6]

There is also a page dedicated to working as an expert (Figure 2.6). An expert should be responsible for the “evaluation of proposals, prize applications and tenders and also monitoring of actions, grant agreements, public procurement contracts”. [6]

To be an expert it is necessary to give some more personal information such as Title, First name, Last name, Date of birth, Place of birth, Country of birth, gender and nationality and besides all that is also mandatory to agree with the terms and conditions.

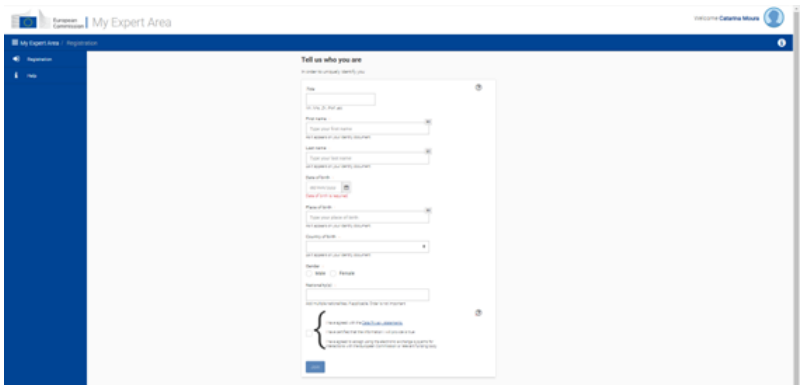

Figure 2.6: Funding & tender opportunities Portal - Expert Sign Up page [6]

The European Commission - Funding and Tender Opportunities has a dedicated page about new versions and their release notes, as shown in Figure 2.7. This page is excellent to give insight into what was lacking and how it was overcome. For example in version 6.1.0 of the Funding & Tenders Portal launched on 31/03/2022 the "My Person Profile"section was enhanced with changes to the "Documents"tab. A upload field was added for attaching recommendation letters. To avoid making the same mistakes, these and other features described on this page are useful when building similar websites.

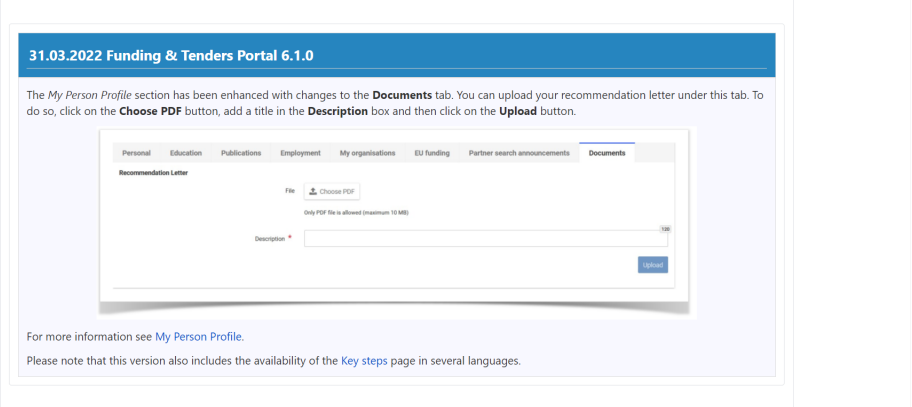

Figure 2.7: Funding & tender opportunities Portal - IT releases[7]

2.1.3 Getsitecontrol

Getsitecontrol[8] is a pop-up application that can be added to a website. It has various articles dedicated to explaining how to build a job application form [9] and that is why this application is considered a similar application. There is a list of ready made templates, or premade widgets as it is called, to create a widget. The templates are discounts, promotions and newsletter subscription forms as it is possible to see in Figure 2.8.

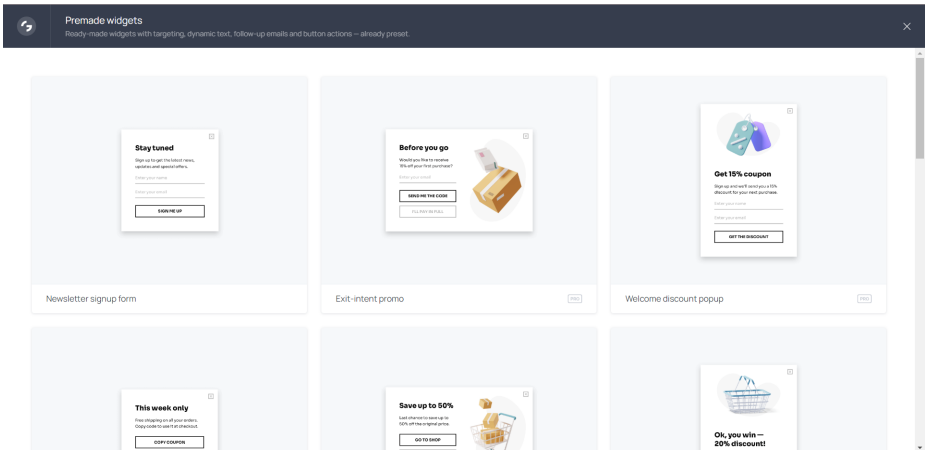

Figure 2.8: Getsitecontrol - Premade widgets

Getsitecontrol has templates dedicated to job applications, such as the one in Figure 2.9.

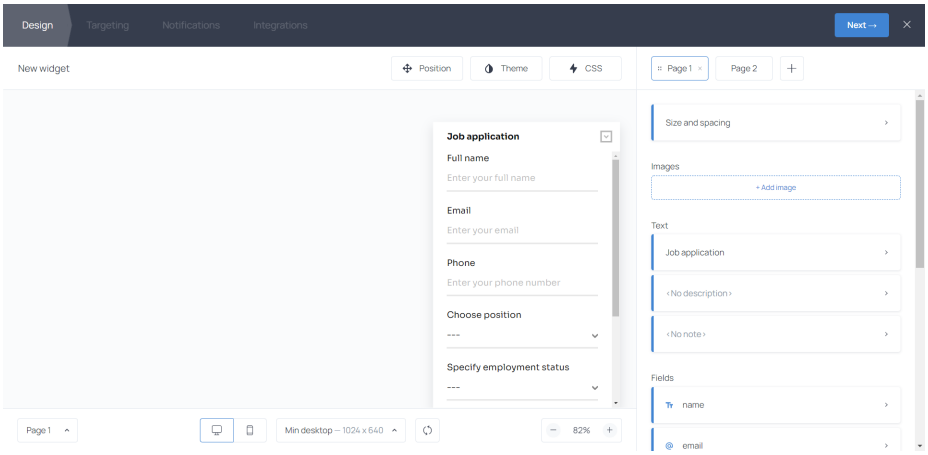

Figure 2.9: Getsitecontrol - Job Application Template

This is a really good add-on to the website but has more functionality for pop-ups. It has innumerable limitations since it is only possible to work with the pre-existing tools and it is paid software. It is also necessary to have a pre-existing website. This tool will only take care of the submission form.

Even with all these downsides, there is a good thing that this app has, its simplicity. It is easy for the user to understand how the application works.

2.1.4 Comparison

All the applications described above have interesting features that is important to take into consideration.

The way the information is organized is similar to all the platforms and that means that that must be an efficient way for the user to interact with the platform. For that reason, it is important to take those details into account when building a platform like this such as: the menu bar on the top of the website, the profile section and the sign-up method.

The European Commission website has the expert area, which is similar to the reviewer profile that is proposed in this dissertation. The expert is an external user that has the function of evaluating the proposals. The Sign Up page is really important because it is possible to see what kind of information should be taken into consideration.

There are many good characteristics to be taken into account from each platform described previously: the European Commission's and the F6S's filtered search and the simplicity of Getsitecontrol. The European Commission's website has a lot of similarities already mentioned with the portal described in this dissertation.

Altough European Commission's website appears to be the best solution between the three presented previously for the candidate and the reviewer to communicate, none of them have a feature to improve the usability of the admin.

Filtered search improves the user-experience, as described in the previous chapter, so that could be a good functionality to add to the portal. Automatically assign a proposal to an expert is a feature that improves the admin's platform experience that all the platforms described previously could benefit from.

2.2 Expert selection support system

In the following chapters, the expert selection support system will be described.

2.2.1 Context about the problem

After a candidate applies to a call, all the proposals will be reviewed. There are many ways that the proposals could be divided by the reviewers such as choosing manually or making a random distribution. There is a lot of information that the reviewer and the candidate give us that could be taken into consideration to make the review of the proposals more efficient and trustworthy. If the number of proposals reviewed in each category, the years of experience, areas of interest and other parameters are taken into account it is possible to optimize the process and assign proposals to the most appropriate reviewer.

In general, the purpose is to create an algorithm that, given certain parameters, distributes the proposals for a call to the reviewers. The administrator accepts the distribution the algorithm generated.

This proposed system could be something like Protasiewicz's solution: a user interface and three modules responsible for data transformation into information and knowledge. The three modules are[10]:

- Data acquisition module - Collects data about the reviewers;
- Information retrieval module - Profiles the reviewers using various machine learning methods for keyword extraction and information classification;
- Recommendation module - Ranks the reviewers based on similarities that they have in the reviewing object.

The next chapter explains how the processes can be divided with data analysis and some classification methods that can be used in this problem. Following, a description of algorithms (Chapter 2.2.5) will be done to explain what current selection methods exist, and how they work.

2.2.2 Data Analysis

Data analysis is a process of inspecting, cleansing, transforming, and modelling data to discover useful information, inform conclusions and support decision-making[11].

As shown in Figure 2.10, the processing of data is divided into four processes[12]:

- Data collection;
- Data preprocessing;
- Data storage;
- Data analysis.

Data collection is the foundation of the whole process, without data there is no analysis to make. Data collection is the process of information acquisition and its extraction.

Data preprocessing is crucial before starting any kind of data treatment. It is important to have the data already sorted, remove invalid data and unify the data format. This is a necessary procedure even before storing the data, it wouldn't make sense to store useless information.

Data storage is the process of saving all the information. That could be done in various ways but it is important to keep in mind that data has to be stored to be analysed and re-accessed.

Data analysis is the process of making the information valuable. There is no point in collecting, processing and storing the information if there is no use to it.

This order of process should be followed. It is impossible to do any kind of processing, storage or analysis without data. Invalid data shouldn't be stored. To interpret and analyse data it needs to be accessible at any time.

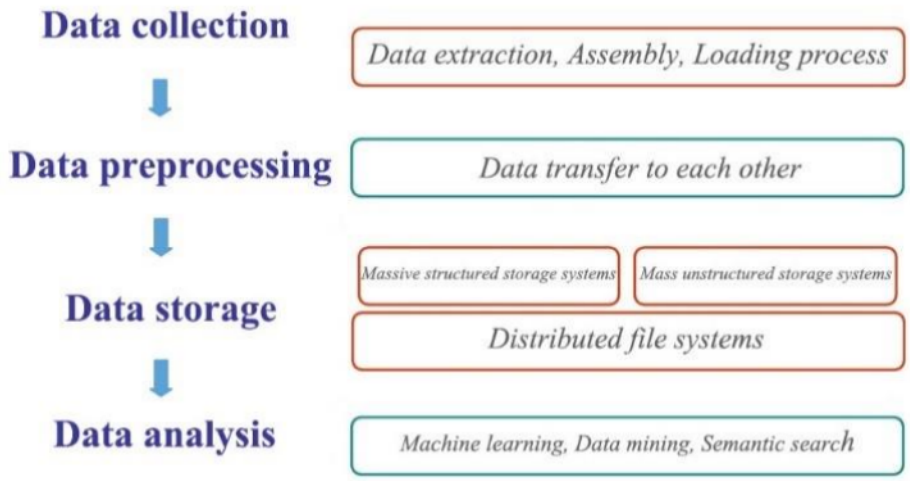

Figure 2.10: Processing Of Data[12]

In this context, it is possible to understand how these processes fit. Data collection would be all the information about the calls, the proposals, the candidates and the reviewers. After that, the information should be checked, in other words, preprocessed. Some examples of verifications are: checking the validity of email addresses for the candidate/reviewer, checking if all mandatory fields of a proposal are filled, etc... All the data collected should be properly stored, the proposals need to be reviewed, and the candidates need to know if the proposal has been accepted, the reviewers need to be allocated to the proposals for the reviewing process. The processes that need to be executed, as stated previously, are part of the data analysis necessary for this context.

2.2.3 Classification methods

Classification is the process of categorizing information based on characteristics or common features. The following list[13], represents some classification methods, used due to their usability and accuracy in categorization:

- **SVM** - Support Vector Machine (SVM) is a non-probabilistic binary linear classifier, seeks out the optimum surface to distinguish between the positive and negative data.
- **ANN** - An Artificial Neural Network (ANN) is a network of connected nodes that mimics the functionality of the brain, transmitting signals between them.
- **NB** - Naive Bayes (NB) Tree is a supervised classifier that combines a decision tree and a Bayesian rule. This algorithm determines the likelihood of each class of instances given, assuming that the properties are conditionally independent of the given label[14], using the Bayes rule, which describes the probability of an event, based on prior knowledge of conditions that might be related to the event [15]. The

Naive Bayes classifier has the benefit of only requiring a small amount of training data to estimate the means and variances of the classification-relevant variables.

- **DT** - A Decision Tree (DT) is a model or graph that resembles a tree. Given that its root is at the top and it develops downward, it is more akin to an inverted tree. Using a variety of separation criteria, the Simple Cart method creates decision trees by splitting each decision node into two distinct branches[14]. In comparison to other methods, this data representation has the advantage of being meaningful and simple to understand[13].

- **C4.5 Decision Tree Classifier** - C4.5 is a machine learning strategy that automates the induction of classification trees from training data. There are typically two phases in a decision tree training algorithm. Tree growth is the first stage, during which a tree is constructed by greedily splitting each node. The overfitted branches of the tree are deleted in the second phase. Overfitting occurs if a classifier is more complex than another that generates the same results or when the training data is limited[16]. The algorithm generates nodes by dividing the data over the attribute with the maximum information gain, and it derives a decision result from those nodes[14].

- **kNN** - Due to its ease of use and accuracy, the k-Nearest Neighbours (kNN) method is a well-known instance-based strategy that has been extensively used for text categorization[13]. In kNN classification, an object is categorized by the majority vote of its neighbors, and is then put into the class with which its k nearest neighbors are most likely to agree (k is a positive integer, typically small). The item is just put in the class of the object's one nearest neighbor if k = 1. The weight of the particular shared category is used to calculate the sum of the similarity scores of the neighbors if more than one of the k-nearest neighbors shares a certain category[17].

2.2.4 Classification methods comparison

After Behrooz Noori studied all six machine learning methods described previously, he came to 4 conclusions[13]:

- DT and C4.5 have the highest accuracy of review classification;

- DT and C4.5 performed the best with a large feature set, then SVM, and ANN was the worst;

- NB performed the best with fewer features;

- The accuracy of the model of almost all methods increased with increasing features except ANN.

This comparison will be extremely useful to choose the right method for the problem of this dissertation. Every situation is different and it is important to understand what can fit best in each case. Nevertheless, it is important to have these results in mind because Noori specifies what fits best for different problems. The next chapter is dedicated to general algorithms. Building an algorithm is something that gets better with time and training and for that reason, it is important to study some algorithms, starting with the basics.

2.2.5 Algorithms

What is an algorithm? An algorithm is a tool that, if well built, can simplify or solve a problem. It is any well-defined computational procedure that takes values, as input and produces values, as output [18]. In other words, an algorithm is a set of steps that, given some information, produces a solution. If there is a well-specified problem, such as the one described in the next chapter, it is possible to build an algorithm that makes solving it an easier task. The next chapter is dedicated to describing the problem that can be solved with an algorithm, understanding how building an algorithm should be done and the existing types of algorithms that could solve the problem.

2.2.5.1 Algorithmic thinking

By definition, an algorithm is "a method to solve a problem that consists of exactly defined instructions"[19]. To be able to build and design a new algorithm it is important to develop a new way of thinking. Algorithmic thinking is "a pool of abilities that are connected to constructing and understanding algorithms"[19]. This thinking method is characterized by a different set of abilities, such as:

- Analyze the problem;
- Specify the problem precisely;
- Search for basic actions that fit into the problem;
- Understand what the algorithm is and how it works;
- Build a correct algorithm to fits the problem using basic actions;
- Think about all the restrictions, special and normal cases of a problem;
- Improve the efficiency of an algorithm.

This ability and way of thinking is something that normally develops with training and solving many problems. It is also important to be familiarized with the programming language chosen in a way that builds a more intuitive solution.

The efficiency of an algorithm is measured in time and space [20] and understanding the way the efficiency is measured is a difficult task. The big O notation is a concept that

is present when referring to this topic. In Figure 2.11 is a graphic that represents the time spent in the function of the time that is being processed. This is where the big O notion enters, to understand how efficiency varies depending on the algorithm in use.

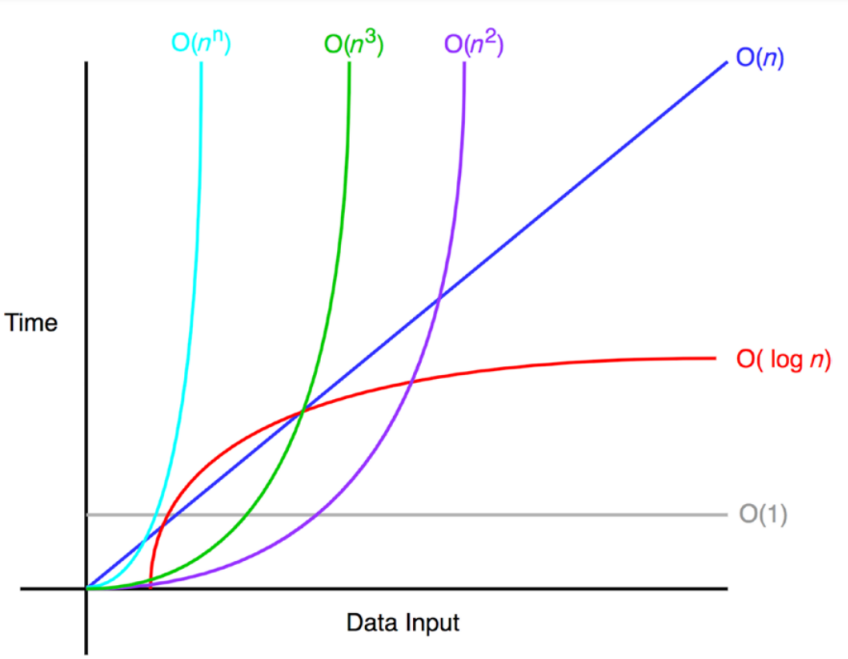

Figure 2.11: Time Complexity [21]

2.2.5.2 Types of algorithms

it is important to understand the types of algorithms that already exist and have been used and tested in very different problems. There are a lot of different ways of building an algorithm and every single one of them has its advantages and disadvantages. The important thing is to understand what fits best for the problem at hand.

2.2.5.3 Divide and Conquer

The name refers to the process that is used in this algorithm. The idea is to recursively divide the problem into 2 or more sub-problems and repeat until these sub-problems are small enough to be solved trivially. The main goal of this process is to transform a complex problem into a problem so basic that the solution can be found in very few steps. Base cases are these basic problems that need to be solved that result from the division of the original problem. Bases cases can be solved using whatever solutions suits best and once all these solutions are combined the overall solution to the main problem is found.

This technique is a powerful tool that can be used to solve complicated problems very easily since all it requires is to break up the larger problem into small simple pieces, solve those pieces trivially and then combine these solutions.

Merge sort is an optimal sorting algorithm that divides an array recursively to obtain an array so small that is already sorted (an array of 1 or 0 elements) and then merges these smaller arrays into a larger sorted array.

Real-world problems won't just wait for the problem to become so small and trivial as the example explained above. There is a hybrid solution using the divide and conquer method where the problem is divided into smaller problems but these base cases can't be solved in one or two steps. The base cases are still complex problems so a different algorithm is used to solve them. Sometimes using purely the divide and conquer method results in unnecessary recursive calls and could be more efficient to leave a larger base case and use a different algorithm to solve it. Reducing the number of these base cases and increasing the efficiency with which they are solved, dramatically improves performance. As said previously, everything depends on the context of the problem. This algorithm is very well suited for parallel processing because the base cases can be run on different processors simultaneously.

There is also another problem with using the divide and conquer algorithm: it could be wasting calls recalculating the same numbers over and over again.

2.2.5.4 Binary search

Binary search is a method to find if an element is present in an array or not. It works in a way that splits the array into two halves and searches if the element is located in the upper half of the lower half and then splits it again until the answer is found. Binary search is more efficient than linear search for larger data. Linear search will analyse element by element sequentially for the entire array. If the element that is being targeted is on the end of the list then it will take a long time to get to the result whereas binary search only "needs to examine just a few array entries (relative to the size of the array) to find the key (or determine that it is not there)."[22] It is also possible to compare a linear search with a binary search with time complexity. In the figure 2.11, it is possible to see that the performance of a linear search, represented with $0(n)$, has a not-so-good result for bigger data structures when compared with the binary search, represented with $0(log(n))$.

2.2.5.5 Fibonacci search

The Fibonacci search technique is a method of searching a sorted array using a divide and conquer algorithm, explained previously, that narrows down possible locations with the aid of Fibonacci numbers. [23]

2.2.5.6 Data-structures

A search algorithm can work faster and more efficiently if there is a data structure design, especially for that purpose. If all the information is well organized and easily accessible, the algorithm can be simpler than the construct. There are many different ways of storing

and organizing data and for that reason it is important to learn about the ones that fit this problem.

Search trees Before understanding the concept of a search tree it is important to know exactly what a tree is. A tree represents hierarchical relationships between items [24] just like in Figure 2.12. In practical terms, it starts with the root node, which has child nodes and those child nodes have their child nodes.

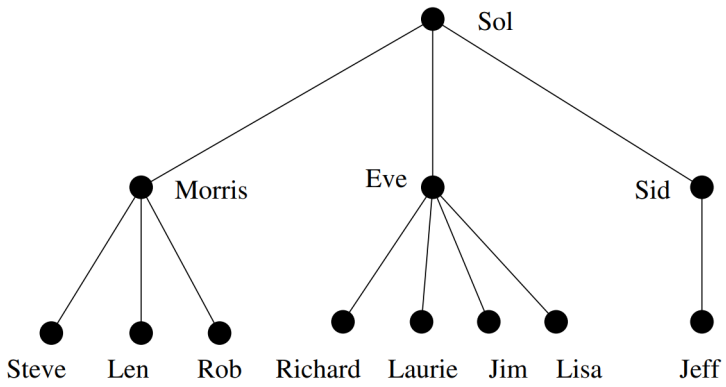

Figure 2.12: Tree data structure[24]

A binary tree works in the same way but one node can only have up to two child nodes. A binary search tree is a binary tree that fulfils a specific ordering property, every left node is always a smaller number than the root node and the right node is a greater value. For that reason it is easy to find a number on this kind of tree in a few steps. For example, a 4 level binary tree can have a maximum of 15 elements. In the worst-case scenario, to find if a certain value is in the tree, only 4 operations are needed. For a 31 element binary tree, a maximum of 5 operations and so on.

Symbol tables A symbol table is a data structure for key-value pairs[22], in other words, it is a symbol associated with some information related to that object. This kind of structure supports two operations: "insert (put) a new pair into the table and search for (get) the value associated with a given key."[22]

In Figure 2.13 the applications for the symbol table are represented, as well as the purpose of search, key and value. This Figure makes a clear definition of examples for the symbol table and how it can be used.

2.2.6 Algorithms comparison

Choosing the right algorithm that fits the problem described in Chapter 2.2.1 will be a trial-and-error task. Trying to understand the problem and how it is possible to organize the information will also be an important process before starting to code.

application	purpose of search	key	value
dictionary	find definition	word	definition
book index	find relevant pages	term	list of page numbers
file share	find song to download	name of song	computer ID
account management	process transactions	account number	transaction details
web search	find relevant web pages	keyword	list of page names
compiler	find type and value	variable name	type and value

Figure 2.13: Symbol Table Applications[22]

Taking into consideration what kind of approaches would best fit the problem alone is not enough. It is important to analyze the algorithms' efficiency. In Figure 2.14 a table is shown where the algorithms' efficiency are compared using the big 0 notation.

Idea	Data structure	Worst Case		Average Case		Memory
		Search	Insert	Search	Insert	(Bytes)
Sequential search	Unordered list	O(n)	O(n)	O(n)	O(n)	O(n)
Binary search	Ordered array	O(lg(n))	O(n)	O(lg(n)	O(n)	O(n)
Binary tree search	Binary tree search	O(n)	O(n)	O(lg(n))	O(lg(n))	O(n)
Balanced binary search tree	2-3 tree search	O(lg(n))	O(lg(n))	O(lg(n))	O(lg(n))	O(n)
	Red-black BST	O(lg(n))	O(lg(n))	O(lg(n))	O(lg(n))	O(n)
Hashing	Separate chaining* (array of lists)	O(1)	O(1)	O(1)	O(1)	O(n+m)
	Linear probing* (parallel arrays)	O(lg(n))	O(lg(n))	O(1)	O(1)	O(n)

- with uniform and independent hash function
- n : Number of the inserted elements
- m : Size of Array of the list (The range of the hashing function results)

Figure 2.14: Algorithms' efficiency comparison[25]

Some of the algorithms explained previously are not comparable because they have different levels of complexity. Some are so rudimentary that they will not solve the problem discussed in this dissertation. Nevertheless, learning about these methods is important to develop algorithmic thinking (Chapter 2.2.5.1). Search trees are a good example of a useful algorithm for this topic.

This kind of algorithms weren't used to build the solution described in this dissertation but, as explained in chapter2.2.5.1, it was really important to approach the problem with a different perspective.

3

Concept and architecture

This chapter describes the algorithms' parameters and its implementation. In order to select the experts for proposals evaluation, certain parameters have to be taken in consideration and be somewhat transformed in numbers to distinguish reviewers expertise and that is what chapters 3.1 and 3.2 are about, respectively. The evaluations system (chapter 3.3) will be described by two different perspectives: admin's and reviewer's. In the end of this chapter, will be described the technological specifications that this work is based on (3.4).

3.1 Algorithm parameters

To make the decision for the best reviewer for a certain proposal there are some parameters that should be taken into consideration based on Liu's approach[1].

- Conflict of interest;
- Recent experience in an area of interest;
- Years of experience in an area of interest;
- Recent experience in a sector;
- Years of experience in a sector;
- Reviewer's work overload;
- Gender.

In the following chapters, all these parameters described above will be explained in terms of concept and how they should be implemented. The conflict of interest parameter and the work overload are not a part of the system to get the best reviewers: the conflict of interest should be an exclusion criteria and the reviewer's overload should just be a warning for the admin to take into consideration the amount of proposals that the reviewer has to evaluate.

3.1.1 Conflict of interest

The first criteria that should be taken into account is the existence of a conflict of interest. This can be measured by the company that the users are currently working at. When a user registers, it should be mandatory to insert the company. Every proposal is associated to a user (candidate), which means that is also associated with a company. A reviewer is also a user, so when they sign in the platform, it will also be mandatory to insert the company. There is an important thing to assure here, the name of the company is something that the user can choose from a list, or create by inserting the name manually. This can lead to users working on the same company in reality but not the platform database, for example: Jon works on Sky, and he introduces the name of the company as "Sky". Mark, that is working at Sky too, is also registering on the platform, and he ruffly searches on the companies list but doesn't find the company name on the list. So he adds a new one, and he writes "SKY". This means that they are registered in different companies on the platform, if Mark upgrades his profile to be a reviewer he shouldn't review any work from Jon because they are both working on Sky, but the platform will wrongly allow that.

When a user is registering in the platform, it should be given a list of all the companies that already exist in the platform. It would be very tedious for the user if they had to leave the users' *registration* page, enter the *add a company* page, and add a new company. It should be possible for the user to create a new company in the *registration page*. This functionality is very important for the user comfortably using the portal. It would also be interesting for the user to be able to search companies by name/Value-Added Tax (VAT)/Participant Identification Code (PIC) because if there are a lot of companies registered in the platform the list could be very long. Another way to make the platform more user-friendly in this aspect is to sort the companies' list alphabetically.

3.1.2 Users' experience and proposal's areas of interest and sectors

In order to be assign a proposal to a reviewer, it's necessary to verify if the reviewer has knowledge in the area of the proposal. When a user submits a proposal, they can choose the areas of interest that fit their project, as well as a sector. Areas of interest is about the technology used in that project, for example sensors and actuators, micro and nano electronics, artificial intelligence etc. Sectors is the industry where that area of interest is applied, for example agriculture, manufacture, tourism, energy, etc. To check if a reviewer is able to evaluate a proposal, it is necessary to know their experience in both fields. It's possible to obtain that information by two different ways:

- Platform experience - A record of evaluations that a reviewer did in a certain area or sector;
- Previous experience - A record of projects they worked on.

If it was only taken into consideration the experience that a reviewer had on the platform, in other word, the areas or sector of the proposals that they had evaluated, new reviewers wouldn't have a chance to be chosen by the algorithm to review a proposal. To get the *Platform experience* is necessary to keep the register of every evaluation that the reviewer did. To get the reviewer's *Previous experience*, the platform should ask the user their experience when they are trying to upgrade their profile to become a reviewer,this allows new reviewers to be eligible to evaluate a proposal. It should be also possible for the reviewer to update their experience and add new ones. The platform could also suggest the reviewer to update their profile when they are six months without adding new experience, for example. Another interesting feature is that the user that is submitting a proposal could add multiple areas of interest and sectors to their candidature. And this could work the same way for the reviewer when adding their experience. It would be very unpleasant for the user to insert the same experience over and over just to change the areas of interest.

3.1.2.1 Recent experience

In technologies the information is constantly changing and for that reason a reviewer with experience in a certain area of interest could not be the best choice to evaluate a proposal in the same area if their experience was too long ago. For that reason, it is very important to know every reviewers' latest work. So when a reviewer is adding their experience, there should be a field for them to introduce the date of that work.

3.1.2.2 Years of experience

If a reviewer has worked their entire life on a certain area of interest, they for sure are an expert in that field. For that reason, when a reviewer is adding their experience, they should be able to introduce the staring date of that project as well as the ending date.

3.1.3 Reviewer's work overload

If a reviewer is currently evaluating a lot of proposals, the platform should alert the admin that this reviewer couldn't be the best choice. Even if the algorithm recommends that user because it has the best compatibility with the proposal, it should come with a warning signal. The limit of proposals per reviewer should be defined by the admin. The reviewer should be able to refuse the revision of a proposal, so that they can have a say in the amount of work assigned.

3.1.4 Gender

There aren't many women working on the technology flied, and one way of getting their participation is to give them some advantage. It's also important to evaluate their experience and compatibility with the proposal, but being a woman increases the change of

being selected by the algorithm.

In figure 3.1 is presented a flowchart that resumes the parameters described above that will be used in the algorithm.

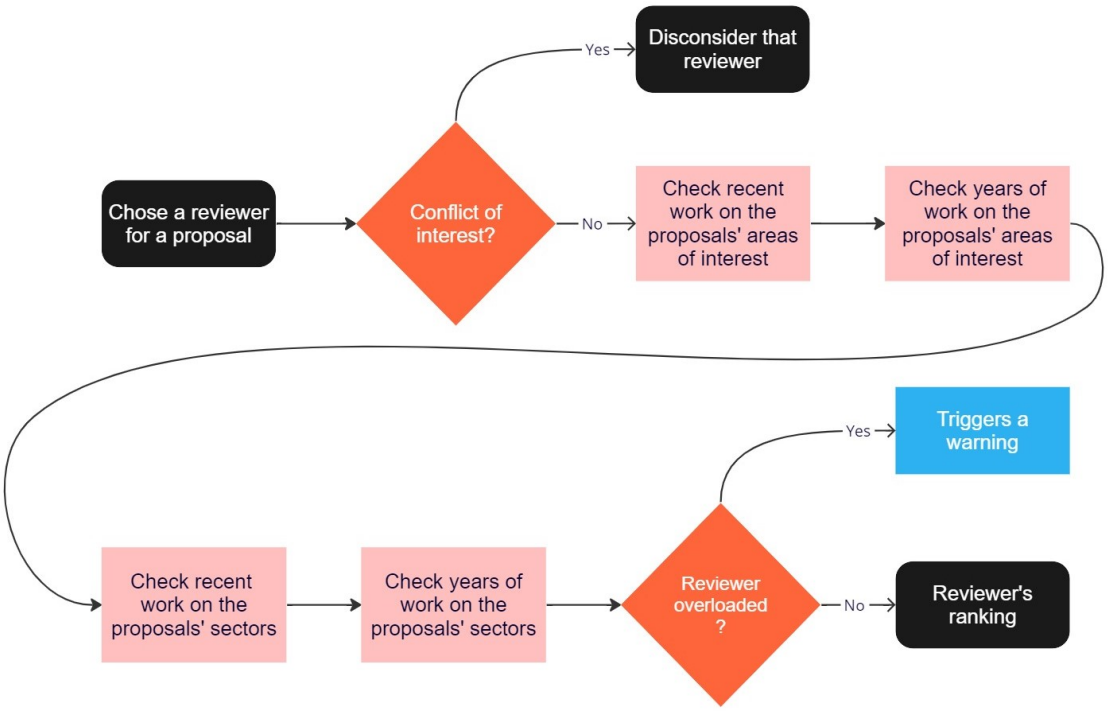

Figure 3.1: Algorithms' flowchart

3.2 Algorithm's structure

As it is explained in chapter 3.1.2, user's experience is divided into two parts (platform experience and previous experience). To measure the expertise of a reviewer in a certain area, it is important to have a profile of both types of experience. The method to calculate the level of experience is similar in both types, they will be explained separately, and then it will be explained how these two are combined.

As said previously, there are some exclusion criteria that will be checked before the algorithm's action. The list presented next is an extended list of every parameter that enters the calculation of the reviewer's expertise:

- Reviewer's previous experience:
  - Number of years passed from last experience on a certain area of interest and sector;
  - Years of experience on a certain area of interest and sector.
- Reviewer's platform experience:

  - Number of years passed from last proposal reviewed on a certain area of interest and sector;
  - Number of proposals reviewed on a certain area of interest and sector.

It is also important to have in mind that it is only being taken into consideration one area of interest in this explanation, but the idea is that the method is applied to all areas of interest and sectors.

3.2.1 Algorithm's previous versions

While building the logic for this algorithm, three different methods occur:

- Ranking;
- Scale;
- Percentage.

The Ranking method consists in evaluating the list of all the reviewers that exist for a certain area of interest and order them by the most to the least qualified. The reviewers are ranked by the ones that have more recent experience as well as the ones who have more years of experience, then these two ranks are summed, and that results on the final rank. This is an operation that gives the best reviewers available, based on comparisons with other reviewers, and for this reason this method is considered fair. It's also important to understand that this method results in multiple operations and comparisons, which leads to a more complex algorithm.

The scale method is based on a fixed table that gives certain points for certain ranges. For example, reviewers with less than three years of experience get one point, between three and five they get two points, and more than five years get three points. A disadvantage of this method is that it could lead to reviewers getting the same point just because they are on the same range, but there are some more qualified than others. As the Ranking method, this method is also based on comparison and could lead to an operation that costs time and memory with a more extend database, and for this reason tends to be a complex method. An advantage of this method is that it is possible to check if the platform is lacking some specific kind of knowledge. With this method, it is easy to check if the reviewer was chosen because there was no one better in that field, or if they are actually experts. Another advantage is that all the values (the ranges and the points attributed) could be saved and alterable by the admin at any time, which can allows the admin to perfect the algorithm according to his or hers preference.

The Percentage method is a method where all the parameters have a certain percentage, for example the *years of experience* weights 20% for the sector and 40% for the areas of interest. This method involves simpler operations when comparing to the previous ones. As the scale method, the values can be also changed by the admin, allowing improvements. A disadvantage is that the number that results from this operation has no

meaning. In the Ranking method the final value meant the position that a reviewer is when comparing to the others, the scale method resulted from a sum and there was a maximum and a minimum value that could come from that operation, but in the percentage method that doesn't apply. In a situation where the weight given to the *years of experience* and the *last experience* is the same, the experience calculation could be a negative result, for example: the *last experience* is the number of years passed from the more recent experience, which means that the higher the number the least qualified they are, so it wouldn't multiply by 50% but instead by -50%. To solve this problem the *years of experience* were multiply by 50%, but the *last experience* was multiply by 1/50%, and this eliminated the negative results, but the result was still meaningless. It is interesting to have a result that the admin can now have the perception if the reviewer is qualified or not. To put the final result in a percentage, it is necessary to make to some comparisons in order to find the maximum and the minimum of a list. This makes the algorithm a little more complex, but not as complex as the ranking method that had to order a list, this is just finding two numbers on a list. So the equations used to get the score for a certain area of interest were the ones presented next (equations3.1 and 3.2).

$$\text{Years of experience score} = Years - \frac{MIN(Years)}{(MAX(Years) - MIN(Years)} \tag{3.1}$$

$$\text{Last experience score} = Last - \frac{MIN(Last)}{(MAX(Last) - MIN(Last)} \tag{3.2}$$

To get the global score from the reviewer's previous experience is necessary to check every area of interest and sectors of a proposal and apply this method.

To get the reviewer's platform experience the method is the same but instead of getting the years of experience, it will be taken into consideration the numbers of proposals that that reviewer had already evaluated. In the end everything will be added up, and it is also possible to give more importance to the platform experience than the previous experience. For that, the admin only has to change the percentages given to each one. A big disadvantage of this method is that the percentage is based on the others reviewers' knowledge, this means that a reviewer with 100% could be a reviewer with a really poor experience in a certain area but is considered an expert.

In figure 3.2, is presented a summarized comparison between all algorithms described above.

3.2.2 Algorithm based on experiences value

This method is based on the idea that all experiences have some kind of value, and that experience loses value as it gets older. For this algorithm, the platform and previous experiences will be part of the calculations, but they are now referred to as internal and external experiences, respectively. In the equation 3.3 and 3.4 is possible to see two parts: calculation of the duration of the experience in years (only applies for external experience,

Method	Description	Advantages	Disadvantages
Rank	Ranking every reviewer in all different criteria to get an order list	Fair	Complex operations - problem for bigger data structures Depends on other reviewers knowledge
Scale	Fixed table that gives certain points for certain ranges	Adjustable Easy to find the areas with less experts	Reviewers could have the same points as less experienced ones, if the ranges are to wide.
Percentage	Convert to percentage based on other reviewers (max and min)	Adjustable	Depends on other reviewers knowledge

Figure 3.2: Algorithms previous versions comparison

as shown in equation 3.4) and deduction of the depreciation associated with the years passed. The depreciation relies on the fact that for every year that has passed from the end of a certain experience, it deducts a certain percentage. That percentage is called Age Depreciation and is a value from 0 to 1 where 0 means that the experiences don't lose value with the years and 1 means that the experience doesn't have value at all. If the Age Depreciation takes a value of 0.1 that means that the experiences loses 10% of its value every year. This method for calculating the value of an experience works the same way for the internal experience, but it doesn't multiply by the duration of the experience because and evaluation as a specific date, and it doesn't extend in time. For this reason, the maximum internal score is one (equivalent to one proposal) and the maximum external score is equal to the duration of that experience. This algorithm was implemented in MySQL using views to get the data from the tables. During this chapter all the equations will be presented as well as the name of the view used for every step of the calculation for the final score.

$$IntScore = (1 - AgeDepreciation)^{\frac{CurrentDate - EvaluationDate}{365,25}} \tag{3.3}$$

$$ExtScore = \frac{(EndingDate - StartingDate)}{365,25} \times (1 - AgeDepreciation)^{\frac{CurrentDate - EndingDate}{365,25}} \tag{3.4}$$

In the views *aoi-user_aoi-score_int* and *aoi-user_aoi-score_ext* the previous equation are used to calculate the score. The internal score is based on the proposals that a certain user evaluates, so it is important to make sure that the proposal is already reviewed and approved by the admin before adding that to the user's experience.

Internal and external score are presented in different units: the internal experience score is a number of proposals, and the external experience score is measured in years. For this reason and to make the values more understandable to the admin, the scores were converted to percentage. For the calculation of the internal score percentage, the score was divided by *Max Proposals* 3.5. *Max Proposals* is the number of proposals evaluated that the administrator of the platform considers to be the best value. For example, if the

admin sets *Max Proposals* to 10, that means that a reviewer with 10 evaluations is a great reviewer. Of course, that if those 10 evaluations were old, they wouldn't be worth 10 (because of the Age Depreciation). For the calculation of the external score percentage instead of the *Max Proposals*, the score is dived by *Max Years Experience*, which is the number of years that are considered to be an excellent experience duration 3.6.

$$InternalScorePercentage = \frac{IntScore}{MaxProposals} \times 100 \tag{3.5}$$

$$ExternalScorePercentage = \frac{ExtScore}{MaxYearsExperience} \times 100 \tag{3.6}$$

The previous calculations have to be made to each experience individually and the next equations represent the sum of all the internal experiences (equation 3.8) and the sum of all external experiences (equation 3.9). The views *aoi-user_aoi-score_aggregate_int* and *aoi-user_aoi-score_aggregate_ext* use those equations and also use the total score equation 3.7, to get the real score instead of the percentage score. This equation is used to get the total of the internal and external score, so it is a summation of all the results of the equation 3.3 for the total internal score and a summation of the results of the equation 3.4 for the total external score.

$$TotalScore = \sum Score \tag{3.7}$$

$$TotalInternalScorePercentage = \sum InternalScorePercentage \tag{3.8}$$

$$TotalExternalScorePercentage = \sum ExternalScorePercentage \tag{3.9}$$

The next step is to get the global score, both internal and external. For that, three views were used: *aoi-int_ext-left*, *aoi-int_ext-right* and *aoi-total_left_right*. These views are responsible to join the results and to give the experience of every user for all the areas of interest. The equation 3.10 was used to calculate the average percentage of the internal and external score.

$$TotalScorePercentage = \frac{TotalInternalScorePercentage + TotalExternalScorePercentage}{2} \tag{3.10}$$

The goal of this algorithm is to choose the best reviewers for a certain proposal, and for that reason is important to check the reviewers' expertise in all the areas of interest that a certain proposal has. The view *aoi-proposal_aoi-total_left_right* joins the result obtain previously with the proposals. The idea is to always get a result in percentage, and if the experiences involved in a proposal were just added to each other, the result could be greater than 100%. To do the average between all the areas is necessary to know the number of areas of interest involved in a proposal and this is given by *aoi-proposal_aoi-count_aoi* view. Finally, *aoi-alg_final* view gives the reviewer's score for a specific proposal using equation 3.11.

$$TotalAOIProposalPercentage = \frac{\sum TotalScorePercentage}{NumberAOIsProposal} \tag{3.11}$$

This algorithm, although it might have complex operations, has multiple advantages: the percentage has a meaning, is adjustable by the admin and is possible to check what areas of interest need more experts. The complex operations involved here are not as complex as the ones involved in the ranking method, or even in the scale method: this algorithm uses addition, subtraction, multiplication, and division operations, instead of endless comparisons.

3.2.3 Data used in the algorithm

Two MySQL tables were the major center of information for the algorithm to work on, demonstrated in figure 3.3. *Proposal_aoi* and *User_aoi* are the tables where the views explained in the previous chapter get most of the information. *Proposal_aoi* is responsible for the internal experience. One proposal can have many areas of interest, and this tables groups that data. Of course that other tables need to be joined with this one, for example is necessary to see the state of the evaluation of that proposal to see if it is finished and also to get the date when it was made. *Users_aoi* is responsible for the external experience. When a user decides to upgrade to be a reviewer, they have the possibility to had experience in multiple areas of interest. This table also registers when the experience started and ended. It is also possible to say that the experience hasn't finished yet.

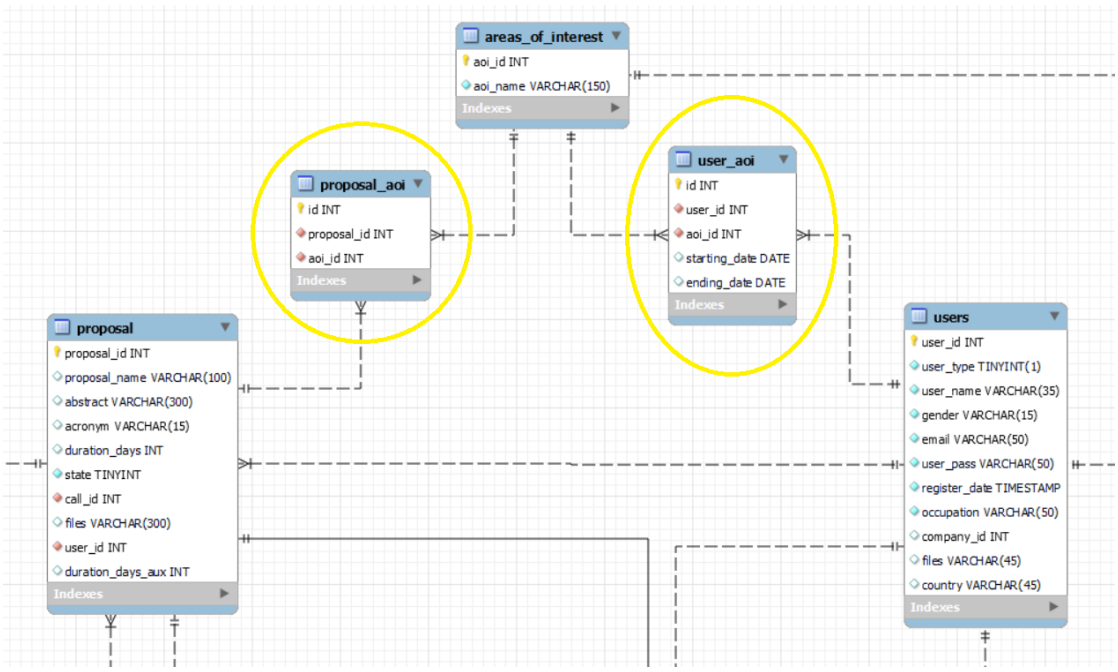

Figure 3.3: Algorithms center of information – MySQL schema

Another important center of information is the table with constants used in the calculations demonstrated previously, presented in figure 3.4. This is the table where the admin can adjust the weights of every variable. For example, if the admin wants to have more weight in the internal experience then in the external, we must reduce *max_evaluations*, which is the number of proposals that correspond to 100%. If, for instance, the admin wants to increase the devaluation in older experiences, then the *age_depreciation* factor (internal or external) must be increased as well.

idscore	name	weight	description
1	age_depreciation_internal	5	Age correction factor for evaluation. Examples: "0" equals no deduction, "10" equals 10% deduction for a one year old evaluation.
2	age_depreciation_external	10	Age correction factor for experience. Examples: "0" equals no deduction, "10" equals 10% deduction for a one year old period.
3	max_years_experience	4	Maximum number of years of experience to score 100%
4	max_evalutions	3	Maximum number of evaluations performed to score 100%
5	sector_weight	40	Percentage for sector to the total score
6	aoi_weight	60	Percentage for areas of interest to the total score

Figure 3.4: MySQL table with auxiliary constants to calculate the score

3.3 Evaluation system

The proposals will be evaluated by different reviewers and for that reason there needs to be some kind of guidance throughout the revision. The person that created the call (admin) is the best one to decide which parameters should be evaluated in a proposal for that call. The admin could also decide if there are parameters more important than others. In figure 3.5, are represented the tables where the information about evaluations are stored. On the bottom of the figure are the tables *evaluation_form* and *question_form*, the data stored here is introduced by the admin. The tables *evaluation* and *question*, on the top, are tables where the actual evaluation is stored. The relation between these tables is one to many, here one evaluation (either the form or the evaluation itself) can have multiples questions associated.

On the platform already built by Laura Martins [26], it was necessary to add a some features to facilitate the reviewer usage of the platform.

3.3.1 Admin's process

When creating a call, the admin needs to create an evaluation for as well, as said previously. It is possible to save the evaluation form as a draft (state 1) but when the call is published, the evaluation form has to be finished (state 2). Then the call is open for proposal submission and when that period ends the admin has to select the reviewers responsible for the evaluation of every proposal submitted. When the admin accepts the reviewers proposed by the algorithm, an evaluation for that proposal is generated for all the users selected. When the evaluation reaches state 2, where it is completed, the admin has to approve the evaluation. If approved, the evaluation was a success and can be used by the algorithm as internal experience. If there is some mistake and the evaluation is not

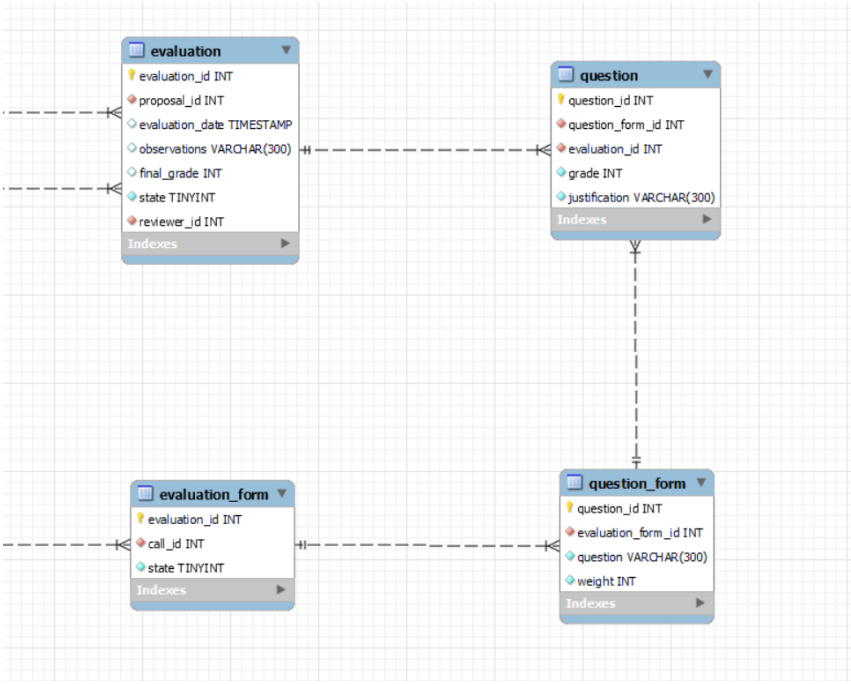

Figure 3.5: Evaluation system – MySQL schema

validated is moved back to the draft state (state 1). This process is represented in figure 3.6 where all the steps explained here are summarized.

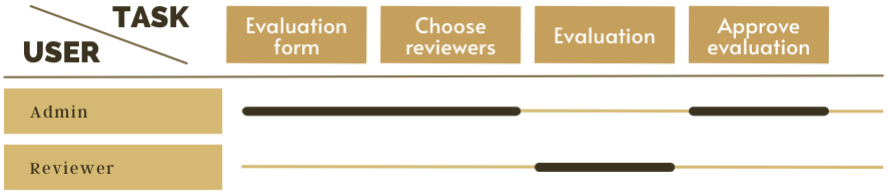

Figure 3.6: Evaluation process

In the next chapters, the menus used by the admin for all the different phases are shown.

3.3.1.1 Menu with proposals that need a reviewer

The admin need to know the proposals that are in the state where they are waiting for a reviewer. Proposals have 4 states [26]: draft, submitted, on evaluation, approved. The goal here is to list all the proposals that are on the stage submitted, to choose the reviewer and move it to the next stage, as it is shown in figure 3.7.

Figure 3.7: List of all proposal waiting for reviewers

3.3.1.2 Info about the reviewer's compatibility with a certain proposal

Based on the algorithm described previously, it is possible to get users information about the compatibility with a certain proposal. After the admin is presented with a list of submitted proposals that need reviewers, the admin has to choose if they either agrees with the algorithm's suggestion or don't, using the page shown in figure 3.8

Figure 3.8: List of the best reviewers found for a certain proposal

If the admin wants more information about a certain user to better understand the compatibility with a proposal, the *More Info* button explains how that total percentage was obtained. Carlos, for example, has experience in four out of the six areas of interest involved in that proposal. The *More Info* results are demonstrated in figure 3.9. Once the reviewers are accepted, the proposal is moved to the state "On evaluation" and evaluation sheets are opened for each reviewer.

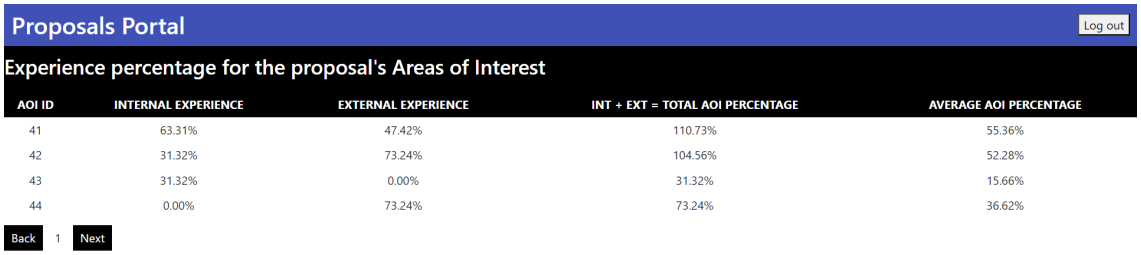

Figure 3.9: Details about a user's score for a certain proposal

3.3.2 Reviewers' process

In the figure 3.6, is represented the reviewer process that occurs in the middle of the admin's process. Once a proposal is attributed to a reviewer the evaluation begins. It can be saved as draft (state 1), and after that submitted (state 2). The reviewer should be able to see all the proposals that they have to review with easy access to its evaluation, like represented in figure 3.10

Figure 3.10: List of proposals to review for a certain user

3.4 Technological specifications

The work developed here was designed on top of an existing data structure created in MySQL. For that reason, the algorithm was built around the preexisting tables and structure. As explained previously, a chain of MySQL views were used to get the information in need. MySQL was a specific way for storing information, which was really useful while design the intelligence to choose reviewers. Being a relational database allows easy access

to information about the users and the proposals, and this is what this dissertation is about: to find a relation between users and proposals. The portal itself was also based in an existing structure and added elements to connected with the proposals' evaluation. The portal uses technologies such as Express, React, and Next. Express is responsible for the connection with the MySQL database, is a JavaScript framework and it was used for the server side (backend). React is a JavaScript library for interface developing. All the user interface was design using react. Next is a react framework responsible to optimize web development. The tools used to build this portal and how the interacted with each other is represented in figure 3.11.

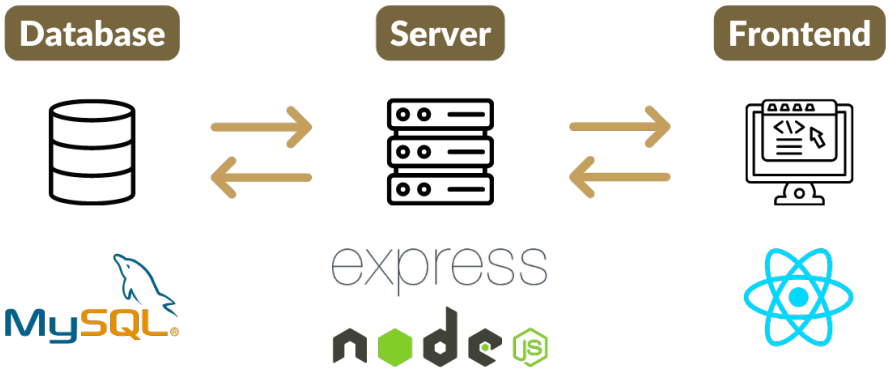

Figure 3.11: Portal tools' interactions

4

Testing and Hypothesis Validation

In the beginning of this dissertation, some questions were established, and this chapter proves the hypothesis studied in the previous chapters. To do that, a series of tests were performed to check the hypothesis viability, those tests and its results are presented in chapter 4.1. Lastly, the research questions will be answered and the hypothesis proved in chapter 4.2.

4.1 Testing

To test the algorithm, six types of tests were executed, with laboratory data:

- Test with only one area of interest;
- Test with multiple areas of interest;
- Test with external experiences with the same duration but one older than the other;
- Test with the same internal experiences but one older than the other;
- Test with evaluations not completed;
- Test with a perfect reviewer.

For all this tests, it will be described the reviewer's experience involved in that test. It is important to have in mind that the external experience is considered 100% if it had lasted four years, and if it is a current experience. It is also considered a perfect internal experience, three proposals reviewed in that day. They have to be recent experiences because age deprecation was set to 5% and 10% (internal and external, respectively). In all tests, the areas of interest will be referred to as numbers for better understanding.

4.1.1 Test with only one area of interest

João is a reviewer that has evaluated two proposals: one with three areas of interest (41, 42, and 43) and another one with one area of interest (41).

Marta has evaluated four proposals: one with three areas of interest (41, 42, and 43) and three proposals with three areas of interest (44, 45, and 52).

Carlos has evaluated the same proposals has João.

João has worked in the area of interest 41 for one year in 2011 and another year in 2022.

Marta doesn't have any external experience.

Carlos has worked two years from 2011 to 2012 in the area of interest 41, and has been working in that same area from 2022 until now. Carlos has also three years of experience from 2020 to 2022 in the areas of interest 42 and 44.

All the users' experience described previously are resumed in figure 4.1.

User	Internal Experiences	External Experiences
João	1 Proposal with areas 41, 42, and 43 1 Proposal with area 41	1 year in area 41 - 2011 1 year in area 41 - 2022
Marta	1 Proposal with areas 41, 42, and 43 3 Proposals with areas 44, 45, and 52	-
Carlos	1 Proposal with areas 41, 42, and 43 1 Proposal with area 41	2 year in area 41 - 2011 -2012 Currently in area 41 - 2022 3 year in area 42 and 44 - 2020 -2022

Figure 4.1: Users' experience

For the first test were created two proposals. Proposal A with area of interest 41; Proposal B is involved with area 44.

João, Marta, and Carlos all have experience in the area of interest 41, and for that reason they are all possible matches for proposal A. João and Carlos have similar backgrounds in this area and for that reason they have similar percentages, as it is possible to see in figure 4.2. They don't have a very high score because they have around two years of experience and two proposals reviewed in that area, so they have a better internal score than external. Marta doesn't have any external score and has reviewed only one proposal, for that reason she couldn't have a percentage higher than $\frac{1/3}{2} = 16,6(6)\%$.

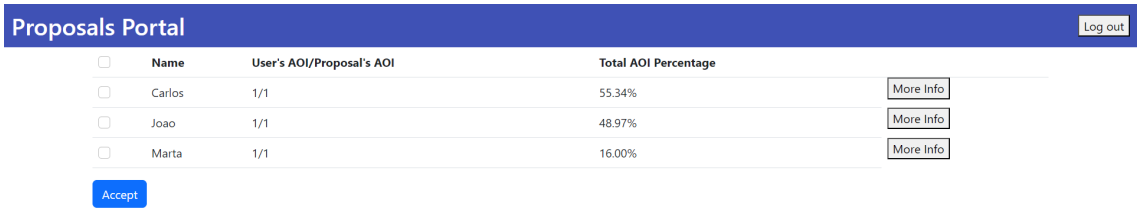

Figure 4.2: Results for the test with a single area of interest

For proposal B, João is excluded because he has never worked in that field. Marta has reviewed three proposals in area of interest 44, and for that reason she has almost 100% in internal experience, because those evaluations are very recent. She doesn't have any external experience and for that reason she only has about 50% compatibility with proposal B. Carlos has three years of experience that took place only 3 months ago, so it couldn't be greater than $\frac{3/4}{2} = 37,5\%$ as it is shown in figure 4.3

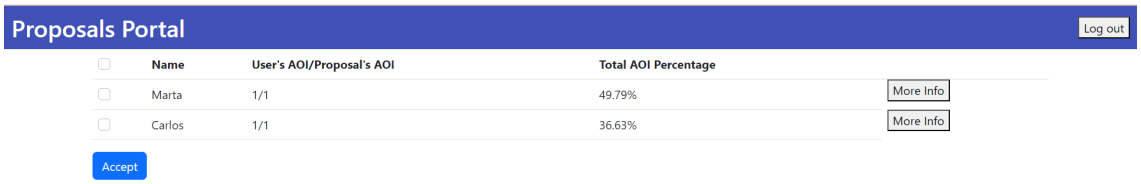

Figure 4.3: Results for the test with a single area of interest

4.1.2 Test with multiple areas of interest

A proposal with the areas 41, 42, 43, 44, 45, and 52 was submitted and all the users described previously, could be a match for this proposal as it is possible to see in the figure 4.4.

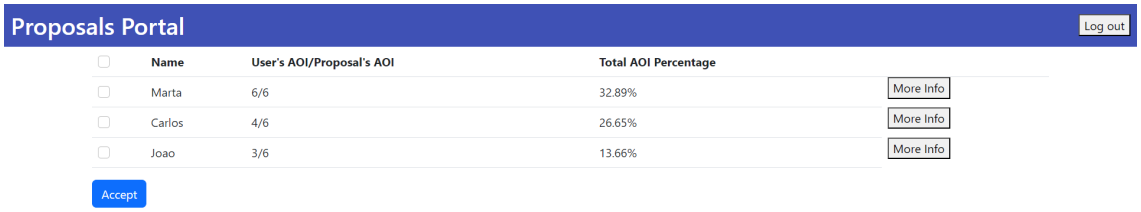

Figure 4.4: Results for the test with multiple areas of interest

4.1.3 Test with external experiences with the same duration but one older than the other

Bárbara has one year of experience in the area of interest 50 starting in 2022 until now. Filipe has also one year of experience, but it happened ten years ago. None of them have internal experience. They are both possible reviewers to evaluate a proposal in that area of interest, as it is possible to observe in figure 4.5. With a one-year experience and no internal experience, the best possible result is $\frac{1/4}{2} = 12,5\%$, which is the result is shown for Bárbara.

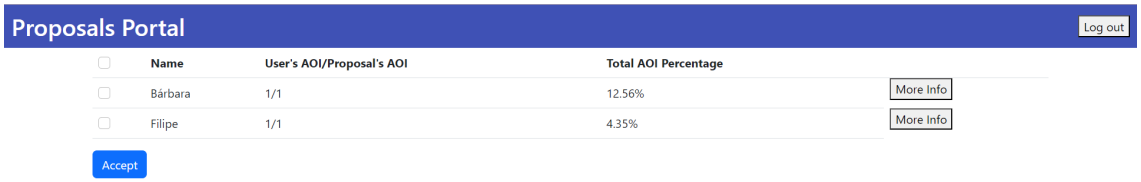

Figure 4.5: Results for the test with external experiences with the same duration at different times

4.1.4 Test with the same internal experiences but one older than the other

Tiago and Marco have both reviewed a proposal in area of interest 59, but Marco reviewed it recently and Tiago had reviewed it 10 years ago, for that reason the best result possible is $\frac{1/3}{2} = 16,6(6)\%$ as it is shown in figure 4.6

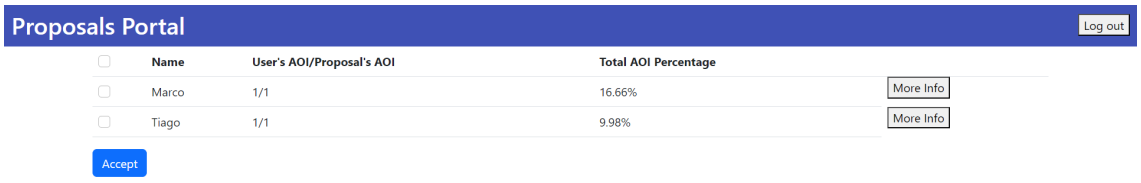

Figure 4.6: Results for the test with internal experiences with the same number of evaluation at different times

4.1.5 Test with evaluations not completed

Tiago and Marco have also reviewed a proposal with the area of interest 55, but Marco haven't submitted is evaluation yet. A proposal with the same area of interest has submitted, but the Tiago is the only reviewer available because Marco doesn't have a valid experience in this area, as it is shown in figure 4.7.

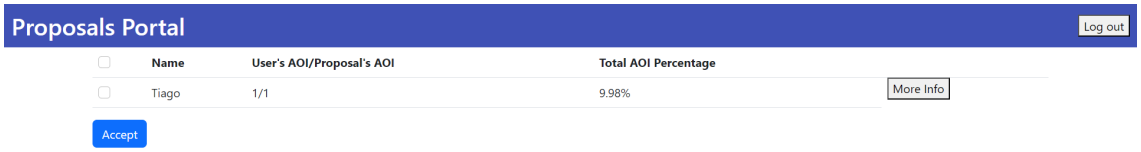

Figure 4.7: Results for the test with not completed evaluations

4.1.6 Test with a perfect reviewer

Maria has been working for four years in the area of interest 56, and she also has reviewed three proposals recently and for that reason she as a perfect score of 100% as it is shown in figure 4.8. In the figure is possible to see that the percentage is above 100% and this means that the parameters defined for a great reviewer (the maximum number of evaluations and maximum number of years of experience) are not right. The platform has reviewers with better experience that the one defined as the best. It could be possible to limit this value to 100% but that would mean reviewers with different levels of expertise would be defined as equal. By not limiting this values the admin is able to know that the criteria that was defined no logger corresponds to the best and maybe it could be better to update the *score_weights* table.

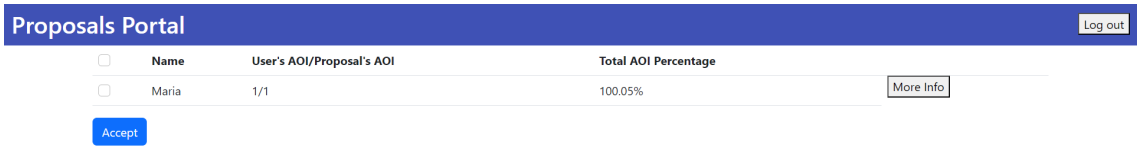

Figure 4.8: Results for the test with a perfect reviewer

4.2 Research Questions and Hypothesis Validation

Two research questions were defined in the beginning of this dissertation:

RQ1: "Is there a way to optimize and digitize the process of proposal evaluation?"

Answering RQ1, there is a way to optimize the evaluation process. In chapter 3.3 is proposed an evaluation method based on MySQL that works with states. Using the state variable present in every table, it is possible to check the stage of the process, for example is possible to know if a certain proposal has been submitted, if the reviewers

have been selected and if it has been evaluated, as well as the state of that evaluation (draft, completed or validated).

RQ2: "Is it possible to create an algorithm that proposes experts to evaluate a proposal based on their areas of expertise?"

For the RQ2, chapter 3.1 has a detailed explanation about the parameters that can be taken into account in order to build an algorithm that proposes reviewers to proposal evaluations. To decide which evaluator is the best, the level of expertise must be calculated, and that is explained in chapter 3.2. A series of calculations is made to get the score in percentage of each user for every proposal, so it is possible to list the best reviewers for a certain proposal. The method for getting that list was studied in many different forms in order to understand which method better fits the context. Every one of them has its advantages and disadvantages but the algorithm presented in chapter 3.2.2 was the one chosen to be implemented with this dissertation.

The hypothesis of this dissertation is:

> "Propose experts for proposals evaluations using the meta-information from the reviewers experience and the proposal's details."

The support system to propose expert peers for document review was built, implemented and tested according to various types of tests (chapter 4.1). The algorithm implemented in this dissertation lists every reviewers that have knowledge in the areas involved in a certain proposal. Is possible to select them to review a proposal and the database is prepared to store the evaluation results. For that reason, it is possible to say that there is a way to create this support system and that it serves its purpose.

5

Conclusions and Future work

In chapter 5.1, is a summary of all the work developed during this project. During the process of designing the algorithm and the evaluation system, some flaws were found that are described in chapter 5.2, followed by comments for the project's improvement.

5.1 Conclusions

The goal of this dissertation was to improve the evaluation process: in the way of making the evaluations more reliable by choosing the right reviewers and by making it easier for everyone involved. Choosing the right characteristic for Liu's work[1], was very important to decide the parameters that dictated the compatibility between a proposal and a reviewer, those parameters can be found in chapter 3.1. This was one of the main focus of this dissertation because defines the database structure. It was necessary to decide early on which parameters were necessary to collect and store, and this took a lot of time just for planning and designing. In order to get to a point where the algorithm to support the decision to propose expert peers was as perfect as possible, many other methods were left behind (chapter 3.2.1), and that lead to an algorithm with mathematical rigor (chapter 3.2.2). This was also a focal point of this dissertation, trying to achieve the best method for expert selection and testing methods that weren't used because of its flaws. Deciding which path was the right one and analysing every method for better results was very time consuming but were crucial to get the an algorithm with a reliable outcome. Of course the algorithm built and described along this dissertation is not perfect, and that will be discussed in the next chapter.

Besides the decision system, a platform support was built to interact with this algorithm and to get its results, in chapter 3.3. To test the strength of this support system six types of tests were performed, in chapter 4.1, and for this reason the hypothesis was validated successfully, as explained in chapter 4.2. All the tests executed were made with laboratory data, it would be very interesting to see how the algorithm would respond to real users, but those kind of tests weren't performed due to lack of time at the end of the development of this dissertation, making it impossible to test with real data. React is a

very useful framework but it would be much easier to get things ready for the user with a tool with a Create, Read, Update and Delete (CRUD) system built in. React has a way of interacting with the database in order to do those operations, but it was very hard to built a page just for creating a new experience and if that involved getting information from the database with a specif key (for example get the percentage score of a reviewer for a certain proposal) the processes to do that was confusing and very detailed. The platform for proposal submissions was already built and for that reason it was chosen to add the reviewers features on top of that portal and that means using React. For the work developed in this dissertation the main focus was to assist the reviewing process by creating a system of expert selection and for that reason building a portal for users interactions was a feature not a goal, so battling against the React complications wasn't a priority.

5.2 Future work

The algorithm was designed to take in consideration the areas of interest as well as the sectors. The database is prepared to get the sector experience: equivalent tables to *proposal_aoi* and *user_aoi* for the sector experience are *proposal_s* and *user_s*. The equations designed for the areas of interest, presented in chapter 3.2.2, work the same way for the sectors. This is an important feature to implement to make the experience calculation more complete and also a fast one to build because all the structure is already constructed and the equations just need to be adapted. The table *score_weights* (3.4) has two percentages that weren't used but were design to the final step of the calculation where the sector score and areas of interest score are added: *sector_weight* and *aoi_weight* are the percentages that dictate which parameter has more importance.

The algorithm had three more parameters that weren't implemented: conflict of interests, work overload and gender equality.

The conflict of interest should compare the companies and the countries of the reviewer and the candidate to check if they have a potential relationship. The reviewer must introduce the company that they are currently working on, and the candidate has to introduce that information while summiting a proposal. The information about the countries could be stored differently than how it is implemented now. Maybe it would be better to have a separate table just for the countries and compare the number id instead of the country name.

Work overload is to prevent reviewers to have a lot of proposals to review at the same time. The admin should get an alert if that situation occurs.

Gender equality is a parameter that gives some relevance to women. Have to separate table results for the best reviewers could be a way of encouraging the admin to select people from both tables, but this could have the opposite result and turns into more discrimination. It is important to give women relevance, but the way of doing this should be well thought.

This project is not prepared for malicious attacks, so it is a very vulnerable structure and can be easily hacked. SQL injection is a problem, but the solution to it comes with many other problems [27]. Preparing this platform for potential attacks would be great, but because it wasn't a priority in this dissertation, it wasn't implemented.

As a final note, the algorithm is built in a way where the best reviewer will always be the best reviewer. Of course, that if the conflict of interests, work overload and gender equality were implemented, there could be an improvement in the variety of the results, but the truth is that the more proposal they review, the more experts they get in that field. This could be solved by adjusting the parameters along the away, or for example give more weight to external experience than internal experience. Another solution could be encouraging the reviewers to learn more about certain areas to give them more experience. This is a problem of the algorithm that should be studied to find a solution that gives less experienced people opportunity to evaluate proposals as well.

Bibliography

[1] D. Liu et al. "How to choose appropriate experts for peer review: An intelligent recommendation method in a big data context". In: *Data Science Journal* 14 (2015) (cit. on pp. 1, 2, 21, 41).

[2] Mar. 2022. URL: https://dih4cps.eu/ (cit. on p. 3).

[3] W. H. A. Al-Zubaidi et al. "Workload-aware reviewer recommendation using a multi-objective search-based approach". In: *Proceedings of the 16th ACM International Conference on Predictive Models and Data Analytics in Software Engineering*. 2020, pp. 21–30 (cit. on p. 3).

[4] L. M. Camarinha-Matos. *Scientific research methodologies and techniques*. 2009 (cit. on p. 4).

[5] F6S. *F6S*. 2018. URL: https://www.f6s.com/ (visited on 06/02/2022) (cit. on pp. 7, 8).

[6] E. Commission. *European Commission- Funding & tender opportunities*. 2018. URL: https://ec.europa.eu/info/funding-tenders/opportunities/portal/screen/home (visited on 06/06/2022) (cit. on pp. 9, 10).

[7] E. Commission. *European Commission Funding & Tenders Portal - IT releases*. 2022. URL: https://webgate.ec.europa.eu/funding-tenders-opportunities/pages/viewpage.action?pageId=11304971 (visited on 06/28/2022) (cit. on p. 10).

[8] Getsitecontrol. *Getsitecontrol*. URL: https://getsitecontrol.com/ (visited on 06/24/2022) (cit. on p. 11).

[9] Getsitecontrol. *Add a job application form to your website*. URL: https://getsitecontrol.com/usecase/add-job-application-form-your-site/ (visited on 06/24/2022) (cit. on p. 11).

[10] J. Protasiewicz. "A support system for selection of reviewers". In: *2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE. 2014, pp. 3062–3065 (cit. on p. 13).

[11] M. S. Brown. "Transforming unstructured data into useful information". In: *Big Data, Mining, and Analytics*. Auerbach Publications, 2014, pp. 227–246 (cit. on p. 13).

[12] P. Zhang et al. "Data quality in big data processing: Issues, solutions and open problems". In: *2017 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computed, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*. IEEE. 2017, pp. 1–7 (cit. on pp. 13, 14).

[13] B. Noori. "Classification of customer reviews using machine learning algorithms". In: *Applied Artificial Intelligence* 35.8 (2021), pp. 567–588 (cit. on pp. 14, 15).

[14] N. Gulsoy and S. Kulluk. "A data mining application in credit scoring processes of small and medium enterprises commercial corporate customers". In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 9.3 (2019), e1299 (cit. on pp. 14, 15).

[15] J. Joyce. "Bayes' Theorem". In: *The Stanford Encyclopedia of Philosophy*. Ed. by E. N. Zalta. Fall 2021. Metaphysics Research Lab, Stanford University, 2021 (cit. on p. 14).

[16] D. M. Hawkins. "The problem of overfitting". In: *Journal of chemical information and computer sciences* 44.1 (2004), pp. 1–12 (cit. on p. 15).

[17] H. Uğuz. "A two-stage feature selection method for text categorization by using information gain, principal component analysis and genetic algorithm". In: *Knowledge-Based Systems* 24.7 (2011), pp. 1024–1032 (cit. on p. 15).

[18] T. H. Cormen et al. *Introduction to algorithms*. MIT press, 2022 (cit. on p. 16).

[19] G. Futschek. "Algorithmic thinking: the key for understanding computer science". In: *International conference on informatics in secondary schools-evolution and perspectives*. Springer. 2006, pp. 159–168 (cit. on p. 16).

[20] U. Manber. *Introduction to algorithms: a creative approach*. Addison-Wesley Longman Publishing Co., Inc., 1989 (cit. on p. 16).

[21] S. Bae. "Big-O notation". In: *JavaScript Data Structures and Algorithms*. Springer, 2019, pp. 1–11 (cit. on p. 17).

[22] R. Sedgewick and K. Wayne. "Algorithms and data structures". In: *Princeton University, COS* 226 (2007) (cit. on pp. 18–20).

[23] D. E. Ferguson. "Fibonaccian searching". In: *Communications of the ACM* 3.12 (1960), p. 648 (cit. on p. 18).

[24] S. S. Skiena. *The algorithm design manual*. Vol. 2. Springer, 1998 (cit. on p. 19).

[25] M. Abukhadijah. *Difference between symbol table and hash map data structures*. Aug. 2020. URL: https://stackoverflow.com/questions/43567461/difference-between-symbol-table-and-hash-map-data-structures (visited on 06/14/2022) (cit. on p. 20).

[26] L. Martins. *Portal for submission and review of proposals for open calls*. Master's thesis. 2023 (cit. on pp. 30, 31).

[27] B. Hoyt. *Don't try to sanitize input. Escape output.* Feb. 2020. URL: https://benhoyt.com/writings/dont-sanitize-do-escape/ (visited on 02/15/2023) (cit. on p. 43).