BUCHAREST UNIVERSITY OF ECONOMIC STUDIES

FACULTY OF CYBERNETICS, STATISTICS AND ECONOMIC INFORMATICS

**Bachelor’s Thesis**

Scientific Coordinator,

PhD Alin Zamfiroiu

Student,

Ana-Maria Schpak

**Bucharest**

**2020**

BUCHAREST UNIVERSITY OF ECONOMIC STUDIES

FACULTY OF CYBERNETICS, STATISTICS AND ECONOMIC INFORMATICS

**Automatization in the Analysis of the Scientific Co-authorship Network**

Scientific Coordinator,

PhD Alin Zamfiroiu

Student,

Ana-Maria Schpak

**Bucharest**

**2020**

Table of Contents

[1. INTRODUCTION 4](#_Toc43995733)

[2. CURRENT STATUS 6](#_Toc43995734)

[2.1. General concepts 6](#_Toc43995735)

[2.2. Existing Applications in the Field 11](#_Toc43995736)

[2.3. Advantages and Disadvantages 16](#_Toc43995737)

[3. PROJECTING THE SOLUTION ARCHITECTURE 17](#_Toc43995738)

[3.1. Requirements Specification 17](#_Toc43995739)

[3.2. Design of the System 21](#_Toc43995740)

[4. SOLUTION IMPLEMENTATION 24](#_Toc43995741)

[4.1. Progressive Web Application 24](#_Toc43995742)

[4.2. Technologies Used in Development 25](#_Toc43995743)

[4.3. Persisting Data 29](#_Toc43995744)

[4.4. Project Version-Control 33](#_Toc43995745)

[5. APPLICATION USAGE 34](#_Toc43995746)

[5.1. User’s Guide 34](#_Toc43995747)

[5.2. Security 35](#_Toc43995748)

[6. CONCLUSIONS 36](#_Toc43995749)

[References 37](#_Toc43995750)

[Annex 40](#_Toc43995751)

# INTRODUCTION

The main purpose of this bachelor’s thesis paperwork is to accommodate people interested in collaboration biometrics with the continuously evolving world of co-authorship networks. Such a network is comprised by authors, co-authors and the relationship between them which is treated as a binary variable: if an author collaborated with another author on at least one article, then there exists a link between them thus, offering means on the connection between these persons.

Secondly, the practical part of the thesis has as objective to create easier access to entities in a network and a better understanding of such social collaborations by providing a beautiful and dynamic graph.

With these being said, the general analysis of social networks and implicitly, of collaborations among authors of scientific papers, has been showing an upward trend so, the demand for developing and researching this domain has become more and more appealing. Many business opportunities arise from the continuous need of efficient tools and software applications. Entrepreneurs may also focus their attention on this topic, as it represents a niche of interest in what regards targeted advertisement and providing solutions for those authors who want to become popular. There are also writers of scientific articles who are interested only in increasing their collaboration score on different research web giants such as ResearchGate, Academia.edu, Google Scholar in order to gain more credibility so, providing a solution for them to easier find other persons who are interested in the same thing may be a profitable business model.

Furthermore, there exists on the research market a large number of scientific engines that provide millions of publications submitted by millions of researchers on which businessmen can conduct market analysis on. ResearchGate, Google Scholar, WorldWideScience, Academia.edu or Mendeley represent just a microscopical number of the total that lays on the World Wide Web. Speaking of the research market, research entities invest in collaboration between authors of scientific papers. One high acknowledged venue in this field is the IEEE Infocom conference which takes part every year and that has the intention to bring researchers, students or any other person together. Their purpose is to provide the means for communication between different types of people so they can exchange information, ideas and advice upon the scientific network thus, assuring costless knowledge for the respective attendees. Their approach is a great way of freely spreading insight on different topics and personal experience of certain trained persons, things that are hard to learn by reading articles or by simply surfing the Internet.

The motive of choosing the Analysis of the Scientific Co-authorship Network, mainly was the fact that it incorporates notions specific to several domains such as psychology, sociology, statistics, graph theory and further consolidates personal knowledge on them and on their fusion. The automatization part was done in order to offer direct access to relationships established among authors who think alike or who are interested in the same fields but also in order to aid in the tough intellectual work done by researchers in their day to day life by offering them a tool that provides a hierarchical dynamic visualization of the co-authors list of a certain researcher. This researcher and/or his collaborators may be of interest either in case of another researcher looking to collaborate in future articles, or an employer looking for well documented experts or a simple ResearchGate user looking for co-authors who think similarly to the author of the article they love.

In what regards the functional module, during the development phase several threats have arisen that put the application at risk. Sadly, external API’s provide different types of access to their pieces of information concerning data about the profile of a certain researcher. The access to the number of articles written with a certain co-author is available if the user is not logged in and the complete list of co-authors is done by being logged into the website, thus this generates a problem which takes a good amount of time to solve. In the complete list of collaborators there are two types, namely co-author and contributor, a thing that was not foreseen and that had to be treated differently than initially supposed. The last problem that appeared was human error, more explicitly the fact that users register several accounts which differ only by a couple of characters but represent in reality the same researcher. This is a well-known vulnerability concerning authorship networks which takes a great amount of effort and time resources to correct.

# CURRENT STATUS

## General concepts

To start with, the analysis of any typology of social network is very often observed making use of a mathematical structure called graph. The cybernetics field generally takes advantage of this tool due to many reasons, of which the following deserve recognition: graphs are easy to observe because they offer a great depiction of hierarchies, of relationships between entities, and without diving into advanced topics many people find graphs as being intuitive and not hard to understand at the first glance. This concept of simply freely looking on a graph, chart or diagram and intuitively drawing conclusions or trying to estimate and forecast either a change, or an event that may occur is called data visualization and is very important in the statistics field and in econometrics. Before conducting analyses, a quick visualization is done in order to non-formally validate future outcomes or just to have a general picture on what the official output resulted from computations may look like.

Not only individuals working in the cybernetics field use graphs, but also programmers do. Graphs and trees, which represent a particular type, are just a small part of what is called data structures. Data structures, as the name suggests, represent different ways in which data is stored into the memory space of a computer but also taking into account constraints regarding effectiveness of how information is stored depending on its type and volume. As mentioned before, this is exactly what graphs do.

In order to further benefit from a good understanding of collaboration networks, basic notions on the subject of graph theory are to be detailed. A graph is composed of a mathematical pair between a set of entities (formally called nodes or vertices) and a set of relationships (formally called edges or links) that are established among nodes if there is or has existed at a certain point in time at least one connection or interaction between their corresponding entities.

Moreover, graphs can be classified into directed type and undirected type. Directed graphs (also known as digraphs) contain edges directed by arrows. The former node of the link is called tail and the destination node is called head. This type of graph shows that through one directed link may only exist one relationship from the source vertex to the destination vertex, but one may add a complementary edge in reverse to show that the latter entity initiates the interaction with the former, too. Intuitively, undirected graphs are comprised by edges represented by simple lines that do not mean to show any direction as it may not be meaningful for any subject. Now that a theoretical distinction has been made between the two types, from now on only the undirected classification will be referenced as the authors collaboration network is represented by non-oriented relationships: main authors may in turn become co-authors in somebody else’s scientific articles.

Subsequently, the graph theory that has existed since the beginnings of mathematics has emerged in what is called network science. Nowadays, various situations in our daily lives from many domains may be depicted through graphs and this is why scientists needed a specialized field that takes this into consideration. Think of the Internet itself: it is represented by an enormous network in which nodes are represented by electronic devices that dispose of internet connection such as modems, routers, computers and so on, and edges are represented by any type of interaction that gets established between these devices: peer-to-peer connection, routing protocols etc. Take a look at Figure 1, this is what just a small part (30% of Class C network addresses) of the whole world’s devices that are connected to the Internet look like when represented as a giant graph.

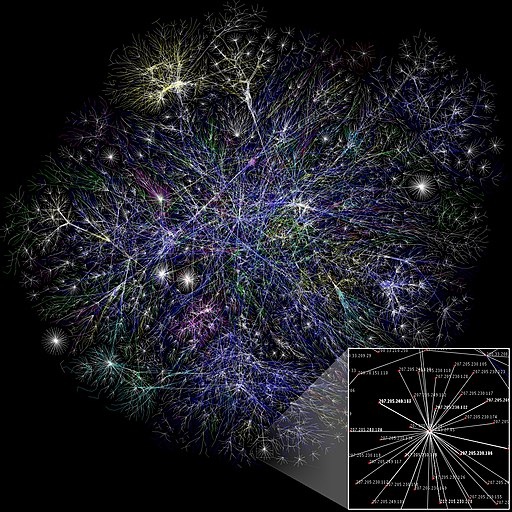


Figure 1. The Internet Map

The Internet is the biggest network that portrays connections between end-points, but in turn it is devised into sub-networks of smaller scale in order to be easier to maintain and troubleshoot, such as: Wide Area Network (WAN), Metropolitan Area Network (MAN) and the Local Area Network (LAN) and all of them represent graphs.

Once again, graphs are not applicable only in the Information Technology field, but also in various fields. When theory of graphs meets sociology, this is the right moment to talk about social networks. A social network is no different than any other network: it is composed of nodes represented by actual human beings and of edges represented by the actual relationships that develop between two different individuals. Although similar to any other network, graphs in this field are overridden as sociograms. One of the most well-known sociograms is every person’s family tree, but a famous one that gets referred in many sources is the social network made up of the characters of Victor Hugo’s historical novel, *Les Misérables*. The work of literature contains a large number of characters and its narrative thread is hard to follow and to remember. This is why people passionate about literature and graphs thought of depicting this huge intrigue as a beautiful social network in order to analyze different indices such as what is the most connected character or what are the social groups that form around. To have a better picture of how would such a complex sociogram look like, please take a look at Figure 2.

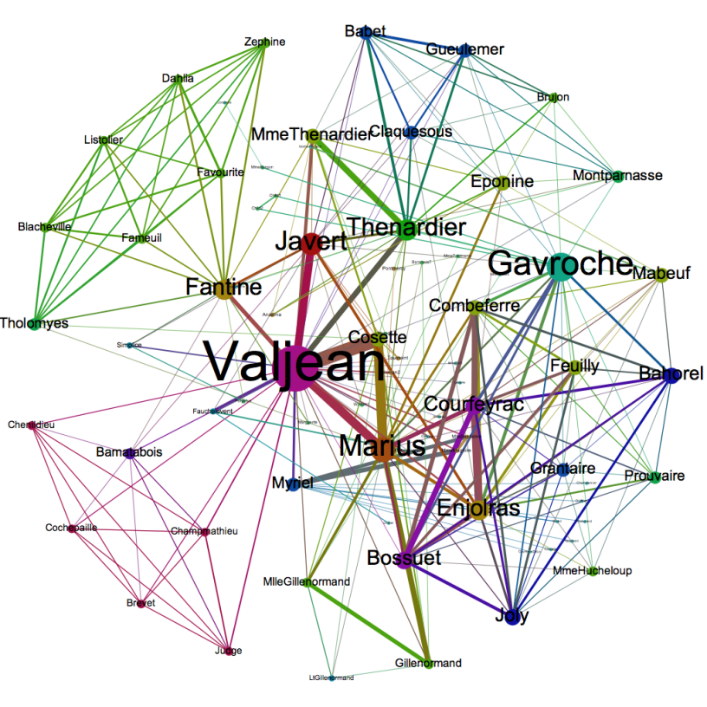


Figure 2. Les Misérables Social Network

Now that the necessary minimum of ideas on understanding graph theory has been established, further collaboration graphs can be detailed. This is what the analysis of scientific co-authorship network is in fact. It represents a collaboration network in which authors of scientific papers write together an article or simply bring contributions in order to benefit of knowledge from one another and share ideas on a certain topic of interest. Thus, the collaboration graph’s nodes are represented by authors from any research medium and edges represent the different interactions trained within. Of course, the same author can collaborate in the future with some of who he already did, but this means that the relationship already exists.

Subsequently, collaboration networks may develop different statistical measurements that may be of interest to its actors, but also to outsiders because they are conducted in order to measure how close or how far apart do certain writers situate on the network, how many nodes are between them, how connected they are with one another or with a certain subject. The input data that is represented by names of authors, is further transformed into information when indicators such as closeness or betweenness are computed in order to see which is the most integrated author. This layer is in turn wrapped and knowledge is resulted when reports are conducted based on computed statistic indices or when a decision to collaborate with an author who has a good betweenness score is taken. These outputs depict high-quality means of assessing attributes of interest for somebody looking to scientifically collaborate or who wants to gain new knowledge on topics developed by good authors.

Concerning the existing research on this topic, a really good analysis on collaborative networks is the one described by Newman (2004) in a series of lecture notes that dates since 1969. Basically, he started by collecting data from four databases for the time span of five years from the beginning of 1995 since the end of 1999. Two of the four databases are Medline which contains data in the medical field, Physics E-print Archive which contains preprints submitted by their own authors and which Newman split in 3 subsets to better differentiate between the various subjects of physics: astro-physics, condensed matter physics and high-energy theory. The other two databases are SPIRES that contains preprints and papers related also to high-energy theory in the physics domain and NCSTRL, a database that contains preprints related to computer science. Because these enumerated databases contain data recorded in different time periods, the extracted data had to be collected in sets on which mathematical union has been done with respect to the resulting common time span of 1995-1999. This way, coherent and representative measures can be done. After the exhaustive process of data collection, Newman computes many statistical indices specific to graph theory in order to find what is the most connected actor in these data sets, the betweenness of each of the scientific authors and the average number of papers written by each author. Out of these indicators, the most connected author is among the ones that deserve special attention. The most connected actor in a network is the one that is the most influential, because he is the one with the largest number of distinct relationships (collaborations on scientific articles), so he is the one who directs the way information goes from one point to another. As the saying goes, all roads lead to him. Speaking of this, another important measure was computed by Newman: the shortest path between any pair of authors in order to observe how one certain author may get in touch the easiest and fastest with another chosen author. The number of such pairs of entities in a network is called betweenness and it represents another important measurement in what regards graph theory. Without diving into hard to follow measurements, what may be of interest is what Newman observed regarding the ratio between the number of total authors and the total number of papers in each of the four data sets making use of the following chart in Figure 3.

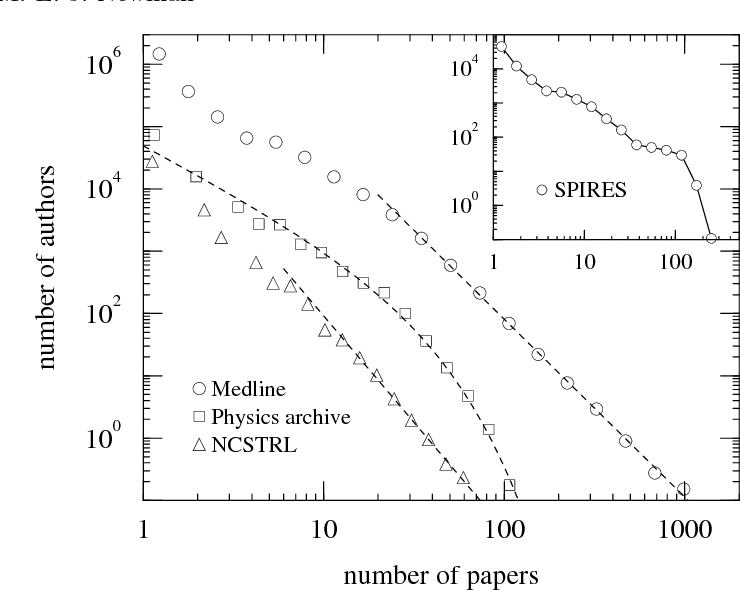


Figure 3. Authors-Papers Newman's Histogram

Newman’s histogram shows how the Medline and NCSTRL curves follow a power law, and contrary to his expectations, the Physics archive curve shows a truncated power law, as the pure one would be a poor measure. The writer also blames the cutoff in the Physics archive curve on the small time-span which limits the measurements making them not that representative. Concerning SPIRES curve, Newman explains that it clearly doesn’t show a power law and he says that this may be due to the fact that this database includes both preprint and papers and this may represent a practice that can alter the skewness of the curve. He also blames it on the small time period of the analysis. Looking on the illustration we can observe that the number of papers published by authors in medicine is exponentially higher than the one in computer science and intuitively we can leave it out on the fact that technology regarding computers was not as evolved back in the 90’s as it is in our times.

## Existing Applications in the Field

Despite the fact that co-authorship networks are a hot topic nowadays, this is also a very niched subject in contrast to the general terms of collaborative networks or social networks. This is why there are few applications that do a recursive search in order to automatically build the network of co-authors of an input author, but there are many applications with which one can build numerous types of graphs depending on what interests them.

The application with the largest number of features in the field is called SocNetV (Social Network Visualizer) and its most pleasing aspect is the fact that it is open-source but a great piece of software, too. It allows users to generate networks of scale-free, random small-world networks, Erdős–Rényi networks etc. or to choose from a gallery of pre-made networks in a drop-down menu with the entry named Famous data sets. The generated graphs are highly customizable, subgraphs can be highlighted, edges can be set to directed or undirected and the network can be transformed to appear symmetric. The solution further offers reports such as different matrices based on the generated network like adjacency, co-citation or degree types. It also offers the analyses specific to graph theory of which betweenness centrality, geodesics (shortest paths) and closeness centrality deserve mentioning.

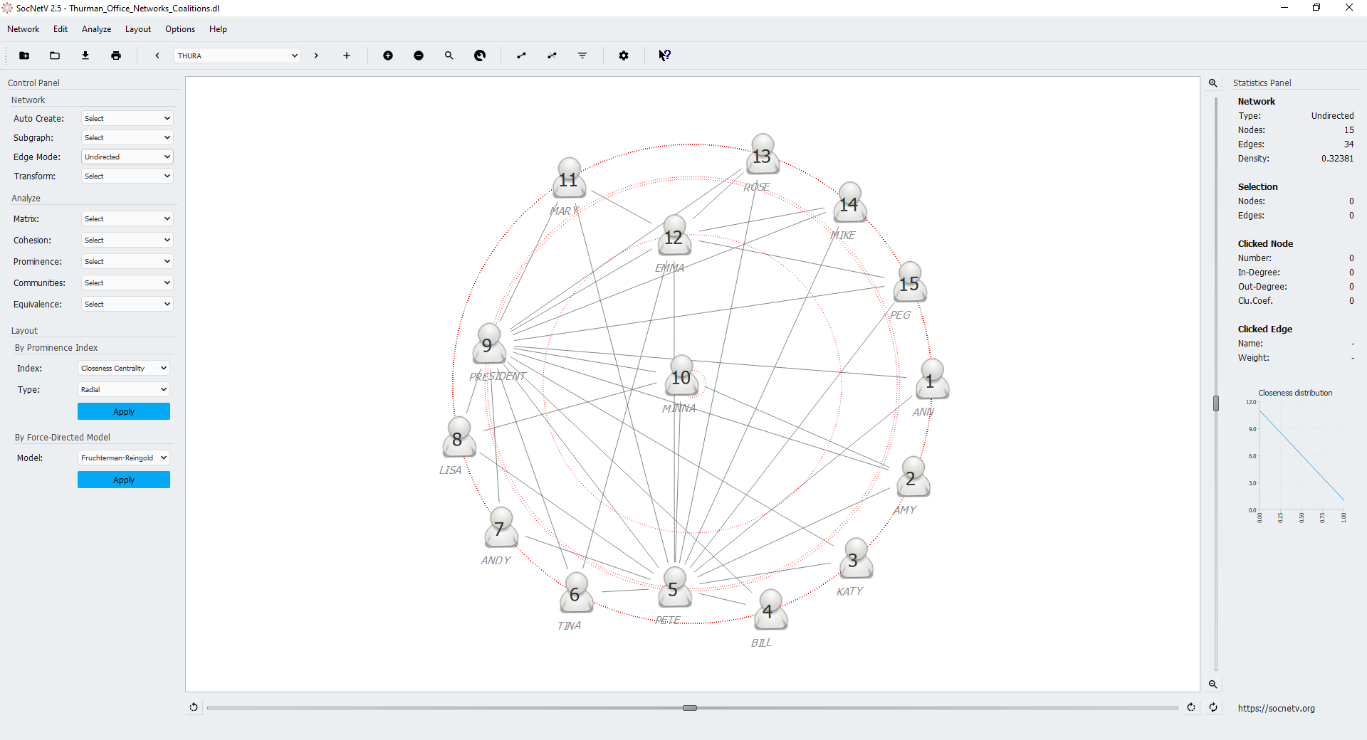


Figure 4. SocNetV Output

Further on, making use of these perks, a famous data set has been chosen in order to exemplify the use of SocNetV. The data set is called Thurman Office Networks Coalitions and depicts the relationships between the manager of a company and 14 other employees. The nodes are numbered and labeled with the surname of all the 15 entities selected. On the right-hand side of Figure 4, a Statistics Panels is available that shows details about the network such as the type (directed or not), the number of nodes and the number of edges, and the density proportion.

The only flaws in the application easy to spot are a little bug: the reports keep popping up in the browser when the drop-down menus in the Control Panel are clicked and the lack of the undo option. Also, what was less pleasing is the fact that once you add a node to the network it cannot be deleted anymore. These are the parts where the developers have to work a bit more.

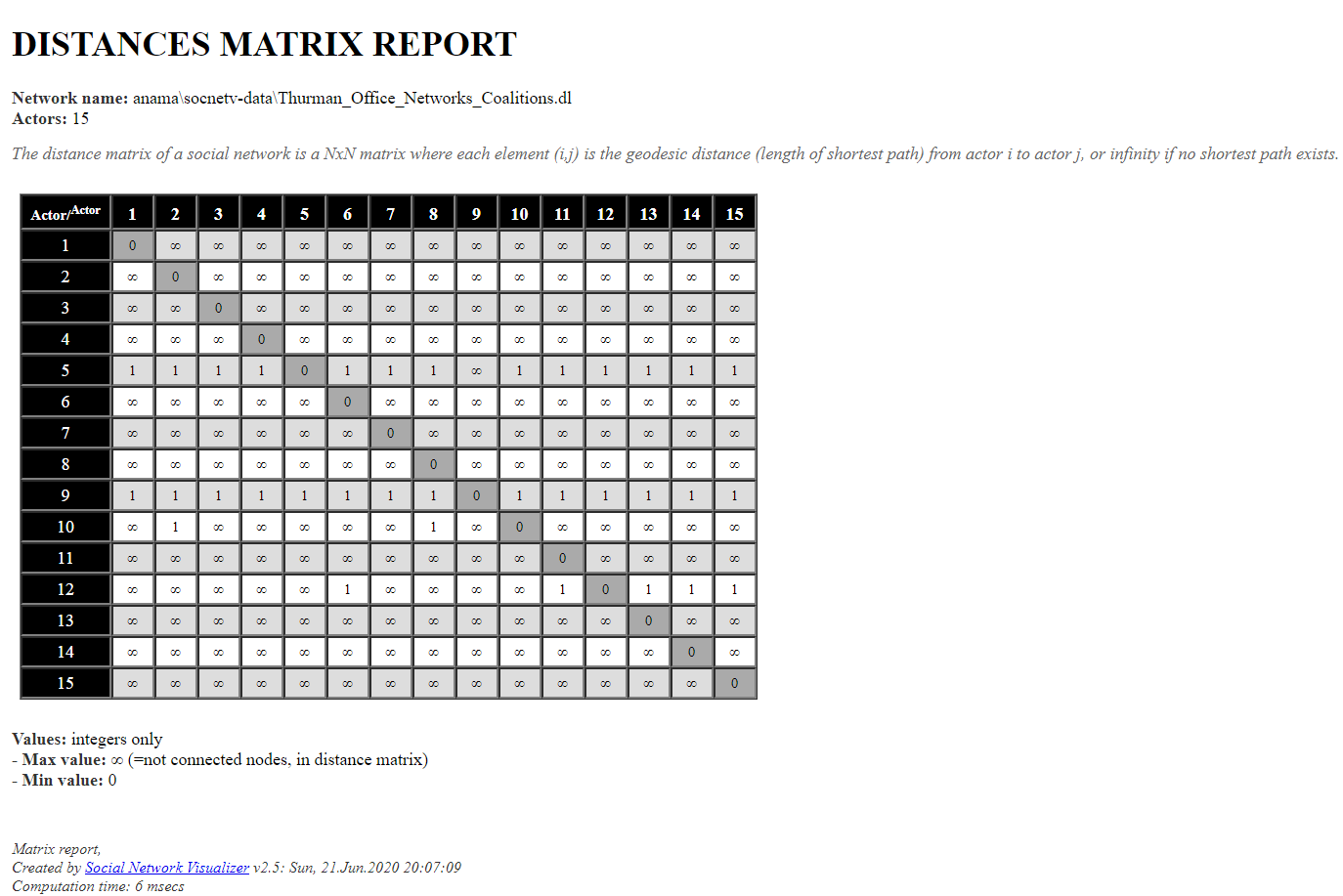


Figure 5. Distances Matrix SocNetV Report

To have an idea on how do reports look when generated by the application the betweenness centrality, geodesics matrix, distances matrix and closeness centrality were chosen and in figures 5 and 6 two of them can be observed. The reports pop out in the preferred browser.

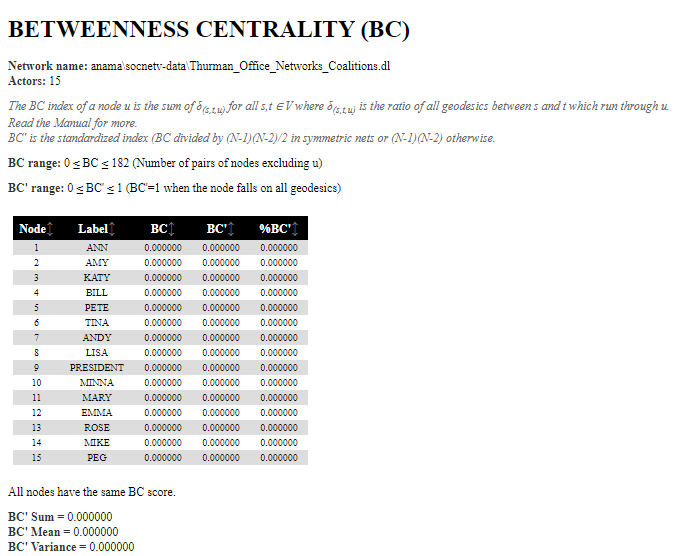


Figure 6. Betweenness Centrality SocNetV

The aforementioned application, SocNetV, is a cross-platform piece of software developed in C++ and Qt5. Its developers encourage programmers who use and like the application to suggest patches and fixes and assure that they will upload good possible updates to their development tree. Their project can be copied or modified as long as the license is maintained and as a plus, its documentation is free to access.

Subsequently, another really great application in the field is called Gephi, an application developed in Java making use of the NetBeans IDE and with the help of the OpenGL cross-platform API for beautiful vector rendering.

This software has an incredibly friendly GUI and it is really responsive and intuitive. It also benefits from statistics for networks, a list of pre-made famous networks including the previously mentioned Les Misérablesnetwork that can nicely be visualized in Figure 7.

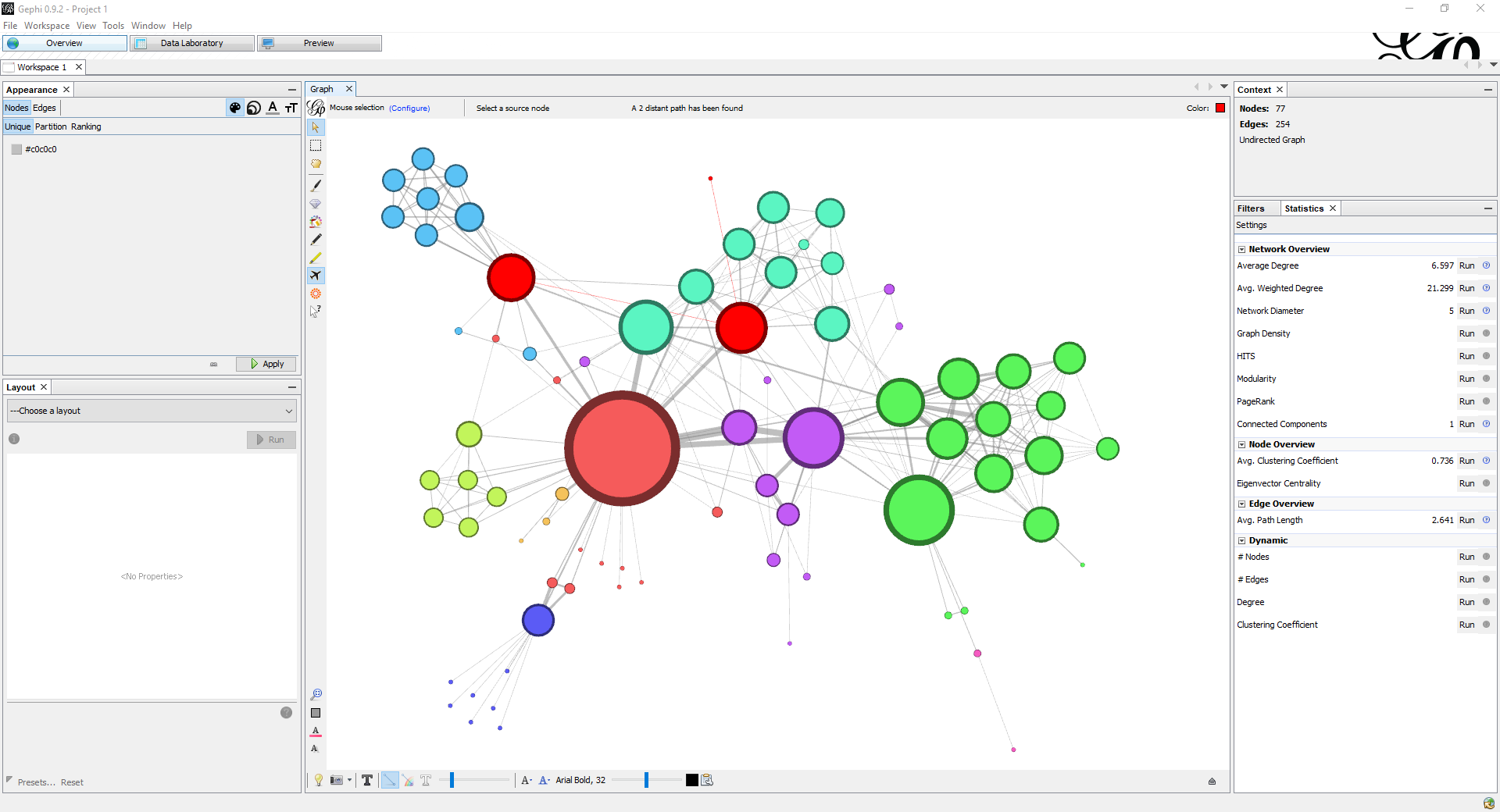


Figure 7. Les Misérables Network in Gephi

A really nice tool that clearly has a competitive advantage on similar applications is the one that realizes the shortest path between a manually selected source node to the selected destination node. The icon is represented by a plane icon which is highlighted and can be visible in Figure 7.

Another delightful aspect is that the application supports multiple languages such as German, French, Spanish and a few more. When generated, reports do not pop out in the browser as they do in SocNetV, but they pop out in a detachable window inside Gephi. A perk that was not available in SocNetV is that Gephi includes a tab called Data Laboratory where a beautifully structured table encapsulates the data set provided for the network that one might build. This table contains columns such as the id of the node, its label (if introduced), the nodes’ degree and the already explained and referred graph indicators previously. Below the table, options related to columns manipulation are available.

Moreover, other two existing applications in the field have been tested and examined. The first one is called Cytoscape and it is session-based software that was manufactured to serve medical purposes. It is meant to analyze effects of certain drugs, the evolution of severe diseases such as cancer and the interactions between genes. It is clear from the front page that this application is niched for medical scientists because the pre-made networks offered to test the environment represent medical examples such as Ivacaftor Coauthor.cys or Yeast Gene Interactions. These pre-made graphs are called Sample Sessions. The most pleasant feature of the application is its Tutorials section. When clicked, the app redirects the user to the project’s Github repository where really explicative and user-friendly instructions are detailed and beautifully presented. The environment provides a proprietary browser inside the application and an App Store from which a great number of complementary extensions can be downloaded to aid in the researchers’ and users’ work. In order to randomly generate a network, the Network Randomizer 1.1.3 extension has to be installed because the application does not provide this tool by default. What is not appealing about Cytoscape is its session-based principle because if closed or if it accidentally crashes the user loses their work. Also, the interface is kept really minimalistic, but it is not as intuitive and easy to use as SocNetV’s or Gephi’s.

The last reviewed piece of software is called UCINET and it uses a client window that encompasses a couple of modules: import of data using Excel, a matrix editor, a spreadsheet interface or even Notepad. This client also provides a Command Line Interface where users can write commands instead of using the given interface. UCINET also provides the so called NetDraw module in which similar to the aforementioned apps, users can randomly generate networks and conduct analysis. Speaking of the interface, it is very basic and it does not look very appealing, reminding of Windows applications’ theme. On the other hand, it allows working with matrices and has a pretty good number regarding statistics options. A really good feature is that the pre-made data sets with which one may draw networks are stored in the local files of the installation folder, making access easy and fast.

## Advantages and Disadvantages

To begin with, SocNetV and Gephi are the best applications in the field without debate also from a visual perspective, but also because they are intuitive and user-friendly. Gephi owns the competitive advantage in what regards performance, responsiveness and graphics. On the other hand, SocNetV has the competitive advantage concerning minimalism, generating random networks and the manner in which the graphical user interface is structured. Depending on user preferences, one may argue on choosing which out of the two is the superior one, personally I couldn’t decide.

Secondly, on an opposite side there are Cytoscape and UCINET because regarding visuals they clearly lack investment. Of course, from a modest user point of view these criteria do not matter, what matters in the end is how performant the app is, how fast it responds to big data sets which are very common in practice, outside simple educational purposes.

Alternatively, from a consumer perspective, visuals are very important because they represent the factor that draw attention and from a marketing point of view this is one of the most important aspects, if not really the most important, when it comes to selling products. Without debate, UCINET and Cytoscape lack a beautiful graphical user interface, but they are oriented into a different direction. Cytoscape is created for medical purposes and UCINET is clearly meant to serve big data sets, situations in which too much details would complicate things even more or would generate large resources costs. Such costs should not exist when speaking about such subjects.

In conclusion, SocNetV and Gephi are consumer-oriented, having beautiful visuals, but lacking support for big data, whilst Cytoscape and UCINET serve niche targets with minimal graphics, but good performance. The decision on what application to use should be taken on what does the user need, rather than on how beautiful the app looks.

# PROJECTING THE SOLUTION ARCHITECTURE

## Requirements Specification

First of all, the functional requirements of an on-going project that must be taken into consideration in the establishment phase of the proposed solution are identified and modelled using Use Case Diagrams. This type of behavioral diagram models the usage of the application by representing entities that take part into the process through the so-called actors that are graphically represented by stick man figures and the interactions that are maintained through use cases and that are graphically represented by labeled ellipses.

Further, a general use case diagram is built that serves the scope of explaining in user-familiar terms what is the application expected to behave like and what it is expected to be able to provide to the end-user. This general diagram is meant to explain the general intended functionality of the application without going into details about how this functionality should be realized. Taking a look at Figure 8, it is visible that the user has the ability to either register a new account into the system whose personal data might be validated or to directly log into the application if they already went through the registration phase. The user also has the possibility to search for an author name somewhere on a page of the application. From this point, the responsibility is transferred to the frontend and to the backend servers to deliver information on the client web page. In some way, on the backend server there will exist some persisted data from where the searched author will be retrieved if it exists. For this result the frontend will be responsible to render the co-authorship network and to display several statistics. As previously said, this phase does not offer information on how will all these functionalities actually be implemented, what technologies will be used or what resources will be needed. Its purpose is only to offer a starting point and an overall picture on what the final product is desired to be able to do in general terms.

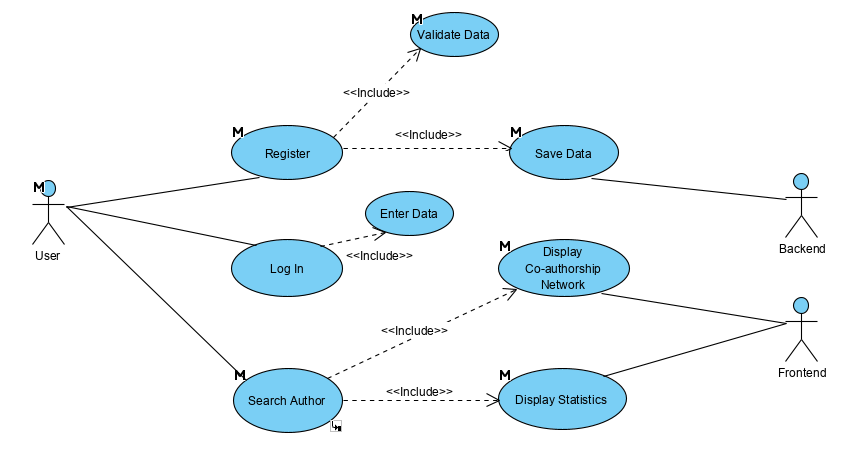


Figure 8. General Use Case Diagram

Moreover, diving deeper into the Search Author specific use case, a more detailed diagram of the same type can be viewed in Figure 9. As states in the general diagram, the Search Author use case includes the functionalities of displaying the co-authorship network and of displaying some correspondent statistics. The functionality provided by rendering the network in turn includes the functionalities of retrieving the possible co-occurrence if the author searched by the user is found and of rendering the actual graph. On the other side, the Display Statistics use case implies estimating some indicators and drawing some charts. This diagram represents a scenario of what the part of searching an author is intended to be functionally capable of doing in the future.

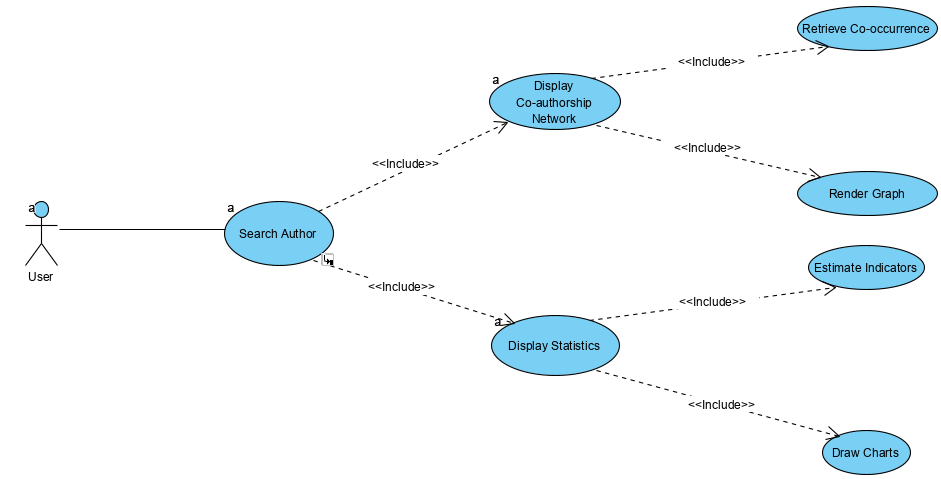


Figure 9. Search Author Use Case Diagram

In addition, each use case diagram has an attached description and each use case is assigned a unique code. In this phase, these diagrams are considered to be drafts due to the fact that they are prone to changes and the usage frequency of this specific use case may be considered as possibly occurring pretty frequent, anytime the user wants to analyze a certain researcher. The primary actor of the search of an author module is the user and it basically implies entering from the keyboard a scientific author’s full name from a specific research website. In case this author is found in the database its co-authorship network is built and rendered as an undirected non-weighted graph in order to analyze what is the most connected author and another significant statistical indicator specific to social networks. The trigger action of this use case is the possible decision of the user that they need to analyze how does a certain researcher collaborate when publishing scientific articles. The main flow of the Search Author use case is the following:

* the user configures the author name in a text field
* the user presses a search button and the name entered is searched in the database
* if found, the collaboration graph is rendered, otherwise an alert is triggered.

Not only use case diagrams are considered to qualify for the initial phase of a project, but also the minimalist initial class diagram with the observation that this diagram is of structural type. This diagram serves the same purpose as a general use case diagram, thus it does not give insight on how will the application be implemented. It is meant to give the client an initial idea on how the objects inside the system interact and this idea can be seen in the following figure, namely Figure 10.

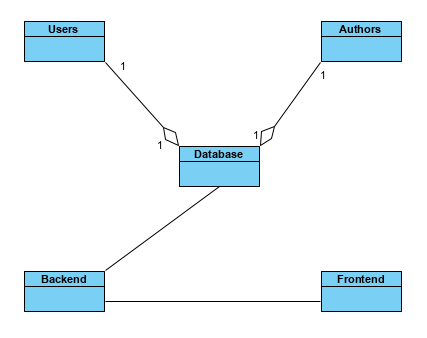


Figure 10. General Class Diagram

Subsequently, the class diagram is structured differently, namely the components are called classes and are represented by rectangles. These rectangles are partitioned because each class is in turn composed of two parts that will get explained in the design of the system section, where a detailed class diagram is going to be presented.

## Design of the System

To begin with, the design stage has the role of establishing how will the requirements previously set be implemented. In this step the project is split into modules and the actual structure of classes is established.

First, the classes of a class diagram are split into two distinct sections. The first one contains the attributes (characteristics) of the object and the second one contains the methods (functions). For example, taking a look at Figure 11, the Author’s private attributes are its name of type string (a sequence of characters) and a Boolean attribute called isParsed that show either a 0 if the author got parsed from an external API or a 1 otherwise. One of the methods of the Author’s class is its constructor which has its visibility set to public and it has the role to build an object of type Author given its name as input parameter of type string. The other method of this class is a setter called setIsParsed() which serves as purpose the possibility to modify if an existing author got parsed or not.

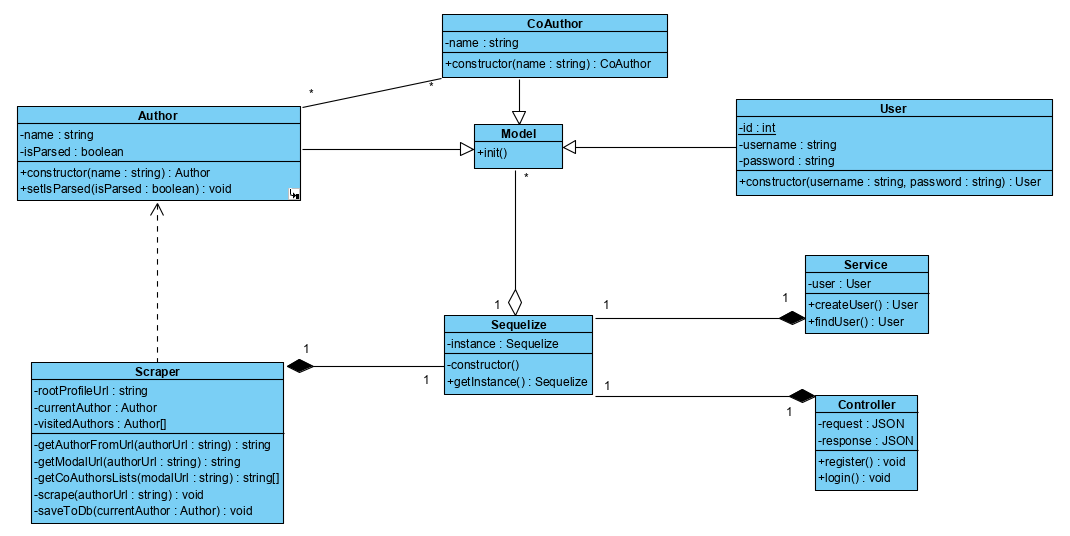


Figure 11. Detailed Class Diagram

Concerning the actual implementation, the system has been thought of as needing a User class that contains attributes specific to the needed credentials for creating an account and a constructor that builds the concrete User objects. Subsequently, the Author and CoAuthor classes were included in the schema both having in common the name attribute and a constructor. The Author class differentiates thanks to the attribute that show if a certain author got parsed from an external API to the database and a method to modify the value of this attribute. The User, Author and CoAuthor classes extend the Sequelize.Model class, which is an embedded class offered by the Sequelize database, which is represented in the diagram as the class named Sequelize implemented following the Singleton design pattern. A design pattern is an implementation solution to a very common problem, a thing similar to a tailor who uses a tailoring pattern to make the same blouse over and over again. The main problem regarding databases is the fact that by mistake, programmers may create more than one connection to the database, so singleton, as its name suggests, ensures the creation of a unique instance that can be globally accessed.

Furthermore, the Service and Controller classes both follow the Model View Controller design pattern. The Service contains a User and has the role of creating a user with certain credentials if it does not already exist in the database or find it and return it otherwise. The Controller contains a request and a response both of type JSON (JavaScript Object Notation) and has the role to either register the user received from the client request or to log them into the application if the account already exists into the database. The Controller also sends response status codes indicating if the operation was successful or if an error occurred on the way, case in which the browser is notified.

Briefly, the most complex class in the system is the Scraper class and contains an attribute of type string, an attribute of type Author and a collection containing authors. Being too complex, this class along with its behavioral section will be detailed in future chapters in which more technical descriptions are provided.

Concerning the design of the database, it is thought of following the Sequelize principles. In Figure 12 the general structure of the tables and the relationships between entities is presented. The User table contains the credentials required by the registration page that are needed in order to create a new account inside the application. These credentials are a unique integer attribute representing the primary key of the table called id, and three attributes of character type namely email, username and the password which can be maximum 255 characters long, the maximum supported number by MySql database dialect. The Author and CoAuthor tables each contain the id attribute representing the primary keys and a character type attribute called name that represents the full name of each author, respectively coauthor. The Author table also contains a binary type variable called isParsed that memorizes if a certain author is already saved to the database.

However, between the Author and CoAuthor tables the database portrays a join table with the role of holding a many-to-many (m:n) relationship. An author can write a scientific article with one or more coauthors. One of this coauthors may in turn, represent a coauthor for one or more authors and this is why the join table is needed. This helper table contains only one composite primary key that is formed from the primary key of the Author table and the primary key from the CoAuthor table.

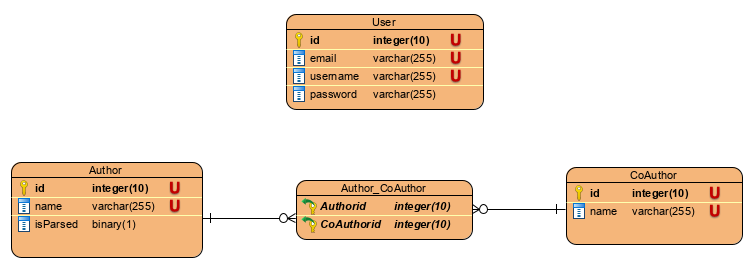


Figure 12. Database Design Diagram

# SOLUTION IMPLEMENTATION

## Progressive Web Application

To begin with, the proposed practical solution to the lack of automatic tools in collaborative networks is represented in this paperwork by a progressive web application (PWA) that runs both on web but also on mobile. I chose to develop such an application and not an ordinary Single Page Application (SPA) because it is very convenient to use for the first time and not only. A potential user can find it directly on the web or can access it directly through its URL and use it like they would with a normal web site, so a PWA holds the capability of being discoverable and linkable. Alternatively, if the user finds the application useful and wants to reuse it in the future they can simply download it directly on the device’s screen and access it anytime. Nowadays, the first contact with an application is very important so offering the user a great experience with your application should be of primary priority.

Second, a progressive web application is basically, a web site that has the aspect of an application, behaves like one and can benefit from a mobile application’s features and capabilities. As a result, a PWA is compatible with any browser and it can transform together with extensions provided by the used browser. This is what progressive means.

Furthermore, running on several types of devices such as desktop and mobile, such an application must be responsive meaning that it should fit any screen size and resources. A desktop will in most cases be more powerful than a mobile phone and that’s why developers should ensure that the app runs on low-end gadgets.

Alongside the installable properties of a PWA another great characteristic is the fact that the application is independent of Internet connectivity. It can run with no connection at all and it does not affect the experience in low-signal areas, this capability being achievable thanks to service workers which cache desired information from the application and give access to it at any moment. They run on a separate thread, so the main execution does not get affected.

## Technologies Used in Development

Firstly, the solution is developed using JavaScript as programming language. Although little familiarized with this language, I preferred choosing it when it comes to web development since it is the top choice in this regard and since it is specially created for web development purposes. Having little experience of three months with it I may say that JavaScript is neither hard nor easy to learn, especially when accustomed with the object-oriented paradigm.

Diving deeper, the application is developed making use of the notorious Node.js runtime environment and its amazing Node package manager (npm) which is somewhat like a Java library that contains predefined functions and modules that can be installed into one’s very own application. Npm also allows developers to run their application, but it does so through a Command Line Interface (CLI) called terminal. Thus, running a project requires entering commands like one does when using the Command Prompt in Windows or the CLI in Linux.

Therefore, the backend server is developed using such a framework called Express.js in order to serve a simple REST (Representational State Transfer) API (Application Programming Interface) whose interactions are visible in Figure 13.

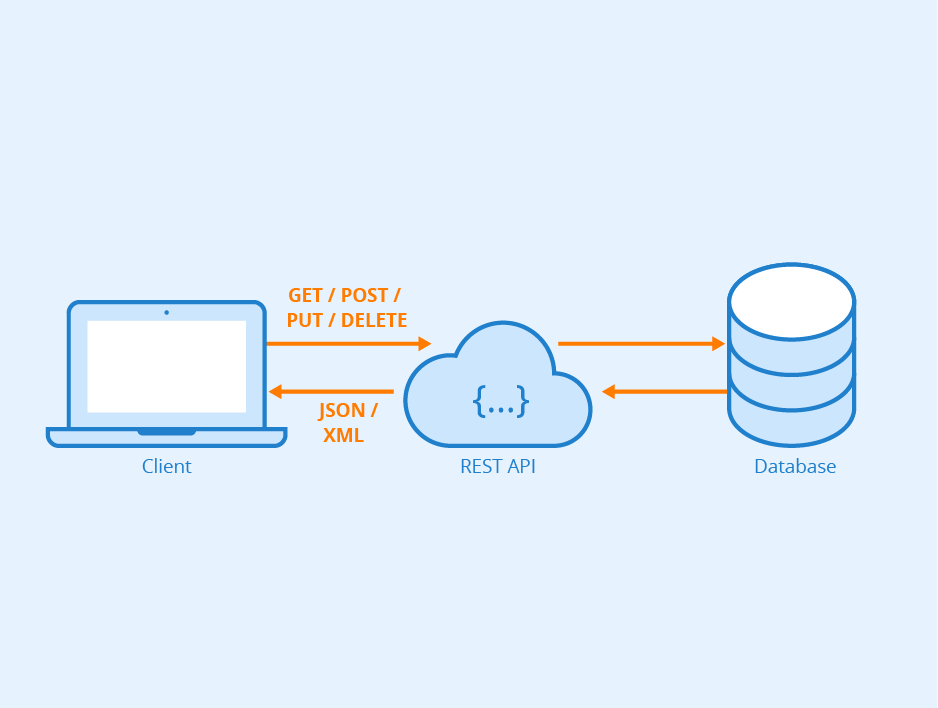


Figure 13. REST API - Author: Seobility - License: CC BY-SA 4.0

In general terms, an API is a collection of rules followed by the programmer who wants to allow the client to communicate with the server. In turn, a REST API is also a collection of rules that establishes that when the client asks to see a resource on the Web he should receive it when he links to a certain URL. This client action is called a request and the response should come from the server in the moment of connecting to the link corresponding to a web page through a port set by the developer. When assigning a port to a server, either frontend or backend, one should take into consideration the several types of port numbers. There exists a list of well-known ports also called system ports, which are reserved for identifying a network service. They range from 0 to 1023 and include ports reserved for well-known protocols such as HTTP (Hypertext Transfer Protocol) with number 80 or numbers 20 and 21 for FTP (File Transfer Protocol). The following category of ports is called registered ports that ranges from 1024 to 49151 and they are assigned by IANA (Internet Assigned Numbers Authority) for very specific services provided by an entity such as a big company. The last category comprises private ports and ranges from 49152 to 65535 and contains dynamic ports that cannot be registered with IANA but rather used for temporary means or automatically assignation.

In order to handle responses from the server to the client based on what does the client request, the application makes use of the Router middleware provided by Express. The routing process redirects responses on the route on which the client made a certain request. These requests can be done through several request methods that follow the CRUD (Create, Read, Update, Delete) principle. The Create function has as correspondent the POST method and it creates an entry in the database based on input data provided by the client. In the proposed application, the login and register routes function on POST methods. Even though the login process may be thought of as a GET request by some people, it is safer to do it by the means of a POST method since this way the username and password credentials will be sent in the HTTP body rather than directly the URL. The GET method only retrieves data and represents the equivalent for Read, while the PUT and PATCH request methods make updates or partial modifications to an existing piece of data and stand for the Update function from CRUD. The last method is called Delete and it is the direct correspondent of the Delete functionality. As expected, this request method removes a specified resource.

On the opposite side, here lies the frontend server implemented using React.js, Facebook’s initial JavaScript proprietary library created for drawing user interfaces that has lately been freed to the public and that has become the number one choice in frontend development. There’s no debate regarding the fact that React ranks high in the top since it is a great tool for user-intensive operations and for rendering constantly-changing views on the web application.

Subsequently, this is the reason for choosing React over other great similar frameworks like Angular or Ember. These frameworks are also ranked to the top, but they definitely have steeper learning curves. In order to understand React, one only has to have a minimal experience with CSS and HTML and in order to master it basic concepts such as state, props and lifecycle of a component have to be understood.

Further, the proposed PWA solution is clearly a user-intensive operations application since the collaboration graph’s input data can be constantly changed, new entries in the database can arise since researchers publish new articles every day or the user may be interested in analyzing several author’s networks in a very short period of time. Imagine re-rendering the entire page every time such a change occurs and think of how this impacts the overall performance of the application. This is how React counteracts such bottlenecks, by making use of a tool called Virtual DOM (Document Object Model) that saves the hierarchical structure of each wrapper component in the application into the browser memory. When a wrapper component suffers any minor change, the Virtual DOM compares the current state of the page with the one saved into memory and only re-renders the component that change, such as a simple button or even a more complex component such as a canvas or an entire form.

Moreover, components in React are reusable thus, the application looks and feels consistent, and the reuse of code makes it easier to be maintained and generally developed. To be more explicit, the registration and login components both make use of the Form React component, that suffered small changes such as the registration form has a double password field in order to make the user sure they did not misspelled when creating a new account while the login form does not need such a double-check condition because if the password is mistyped an alert is simply triggered and the user is allowed to re-enter it. This property of reusability also saves developers time, which is crucial nowadays.

Last, but not least one of the greatest tool that React provides for developers and that brings them competitive advantage is the React Developer Tools, a Chrome extension that can be easily installed directly in the browser and that offers support for inspecting the hierarchy of each component inside the application’s frontend. It also offers visualization upon each component’s state and props and a snapshot can be seen in Figure 14.

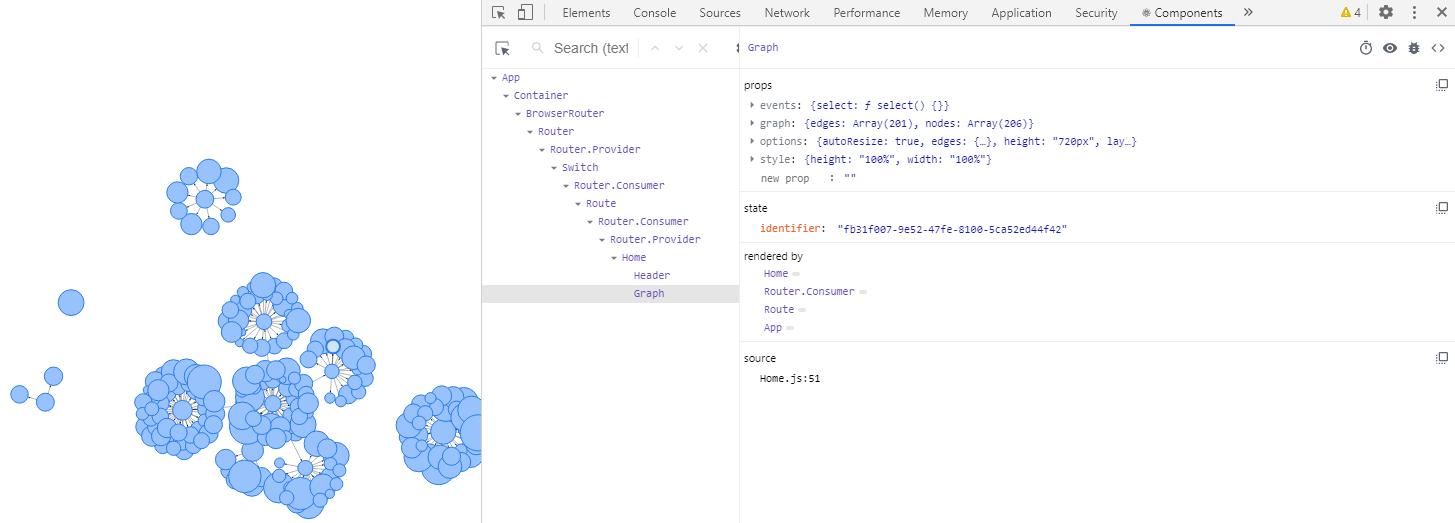


Figure 14. React Developer Tools

Accordingly, in order to keep a uniform theme for the application and to offer beautiful design, Semantic UI React has been chosen. Many aspects have been taken into consideration when making this choice of which the following may be recalled: Semantic UI React is a concise design tool created for theming minimalistic web applications that do not need complex views like the ones provided for example by Material UI. Moreover, Semantic offers support and compatibility for a mobile responsive experience. To wrap up, Semantic UI React is a great choice for PWA’s and generally for basic web solutions with few components and that do not need high-customized views.

Ultimately, for the rendition of the co-authorship network component Vis.js’s Network component has been used that aids in displaying networks consisting of nodes and edges. This component runs smoothly for networks of thousand nodes scale and provides great customization for the network such as data manipulation and physics methods to control the dynamics of the graph, all being drawn on a canvas HTML element.

## Persisting Data

In the first place, the proposed solution needed a database that had to contain existing scientific authors real names. Because one may not simply find such a list on the Internet at the moment of writing, an external API must be parsed in order to obtain such data.

In addition to this, it is a good practice to always eliminate risky external dependencies by persisting parsed data from the Internet into your very own application’s backend database. Hence, here comes the need of an ORM (Object-Relational Mapper). The relational property is imperative in the collaborative network application because an author may write articles with one or more co-authors and a co-author may in turn belong to one or more authors’ co-authors list. Long story short, many-to-many collaborative relationships develop between the entities of the network and this is the reason of needing a relational database. Also, the idea of writing SQL queries in an object-oriented manner using JavaScript as language could not be left out and this is the reason for choosing the Sequelize ORM. The Sequelize community of developers is also continuously growing thus, solutions to possible problems can be easier understood and faster solved this being also an important factor when choosing a certain technology out of a pool of numerous options.

To clarify this important decision of persisting external data to the internal medium of the bachelor’s thesis practical solution, in the rows to come more technical and in-depth analyses of written code will be presented. To illustrate the complex process of parsing data from an external API and saving it to the application’s database the explanations will be accompanied by figures.

All this process is done through the use of a non-anonymous function, called scrape that takes as its sole input parameter the URL from any research website of any scientific author the user is interested into, as seen in Figure 15. Further, the whole body of the function is wrapped into a try-catch block in order to intercept any possible upcoming error and prevent it from causing the program to crash.

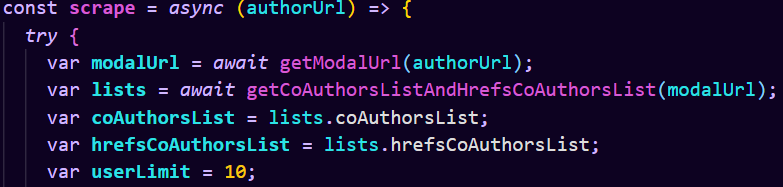


Figure 15. Scrape Function Signature and Variables Declaration

Here comes the part that needs focus because it makes use of recursion, a mathematical concept adopted in programming that is represented by a loop that stops when an imposed condition is met. A recursive function is a subprogram that calls itself until the initial condition imposed by the programmer is met. The instructions will keep executing for each entity that takes part into the loop. In the scrape process the following instructions will execute first for the provided author in the parameter of the call of the function, then for each of their co-author in their list of co-authors and then for each co-author of the co-authors of the author and so on until a certain limit of rendition for the network on the web page is met and as long as the current author includes a certain HTML attribute. This limit will have a possible maximum imposed by performance capabilities but, the user may choose a lower practical limit since they may be interested in visualizing only e.g., the third level of the hierarchy of the author provided in the call of the function. This value that the user is able to set will be introduced in a text field on the web page and the scrape function will be called when the user presses a suggestive button. The local variables declaration is done in the first lines of the function’s body in order to comply with clean code principles (note: at the moment of writing the practical solution has been finding itself in an in-progress state, so hardcoded values or logs to the console may appear but their scope is only for testing purposes). These variables are going to be used further in the scrape process. Further, the set of instructions that will get recursively executed is going to be detailed and explained.

The first instruction is to mark the provided author as visited by pushing him into a collection of visited authors. Because the input parameter is a URL of an author, a method that parses the exact full name out of that string is needed in order to be pushed into the collection and it can be observed in Figure 16. The substring() function returns the characters from the authorUrl string between the start parameter represented by the first character after the last slash and the end of the string. It also replaces the found underline characters inside the returned substring with a white space in order to separate the surname from the last name of a researcher.

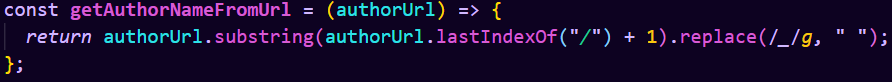


Figure 16. getAuthorFromUrl() Function

The second instruction is inserting the author into the database along with his list of co-authors. This is done using the findOrCreate() Sequelize.Model operation. Basically, if an entry row with the given name already exists in the Author table, the function returns value 0. An author has already been inserted in the database if and only if his isParsed flag is true. If the function returns 1, it means that the author has not already been found so it gets created on the spot with the given full name as visible in Figure 17.

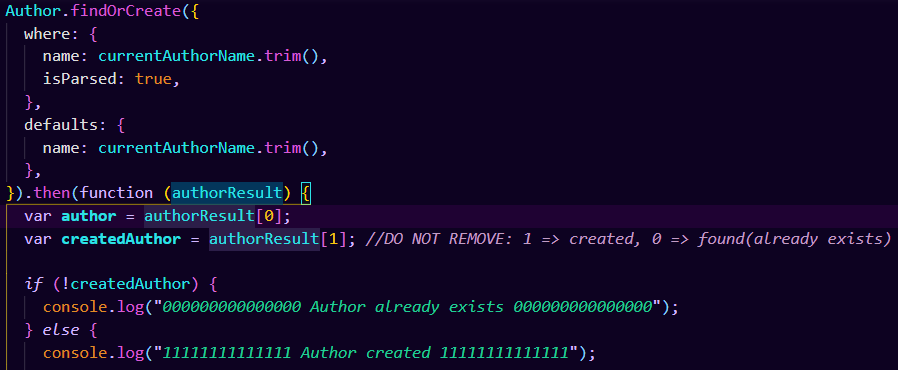


Figure 17. Inserting Current Author into Database

Further, the author’s corresponding list of co-authors previously obtained is getting traversed and the process is being repeated for each person in the co-authors list of the current author. The insertion of each co-author is done in the CoAuthor Sequelize.Model table. If the current co-author has already been inserted in the table because he might represent a co-author also for another already parsed previous author, his correspondent authorId is added to the through-table (many-to-many or join table term specific to Sequelize). If the coauthor does not already exist in the CoAuthor table, his entry inside the CoAuthor table is being created now and he also gets added in the AuthorCoauthors through-table in the coauthorId column corresponding to his authorId entry. Only now, may the current author be considered completely parsed and its isParsed flag be marked as true. Unless the save() method is called for a new entry, it will not appear in the table. All these instructions can be looked over in Figure 18.

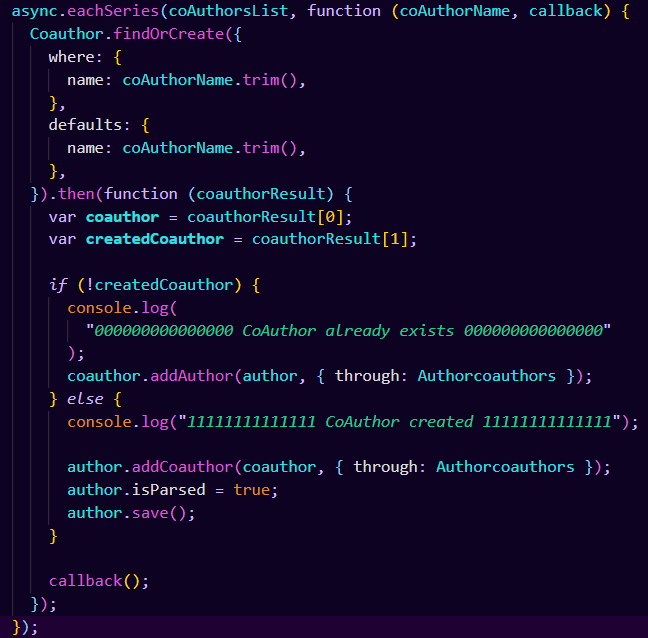


Figure 18. Co-authors List Persistence to Database

The following code snippet, available in Figure 19, represents the iterative traversing of the collection containing the href HTML attributes corresponding to each of the author’s coauthors. These href attributes contain the URL to the profile of each of the author’s coauthors. This is how the link to the profiles of the coauthors of the input author parameter is done in order to pass the scraping process further to each of them until the network reaches the limit of hierarchy levels or until a coauthor has already been previously pushed to the collection of visited authors.

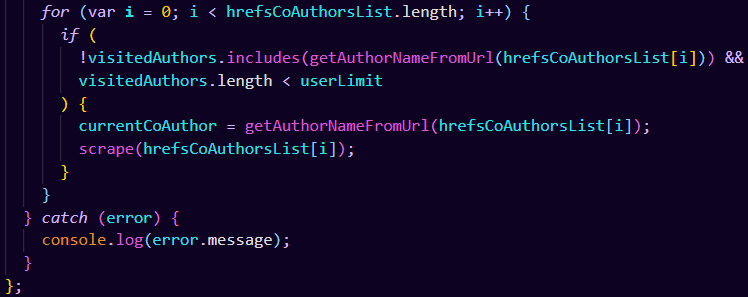


Figure 19. The Loop of Recursion

## Project Version-Control

Following the train of good-practices, in order to make sure that the project does not get lost because of several problems specific to locally storing a project such as hardware malfunctions or in order to be able to return to a previous version of the project in case a bug appears or a wrong path is followed, the practical solution is also stored on cloud through the help of Github in a private repository. Returning to a previous version is achievable thanks to commits, snapshots of each staged change made to the project.

This is what Github is used for, being a distributed version control system. It is also used by companies for branching their large-scale projects and for giving the employees a collaborative environment where they can coordinate their work.

# APPLICATION USAGE

## User’s Guide

First of all, because the application is not deployed into production, the only way it can be accessed is through what is called its network socket. The socket is formed from the IP address plus the port, the IP address being represented by the one belonging to the computer on which the frontend server is hosted and the port is set to 3000. If the frontend server is hosted on the same machine as the one accessing the browser the user can simply type in the search bar localhost:3000 and the Login page of the application gets rendered.

Further, if the user already has an account previously created he simply has to introduce their e-mail address and password and access will be granted to the Home page. Unless he has an account he must click on the Sign Up hyperlink that redirects them to the Register page where they are prompted to introduce personal credentials such as their e-mail address, a preferred username and a password. In order to prevent misspelling the user is prompted to confirm their freshly introduced password by typing it again in a separate text field called Confirm password. If the contents of the two password fields match and the user presses the Sign Up green button, the account is created.

After the new account has been created, the user gets automatically redirected to the Login page where they can authenticate into the application using the freshly created account. When ready, they can press the Login button and the Home page gets rendered.

From here, it is up to the user’s preferences on what they desire to make the application do. On the Home page the user may enter the full name of an author of scientific papers published on a preferred research web site. If the name does not exist in the application’s database, an alert is triggered that informs the user that no network can be rendered for the provided researcher name and the user is also prompted to check for any misspelling. On the Home page the user also has the possibility to set the number of levels of hierarchy that they want to get displayed for the to-be-rendered network. If the user introduces an abnormal value, they will be prompted to choose a number between certain limits. Unless specified, a default convenient value in terms of performance aspects is automatically chosen by the application.

In order to see graph-specific statistics such as what is the most connected author in the rendered network for which the user chose parameters, the user can press the Show Statistics button. Another interesting analysis based on the output graph is through how many nodes should one get in order to get from node X to node Y. This is also a statistic that the user can choose to see if they press the Show Statistics button.

## Security

To begin with, one of the most basic actions in counteracting malicious external attacks such as identity theft is to avoid keeping passwords in plain text inside the database by encrypting them. The encryption part is taken care of using Crypto.js in SHA-256 hashing algorithm. The framework prevents password from saving in plain text by directly saving their encrypted value and checking against the entered password to see if its encrypted value matches the persisted one. This way, the database does not even get to see the actual text of the password.

On the other hand, the most common cybernetic threat is SQL Injection. The application makes use of the Sequelize ORM whose developers say that it’s been taken care of and at the very moment Sequelize is not vulnerable to such threats. Although, in 2019 a vulnerabilities-detection oriented website called Snyk discovered three vulnerabilities in Sequelize identified by their CVE number as CVEs CVE-2019-10752 CVE-2019-10749 and CVE-2019-10748. They notified Sequelize’s developers about their SQL Injection vulnerabilities and by the end of June 2019, all three issues have been eradicated. Thus, Sequelize and its users enjoy its capabilities without fear of SQL Injection as long as enthusiasts keep their versions up-to-date.

# CONCLUSIONS

The vast notions pool regarding graph theory and network science alongside social collaborative networks has given readers a glimpse on how complex the mathematical notion of graph is and accommodated them with the basics of this topic that is so important in understanding social networks at their best.

The topic of co-authorship networks is one of great interest in the evolving world of technology, people being keen on collaborating more and more thanks to the easiness of getting in touch digitally through different communication media that become more familiar to us. Another important factor that facilitates scientific collaboration is how does transport get cheaper day by day a thing that was costly back in the days. We hear around us a lot about how people travel simply for attending conferences on different topic of personal interest, things that were not that easy ten years ago. Nevertheless, an author’s personal experience is an insight of great value to gain directly from a human being instead of spending time researching on your own on the Internet or though books and journals.

The practical solution proposed by this thesis brings a new tool to the table that can be used by researchers in order to make a piece of their scientific work easier and faster. The visuals of their colleagues networks or of their idols in the research world can represent the means to rapidly observing changes in these networks and analyzing statistics similar to what is the most connected co-author they personally know or they want to get to know in the future from a scientific perspective.

On what regards future development of this research paper and its practical solution, one may develop a good open-source API containing a great list of scientific researchers, their co-authors, how many articles have they written with author X and on what general topic. On a different direction, making use of such a complete and easy to access potential API developers may focus their attention into extending the network provided by the current solution through transforming the graph into a weighted one that should show number of articles written between two different authors and what topics do those articles regard. Sadly, this is not achievable in the lack of a rigorous open-source API so this is the greatest current limitation of the solution.

# References

addupe. 2019. *Dev.* 12 8. https://dev.to/addupe/react-state-4oip.

Bhushan, Mukundh. 2019. *Medium.* https://medium.com/faun/beginners-guide-to-version-control-using-git-and-github-8bf44b421140.

Britt, Matt. 2006. "WIKIMEDIA COMMONS." December 1. https://commons.wikimedia.org/wiki/File:Internet\_map\_1024.jpg.

Buna, Samer. 2017. *freeCodeCamp.* https://www.freecodecamp.org/news/yes-react-is-taking-over-front-end-development-the-question-is-why-40837af8ab76/.

Crutchfield, Constance. 2018. *Medium.* 10 14. https://medium.com/@cgcrutch18/stateful-vs-stateless-react-components-13f647f7fc4.

Depold, Sascha. 2014. *MIT License Copyright (c) 2014-present Sequelize contributors.* Berlin: Sascha Depold.

ELUWANDE, YOMI. 2019. *rwieruch.* https://www.robinwieruch.de/react-semantic-ui-tutorial.

Expressjs. 2018. *Expressjs.* https://expressjs.com/en/guide/routing.html.

Farrugia, Kevin. 2016. *SMASHING MAGAZINE.* https://www.smashingmagazine.com/2016/08/a-beginners-guide-to-progressive-web-apps/.

Glänzel, Wolfgang. 2016. "National characteristics in international scientific co-authorship relations." *Scientometrics* 51 (1): 69-115.

Google Developers. 2019. *Web.* https://developers.google.com/web/ilt/pwa/introduction-to-service-worker.

Haldar, Mahesh. 2018. *Medium Bits and Pieces.* https://blog.bitsrc.io/what-is-a-pwa-and-why-should-you-care-388afb6c0bad.

Hoyos, Mario. 2018. *Bits and Pieces.* https://blog.bitsrc.io/what-is-an-orm-and-why-you-should-use-it-b2b6f75f5e2a.

Joshi, Vaidehi. 2017. *Basecs: A Gentle Introduction To Graph Theory.* https://medium.com/basecs/a-gentle-introduction-to-graph-theory-77969829ead8.

Katz, J. Sylvan, and Ben R. Martin. 1997. "What is research collaboration? ." *Research Policy* 26 (1): 1-18.

Laudel, Grit. 2002. "What do we measure by co-authorships?" *Research Evaluation* 11 (1): 3-15.

Liew, Zell. 2018. *SMASHING MAGAZINE.* https://www.smashingmagazine.com/2018/01/understanding-using-rest-api/.

Liu, Xiaoming, Johan Bollen, Michael L. Nelson, and Herbert Van de Sompel. 2005. "Co-Authorship Networks in the Digital Library Research Community." *Information Processing & Management* 41 (6): 1462-1480.

Małek, Piotr. 2019. *Railsware.* https://railsware.com/blog/why-use-react/.

MILINAUDARA. 2014. *Optimism.* https://milinaudara.wordpress.com/2014/05/24/an-introduction-to-sequelize-js/.

Newman, Mark E.J. 2004. "Who Is the Best Connected Scientist?A Study of Scientific Coauthorship Networks." In *Complex Networks. Lecture Notes in Physics, vol 650*, edited by Hans Frauenfelder, Zoltan Toroczkai Eli Ben-Naim, 337–370. Berlin: Springer.

Newman, Mark. 2004. "Fig. 1. Histograms of the number of papers written by authors." *Complex Networks. Lecture Notes in Physics, vol 650.* Edited by Frauenfelder H., Toroczkai Z. Ben-Naim E. Berlin: Springer.

Newman, Mark. 2004. "Who Is the Best Connected Scientist?A Study of Scientific Coauthorship Networks." In *Complex Networks. Lecture Notes in Physics, vol 650*, edited by Frauenfelder H., Toroczkai Z. Ben-Naim E., 337-370. Berlin: Springer.

Pandit, Nitin. 2020. *C# Corner.* https://www.c-sharpcorner.com/article/what-and-why-reactjs/#:~:text=React%20allows%20developers%20to%20create,view%20in%20the%20MVC%20template.

Patel, Priyesh. 2018. *freeCodeCamp.* https://www.freecodecamp.org/news/what-exactly-is-node-js-ae36e97449f5/.

Sahin, Kutlu. 2017. *JotForm.* https://stories.jotform.com/7-reasons-why-you-should-use-react-ad420c634247.

Savić, Miloš, Mirjana Ivanović, and Lakhmi C. Jain. 2019. *Complex Networks in Software, Knowledge, and Social Systems.* 1. Warsaw: Springer International Publishing.

Sridharan, Mithun. 2015. *Social Network Analysis of Les Miserables.* May 20. https://jixta.wordpress.com/2015/05/20/social-network-analysis-of-les-miserables/.

TAL, LIRAN. 2019. *Snyk.* https://snyk.io/blog/sequelize-orm-npm-library-found-vulnerable-to-sql-injection-attacks/.

Vivah, Linda. 2017. *Medium.* https://medium.com/@LindaVivah/the-beginners-guide-understanding-node-js-express-js-fundamentals-e15493462be1.

W., Benjamin. 2018. *Medium.* https://medium.com/@benjaminpwagner/using-sequelize-hooks-and-crypto-to-encrypt-user-passwords-5cf1a27513d9.

Web Stack Academy. 2019. *Blog.* https://www.webstackacademy.com/blog/top-5-reasons-choose-javascript-java-web-development/#:~:text=JavaScript%20has%20emerged%20are%20more,it%20is%20still%20widely%20used.

Wikipedia. 2020. *Wikipedia.* https://en.wikipedia.org/wiki/List\_of\_TCP\_and\_UDP\_port\_numbers#Dynamic,\_private\_or\_ephemeral\_ports.

Annex

Figures List

[Figure 1. The Internet Map 9](#_Toc43992220)

[Figure 2. Les Misérables Social Network 10](#_Toc43992221)

[Figure 3. Authors-Papers Newman's Histogram 12](#_Toc43992222)

[Figure 4. SocNetV Output 13](#_Toc43992223)

[Figure 5. Distances Matrix SocNetV Report 14](#_Toc43992224)

[Figure 6. Betweenness Centrality SocNetV 15](#_Toc43992225)

[Figure 7. Les Misérables Network in Gephi 16](#_Toc43992226)

[Figure 8. General Use Case Diagram 20](#_Toc43992227)

[Figure 9. Search Author Use Case Diagram 21](#_Toc43992228)

[Figure 10. General Class Diagram 22](#_Toc43992229)

[Figure 11. Detailed Class Diagram 23](#_Toc43992230)

[Figure 12. Database Design Diagram 25](#_Toc43992231)

[Figure 13. REST API - Author: Seobility - License: CC BY-SA 4.0 27](#_Toc43992232)

[Figure 14. React Developer Tools 30](#_Toc43992233)

[Figure 15. Scrape Function Signature and Variables Declaration 32](#_Toc43992234)

[Figure 16. getAuthorFromUrl() Function 33](#_Toc43992235)

[Figure 17. Inserting Current Author into Database 33](#_Toc43992236)

[Figure 18. Co-authors List Persistence to Database 34](#_Toc43992237)

[Figure 19. The Loop of Recursion 35](#_Toc43992238)