GitCV

GitCV

Design and implementation of a software system for human resourcing in programming

by

D. Greasidis

to obtain the diploma of Electrical and Computer Engineering
at the University of Thessaly,
to be defended publicly on Tuesday June 20, 2017 at 13:00 PM.

GitCV can be found at
`https://www.gitcv.com`
An electronic version of this thesis is available at
`https://www.e-ce.uth.gr/research/theses-technical_reports`

Student number:	1624, 1712066	
Project duration:	February 1, 2017 – June 1, 2017	
Thesis committee:	Prof. M. Vavalis,	Univ. Of Thessaly, supervisor
	Ass. Prof. D. Katsaros,	Univ. Of Thessaly, supervisor

Contents

1 Introduction

1.1. Abstract

1.1.1. Abstract in Greek

Στη σημερινή εποχή, ο μεγάλος όγκος μη φιλτραρισμένων δεδομένων είναι εμφανής. Προκειμένου να εξαχθούν χρήσιμα συμπεράσματα από τα δεδομένα, είναι απαραίτητο να φιλτραριστούν, να διαγραφούν άχρηστες πληροφορίες και να παρουσιαστεί η πληροφορία που εξήχθη με εύληπτο και ουσιαστικό τρόπο. Τα δεδομένα συνήθως χαρακτηρίζουν τους ανθρώπους της σημερινής κοινωνίας σε μεγάλο βαθμό και προέρχονται από τη δραστηριότητά τους σε διάφορα κοινωνικά μέσα, κοινότητες και άλλους σχετικούς ιστότοπους. Θα μπορούσαμε να αναλύσουμε αυτά τα δεδομένα για να χαρακτηρίσουμε σε μεγάλο βαθμό το κάθε άτομο. Ωστόσο, αυτό είναι κάτι πολύ γενικό που δεν στοχεύει στην επίλυση κάποιου εμφανούς προβλήματος. Έτσι, στοχεύουμε στην ανάλυση των δεδομένων και της δραστηριότητας ενός ατόμου για την εξαγωγή πληροφοριών σχετικά με τον εργασιακό τομέα που ανήκει και τις εργασιακές του δεξιότητές.

Οι εργοδότες πρέπει να αξιολογήσουν έναν υποψήφιο για μια θέση εργασίας πριν από μια συνέντευξη. Αυτό το πρόβλημα επιλύεται εν μέρει από ιστότοπους που φιλοξενούν προφίλ χρηστών. Το προφίλ ενός χρήστη περιέχει πληροφορίες, σχετικά με τους τομείς της γνώσης που γνωρίζει, τις προηγούμενες εργασιακές θέσεις, τα διάφορα έργα που έχει ολοκληρώσει και πολλά άλλα. Ορισμένες από αυτές τις πληροφορίες είναι κρίσιμες για την αξιολόγηση ενός ατόμου και ορισμένες φορές μπορεί να είναι ψεύτικες. Η επίτευξη της καλύτερης δυνατής αξιολόγησης είναι σήμερα ένα κρίσιμο έργο. Το τμήμα των ανθρωπίνων πόρων κάθε εταιρείας προσπαθεί να το πράξει αποτελεσματικά και σύντομα. Αυτό είναι δυνατό μόνο αν διερευνήσουμε και εντοπίσουμε τα χαρακτηριστικά κάθε τομέα που χαρακτηρίζουν την ποιότητα ενός υπαλλήλου. Μετά από την εξέταση των αποτελεσμάτων της έρευνας μας θα πρέπει να κατηγοριοποιήσουμε αυτά τα χαρακτηριστικά ανά τομέα και να βρούμε έναν τρόπο να μετρήσουμε την αξία του κάθε ατόμου. Με αυτόν τον τρόπο, θα μπορέσουμε να εξάγουμε μετρήσεις για κάθε άτομο, οι οποίες θα περιγράφουν το επίπεδο και τους τομείς των ικανοτήτων / δεξιοτήτων του. Η προηγούμενη διαδικασία πρέπει, λόγω του πλήθους των πληροφοριών, να είναι αυτοματοποιημένη και ο καλύτερος τρόπος για να γίνει αυτό είναι ο Παγκόσμιος Ιστός. Μια πλατφόρμα που θα ενσωματώ- νει πολλές υπηρεσίες ιστού και άλλες πλατφόρμες, θα αναλύει τα δεδομένα και τις γνώσεις τους πάνω σε ένα άτομο και θα αξιολογεί τελικά όλα τα προηγούμενα, με στόχο την δημιουργία ενός πλήρους και στοχευμένου βιογραφικού.

1.1.2. Abstract in English

Nowadays, the large volume of unfiltered data is evident. In order to extract useful conclusions from data, it is commonly necessary to filter them out, delete useless information and present what remains in a concrete and meaningful way. Data commonly characterize people from today's society to a large extent, who come from their activity in various social media, communities and other related websites. We could analyze this data to determine the personality of each individual. However, this is a very general one that does not aim to resolve any obvious problem. Thus, we aim at analyzing a person's data and activity to the export information about the labor sector the person belongs and his work skills.

Employers need to evaluate candidates for a job prior to their usual interview. This is partially achieved by visiting sites which host their profiles. A user profile contains information, about his specialties, previous jobs, projects and much more. Some of this information is crucial to evaluate a candidate some others not while some are miss-presented and even are just fake. Achieving the best evaluation is a critical task today. The human resource department of each company is trying to achieve that, efficiently and shortly. This is only possible, by researching and finding the features of each knowledge sector, that characterize the quality of a person as an employee. After, examining the results of our research we should categorize those characteristics by sector and find a way evaluate them for each person. By doing that, we will be able to export metrics, which will describe the levelness and multitude of each candidate's abilities/skills. The previous procedure should be automated and the best way to achieve this, is the Web. A platform which will integrate many web services and platforms, analyze their data of a person and evaluate all the previous.

1.2. Evaluation in every labor Sector

The complexity in the evaluation of data and specifically the skills of a candidate is really difficult, but if we know what is notable for each labor sector the complexity of this task will be significantly reduced. Firstly, we should categorize the labor sectors by scientific field and then add the variables of evaluation in each of them. In this paper we will focus on science and its subcategories, but mostly on the software engineer-developer part.

Science has a lot of specialties, each of them focusing in different fields like physics, mathematics, biology, mechanical engineering and more. In every field of the previous the most crucial feature that a person must have is previous experience. We can understand how important experience is to find a job from classifieds of companies all over the world, and usually it is the first prerequisite. The second most important feature, for all the previous sectors, is the publications. Publications can be categorized, depending on whether it is a book, a paper published in a science journal and more. We can evaluate those publications depending on their type and their acceptance/popularity.

A book is evaluated from the readers, its content and obviously the number of sales. On the other hand a scientific paper can be evaluated firstly by the impact factor of the journal that published it and the citations number. Also, the h-index created by Jorge Hirsch of the University of California, which integrates and solves problems of conflicting metrics like the latter two. Secondly, the evaluation of a paper is based on the references by other papers, books, scientists and the article views in social media (Page Hits) [29, 46]. All the previous are carried out with success in our days by various tools and web services.

In the sector of the software engineer-developer there is an additional characteristic beyond a CV that really matters. It is generally the projects of a candidate. A person of this sector can create a very good CV by working on/completing/implementing projects that get popular, like applications, web pages, smart algorithms and software libraries. Also, the latest trend in programming is contributing to open-source projects and communities. Thus, we need to evaluate this activity of the developers by searching and analyzing the usefulness and the acceptance of their projects by other people.

1.3. Thesis Structure

In the rest of the thesis, we will use concepts like data analysis, evaluation of data and applications of the previous to solve todays problems. In the following chapters, we will use the CV of software engineer-developer as example for our analysis. We will evaluate some major services/websites that are dealing with the resume of a person, open-source projects and communities and we will refer to some similar web services, which solve the previous problem. Lastly, we will analyze deeply our proposal to this problem, describe the tool/service that was already developed and state some future prospects of it.

2 State of the Art

In this chapter we will enumerate some useful techniques/definitions of data science, that will help us understand the objective of this thesis, and the general steps that are required to build an analyzing tool.

2.1. Data Analysis and Evaluation

The process of inspecting, cleansing, transforming, and modeling data with the goal of discovering useful information, suggesting conclusions, and supporting decision-making is known as data analysis. This process has multiple facets and approaches, encompassing diverse techniques under a variety of names, in different business, science, and social science domains.

The process of data analysis Analysis refers to breaking a whole into its separate components for individual examination. Data analysis is a process for obtaining raw data and converting it into information useful for decision-making by users. Data is collected and analyzed to answer questions, test hypotheses or disprove theories [15].

Data processing The initially obtained data must be processed or organized for analysis. For instance, these may involve placing data into rows and columns in a table format (i.e., structured data) for further analysis, such as within a spreadsheet or statistical software [15, 21].

Data cleaning Once processed and organized, the data may be incomplete, contain duplicates, or contain errors. The need for data cleaning will arise from issues in the process that was used for collecting and storing data. Data cleaning is the process of preventing and correcting these errors. Common tasks include record matching, identifying inaccuracy of data, overall quality of existing data, deduplication, and column segmentation. Such data problems can also be identified through a variety of analytical techniques. For example, regarding financial information, the totals for particular variables may be compared against separately published numbers believed to be reliable [15].

Unusual amounts above or below pre-determined thresholds may also be reviewed. There are several types of data cleaning that depend on the type of data such as phone numbers, email addresses, employers etc. Quantitative data methods for out-lier detection can be used to get rid of likely incorrectly entered data. Textual data spell-checkers can be used to lessen the amount of mistyped words, but it is harder to tell if the words themselves are correct.

Evaluation of data After all the above steps, we need to evaluate which data are important for our cause. To do so, we need to know precisely the data structure and the meaning of each individual field of the data.

2.2. General steps to create a (data) analysis tool

Creating a data analysis tool, as a web platform that integrates different services, most of times is really difficult and time consuming. In order to have/end up with an effective and efficient platform, several steps should be followed. Those steps are an endless loop but a developer should start from a point [21, 40].

A good beginning could be the analysis stage. In this step, developers need to specify the data or the services they will analyze in their project. This step should be completed very carefully to ensure the target group of the platform and avert fall-backs.

The next logical step is designing. Ensuring the correctness of the analyzing stage and designing the basic tools, functions and dashboards are the most basic and important tasks in this stage. After completing all the aforementioned steps, we must be sure enough that everything is designed clearly and neatly, to avoid undesirable delays.

The next and most common step is development. This stage should be easy enough, if the previous steps have been completed with success and the results that start to arise are clear and unequivocal. The developers will follow the instructions that were defined in the design stage and materialize the idea.

Last in this loop, is the implementation step. In this step, the developers will analyze and understand in depth the outside services, which they want to implement in their platform. And again we end up where we start, in the analyzing step. In case that the results after the implementation need further fine tuning, the whole process starts over with further analysis for each incremental step of improvements or modifications that might be needed. The procedure that described above is shown as a brief graph below.

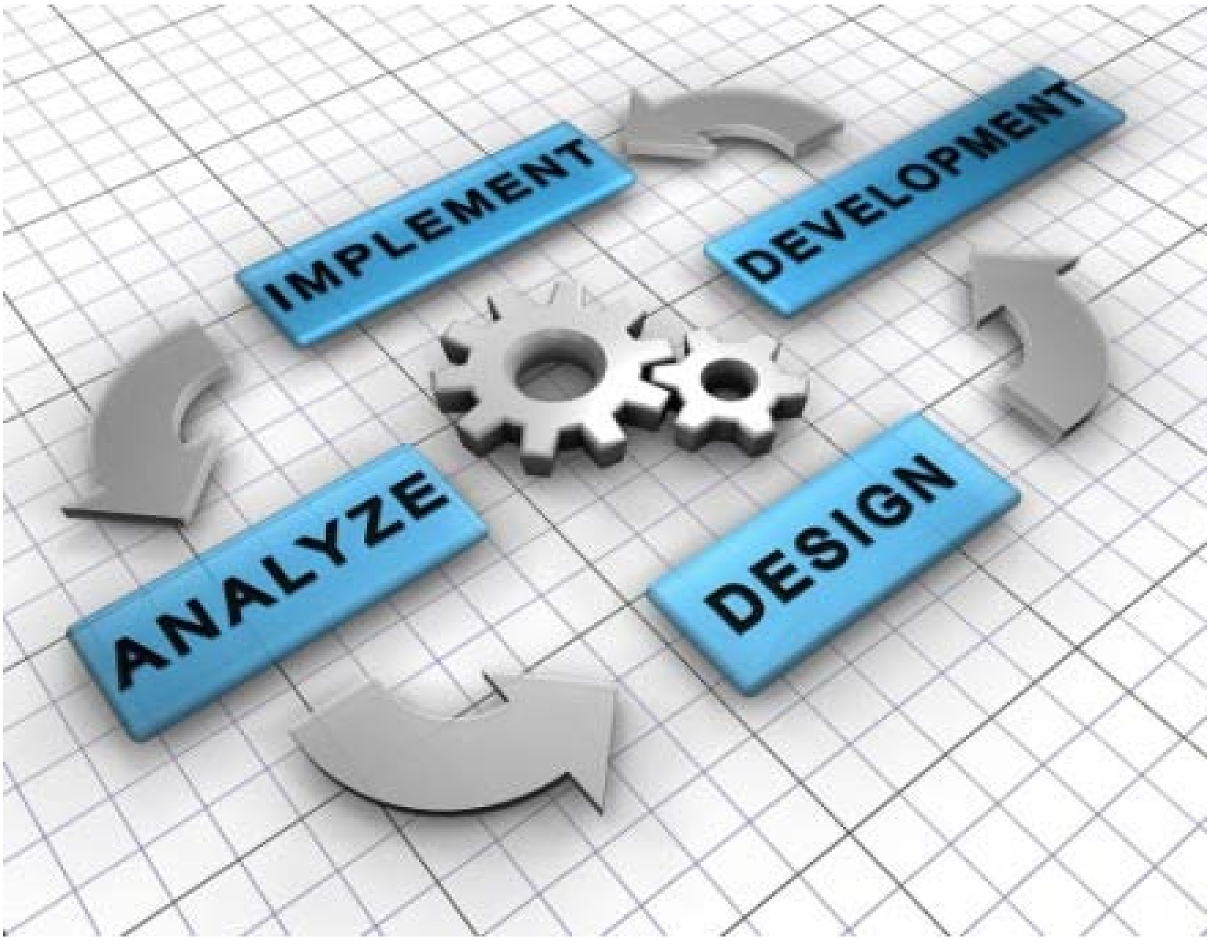


Figure 2.1: Analyzing tools (no rights reserved)

2.3. Data Analysis tools for web profiles

Analyzing a web profile or more generally a website, is currently really important because of the extraordinary results we extract. A lot of websites serve multiple data, usually raw, with no obvious meaning. Thus, analyzing tools were developed, so we can understand the meaning of such data and export the important information. Such information, is usually materialized into representative charts to make them easy to understand and explain. That information could vary, from personal information to a person's activity on social networks, web traffic and the trends of a website. Such tools are frequently used by companies for various different purposes, with the primary one being marketing.

A famous tool which analyzes and exports significant information from a facebook profile is ”fanpage karma”. Another popular tool for identification of influencers is ”Klear”, an influencer-identification platform and an analytics dashboard. You can search for influencers by skill and/or location and ”Klear” will generate 10 influencers in multiple categories. Also, a really useful tool to monitor which are the trends in the world, retrieving its data from the well known social network Twitter, is ”Twitonomy”. Another web service that integrates a lot of platforms, particularly social networks, and operates like a search engine, getting its data from publicly available records that have been indexed by Google, is ”Recruit'em” termed as ”X-Ray Search User”. This tool integrates the most popular platforms like LinkedIn, GitHub, Stack Overflow, Twitter and more. It redirects you to the user's profile in each platform which occurred by the profile given keywords.

In conclusion, the number of such tools is constantly rising and that demonstrates the need to analyze and evaluate data from such websites. In all the previous examples those services just filter the results they get from each platform - social network. But what if we could represent the information of a website better and more sufficiently by linking the activity of the user with the ”objects” that interacts with?

3

Problem Statement

Nowadays, time and money are interconnected and both of them are very valuable. Every company is trying to find the best match for their job openings. Finding the ideal candidate is usually very difficult and even impossible. As a result, the human resources management(HR) of each company has started using new technologies to solve this crucial problem.

One of the solutions to this problem are business and employment-oriented social networks that operate via websites and contain a brief resume of each user. The most popular among them is LinkedIn, which we will analyze later. This solution works to some extend but with many limitations and many times is a time-money loss factor. The above occur, because the structure of those web services ”helps/pushes” the user to create a ”strong” but partially inaccurate CV, by declaring overstated data about their profile, seeking an interview with a company. Therefore, there is a great need for this service to be ”strengthened-fixed”.

A supplement to this service could be a tool/service, which uses the data of a user as an input, such as his publicly available projects, from similar websites (like a portfolio), which are evaluated by the community and data analysis algorithms, and outputs a completely real CV of the user [28, 39, 41, 44]. This solution could also help candidates, without work experience but with a lot of contribution to open-source projects to find a job.

The above supplement fits mostly for developers, Software engineers and Computer Science engineering. Furthermore, a larger percentage of candidates belonging to the previous categories, mostly believe that their CV is their profile in websites that contain open-source projects and development communities. A website similar to the previous description is GitHub, which we will also analyze later. So, why we need to create a tool/service if GitHub exists. The reason is, that GitHub and other websites in this field, are community driven. What does this mean? There is minimal to no information about the skills, efficiency and contribution to the community of the user, but a lot of information about the open-source projects and community [18, 25, 37].

Therefore, a tool/service, which generates a CV by analyzing the a user’s data, projects, events, activity of the user and exports a CV that contains verified data about the previous, can largely help the HR departments of companies, saving resources and finding the perfect match for their current vacancies.

Also this tool could solve an additional serious and frequent problem nowadays. Developers, especially young and with no experience, do not know how to write a proper CV and present their work and skills. Consequently, our tool could also make it easier for new developers to pursue a job interview and subsequently a job. Another use case of our service is collaboration. This tool can also help developers, that seek collaborators to materialize a project, to evaluate whether someone is qualified enough to support him with his knowledge and expertise. Our tool will make the decision of collaborating with a person really easy by pointing out the quality of his expertises.

3.1. LinkedIn Analysis

LinkedIn is a business and employment-oriented social networking service that operates as a web site. Founded on December 28, 2002, and launched on May 5, 2003, it is mainly used for professional networking, including employers posting job openings and job seekers posting their Curriculum vitae, known as CV. As of 2015, most of the site's revenue came from selling access to information about its users to recruiters and sales professionals. As of September 2016, LinkedIn has more than 467 million accounts, out of which more than 106 million are active. LinkedIn allows its users (workers and employers) to create profiles and ”connect” to each other in an on-line social network which may represent real-world professional relationships. Users can invite anyone (whether a site user or not) to become a connection. The ”gated-access approach” (where contact with any professional requires either an existing relationship, or the intervention of a contact of theirs) is intended to build trust among the service's users.

LinkedIn has an Alexa Internet ranking as the 20th most popular website (October 2016). The site participated in the EU's International Safe Harbor Privacy Principles. According to the New York Times, US high school students are now creating LinkedIn profiles to include with their college applications [17]. A graph describing the architecture of LinkedIn, created from ”JavaOne” conference, is diagrammed below.

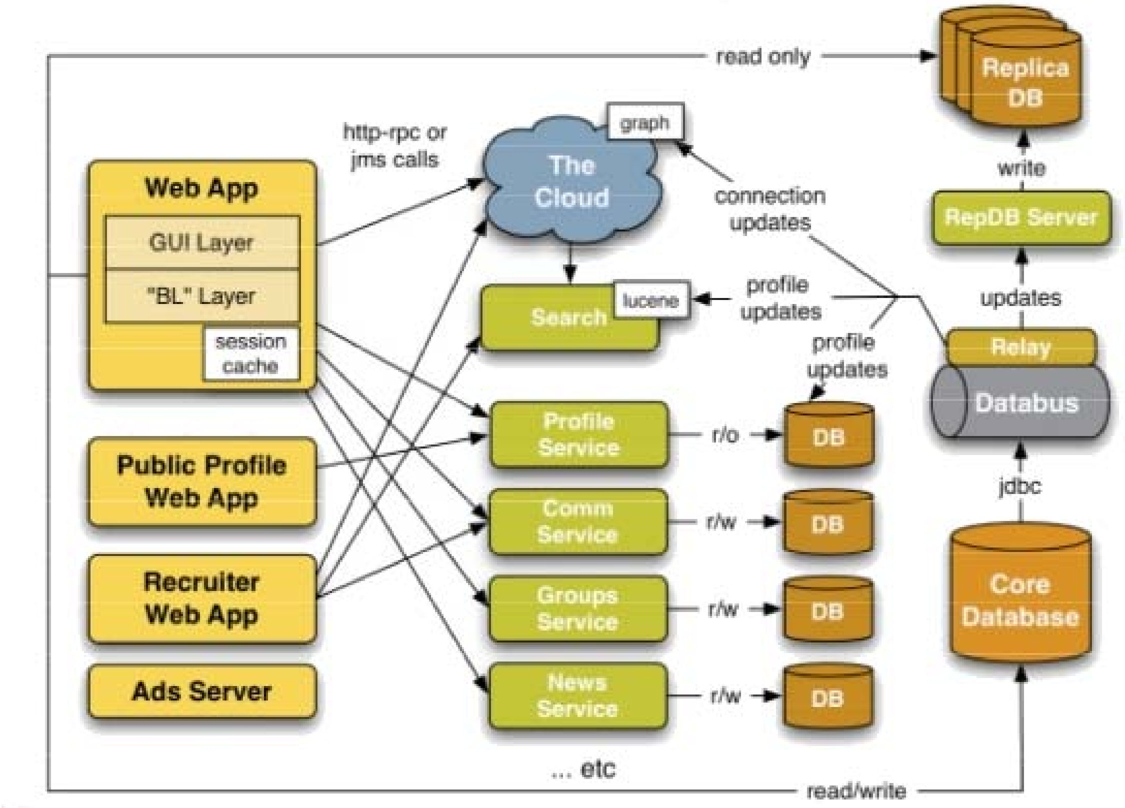


Figure 3.1: LinkedIn Architecture Scheme (no rights reserved, Scheme from 2008 JavaOne Conference)

3.2. Github Analysis

Github is a web-based repository hosting service that uses Git for version control. It offers all the distributed version control and source code management (SCM) functionality that Git provides as well as adding its own features. It provides access control and several collaboration features such as bug tracking, feature requests, task management, release milestones and ”wikis”(information about the project) for every project. Currently, it is the most popular website among several others and many developers host their projects and ideas there.

GitHub provides 18 event types, which range from new commits and fork events, to opening new tickets, commenting, and adding members to a project. The aforementioned events can be represented according to the popular property graph data model [22, 38]. A graph schema describing the types of “things” and relationships between them is diagrammed below [38].

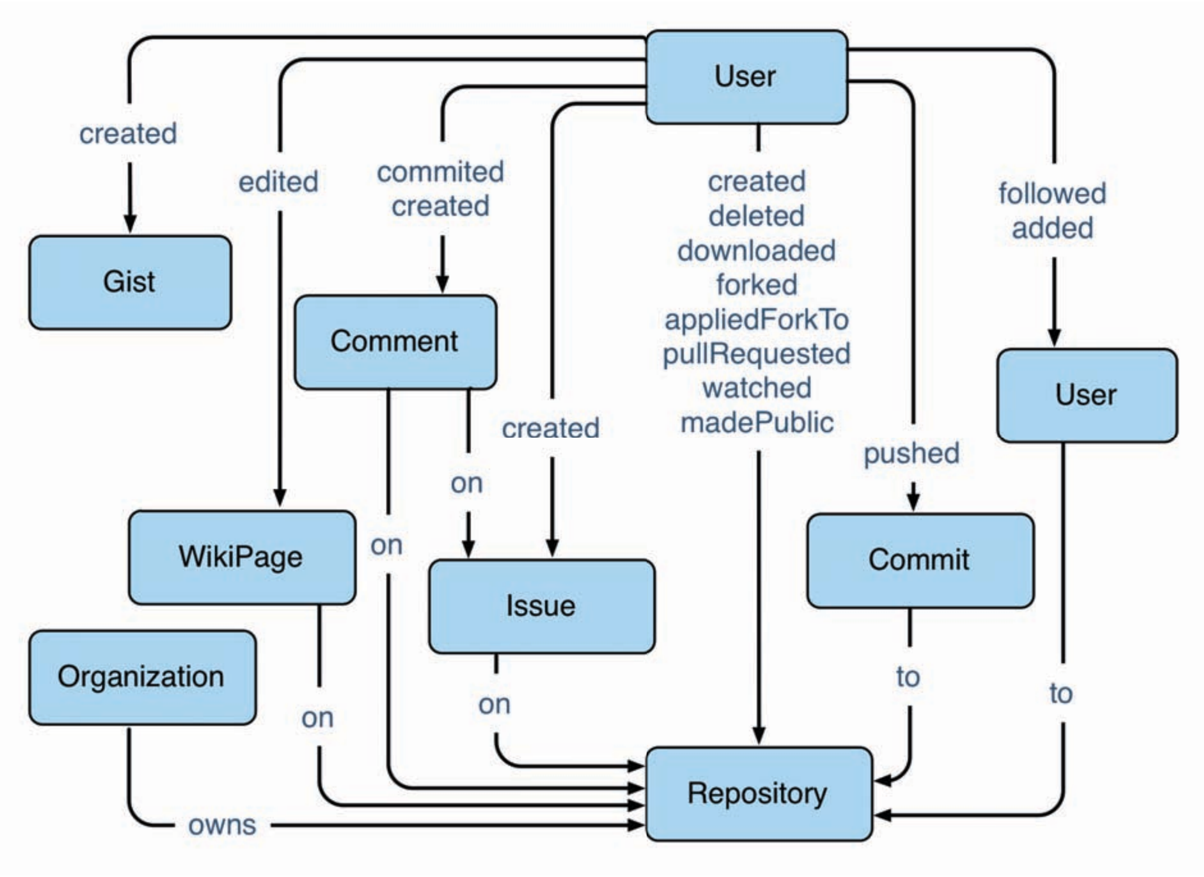


Figure 3.2: GitHub graph schema (no rights reserved)

Github is not just for developers. All this talk about how Github is ideal for programmers may have you believing, that they are the only ones who will find it useful. Although it is a lot less common, Github can actually be used for any types of files, so if you have a team that is constantly making changes to a text document, you can actually use Github as your version control system and benefit from the reviewing features on each suggested change. This practice is not common as there are better alternatives, but keep it in mind [22].

Link those Services into One. Summarizing, the target is to create a profile for a Github user that has a similar structure/format with a LinkeIn profile, in order to serve the human recourses department of companies, targeting mainly developers.

3.3. GitHub vs LinkedIn

Because engineers and designers can post their work and make it available to everyone, more and more companies are realizing they can see what people can actually do, instead of just listening them say what they can do. In the red-hot market for skilled software engineers, companies looking to make great hires are discovering that relying on traditional services that showcase candidates' work history – but not their actual work – is a great way to miss out on the best available talent [45]. So, LinkedIn in one way, might be abandoned in the future from companies that seek new talented engineers.

GitHub is a place where hiring managers and recruiters alike are increasingly turning to find not just the potential employees who look best on paper, but the ones that actively (and publicly) demonstrate their capabilities. GitHub has become a place where the hottest engineers are coming together to share their code, and as a result, the service is a home for the most important project and collaboration tools, and is also used as an application life cycle management systems in the business [32]. This assessment is shared widely throughout the tech industry. From small startups to established companies, household name powerhouses, GitHub is now seen as the go-to place to spot quality talent [45, 47].

In many companies, the feeling is that engineers who take the time to develop a GitHub profile and put in the energy to participate actively in the community can be better evaluated in advance than others. GitHub itself has been looking to its own service's community for talent, sometimes hiring people that may not present the best picture on paper, but who show off stellar programming skills in real life [19, 45, 47].

One thing that is particularly attractive to the people who work at GitHub, is that the service has become such a great way for developers to distinguish themselves, even as it got its start more as a place where people were sharing their work for no reason other than to do so [12, 31]. But that sense of selflessly participating in open-source projects is something that is increasingly attractive to hiring companies [27, 47]. GitHub is hardly the only open-source community that is being looked at by companies searching for technical talent.

4
Design and Implementation

After considering all the above, it was decided to create a platform that will integrate services like GitHub and generate a CV for any user by analyzing all his data and evaluating them accordingly. The prototype that was developed integrates the GitHub web API [22] and the NpmJs web API [7], which we will analyze below briefly and with more detail at the end of this chapter.

4.1. GitHub - NpmJs Design Analysis and Integration

As we mentioned earlier GitHub is surrounded by a community that creates and maintains open-source projects and lets anyone interested, to contribute on them. Github's web API primarily targets the projects and the events that occur on them [14]. To create our platform, the necessary and sufficient condition that are required is an open API from GitHub. Fortunately, GitHub has and maintains one of the best open APIs for developers. By using this API the materialization of our platform is achievable, because we can use "HTTP" requests to retrieve information from GitHub about any user or any public project and even more specific and complex data.

On the other hand, NpmJs is a service that makes possible, easy installations of javascript packages to a personal computer, just by installing NpmJs and using its CLI (command-line interface). Npm's web API is firstly used to check whether a project has been published as an npm package and secondly to retrieve more information about it, such as the number of times the package gets downloaded per month.

Let's now get introduced to the design of our platform in depth alongside with the problems we faced during the design and development.

Figure 4.1: GitCV icon (All rights reserved)

4.2. System Design (GitCV)

The first step for the design and development of our platform was to ensure that we will obtain the necessary data for each user. One of the first problems was the limitation in the number of requests served by the GitHub API. According to the official site of the Github API [11], requests using Basic Authentication or OAuth, are allowed up to the limit of 5,000 requests per hour. Regarding unauthenticated requests, which are associated with your IP address, the rate limit allows you to make up to 60 requests per hour.

Thus, to create a platform that can be easily used by anyone we ought to solve the problem of the requests limit, otherwise a user analyzing a big GitHub profile might end up with incomplete results because of it. This limit essentially blocks our platform from getting more information, so we can only use it to analyze a few users. Assuming, that our platform has a server and makes the requests to the GitHub API, analyzing a single user requires about ten requests on average. The service will be blocked from the GitHub API just by searching six users in one hour by anyone in the world. The idea to solve this problem partially, is to build the platform only in the client-side and use no server-side code for the API requests, so that we can take advantage of the rate limit per IP address (since each client will have different IP). An additional requirement that comes as a result of this limitation is, that the algorithm that will be developed should be optimum and make as few request as possible, so it could be executed fast enough from the client to create a user-friendly experience.

Nevertheless, another problem of the rate limit occurred. An HR employee would want to make multiple requests and check a lot of GitHub users, which would quickly exceed the aforementioned limit. Thankfully, the GitHub API provides another way to authenticate via a token. So, in case that the rate limit is exceeded, a pop-up dialog will show up and ask for a token generated from the GitHub web site that would leverage the limit to a higher value. The process is really simple, just by clicking the redirection button to GitHub, the user is requested to login and generate the token and paste it to our platform. As a result, the HR personnel now has a far more relaxed request limit, up to five thousand per hour, which is really handy in our case.

Deciding to develop only a client-side platform has some obvious problems as mentioned and some that are not so obvious. Making requests to external sites in a web platform is one of the most common things. Those requests, in some websites, may have a respond without any problem, but other websites need some headers included to the requests to respond, which also is in order to allow requests from different domains. Those requests are called ”CORS” requests [2]. If you want to make CORS requests from an application the best way to do this is from your server side code, which maintains the appropriate libraries. In our case, the platform is purely client side, which means it cannot handle CORS requests easily, so it is necessary to solve this problem. The solution is to create a intermediate server, which makes these requests, after exchanging the required headers, and returns the result of each request to our application.

Therefore, after solving those crucial problems, follow the implementation steps of our platform in detail.

Searching a profile In the home page of our platform you can search a user, by first-name, last-name or user-name and if such a profile exists in GitHub, all the possible outcomes of the given query will show up. Clicking on one of the results will trigger our GitCV formula and will gather all the needed information for the selected user.

4.2.1. Basic information of User

Initially, it will be requested from the GitHub API to provide the basic information of the user that was selected. Such information is the unique login and id of the user, the link to his avatar image so we can display it in our platform, the link to his GitHub profile and other similar links for further personal information.

But to achieve and create a fast and efficient platform it is necessary to keep only the required data. Thus, the basic information that are maintained for each user are: login, avatar link, type of user, name of user, blog of user, company, location, email, a boolean if he is hireable, the information he stated about his personal resume and lastly the number of public repositories, public gists, followers, following and the date he joined GitHub. All the previous are used either in the basic information that is presented for each user or for the next gathering data steps.

GitCV Data

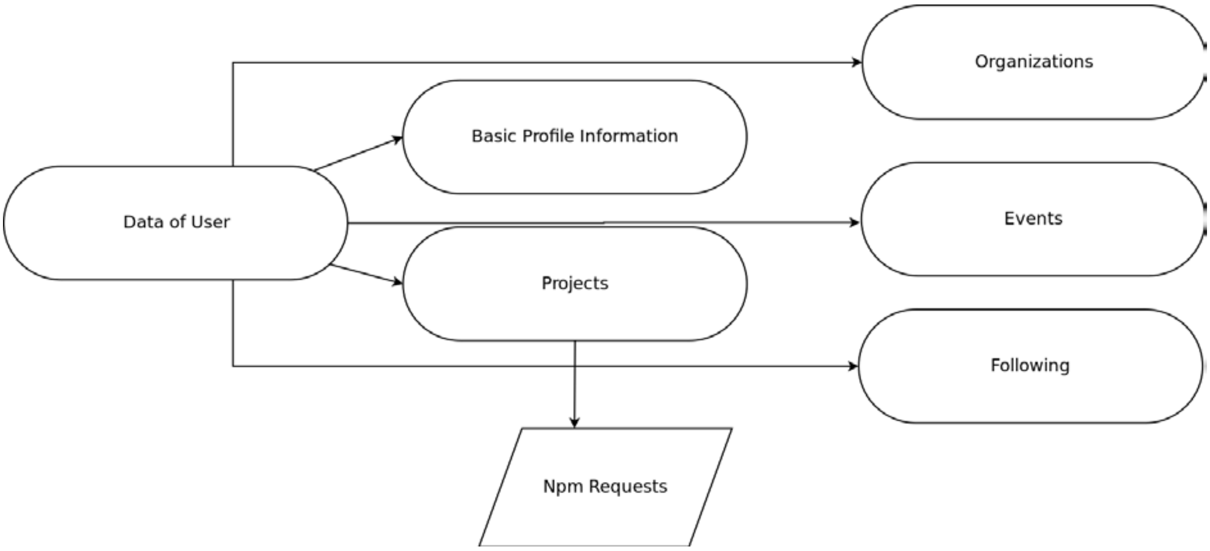


Figure 4.2: Data graph of GitCV profile (All rights reserved, created with Dia)

4.2.2. Projects (Repositories)

One of the most critical tasks is to get and evaluate the projects of the user. As stated earlier, it is great need to gather these data really fast. The first attempt, was to research the GitHub API, but it was found incomplete. By reading the GitHub API about the repositories of the user [3], it was obvious that with the proposed ”request/link” was achievable to get all the repositories of the user. Unfortunately, this was not the case. The ”request/link” that the GitHub API proposed to get the repositories, returned only the first thirty of them.

After a bit of searching it was found, that GitHub uses pagination for the repositories of each user. So, there are pages that contain by default thirty repositories of the user. Thus, it was necessary to add an extra variable in the request, like ”$page = 2$” to get the second page of the repositories and finally to get all the pages with a simple for loop. Nevertheless, it was required to minimize the number of the requests for each user, thus another more efficient implementation was followed.

Again, after searching the aspects of the GitHub API in different blogs, it was found that an extra parameter exists, which allows the developer to get with one request, a max number of one hundred repositories. The parameter syntax after this change was like ”$page = 2 + perpage = 100$”. But another problem occurred. The pagination is really strict, meaning that each page only holds thirty repositories and no more. So, if a request such as ”$page = 1 + perpage = 100$” was performed, it was returned in fact the repositories of pages 1,2,3 and 10 from page 4. If the same request was performed for page 2, it was returned repositories from pages 2,3,4 and 10 repositories from page 5. So, a simple switch case was needed to solve the problem. Subsequently, the following requests were made to get the repositories of the user depending on their number. In page one -> ninety repositories, which will return pages 1,2 and 3, in page 4 again ninety and continue likewise. But this was not the desirable case again.

By experimenting with these requests, it was found that the second parameter about the number of the repositories was not working as expected. Initially it was requests from page one, ninety repositories and it worked, next from page four, ninety repositories, but this request returned only forty. Solving this unexpected problem was really time consuming and the only way was testing again and again. After, a lot of time the following formula of requests was constructed: 90 repositories - page=1 (page 1-3), 30 repositories for each next page until the number of repositories of the user. So, if the user has 210 repositories five requests are required to get them, one of ninety repositories and four of thirty. The reason that a request of 100 repositories was not performed was that, because of the structure of the pagination it was not achievable to get from one page only the last twenty repositories. But it was used the request that returns the first one hundred repositories of the user by applying it on users that don't have more than one hundred repositories. In this case with one request, it was accomplishable to get all the projects of a user.

After solving this problem, the next step is to evaluate the projects of the user, but to do that it is necessary to synchronize the asynchronous ”http” requests that are performed. This can be achieved by using Observables [8], which briefly is a way to sync and wait all the data from external sources, so they can be filtered as a total.

Filtering It is obvious the need of filtering, sorting and separating the repositories into categories depending on the type of each repository. Every repository in GitHub has some attributes like id, name, owner, description, stars, watchers and more. A basic metric to sort these repositories is the stars each one has. Those stars can be given by other users to point out that this repository is useful and does what it describes. Thus, the repositories can be sorted by stars and be presented in our platform for the selected user. Looking into a profile on the official page of GitHub, it is noticeable that the repositories of the user are presented in one category all together.

The starting point is the indication of the categories that a repository can belong. Initially, there is the category of ”owned” repositories . Next, are the ”forked” repositories, these repositories belong to a different user but any user can ”fork” a repository which doesn't own. Briefly, a repository can be added to the profile of a user (forked), so the user can use it later, or contribute to this project by making changes in this ”forked” project. Then he ca merge it with the original one so that other users can reclaim and incorporate his changes-additions-fixes. Later, in this chapter it is stated the way to distinguish the repositories that the user has contributed. Lastly, are the ”member” repositories that a user has been assigned on them as a collaborator, by the maintainer of the project, and has the same permissions on this project as the last one. Member projects can only be retrieved if another parameter is used in the ”http” requests, which is about the type of the repositories. By default the type is ”owner”, but it is possible to obtain these repositories just by changing the type parameter from ”owner” to ”member”. Concluding, there are the above three categories, ”owned”, ”member” and ”forked”. In the GitHub profile are presented the repositories of the categories ”owned” and ”forked” together, without any separation.

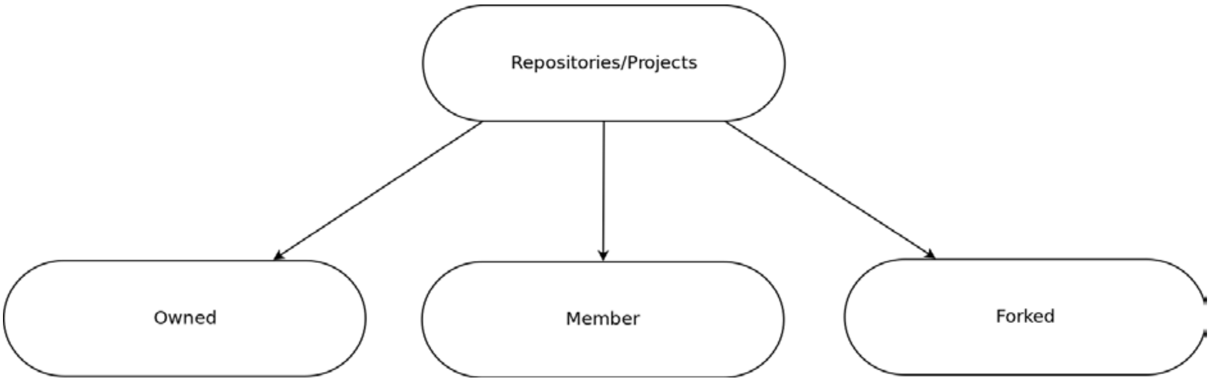


Figure 4.3: Types of GitHub repositories/projects (All rights reserved, created with Dia)

As mentioned earlier, those repositories will be sorted, in each category, by their stars. Although, after consideration a reflection came up about the last metric. The fact is that only the stars for each project are not a good metric to distinguish which is better. A simple scenario is that not all users will star a project they used, even if it was the best they ever used and was really helpful, or they will simply forget to do it. Thus, a second metric was required to characterize better every project and in the end the user himself. The monthly downloads of a repository is a good metric, because if someone downloads a project then it is helpful, but this was achievable with the GitHub API until the end of year 2012 [4]. (The watchers of each repository is not used as a second metric because if a user stars a project he will ”watch” it by default). So, the integration of NpmJs [7] to get the monthly downloads of projects that are published in this site was the solution. NpmJs as mentioned is one of the best and most popular tools to install web projects and libraries and it is used extensively all over the world. After syncing all the requests to get the projects, two requests are performed for each repository at NpmJs, firstly to verify if the project belongs to this user and secondly to get the monthly downloads of the project, if available. After all the previous it is easy to distinguish if a project is good or not and in the end of the day, evaluate the user.

All the previous details are summarized in the profile of the user with a simple and unequivocal way. The stars the user has collected from the projects he owns are summed up and stated at the top of his GitCV profile. Also the downloads of his projects are stated in the same way. After this basic card, which contains the basic information of the user, stars, downloads of owned projects and number of followers, it is generated a chart that represents the programming languages that the user has used in his projects. This information is exported from all owned projects and the ones that the user has contributed. In the next card, there is a distinctly separation and presentation of the number of the repositories, the stars and the downloads that the user has collected for each category of his repositories-projects. All the above information is exported just from the projects. It is possible to navigate further into a project and read more details about it or redirect to his official page on GitHub.

A Collaborator has the same permissions with the maintainer of the project

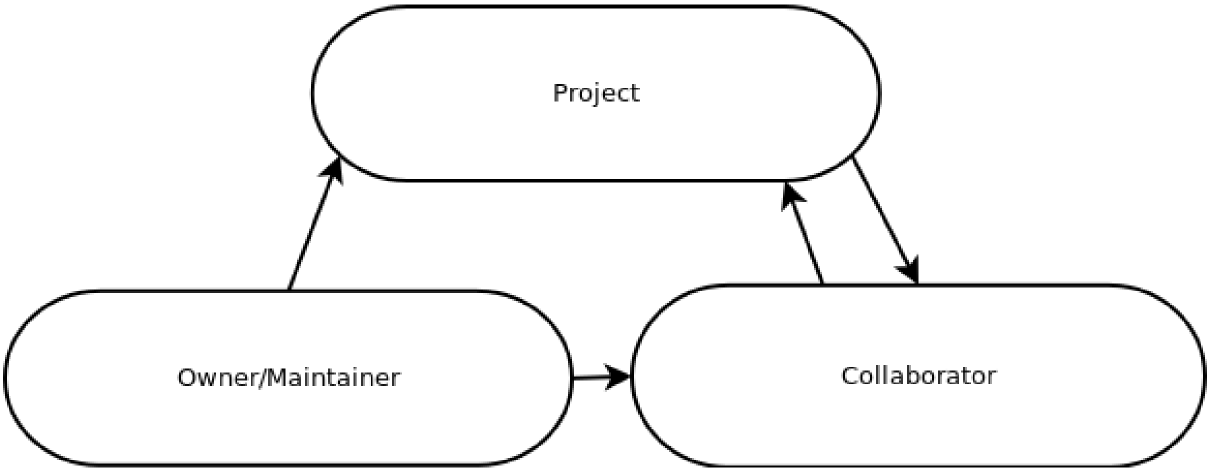


Figure 4.4: Owner - Collaborator Graph (All rights reserved, created with Dia)

Gists Gists are a great way to share your work. You can share single files, parts of files, or full applications. They are usually used to save and share algorithms that are independent and can be implemented by nearly everyone. In the GitHub profile of a user there is no redirection button to check his gists. Any information about the gists partition of the GitHub can be found on their help page [23], where they inform the user that the gists can be accessed by the link ”https://gist.github.com/”. In fact, if you want to see the gists of a user, you must access the link ”https://gist.github.com/username”. Thus, it was materialized a page in our platform, where anyone can check all the gists of the selected user and also there is a redirect button to the official page of the gists. A chart is generated, showing the programming languages that were used to develop those gists. The requests that need to be performed to GitHub API are as much as the gists of the user, to wit one request for each gist. For that reason the gists are retrieved if requested by accessing the particular page in order to minimize the requests for each user.

Organizations of the user Organizations are shared accounts where groups of people can collaborate across many projects at once. Owners and administrators can manage member access to the organization’s data and projects with sophisticated security and administrative features [24]. The organizations that a user belong, are included to his GitCV profile along with several information about each organization. It is possible to search them in the platform by changing a few things in the evaluation system, but this will be further analyzed later.

Following Like all social networks, in GitHub a user a can follow other users and be informed about their progress and their new projects or their actions in GitHub generally. The following people are represented for each GitCV profile to link the GitCV profiles and make easier the switch to another user.

GitCV for Organizations In the platform it is possible to search and evaluate Organizations. The only information that is retrieved is about the repositories/projects for each organization and the stars, as well as their downloads from NpmJs. Thus, there are performed extremely less requests, because the organizations do not have all the previous activities that were described like the users have, but also because the platform’s target is the users.

4.2.3. Contribution

One of the most important actions of the user in communities like GitHub is the contribution to this community. As stated earlier, GitHub does not supply any information about the contribution actions of each user, anywhere in his profile. So, if the user is contributing, like a lot, in a library of Google or Mozilla, it is really important for a company to be aware of it [32, 42, 43]. Thus, by analyzing the structure of GitHub and using his API efficiently, it was possible to represent the contribution of each user.

Firstly, as shown to the graph below, it is necessary to navigate from the top layer of the user activity to his events [35]. All the other categories do not supply any information about his contribution activity.

Activity

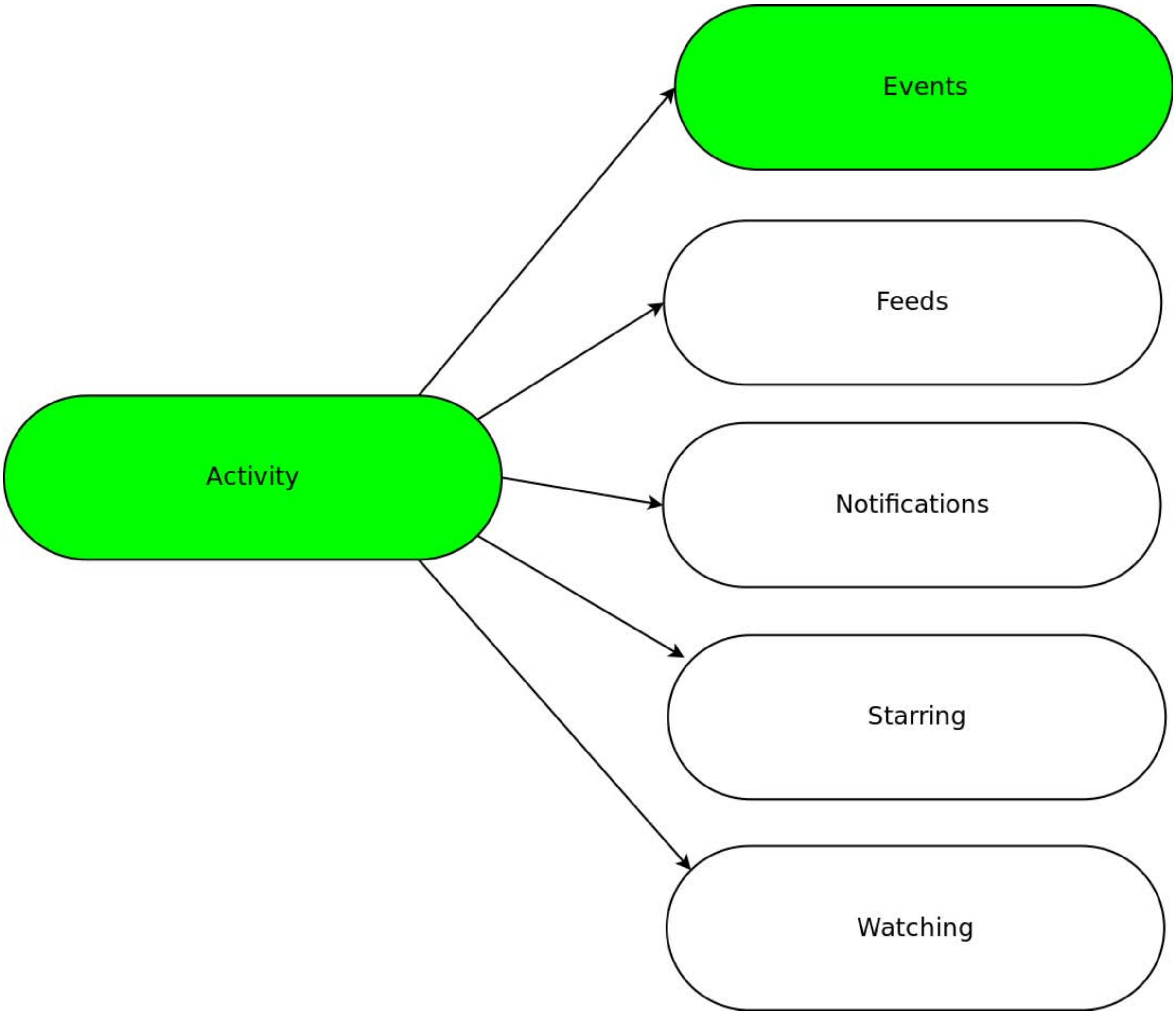


Figure 4.5: Activity Graph (All rights reserved, created with Dia)

Events The bext step is to filter the events of the user, and discard anything that does not apply to the platform's use. So, by the below graph it is only needed the "public user events (received)" to export the contribution. Unfortunately, the GitHub API provide only the last 300 events for the user in a period of one and a half month. Thus, it is not possible to get all the contribution of the user just by using the GitHub API. Events have a lot of types such as commit, issue, star, push and more. The needed types are "PushEvent" and "PullRequestEvent". This is because if the user has ever contributed to an outer project, then he necessarily made a push to this project. It is needed to check that this "PushEvent" is not about an owned repository. About the second type, if the user has made a pull request, that means he has forked the project and he is developing the code on the repository on his profile, in a new branch. Then, when the development is done, he will make a pull request to the original project. If the pull request is merged then this project will be added to his contribution page. The navigation to this page in the GitCV is possible by the basic button. It is worth mentioning that with this approach is not possible to get all the contribution the user has ever made, because the GitHub API restricts the data to a period of ninety days.

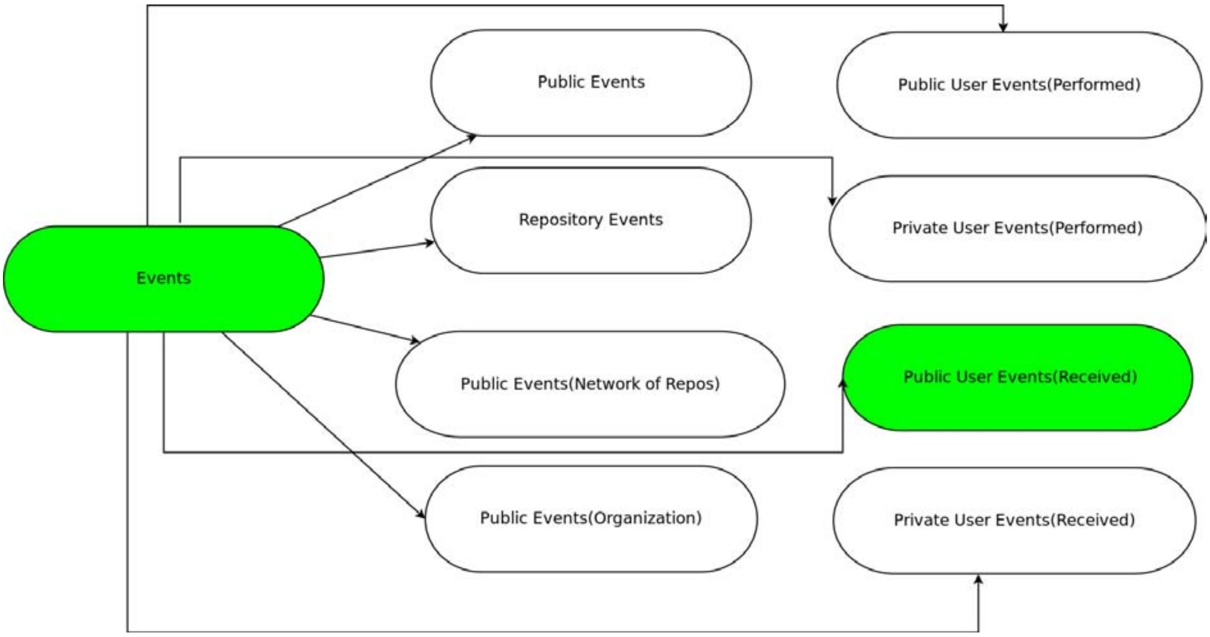

Events
Public Events
Repository Events
Public Events(Network of Repos)
Public Events(Organization)
Public User Events(Performed)
Private User Events(Performed)
Public User Events(Received)
Private User Events(Received)

Figure 4.6: Events Graph (All rights reserved, created with Dia)

Search Commits Another possible way to find the projects that the user has contributed, is to search his commits. The GitHub API provides the function to search the commits of the user and keep those that has as target forked or unowned projects. The problem of this method is that you cannot check by making those requests, if the commits have been merged to those projects. This means that the needed information to add those projects in the contribution list of the user is not provided. Also, the number of the total requests would have been increased a lot. Thus, this approach is not used.

Forked Projects The projects that the user has contributed can be found by checking the events history of his forked projects. If any commits are created in his forked projects, then it is possible to search further by checking if he made a pull request which was merged by the maintainer of the project. Then, this project can be added to the list of the projects that the user has contributed. The above way of getting the contribution is much less efficient than checking the events as described previous, thus it was not materialized and used.

4.2.4. NpmJs Downloads

As stated previously, NpmJs is used to get the downloads of published projects but some problems were emerged. More particularly, NpmJs has no official site that presents and analyzes his API. Meaning that the NpmJs API officially does not exist. Through experiments and time loss, communities found out that there is an API for NpmJS. Thus, to integrate this API further research was necessary over the requests that should have performed. The result aimed to characterize better each project of the user. No library was used to make those requests, but the raw ”http” links were used directly to get the desirable results.

Lets continue with the procedure to get the project's downloads. It is worth mentioning that the most published projects in NpmJs carried out through GitHub. There is a function in NpmJs that permits the user to publish the GitHub project. Any published GitHub project on NpmJs has a ”package.json” file, which contains the required information about the publication of this project. Thus, this file was downloaded for every project and was used to search in NpmJs with the name of the package. This approach is not a good solution because it is needed one extra request for each project of the user to get this file, which will result the limit excess of the GitHub API requests.

Thus, another approach was adopted to solve this problem. The solution is to get the name of the GitHub project and make one first request to NpmJs to find if the project is published. It is also necessary to transform the name of the project to lowercase because NpmJs is case-sensitive and only uses lowercase names for the projects. If the project exists, the next step is to authenticate if it is published from the selected GitHub user. This is achieved by checking, if the home page of the GitHub project is the same with the one of the NpmJs project. All this info is retrieved from the first request to NpmJs for the project. But if the project does not exist in NpmJs, there are two possibilities. Either the project indeed does not exist, or it is published with a different NpmJs name. So, the next step to really verify if the project is not published, is to check if there is a ”package.json” file uploaded in the GitHub project. If there is, we need to make a GitHub request to download it. Next, the previous procedure is repeated with the name from the ”package.json” file and by checking again if the published project is owned from the specified GitHub user. After the authentication is successful, the second request is performed to get the downloads for this project.

Another possibility is that the project was previously owned from the user, but now has been migrated to another user or to an organization but it is still published with the old info about the owner. This means that this project cannot be found in the owned GitHub projects of the user but in the member projects list. To find out the previous case, it is necessary to get the ”package.json” file of this GitHub project and then check if the request redirected to another link. This means that the project has been migrated, possibly to an organization. Thus, a crosscheck between the old home page of the GitHub project and the one of the published Npm project is required. This is due to that the previous owner did not change the ownership of the NpmJs project, but he migrated his project to an organization. If everything went as expected, it is achievable to get the downloads for this project and add them to the project on his member projects list on GitCV. On the other hand, it may be useful to display the downloads of this project in the GitCV profile of the new owner, even if the previous owner mistakenly did not change the info in the ”package.json” file. To do so the first request must be performed, as previously, to get the basic info of the published project from NpmJs. Then, the home page of the published project must be accessed, which in this case is different from the GitHub project, and check if it redirects us to another user/organization. If so, there is the need for an equivalence comparison between the redirected user/organization and the user that now owns this project. The last procedure is represented below with a diagram.

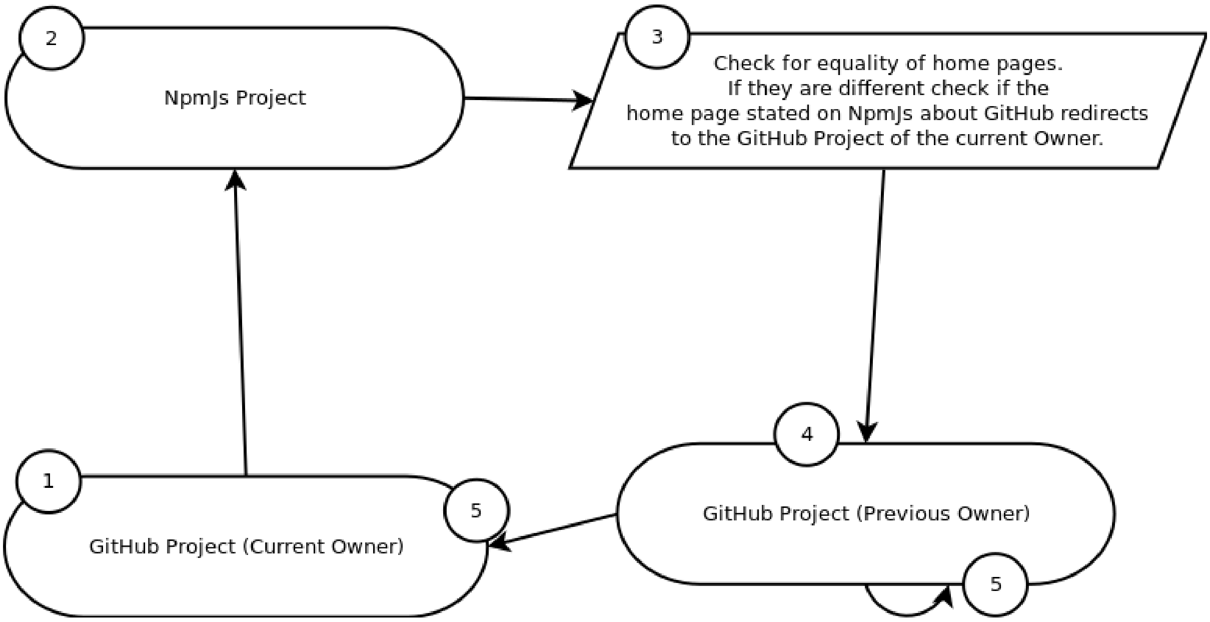

2
NpmJs Project
3
Check for equality of home pages.
If they are different check if the
home page stated on NpmJs about GitHub redirects
to the GitHub Project of the current Owner.
4
GitHub Project (Previous Owner)
5
1
GitHub Project (Current Owner)
5

Figure 4.7: NpmJs, Get downloads procedure (All rights reserved, created with Dia)

4.2.5. A website for all devices

The website must be accessible and easy to use from any device. After taking into consideration all the devices that the user has access at a web browser and that in each device, different resolution and screen dimensions are applied, the creation of a global website seems ideal, so that the service would be widely accepted. The solution is CSS3. One of the three basic elements of all websites, gives the developer potentiality of changes and creation of global rules for all devices. Thus, one template was created, that adapts to the screen dimensions of all devices.

Another aspect of this problem is the need of the users to use mobile applications. It is commonly accepted that a person will use far more an application on his portable device rather than a website. Thus, by using the function ”Add to Homescreen” developed by Google Chrome [16], which is just a bookmark added to the home-screen of the device, and by setting up our website with the proper application icons for each device, an application user experience was achieved, even that GitCV is a website. The application/website is opening as an application installed on the device but it is a website served from Google Chrome in full screen mode. In conclusion, with the above the portability and the user friendly experience was achieved for any device.

4.3. System Implementation

By searching and finding the needs of the market, the decision to develop our platform, a new innovative tool/service, with new tools to get the best of the Web was radical. So, it was used the Ionic 2 [6], a hyper-framework which purpose is to create applications for mobiles and desktops with source code web based languages. Ionic 2, currently in development, uses AngularJs 2 [1], one of the best and well known front-end frameworks for web development. The idea of using new technologies, that are under development, is focusing to implement new features to our platform, update constantly and easily to more stable and newer versions. Also, new open-source platforms can be helped, by using them, to be better and evolve much faster with the support of the community. As mentioned before, the platform maintains only a client-side and no server-side to achieve an efficient and fast service for every user, without spending money to outer servers.

4.4. Similar web applications

Creating a CV for a Github profile, in the format of LinkedIn, with all the needed information has never been done this way. Similar web applications focused in particular data, such as the contribution of the user, or the total activity of the user, are very frequent and we will analyze some of them.

Open-source Contributors (Total Contribution of User) This application was created by Tenex Developers [10] and represents the total contribution of the requested user since 2011-01-01, as and all of his events. This project could not be implemented, because we do not have a server-side in our platform. Nevertheless, the code of this project is free and available for everyone at GitHub. This platform retrieves its data from Google's BigQuery and GitHub Archive. Because of the size of the data, it is obligatory to wait a bit for the result, as it was found out from the website of the platform. Thus, we could not implement this tactic for the contribution part because of the low performance. A reference to this project is mentioned here for further info [9].

Git Inspective (Events of User) Likewise, this platform is a lot simpler than the previous and represents, in a tree structure, the events of a GitHub user. In this implementation the period of time of the events is restricted to ninety days, because of the GitHub API, like in GitCV. No more analysis has been made to those events, as shown in the website which can be found here [33].

GitHub Profile This platform's target has the most in common with our platform, from all the other web applications that was found. It was developed in 2012 and it is published at GitHub. Currently, this site is not anymore served anywhere, so only by looking an article [26], where also can be found the link to the GitHub project, some details came up about it. The project is a representation of the GitHub profile with a chart about the languages of the projects that the user has developed and a list of his public projects. Also, no metrics are presented for the user, so that an outer user or company could comprehend the level of the developer.

Other applications There are a lot of android applications that integrate GitHub, such as ”Gito for GitHub” and ”Top Github”, which can be found in android store, but their target/purpose is to demonstrate the new and most popular repositories/projects and not the user at all. Thus, those applications can not even be compared with GitCV because of the different target and cause of each platform.

Conclusion None of the previous platforms integrates all the necessary components like GitCV, so that a full CV of a GitHub user can be presented, that includes a lot of details about him and his projects. Thus, it was decided to design and develop GitCV.

4.5. Time Consuming Problems

In this section will be mentioned the basic points through the development of the platform that were time consuming and possibly dead ends.

4.5.1. APIs

As stated in the previous chapters working with APIs, is really stressful because some times anything you will try will not work and it will not be your fault. Usually APIs have bad documentation, which is really no help for a developer and even more time consuming. Even the GitHub API, which has one of the best documentation has a lot of ”gaps”. The missing information sometimes may be really necessary to solve a problem and be crucial to understand the structure of each service. The ”gaps” about the GitHub API were mentioned on previous chapters. The second API that was used is the NpmJs API. It was mentioned earlier that this service officially has no API which means it does not exist. Thus, it was needed further research into blogs and communities and also endless tests on these requests to verify their validity and performance.

Another problem that emerged and really delayed the development of the project was the limit of requests in GitHub API. It was mentioned to the previous chapters that this problem was solved by adding a functionality in our platform, so that the user can authenticate with a GitHub account and make more requests. Even so, it was necessary to achieve a number of pretty much ten GitCV users without asking the user to authenticate, so that the platform would be user friendly. To do that, it was obligatory to select carefully which information was needed from the GitHub to create a CV for each GitHub profile. Thus, the minimization of the requests, as referred previously, was really a time consuming task. Also, all the requests were performed from the client-side, to take advantage of the limit of the sixty requests per hour for each IP of the GitHub API. So, the platform was developed without a server-side. This was the cause of even more problems.

Firstly, seems that it is not such a bad idea to have only a client-side in your website. But a lot of things that were necessary were materialized for servers and not clients. The first problem that surfaced, was that some ”http” requests are CORS requests [2], as mentioned earlier. These requests are not achievable with a client. The solution was stated at section 4.2. Nevertheless, finding an intermediate server to continue the development was crucial. So, once again searching and testing was the tool to find the solution but also loss of time.

4.5.2. Progressive Web Apps and more

As mentioned in section 4.2.5, there was a imperative need to develop the platform in order to be responsive and easy to use in any device of any resolution and screen dimensions. By adopting, the way of Google and creating the required icons, services and a manifest file to create a progressive web app (PWA) [20], as also using some html5 apis, this was achieved. But once again it was necessary to stick with the schedule, even though the tests for this functionality were really endless.

In the System Implementation chapter it was mentioned that it was used a new framework, which is still under development, Ionic 2 that integrates AngularJs 2. It is really good to use new technologies and techniques on web development but the most of the times it is a lot of time consuming. So, it was decided to use, but also to learn a new open-source framework, since it had only been created three months ago. Bugs of the framework came up during the development, so it was needed to wait for patches that would fix those bugs, even contribute to the framework to make faster the updates. When you need to use new technologies and features there is usually a price and this is time. Nevertheless, it was worth it.

Another problem in every web framework is the dynamic creation of elements, depending on the extracted data. The site has to be formed, so that it can catch all the extreme case scenarios and export a satisfying result for each user. This problem exists almost everywhere nowadays and it may be the biggest problem in the development of such platforms.

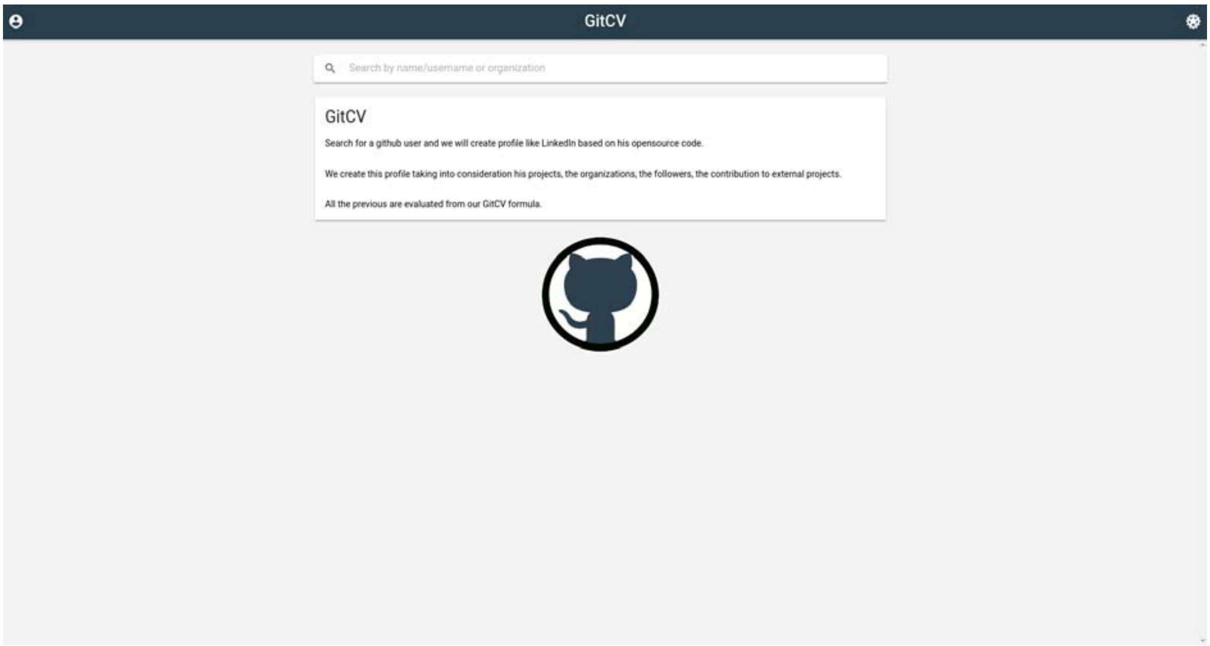
Lastly, the design of the website should be appealing to use. This is something that a platform may not achieve for all of his users, due to the different taste of the people. Thus, it is inevitable to search and get ideas from many platforms, check for appealing and beautiful colors to use and create a beautiful logo. The design really is a mess because you cannot satisfy everyone. A conservative but beautiful design was adopted for our site that would sting and raise the interest of most people. In the next section is presented a part of the design.

4.6. User flow demonstration

By entering in the website some basic information about the platform can be seen, a search-bar where we can search by name, full or not, or by GitHub user-name, and two buttons on the top corners. The left one contains some info about the developer and the framework that was used to develop this platform and from the right one you can authenticate by redirecting to GitHub, so you can make more requests. Some screen-shots are following from a computer and a mobile device.

4.6.1. Screen-shots from various use cases

Home Page



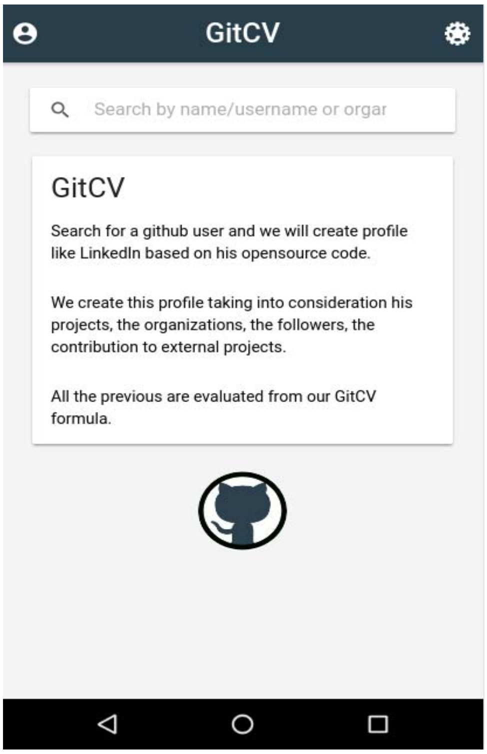
GitCV

Search for a github user and we will create profile like LinkedIn based on his opensource code.

We create this profile taking into consideration his projects, the organizations, the followers, the contribution to external projects.

All the previous are evaluated from our GitCV formula.

Figure 4.8: Computer screen-shot of www.gitcv.com



GitCV

Search for a github user and we will create profile like LinkedIn based on his opensource code.

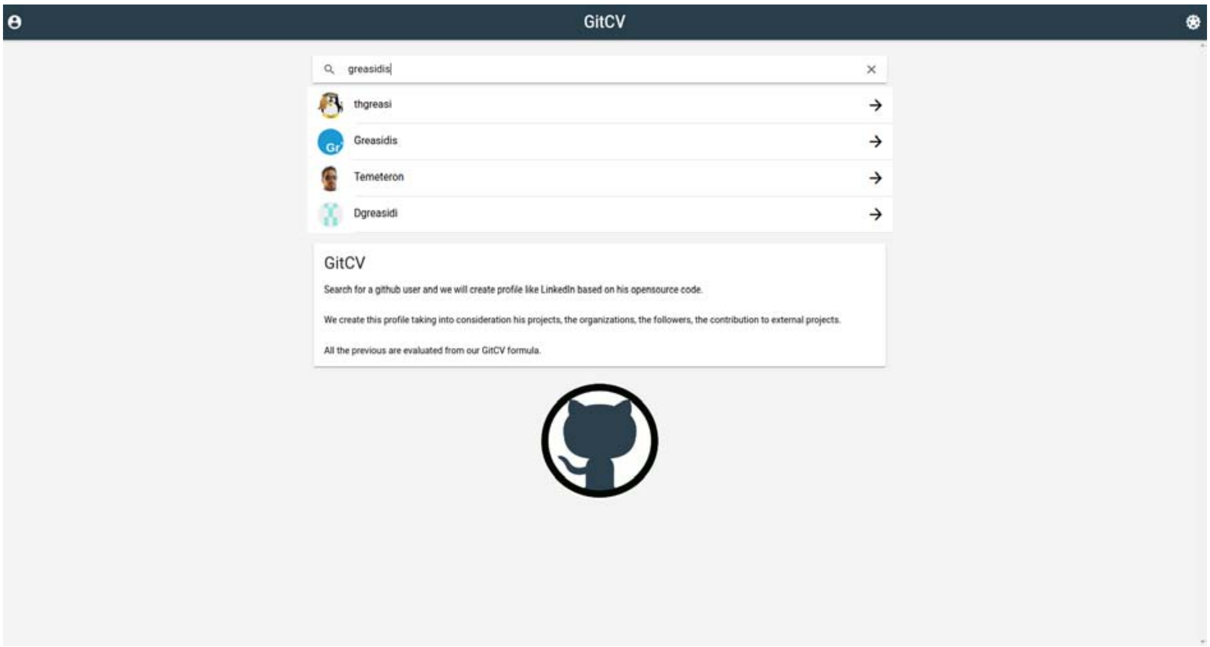
We create this profile taking into consideration his projects, the organizations, the followers, the contribution to external projects.

All the previous are evaluated from our GitCV formula.

Figure 4.9: Mobile screen-shot of www.gitcv.com

If you search with a name, let’s say ”greasidis”, and press enter a list will show up with GitHub profiles that contains the previous query in their name.

List of Profiles



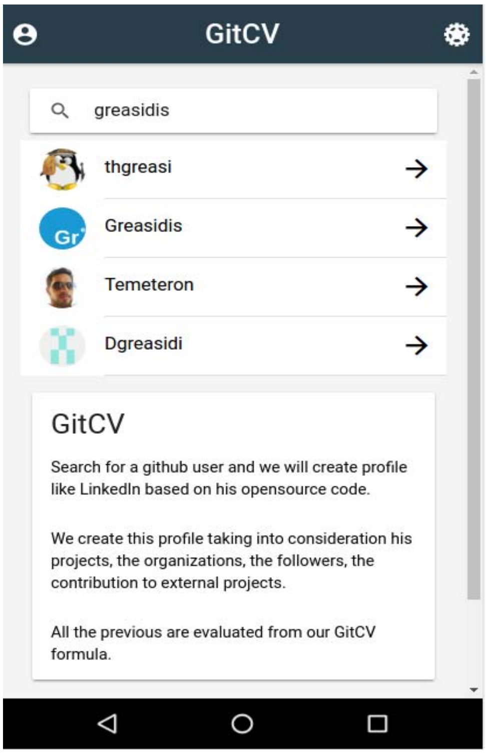
GitCV

Search for a github user and we will create profile like LinkedIn based on his opensource code.

We create this profile taking into consideration his projects, the organizations, the followers, the contribution to external projects.

All the previous are evaluated from our GitCV formula.

Figure 4.10: Computer screen-shot of www.gitcv.com



GitCV

Search for a github user and we will create profile like LinkedIn based on his opensource code.

We create this profile taking into consideration his projects, the organizations, the followers, the contribution to external projects.

All the previous are evaluated from our GitCV formula.

Figure 4.11: Mobile screen-shot of www.gitcv.com

By clicking one of them you will go to the details page of the user. Here you can see the total stars and downloads the user has achieved through his owned projects, which is a good way to decide if the user is good enough or not.

GitCV Profile

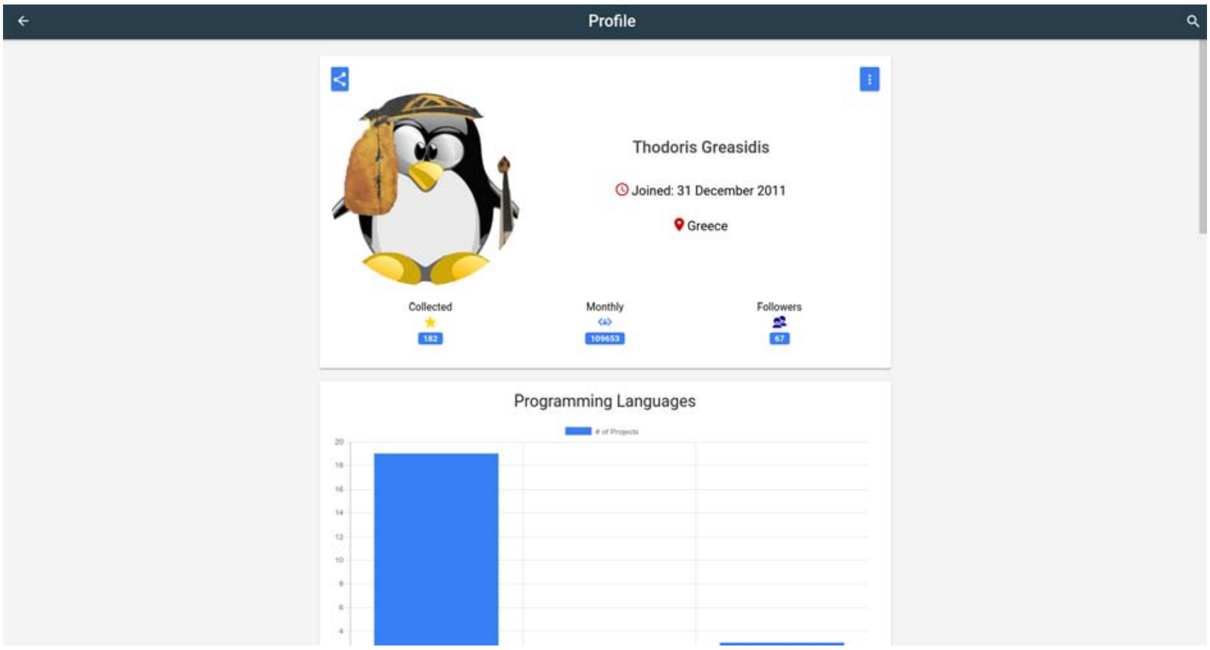


Figure 4.12: Computer screen-shot of www.gitcv.com

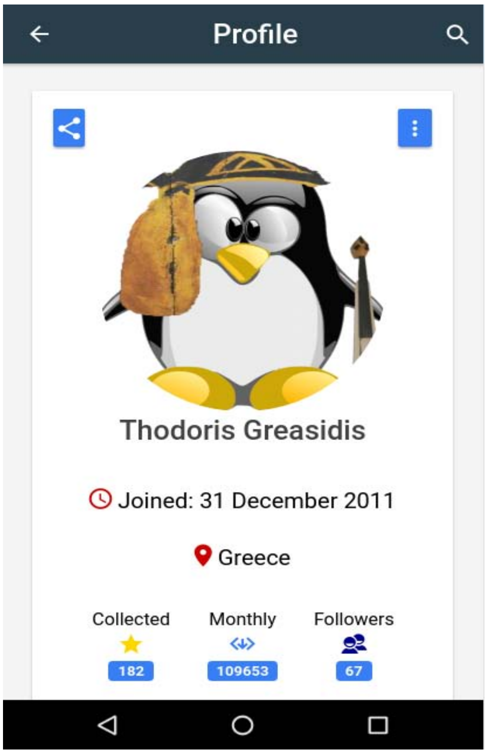
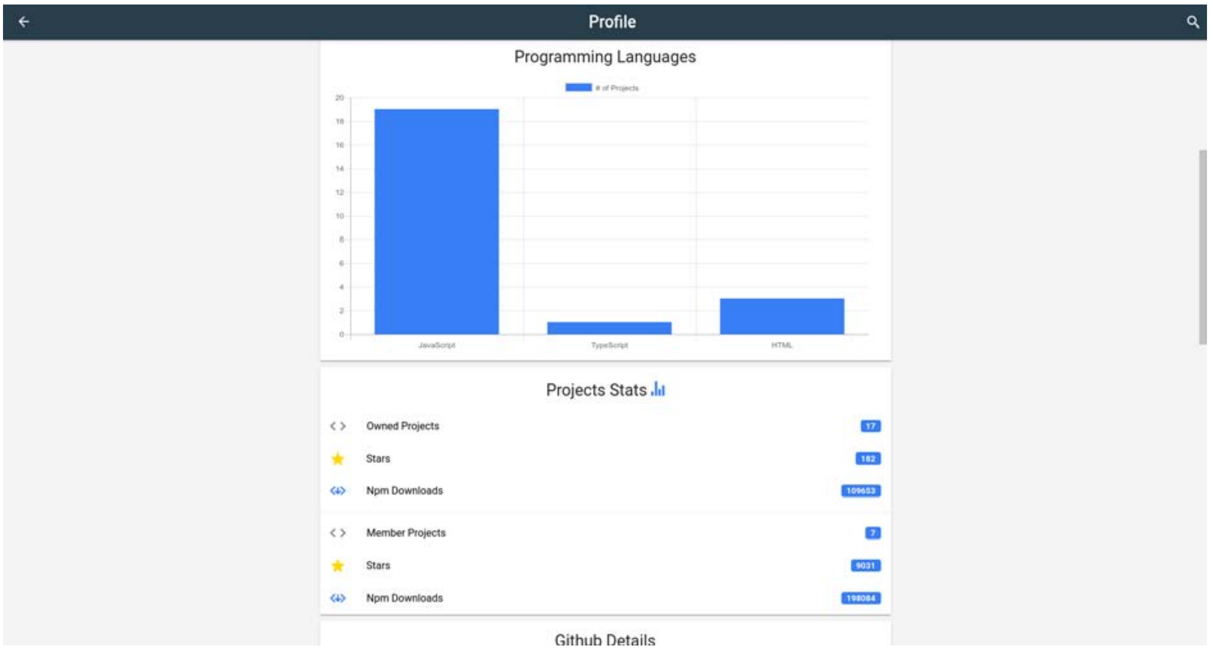


Figure 4.13: Mobile screen-shot of www.gitcv.com

And by scrolling/sliding you can see more details about the user.

Graph and Details



Programming Languages

Projects Stats

Owned Projects 17

Stars 182

Npm Downloads 109653

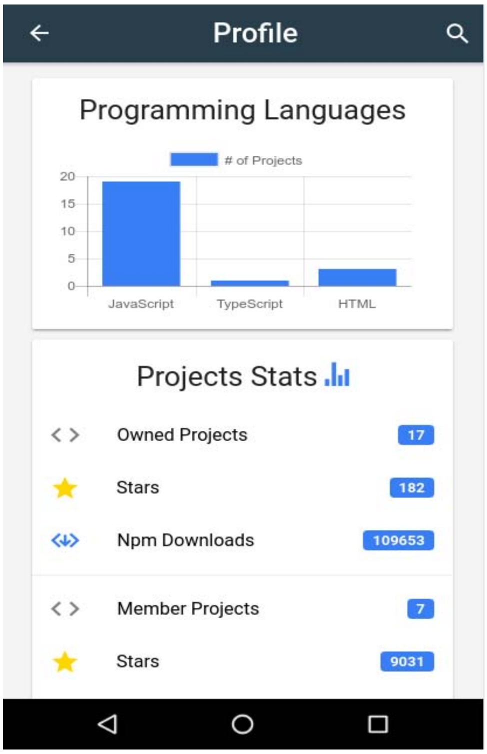
Member Projects 7

Stars 9031

Npm Downloads 198084

Github Details

Figure 4.14: Computer screen-shot of www.gitcv.com



Programming Languages

Projects Stats

Owned Projects 17

Stars 182

Npm Downloads 109653

Member Projects 7

Stars 9031

Figure 4.15: Mobile screen-shot of www.gitcv.com

Clicking the more info button on the right of the profile image of the user will show a dialog from where you can navigate to pages for more information about the user. By clicking the projects option you are redirected to the page that contains all the projects of the user and their details.

Projects Lists

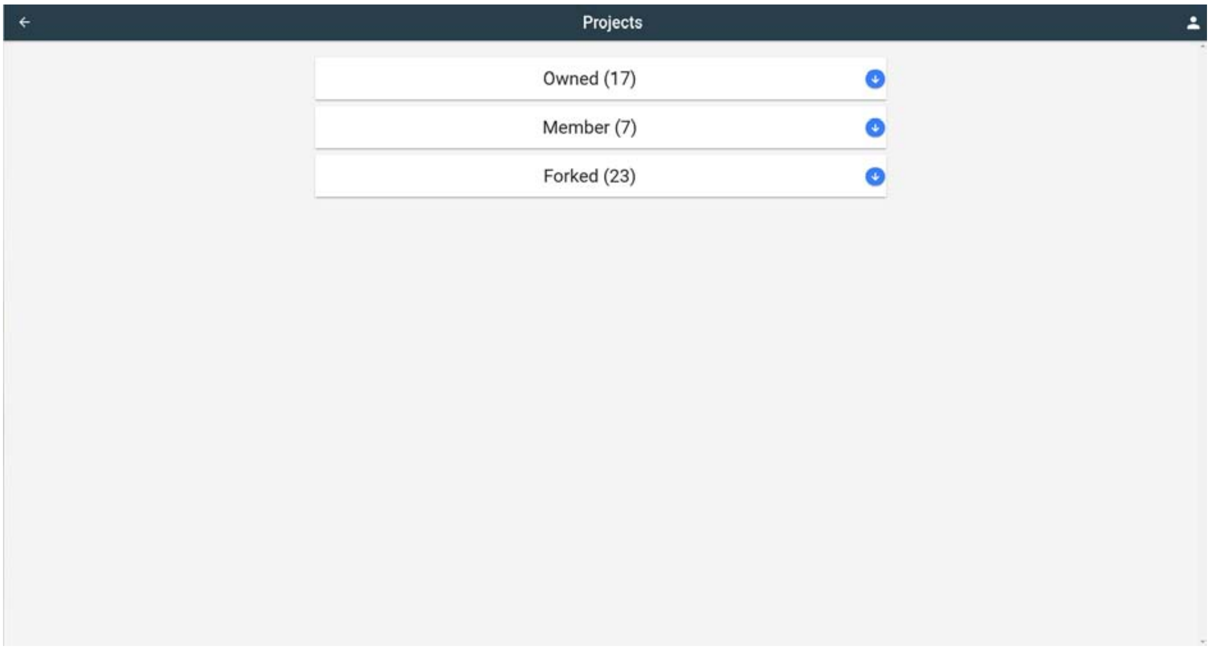


Figure 4.16: Computer screen-shot of www.gitcv.com

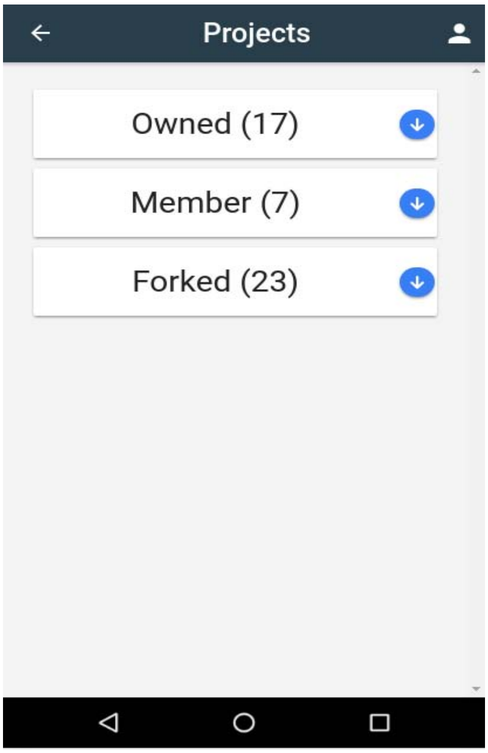
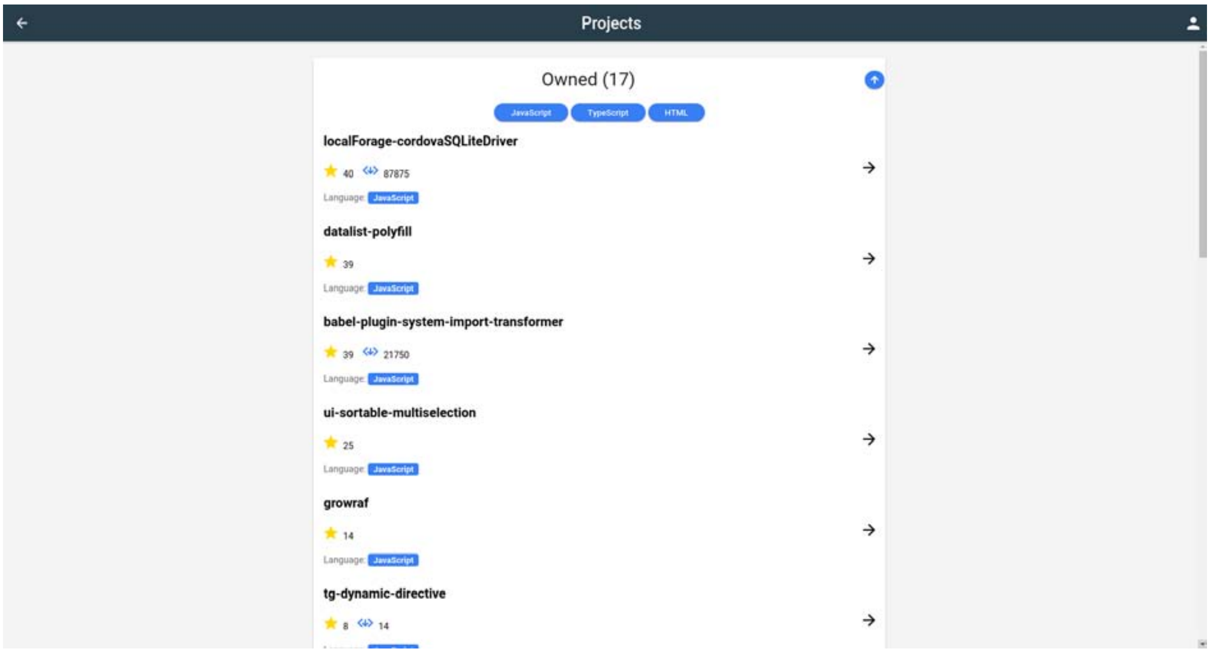


Figure 4.17: Mobile screen-shot of www.gitcv.com

Three categories can be seen as mentioned in the previous chapters. You can open any of them and use the available filters to get only projects for the specified languages.

Owned Projects



localForage-cordovaSQLiteDriver
40 87875
Language: JavaScript

datalist-polyfill
39
Language: JavaScript

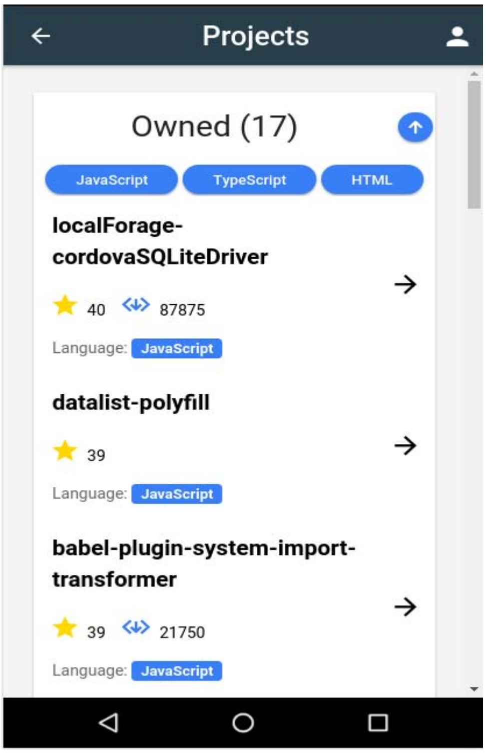
babel-plugin-system-import-transformer
39 21750
Language: JavaScript

ui-sortable-multiselection
25
Language: JavaScript

growraf
14
Language: JavaScript

tg-dynamic-directive
8 14

Figure 4.18: Computer screen-shot of www.gitcv.com



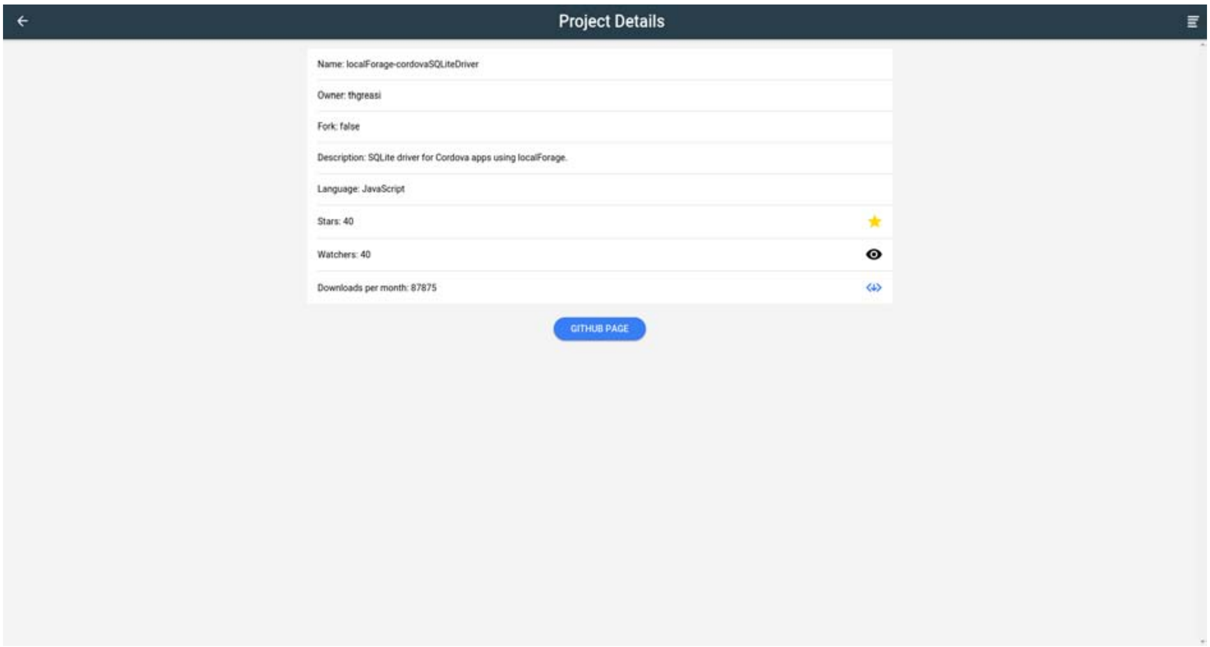
localForage-cordovaSQLiteDriver
40 87875
Language: JavaScript

datalist-polyfill
39
Language: JavaScript

babel-plugin-system-import-transformer
39 21750
Language: JavaScript

Figure 4.19: Mobile screen-shot of www.gitcv.com

You can navigate further inside a project by clicking on it. In that page you can see more details about the project and a redirect button to the official GitHub page.



Name: localForage-cordovaSQLiteDriver

Owner: thgreasi

Fork: false

Description: SQLite driver for Cordova apps using localForage.

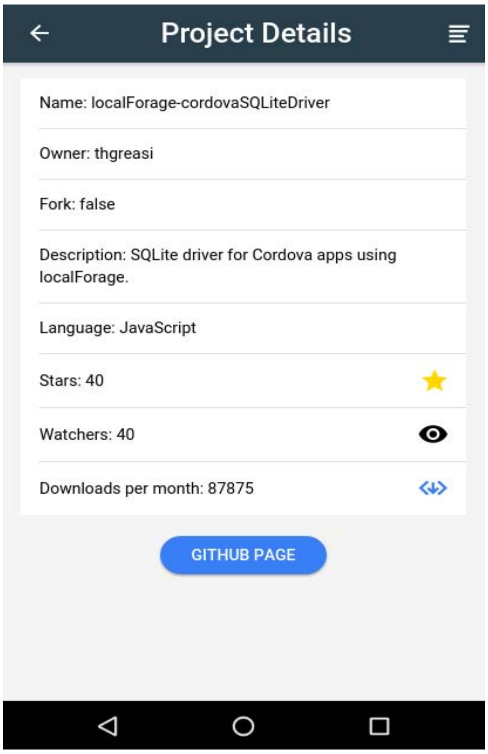
Language: JavaScript

Stars: 40

Watchers: 40

Downloads per month: 87875

Figure 4.20: Computer screen-shot of www.gitcv.com



Name: localForage-cordovaSQLiteDriver

Owner: thgreasi

Fork: false

Description: SQLite driver for Cordova apps using localForage.

Language: JavaScript

Stars: 40

Watchers: 40

Downloads per month: 87875

Figure 4.21: Mobile screen-shot of www.gitcv.com

You can also navigate to the page of the followings, from where you can ”GitCV” their profile, to the Gists page of the user, to the contribution page, to his GitHub profile and to the Organizations page. All the previous and even more can be seen on the website at: https://www.gitcv.com

5

Future Prospects and Thesis Summary

5.1. Future Prospects

The basic goal of this project is to be established as the best tool to search and evaluate firstly a developer and subsequently every person from all labor sectors. The development line that was adopted from the beginning of this project will be followed in the future, meaning that the retrieval of data from outer services/websites and the evaluation of will continue accordingly. Thus, based on the above some extra functions/features will be mentioned that could be added over the next years.

5.1.1. Integrate new services on Client-side

A part of the original conception of the idea to create GitCV, was to integrate LinkedIn too. By that it could be possible to represent a full CV of a user, because the evaluated data from GitHub profile could be represented, alongside with the information from LinkedIn. But because LinkedIn does not currently provide an open API it was not possible to integrate this service on the prototype of the platform. Nevertheless, it is a great necessity to present a complete profile of a person. Thus, it is possible to create an agreement with LinkedIn to provide us the necessary data for a person. The possibility this service will have a cost is high, because nowadays data costs even more if the provider is well known. Likewise, it is possible to get information from other social media like Facebook, to get advantage of the possible information about the current working position of the user. Once again, the most common and possible problem is that such websites do not maintain open APIs for developers.

Another service that will be positive to integrate in the future, is Stackoverflow [13, 44]. One of the most well known communities for programmers and computer engineers. So why this integration could make a difference for the platform? A use case of Stackoverflow is: a user posts a question about a programming problem or bug that he is not capable to solve, any other user can answer the question and solve this problem. The answer will be evaluated by other users and every evaluation will increase or decrease the rating of the latter. The evaluation system of an answer in Stackoverflow is really sophisticated since an up-vote of an answer is only taken into consideration, if the voting user has a high validity over the community. This is achieved by answering on similar posts correctly and by taking votes from other valid community members. Thus, if a user is capable to answer a question and solve a problem, then it is highly possible that a company would be interested in him since he is a good programmer and answers efficiently on particular problems. So, it could be really helpful to include a list of the best answers of the user in GitCV, so that an HR employee could decide much easier and more efficiently if a person has the knowledge to be part of his company. Unfortunately, a problem occurred on the integration step. The only unique identifier the Stackoverflow profile has, is an id number and nothing else. It is not possible to get access to his email or anything else that will make possible to connect the profiles among the other platforms. The obvious identifier that was needed was the email address of the profile, like it was materialized for GitHub. But, as explained this is not possible. Nevertheless, if the policy of Stackoverflow changes direction, or the company is influenced from other companies and adds a public email address on every user profile it could be easy to integrate their service[12].

Other services that could be integrated are Bitbucket and GitLab, which are similar platforms with GitHub, less known and obviously from different companies. The reason behind this integration is that a user may use GitHub or GitLab or another web version control system, so it is wrong to reject those information [13]. There is no obvious problem integrating those services now but some problems that could possibly exist are related with the API that every website maintain.

All the previous could be integrated on the client-side and as it seems there is still no need for a server-side. After, all the data that is gathered for a person, from all the different websites, it is a good idea to save those data locally. Thus, these data could be used again if needed instead of making and executing all the necessary requests and algorithms to export them. This will save time and computing power from the device of the user as it will be much faster and less battery draining. This can be achieved with the open-source localForage library of Mozilla [34], which lets the user to store the data on the cache memory of the browser and manipulate them as he wishes. Because of the non-existence of a database in the platform so far, it was not possible to use an indexer to compare users. But with this addition to the platform, any time a user is accessed, it can firstly be saved so it could be possible to load him later from the local database, and secondly he could be added on a list with users. On this list an indexer can be applied as mentioned earlier, so that a company can compare them and decide which suits for them. This would be a great tool for the companies and his development has already started.

In conclusion, all the previous features can be added on the platform with no huge changes on the basic architecture of GitCV but those features can increase the quality of the developer's evaluation system.

5.1.2. Server-side

However, some features require a server-side to be added at GitCV. A very a good idea is to Rank a user by his followers, like the PageRank algorithm of the Google Search Engine [30, 36]. The basic on this algorithm, in GitCV's case, is that if a user has followers with a high Rank then his Rank will increase because of the latter. This idea is easier to understand if the algorithm is represented with network nodes. Lets see an example on the graph below.

Nodes

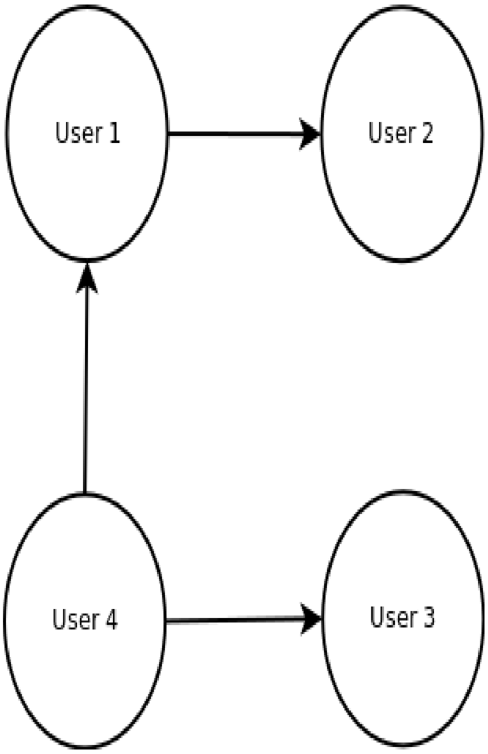


Figure 5.1: Network nodes(All rights reserved, created with Dia)

Lets name the Rank of each user Gr-Rank. In this example user 1 has high Gr-Rank, user 4 has low Gr-Rank and user 2,3 have identical Gr-Ranks. Now, if user 1 follow user 2 as shown in the graph and user 4 follow user 3 then the Gr-Rank of each one will change. Keep in mind that Gr-Rank(2) == Gr-Rank(3) before the connections are added that are seen on the graph. Now, the Gr-Rank(2) will increase because of the high Gr-Rank(1) and Gr-Rank(3) will be less than Gr-Rank(2), because it is connected with a lower Gr-Rank user than user 2.

Thus, by using this technique an extra metric can be added about the level of each user and characterize him better, more widely and more efficiently. The necessary and sufficient condition to add this feature is a server-side with a really big database, so that all the data can be saved and then apply the Gr-Rank.

Another feature that could be added on the platform, if all the data are saved and obviously exists a server-side, is static code analysis on the projects of each user. By doing that, it could be possible to export valuable information about the quality of the code for each user. In brief, the identification of the quality of the structure of a project or a certain file and the programming handwriting of each user could be really easy and fast. Thus, by comparing the structurability of the coding skills of the users, another metric could be added to evaluate better each user.

Summarizing, all the previous features that were proposed, are aiming at optimizing the user's evaluation process, so that anyone can identify the quality of each user's skills. Thus, the target is to materialize all the previous ideas over time and show off the usefulness of this platform.

5.1.3. GitHub Marketplace

GitHub announced the Marketplace at 15 May of 2017. GitHub Marketplace, just like it sounds, is a storefront for applications aimed at integrating and optimizing workflow across the development process. The Marketplace contains add-on tools for projects, either simple or specific. GitHub Marketplace aims to be one-stop shopping for the busy developer who doesn't have time for rabbit holes. Users can discover and purchase applications that cover the entire development process, from project management to continuous integration to code review — all pre-wired to integrate seamlessly into the GitHub workflow. A major benefit to the marketplace setup is the ability to shop using your GH account, with payment via your existing billing info. Initially, there are about a dozen "integrator" companies offering apps in the GH Marketplace, with many more to come. Apps fall into four categories (though more categories will be added over time in response to user demand): Code quality, continuous integration, monitoring, and project management. Current Marketplace members include Travis CI, Appveyor, Waffle, ZenHub, Sentry and Codacy. All the previous "GitHub apps" use the same API. It is really obvious, that with all this power that GitHub Marketplace provide, anyone can build a GitHub app and upload it to the Marketplace. Of course there are some requirements to be met [5]. The purpose of GitHub Marketplace is to make it easier for everyone on the team, like marketing and customer support, to harness the same tools, so they can collaborate better with the builders.

But what about GitCV? Can this application be integrated, which has not as primary target the developers, on GitHub Marketplace? The answer is yes, but there are some changes and additions on the architecture of the system that must be done first. In the application is gathered plenty information about the user and his activity but currently the target is to create a CV for the developer and not help him with his workflow development. The good part of our system architecture is that the gathered and evaluated data for each developer, can be used for other purposes. The extra functionality that could be provided and help the workflow of a developer, is about "pull requests". As described earlier at chapter 3, pull requests are really important for collaboration and particularly often. Thus, the percent of the successfully merged "pull requests" on a particular project can be provided, by commenting on the action of the "pull request". This information will help the maintainer of the project on anticipating a good addition/bug fix or whatever this pull request can be. Lets mention a scenario for this feature. We have two actors and one repository. Actor 1 is the maintainer of the repository and actor 2 is a contributor on this project. When actor 2 makes a "pull request" on the project, GitCV will comment on his action, by stating the number of the "pull requests" this contributor has ever done on this project and the number of the merged ones, with the following format "3/5 Merged". This comment will characterize the possibility of the usefulness of this "pull request". It is obvious that if the most "pull requests" of the developer on a certain project are merged, the contributor is reliable and it can be expected that his next "pull request" will be similarly good, targeted and clear. All this information about the "pull requests" is gathered from GitCV, to export the projects that the user has contributed. Thus, actor 1 can check out the "pull request" at once or not, depending the information that GitCV provided about this contributor. A good percentage on the merged "pull requests" of a contributor, will automatically push actor 1 to check and merge the "pull request" right away. In the opposite case, the maintainer will check the pull request noticeably later, because of the bad quality of the previous "pull requests" of the particular contributor and due to the lack of time. This feature will help the maintainer to organize his tasks better and deal first with the "pull requests" that have higher possibility to increase the level, usefulness of the project.

All the above can be materialized after adding a different system design for GitCV, so that it can be used as a GitHub App. The first thing that must be done, is to provide two options on the home-screen of our web application. The first option is what was developed so far and can be used from anyone but mainly the HR department. The second option is only for the developers. This option will need a token from GitHub, which is easily generated by using OAuth. This token will give to GitCV the needed information about the developer (which is actor 2 on the previous scenario) and some extra permissions, so that the platform can comment the exported information for the user. There is no reason to defy this information because it is exported from the actions and generally the activity of each developer. So it is necessarily reliable.

This is just a glance of what can be achieved, with all the information that is gathered for each developer, and this feature will be added soon enough.

5.2. Thesis Summary

In conclusion, after analyzing the significance of the data analysis on social medias, and more importantly the export of major,descriptive and substantial results about a person, the necessity to use such techniques, so that a person can be evaluated for his skills, is really big and obvious. Nowadays, people do not have the time to determine if a person is valuable for a project or a problem they need to solve. Thus, this operation must be automate, so that everyone can evolve more as a civilization, as a specie without consuming valuable time and human resources on such problems. It is necessary to find the perfect fit for a job.

As far as concerned our platform, more features and services must be integrated to create an optimization tool so that anyone could be able to choose the best candidate worker for each project, depending on the nature of each one. We hope that we can fulfill this project and make the world a better place.

Bibliography

[1] Angularjs 2. URL https://angular.io/.

[2] Http access control. URL https://developer.mozilla.org/en-US/docs/Web/HTTP/Access_control_CORS.

[3] Github api repositories, . URL https://developer.github.com/v3/repos/.

[4] Github api repositories downloads, . URL https://developer.github.com/v3/repos/downloads/.

[5] Github marketplace, . URL https://github.com/marketplace.

[6] Ionic 2. URL http://ionic.io/2.

[7] Npmjs. URL https://www.npmjs.com/,https://docs.npmjs.com/getting-started/what-is-npm.

[8] Observables in http requests. URL https://angular.io/docs/ts/latest/guide/server-communication.html,https://angular-2-training-book.rangle.io/handout/observables/using_observables.html.

[9] Open-source contributors. URL https://github.com/tenex/opensourcecontributors.

[10] Tenex developers. URL https://github.com/tenex.

[11] GitHub API. Rate limiting. URL https://developer.github.com/v3/#rate-limiting.

[12] Ali Sajedi Badashian, Afsaneh Esteki, Ameneh Gholipour, Abram Hindle, and Eleni Stroulia. Involvement, contribution and influence in github and stack overflow. In *Proceedings of 24th Annual International Conference on Computer Science and Software Engineering*, pages 19–33. IBM Corp., 2014.

[13] Andrew Begel, Jan Bosch, and Margaret-Anne Storey. Social networking meets software development: Perspectives from github, msdn, stack exchange, and topcoder. *IEEE Software*, 30(1):52–66, 2013.

[14] John D Blischak, Emily R Davenport, and Greg Wilson. A quick introduction to version control with git and github. *PLoS Comput Biol*, 12(1):e1004668, 2016.

[15] Rachel Schutt Cathy O´Neil. *Doing Data Science, Straight Talk from the Frontline.* O´REILLY MEDIA, 2013. ISBN 1-449-35865-9.

[16] Google Chrome. Add to homescreen. URL https://developer.chrome.com/multidevice/android/installtohomescreen.

[17] LinkedIn Corporation. About linkedin, 2015. URL https://engineering.linkedin.com/architecture/brief-history-scaling-linkedin.

[18] Laura Dabbish, Colleen Stuart, Jason Tsay, and Jim Herbsleb. Social coding in github: transparency and collaboration in an open software repository. In *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work*, pages 1277–1286. ACM, 2012.

[19] Paul Sloan Daniel Terdiman. Forget linkedin: Companies turn to github to find tech talent. URL https://www.cnet.com/news/forget-linkedin-companies-turn-to-github-to-find-tech-talent/.

[20] Google Developers. Progressive web apps. URL https://developers.google.com/web/fundamentals/getting-started/codelabs/your-first-pwapp/.

[21] John Charles Gober Diana Zhang Wobus. A step-by-step illustration of building a data analysis tool with macros. 1997. URL http://www2.sas.com/proceedings/sugi22/ADVTUTOR/PAPER42.PDF.

[22] Alexander Serebrenik Andy Zaidman Georgios Gousios, Bogdan Vasilescu. Lean ghtorrent: Github data on demand, 2014. URL http://www.win.tue.nl/~aserebre/msr14georgios.pdf.

[23] GitHub. About gists, . URL https://help.github.com/articles/about-gists/.

[24] GitHub. Organizations, . URL https://help.github.com/articles/what-s-the-difference-between-user-and-organization-accounts/.

[25] Georgios Gousios, Bogdan Vasilescu, Alexander Serebrenik, and Andy Zaidman. Lean ghtorrent: Github data on demand. In *Proceedings of the 11th Working Conference on Mining Software Repositories*, pages 384–387. ACM, 2014.

[26] Robert Greiner. Github profile. URL http://robertgreiner.com/2012/02/introducing-gitcv/.

[27] Claudia Hauff and Georgios Gousios. Matching github developer profiles to job advertisements. In *Proceedings of the 12th Working Conference on Mining Software Repositories*, pages 362–366. IEEE Press, 2015.

[28] Eirini Kalliamvakou, Georgios Gousios, Kelly Blincoe, Leif Singer, Daniel M German, and Daniela Damian. The promises and perils of mining github. In *Proceedings of the 11th working conference on mining software repositories*, pages 92–101. ACM, 2014.

[29] M. J. Kumar. Evaluating scientists: Citations, impact factor, h-index, online page hits and what else? *IETE Technical Review*, 26:165–168, 2009.

[30] Amy N Langville and Carl D Meyer. *Google's PageRank and beyond: The science of search engine rankings*. Princeton University Press, 2011.

[31] Antonio Lima, Luca Rossi, and Mirco Musolesi. Coding together at scale: Github as a collaborative social network. *arXiv preprint arXiv:1407.2535*, 2014.

[32] Nora McDonald and Sean Goggins. Performance and participation in open source software on github. In *CHI'13 Extended Abstracts on Human Factors in Computing Systems*, pages 139–144. ACM, 2013.

[33] Zach Moazeni. Events of a github user. URL http://zmoazeni.github.io/gitspective/.

[34] Mozilla. Localforage, offline storage. URL https://localforage.github.io/localForage/.

[35] Saya Onoue, Hideaki Hata, and Ken-ichi Matsumoto. A study of the characteristics of developers' activities in github. In *Software Engineering Conference (APSEC), 2013 20th Asia-Pacific*, volume 2, pages 7–12. IEEE, 2013.

[36] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.

[37] Baishakhi Ray, Daryl Posnett, Vladimir Filkov, and Premkumar Devanbu. A large scale study of programming languages and code quality in github. In *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pages 155–165. ACM, 2014.

[38] Marko Rodriguez. Big graph data on hortonworks data platform, 2012. URL https://hortonworks.com/blog/big-graph-data-on-hortonworks-data-platform/.

[39] Matthew A Russell. *Mining the Social Web: Data Mining Facebook, Twitter, LinkedIn, Google+, GitHub, and More.* " O'Reilly Media, Inc.", 2013.

[40] Nancy R. Tague. *The Quality Toolbox, Second Edition.* ASQ Quality Press, 2005. ISBN 978-0-87389-639-9.

[41] Ferdian Thung, Tegawende F Bissyande, David Lo, and Lingxiao Jiang. Network structure of social coding in github. In *Software maintenance and reengineering (csmr), 2013 17th european conference on*, pages 323–326. IEEE, 2013.

[42] Jason Tsay, Laura Dabbish, and James Herbsleb. Influence of social and technical factors for evaluating contribution in github. In *Proceedings of the 36th international conference on Software engineering*, pages 356–366. ACM, 2014.

[43] Jason Tsay, Laura Dabbish, and James Herbsleb. Let's talk about it: evaluating contributions through discussion in github. In *Proceedings of the 22nd ACM SIGSOFT international symposium on foundations of software engineering*, pages 144–154. ACM, 2014.

[44] Bogdan Vasilescu, Vladimir Filkov, and Alexander Serebrenik. Stackoverflow and github: Associations between software development and crowdsourced knowledge. In *Social computing (SocialCom), 2013 international conference on*, pages 188–195. IEEE, 2013.

[45] Bogdan Vasilescu, Yue Yu, Huaimin Wang, Premkumar Devanbu, and Vladimir Filkov. Quality and productivity outcomes relating to continuous integration in github. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*, pages 805–816. ACM, 2015.

[46] Sarah von der Mühlen, Tobias Richter, Sebastian Schmid, Elisabeth Marie Schmidt, and Kirsten Berthold. Judging the plausibility of arguments in scientific texts: a student–scientist comparison. *Thinking & Reasoning*, 22(2):221–249, 2016. doi: 10.1080/13546783.2015.1127289. URL `http://dx.doi.org/10.1080/13546783.2015.1127289`.

[47] Yu Wu, Jessica Kropczynski, Patrick C Shih, and John M Carroll. Exploring the ecosystem of software developers on github and other platforms. In *Proceedings of the companion publication of the 17th ACM conference on Computer supported cooperative work & social computing*, pages 265–268. ACM, 2014.