FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Quantified Self for Developers

Eduardo Manuel Oliveira Taveira Baptista de Almeida

Mestrado Integrado em Engenharia Informática e Computação

Supervisor: Ademar Manuel Teixeira de Aguiar

June 27th, 2017

© Eduardo Manuel Oliveira Taveira Baptista de Almeida, 2017

Quantified Self for Developers

Eduardo Manuel Oliveira Taveira Baptista de Almeida

Mestrado Integrado em Engenharia Informática e Computação

Approved in oral examination by the committee:

President: Prof. Hugo José Sereno Lopes Ferreira
External Examiner: Prof. Jorge Manuel de Azevedo Pereira Simões
Supervisor: Prof. Ademar Manuel Teixeira de Aguiar

June 27th, 2017

Abstract

Quantified Self is the process of an individual self-tracking his biological, physical, behavioral or environmental information. This is a relatively recent topic that has been generating an increasingly higher amount of interest and traction that began with the launch of smartphones. Since then, people's phones started acting more as personal devices and bundling sensors like accelerometers, gyroscopes and dedicated health processors. These, just by themselves, can be used to gather some data about the user, like sleep analysis and fitness activity. The rise of the smartwatches has contributed even more to the awareness of Quantified Self, as personal analytic data can be obtained in an even easier and more accurate way, since this kind of devices is in constant contact with a person's skin. This allows for relevant data like the user's heart rate to be continuously monitored and analyzed.

Applied to software development, and more specifically to software developers themselves, personal and team metrics can easily be acquired from widely used project management and code quality software, of which some examples are *GitHub*, *JIRA* and *Bugzilla*. This data can then be correlated with the personal analytics mentioned above to measure productivity and happiness, not only of a single person, but also of the team as a whole. Happiness is a metric used mostly on agile software development teams, and called that way based on the assumption that the success is based on the well-being of a team, which is consequently more efficient and productive the happier it is. There were, at the time, no applications that correlate both personal and work-related metrics, and these together can be used to detect issues in order to aid in finding improvements to a person's habits and workflow.

As a result of this work, a platform was developed, with four separate applications: one for *iOS*, one cross-platform application for *Android* and *iOS*, a web client application and, finally, a server application.

The *iOS* application, *QuantiDev*, allows for the acquisition of personal data, gathered from the phone and connected devices like smartwatches and fitness trackers, and for the correlation of that same data with work-related information. The cross-platform application, *InteractDev*, is a note-taking and interaction evaluation application, which on top of allowing for the sharing of interaction notes between team members, also provides the platform with data about the health of an interaction between the same team members. The web client application, *QuantiDev Web*, allows for the management of developers and their teams, data acquisition, generation of charts and the analysis of data about these teams by their leaders. Finally, the developed server application, *quantiserver*, establishes the connection between the three other applications along with being tasked with the acquisition of work-related data from external services and scheduled maintenance tasks.

This platform has, then, the objective of helping software developers and the teams they are part of, identifying patterns, problems and, as a result, improve both their personal and work life.

A validation phase was conducted, in order to validate the concept behind the platform, by software developers with experience of working in teams and of being a leader in these same teams. The results obtained confirmed that software developers feel that their productivity may be affected by personal and external factors, and revealed a great interest in a platform like the one that was developed. Along with that validation, the platform was also tested in a real environment with some selected users, where usability tests and follow-up studies were conducted.

Resumo

Quantified Self é o processo de um indivíduo registar a sua informação tanto biológica, física, comportamental como do meio envolvente. Este é um tópico relativamente recente, que tem vindo a gerar cada vez mais interesse desde o lançamento dos *smartphones*. Desde então, estes têm vindo a ser cada vez dispositivos mais pessoais, com sensores como acelerómetros, giroscópios, processadores dedicados a saúde, entre outros. Estes, só por si, podem ser usados para adquirir informação sobre o utilizador, como hábitos de sono e atividade física. O aparecimento dos *smartwatches* contribuiu ainda mais para o sucesso do *Quantified Self*, já que analíticas pessoais podem ser obtidas de uma forma ainda mais fácil e precisa, uma vez que este tipo de dispositivos está sempre em contacto com a pele. Assim, é possível que dados relevantes, como a pulsação cardíaca do utilizador, estejam continuamente a ser monitorizados.

Aplicado ao desenvolvimento de software, e mais especificamente aos desenvolvedores de software, é possível obter métricas sobre a equipa e cada membro, a partir de software de gestão de projetos e de qualidade de código utilizado por estes, dos quais são exemplo o *GitHub*, *JIRA* e *Bugzilla*. Os dados recolhidos podem então ser relacionados com os obtidos da pessoa, como descrito anteriormente, de modo a analisar a produtividade e a *happiness*, não só de uma pessoa, mas da equipa como um todo. *Happiness* é uma métrica utilizada, principalmente nas equipas de desenvolvimento de software ágeis, e designada desta forma, com base na assunção que a felicidade está relacionada com o bem-estar de uma equipa, que é consequentemente mais eficiente e produtiva quanto mais feliz é. Não existiam aplicações que relacionassem métricas pessoais e de trabalho, que quando analisadas em conjunto, pudessem ser utilizadas para detetar problemas e auxiliar a descoberta de melhorias de hábitos e modo de trabalho de um indivíduo.

Como resultado deste projecto, uma plataforma foi desenvolvida, composta por quatro aplicações separadas: uma para *iOS*, uma multi-plataforma para *Android* e *iOS*, uma aplicação web e, finalmente, uma aplicação para servidor.

A aplicação de *iOS*, *QuantiDev*, permite a aquisição de dados pessoais recolhidos pelo telemóvel do utilizador e de dispositivos a este conectados como *smartwatches* e *fitness trackers*, podendo então correlacionar esses dados com informação relacionada com o trabalho do utilizador. A aplicação multi-plataforma, *InteractDev*, é uma aplicação de notas e avaliação de interação, que para além de permitir a partilha de notas sobre uma interação entre os intervenientes, também fornece dados sobre a qualidade da mesma à plataforma. A aplicação web, *QuantiDev Web*, permite a gestão de desenvolvedores de software e das equipas na qual estão contidos, aquisição de dados, formulação de gráficos e a análise de dados sobre equipas pelos seus líderes. Finalmente, a aplicação de servidor, *quantiserver*, estabelece a ligação entre as outras três aplicações, para além de ser responsável pela aquisição de dados relacionados com o trabalho dos utilizadores, bem como por tarefas de manutenção.

A plataforma tem então o objetivo de ajudar desenvolvedores de software e as equipas nas quais

estão contidos a identificar padrões, problemas e, como resultado desta análise, melhorar a sua vida tanto a nível pessoal como profissional.

Foi também conduzida uma fase de validação, de modo a validar o conceito por detrás da plataforma, recorrendo a programadores com experiência de trabalho em equipa, bem como líderes das mesmas. Através dos resultados obtidos, pode concluir-se que os desenvolvedores de software sentem que a sua produtividade pode ser afetada por fatores tanto pessoais como externos, e revelaram um grande interesse numa plataforma como a que foi desenvolvida. Para além dessa validação, a plataforma foi também testada num ambiente real por utilizadores seleccionados, em que testes de usabilidade e *follow-up studies* foram conduzidos.

Acknowledgements

After a whole semester of working on my dissertation, I am now writing this note as I finish it and complete a phase of my life.

This note serves as a “thank you” to everyone that directly or indirectly helped me during its development.

First, I would like to start by thanking the most important people in my life: my family, more specifically my parents, brother, grandparents and uncle. Without them, and without all the opportunities they have given me, I would surely not be writing this.

I would also like to thank my supervisor, Prof. Ademar Aguiar, for all the support during the development of this work and for giving me the freedom of letting it be what I wanted it to, while also helping me whenever I needed. Lachlan Heasman, a renowned agile coach, was also a great help by assisting me on the topics of team productivity and happiness. I also need to thank my “thesis colleague”, Bruno, for giving me something to look forward to whenever I worked on this project, discussing issues with me, proof-reading everything I did (more than once!)... or just goofing around at times.

I end this by thanking all my friends for bearing with me and always being there when I needed them the most (or when I just wanted to hang out!). I am not going to mention them by name, since fortunately I am lucky that they are more than just a handful, but they know who they are.

Eduardo Almeida

"If you ever code something that feels like a hack but it works, just remember that a CPU is literally a rock that we tricked into thinking."

Ben Driscoll

Contents

List of Figures

List of Tables

Abbreviations

API	Application Programming Interface
CRUD	Create, Read, Update and Delete
CSS	Cascading Style Sheets
CVS	Concurrent Versions System
FEUP	Faculdade de Engenharia da Universidade do Porto
GNU	GNU's not UNIX
GPS	Global Positioning System
HTTP	Hypertext Transfer Protocol
IBM	International Business Machines Corporation
JSON	JavaScript Object Notation
KPI	Key Performance Indicator
NASA	National Aeronautics and Space Administration
PCL	Portable Code Library
PDA	Personal Digital Assistant
RCS	Revisions Control System
REST	Representational State Transfer
SDK	Software Development Kit
SVG	Scalable Vector Graphics
UML	Unified Modeling Language

Chapter 1

Introduction

In the initial chapter of this report, the subject of the present dissertation is introduced by describing its context, along with the motivation to have performed the work, its goals and the layout of the present report.

1.1 Context

Nowadays, it is easy to gather data from most things people do in their lives, be it from what they do for fun or for work. This data can be used to acquire insights about a single person or from work teams, though the personal data is normally disregarded in the context of software development teams, as it is usually only associated to the Quantified Self.

Existing work on the area of agile process improvement mostly focus on the teams themselves and not on the individual software developers. It can then be said that there was a gap in this field that this work attempted to fill.

This project addresses the topics of agile methodologies, personal analytics, process improvement and software development. It is, though, mostly focused on process improvement on a personal level.

This topic was proposed by the *Software Engineering Research Group* at *Faculdade de Engenharia da Universidade do Porto*, on the context of agile methods, a subject related to software engineering.

1.2 Motivation

Professionals in the field of software engineering often have not so good days, on which they are not as focused or as productive, but also have excellent days, where they complete more work than they ever considered feasible. There must be reasons behind this phenomenon, and the answers are

normally hard to acquire, at least without a help of a dedicated tool to do so. Unfortunately, there was none when this report was written.

There are tools that analyze metrics from a person's life, and others that analyze the quality of the code written by a given person. There are, though, no tools that relate these two. They are not, nevertheless, completely unrelated.

As of that, a tool to fill this gap was developed. This work is relevant, as it can be used to assist software developers understand the issues with their productivity. It is different from what exists as there are, at the time of writing this report, no other solutions that relate personal analytics data with data gathered from tools used by teams of software developers. The information acquired can then be used to help professionals find solutions for and the reasoning behind their issues.

1.3 Goals

The main objective of this work was to create tools that collect data about behaviors and the environment of software developers and the agile software development teams they may be contained in. This data can then be later used to attempt to find correlations between these factors and the users' productivity.

In order to do so, a set of applications that collect data about and correlate the behaviors and environment of their users with work-related data were developed.

While that there are behaviors that will affect (positively or negatively) most users in the same way, others will probably only have any meaning when associated with a certain user. With that in mind, the analysis of the data must not be generalized but tailored by the system to each individual user.

1.4 Document Structure

Apart from the introduction, this document includes the following chapters:

- In chapter 2, the concept of Quantified Self is introduced, along with available metrics, indicators and measurements, as well as ways to measure them;
- In chapter 3, developer and team productivity, along with the related sub-topics Quantified Team and Happiness are depicted;
- In chapter 4, the state of the art with regards to related tools and technologies in the scope of this work is described;
- In chapter 5, an overview about data visualization, analysis, and how they were used in this work is given;

- In chapter 6, the developed solution is detailed, along with the reasoning behind the platform and technology choices;
- In chapter 7, the methodology, results and the analysis of the validation of the project is described;
- Finally, in chapter 8, conclusions are drawn from the work conducted, and the future work is described.

Chapter 2

Quantified Self Concepts, Tools and Technologies

In this chapter, the state of the art and related work of tools for Quantified Self and software developers are presented, along with technologies used by them, in order to document and evaluate what existed in the market. This is a recent and innovative subject, which leads to a lot of open problems and few products that provide a feasible solution to parts of the presented question.

The analysis of the state of the art allows for the assessment of existing functionalities and how these could be integrated and improved upon in order to create a solution to the problem.

2.1 Quantified Self

Quantified Self is the process of self-tracking an individual's biological, physical, behavioral, or environmental information [Swa13]. Its associated movement is a global effort, continuously growing in followers, to use new technologies, both mobile and wearable, in order to acquire data about the everyday activities of a user without the need of any intervention by their part.

It started to gain traction since the *boom* of the smartphones, which was when mobile phones started to be bundled with sensors like accelerometers, gyroscopes and dedicated health processors. The reliability of the data acquired has also increased with the launch of the smartwatches. They can be constantly gathering data about the user without the need of any action from the wearer as they are in constant contact with the body.

2.1.1 Metrics

The Quantified Self movement does not define a strict set of metrics, but simply a way for metrics to be gathered.

A metric is a term used to refer to a quantitative measure in a given realm [tra05] – in this case, related to a person's life. There are a lot of interesting metrics that can be gathered from an individual and that can be useful to this work.

- **Hours of Sleep** — The number of hours that a person sleeps can directly affect that individual's performance, since a tired brain can have some correlation with the productivity of a person [Vil96]. In the case of the topic that is being approached, the capability of a person solving problems and fixing issues may be affected;
- **Commute Time** — The number of hours a person spends on their daily commute can have an impact on its productivity [vOiP11], since it can, as an example, negatively affect the levels of stress of a person if too much time is spent stopped in traffic. The type of transport used on the commute can also have an impact, since a person that travels by public transportation may not be affected by the same levels of stress;
- **Stress Levels** — The stress levels of a person can both positively and negatively impact their ability to perform well at work. Some people work well (and some, even better) under pressure, while others can not handle it too well and are negatively affected [BD07];
- **Time Exercising** — The time a person spends exercising (at the gym, for example) in a given day may have an impact in their work performance.

2.1.2 Indicators

Indicators are similar to metrics, but instead of referring to the quantitative measures in a given realm, they refer to the qualitative measures [tra05]. Like the metrics, some relevant indicators that can be appropriate to this domain were also gathered.

- **Wake-Up Time** — The time a given person wakes up can influence its performance while working or studying [MPP00]. Consequently, an analysis can be done in order to help a user understand at which time he should wake up in order to maximize his efficiency while doing work;
- **Light Sickness** — Light sicknesses, like headaches and colds, can have an impact on the performance level of some people, even though the sickness is light enough not to affect the person's ability to go to work. This metric can be used in order to investigate if and how much a sickness affects a given worker's ability to perform their job efficiently. This would allow the application to give an insight about if more productivity is lost from calling in sick or going to work normally [Mid07] (and potentially taking longer to recover from the sickness);
- **Mood** — The mood of a person can have an effect on their behavior at work, which can affect the worker's performance [Geo91]. Consequently, asking a person how their mood is

at some key points of the day (for example, when the person starts working and after lunch) can help analyze why their performance levels are higher or lower at given times;

- **Diet** — The diet and type of food that a person eats can have an impact on its work performance, mainly if the person in question is under a (new) diet. Changes to the nutrition habits of a person can directly affect their levels of stress, and, as of that, gathering data from a person's eating habits (or the lack of) can then be considered useful on the realm of this topic [POL96];

- **Menstrual Cycle** — The current menstrual cycle of a woman can have an affect, both positive and negative, on physiological functions such as "attention, short-term memory, perceptual speed, perception of time, and reaction time" [GSW75]. Most of these factors affect the performance in the workplace, so it can be concluded that this metric is relevant to the topic being studied;

- **Weather** — Depending on the individual, the current weather may have an effect on a person's mood [KFY+05]. This can consequently affect that person's productivity, as described earlier.

2.1.3 Key Performance Indicators

Performance measurement is one of the prime principles of management. *KPI*s, or Key Performance Indicators, are measurable values that demonstrate how given objectives are, or not, being met. Identifying the indicators that are underperforming is useful in order to figure out what needs to be improved, and what is already at optimal levels.

Some examples of *KPI*s that can be used in the realm of this work are:

- **Story Points per Sprint** — In agile development, the amount of work required to complete a given task is measured by the means of story points, which are arbitrary values that define how much effort a task requires compared to others on the same sprint [Ham14] (a period of time in which work has to be done and is then evaluated at its end). The number of story points a software developer completes in a week is directly related to the work done, and can consequently be used as an indicator of their performance;

- **Number of Bugs in Code** — A developer may write a lot of code, but if the code written contains a great number of software bugs (which are, in basic terms, errors in the code, or more specifically, violated specifications [Kup89]), the time spent will not have been as useful as if the code was bug-free – which means that the time spent by the developer could have been more productive. Time will have to be spent in hunting down and fixing bugs which could have been avoided in the first place if the code written was of better quality.

2.1.4 Measurements

Measurements are similar to metrics, but instead of being an unbound value, there is a defined scale. Similar to the other two dimensions, there is also a set of measurements that can be gathered, and that can be considered useful in the scope of this work.

- **Hours at Work** — The number of hours that a person is at work / actively working can affect the person's performance per hour ratio. There is only a limited time that a person can stay focused on a given task, which varies from person to person;
- **Heart Rate** — The heart rate is related to the stress levels of a person as its variability can be used as a marker of stress [TAF+12]. Unlike the stress levels, the heart rate (and its variability) can be directly monitored easily by some devices, of which wearables are a good example.

Humans are good at gaming performance systems. As of that, metrics, indicators and measurements are usually not a good way of measuring employee performance in the software development industry [App16]. This does not apply to the scope of this project, though, as these dimensions are being used in order for developers to self-improve their happiness, quality of life and quality of their work, and not to be evaluated upon by their employees.

2.2 Developer and Team Productivity

2.2.1 Quantified Team

The Quantified Self movement can also be applied to teams rather than individuals. The domain of the data can not be the same, though, since it is not useful nor practical to analyze personal data (like the heart rate) in the context of a whole team.

It makes sense, though, to measure data that can be used to compare team members against other team members.

Agile Software Development is a term that describes a collection of "methods and practices based on the values and principles expressed in the Agile Manifesto" [All17b].

These principles are based on what is described by the Agile Alliance [All17a]:

- The priority of the developers should be to deliver continuously and early valuable software to the customer;
- Requirements should not be set in stone and may change at any time;
- New software updates should be released frequently;
- Both developers and business people should be working together daily;

- Developers should be motivated to perform their jobs;
- Face-to-face conversations are one of the most efficient and powerful ways of sharing information;
- The best way to measure progress is working software;
- Everyone related to the project should be able to work at a constant rate;
- There should be an endless attention to make the best product possible;
- It is essential to maximize the work not done;
- Teams should be self-organized;
- There should be regular reflections on how the team can become more effective, and the team should then change their behaviors appropriately.

Working with metrics in parallel with an agile software development cycle allows for small adjustments to be made in relatively short periods (at the end of each sprint), which results in the team working in a smarter way. In order to do this, the following steps can be followed, according to rules created by Cristopher W. H. Davis [Dav15]:

- **Collect** — The first step is to collect all relevant data from the team and its performance. Understanding what is happening is important before attempting to make any changes.
- **Measure** — After collecting data, it needs to be analyzed. One should look for tendencies in the data, define questions about the team, workflow, progress and then define what should be changed based on what was measured.
- **React** — Based on the findings, adjustments and changes are able to be made.
- **Repeat** — Past data should be saved in order to correlate with the new data gathered. This way, the teams can be continuously analyzed and the adjustments and changes can be improved steadily.

Agile is not easy to measure for. There are a lot of people involved, and they all have different roles, and different means of working. In addition to this, what someone considers good, someone else may not. For example, a developer may think that good work refers to the quality of the engineering of a block of code, while that a project manager could think that good work refers instead to work that was done both on time and on budget (without any regard to the quality of the code) [Dav15].

With regards to how the agile measurements can be done, it can then be concluded that, unlike the measurement of the quality of workers from some other industries, it is not straightforward,

and both the data and software are generated continuously, instead of at only some specific points in time. This means that, with regards to the way of working, a product mentality has to be used instead of a project one.

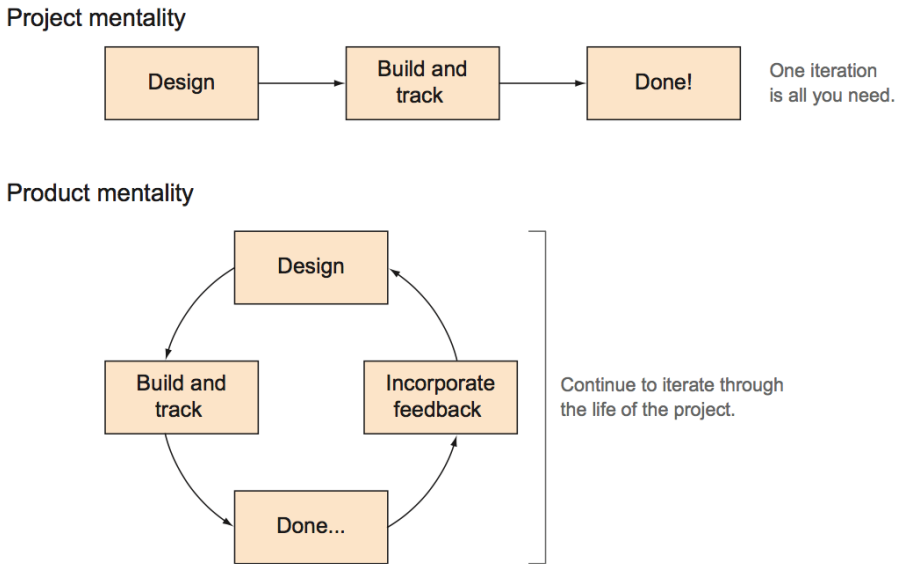


Figure 2.1: Project vs. Product Mentality [Dav15]

It is also not straightforward to gather data from the tools used by software developers. There are lots of tools used by them, and normally each person only uses a subset of the available tools.

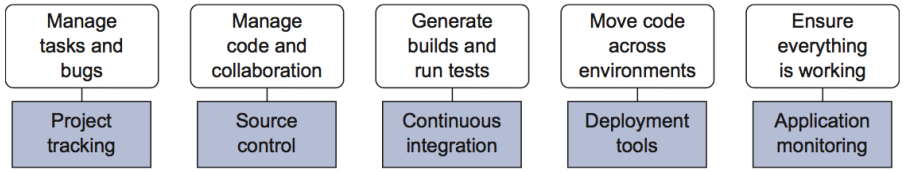


Figure 2.2: Systems used to deliver software and their features [Dav15]

There are, though, some questions that are better answered by some systems and other questions that are better answered by others, as can be seen in the figure below.

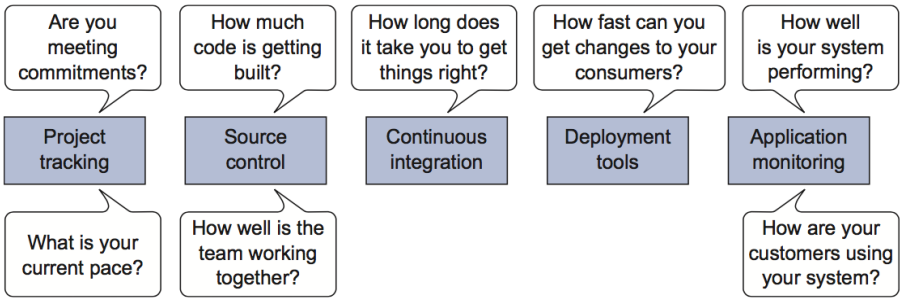


Figure 2.3: Data gathered from systems in the software development lifecycle [Dav15]

As for how the measurements themselves should be used, there are some rules that should be followed, in order for a concrete management practice to be pursued. The following list is based on the "Rules for Measurement", created by Jurgen Appelo [App16].

1. **Measure for a purpose** — The metrics being measured are not goals, and are not always directly related to performance. When a value is low, a question should be asked: Why it is low and how can it be improved?

2. **Shrink the unknown** — A metric is no more than a replacement for the knowledge that wants to be acquired. One should not gather conclusions from what is not fully understood.

3. **Seek to improve** — There are many metrics that can be measured, but only some really matter to the scope of what is being analyzed. One should not only look at values that make them look good, but at the ones that enable the work to be improved.

4. **Delight all stakeholders** — The work done by a person normally both depends on the work done by others and the opposite can also be verified. As of that, all measures should be analyzed from more than one perspective.

5. **Distrust all numbers** — Whoever looks at metrics is commonly biased in order to make their own results look superior to others'. One should always doubt the results and never gather conclusions without proper analysis.

6. **Set imprecise targets** — Having precise, numeric targets may result in people getting demotivated, if they are far from reaching the targets, or stop caring, if they have already reached the minimum defined threshold. By having vague targets, people will focus less on them which will yield better results.

7. **Own your metrics** — A person's work is of their own responsibility, and so should their metrics.

8. **Don't connect metrics to rewards** — Rewards may shortly motivate whoever wins them, but for each person who gets a motivation boost, a lot more get demotivated as they did not get rewarded. As such, the net result is, more often than not, negative and not positive.

For every person that is made employee of the month, dozens, hundreds, or thousands of colleagues are made losers of the month [App16].

9. **Promote values and transparency** — The human brain is smart enough to try and find ways to game a system. In order to prevent this, transparency about the values, intentions and metrics people are using should be promoted.

10. **Visualize and humanize** — When people look at numbers, they only see a value, and not the human beings behind them. Using figures and colors instead of numbers allows for the measurements to be kept close to the work that is being done.

11. **Measure early and often** — Sometimes, measurements are not performed as much as they should be. The sooner the measurements are done, the earlier the problems are found and the risks can be prevented.

12. **Try something else** — Doing the same things over and over again leads to exhaustion, which is never a good thing. The environment is not static, and the same should apply to what is being measured.

A good use of the measurements in the realm of the Quantified Team can assist in increasing the happiness of the software developers contained in a given team.

2.2.2 Happiness

> "Some people believe that happiness is something they will achieve when they are famous or successful, or when they acquire enough money. But research tells us that happiness is more a decision than a destination." [App16].

Happiness is a term that is discussed a lot, but not often understood. It is commonly defined as a "state of well-being, that's supported by positive or pleasant emotions ranging from relaxed contentment to intense enjoyment" [App16]. It can be concluded, then, that happiness can be either of short or long-term, an explosion of joy caused by oneself or others or a positive feeling that is buried in a person's brain. It can also be assumed that happy workers are more productive [PLTC16].

There is not a fail proof formula for happiness, but there are some steps that can be followed to achieve it. This list is based on the "12 Steps to Happiness", created by Jurgen Appelo [App16].

- **Thank your coworkers** and always try to be appreciative. Appreciating the work of others and having your own work appreciated gets the morale of the team up and can result in the work environment being better overall.

- **Give a little present** to a team member or make it possible for team members to offer each other gifts. Gifting is an act that makes both the sender and the receiver happier, and can be turned into a vicious cycle of giving and receiving which results in increased happiness.

- **Help someone out** who is in need of assistance. Lending a coworker a hand makes both the helper and the receiver feel good, which in turn increases both their happiness.
- **Eat well** and make sure healthy foods are available for every co-worker.
- **Exercise regularly** and allow others to do the same.
- **Resting well and sleeping sufficiently** is one of the most suggested tips for living a happier and healthier life, which also shows its effects on the workplace.
- **Experience new things**, as the happiness gained from experiences is longer lasting than the one that is gained from material things.
- **Hike outdoors** in order to escape the work environment once in a while.
- **Meditate every day** or adopt regular mindfulness practices.
- **Socialize with people** and make it easy for coworkers to develop connections.
- **Aim for a defined purpose** and help people achieve their goals. People should not aim to be happy, but to do something meaningful. Happiness will then come naturally.
- **Smile to make everyone feel better**. Most good moments are associated with smiles, and having a smile in one's face, even if fake, can have a positive effect on that person's mood.

It can then be considered that the happiness is something that is only attainable with the work of both the developer and the team that contains it.

Happiness is, though, as seen, mainly an individual level construct and not exactly a team level construct. Thinking about a team as "happy" does not necessarily mean that every single person in the team is happy, that everyone is happy about the work the team is producing, happy to be a part of the team, or happy about the way the team is working. A team cannot be "happy" without its team members being happy in some kind of way. There are, then, many ways to examine this idea, of which many can be understood using existing constructs like the evaluation of cohesion of a team.

2.2.3 Team Cohesion

The cohesion of a group is a widely studied characteristic of team process [KI06]. There are many definitions for cohesion, one of which, defined by L. Festinger, is "the resultant of all the forces acting on the members to remain in the group" [Fes50].

One of the ways to evaluate the team cohesion is by asking each team member to fill out a questionnaire at pre-defined intervals of time. This interval can be, for example, at the end of each sprint, and the user would then be asked to evaluate the cohesion of the team for the last sprint. The questionnaire used in this project is based on a questionnaire made by Carless & De Paolo (2000), and is represented in appendix A.

The questions in the questionnaire used can be broke down into the components of cohesion.

- **Task Cohesion** – Questions 1, 2, 3 and 4;
- **Social Cohesion** – Questions 5, 6 and 7;
- **Attractiveness to Team** – Questions 8 and 9.

Each question in the questionnaire can then be assigned a score, from 1 to 5, which summed up represents the cohesion score for each team member of the team. Since there are 9 questions, the score ranges from 9 to 45. This score can also be broken down into the three components of cohesion mentioned above, so the Task Cohesion can have a score that ranges from 4 to 20, the Social Cohesion a score from 3 to 15, and, finally, the Attractiveness to Team a score from 2 to 10. The bigger these scores are, the bigger is the cohesion of a team. These values can then be used to compare how each team member feels about the cohesiveness of the team, how the cohesiveness of the team as a whole is improving as the sprints and time goes by, among others.

2.3 Quantified Self Tools

The data gathered in the realm of the Quantified Self movement can be parsed and analyzed, which allows for graphs and insights to be generated and presented to the end user. This can be achieved with the help of existing applications, which some of the most relevant will be described below.

2.3.1 Strava

Strava is an app created for cyclists and runners to track their activities, while allowing insights about their workouts and activities to be created. It works by using the phone's *GPS* or the user's favorite *GPS* device to map the route the user takes on a given training session and all the related information like speed, time, calories, between others. One of the main reasons this service became so popular was its emphasis in competition. Leaderboards are automatically created for a single route, which increases engagement and drives friendly rivalry. Users can also choose to create customized leaderboards, which increases the social interactions with their friends and relatives [Wes15].

2.3.2 Runkeeper

Runkeeper is a running community that allows users to track their workouts and set goals, by the means of *iOS*, *Android* and web applications. Their web application is limited in functionality, though, not allowing for real-time data tracking and analysis. Users are able to track their workouts, set goals in order to stay motivated, follow workouts custom-made just for them and share their data and achievements with other people. This sharing feature allows for the creation of competitions with friends and relatives, which ultimately makes the users of the service stay motivated and increase the chances of achieving their objectives [Run17].

Figure 2.4: *Runkeeper* (*Apple Watch* version) [Per15]

2.3.3 BACtrack

BACtrack is a company founded in 2001 in San Francisco, California that designs and sells breath analyzers – a family of devices that tests a person's breath in order to check for alcohol intoxication [Dic17a]. They have a range of smartphone breath analyzers that connect to *iOS* and *Android* devices by means of a *Bluetooth* connection [BAC17]. This connection is then used for many useful features, like registering the user's blood alcohol content in a log on the user's device and providing an estimate of when the user will be sober. This information can also be made easily available for other applications on the user's device.

2.3.4 Sleep Better

Sleep Better is an app by *Runtastic* for both *iOS* and *Android*, released in late 2014 [Kah], that allows users to track their sleep and improve their bedtime habits. It can be used to help users learn how their day time activities affect their sleep efficiency and wake up refreshed by triggering the alarm when they are on a light sleep phase [rG17].

It can also log the information to a centralized location on the user's device, in the case of *iOS*, where it can then be used by other applications or services for all kinds of purposes. The data logged is varied and can be useful for a multitude of different analyses. Some examples are the sleep efficiency, which is related to the percentage of time the user spent on light and deep sleep phases, and the time the user took to fall asleep after going to bed, which can be related to a lot of factors, including its tiredness [rG17].

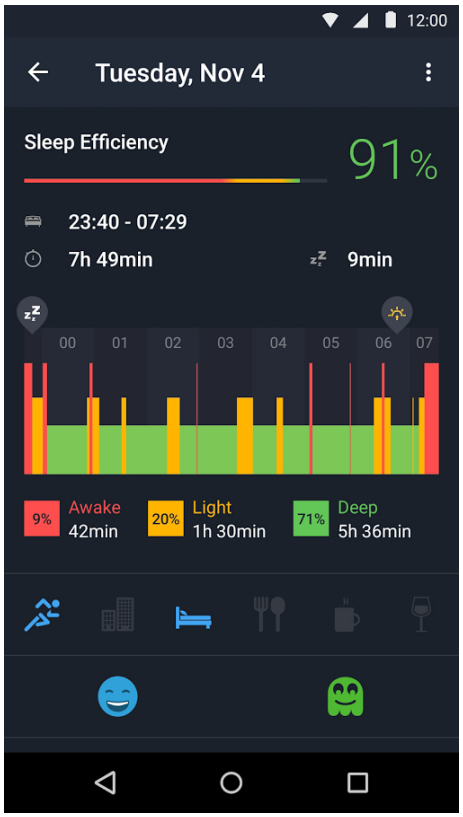


Figure 2.5: Sleep detail for *Sleep Better* (*Android* version) [ban14]

2.3.5 Apple Health

Health is a core application of the *iOS* mobile operating system, released along *iOS* 8 in late 2014, that provides a central location for all the health data collected by the iPhone, *Apple Watch* and related third-party applications. This way, all information can be viewed and analyzed by the user on a single place, and applications that require the usage of health information may gather it using a unified interface, as long as allowed by the user [App17].

It should be noted that this application is only available for devices running *iOS*, and there is no comparable service for *Android* devices. *Google Health* provided a subset of the same features, but was discontinued in 2013, with the reason that it did not have the number of users *Google* had initially hoped [Goo13].

Figure 2.6: *Apple Health* "Today" screen

2.3.6 Moves

Moves is an activity and fitness tracking app acquired by *Facebook* that automatically records any walking, cycling and running its user does, without requiring any user intervention. It then creates a diary-like view of the day of the user, helping the user think about his life in a new way, leading to healthier habits [Fac17].

2.4 Source Control Systems

Source control systems are systems that manage changes and revisions of documents, files, websites and other information. It is considered by some a sort of "time machine" for code [PCSF08].

> "The central challenge in managing software development is scaling the change process up to large numbers of possibly geographically-distributed software developers without sacrificing quality or introducing undue overhead." [dAS09a].

This type of systems are very important for software developers. Developers need to be able to track and log their work, since changes done to an application may need to be reverted, and this can be easily be done with a source control system by rolling back to a previous commit. The process of working on a single project with other developers is also greatly simplified, as everyone can

work on a separate "branch" (a separate version) of a single repository, which can then be merged later when the feature in development is complete.

2.4.1 Centralized

Centralized source control systems require that the developers have a central server where the repository data is stored. This allows for a more fine-grained control of access, but takes some freedom away from the developers themselves. This kind of software control systems allows for restrictions on the layer of the files, which means that a developer may have access to a file but another may not, which can be useful when dealing with sensitive data. Unfortunately, every time a developer wants to commit their changes or work, a network connection is required, which may not be ideal to all cases. This form of source control system precedes the more recent and more used decentralized systems, even though they are not necessarily better in all cases [dAS09b].

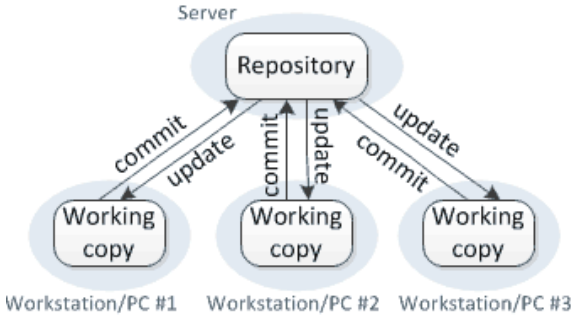


Figure 2.7: Diagram demonstrating the workflow of a centralized repository [Ern16]

RCS

RCS, or Revisions Control System, is a system that "automates the storing, retrieval, identification and merging of revisions". Its initial release was in 1982, as part of the *GNU* Project, a free software and mass collaboration project [Sta99]. It was developed with files whose content changes frequently in mind, and not specifically for source code [Fou15], which made it superseded by *CVS* for software developers.

CVS

CVS, or Concurrent Versions System, was one of the earliest version control systems to be widely used, and one that is still being actively used in some projects, despite its age. It was initially released in late 1990, and its last stable release was in 2008, which makes it abandoned, even if it is not officially announced as so [dpr08]. It is based in the *RCS* and extends this system by changing the notion of revision control from a collection of sparse files to a hierarchical tree of files and directories, each containing different versions of files with distinct timestamps [B+90]. This makes this system more suitable for source code than its predecessor.

Subversion

Subversion is another example of a free and open source version control system. It was originally developed by *CollabNet*, and saw its first release in late 2000, but is, at the moment of writing, being maintained by the *Apache Software Foundation* [Fou16]. Like *CVS*, it manages files, directories, and the changes made to them since the repository was created, allowing the history of the files to be analyzed and older versions of those to be recovered [PCSF08].

Subversion was designed to be a sucessor to *CVS*, trying to be similar in design while attempting to avoid all of its design flaws. It also aims to be compatible with *CVS*, in order to better attract users of that system, by allowing them to migrate their old repositories to this system. While *CVS* tracks modifications on a file-by-file basis, *SVN* introduces the notion of commits, which are a group of modifications with a name, resulting in an easier to follow history of the project, with the specific changes being easier to pinpoint in time. Also, Subversion supports atomic commits, and *CVS* does not, which prevents the loss of files in the event that two developers try to commit code to a central server at the same time [CS02].

2.4.2 Decentralized

Decentralized source control systems differ from centralized in the fact that they are more relaxed, since there is no need for a central, master repository, unlike what happens with centralized source control systems. Each developer has its own, private copy of the repository and every copy is a first-class repository in its own right, with the full commit history, and where the user has full read and write access to it [dAS09a].

In addition, a user can commit work to its local repository even when offline, and then sync the changes with a remote server at its own desire. A repository may even be only stored on a user's computer, and never be synced with a server, if the user so wishes. This way, commit history is not lost and the information gathered by commits is more reliable. This also allows for a much more flexible system, opening up more use cases when compared to centralized source control systems.

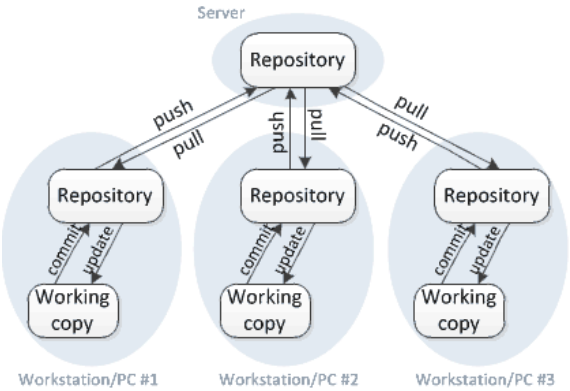


Figure 2.8: Diagram demonstrating the workflow of a decentralized repository [Ern16]

Git

The *Git* source control system is a free and open source decentralized source control system designed to handle all kinds of projects, optimized for speed and efficiency. *Git* is natively supported in *macOS*, *Linux* and *Solaris*, and supported in *Windows* by the means of *Cygwin*, a wrapper for running *UNIX* applications and utilities under this operating system [TH17]. The usage of *Git* in *Windows* is not as easy and intuitive as in the other operative systems, but this does not appear to have slowed down *Git*'s popularity increase.

A user can get started with *Git* by initializing a local repository in a computer or cloning an existing one from a remote server. Both repositories are full-fledged repositories, with all functionalities available. Files and directories can then be easily added, modified and deleted on the repository, and these changes can then be committed or discarded. If the user desires to sync the changes with a server, that can be easily done, but if that feature is undesirable or if an internet connection is not available, a sync is not required.

One of the main advantages of *Git*, compared to other source control systems, is that it includes a staging area, where commits can be reviewed and changed before being complete. That also makes it possible for only some files to be quickly staged and committed without fumbling with the other files in the repository [TH17].

Mercurial

Mercurial is also a free and open-source decentralized source control system, sharing most of its main features with *Git*.

Git and *Mercurial* are very similar in nature, but *Mercurial* has some advantages, like being platform-independent and easily extensible [Com17]. *Mercurial* requires no additional software or dependencies in order to run in *Windows*, which is a great advantage over *Git* for developers that work mainly with this operating system.

2.5 Developer Software

There are a lot of tools that software developers use in their daily lives. Developers can easily lose track of what needs to be done if there are no tools to help them. Also, tools are needed for code collaboration, since there is the need for developers to work with colleagues and have their work logged. Some of those tools are relevant to this topic, as they allow us to extract information and metrics relevant to this work.

2.5.1 JIRA

JIRA is an issue tracking system developed by *Atlassian Corporation*. Its development started in 2002 and it is, at the time of writing, being actively maintained. It is not free software, but

has a starter license with pricing that is reasonable even for hobbyists and small companies. Its most common use case is for software bug tracking, but due to improvements and its advanced customization features, it is highly suitable for some kinds of ticketing systems (work orders, help desks, etc.) and project management. Some of these use cases are even officially supported with (paid) plugins made by *Atlassian* itself. *JIRA* also provides a mature and powerful toolset for local customizations which can be applied in order to meet specific project needs. These customizations include but are not limited by custom fields, issue types, workflows, notifications, and user entry screens [FKL13].

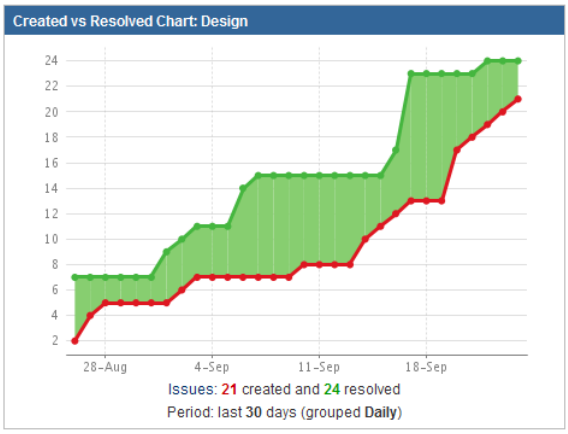


Figure 2.9: *JIRA*'s Created vs Resolved issues chart [Atl17a]

2.5.2 GitHub

GitHub is a collaborative code hosting platform, which uses the Git decentralized version control system. Using this service, users are able to publish the source code to their projects and collaboratively work with others. Everyone is able to create an account and create public repositories, but the creation of private repositories is only available for paid subscribers.

In addition to allowing its users to host their code, review it with the help of other users and track issues, *GitHub* can also be considered a social network, since users are able to subscribe and be informed of what is currently happening in the platform by the means of "watching" projects and "following" users. This results in a feed of information, generated from the activity of the user and the users it is connected with. The users can also personalize their profiles with their information and the feed of information mentioned. *GitHub* is one of the largest code hosting platform in the world, having surpassed the 10 million repositories mark in 2013 [Bri13].

The popularity of the platform, its collaboration and social features and the ease of use of data by the means of an easy to use interface has made *GitHub* a very appealing platform for researchers to study.

Examples of studies done using data gathered from *GitHub* have focused on how developers use the social features of this platform, the development practices of the service's users and its network structure [KGB+14].

In the specific case of this work, there is a great deal of relevant data that can be gathered from their *API*s. For instance, the number of commits, lines of code, code coverage and code quality, between other metrics, can be used and correlated in order to generate reports and give insights about a person's productivity and happiness.

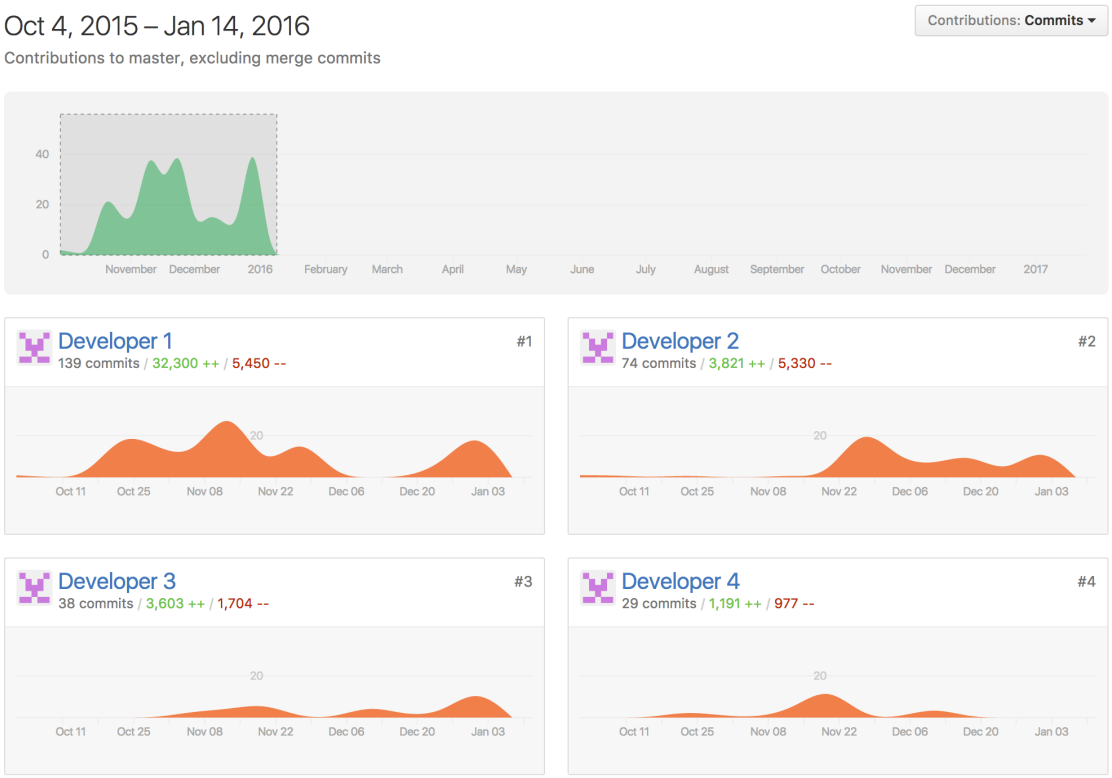


Figure 2.10: *GitHub* statistics for a college project

2.5.3 Bitbucket

Like *GitHub*, *Bitbucket* is also a collaborative code hosting website, but instead of only supporting the *Git* version control system, it also supports *Mercurial*.

It is overall a more attractive service for new users than *GitHub*, since private repositories are free, albeit with a user limit, and it is cheaper for teams.

An *API* is also available, where a lot of interesting data can be gathered, both from the users themselves and from the teams, much like what happens with *GitHub* [Atl17b].

2.5.4 Bugzilla

Mozilla is the company behind the open-source Web browser *Mozilla Firefox*. When their developers needed a bug tracking system, they built *Bugzilla*, also an open-source project. It is still being actively maintained, even though it suffered some rewrites — *Bugzilla* was originally written in *TCL*, and was later ported to *Perl* upon its 2.0 release.

Bugzilla is a bug and issue-tracking system, which allows both individual and groups of developers to keep track of pending problems with their products. It gained popularity since it was released at a time when defect-tracking systems were very expensive, and *Bugzilla* was free to use and open-source [Fou06].

At the moment of writing, many organizations, both open source (*Apache*, *Linux*, *Open Office*) and public / private (*NASA*, *IBM*) are using *Bugzilla* [SC05].

Some of its main features are its powerful searching capabilities, email notifications, robust backend, extensive configurability and four distinct interfaces — web, *XML*, email and console.

2.5.5 Pivotal Tracker

Pivotal Tracker is a project management web-based tool for agile software development teams. It was launched in 2006 by *Pivotal Labs* and is being actively developed and maintained at the date of writing.

It includes most of the tools required for agile planning, from story backlogs to utilities that assist teams in planning their sprints [Lab17].

A large number of developer tools that have not been referred here exist, but the most relevant were already referred [Bur16].

2.6 Smartphones

Most, if not all software developers carry around mobile devices with them everyday. Smartphones are, roughly speaking, personal digital assistants (*PDA*s) equipped with integrated wireless connections that are capable of handling phone calls and text messages [PC07]. These are very popular among people that work with technology everyday, which makes smartphones a good deployment target for applications that measure the Quantified Self. There are various operating systems that run on smartphones, which are not necessarily better or worse than the others.

2.6.1 *Android*

Android is an open source software stack of software for mobile devices, and at the time of writing the most used mobile operating system, by far [IDC16a]. It is composed by an operative system (based on *Linux*), middleware (which are frameworks that aid developers in the development

of applications) and key applications, which handle basic functionalities like phone calls, text messages and web browsing [Dev11].

Developers have access to the same tools and *API*s used by the phone's core applications, so any programmer can write an application that replaces one of the functionalities of the system, making the *Android* platform very open and developer-friendly.

Applications developed for *Android* are mainly written in *Java* with the aid of the *Android* framework, with the exception of games that are normally written in *C/C++*, as these languages are more normally efficient and faster than *Java* for applications that require raw performance.

2.6.2 *iOS*

Apple's *iOS* is another major mobile operating system, which comes pre-installed on *Apple*'s *iPhone*, *iPad* and *iPod Touch* family of devices and can not run or be installed in any other kind of devices. *iOS* is based on *macOS*, the operating system used by the *Mac* family of computers. Both are based on the same core, *Darwin* [AMN+13].

This operating system is not as open as *Android*, since the use of some *API*s is restricted. Any applications that attempt to use them will not be able to be distributed on the *App Store*, the only source where users can install new applications.

iOS applications are normally written in either *Objective-C* or *Swift*.

2.6.3 Windows Mobile

While not nearly as popular as the other two operating systems [IDC16a], *Windows Mobile* is also a notable contender. One of the main advantages of developing applications for this operative system is that they can also run on *Windows*-based PCs without significant changes to the code of the application.

Universal Windows Platform applications are written in either *C#* or *Visual Basic*, with the aid of the *.NET Framework*.

2.7 Wearables

Wearables refer to a wide range of technologies or small computers that are designed to fit within a person's style and can be worn on the body [GLL15]. There are a lot of types of wearables, of which the most notable are fitness trackers are smartwatches. Usually, these kind of devices (especially smartwatches) are constantly connected to mobile phones in order to export health data, receive notifications, get data relevant to the user, between other features.

2.7.1 Fitness Trackers

A lot of fitness trackers are available on the market, of which some notable examples are the ones designed by *Fitbit* and *Jawbone*, since they are, at the time of writing, the most popular brands. This is also the kind of wearable devices that are the most popular between the young and healthy users [GLL15]. While different fitness trackers are specialized in different functionalities and have different specifications, they are all wearable devices and all track at least the same basic parameters - calories burned, heart rate, number of steps taken, sleep rhythm, between others [ZP14].

Jawbone is a brand that designed and builds one of the most popular fitness trackers, the *Jawbone UP*. It is designed to be lightweight, comfortable and stylish, and passively tracks data like fitness activity, heart rate and sleep patterns. There is an app for *iOS*, *Android* and *Windows* that handles the communication between the mobile device and the fitness tracker and allows users to track theirs and their friends' activity, as long as they allow it [Jaw17].

Fitbit is another brand of fitness trackers, whose products are functionally similar to *Jawbone*'s. There are some differences in what both track, though. For example, while the *Fitbit* tracks the number of stairs climbed, the *Jawbone* tracks the mood of the person. There is also a set of *iOS*, *Android* and *Windows* applications that offers features similar to the *Jawbone*'s, but also suggests workouts to the user, based on activity and goals [Fit17].

Figure 2.11: *Fitbit* watch face, showing the time, steps and heart rate [Tec17]

2.7.2 Smartwatches

Smartwatches work as an extension of mobile phones, allowing for some functionalities like phone calls, messages and notifications to be redirected directly to the user's wrist, which allows for a faster and easier communication. Apart from this, some of these devices also allow for small apps to be run, from simple health tracking to games.

The *Pebble* was one of the first smartwatches that reached a broad range of users, being one of the most successful campaigns in the history of crowd funding [RPP14]. The device's first version was limited in features and capabilities, since it only featured an e-paper black and white display, limited capabilities with regards to the applications that could be installed and did not feature any health tracking features. Starting with its second iteration, it had health tracking features,

whose notable examples are step count, heart rate monitoring and sleep tracking. Unfortunately, the hardware of the *Pebble* is very weak compared to its competitors and its operative system is also lacking. Its parent company has been purchased by *Fitbit*, which have decided to no longer manufacture Pebble smartwatches, and instead switch the *Pebble*'s team focus to their own Fitbit products. Support for these devices has also since ceased [Peb16].

Android Wear smartwatches are based on *Google*'s *Android* operating system, and only work at their full potential when paired with an *Android*-based smartphone. Only basic support is provided when paired with an *iPhone*-based smartphone, and there is no support for *Windows Phone* devices. Most of the *Android Wear* smartwatches provide health tracking features, much like *Pebble*, which are useful in the realm of Quantified Self. Unfortunately, the first release of *Android* Wear was in early 2014 and there have been no major upgrades, which has resulted in the sales of *Android* Wear watches being low. *Android Wear* 2.0 should be available to a wide public in the near future, but at the time of writing only few watches had support to it. [And17].

On the other hand, *Apple Watch*es run a variant of *Apple*'s *iOS* operating system, *watchOS*, and can only be used when paired with *Apple iPhone* smartphones. Unlike *Android Wear* smartwatches, which provide basic support for *iPhone* devices, the *Apple Watch* provides no support for *Android* or *Windows Phone* devices. That being said, *Apple Watch* devices are the devices of the smartwatch family that have, at the time of writing, the biggest market share, with the number being more than 70% [IDC16b]. *Apple* was also the biggest wearables vendor in the first quarter of 2017, which further indicates that investing on *Apple Watch* is one of the safest choices [MM17]. The *Apple Watch* provides a myriad of health tracking features by default, which are automatically tracked and registered on the paired phone's *Health* application. This data can then be collected by any application that desires to use it, as long as allowed by the user. It should be noted that this family of devices provides no sleep tracking capabilities by default, but this functionality can be provided by third-party applications.

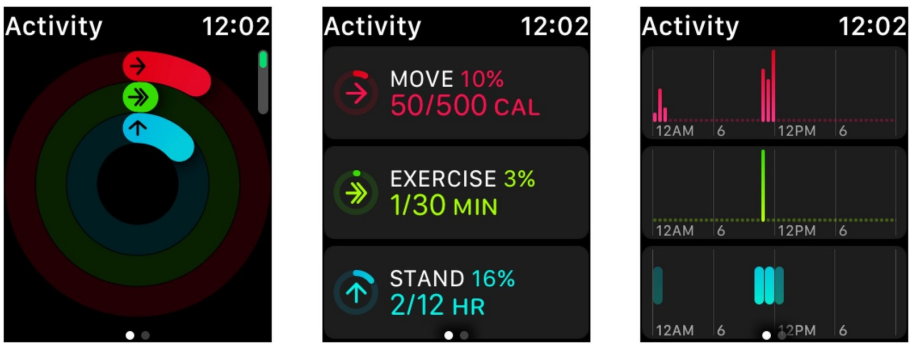


Figure 2.12: *Apple Watch*'s activity circles [LF16]

Finally, *Samsung* also builds smartwatches, namely their *Samsung Gear* line of products. These devices run a wearable version of *Tizen*, and are not compatible with any *Apple Watch* or *Android Wear* applications. Even though these devices also provide all of *Apple Watch* and *Android Wear*

health tracking features by default, their choice of applications is reduced since they have to support both *iOS* and *Android* and do not have *SDK* support from *Apple* nor *Google* [Sam17].

2.8 Potentiality of the Analyzed Tools and Tecnologies

After the analysis of the state of the art, it can be concluded that there are various popular tools and technologies that can be useful in the realm of Quantified Self and Quantified Team. They provide data such as sleep habits and code metrics that can be analyzed and used as a whole in order to get conclusions about the user's productivity and habits.

Chapter 3

Data Visualization and Analysis

"Data Visualization is the representation of data in a graphical format" [Ins17]. This project involves acquiring a large number of data from a variety of sources, which then needs to be able to be visualized by the end users. Analyzing data in a graphical format is often easier than on a textual format, since, for example, some correlations can be visualized just by looking at the generated graphics. "Visualizations are the single easiest way for our brains to receive and interpret large amounts of information." [Ins17]

In this chapter, the methods of data visualization used will be described, along with the reasoning behind the choices of graphs. Choosing the right graph can make the difference between a person being able to understand, or not, the data.

There are primarily two types of data, or messages, that are used in tandem in order to generate a relevant visualization. These are categorical and quantitative. Categorical data describes the nature of the information being analyzed, like "Sprint Points" and "Time". On the other hand, quantitative data are numerical values, like "60" and "125", that can refer to many concepts, like an age or time, depending on the categorical data associated [Ins17].

Choosing the right graph is a decision mostly influenced by the type of data that is being displayed.

There are seven main different types of messages that may be communicated using a chart, which mainly differ in the way separate values relate to others. Unless the data set that is being dealt with is too specific, it should fit one of these categories, as defined by Stephen Few in "Enie, meenie, minie, moe: selecting the right graph for your message" [Few04]:

- **Nominal comparison**, which refers to a comparison of subdivisions of the measures being analyzed, without any specific ordering;
- **Time-series**, which designate multiple instances of one or more measures displayed over time;

- **Ranking**, which specifies subdivisions of measures ordered by groups that refer to their sizes;
- **Part-to-whole**, which refers to subdivisions of measures, that add up to the whole;
- **Deviation**, a category in which data is displayed compared to a reference measure;
- **Frequency distribution**, in which values of a data set are divided in specific intervals;
- **Correlation**, where a pair of sets of measures are compared in order to determine if as one set goes up or down, the other follows accordingly, and if yes, how strongly that occurs.

In this project, data sets that fit within two of these categories are used, namely Time-series and Correlation.

3.1 Time-series

There are multiple types of charts that can be generated when data fits this category, depending on the expected end result. When the overall patterns are to be emphatized, line charts should be used. Alternatively, when individual values are to be emphatized, bar charts should be used. The major requirement for this data category is that the variable "time" be always present and placed in the horizontal axis of the chart [Few04].

In this project, charts with data adhering to this category are mostly used when team data is displayed, in order for the team leaders to be able to acquire data useful for their jobs, like the variance of relevant team data over time.

3.2 Correlation

In order to show correlations between a pair of variables, a scatter plot is the most desirable choice, which is built using points and a trend line. In the case a scatter plot is unfamiliar to the end users reading the data, bars may also be used, but this method of visualization is not as common (and was not used in this project) [Few04].

In this project, charts with data adhering to this category are used when personal, lifestyle data is displayed alongside work data, in order to aid the user in finding a correlation – or the lack of it – between these two kinds of data.

Whenever a chart of this type is shown, the user is also allowed to ask the system to generate an analysis about the data set. While that a chart of this type normally allows a user to acquire some data simply by looking at it, it is sometimes not easy to find correlations between data sets by just doing a visual analysis. Ways to better analyze this data, and not just visualize it, are described next.

Data analysis is the process of discovering useful information, aiding with the making of decisions and suggesting conclusions by using previously acquired raw data and converting it.

There are various tasks that can be performed with acquired data, one of which is attempting to find correlations between two variables.

3.3 Pearson Correlation

Pearson's coefficient of correlation, also called bivariate correlation, Pearson correlation coefficient and Pearson product-moment correlation coefficient, is a "measure [of] the strength and direction of linear relationships between pairs of continuous variables" [Lib17]. It was first discovered by Bravais in 1846, but was only documented and described by Karl Pearson 50 years later, in 1896, hence its name [JH11].

When applied to a data set, it produces a value between -1 and 1. Values bigger than 0 represent a positive linear correlation, and 1 represents a perfectly positive linear relationship. Values smaller than 0 represent a negative linear correlation, with -1 representing a perfectly negative linear correlation. Finally, a value of 0 means that there is no linear correlation at all.

This coefficient of correlation can then be calculated using the following equation:

$$\rho = \frac{\text{cov}(X,Y)}{\sigma_x \sigma_y} \tag{3.1}$$

In the previous equation, *cov* represents the covariance and σ the standard deviation [Wan13], where the covariance is a representation between two random variables of their joint variability [Ric07] and the standard deviation is a measurement that quantifies the variation of a data set [BA96].

In order to analyze the value, the following guidelines were used, based on the guidelines determined by Cohen J. [Coh88]:

Range	Strength
... < $\lvert\sigma\rvert$ < .1	No Correlation
.1 < $\lvert\sigma\rvert$ < .3	Weak Correlation
.3 < $\lvert\sigma\rvert$ < .5	Moderate Correlation
.5 < $\lvert\sigma\rvert$ < ...	Strong Correlation

Table 3.1: Pearson's Coefficient of Correlation Strength Assessment Guidelines

This method of calculating the correlation was used whenever a correlation value was to be calculated, unless non-quantitative variables were used, which only happens once.

3.4 Median

Pearson's coefficient of correlation only works on quantitative variables, that is, variables that can have a numeric value associated to them [otWoE17]. This is the case for the majority of data sets used in this project, but there is a notable example – the weather. While that the number of weather states is finite, and a number could be associated to each one of them, weather can not be ranked. Considering that there are 40 possible weather states and associating, for example, the *Sunny* state with the value "1" or "15" could provide very different Pearson Correlation values, which is not acceptable.

With that in mind, another solution had to be found in order to try and find relations between weather values and other types of data.

It was decided to create a simple ranking for weather, where the values would be ranked by the value of the dependent variable. The median value of the dependent variable was also measured. The weather types which had associated values that were higher than the median would then be considered more productive than the ones below that value.

That way, it was possible to give an insight about how (and if) the weather affected the productivity of the user, even if it could not be done the same way as the other data types.

Chapter 4

QuantiDev

In this chapter, the identified problem is described and a solution, by the name of *QuantiDev*, is characterized.

4.1 Problem

The productivity and efficiency of a software developer is not constant along the time. There are various factors, already described before, that can affect them, since it is a job that requires focus and, at times, creativity. Most people do not know for sure which aspects influence the ability to do their jobs, or which aspects may even have an influence on it, since it can also vary from person to person. People just notice that their productivity is affected – be it higher, or lower. It would be very useful to know why and when such happens.

After the analysis of the state of that art in the realm of this project, there is something that can be concluded. There was, at the moment of writing, no solution tailored to professionals in the software development industry that allows them to accomplish the feat of analyzing and generating data about their productivity and what could possibly be affecting it.

There were also no tools that give agile team leaders the possibility to analyze their team's productivity and communication health. This can sometimes be a more important metric than raw data like lines of code and number of commits, since these do not necessarily translate to a better quality of work, productivity or work environment.

4.2 Solution

Various applications were developed in order to reach the desired goal, which are, as described, a web backend, a web application and two mobile apps. These applications allow for the gathering and analysis of Quantified Self data, while a web backend handles the collection of data from

external services used by software developers and (optional) collection of team data. This data can then be used with the objective of attempting to improve the productivity and work quality of software developers.

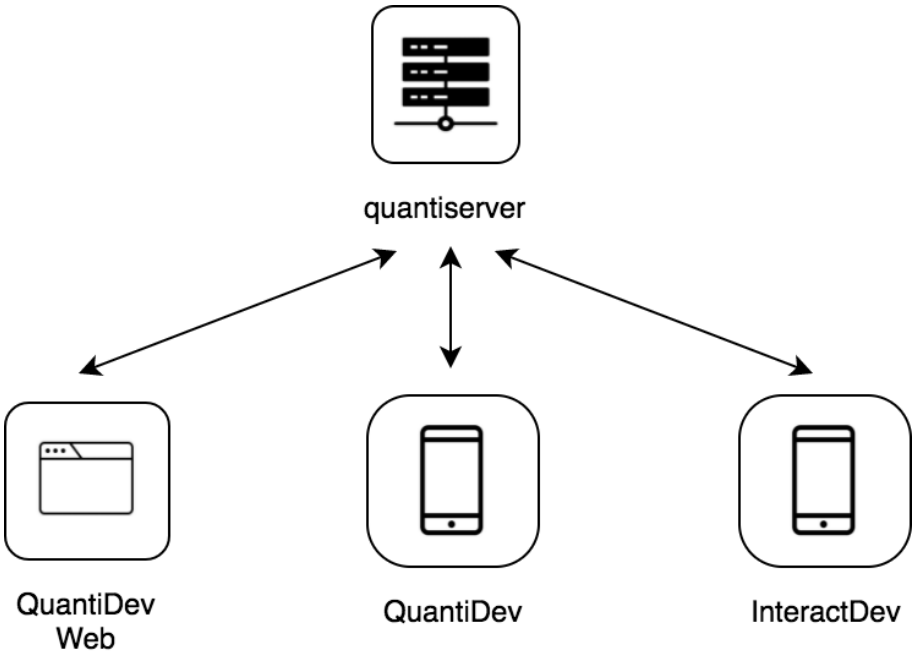


Figure 4.1: Components of the *QuantiDev* platform

4.3 Web Backend

A web backend, codenamed as *quantiserver* and written in *Node.js*, handles requests from all user-facing applications. Those requests can be simple *CRUD* operations, or more complex inquiries, such as the median of story points accomplished by a given team on a single day.

This type of application is required since there needs to be a central location for the storage / correlation of data for members of the same development team. Also, this same backend eases the management of sprints and the gathering of data from external services (like *GitHub*, *JIRA*, ...) since processing power, internet connectivity and battery life is limited on mobile devices and as thus should be saved as much as possible. Acquiring this data on predefined intervals using a server and having the clients acquire this data when they need it from the server makes this task much easier.

This application also serves as a *OAuth* proxy. *OAuth* is an open standard that enables users to grant services access to their information on other websites without disclaiming their access details [Gor12]. Unfortunately, the *GitHub*'s implementation of *OAuth* does not allow for a full client-only, browser-based flow. This would be required for *QuantiDev Web* to communicate with *GitHub* without resorting to any workarounds.

This backend uses a *MongoDB* database as its data store. This system was chosen since it is a powerful, flexible and scalable *NoSQL* database. Instead of working with the concept of rows, like traditional database systems, data is stored using documents, which don't even need to have a well-defined schema. These documents have a *JSON*-like syntax, which makes them easy to work under *JavaScript* (and, consequently, *Node.js*) while also allowing for more complex data representations, like arrays and dictionaries, under the same document [Cho13].

Applications may communicate with this backend through a well-defined *REST*-based *API* without the need of any additional credentials, which means that any developer may create an application that consumes or feeds data to the platform. This is the exact same *API* that powers both mobile and web applications, and is described in appendix B.

4.3.1 MongoDB Schemas

Even though *MongoDB* is a schema-less data store, applications may still choose to use schemas. A schema represents the facts that can be inserted into a database, and constraints the inserted documents to those facts [IL82].

As specified before, this application makes use of *MongoDB* as its data store, and the schemas used in order to aid the storage of documents are as follows. All types represented are primitive *JavaScript* types, except for *ObjectId*, a *MongoDB* type, which represents a pointer to a different document (similar to a foreign key in relational databases). The schemas used are represented in appendix C, and a UML class diagram of that data store is represented next.

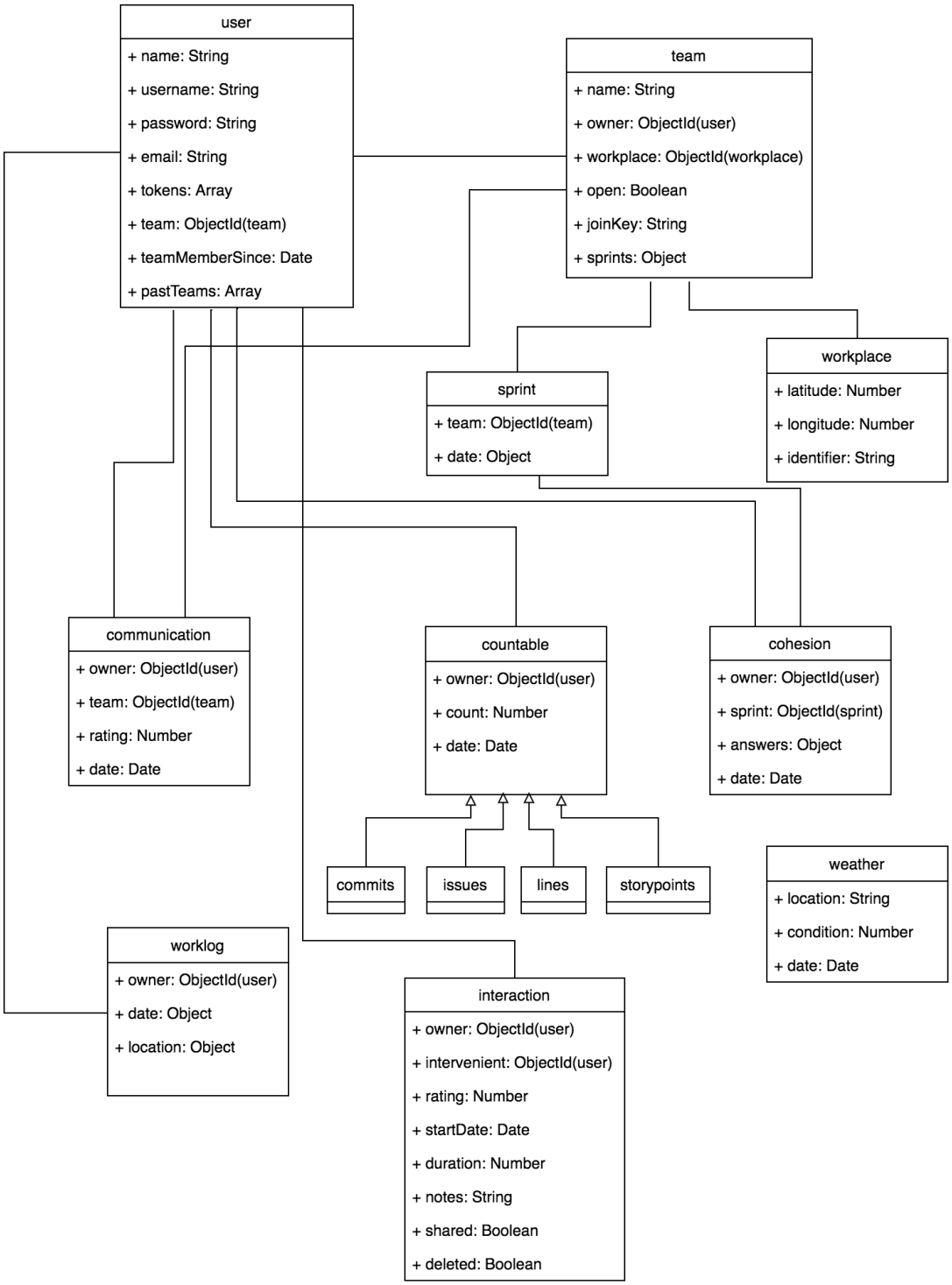


Figure 4.2: Class *UML* diagram for *quantiserver*'s database

It can be noted that the schemas for *commits*, *issues*, *lines* and *storypoints* are the same. This is done on purpose, and they are all implemented based on the same base schema, namely *Countable*. This makes it trivial to create new schemas that store countable values, as is the case with the previously mentioned ones. The system was programmed this way in order to ease the addition of new features to the system (which makes it easily extensible).

4.3.2 Third-Party Libraries

A number of third-party libraries where used in order to aid the development of this application:

- **bcrypt**, which is a password hashing library, used to aid in the storage of passwords on the database [def17].
- **btoa**, a *polyfill* for *Node.js* that adds the native *ECMAScript* function call *btoa()* to *Node.js*. It is used to convert String objects to Base-64 encoded strings. In this project, it was used to aid in the transmission of data to *GitHub*'s servers [coo14].
- **cors**, a library that enables cross-origin *HTTP* requests in *Node.js* applications. Specifically, it was used in order to allow the *QuantiDev Web* to communicate with the *quantiserver*'s *API* [dou17a].
- **crypto-extra**, a set of functions that build on top of Node.js' *crypto* functions in order to ease the process of generating cryptographically secure strings. It is used to generate secure strings, like user tokens [jso17].
- **express**, a fast and simple web development framework for *Node.js*. It is used as the base framework of the project, handling all the *HTTP* requests and calling the correct functions depending on the *HTTP* routes requested [dou17b].
- **geolib**, a library that provides an application with the ability to perform a myriad of geospatial calculations. It is used to calculate distances between two points (the user and its workplace, for example) [man16].
- **github**, an abstraction that allows for *GitHub*'s *API* calls to be performed as a *JavaScript* function call. It is used in order to get user data from *GitHub* repositories [kai16].
- **lodash**, a *JavaScript* utility library that adds utility functions that allow for an easier manipulation of *JavaScript*'s types and objects. It is used in various places throughout the code, with no specific use case [kai16].
- **moment**, a date parsing, validation and manipulation library. It is used whenever date manipulation and parsing is required, which is in a lot of places throughout the code (for example, whenever data is saved in the database) [ich17].

- **mongoose**, a *MongoDB* object modeling tool that assists in the connection of a *Node.js* application to MongoDB databases. It is used in all the *CRUD* calls to *quantiserver*'s database [vka17].
- **node-schedule**, a job scheduler for *Node.js* that allows for custom jobs (tasks described by the developer) to be ran at specific dates. It is used to automatically get data from external services and to create new sprints when time is due [sgi17].
- **passport**, a middleware that handles authentication for *Node.js* applications. A middleware is a piece of code that sits between other two, acting like a pipe, validating and/or modifying data. It is used whenever any functionality related to the authentication of an user is required [jar17].
- **pivotaltracker**, an abstraction that allows for *Pivotal Tracker*'s *API* calls to be performed as a *JavaScript* function call. It is used in order to acquire information from *Pivotal Tracker*'s boards [smi15].
- **randomstring**, a simple library that generates random strings. It is used to generate a team's join key, where the key needs to be random but readable [eli16].
- **unirest**, a set of lightweight *HTTP* libraries for multiple languages, where the *Node.js* package was used. It is used for communication with external servers, of which an example are *Bugzilla* servers [nij16].

4.4 QuantiDev for *iOS*

The *iOS* part of this project is an application, written in *Swift 3*, which allows for the acquisition, generation and visualization of personal data.

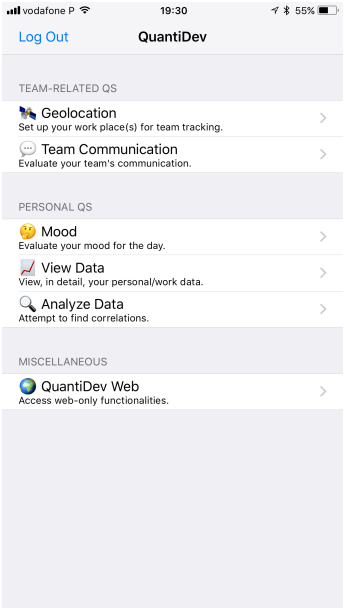


Figure 4.3: Main view for *QuantiDev*

4.4.1 Requirements

There are three different types of actors in the mobile app of *QuantiDev*, the unauthenticated user, the authenticated user, and a team member, which is an authenticated user that belongs to a team.

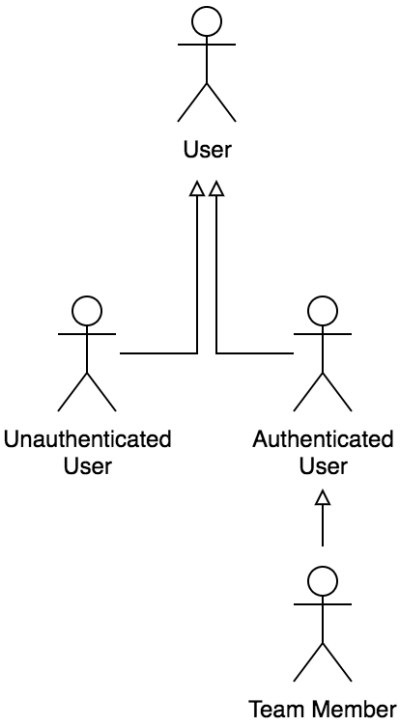


Figure 4.4: Users *UML* diagram for *QuantiDev* for *iOS*

Identifier	Description
Unauthenticated User	May only perform authentication and registration calls to the system.
Authenticated User	Authenticated user. May perform all functions allowed by the application, except the ones that require the user to be a member of a team.
Team Member	Authenticated user, belonging to a team. May perform all functions allowed by the application.

Table 4.1: Actor Description for *QuantiDev* for *iOS*

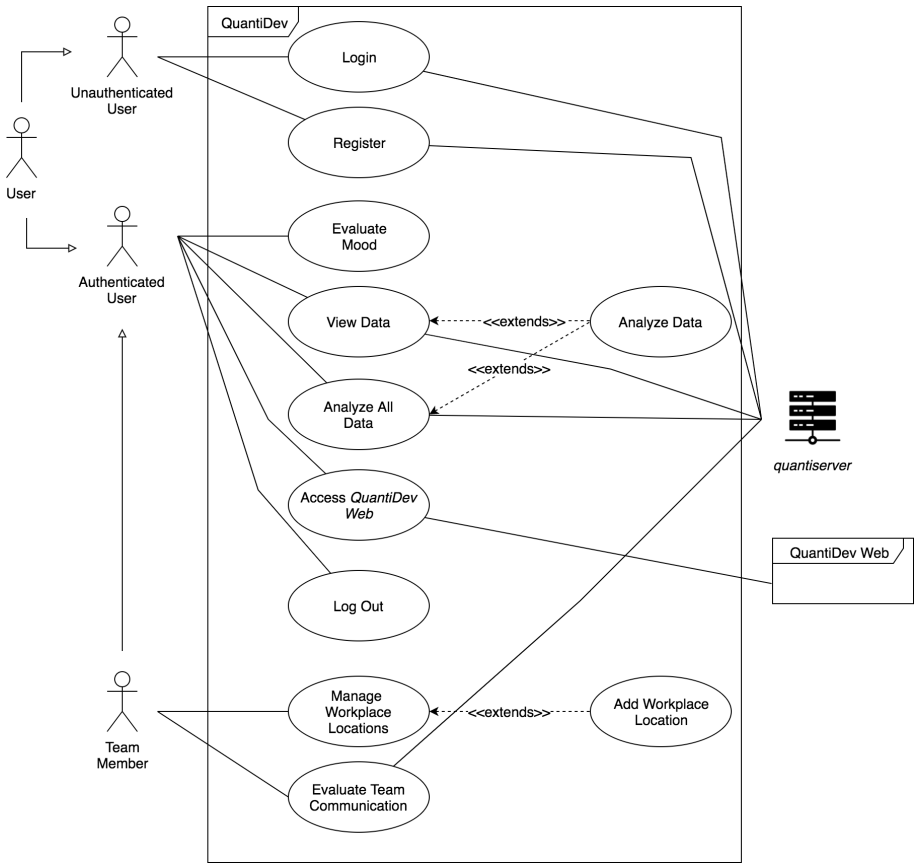


Figure 4.5: Use Case *UML* diagram for *QuantiDev*

Identifier	Name	Priority	Description	Interface
US101	Login	High	As an unauthenticated user, I want to login to the system so that I can use the application.	F.1
US102	Register	Medium	As an unauthenticated user, I want to create an account with the system in order to be able to login.	F.1

Table 4.2: User Stories for an unauthenticated user

Identifier	Name	Priority	Description	Interface
US201	Evaluate Mood	High	As an authenticated user, I want to evaluate my mood so that I can use this data to generate charts about myself.	F.2
US202	View Data	High	As an authenticated user, I want to generate charts and reports that relate my personal data with my work-related data so that I can visualize this data and try to find correlations therein.	F.3, F.4 F.5
US203	Analyze Data	High	As an authenticated user, I want to be able to generate a simple analysis about personal data and work-related data selected by me so I can acquire an insight on if a given metric about myself or my environment affects my performance.	F.5
US204	Analyze All Data	Medium	As an authenticated user, I want to be able to generate a simple analysis about all the personal and work-related data acquired by the platform so I can acquire an insight on what may be affecting either positively or negatively my performance.	F.6, F.7
US205	Access *QuantiDev Web*	Medium	As an authenticated user, I want to easily access the Web portion of the platform so that I can perform functionality that is not available in the mobile app.	F.8
US206	Log Out	Medium	As an authenticated user, I want to be able to log out from the system so I can terminate my session with the sytem.	F.9

Table 4.3: User stories for an authenticated user

Identifier	Name	Priority	Description	Interface
US301	Manage Workplace Locations	High	As an authenticated user, I want to view the locations that are registered as my workplace so that I can know what they are and keep this information up to date.	F.10
US302	Add Workplace Location	High	As an authenticated user, I want to register a new location as one of my workplaces so my work status can be automatically tracked.	F.11
US303	Evaluate Team Communication	High	As an authenticated user, I want to evaluate my team's communication for the present day so that this status can be shared with my team leader.	F.12

Table 4.4: User stories for a team member

4.4.2 Features

After the user logs in, the main view is presented, which serves as a main menu of sorts for the application 6.1. From there the user is able to perform all the functions available in the application by tapping each of the table cells.

The first row, "Geolocation", allows the user to set up work places. After a work place is set, the platform will be able to track when the user enters or leaves the workplace (F.10).

The next row, "Team Communication", enables users to evaluate the communication of their team for the current day, based on a scale from 1 to 5 (represented by smileys, which range from very sad to very happy) (F.12).

The following row, "Mood", allows the user to evaluate his personal mood for the current day, based on a scale from 1 to 5 (with the same representation as Team Communication) (F.2).

The subsequent row, "View Data", presents the user with a view where two different data sets can be compared in order to generate a chart (or in the specific case of the weather, a table) with the intent of aiding the user in finding a correlation between the specified sets of data (F.3).

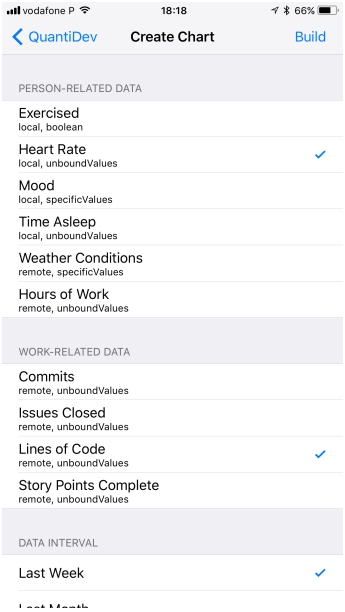


Figure 4.6: Chart creation view for *QuantiDev*

The data sets available to the application are divided in two groups, which are:

- Person-related data sets – Data related to the user's health, lifestyle and behaviors;
  - Exercised – If the user has, or not, exercised on a given day, as provided by the phone's health data provider;
  - Heart Rate – The daily median heart rate of the user, as provided by the phone's health data provider;
  - Mood – The mood of the user on a given day, as acquired by *QuantiDev*;
  - Time Asleep – The number of hours the user slept on a given day, as provided by the phone's health data provider;
  - Weather Conditions – The weather conditions of a given day, as provided by the *Yahoo!* Weather *API*;
  - Hours of Work – The number of hours the user was in the workplace in a given day, as tracked by *QuantiDev*.
- Work-related data sets – Data related to the job of the user;
  - Commits – The number of commits a user has done in a given day, as provided by *GitHub*;
  - Issues Closed – The number of issues a user has closed in a given day, as provided by *Bugzilla*;
  - Lines of Code – The number of lines of code a user has written in a given day, as provided by *GitHub*;

– Story Points Complete – The number of story points a user has completed in a given day, as provided by *Pivotal Tracker*.

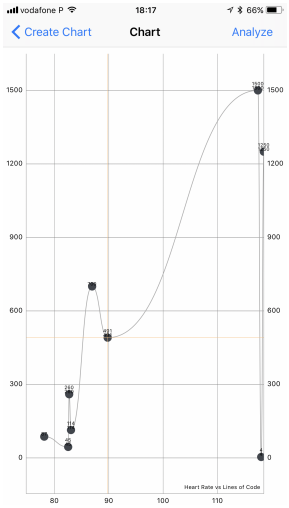


Figure 4.7: Chart view for *QuantiDev*

By tapping the following row, “Analyze Data”, the application attempts to analyze all the previous data sets and find correlations between them, by generating a Pearson correlation value for each pair of data sets. This analysis is obtained by correlating each person-related data set with each work-related one. The values are then ordered by their correlation value and presented on a table (F.6).

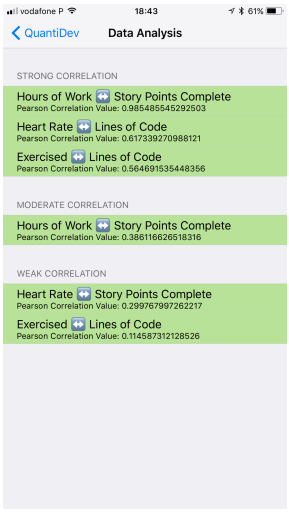


Figure 4.8: Data analysis view for *QuantiDev*

The last row, “*QuantiDev Web*”, provides the user with an easy access to the web interface of *QuantiDev*, by opening a web view where the user is already authenticated on the website (F.8).

Finally, on the left corner of the navigation bar is “Log Out”, a button that allows the user to terminate his session on *QuantiDev* (F.9).

4.4.3 Implementation

There is reasoning behind this part of the application being *iOS*-only. As described in chapter 4, a unified solution to acquire and access health data existed on *iOS* when the platform started to be developed, but not on *Android*. Devices from different manufacturers have different ways of storing health data. For example, *S Health* is used on *Samsung* smartphones and smartwatches, and *Google Fit* on *Android Wear* smartwatches and some Android phones. At least two different *API*s would need to be implemented, as opposed to just one on *iOS*. As of that, it was decided to focus the development on *iOS* only.

Regarding data generated specifically by the application, it includes:

- ***GPS* Data**, which allows for the geolocation of the user. In order not to be too invasive, the workplace(s) of the user are requested, and virtual perimeters (*geofences*) are created on the user’s phone. Whenever the user enters or exits one of those virtual perimeters, the application is notified. This allows for the creation of a log with information about when a user was, or not, working. If the user is contained in a team, information about when a user enters or exists a workplace is also stored on the *QuantiDev*’s servers, and that information can then be reviewed by the team leader.
- **Team Communication Evaluation**, where the user is asked to submit a rating, from 1 to 5 (represented to the user by very sad, sad, neutral, happy and very happy smileys), regarding the user’s communication with the team on the present day. The result of this data acquisition is sent to the *QuantiDev*’s servers, and the team leader is allowed to generate charts based on this feedback.
- **Mood Evaluation**, where, much like what happens with the Team Communication Evaluation, a user is asked to express his mood, in a rating from 1 to 5, also represented by very sad to very happy smileys. This data is **not** sent to *QuantiDev*’s servers and is only kept locally.

Apart from the generation of data, the application can also acquire data (with the user’s consent) from a myriad of health applications and devices, of which an example is a connected *Apple Watch*. Connected health devices are always generating data, most of which is relevant to the scope of this work.

On one hand, the *Apple Watch* is passively gathering as much information as possible about the habits and life of the user. This acquisition works in a seamless way to the user, that is rarely disturbed, except when the answers to fast to answer questions are required – an example of his is the user’s mood for the current day. As such, users are ideally expected to wear their smartwatches daily and during as much time as possible, in order to get the most accurate data. In the event

that the user does not wish to wear a smartwatch for a period of time – for example, during the night – this data would not be taken into account, but doesn't prevent the analysis of data throughout other parts of the day.

On the other hand, the associated *iPhone* is passively gathering data about the user that can not be acquired only using a smartwatch and data from other applications installed in the user's device. An example of data that can not be gathered by using only a smartwatch is the number of hours at the workplace, since not all models of *Apple Watch* contain a *GPS*. This work is assisted by *Apple*'s *HealthKit* framework, which provides a single and unified interface to access health data from applications mentioned before, like *Runkeeper* (of which information about the user's workouts can be gathered) and *Sleep Better* (which registers information about the user's sleep habits and its sleep efficiency).

This data can then be used to generate charts and reports, that can be used to find the correlation (or the lack thereof) between factors that can affect a person's performance and productivity. With regards to what can affect a person's performance, the user's heart rate, time asleep, mood, between other factors are used, and compared to indicators that can suggest the productivity, of which are examples the number of commits, issues closed, lines of code and story points complete. The generated charts can have a timeframe relating to the past week, month, three months or six months.

These charts and reports can only be generated and viewed on the user's mobile device. A user's health and fitness data is sensitive information, subject to privacy laws [Hea17], and as of that should never leave the user's mobile device. There is always the risk of breaches when storing personal data in a remote location, and health data is too sensitive and should be protected at all costs. With that in mind, this type of data is never sent to *QuantiDev*'s servers.

4.4.4 Third-Party Libraries

In order to aid the development of this application, some third-party libraries were used:

- **Alamofire**, a *Swift HTTP* networking client library, that is used in order to establish a communication with *quantiserver* [Ala17].

- **Cereal**, a *Swift* framework for the serialization and deserialization of *Swift structs*, *enums* and classes. It is used to store location data in a way that can be easily loaded to and written from disk [Wee17].

- **Charts**, a *iOS* port of *Android*'s *MPAndroidChart* library which allows for the creation of charts in *iOS* applications. It is used in the generation and visualization of all data charts. It's important to mention that, at the time of development, some features required by the application were broken. These were fixed in the context of this project, and the fixed code was published as open source on *GitHub* [dan17].

- **EZLoadingActivity**, a simple and lightweight activity indicator for *iOS* projects. It is used to show the user a progress indicator when data to generate a chart is being acquired from *quantiserver* [gok17].

- **Realm Cocoa**, a database system specifically designed to run on mobile devices like phones, tablets and wearables [Rea14]. It is used to store data locally that does not need to or should not be stored on *quantiserver*. Examples are mood and workplace data [Rea17a].

4.5 *InteractDev* for *iOS* and *Android*

A good metric regarding team cohesion is the number of interactions and its quality between members of the same team. In order to get data about this metric, though, it needs to be known when two team members are interacting, and then get acquire details related to that interaction.

Three different approaches for this problem were considered.

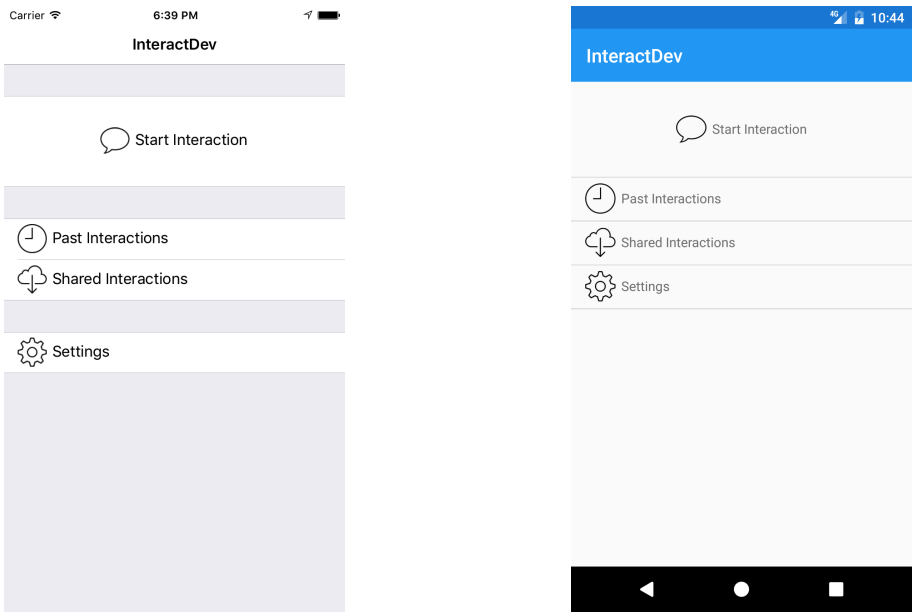
The first one would use the team members' phones as *Bluetooth* transmitters and receivers, which would detect each other when they were in close range (less than one meter). If this condition was true for more than two minutes, an interaction would be recorded and both users would be asked for a review as soon as the two devices were apart. This approach was, unfortunately, not possible. It was not possible, at least while adhering to *Apple*'s guidelines, for two devices to detect each other while they are in standby mode [Dev16]. For this to work, at least one of the devices would need to have its screen always on, which is not an acceptable solution.

A second approach involved beacons, more specifically *iBeacons*. Each team member would carry a beacon, which would be easy to carry around, like a keychain. When the phone of a team member detected a beacon (different than the user's), it could be assumed that two team members (the owner of the phone and of the detected beacon) were communicating. Unfortunately, this is also not possible, since beacons transmit at a relatively high range, and it is not possible to reliably track the distance to a beacon over long periods of time (mostly because this would drain both the phone's and the beacon's battery very fast).

Watching for locations does not seem to be feasible, since two methods were already thought about and were both deemed unfeasible. A mobile application where users evaluated their interactions would be enough to get what is desired, but how do we get the people to use it? The application has to create value to the user, else there is a great chance people will just forget to use it or deem that the application does not provide enough value in exchange their trouble. Also, it needs to be made sure that interactions last at least a minimum amount of time, and the application also needs to do its best in order to verify if the interactions actually happened, so that valid data can be gathered.

With that in mind, an interaction report and note-taking application was created, where users are allowed to take notes about a given interaction, evaluate it by the means of a rating and, if

desired, automatically share the interaction notes with the other party. Users are supposed to start a new interaction as soon as they start discussing with another person, in order for the conversation time to be automatically and correctly measured.



(a) iOS (b) Android

Figure 4.9: Main view for *InteractDev*

4.5.1 Requirements

There are two different types of actors in *InteractDev*, the unauthenticated and the authenticated user.

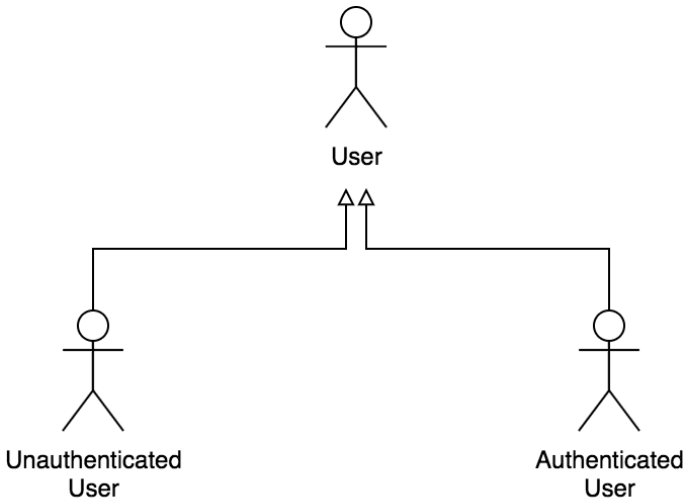


Figure 4.10: Users *UML* diagram for *InteractDev*

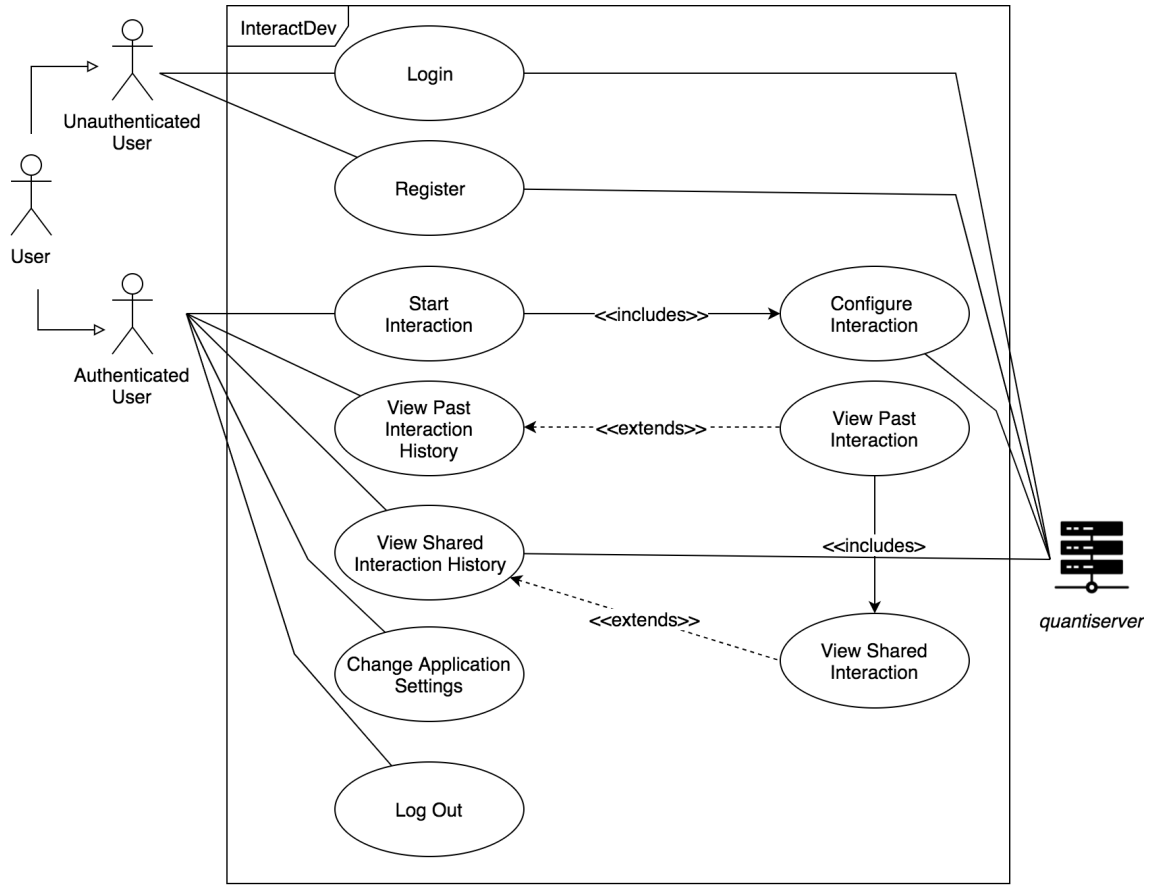


Figure 4.11: Use Case *UML* diagram for *InteractDev*

Identifier	Description
Unauthenticated User	May perform authentication and registration calls to the system.
Authenticated User	Authenticated user. May perform all functions allowed by the application.

Table 4.5: Actor Description for *InteractDev*

Identifier	Name	Priority	Description	Interface
US101	Login	High	As an unauthenticated user, I want to login to the system so that I can use the application.	F.13
US102	Register	Medium	As an unauthenticated user, I want to create an account with the system in order to be able to login.	F.13

Table 4.6: User Stories for an unauthenticated user

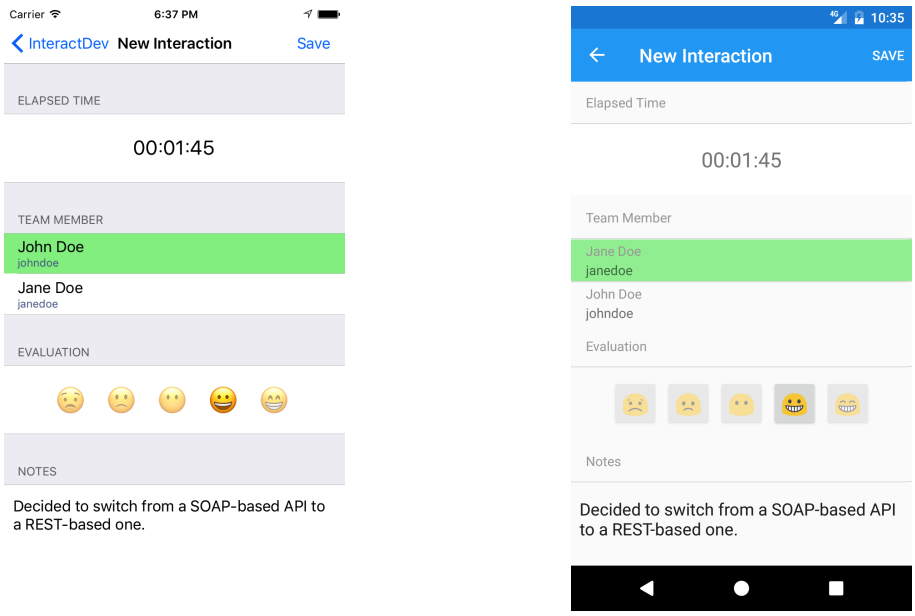
Identifier	Name	Priority	Description	Interface
US201	Start Interaction	High	As an authenticated user, I want to start a new interaction so that it can be logged and possibly shared with my colleague.	F.14
US202	Configure Interaction	High	As an authenticated user, I want to fill an interaction with details so that I can save and record it.	F.14
US203	View Past Interaction History	High	As an authenticated user, I want to view the list of past interactions I logged so that I can check the details and notes of a past interaction created by me.	F.15
US204	View Past Interaction	High	As an authenticated user, I want to view more details about a past interaction I created so that I can check information that can be of use to me.	F.16
US205	View Shared Interaction History	High	As an authenticated user, I want to view the list of past interactions shared with me so that I can check the details and notes of a past interaction created by a colleague.	F.17
US206	View Shared Interaction	High	As an authenticated user, I want to view more details about a past interaction created by a colleague so that I can check information that can be of use to me.	F.18
US207	Change Application Settings	Medium	As an authenticated user, I want to change application settings so that some functions may be better suited to my needs.	F.19
US208	Log Out	Medium	As an authenticated user, I want to be able to log out from the system so that my session can be terminated.	F.19

Table 4.7: User stories for an authenticated user

4.5.2 Features

After the user logs in, the main view is presented, which serves as a main menu of sorts for the application (6.3). From there the user is able to perform all the functions available in the application by tapping each of the table cells.

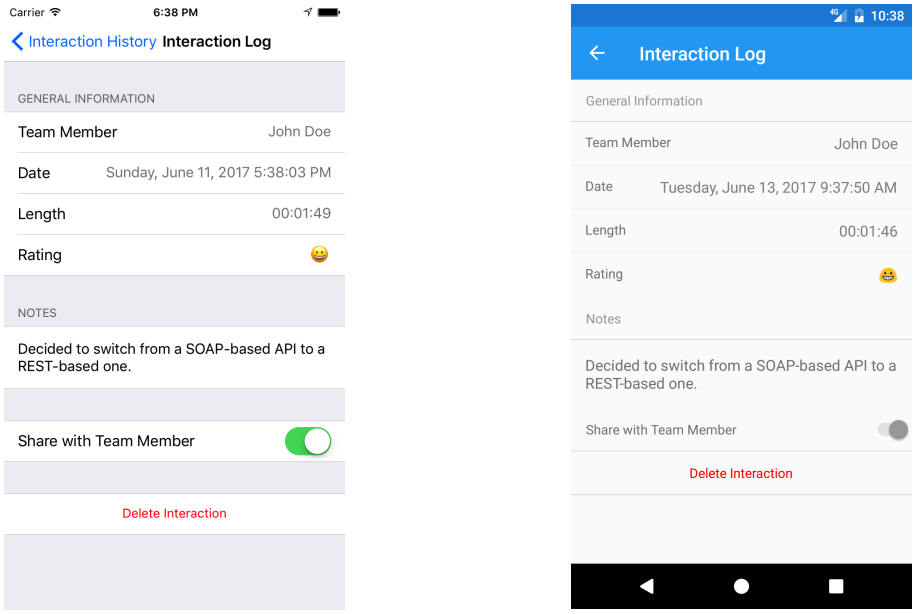
The first row, "Start Interaction", starts the interaction process. The user is supposed to tap this button as soon as an interaction with a colleague begins. Upon tapping it, a new view appears, that includes an interaction timer and requests the user to fill in some information (F.14). This timer represents the time elapsed since the interaction has started, and can be paused and resumed by just tapping on it. On the same view, the user is asked to select which team member the interaction was conducted with, an evaluation (on a scale from 1 to 5, but represented by "very sad" to "very happy" smileys) and notes, that can be read later and optionally shared with the other team member. All fields are required with the exception of the notes field. After the interaction is finished and all fields are filled, the interaction can be saved and the user is redirected back to the main screen.



(a) iOS (b) Android

Figure 4.12: New interaction view for *InteractDev*

The second row, "Past Interactions", presents the user with a view that lists all interactions recorded by the user in the past, ordered by date (F.15). Upon tapping on one of the interactions, another view with the details of the interaction appears. The user is then allowed to view the details of that interaction, choose if it should be shared with the team member the interaction was performed with, or even delete it from the system (F.16).



(a) iOS (b) Android

Figure 4.13: Past interaction detail view for *InteractDev*

The following row, "Shared Interactions", presents the user with a view similar to "Past Interactions", but shows the interactions that are shared with the user instead (F.17). Upon tapping one of the interactions, the user is also shown a view with the detail of the interaction, but is not allowed to make any changes to it (F.18).

Finally, upon tapping the "Settings" row, the user is presented with a view that allows the user to choose if interactions should be automatically shared with the team members and to log out from the system (F.19).

4.5.3 Implementation

Since this application does not make use of any functionality that only exists in *iOS* or *Android* devices, the idea of creating a cross-platform app emerged. There are two different approaches to cross-platform apps, hybrid and native. Hybrid applications use web technologies (normally *HTML* and *JavaScript*) as the languages for the views and logic of the applications, which makes them effectively websites packaged as a mobile application. Native applications are programmed in a single (normally, not-web) language and are then translated (either at compilation or runtime) to native (hence the name) code. Native code has a lot less overhead than the hybrid approach, due to not needing to be run on top of another layer.

Xamarin.Forms is a way to build native apps for *iOS*, *Android* and *Windows Phone* by sharing a single (*C#*) codebase, and the chosen one for this project. It was bought by *Microsoft* in 2016, and is currently supported by them, which makes this platform a safe choice for development, as it is backed by a major company [Gre16]. The most common interface controls are available, so if the

application does not need to any special functionality that is out of the ordinary, there should be no need to write any platform-specific code at all. That was the case with this application. There are also a lot of packages written by other developers that can be used with *Xamarin*, which includes not only code written specifically for *Xamarin*, but also most *.NET* Portable Code Library (*PCL*) libraries. *PCL* libraries are libraries that can be run on the multiple platforms that support .NET, be them desktop, mobile devices or consoles, without the need of re-compilation [pcl17].

4.5.4 Third-Party Libraries

Regarding this application, the following third-party libraries (NuGet packages) were used:

- **Ansuria.XFGloss**, a library that adds a large number of styling properties to *Xamarin.Forms* components. It is used to add a background color to table rows [tba17].
- **Microsoft.Net.Http**, a *HTTP* networking client library, that is used in order to establish a communication with *quantiserver* [Mic17].
- **Newtonsoft.Json**, a *JSON* parser and deserializer for *PCL C#*, that is used to parse messages from *quantiserver* [New17].
- **Realm**, a database system specifically designed to run on mobile devices like phones, tablets and wearables [Rea14]. It is used to store interaction logs locally [Rea17b].

4.6 QuantiDev Web

A big part of the features of *QuantiDev* require an *Apple Watch* or, at the very least, an iOS device with a fitness tracker. There are some features that don't require even a smartphone, though. That being the case, it does not make much sense for one to be required in order to access at least a part of the functionalities provided by the platform. In order to reach the widest number of users, a web application was also created, which only requires a modern web browser.

QuantiDev Web handles all the details regarding the correlation of individual and team data, and allows for the connection to external services, in order to supplement the Quantified Self data acquired from the user with external data and for it to be available to team leaders. These external services are the developer tools described in chapter 4, of which some examples are *JIRA* (which can be used to gather the number of issues solved and their difficulty), *GitHub* (where the number of commits can be gathered) and *Pivotal Tracker* (which can give information about the number of sprint points complete in a week). This web interface also allows for the management of software development teams, and analysis of this data by the team leaders.

4.6.1 Requirements

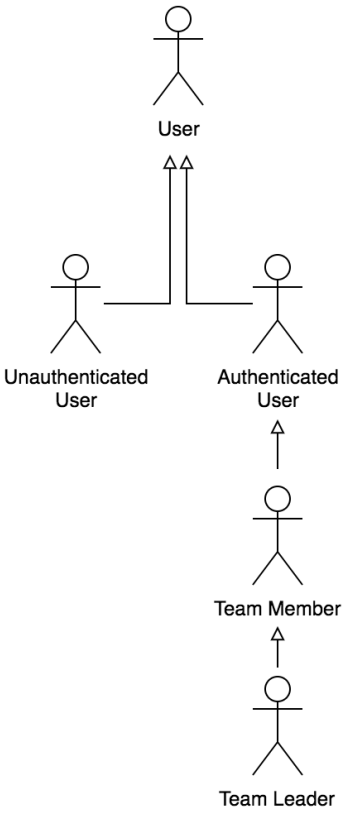


Figure 4.14: Users *UML* diagram for *QuantiDev Web*

Identifier	Description
Unauthenticated User	May perform authentication and registration calls to the system.
Authenticated User	Authenticated user. May perform all functions allowed by the application, except the ones that require the user to be a member of a team.
Team Member	Authenticated user, belonging to a team. May perform all functions allowed by the application, except the ones that allow the user to be a leader of a team.
Team Leader	Authenticated user, leader of a team. May perform all functions allowed by the application.

Table 4.8: Actor Description for *QuantiDev Web*

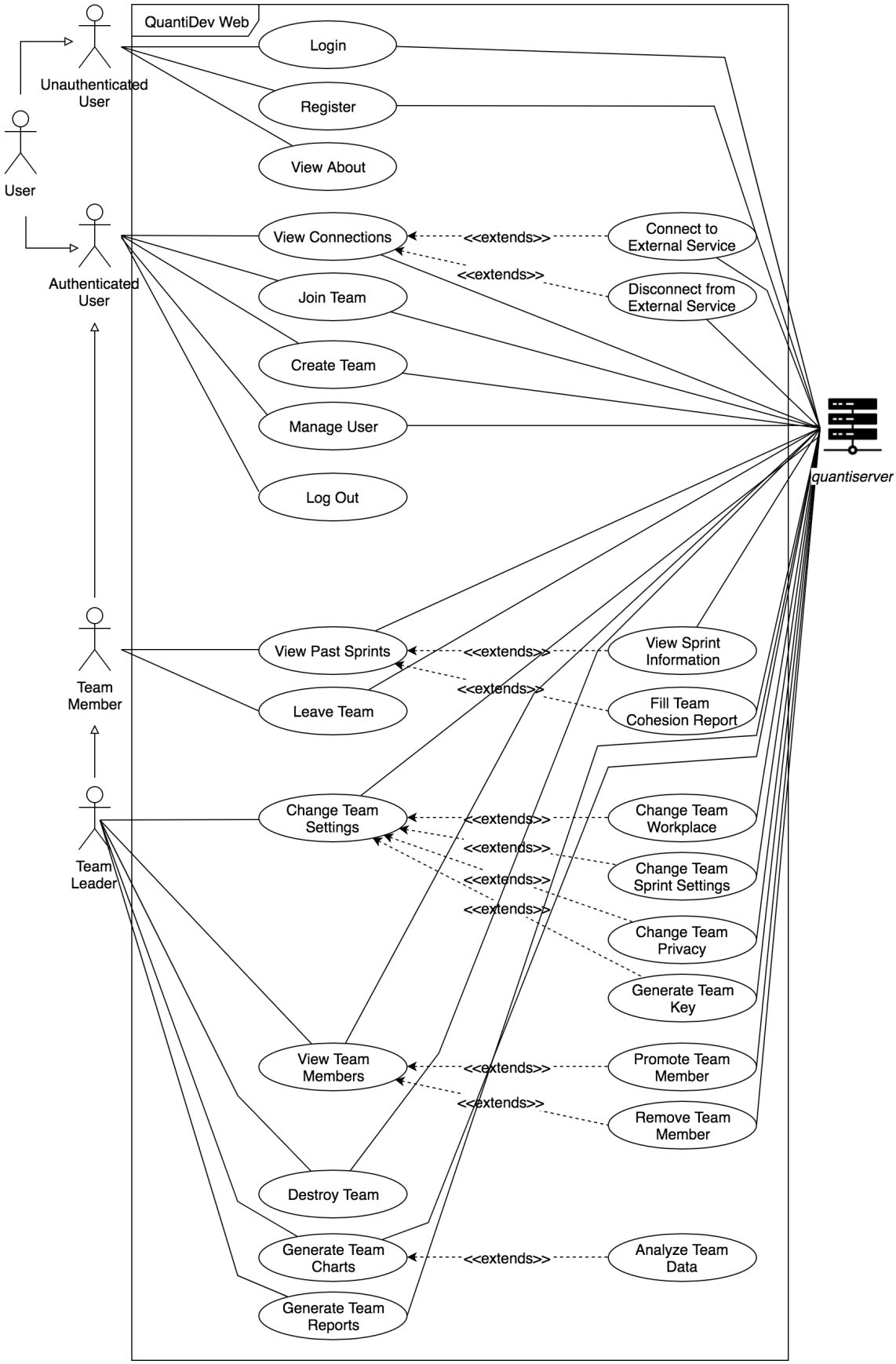


Figure 4.15: Use Case *UML* diagram for *QuantiDev Web*

Identifier	Name	Priority	Description	Interface
US101	Login	High	As an unauthenticated user, I want to login to the system so that I can use the application.	F.20
US102	Register	High	As an unauthenticated user, I want to create an account with the system in order to be able to login.	F.21
US103	View About	Medium	As an unauthenticated user, I want to read about the system so that I can learn about its purpose.	F.22

Table 4.9: User Stories for an unauthenticated user

Identifier	Name	Priority	Description	Interface
US201	View Connections	High	As an authenticated user, I want to check my connections with external services so I can know if the data is correct.	F.23
US202	Connect to External Service	High	As an authenticated user, I want to connect my account to an external service so that data can be automatically acquired by the system.	F.23
US203	Disconnect from External Service	High	As an authenticated user, I want to disconnect my account from an external service so that the system stops automatically acquiring data.	F.23
US204	Join Team	High	As an authenticated user, I want to be able to join an existing team so that I may be a member of that team on the system.	F.24
US205	Create Team	High	As an authenticated user, I want to be able to create a team so that others may join it.	F.24
US206	Manage User	Medium	As an authenticated user, I want to be able to change details of my account so that they are up to date.	F.25
US207	Log Out	Medium	As an authenticated user, I want to be able to log out from the system so that my session can be terminated.	Any

Table 4.10: User Stories for an authenticated user

Identifier	Name	Priority	Description	Interface
US301	View Past Sprints	High	As a team member, I want to be able to check information on past sprints so that I can view which sprints have been tracked by the application.	F.26
US302	View Sprint Information	High	As a team member, I want to be able to check information about the current sprint so that I can know when it started and when it is due to end.	F.26
US304	Leave Team	High	As a team member, I want to be able to leave my current team so that I can create my own, join another or use the platform without being associated to a team.	F.26
US304	Fill Team Cohesion Report	High	As a team member, I want to be able to fill a questionnaire about the cohesion of my team on the last sprint so that the team leader can view that data.	F.27

Table 4.11: User Stories for a team member

Identifier	Name	Priority	Description	Interface
US401	Change Team Settings	High	As a team leader, I want to change my team's settings so that it can more accurately represent the reality.	F.28
US402	Change Team Workplace	High	As a team leader, I want to be able to set my team's workplace so that I can track when the members of my team are in the workplace.	F.28
US403	Change Team Sprint Settings	High	As a team leader, I want to be able to setup the sprint settings so that the system can automatically start and end sprints when time is due.	F.28
US404	Change Team Privacy	High	As a team leader, I want to be able to change the privacy of my team so that only people I want may join it.	F.28
US406	Generate Team Key	High	As a team leader, I want to be able to generate a key so that only people that I share the key with may join the team.	F.28
US407	View Team Members	High	As a team leader, I want to view the current list of members so that I know who are the members of my team.	F.28
US408	Promote Team Member	High	As a team leader, I want to be able to promote another team member to leader, so that I can stop fulfilling that role.	F.28
US409	Remove Team Member	High	As a team leader, I want to be able to remove a team member from my team, so that person can no longer perform functions related to my team.	F.28
US410	Destroy Team	High	As a team leader, I want to be able to delete the team from the system so that there are no more records about the team on the system.	F.28
US411	Generate Team Charts	High	As a team leader, I want to be able to generate charts with information regarding my team so that I can use that information to aid me in analyzing this data and finding problems within my team members.	F.29, F.30
US412	Generate Team Reports	High	As a team leader, I want to be able to generate reports with information regarding my team so that I can use that information to aid me in finding problems within my team members.	F.31
US412	Analyze Team Data	High	As a team leader, I want to be able to obtain the Pearson's correlation coefficient for correlation charts, so that I can use that information to decide if two metrics, measurements or indicators are related.	F.32

Table 4.12: User Stories for a team leader

4.6.2 Features

A registered user can perform many actions using this web interface. This interface is used to change all the settings related to a user, for example, all the connections to external services (currently *GitHub*, *Pivotal Tracker* and *Bugzilla* — F.23), the user's location and account details (F.25).

If the user is contained in a team, a list of past sprints can also be seen, with the option of filling a team cohesion report for the last sprint (F.26). Its format is available in appendix A. The results of the report are then made available to the leader of the team.

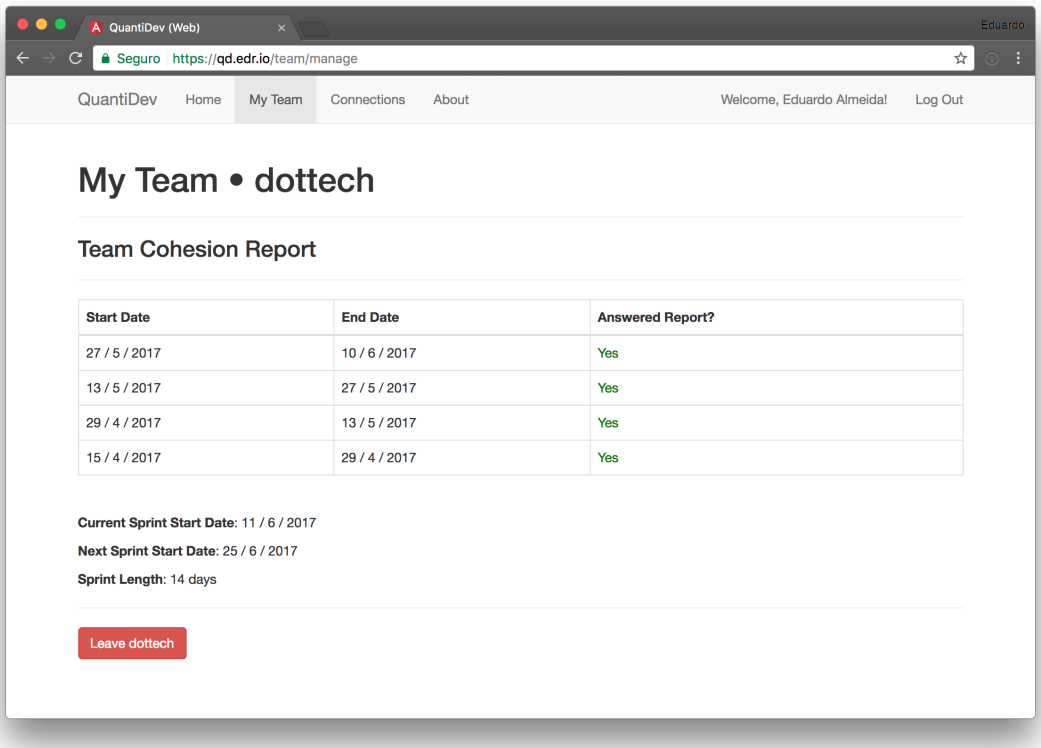


Figure 4.16: Team view (as a team member) for *QuantiDev Web*

If the user is, instead, leader of a team, a whole new range of possibilities opens up. Regarding the management of a team itself, a team leader is allowed to change the team name, the latitude and longitude of the workplace (in order to be able to track when team members are working — F.28), change sprint (start, end date and length – F.28) and privacy (F.28) settings, and also manage the team members (F.28).

Regarding data analysis, the team leader is also allowed to generate a whole range of charts and reports:

- **Data Charts** – Graphical representations of numerical or qualitative data [JA91].
  - **Cohesion** – A line chart showing the results of the team cohesion questionnaire, as filled by the team members, for each sprint.

- **Commits** – A line chart showing the number of commits done per day, per team member, as acquired from *GitHub* data.
- **Communication** – A line chart displaying the median of the evaluations of communications between team members, as answered by the team members themselves.
- **Interactions** – A line chart showing the median of interaction evaluations for a single day, per team member, as acquired from *QuantiDev*.
- **Lines of Code** – A line chart showing the number of lines of code written per day, per team member, as acquired from *GitHub* data.
- **Story Points** – A line chart showing the number of story points complete per day, per team member, as acquired from *Pivotal Tracker* data.
- **Work Detail** – A bar (timeline) chart showing the intervals of time a team member was in the workplace, per day, as acquired from *QuantiDev*.

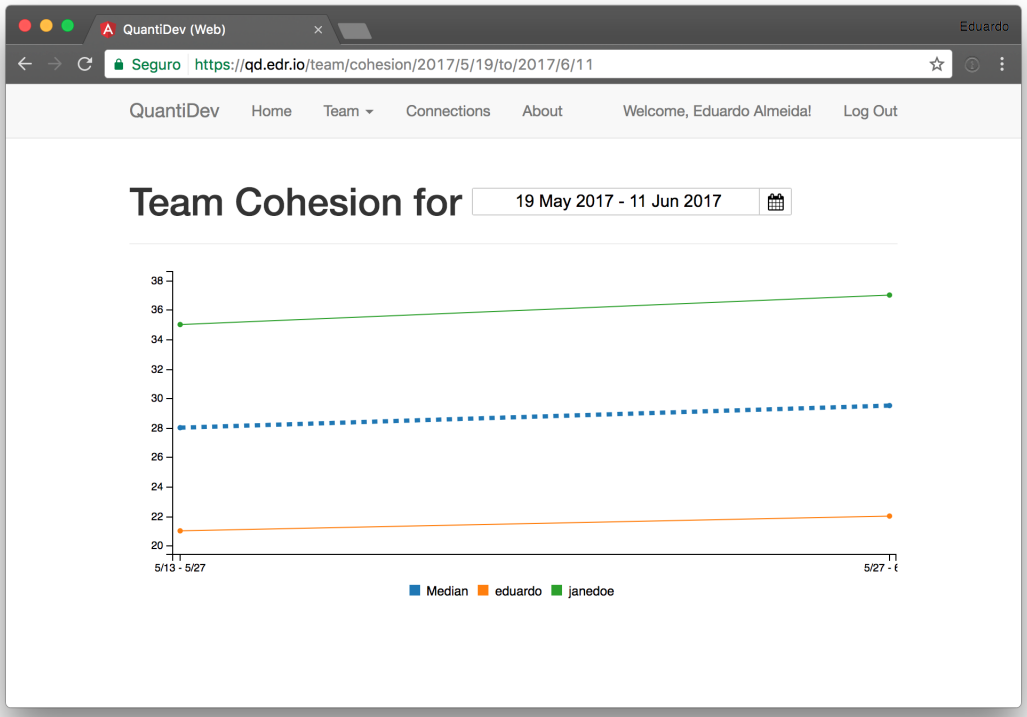


Figure 4.17: Team chart (cohesion) view for *QuantiDev Web*

- **Correlation Charts** – Graphical representations of numerical or qualitative data [JA91] that attempt to demonstrate a correlation, or lack of it, between two different types of data. Similar to what happens in *QuantiDev iOS*, the platform attempts to measure the level of correlation by the means of Pearson's coefficient of correlation.

- **Communication** ↔ **Story Points** – A scatter chart that attempts to demonstrate the correlation (or lack of it) between the communication score of a team and the story points complete, in any given day.
- **Cohesion** ↔ **Story Points** – A scatter chart that attempts to demonstrate the correlation (or lack of it) between the cohesion score of a team and the story points complete, in any given sprint.
- **Together** ↔ **Story Points** – A scatter chart that attempts to demonstrate the correlation (or lack of it) between the time the whole team was together in the workplace and the story points complete, in any given day.

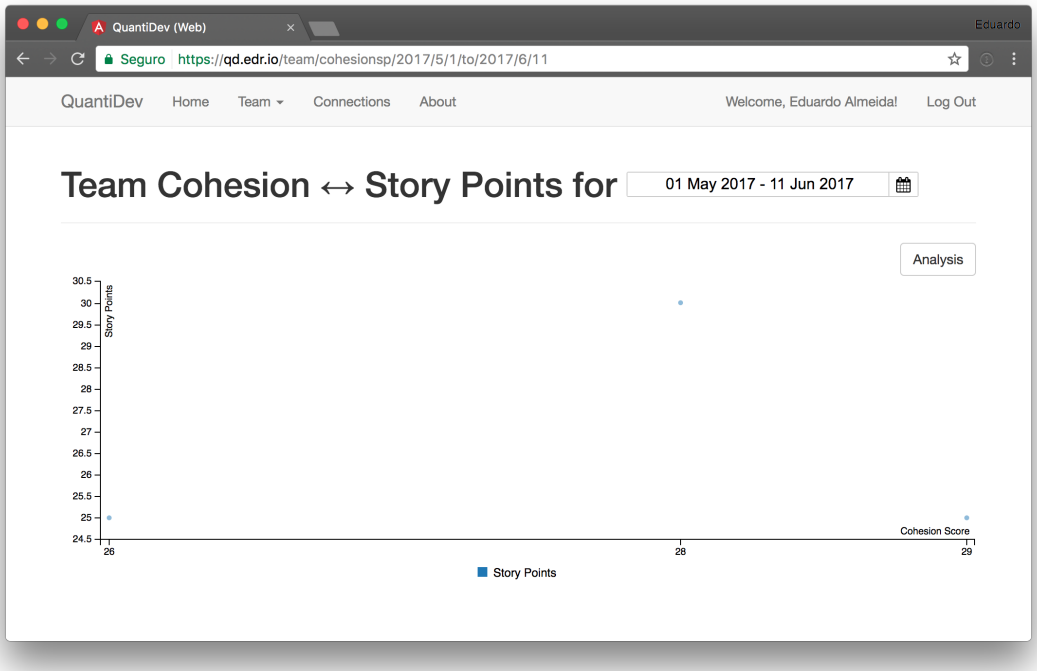


Figure 4.18: Team chart (scatter) view for *QuantiDev Web*

- **Reports** – Outputs that describe in detail a situation, as the result of inputs (in this case, inputs generated by the user and gathered by the platform) [Dic17b]
  - **Team Cohesion** – A report, generated for each sprint of each team, that displays the team cohesion scores for each team member in a given sprint.

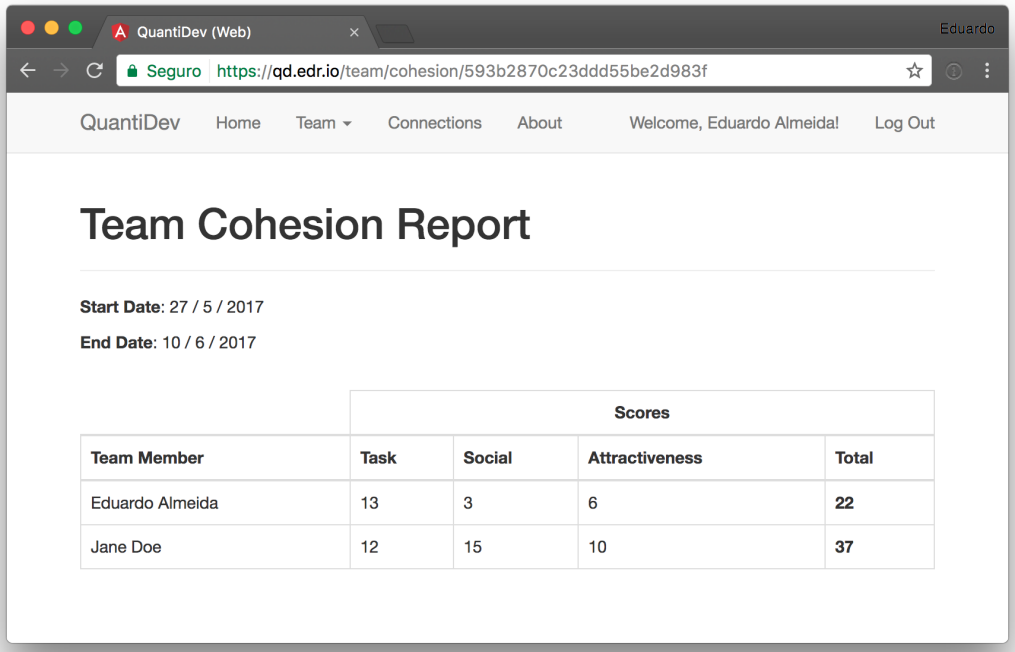


Figure 4.19: Team cohesion report view for *QuantiDev Web*

The backend system, along with the raw data that is on the system, also provides to the web interface some calculated values, namely median, union and intersection values, depending on which value is more appropriate to each data set. Other applications, like this one, are also able to calculate other values, to complement the ones provided by the backend. *QuantiDev Web*, for example, provides the user with Pearson correlation values whenever a correlation chart is shown (F.30). Team leaders are then allowed to use these values to aid them in evaluating the team they lead.

4.6.3 Implementation

The goal for *QuantiDev Web* was to create a fast, easy-to-use, dynamic web interface for the project. Since a stable *REST API* already exists (*quantiserver*), only a client-side web application was required. *Angular* was then a good choice for the project. It should not be confused with *AngularJS*, an old version of the this project that has been superseded. *Angular* is a JavaScript framework, that excels at dynamic, single-page web applications [Wod17]. Applications made with this *framework* are able to run fully on a user's browser without any special needs for the web server hosting the application. In the specific case of this application, the *QuantiDev Web* server is only required and contacted once, when the user requests the application to be loaded. The rest of the calls are performed directly with *quantiserver*'s *REST API*.

4.6.4 Third-Party Libraries

Regarding the development of this web-based application, the following third-party libraries (*npm* packages) were used:

- **Bootstrap 3**, a *HTML*, *CSS* and JavaScript framework to develop responsive web applications, which handles the look and feel of the application [twb16];
- **C3.js**, a *JavaScript* abstraction for *D3* that simplifies the process of creation of charts, by making use of the powerful *API*s of *D3.js*. Used to create all the charts in this application except the timeline charts [mas16];
- **D3.js**, a JavaScript library that is used to aid developers in creating dynamic and interactive data visualizations in web pages, by making use of the *SVG*, *HTML 5* and *CSS* standards, making it compatible across a wide range of devices. Used in tandem with *C3.js* [mbo17];
- **Google Charts**, a simple to use library by *Google* that allows for simple charts to be created. Used in this application to create timeline charts, as these are not supported by *C3.js* [Goo17];
- **mydatepicker**, an *Angular* library to display date pickers. Used to aid in the picking of a date for the chart generators [kek17a];
- **mydaterangepicker**, an *Angular* library to display date range pickers. Used to aid in the picking of date ranges for the chart generators [kek17b];
- **ng2-bootstrap**, *Bootstrap* bindings for *Angular*, allowing for a simpler usage of *Bootstrap* components within *Angular* applications [val17];
- **ng2-google-charts**, *Google Charts* bindings for *Angular*, allowing for a simpler usage of *Google Charts* components within *Angular* applications [gma17].

Chapter 5

Validation

In this chapter, the validations performed in order to understand the usefulness and viability of the platform developed are presented.

5.1 Methodology

In order to evaluate the usefulness of the developed platform, a questionnaire was formulated and then sent, via e-mail, to students that were at the time enrolled in the 4th or 5th year of *Mestrado em Engenharia Informática e Computação*, at *Faculdade de Engenharia da Universidade do Porto*. The project, the questionnaire, and a call to action to fill the questionnaire were presented to a few students during a practical class of *Laboratório de Gestão de Projectos* on May 19th, 2017.

5.2 Questionnaire

A description of the questionnaire sent is located in appendix D. This questionnaire was answered anonymously, and directed to two different groups, members of software development teams and team leaders. Only one version of the questionnaire was sent out, but the groups of questions that did not concern to a subset of users were not presented to them. The order of the questions was also well thought of [DB16b]. First, some introductory questions were asked, regarding if the respondant felt that the problems the platform tries to solve apply to the user. Then, for each component of the platform (in this order, *QuantiDev*, *QuantiDev Web* and *InteractDev*), the respondants were asked about the relevance of the application, and if they (or their team) would use it. Finally, in the last section, some questions about the respondant (smartphone operating system, ownership of a smartwatch, ...) were asked.

A small range, with only three options (*Yes*, *Maybe* and *No*) was used for simple questions, while a *likert scale* was used for measuring opinions about a given subject. No open-ended questions were used [DB16b].

5.3 Result Analysis

In this section, the results acquired from the questionnaire are analyzed and discussed.

5.3.1 Characterization of the Respondents

The respondents of the sent questionnaire are people who were, in the past or at the time of answering the questionnaire, enrolled in at least one course where they had to be either a member of a software development team or the scrum master of one, in a real environment or in an environment that simulated these jobs.

The questionnaire was sent to 310 people, and 30 answers were obtained (roughly 10%). It was open for a week, from May 19th, 2017 to May 26th, 2017. Ideally, more answers were to be expected, but given that the questionnaire was only sent at the end of the semester, it is understandable that only these answers were acquired due to the workload of the respondants at that time.

5.3.2 Problem Overview

In the beginning of the questionnaire, the respondants were asked whether they considered that their productivity was different on distinct days and if it could be related to factors external (not directly related) to their work. All of the participants answered *yes* to the first question, and a majority answered positively to the following question, with 63% answering *yes* and 37% *maybe* – there were no negative answers.

With this, it can be inferred that there is a great interest by software developers to a solution for the problem this software platform tries to solve.

The same individuals were also asked which factors they considered affected their performance. A overwhelming majority (˜ 96%) of them answered that they considered their stress levels and their number of hours of sleep to affect them the most. Those two are metrics that *QuantiDev* is able to track, with no user interaction, given that there are already data sources configured on the user's phone.

5.3.3 Personal Solution Overview

Then, the respondants were asked whether they would use a software like *QuantiDev*, the *iOS* application. Roughly half (˜ 56.7%) of the participants answered *yes*, with ˜ 40% of the participants answering maybe and only one (˜ 3.3%) answering *no*.

The respondants were also asked whether they considered important that the application only required an initial configuration, and then needed no or minimal interaction by the user, and if they considered important that their personal data never left their own personal devices. In both questions, the majority (˜ 63.3% and ˜ 56.7% for both questions, respectively) answered that it was

very important – 5, in a scale from 1 to 5 – which is good, since this is something that was always in mind while the platform was being developed.

5.3.4 Team Solution Overview

The same respondants were also asked if they were – ever, or at the moment – team leaders. If the answer was positive, they would be asked to consider the web part of *QuantiDev*, more specifically if they feel their team could make use of an application like that and if they would use it. People with no team leading experience were not asked to fill this block of questions since most of the functionalities associated with *QuantiDev Web* can only be accessed by team leaders (and are, as thus, hidden from normal team members and users that are not contained in any software development team).

Regarding if the team leaders felt their team could improve their productivity with the existence of *QuantiDev Web* (question A), 70 % were confident of that, answering *yes*, 30 % were unsure, answering *maybe*, and there were no negative answers at all. Considering the next question – if the team leaders would use a tool like the one described (question B) – the answers were perfectly divided between *yes* and *maybe*, with each answer getting exactly 50 %, again with no negative answers. It can be assumed, then, that the ones who answered at the same time *yes* to question A and *maybe* to question B liked the idea of the application, but were unsure if the team would be able to use it or would be receptive to the idea of using the application.

5.3.5 Interaction Overview

In the following section, all of the participants (team leaders or not) were requested for their opinion about if they considered the existence of an application like *InteractDev* useful. While that the answers were not as positive as the other applications, they were still considered satisfying, with ˜ 43.3% of the responses being *yes*, ˜ 46.7% *maybe* and only ˜ 10% *no*.

As stated before, in chapter 6, this application was not the perfect solution to the problem, but only a possible one, so it is positive that such a relatively large number of people thought that the application was useful. Of all the applications developed, this was the one that generated more uncertainity about if people would use it or not.

5.3.6 Final Questions

Finally, the respondants were asked if they felt that the platform as a whole could help improve the performance of software developers, what is the operating system of their mobile phones, and if they use a smartwatch, fitness tracker or similar.

More than half (˜ 53.3%) of the participants answered that they were confident the platform could indeed help improve the performance of software developers, with only one answering negatively and the remaining (˜ 43.3%) being unsure.

Regarding the question whether the respondants wore daily a smartwatch, fitness tracker, or any device that provided their phones with health data, 36.7 % (11) answered positively, which is not a dissatisfying number considering the popularity of these types of devices.

5.4 Usability Evaluation

Apart from the questionnaire, which evaluated the usefulness of the whole platform, usability tests were also conducted. These tests were performed by two people in the area of software engineering, one of which owned an *iPhone* and an *Apple Watch* (here described as *User A*), and the other only owned an *iPhone* (described as *User B*). The two users were not contained in software development teams, so only the personal part of *QuantiDev* was evaluated in these tests.

Both usability tests and follow-up studies were conducted. Usability tests are an "evaluation approach that involves measuring [a] typical user's performance on typical tasks" [DB16a]. On the other hand, follow-up studies are studies where real users give feedback about their interaction with the platform, with the objective of improving the application on the next release [Nie93].

Both users were observed using the applications, and were positively pleased about how easy it was to set up the application, and that all of the data was then gathered with only minimal interaction by the user. One of the users noted that there was (at the time) no easy way to find correlations between all the data sets, a functionality that was since added to *QuantiDev*.

All in all, none of the users had much difficulty in setting up and using the platform, and were satisfied by its design and ease of use.

Even though it was not the main scope of the tests, there was also an interest in finding out if any data correlations were found with the data gathered from the users that tested the app. In order to do this, the users used the application as part of their daily lives for a week (specifically, between May 22nd, 2017 and May 29th, 2017).

With regards to *User A*, a positive, strong correlation was found between the heart rate and lines of code (D.1), and between if the user had exercised and the lines of code written (D.2). Considering *User B*, not as much data was available for the platform, but a strong correlation was found between the hours of work and the story points complete (D.3). Some weaker correlations were also found, but these were not included since they were not deemed relevant to what was being investigated.

The raw data collected by the app and used to create these correlations is located in appendix D. This data is anonymized, since it includes information related to the health and activities of users, which is private data. The two users that tested the platform gave permission for this data to be shared as long as it was not linked to them.

5.5 Conclusion

With this validation phase, the interest of software developers and their expectations regarding a tool like *QuantiDev* was better understood.

It can be said, then, as a result of what was acquired from the described validation phase, that there is a great interest from both individual software developers and agile team leaders for a tool of this type. It is also concluded that their expectations with regards to security and usability were fulfilled.

Considering what was learnt from the usability tests and follow-up studies, it can be said that the personal part of the platform is validated, since it was tested in a real environment and considered to be working by its users.

The team part of the platform was not tested in a real environment, but a possible solution is referred in the next chapter.

Chapter 6

Conclusions

This work begins by explaining the Quantified Self, developer and team productivity and the currently existing tools and technologies that can be used to aid in their evaluation.

The methods used to evaluate and display the acquired data were also described along with a reasoning for the choices made.

After an analysis of the state of the art and existing products, it was concluded that there was no solution like the one developed, which makes this project innovative and necessary.

6.1 Contributions

The developed software package is *QuantiDev*, a fully-fledged solution, consisting of two mobile applications (one for *iOS* and one for *iOS/Android*), a web client application and a web backend. This allows for the gathering of data from the users of the application and the teams they are contained in, in the realm of the software development industry, along with assisting with the analysis of the acquired data. It can then be concluded that the objective of the dissertation – the creation of tooling for the collection of personal and team data related to software developers and their teams – was attained.

With the analysis of the data acquired on the validation phase, it was concluded that there is a clear interest by software developers and their teams on a product like the one developed. Also, the personal part of *QuantiDev* was also deemed to be working by software developers who tested it in a real environment.

Finally, this dissertation, as a learning experience, made it possible to conclude the importance of the data acquired from the devices that software developers usually carry every day, and how it can be used in new ways. If software developers are able to improve aspects in their lives that allow them to work more efficiently, the developed work by them will certainly be of higher quality.

6.2 Future Work

While that the proposed work was concluded and is deemed working, there is still work that can be done.

There is, currently, a stable *REST*-based web service and *API* that communicates with the client applications created. This *API* is not locked down in any way, which means that it is trivial to create new applications that use the data acquired by and generated on the platform.

The web backend currently acquires data, and generates derived information from the acquired inputs, which is then presented to the user by the means of the client applications. The platform only does a simple correlation analysis on the data sets, using the Pearson's correlation coefficient, and generates basic outputs like medians, unions and intersections. A more in-depth and accurate way of analyzing the data would be something interesting to be added at a later date, but is out of the scope of this work as it requires advanced knowledge in statistics and data analysis. As described in the paragraph above, integrating new ways of data analysis into one of the already existing client applications or creating a new one for this purpose would be trivial, except for the actual data analysis part.

It could also be interesting to add new metrics, measurements and indicators to the platform that were not thought of or practical to add during the development of the project.

Regarding the correlations that are made, it should be noted that a correlation does not imply causation, so the user should not take the results given by the application as truthy without carefully analyzing them first. It would be useful to investigate which pairs of variables that when correlated do imply causation and indicate these in the platform.

Some data acquired from the validation phase can also be used to investigate what may be useful with regards to future work on the project. An example is another way of gathering data about an interaction between two team members, since the questionnaire's respondants appear to be much more interested in *QuantiDev for iOS* and *QuantiDev Web* in comparison to *InteractDev*.

The source code for all four parts of the platform was made open-source, and promoted as an open-source project from *FEUP*, under the *MIT* license (available in appendix H) in order to encourage new features to be built upon what is already developed. This way, there is a smaller chance of this project becoming abandoned. In order to aid this, *README*s were added to all projects, a copy of which are available in appendix G.

It would also be useful for the project to be used (even in its current state) by software development teams, in order to validate the usability of the platform in a team environment , as only the personal features were tested in a real environment. It is proposed for this to be done during the next sessions of *Laboratório de Desenvolvimento de Software* (*LDSO*) and *Laboratório de Gestão de Projetos* (*LGP*), both curricular units of the *Mestrado Integrado em Engenharia Informática e Computação*, at *FEUP*.

A paper for the agile conference *XP 2018* was also being prepared at the time this document was being developed, in order to present the platform to agile influencers worldwide.

Appendix A

Team Cohesion Scale

#	Question					
1	Our team works together to reach its goals for performance	1	2	3	4	5
2	I'm happy with my team's level of commitment to the task	1	2	3	4	5
3	Everyone in my team wants to achieve the same level of performance	1	2	3	4	5
4	I have enough opportunities to improve my personal performance	1	2	3	4	5
5	Our team would like to spend time together outside of work hours	1	2	3	4	5
6	Members of our team stick together outside of work time	1	2	3	4	5
7	Members of our team would go out together as a team	1	2	3	4	5
8	For me this team is one of the most important social groups to which I belong	1	2	3	4	5
9	Some of my best friends are in this team	1	2	3	4	5

Table A.1: Team Cohesion Scale Questionnaire [CP00]

Appendix B

quantiserver API

This appendix describes the *API* calls for *quantiServer*, along with the documentation for each call.

In all API calls, <*Authorization Token*> describes an authentication token that should be sent in all requests except the ones that do not require authentication. The only routes exempt from authentication are */login* and */register*. This token has the format <*username*>;<*session token*>, where <*username*> is the username associated with the user and <*session token*> is the token sent by the */login* route.

In addition, all API responses return at least a "success" parameter, with the value "true" if the call was successful or "false" otherwise. An "error" parameter will be sent in the event "success" is "false".

Every time a non-primitive object is specified – for example, UserObject – the model is specified in the section "Models" of this chapter.

B.1 Authentication Routes

- **GET** /login – Authenticates the user with the system, acquiring an authentication token.
  - **Parameters** (as form data) – username: string, password: string
  - **Response** (as JSON) – { success: boolean, error: string, user.name: string, token: string }
- **POST** /signup – Registers a new user with the system.
  - **Parameters** (as form data) – name: string, username: string, password: string, email: string
  - **Response** (as JSON) – { success: boolean, error: string }

- **POST** /logout – Logs out a user from the system, invalidating the associated authentication token.
  - **Parameters** (as headers) – Authorization: "QD-TOKEN <Authorization Token>"
  - **Response** (as JSON) – { success: boolean, error: string }

B.2 User Routes

- **GET** /user/by/id/:id – Returns information about a user in the system.
  - **Parameters** (as headers) – Authorization: "QD-TOKEN <Authorization Token>"
  - **Parameters** (as URL parameters) – id: string
  - **Response** (as JSON) – { success: boolean, user: UserObject }
- **GET** /user/by/username/:username – Returns information about a user in the system.
  - **Parameters** (as headers) – Authorization: "QD-TOKEN <Authorization Token>"
  - **Parameters** (as URL parameters) – username: string
  - **Response** (as JSON) – { success: boolean, user: UserObject }

B.3 Team Routes

- **POST** /team – Creates a new team, with the current user as the team leader.
  - **Parameters** (as headers) – Authorization: "QD-TOKEN <Authorization Token>"
  - **Parameters** (as form data) – name: string
  - **Response** (as JSON) – { success: boolean, error: string, teamId: string }
- **GET** /team/list – Returns information about all the teams in the platform.
  - **Parameters** (as headers) – Authorization: "QD-TOKEN <Authorization Token>"
  - **Response** (as JSON) – { success: boolean, teams: TeamInfo }
- **POST** /team/join – Joins an existing team.
  - **Parameters** (as headers) – Authorization: "QD-TOKEN <Authorization Token>"
  - **Parameters** (as form data) – id: string, key: string
  - **Response** (as JSON) – { success: boolean, error: string }

B.3.1 Team Member Routes

These routes can be performed by every user that is a member or leader of a team.

- **GET** /team – Returns information about the team the current user is contained in.
  - **Parameters** (as headers) – Authorization: "QD-TOKEN <Authorization Token>"
  - **Response** (as JSON) – { success: boolean, team: object }
- **POST** /team/leave – Removes the current user from its team.
  - **Parameters** (as headers) – Authorization: "QD-TOKEN <Authorization Token>"
  - **Response** (as JSON) – { success: boolean, error: string }
- **GET** /team/settings – Returns the settings of the current user's team.
  - **Parameters** (as headers) – Authorization: "QD-TOKEN <Authorization Token>"
  - **Response** (as JSON) – { success: boolean, error: string, settings: TeamSettings }
- **GET** /team/sprint/current – Returns information about the current sprint of the current user's team.
  - **Parameters** (as headers) – Authorization: "QD-TOKEN <Authorization Token>"
  - **Response** (as JSON) – { success: boolean, sprint: DateStartEnd }
- **GET** /team/sprint/:identifier – Returns information about a past sprint of the current user's team.
  - **Parameters** (as headers) – Authorization: "QD-TOKEN <Authorization Token>"
  - **Response** (as JSON) – { success: boolean, error: string, sprint: object }
- **GET** /team/sprint/:identifier/user – Returns a cohesion report for the current user, corresponding to a past sprint of the current user's team.
  - **Parameters** (as headers) – Authorization: "QD-TOKEN <Authorization Token>"
  - **Parameters** (as URL parameters) – identifier: string
  - **Response** (as JSON) – { success: boolean, error: string, sprint: SprintCohesion }
- **GET** /team/sprints/user – Returns the cohesion reports for the current user, corresponding to the past sprints of the current user's team.
  - **Parameters** (as headers) – Authorization: "QD-TOKEN <Authorization Token>"
  - **Response** (as JSON) – { success: boolean, error: string, sprints: [CohesionReport] }
- **POST** /team/sprint/:identifier/cohesion – Fills a cohesion report for a given sprint.

  - **Parameters** (as headers) – Authorization: “QD-TOKEN <Authorization Token>”
  - **Parameters** (as URL parameters) – identifier: string
  - **Parameters** (as form data) – q1: integer, q2: integer: q3: integer, q4: integer, q5: integer, q6: integer, q7: integer, q8: integer, q9: integer
  - **Response** (as JSON) – { success: boolean, error: string }

- **GET** /team/workplace – Returns the workplace information from the team the current user is contained in.
  - **Parameters** (as headers) – Authorization: “QD-TOKEN <Authorization Token>”
  - **Response** (as JSON) – { success: boolean, error: string, workplace: WorkplaceSettings }

B.3.2 Team Leader Routes

These routes can be performed by every user that is a leader of a team.

- **DELETE** /team – Deletes the current user’s team, and all the associated information.
  - **Parameters** (as headers) – Authorization: “QD-TOKEN <Authorization Token>”
  - **Response** (as JSON) – { success: boolean, error: string }
- **POST** /team/settings – Change the settings of the current user’s team.
  - **Parameters** (as headers) – Authorization: “QD-TOKEN <Authorization Token>”
  - **Parameters** (as form data) – name: string, open: boolean, joinKey: string
  - **Response** (as JSON) – { success: boolean, error: string }
- **GET** /team/sprint – Returns the sprint settings of the current user’s team.
  - **Parameters** (as headers) – Authorization: “QD-TOKEN <Authorization Token>”
  - **Response** (as JSON) – { success: boolean, error: string, sprint: SprintSettings }
- **POST** /team/sprint – Changes the sprint settings of the current user’s team.
  - **Parameters** (as headers) – Authorization: “QD-TOKEN <Authorization Token>”
  - **Parameters** (as form data) – length: integer, year: integer, month: integer, day: integer
  - **Response** (as JSON) – { success: boolean, error: string }
- **GET** /team/sprint/:identifier/cohesion/summary – Returns cohesion reports for all the members of a team, corresponding to a past sprint of the current user’s team.
  - **Parameters** (as headers) – Authorization: “QD-TOKEN <Authorization Token>”
  - **Parameters** (as URL parameters) – identifier: string

  - **Response** (as JSON) – { success: boolean, error: string, sprints: [CohesionReport] }

- **GET** /team/sprints/:fromYear/:fromMonth/:fromDay/to/:toYear/:toMonth/:toDay – Returns the cohesion scores for sprints between a given date range, corresponding to the current user's team.
  - **Parameters** (as headers) – Authorization: "QD-TOKEN <Authorization Token>"
  - **Parameters** (as URL parameters) – fromYear: integer, fromMonth: integer, fromDay: integer, toYear: integer, toMonth: integer, toDay: integer
  - **Response** (as JSON) – { success: boolean, error: string, sprints: [SimplifiedCohesionReport] }
- **POST** /team/key – Generates a new, random join key for the current user's team.
  - **Parameters** (as headers) – Authorization: "QD-TOKEN <Authorization Token>"
  - **Response** (as JSON) – { success: boolean, error: string, key: string }
- **POST** /team/leader – Promotes another user to leader of the current user's team.
  - **Parameters** (as headers) – Authorization: "QD-TOKEN <Authorization Token>"
  - **Parameters** (as form data) – username: string
  - **Response** (as JSON) – { success: boolean, error: string }
- **POST** /team/kick – Remove another user from the current user's team.
  - **Parameters** (as headers) – Authorization: "QD-TOKEN <Authorization Token>"
  - **Parameters** (as form data) – username: string
  - **Response** (as JSON) – { success: boolean, error: string }
- **POST** /team/workplace – Changes the workplace information for the current user's team.
  - **Parameters** (as headers) – Authorization: "QD-TOKEN <Authorization Token>"
  - **Parameters** (as form data) – latitude: double, longitude: double, identifier: string
  - **Response** (as JSON) – { success: boolean, error: string }
- **GET** /team/intersect/:year/:month/:day – Returns a value referring to the number of hours every team member was in the workplace at the same time, during the specified day.
  - **Parameters** (as headers) – Authorization: "QD-TOKEN <Authorization Token>"
  - **Parameters** (as URL parameters) – year: integer, month: integer, day: integer
  - **Response** (as JSON) – { success: boolean, error: string, intersect: double }
- **GET** /team/union/:year/:month/:day – Returns a value referring to the number of hours any team member was in the workplace at any time, during the specified day.

  - **Parameters** (as headers) – Authorization: “QD-TOKEN <Authorization Token>”
  - **Parameters** (as URL parameters) – year: integer, month: integer, day: integer
  - **Response** (as JSON) – { success: boolean, error: string, intersect: double }

- **GET** /team/metric/:year/:month/:day – Convenience route that returns the values of the intersect and union routes above.

  - **Parameters** (as headers) – Authorization: “QD-TOKEN <Authorization Token>”
  - **Parameters** (as URL parameters) – year: integer, month: integer, day: integer
  - **Response** (as JSON) – { success: boolean, error: string, intersect: double }

- **GET** /team/communication/median/:year/:month/:day – Retrieves the communication health score for a team, referring to a specific day.

  - **Parameters** (as headers) – Authorization: “QD-TOKEN <Authorization Token>”
  - **Parameters** (as URL parameters) – year: integer, month: integer, day: integer
  - **Response** (as JSON) – { success: boolean, error: string, median: double }

- **GET** /team/sp/median/:year/:month/:day – Retrieves the median number of story points performed by each member of the team, referring to a specific day.

  - **Parameters** (as headers) – Authorization: “QD-TOKEN <Authorization Token>”
  - **Parameters** (as URL parameters) – year: integer, month: integer, day: integer
  - **Response** (as JSON) – { success: boolean, error: string, median: double }

- **GET** /team/communication/:year/:month/:day – Retrieves the communication health score, for each member belonging to the current user’s team, referring to a specific day.

  - **Parameters** (as headers) – Authorization: “QD-TOKEN <Authorization Token>”
  - **Parameters** (as URL parameters) – year: integer, month: integer, day: integer
  - **Response** (as JSON) – { success: boolean, error: string, communication: CommunicationObject }

- **GET** /team/together/:year/:month/:day – Returns whether everyone was working at the same time in the workplace, at least once, for the current user’s team, referring to a specific day.

  - **Parameters** (as headers) – Authorization: “QD-TOKEN <Authorization Token>”
  - **Parameters** (as URL parameters) – year: integer, month: integer, day: integer
  - **Response** (as JSON) – { success: boolean, error: string, together: boolean }

- **GET** /team/external/:year/:month/:day – Returns the data acquired from external services (eg.: *GitHub*, *JIRA*, ...) regarding every member of the current user’s team, referring to a specific day.

  - **Parameters** (as headers) – Authorization: “QD-TOKEN <Authorization Token>”
  - **Parameters** (as URL parameters) – year: integer, month: integer, day: integer
  - **Response** (as JSON) – { success: boolean, error: string, data: ReturnedData }

- **GET** /team/interactions/summary/:year/:month/:day – Retrieves the interaction median score for each team member of the current user’s team, referring to a specific day.
  - **Parameters** (as headers) – Authorization: “QD-TOKEN <Authorization Token>”
  - **Parameters** (as URL parameters) – year: integer, month: integer, day: integer
  - **Response** (as JSON) – { success: boolean, error: string, summary: [InteractionSummary] }

- **GET** /team/interactions/detail/:year/:month/:day – Retrieves the detailed interaction log for each team member of the current user’s team, referring to a specific day.
  - **Parameters** (as headers) – Authorization: “QD-TOKEN <Authorization Token>”
  - **Parameters** (as URL parameters) – year: integer, month: integer, day: integer
  - **Response** (as JSON) – { success: boolean, error: string, interactions: InteractionData }

B.4 Log Routes

- **POST** /log/leave – Signals the user as having left the workplace.
  - **Parameters** (as headers) – Authorization: “QD-TOKEN <Authorization Token>”
  - **Response** (as JSON) – { success: boolean, error: string }

- **POST** /log/enter – Signals the user as having entered the workplace.
  - **Parameters** (as headers) – Authorization: “QD-TOKEN <Authorization Token>”
  - **Response** (as JSON) – { success: boolean, error: string }

- **GET** /log/:class/:year/:month/:day – Retrieves the records logged for a specific class of external events, regarding the current user, referring to a specific day.
  - **Parameters** (as headers) – Authorization: “QD-TOKEN <Authorization Token>”
  - **Parameters** (as URL parameters) – class: string, year: integer, month: integer, day: integer
  - **Response** (as JSON) – { success: boolean, error: string }

- **POST** /log – Adds a record of an external event to the database (eg.: an interaction report)
  - **Parameters** (as headers) – Authorization: “QD-TOKEN <Authorization Token>”
  - **Parameters** (as form data) – class: string, (other parameters depending on the type of event)
  - **Response** (as JSON) – { success: boolean, error: string, (other parameters depending on the type of event) }

B.5 InteractDev Routes

- **GET** /interact/shared – Retrieves the list of past interactions shared with the current user.
  - **Parameters** (as headers) – Authorization: "QD-TOKEN <Authorization Token>"
  - **Response** (as JSON) – { success: boolean, interactions: [PastSharedInteractionObject] }
- **GET** /interact/my – Retrieves the list of past interactions recorded by the current user.
  - **Parameters** (as headers) – Authorization: "QD-TOKEN <Authorization Token>"
  - **Response** (as JSON) – { success: boolean, interactions: [PastPersonalInteractionObject] }
- **POST** /interact/:identifier – Changes the privacy of a past interaction created by the current user.
  - **Parameters** (as headers) – Authorization: "QD-TOKEN <Authorization Token>"
  - **Parameters** (as URL parameters) – identifier: string
  - **Parameters** (as form data) – shared: boolean
  - **Response** (as JSON) – { success: boolean, error: string }
- **DELETE** /interact/:identifier – Deletes a past interaction log created by the current user.
  - **Parameters** (as headers) – Authorization: "QD-TOKEN <Authorization Token>"
  - **Parameters** (as URL parameters) – identifier: string
  - **Response** (as JSON) – { success: boolean, error: string }

B.6 Connection Routes

- **GET** /connection – Retrieves the list of external service connections established for the current user.
  - **Parameters** (as headers) – Authorization: "QD-TOKEN <Authorization Token>"
  - **Response** (as JSON) – { success: boolean, error: string, connections: [ConnectionObject] }
- **POST** /connection/:type – Establishes a connection to an external service.
  - **Parameters** (as headers) – Authorization: "QD-TOKEN <Authorization Token>"
  - **Parameters** (as URL parameters) – type: string
  - **Response** (as JSON) – { success: boolean, error: string }
- **DELETE** /connection/:type – Revokes a connection with an external service.
  - **Parameters** (as headers) – Authorization: "QD-TOKEN <Authorization Token>"

  - **Parameters** (as URL parameters) – type: string
  - **Response** (as JSON) – { success: boolean, error: string }

- **GET** /connections – Retrieves the list of available external service connections.
  - **Parameters** (as headers) – Authorization: "QD-TOKEN <Authorization Token>"
  - **Response** (as JSON) – { success: boolean, error: string, connections: [AvailableConnectionObject] }

B.7 QuantiDev Routes

- **GET** /data/since/:year/:month/:day – Retrieves the data acquired for the current user, since a specific day up to the current day.
  - **Parameters** (as headers) – Authorization: "QD-TOKEN <Authorization Token>"
  - **Response** (as JSON) – { success: boolean, error: string, data: [KeyValuePair] }

- **GET** /data/:year/:month/:day – Retrieves the data acquired for the current user, referring to a specific day.
  - **Parameters** (as headers) – Authorization: "QD-TOKEN <Authorization Token>"
  - **Response** (as JSON) – { success: boolean, error: string, data: KeyValuePair }

- **GET** /data/today – Retrieves the data acquired for the current user, referring to the current day.
  - **Parameters** (as headers) – Authorization: "QD-TOKEN <Authorization Token>"
  - **Response** (as JSON) – { success: boolean, error: string, data: [KeyValuePair] }

B.8 Models

- **UserObject** – Represents public information about a user.
  - **id** : string – User identifier, as represented in the database.
  - **name** : string – User's real name
  - **username** : string – User's username
  - **team** : string – Identifier for the team the user is contained in, as represented in the database.

- **TeamInfo** – Represents public information about a team.
  - **id** : string – Team identifier, as represented in the database.
  - **name** : string – Team name

  - **owner** : string – Team owner's real name
  - **open** : boolean – Team public status

- **TeamSettings** – Represents a team's settings.
  - **joinKey** : string – Team join key
  - **open** : boolean – Team public status
  - **sprint** : SprintSettings – Team's sprint settings

- **DateYMD** – Represents the day, month and year of a date.
  - **year** : integer – Year
  - **month** : integer – Month
  - **day** : integer – Day

- **SprintSettings** – Represents the sprint settings of a team.
  - **length** : integer – Sprint length, in days
  - **date** : DateYMD – Sprint start date

- **DateStartEnd** – Represents a time interval.
  - **start** : DateYMD – Sprint start date
  - **end** : DateYMD – Sprint end date

- **PastSprintData** – Represents the time a past sprint has began and ended.
  - **date** : DateStartEnd – Start and End of sprint date

- **SprintCohesion** – Represents the sprint cohesion report, for a given sprint.
  - **date** : DateYMD – Sprint start and end date
  - **cohesion** : CohesionReport – Sprint Cohesion Report

- **QuestionAnswer** – Represents a question / answer pair.
  - **question** : integer – Question Number
  - **answer** : integer – Answer (1 to 5)

- **CohesionReport** – Represents the answer of a cohesion report, as filled by an user.
  - **owner** : string – User identifier, as specified in the database
  - **sprint** : string – Sprint identifier, as specified in the database
  - **answers** : [QuestionAnswer] – Question identifier and answer
  - **date** : DateYMD – Date of cohesion report was submitted

- **SimplifiedCohesionReport** – Represents a simplified version (only the median) of the cohesion reports associated with a given sprint.
  - **id** : string – The identifier of the sprint, as specified in the database
  - **date** : DateYMD – Date the cohesion report was submitted
  - **cohesion** : number – The median of the cohesion reports total value for a given team.
- **WorkplaceSettings** – Represents the settings associated with a given workplace, more specifically its geographic location.
  - **latitude** : number – Latitude part of the location (coordinate) of the workplace.
  - **longitude** : number – Longitude part of the location (coordinate) of the workplace.
  - **identifier** : string – Internal identifier for the location.
- **CommunicationObject** – Represents a communication rating, as evaluated by an user.
  - **member** : string – User identifier, as represented in the database.
  - **rating** : integer – Rating (from 1 to 5).
- **ReturnedData**

  A *map*, with the data types as keys, and its values as the value.
- **InteractionSummary** – Represents an interaction score for a single user, referring to a specific day.
  - **member** : UserObject – The user whose interactions are being evaluated.
  - **median** : integer – The summary score for the daily interactions of an user.
- **InteractionData** – Represents the detail of an interaction between two different users.
  - **owner** : string – The username of the user that created the interaction.
  - **intervenient** : string – The username of the intervenient of the interaction.
  - **duration** : integer – The duration, in seconds, of the interaction.
  - **rating** : integer – The score for the interaction.
- **PastSharedInteractionObject** – Represents a shared interaction between two users. This is the object that the user the interaction was shared with should see.
  - **owner** : UserObject – The user that created the interaction.
  - **startDate** : String – The date that the interaction started, in ISO format.
  - **duration** : integer – Duration of the interaction, in seconds.
  - **notes** : string – Notes taken during the interaction.

- **PastPersonalInteractionObject** – Represents a past interaction between two users. This is the object that the interaction creator should see.
  - **intervenient** : UserObject – The user that created the interaction.
  - **identifier** : string – Internal identifier for the interaction.
  - **rating** : integer – Interaction rating, as evaluated by the owner (from 1 to 5)
  - **startDate** : DateYMD – Date that the interaction started.
  - **duration** : integer – Duration of the interaction, in seconds.
  - **notes** : string – Notes taken during the interaction.
  - **shared** : boolean – Denotes if the interaction is shared with the intervenient.
- **ConnectionObject** – Represents an established connection to an external service.
  - **internal** : string – The identifier for the service for which a connection was established.
  - **username** : string – The username of the current user on the remote service.
- **KeyValuePair** – Represents a generic key-value pair.
  - **key** : string – A key, belonging to a key-value pair.
  - **value** : string – A value, belonging to a key-value pair.
- **AvailableConnectionObject** – Represents an external services that the platform has information on how to connect with, and the information required for a connection to be established by the user.
  - **name** : string – The name of the external service for which a connection can be established.
  - **internal** : string – The internal name for the external service.
  - **fields** : [KeyValuePair] – A key-value pair of required fields in order to establish a connection to the external service.
  - **tip** : string – An optional help tip to aid the user in establishing a connection to the external service.

Appendix C

quantiserver MongoDB Schemas

- cohesion – Stores the answer of the cohesion report for a given user, pertaining to a given sprint.
  - owner : ObjectId(user)
  - sprint : ObjectId(sprint)
  - answers
    - question : Number
    - answer : Number
  - date – Date
- commits, issues, lines, storypoints – Stores the number of commits, issues, lines and story points complete by a given user, on a given day.
  - owner : ObjectId(user)
  - count : Number
  - date
    - start : Date
    - end : Date
- communication – Stores the team communication score for a single day, as evaluated by a team member. This score rating can range from 1 to 5.
  - owner : ObjectId(user)
  - team : ObjectId(team)
  - rating : Number
  - date : Date

- interaction – Stores the rating and detail of an interaction between two team members.
  - owner : ObjectId(user)
  - intervenient : ObjectId(user)
  - rating : Number
  - startDate : Date
  - duration : Number
  - notes : String
  - shared : Boolean
  - deleted : Boolean
- sprint – Stores a sprint related to a team, and its associated data.
  - team : ObjectId(team)
  - date
    - start : Date
    - end : Date
- team – Stores information about a team and its settings.
  - name : String
  - owner : ObjectId(user)
  - workplace : ObjectId(workplace)
  - open : Boolean
  - joinKey : String
  - sprints
    - length : Number
    - start : Date
- user – Stores information about a user.
  - name : String
  - username : String
  - password : String
  - email : String
  - [tokens] : Array
    - token : String
    - created : Date

  - team : ObjectId(team)
  - teamMemberSince : Date
  - [pastTeams] : Array
    - team : ObjectId(team)
    - timeframe
      - start : Date
      - end : Date

- weather – Stores information about weather conditions, for a given location and date. The weather's location refers to the name of the city, while the condition represents a given weather state in the *Yahoo!* Weather API.
  - location : String
  - condition : Number
  - date : Date

- worklog – Stores the date a given user has entered and exited a given workplace.
  - owner : ObjectId(user)
  - date
    - start : Date
    - end : Date
  - location
    - latitude : Number
    - longitude : Number

- workplace – Stores information about a geographical place.
  - latitude : Number
  - longitude : Number
  - identifier : String

Appendix D

Questionnaire

D.1 QuantiDev

In the context of the dissertation project "Quantified Self for Developers", of the Integrated Masters in Informatics and Computer Engineering, the following questionnaire was developed with the objective of obtaining feedback on the platform developed.

- Are you a software developer?
  - ○ Yes
  - ○ No (*End the questionnaire*)

D.2 Quantified Self for Software Developers

It is common for the productivity of those who work in the software development sector to be different from day to day, and often the reasons for this to happen are unknown. Some days, a lot of work is done, and on others, not so much.

- Do you feel that this problem applies to you?
  - ○ Yes
  - ○ No
- Do you believe that, in some cases, this problem may be caused by factors external to your work environment?
  - ○ Yes
  - ○ Maybe
  - ○ No

- If you answered “yes” or “maybe” to the last question, which factors do you consider may affect your performance?

  □ Stress Levels

  □ Sleep Habits

  □ Mood

  □ Physical Activity

  □ Weather

  □ Others (*Please specify which...*)

Consider, now, an application that retrieves your health data (physical activity, sleep habits, among others) and relates them with work data (lines of code, number of commits, among others).

- Do you believe an application of this kind would be useful in finding causes for the problem mentioned above?

  ○ Yes

  ○ Maybe

  ○ No

- Do you value the fact of an application of this type only requiring minimal interaction from the user, only requiring an initial configuration?

○	○	○	○	○
1	2	3	4	5

- Do you consider important the fact of this kind of data only being stored on the device of the user, and being never, in any case, shared with third parties?

○	○	○	○	○
1	2	3	4	5

D.3 Quantified Team

- Are you, or were you ever, leader of a team / scrum master?

  ○ Yes

  ○ No (*Jump to section 5*)

D.4 Quantified Team for Software Development Teams

It is not easy to evaluate the performance of a software development team, since values like the number of lines of code, sprint points per sprint, etc., do not make sense when compared between different teams.

Furthermore, there are various factors inside the teams, like if the team members are working together in the same physical place, and the quality of the communication between its members, that may affect the performance of the team as a whole.

- Do you believe that your team could benefit from a tool that would help analyse these factors that can affect its performance?
  - ○ Yes
  - ○ Maybe
  - ○ No
- Would you use a tool of this kind?
  - ○ Yes
  - ○ Maybe
  - ○ No
- Do you consider important the fact of an application of this kind only requiring a minimal interaction by the team members, only required a fast, initial configuration by them?

○	○	○	○	○
1	2	3	4	5

D.5 InteractDev

Communication and its quality, between team members, can affect the productivity of the team as a whole.

However, it is not easy to gather data about the communication between team members.

In order to try and find a solution to this issue, an application was created that allows for the recording of the notes of an interaction between two team, with an optional sharing of these between the intervenients, and with the gathering of feedback regarding same interaction.

- Do you consider the existence of an application of this kind useful?
  - ○ Yes
  - ○ Maybe
  - ○ No

D.6 Final Questions

- Do you consider that this platform, which consists in the applications described in the questions above, could aid in the global performance improvement of software development teams?

  ○ Yes

  ○ Maybe

  ○ No

- What is the operative system of your smartphone?

  ○ iOS

  ○ Android

  ○ Windows Phone

  ○ Other (*Please specify which...*)

- Do you use daily a smartwatch, fitness tracker, or another device which provides your smartphone with health data?

  ○ Yes

  ○ No

Thank you for your collaboration in filling out this questionnaire!

Appendix E

Test Data

E.1 User A: Median Heart Rate ↔ Median Lines of Code

Heart Rate	Lines of Code
82.4	127
83.2	86
85.3	135
87.6	124
92.7	149
93.2	275
96.2	186

Table E.1: Median Heart Rate ↔ Median Lines of Code data, according to the data gathered from User A

Pearson Correlation Value: 0.711487708

E.2 User A: Exercised ↔ Median Lines of Code

Exercised	Lines of Code
No	86
No	124
No	127
No	135
Yes	149
Yes	186
Yes	275

Table E.2: Exercised ↔ Median Lines of Code data, according to the data gathered from User A

Pearson Correlation Value: 0.748154323

E.3 User B: Hours of Work ↔ Story Points Complete

Hours of Work	Story Points
0:30	0
1:40	2
4:22	3
5:12	6
5:23	7
5:34	4
6:53	5

Table E.3: Hours of Work ↔ Median Story Points complete data, according to the data gathered from User B

Pearson Correlation Value: 0.842371949

Appendix F

Screenshots

F.1 QuantiDev for iOS

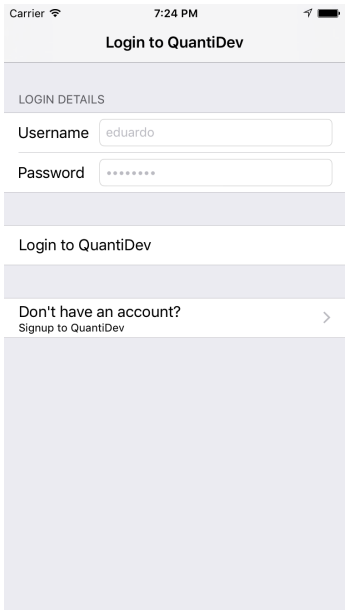


Figure F.1: Login view for *QuantiDev*

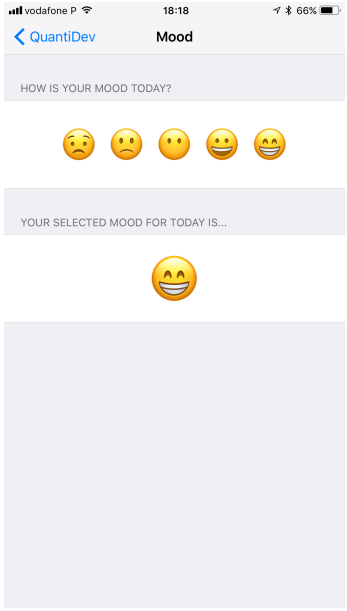


Figure F.2: Daily mood view for *QuantiDev*

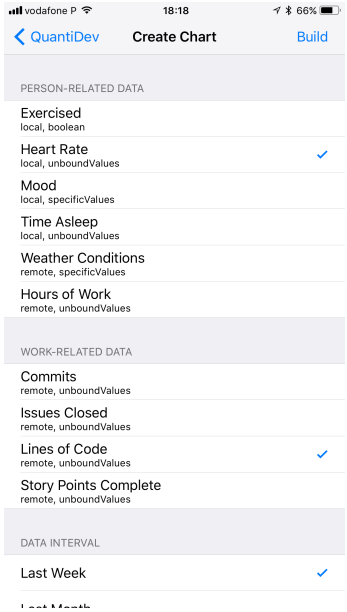


Figure F.3: Chart creation view for *QuantiDev*

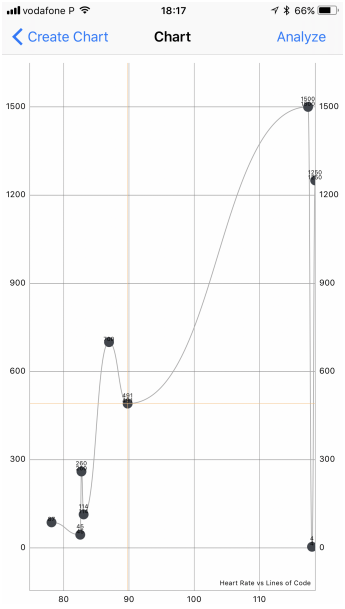


Figure F.4: Chart view for *QuantiDev*

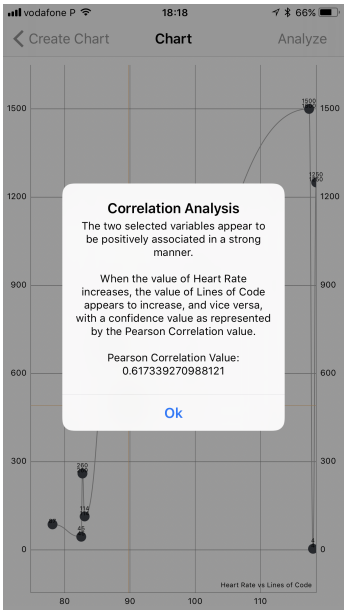


Figure F.5: Chart pearson analysis for *QuantiDev*

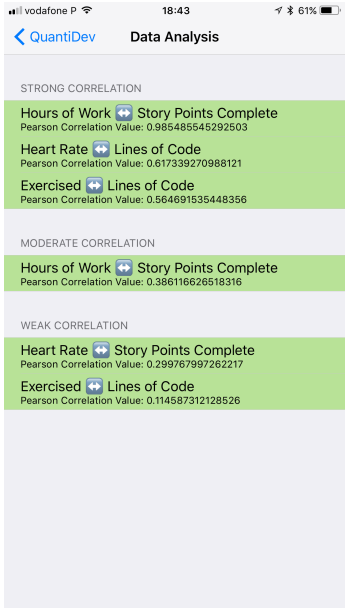


Figure F.6: Data analysis view for *QuantiDev*

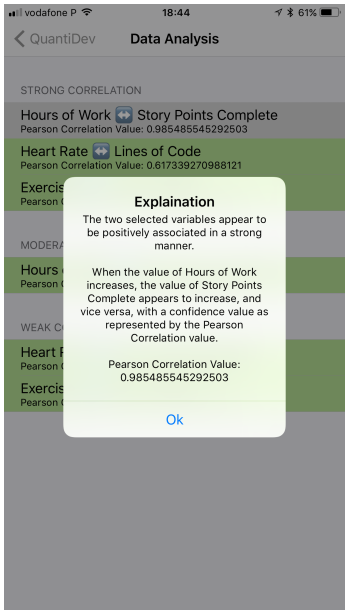


Figure F.7: Data pearson analysis for *QuantiDev*

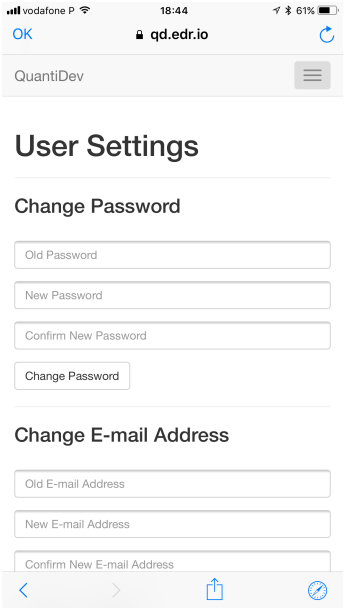


Figure F.8: In-app web browser for *QuantiDev*

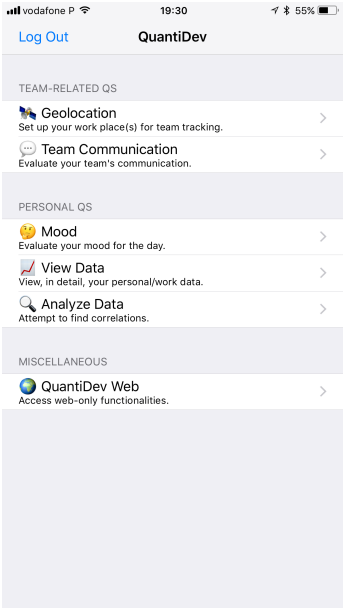


Figure F.9: Main view for *QuantiDev* (which includes log out)

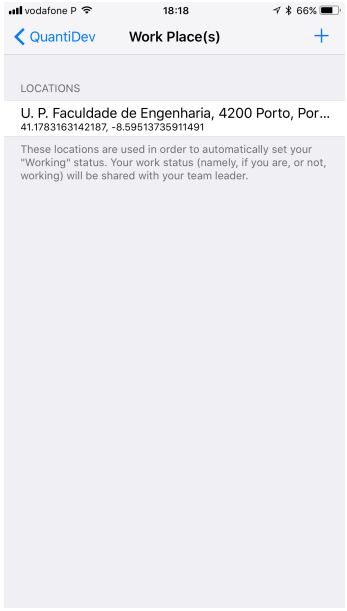


Figure F.10: Workplace locations view for *QuantiDev*

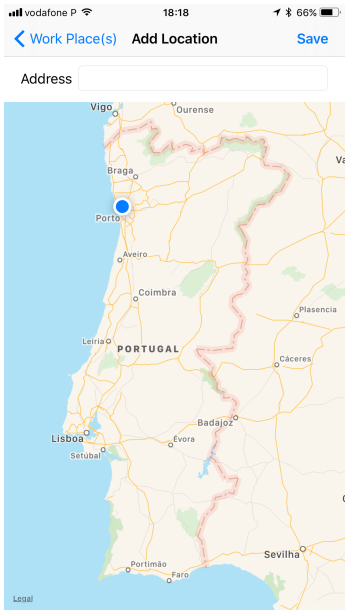


Figure F.11: New workplace location view for *QuantiDev*

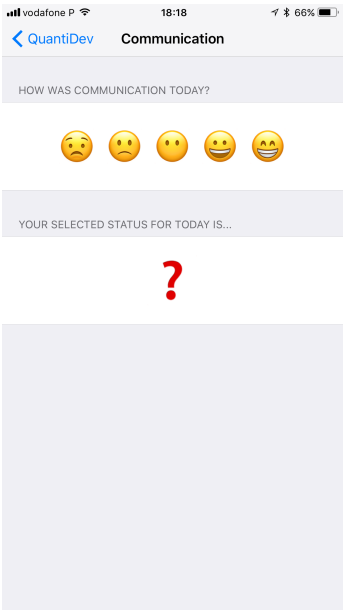
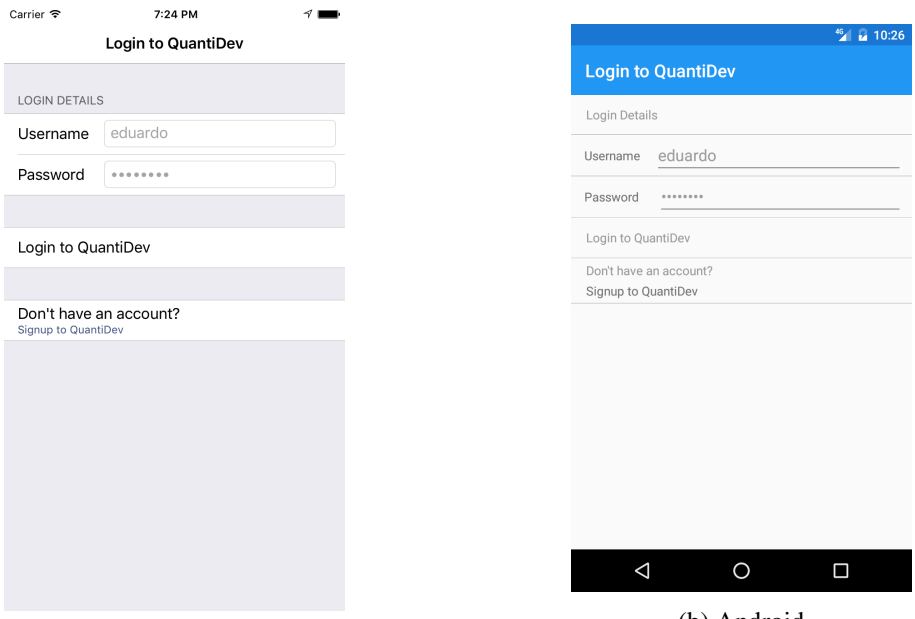


Figure F.12: Team communication view for *QuantiDev*

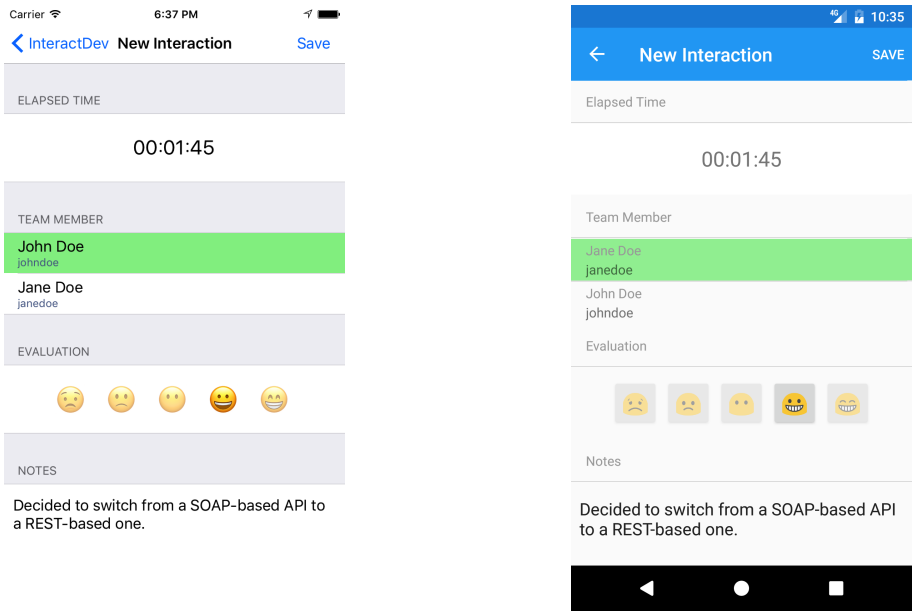
F.2 InteractDev



(a) iOS

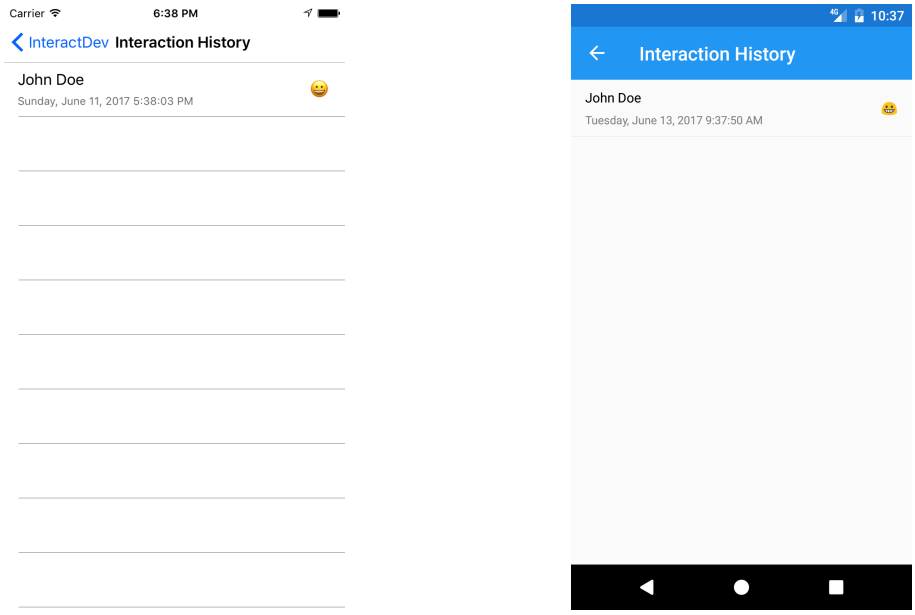
(b) Android

Figure F.13: Login view for *InteractDev*



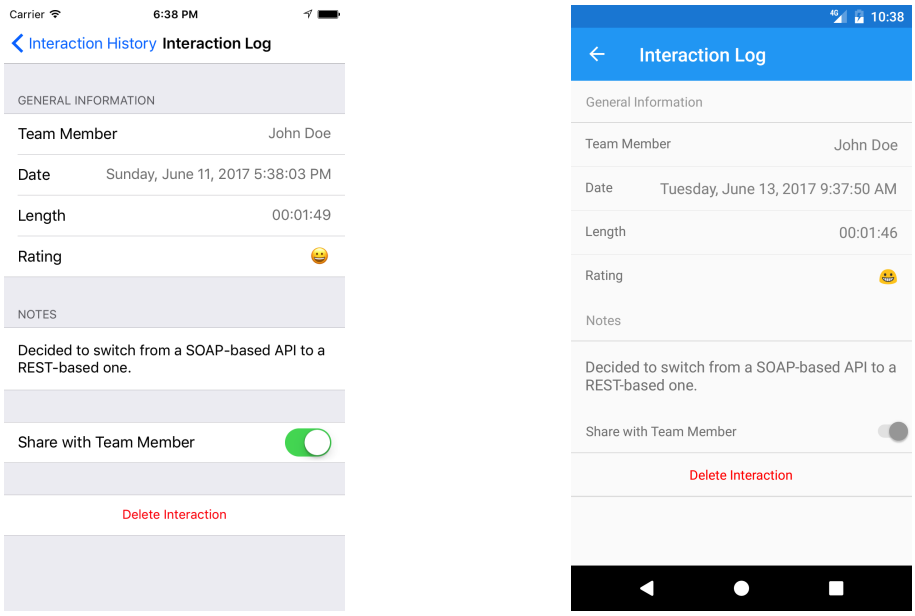
(a) iOS (b) Android

Figure F.14: New interaction view for *InteractDev*



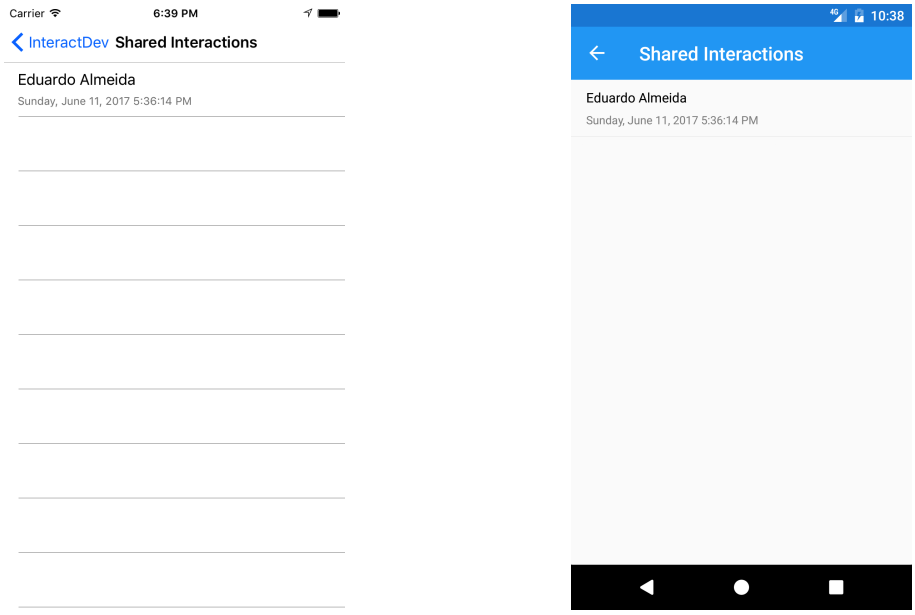
(a) iOS (b) Android

Figure F.15: Past interactions view for *InteractDev*



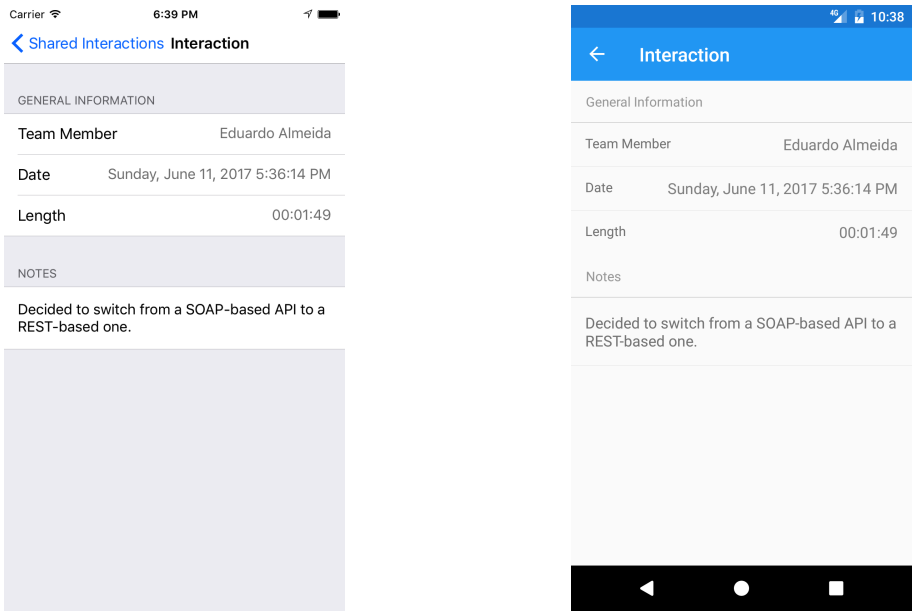
(a) iOS (b) Android

Figure F.16: Past interaction detail view for *InteractDev*



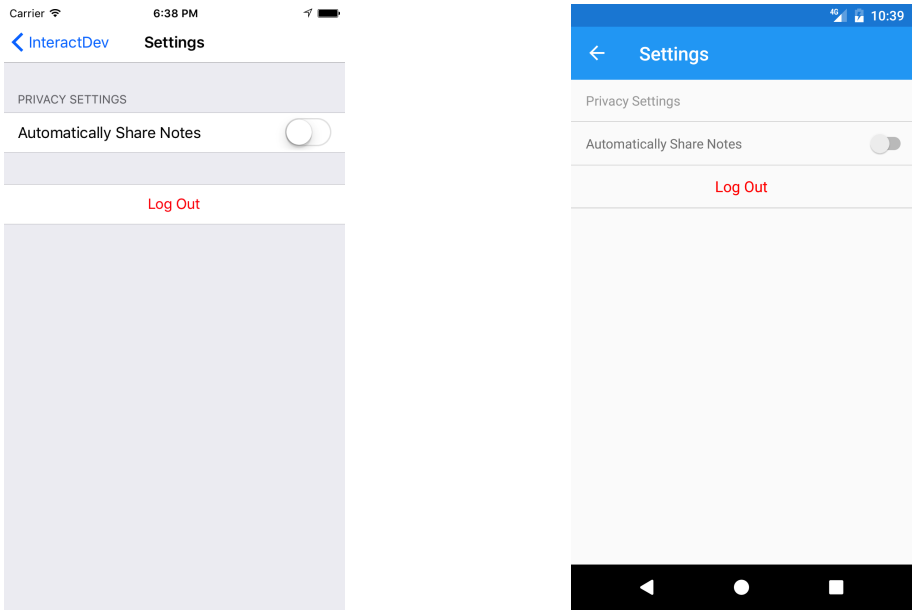
(a) iOS (b) Android

Figure F.17: Shared interactions view for *InteractDev*



(a) iOS (b) Android

Figure F.18: Shared interaction detail view for *InteractDev*



(a) iOS (b) Android

Figure F.19: Settings view for *InteractDev*

F.3 QuantiDev Web

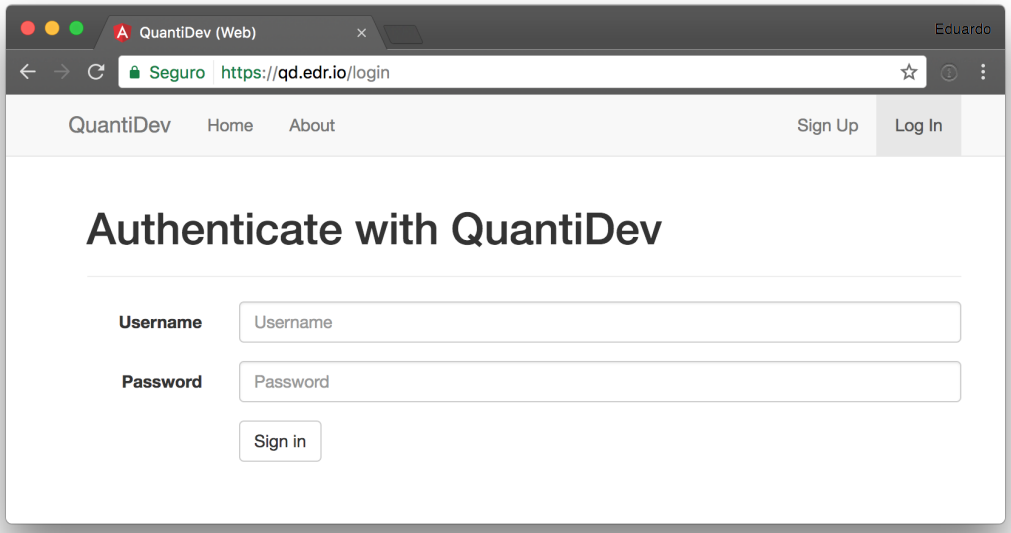


Figure F.20: Login view for *QuantiDev Web*

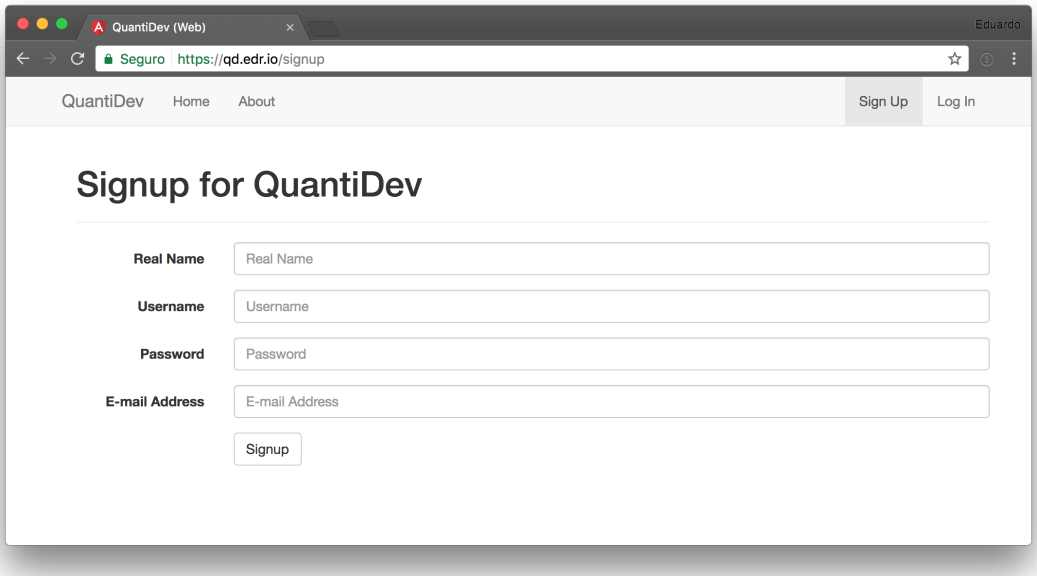


Figure F.21: Registration view for *QuantiDev Web*

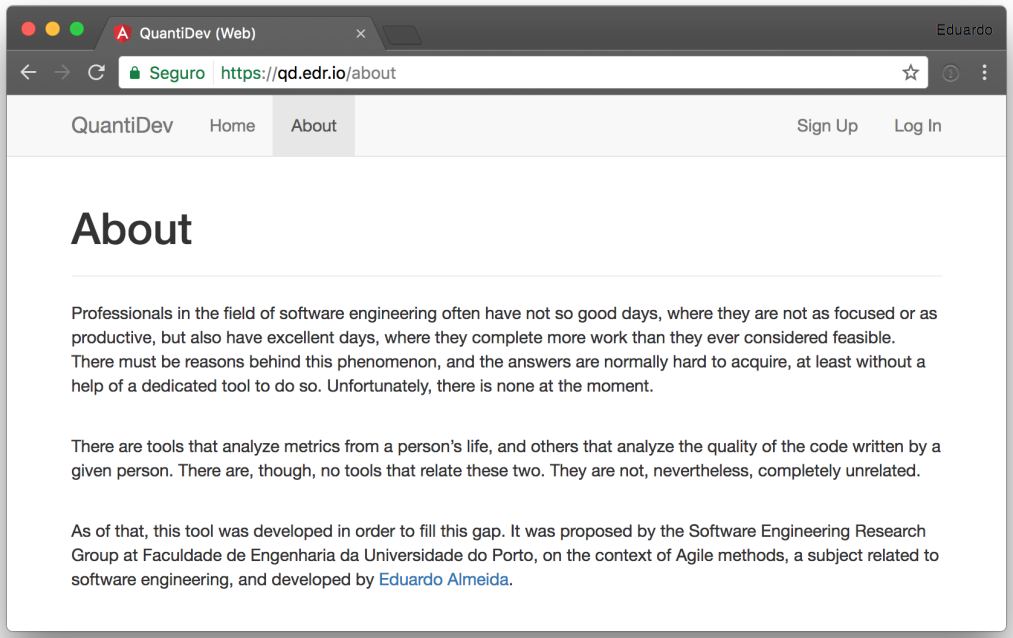


Figure F.22: About view for *QuantiDev Web*

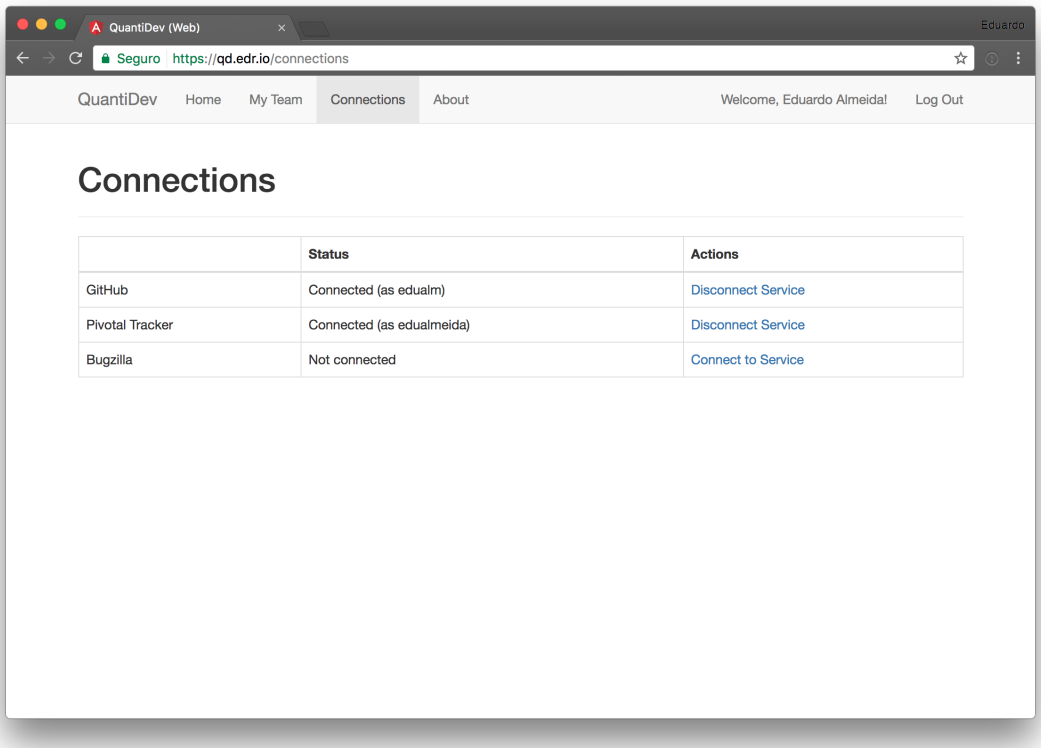


Figure F.23: Connections view for *QuantiDev Web*

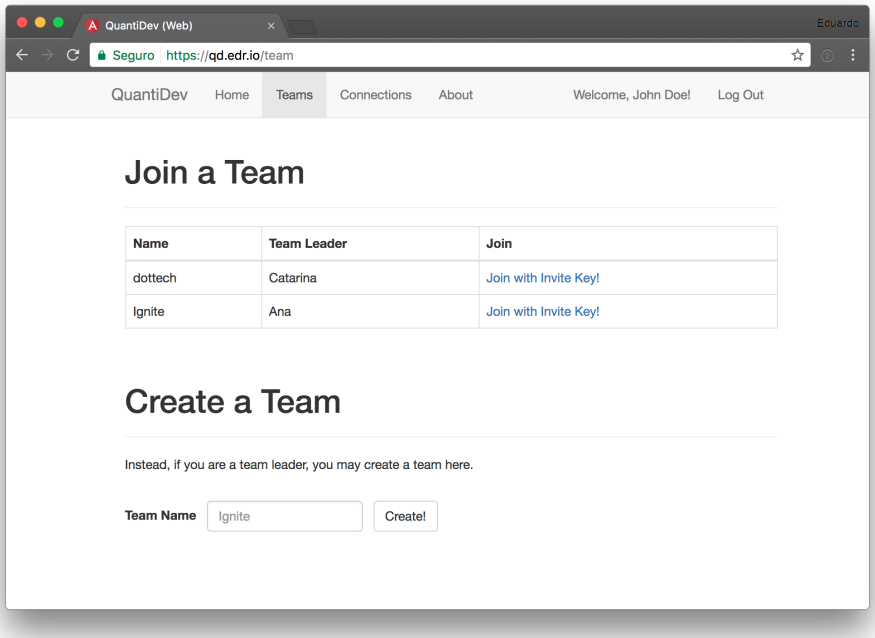


Figure F.24: Team listing view for *QuantiDev Web*

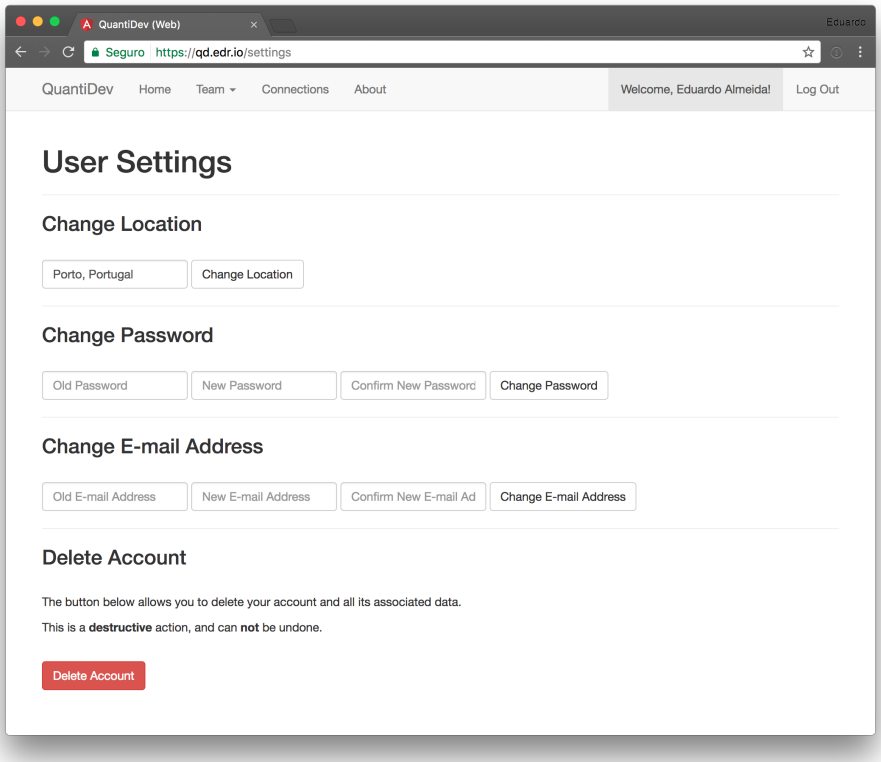


Figure F.25: Settings view for *QuantiDev Web*

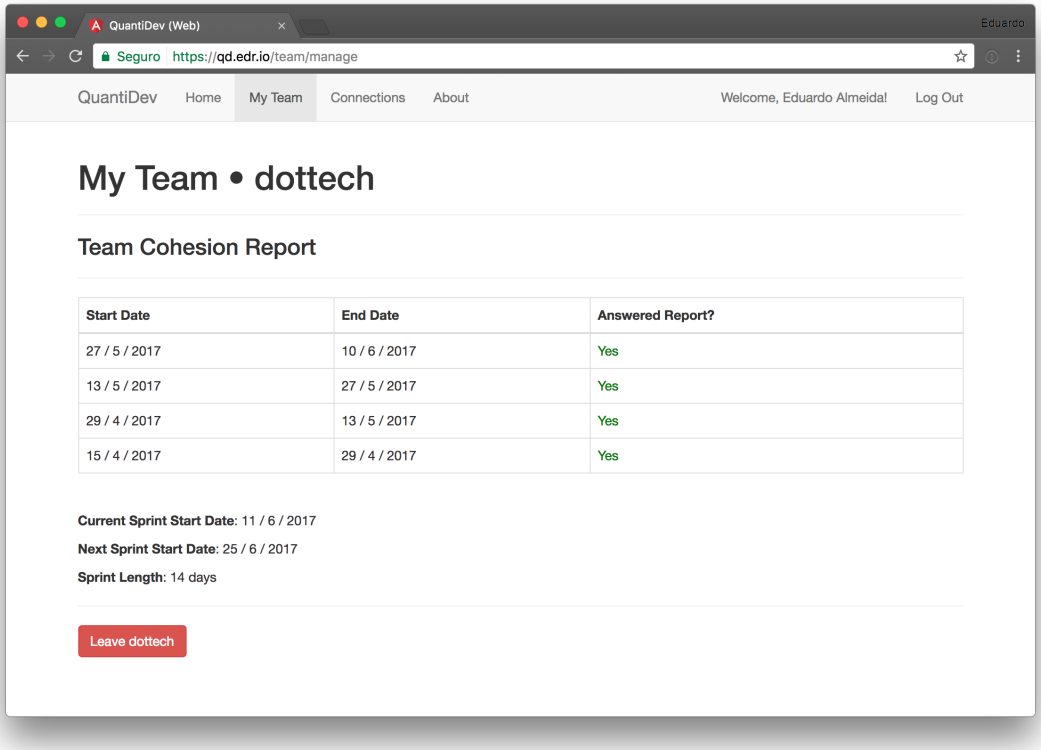


Figure F.26: Team view (as a team member) for *QuantiDev Web*

Figure F.27: Cohesion questionnaire for *QuantiDev Web*

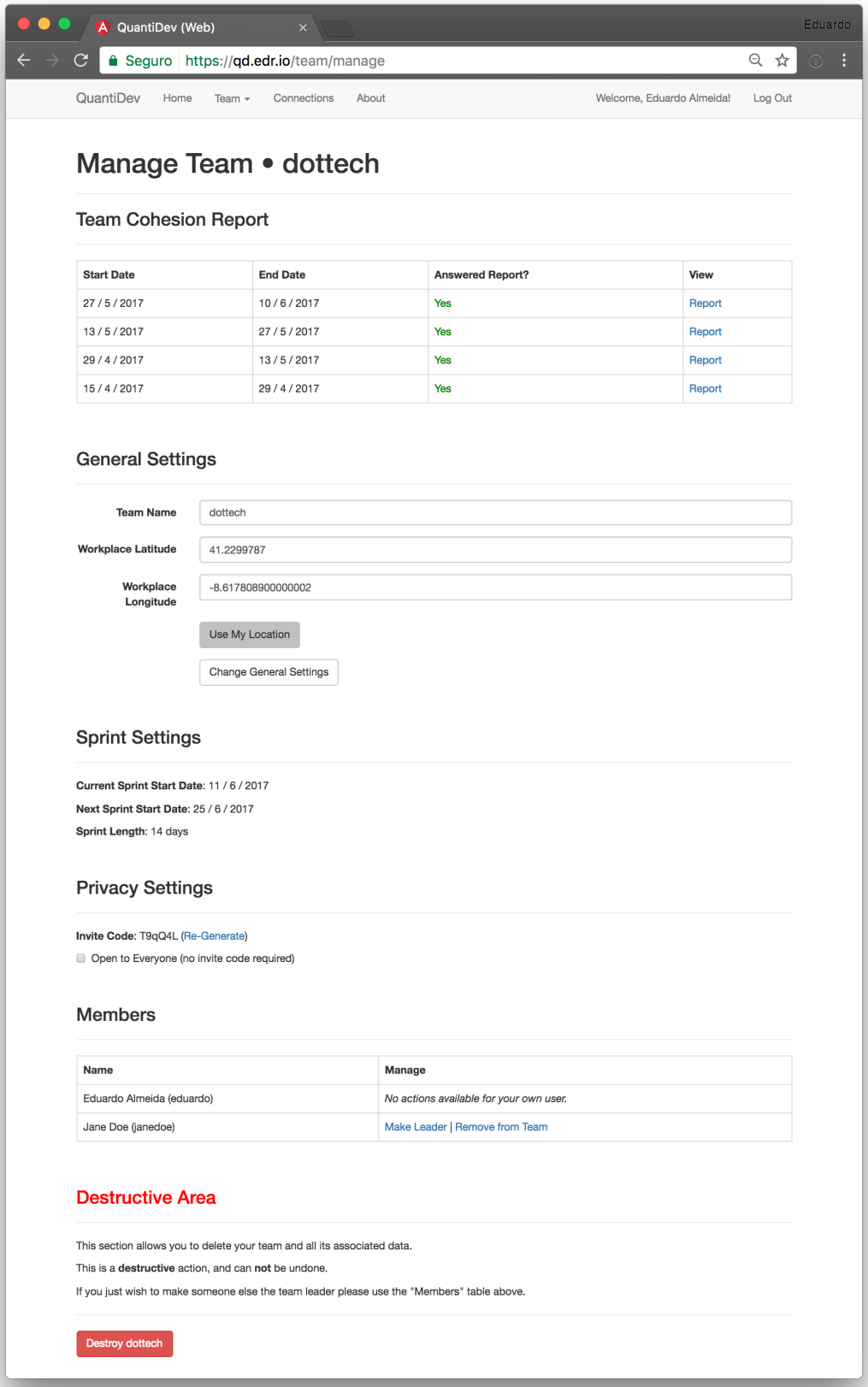


Figure F.28: Team management view for *QuantiDev Web*

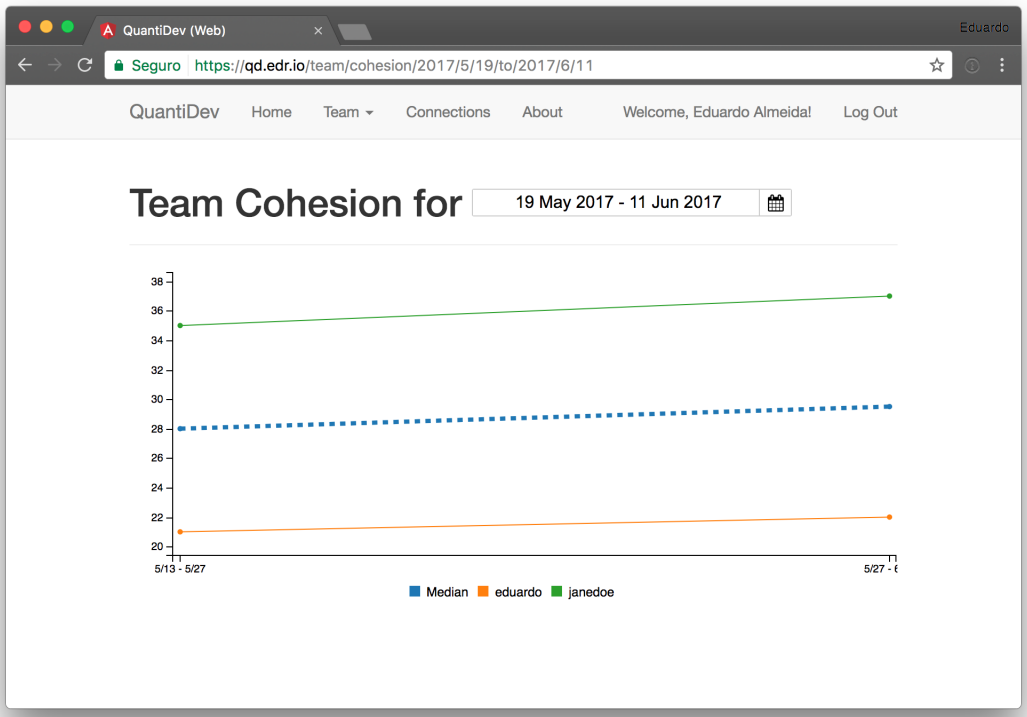


Figure F.29: Team chart (cohesion) view for *QuantiDev Web*

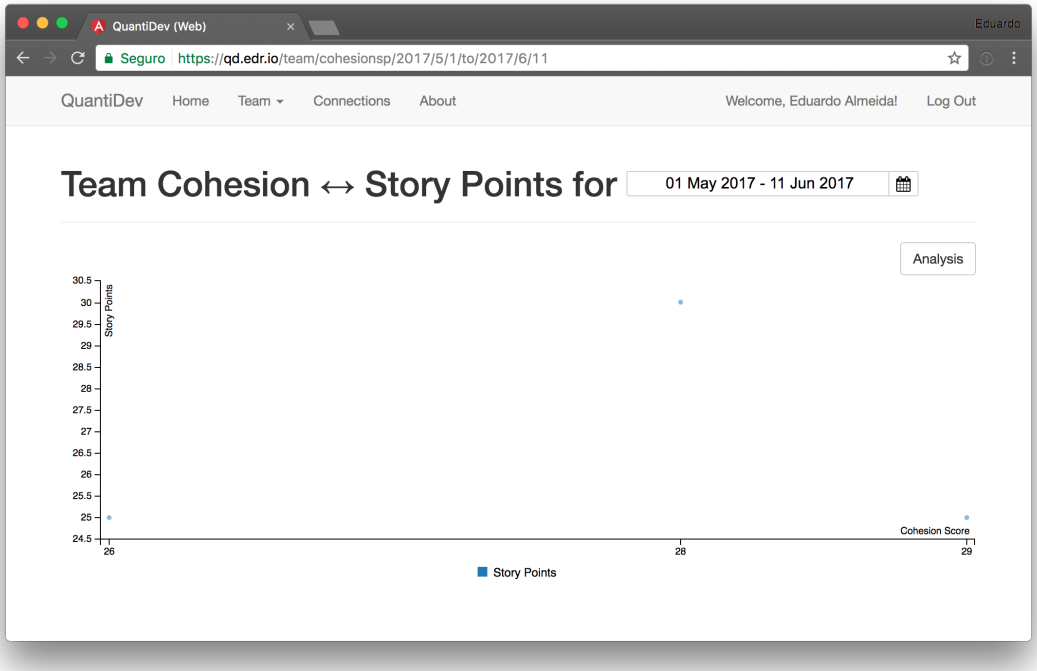


Figure F.30: Team chart (scatter) view for *QuantiDev Web*

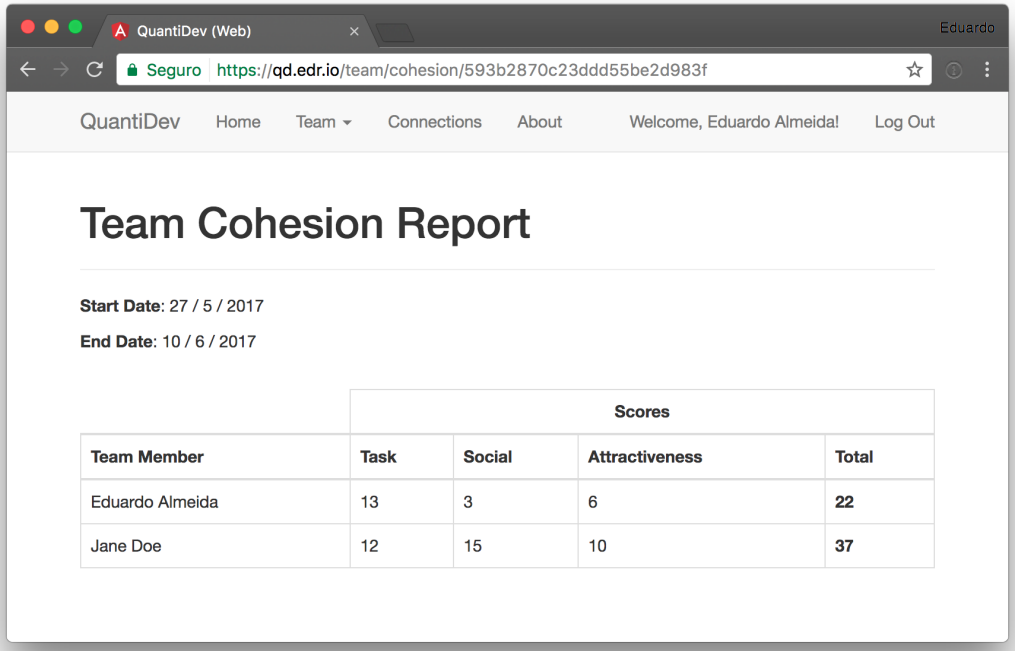


Figure F.31: Team cohesion report view for *QuantiDev Web*

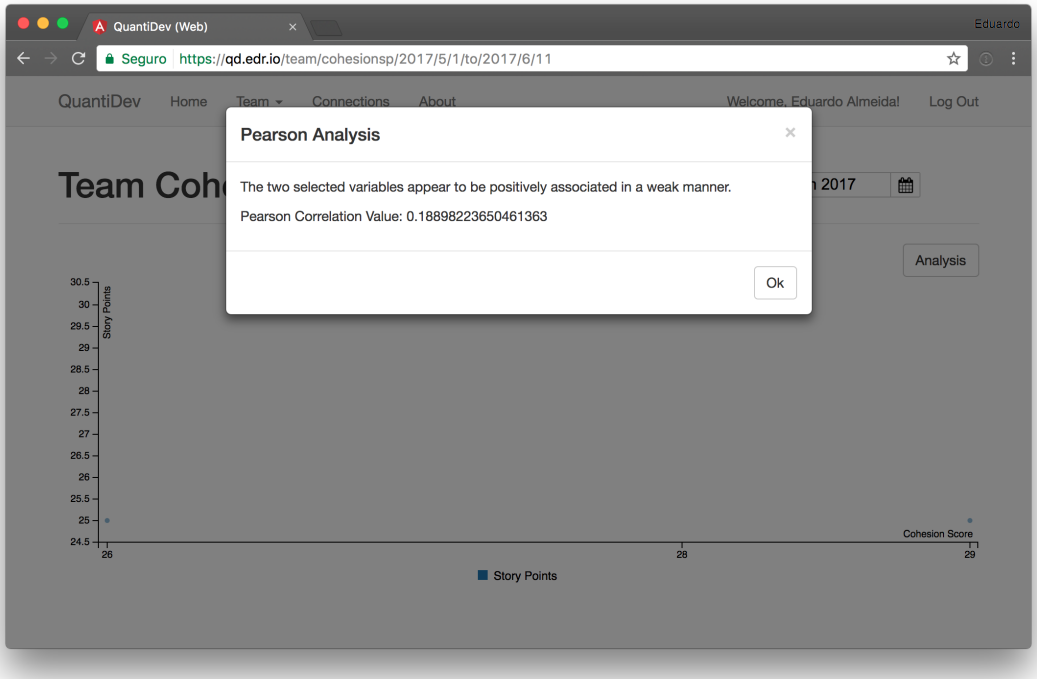


Figure F.32: Team scatter view (pearson analysis) for *QuantiDev Web*

Screenshots

Appendix G

Project READMEs

G.0.1 quantiserver

Prerequisites

- Node.js (tested with 7.x and 8.x, other versions may or may not work)
- MongoDB (tested with 3.4, other versions may or may not work)

Installation

1. Clone the repository;
2. Run `npm install`;
3. Copy *config.dist.js* as *config.js*;
4. Edit the configuration file (*config.js*) as required;
5. Run the project, either with a process manager such as pm2 or by running the command `npm start` directly.

License

MIT

G.0.2 QuantiDev for *iOS*

Prerequisites

- macOS 10.12+
- Xcode 8+ (with the iOS 10 SDK installed)
- Carthage

Compilation

1. Clone the repository;
2. Acquire the dependencies with `carthage bootstrap`;
3. Change the `ServerURL` and `QuantiWebURL` on *QuantiServer.swift*, so that they point to the public url of *quantiserver* and *QuantiDev Web* (respectively);
4. Run or archive the project with Xcode.

License

MIT

G.0.3 InteractDev for *iOS* and *Android*

Prerequisites

- macOS
  - macOS 10.11+
  - Xamarin Studio
  - Android SDK (tested with API level 23 and 25) – **Required for Android compilation/deployment**
  - Xcode + iOS SDK (tested with iOS 10 SDK) – **Required for iOS compilation/deployment**
- Windows
  - Windows 10+
  - Visual Studio 2015 with Xamarin
  - Android SDK (tested with API level 23 and 25) – **Required for Android compilation/deployment**
  - **iOS compilation/deployment is not supported under Windows**

Compilation

1. Clone the repository;
2. Open the project in either *Visual Studio* or *Xamarin Studio*;
3. Run or archive the project, which should also acquire the *NuGet* dependencies.

License

MIT

G.0.4 QuantiDev Web

Development Prerequisites

- Node.js (tested with 7.x and 8.x, other versions may or may not work)
- @angular/cli (tested with 1.0rc2 and 1.1.0, other versions may or may not work)

Deployment Prerequisites

- HTTP server (Apache, lightppd, nginx, ...)

Installation

1. Clone the repository;
2. Edit *src/environments/environment.ts* and *src/environments/environment.prod.ts* to point to the public URL of quantiserver;
3. Compile with `ng build`;
4. Copy the *dist* folder to a public folder on your http server.
5. Make sure to configure your http server, as described in Angular documentation - Server configuration (https://angular.io/guide/deployment#server-configuration)

License

MIT

Appendix H

MIT License

Copyright 2017 Eduardo Almeida

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the “SOFTWARE”), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.T WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

(Adapted from the MIT license available at https://opensource.org/licenses/MIT)

MIT License

References

[Ala17] Alamofire. Alamofire, 2017. Repository on GitHub. URL: https://github.com/Alamofire/Alamofire [accessed May 28th, 2017].

[All17a] Agile Alliance. 12 principles behind the agile manifesto, 2017. URL: https://www.agilealliance.org/agile101/12-principles-behind-the-agile-manifesto/ [accessed June 16th, 2017].

[All17b] Agile Alliance. What is agile?, 2017. URL: https://www.agilealliance.org/agile101/ [accessed June 16th, 2017].

[AMN+13] Mohd Shahdi Ahmad, Nur Emyra Musa, Rathidevi Nadarajah, Rosilah Hassan and Nor Effendy Othman. Comparison between android and iOS Operating System in terms of security. In *Information Technology in Asia (CITA), 2013 8th International Conference on*, pages 1–4. IEEE, 2013.

[And17] Android. Building Apps for Wearables, 2017. URL: https://developer.android.com/training/building-wearables.html [accessed January 20th, 2017].

[App16] Jurgen Appelo. *Managing for Happiness: Games, Tools, and Practices to Motivate Any Team*. Wiley, 2016. URL: https://www.amazon.com/Managing-Happiness-Games-Practices-Motivate-ebook/dp/B01GQWKHXK%3FSubscriptionId%3D0JYN1NVW651KCA56C102%26tag%3Dtechkie-20%26linkCode%3Dxm2%26camp%3D2025%26creative%3D165953%26creativeASIN%3DB01GQWKHXK.

[App17] Apple. iOS - Health - Apple, 2017. URL: https://www.apple.com/ios/health/ [accessed January 30th, 2017].

[Atl17a] Atlassian. All Atlassian Gadgets - Atlassian Documentation, 2017. URL: https://confluence.atlassian.com/display/GADGETS/All+Atlassian+Gadgets [accessed January 31th, 2017].

[Atl17b] Atlassian. Bitbucket, 2017. URL: https://bitbucket.org/product [accessed January 16th, 2017].

[B+90] Brian Berliner et al. CVS II: Parallelizing software development. In *Proceedings of the USENIX Winter 1990 Technical Conference*, volume 341, page 352, 1990.

[BA96] J Martin Bland and Douglas G Altman. Statistics notes: Measurement error. *BMJ*, 312(7047):1654, 1996. URL: http://www.bmj.com/

content/312/7047/1654, arXiv:http://www.bmj.com/content/312/7047/1654, doi:10.1136/bmj.312.7047.1654.

[BAC17] BACtrack. BACtrack, 2017. URL: https://www.bactrack.com/ [accessed January 30th, 2017].

[ban14] bandla. Runtastic release Sleep Better sleep tracker application, 2014. URL: http://www.gadgetdetail.com/runtastic-release-sleep-better-sleep-tracker-application/ [accessed January 30th, 2017].

[BD07] Sian L Beilock and Marci S DeCaro. From poor performance to success under stress: working memory, strategy selection, and mathematical problem solving under pressure. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 33(6):983, 2007.

[Bri13] Briandoll. 10 Million Repositories, 2013. URL: https://github.com/blog/1724-10-million-repositories [accessed January 19th, 2017].

[Bur16] Rachel Burger. 7 of the top agile project management software, 2016. URL: http://blog.capterra.com/agile-project-management-software/ [accessed July 2nd, 2017].

[Cho13] Kristina Chodorow. *MongoDB: the definitive guide*. " O'Reilly Media, Inc.", 2013.

[Coh88] Jacob Cohen. Statistical power analysis for the behavioral sciences Lawrence Earlbaum Associates. *Hillsdale, NJ*, pages 20–26, 1988.

[Com17] Mercurial Community. Mercurial SCM, 2017. URL: https://www.mercurial-scm.org [accessed January 19th, 2017].

[coo14] coolaj86. btoa, 2014. Package on the npm registry. URL: https://www.npmjs.com/package/btoa [accessed May 28th, 2017].

[CP00] Sally A. Carless and Caroline De Paola. The measurement of cohesion in work teams. *Small Group Research*, 31(1):71–88, 2000. URL: http://dx.doi.org/10.1177/104649640003100104, arXiv:http://dx.doi.org/10.1177/104649640003100104, doi:10.1177/104649640003100104.

[CS02] Ben Collins-Sussman. The subversion project: buiding a better CVS. *Linux Journal*, 2002(94):3, 2002.

[dan17] danielgindi. Charts, 2017. Repository on GitHub. URL: https://github.com/danielgindi/Charts.

[dAS09a] B. de Alwis and J. Sillito. Why are software projects moving from centralized to decentralized version control systems. In *2009 ICSE Workshop on Cooperative and Human Aspects on Software Engineering*, pages 36–39, May 2009. doi:10.1109/CHASE.2009.5071408.

[dAS09b] Brian de Alwis and Jonathan Sillito. Why Are Software Projects Moving from Centralized to Decentralized Version Control Systems. In *Proceedings*

of the 2009 ICSE Workshop on Cooperative and Human Aspects on Software Engineering, CHASE '09, pages 36–39, Washington, DC, USA, 2009. IEEE Computer Society. URL: http://dx.doi.org/10.1109/CHASE.2009.5071408, doi:10.1109/CHASE.2009.5071408.

[Dav15] Christopher W. H. Davis. *Agile Metrics in Action: Measuring and Enhancing the Performance of Agile Teams*. Manning Publications, 2015. URL: https://www.amazon.com/Agile-Metrics-Action-Measuring-Performance/dp/1617292486%3FSubscriptionId%3D0JYN1NVW651KCA56C102%26tag%3Dtechkie-20%26linkCode%3Dxm2%26camp%3D2025%26creative%3D165953%26creativeASIN%3D1617292486.

[DB16a] Maria Dias and José Borges. Introducing evaluation. Interação Pessoa-Computador, MIEIC - FEUP — 2016/2017 Class Handouts, 2016.

[DB16b] Maria Dias and José Borges. The process of interaction design. Interação Pessoa-Computador, MIEIC - FEUP — 2016/2017 Class Handouts, 2016.

[def17] defunctzombie. bcrypt, 2017. Package on the npm registry. URL: https://www.npmjs.com/package/bcrypt [accessed May 28th, 2017].

[Dev11] Android Developers. What is Android?, 2011. URL: http://developer.android.com/guide/basics/what-is-android [accessed May 4th, 2017].

[Dev16] Apple Developer. Region Monitoring and iBeacon, 2016. URL: https://developer.apple.com/library/content/documentation/UserExperience/Conceptual/LocationAwarenessPG/RegionMonitoring/RegionMonitoring.html [accessed March 22th, 2017].

[Dic17a] Dictionary.com - Breath Analyzer. Jan 2017. URL: http://www.dictionary.com/browse/breath-analyzer.

[Dic17b] Dictionary.com - Report, Jun 2017. URL: http://www.dictionary.com/browse/report [accessed June 5th, 2017].

[dou17a] dougwilson. cors, 2017. Package on the npm registry. URL: https://www.npmjs.com/package/cors [accessed May 28th, 2017].

[dou17b] dougwilson. express, 2017. Package on the npm registry. URL: https://www.npmjs.com/package/express [accessed May 28th, 2017].

[dpr08] dprice. Concurrent Versions System - News, 2008. URL: https://savannah.nongnu.org/news/?group=cvs [accessed January 30th, 2017].

[eli16] eliaskg. randomstring, 2016. Package on the npm registry. URL: https://www.npmjs.com/package/randomstring [accessed May 28th, 2017].

[Ern16] Michael Ernst. Version control concepts and best practices, 2016. URL: https://homes.cs.washington.edu/~mernst/advice/version-control.html [accessed January 30th, 2017].

[Fac17] Facebook. Moves - Activity Diary for iPhone and Android, 2017. URL: https://moves-app.com/ [accessed January 30th, 2017].

[Fes50] Leon Festinger. Informal social communication. *Psychological review*, 57(5):271, 1950.

[Few04] S Few. Enie, meenie, minie, moe: selecting the right graph for your message. *Intelligent Enterprise*, page 13, 2004.

[Fit17] Inc. Fitbit. Fitbit - Track your runs, walks, and more, 2017. URL: https://www.fitbit.com/ [accessed January 19th, 2017].

[FKL13] John Fisher, D Koning and AP Ludwigsen. Utilizing Atlassian JIRA for Large-Scale Software Development Management. In *14th International Conference on Accelerator & Large Experimental Physics Control Systems (ICALEPCS)*, 2013.

[Fou06] Mozilla Foundation. The Bugzilla Guide, 2006. URL: https://www.bugzilla.org/docs/2.16/html/ [accessed January 20th, 2017].

[Fou15] Free Software Foundation. GNU RCS, 2015. URL: https://www.gnu.org/software/rcs/ [accessed January 20th, 2017].

[Fou16] Apache Software Foundation. Apache Subversion, 2016. URL: https://subversion.apache.org/ [accessed January 30th, 2017].

[Geo91] Jennifer M George. State or trait: Effects of positive mood on prosocial behaviors at work. *Journal of applied Psychology*, 76(2):299, 1991.

[GLL15] Yiwen Gao, He Li and Yan Luo. An empirical study of wearable technology acceptance in healthcare. *Industrial Management and Data Systems*, 115(9):1704–1723, Oct 2015. doi:10.1108/imds-03-2015-0087.

[gma17] gmazzamuto. ng2-google-charts, 2017. npm. URL: https://www.npmjs.com/package/ng2-google-charts [accessed May 28th, 2017].

[gok17] goktugyil. EZLoadingActivity, 2017. Repository on GitHub. URL: https://github.com/goktugyil/EZLoadingActivity [accessed May 28th, 2017].

[Goo13] Google. Google Health, 2013. URL: https://www.google.com/intl/en_us/health/about/ [accessed January 30th, 2017].

[Goo17] Google. Google Charts, 2017. URL: https://developers.google.com/chart/ [accessed May 28th, 2017].

[Gor12] Whitson Gordon. Understanding oauth: What happens when you log into a site with google, twitter, or facebook, 2012. URL: https://lifehacker.com/5918086/understanding-oauth-what-happens-when-you-log-into-a-site [accessed June 2nd, 2017].

[Gre16] Jay Greene. Microsoft agrees to acquire xamarin, 2016. URL: http://www.wsj.com/articles/microsoft-agrees-to-acquire-xamarin-inc-1456340494 [accessed March 2nd, 2017].

[GSW75] FRANCESCO GAMBERALE, LOTTEN STRINDBERG and INGER WAHLBERG. Female work capacity during the menstrual cycle: Physiological and psychological reactions. *Scandinavian Journal of Work, Environment and Health*, 1(2):120–127, 1975. URL: http://www.jstor.org/stable/40964545.

[Ham14] A. E. D. Hamouda. Using Agile Story Points as an Estimation Technique in CMMI Organizations. In *2014 Agile Conference*, pages 16–23, July 2014. doi:10.1109/AGILE.2014.11.

[Hea17] HealthIT. Model Privacy Notice, 2017. URL: https://www.healthit.gov/policy-researchers-implementers/model-privacy-notice-mpn [accessed May 28th, 2017].

[ich17] ichernev. moment, 2017. Package on the npm registry. URL: https://www.npmjs.com/package/moment [accessed May 28th, 2017].

[IDC16a] IDC. Smartphone OS Market Share, 2016 Q3, 2016. URL: https://www.idc.com/promo/smartphone-market-share/os [accessed January 22nd, 2017].

[IDC16b] IDC. Smartwatch Market Declines 51.6and Vendors Realign, IDC Finds, 2016. URL: https://www.idc.com/getdoc.jsp?containerId=prUS41875116 [accessed January 20th, 2017].

[IL82] Tomasz Imielinski and Witold Lipski, Jr. A Systematic Approach to Relational Database Theory. In *Proceedings of the 1982 ACM SIGMOD International Conference on Management of Data*, SIGMOD '82, pages 8–14, New York, NY, USA, 1982. ACM. URL: http://doi.acm.org/10.1145/582353.582356, doi:10.1145/582353.582356.

[Ins17] SAS Insights. Why your brain needs data visualization, 2017. URL: https://www.sas.com/en_us/insights/articles/analytics/why-your-brain-needs-data-visualization.html [accessed May 18th, 2017].

[JA91] Cary Jensen and Loy Anderson. *Harvard Graphics 3: The Complete Reference*. Mcgraw-Hill Osborne Media, 1991. URL: https://www.amazon.com/Harvard-Graphics-3-Complete-Reference/dp/0078817498?SubscriptionId=0JYN1NVW651KCA56C102&tag=techkie-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=0078817498.

[jar17] jaredhanson. passport, 2017. Package on the npm registry. URL: https://www.npmjs.com/package/passport [accessed May 28th, 2017].

[Jaw17] Jawbone. UP by Jawbone | Fitness trackers for a healthier you, 2017. URL: https://jawbone.com/ [accessed January 19th, 2017].

[JH11] Tomasz Kossowski Jan Hauke. Comparison of values of Pearson's and Spearman's correlation coefficients on the same sets of data. 2011. URL: https://www.degruyter.com/downloadpdf/j/quageo.2011.30.issue-2/v10117-011-0021-1/v10117-011-0021-1.pdf.

[jso17] jsonmaur. crypto-extra, 2017. Package on the npm registry. URL: https://www.npmjs.com/package/crypto-extra [accessed May 28th, 2017].

[Kah] Jordan Kahn. Runtastic releases 'Sleep Better' app for tracking sleep habits without additional hardware | 9to5Mac. URL: https://9to5mac.com/2014/11/06/runtastic-releases-sleep-better/ [accessed January 30th, 2017].

[kai16] kaizensoze. express, 2016. Package on the npm registry. URL: https://www.npmjs.com/package/github [accessed May 28th, 2017].

[kek17a] kekeh. mydatepicker, 2017. Package on the npm registry. URL: https://www.npmjs.com/package/mydatepicker [accessed May 28th, 2017].

[kek17b] kekeh. mydatepicker, 2017. Package on the npm registry. URL: https://www.npmjs.com/package/mydaterangepicker [accessed May 28th, 2017].

[KFY+05] Matthew C Keller, Barbara L Fredrickson, Oscar Ybarra, Stéphane Côté, Kareem Johnson, Joe Mikels, Anne Conway and Tor Wager. A warm heart and a clear head: The contingent effects of weather on mood and cognition. *Psychological Science*, 16(9):724–731, 2005.

[KGB+14] 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*, MSR 2014, pages 92–101, New York, NY, USA, 2014. ACM. URL: http://doi.acm.org/10.1145/2597073.2597074, doi:10.1145/2597073.2597074.

[KI06] Steve WJ Kozlowski and Daniel R Ilgen. Enhancing the effectiveness of work groups and teams. *Psychological science in the public interest*, 7(3):77–124, 2006.

[Kup89] Ron I Kuper. Dependency-directed localization of software bugs. 1989. URL: http://hdl.handle.net/1721.1/6838.

[Lab17] Pivotal Labs. Pivotal Tracker, 2017. URL: https://www.pivotaltracker.com/ [accessed January 31th, 2017].

[LF16] Cella Lao Rousseau Luke Filipowicz, Allyson Kazmucha. How to set goals and view progress in Activity for Apple Watch, 2016. URL: http://www.imore.com/how-set-calorie-goal-activity-apple-watch [accessed January 31th, 2017].

[Lib17] Kent Library. Pearson Correlation, 2017. URL: http://libguides.library.kent.edu/SPSS/PearsonCorr.

[man16] manuelbieh. express, 2016. Package on the npm registry. URL: https://www.npmjs.com/package/geolib [accessed May 28th, 2017].

[mas16] masayuki0812. c3, 2016. Package on the npm registry. URL: https://www.npmjs.com/package/c3 [accessed May 28th, 2017].

[mbo17] mbostock. d3, 2017. Package on the npm registry. URL: https://www.npmjs.com/package/d3 [accessed May 28th, 2017].

[Mic17] Microsoft. Microsoft.Net.Http, 2017. Package on the NuGet registry. URL: https://www.nuget.org/packages/Microsoft.Net.Http/ [accessed May 28th, 2017].

[Mid07] Donna J Middaugh. Presenteeism: sick and tired at work. *Dermatology Nursing*, 19(2):172, 2007.

[MM17] Chance Miller and Chance Miller. Apple overtakes Fitbit to become world top wearables vendor thanks to Apple Watch growth, 2017. URL: https://9to5mac.com/2017/05/04/apple-overtakes-fitbit-to-become-worlds-top-wearables/ [accessed May 4th, 2017].

[MPP00] Mickey T. Trockel MS, Michael D. Barnes PhD and Dennis L. Egget PhD. Health-Related Variables and Academic Performance Among First-Year College Students: Implications for Sleep and Other Behaviors. *Journal of American College Health*, 49(3):125–131, 2000. URL: http://dx.doi.org/10.1080/07448480009596294, arXiv:http://dx.doi.org/10.1080/07448480009596294, doi:10.1080/07448480009596294.

[New17] Newtonsoft. Newtonsoft.Json, 2017. Repository on GitHub. URL: https://github.com/JamesNK/Newtonsoft.Json [accessed May 28th, 2017].

[Nie93] Jakob Nielsen. *Usability engineering*. Academic Press, Boston, 1993.

[nij16] nijikokun. unirest, 2016. Package on the npm registry. URL: https://www.npmjs.com/package/unirest [accessed May 28th, 2017].

[otWoE17] University of the West of England. Data Analysis [online]. 2017. URL: http://learntech.uwe.ac.uk/da/Default.aspx?pageid=1442 [accessed May 17th, 2017].

[PC07] Yangil Park and Jengchung V. Chen. Acceptance and adoption of the innovative use of smartphone. *Industrial Management & Data Systems*, 107(9):1349–1365, 2007. URL: http://dx.doi.org/10.1108/02635570710834009, arXiv:http://dx.doi.org/10.1108/02635570710834009, doi:10.1108/02635570710834009.

[pcl17] Cross-Platform Development with the Portable Class Library [online]. 2017. URL: https://msdn.microsoft.com/en-us/library/gg597391(v=vs.110).aspx [accessed June 5th, 2017].

[PCSF08] C Michael Pilato, Ben Collins-Sussman and Brian W Fitzpatrick. *Version control with subversion*. "O'Reilly Media, Inc.", 2008.

[Peb16] Pebble. Pebble, 2016. URL: https://www.pebble.com/health [accessed January 16th, 2017].

[Per15] Sarah Perez. Runkeeper's New Apple Watch App Lets You Ditch Your iPhone When Tracking Your Runs, 2015. URL: https://techcrunch.com/2015/10/22/runkeepers-new-apple-watch-app-lets-you/ [accessed January 31th, 2017].

[PLTC16] R. Prikladnicki, C. Lassenius, E. Tian and J. C. Carver. Trends in Agile: Perspectives from the Practitioners. *IEEE Software*, 33(6):20–22, Nov 2016. doi:10.1109/MS.2016.152.

[POL96] JANET POLIVY. Psychological consequences of food restriction. *Journal of the American Dietetic Association*, 96(6):589 – 592, 1996. URL: //www.sciencedirect.com/science/article/pii/S0002822396001617, doi:http://dx.doi.org/10.1016/S0002-8223(96)00161-7.

[Rea14] Realm. Realm, 2014. URL: https://news.realm.io/news/introducing-realm/ [accessed May 2nd, 2017].

[Rea17a] Realm. Realm Cocoa, 2017. Repository on GitHub. URL: https://github.com/realm/realm-cocoa [accessed May 28th, 2017].

[Rea17b] Realm. Realm DotNet, 2017. Repository on GitHub. URL: https://github.com/realm/realm-dotnet [accessed May 28th, 2017].

[rG17] runtastic GmbH. Sleep Better, 2017. URL: https://www.runtastic.com/en/apps/sleepbetter [accessed January 28th, 2017].

[Ric07] John Rice. *Mathematical statistics and data analysis*. Thomson/Brooks/Cole, Belmont, CA, 2007.

[RPP14] Reza Rawassizadeh, Blaine A. Price and Marian Petre. Wearables: Has the Age of Smartwatches Finally Arrived? *Commun. ACM*, 58(1):45–47, December 2014. URL: http://doi.acm.org/10.1145/2629633, doi:10.1145/2629633.

[Run17] Runkeeper. Runkeeper - Track your runs, walks, and more, 2017. URL: https://runkeeper.com/ [accessed January 19th, 2017].

[Sam17] Samsung. Samsung Mobile - Wearables, 2017. URL: http://www.samsung.com/us/mobile/wearables/ [accessed January 20th, 2017].

[SC05] Nicolas Serrano and Ismael Ciordia. Bugzilla, ITracker, and other bug trackers. *IEEE software*, 22(2):11–13, 2005.

[sgi17] sgimento. node-schedule, 2017. Package on the npm registry. URL: https://www.npmjs.com/package/node-schedule [accessed May 28th, 2017].

[smi15] smilingrob. pivotaltracker, 2015. Package on the npm registry. URL: https://www.npmjs.com/package/pivotaltracker [accessed May 28th, 2017].

[Sta99] Richard Stallman. The GNU operating system and the free software movement. 1999.

[Swa13] Melanie Swan. The quantified self: Fundamental disruption in big data science and biological discovery. *Big Data*, 1(2):85–99, 2013.

[TAF+12] Julian F. Thayer, Fredrik Ahs, Mats Fredrikson, John J. Sollers III and Tor D. Wager. A meta-analysis of heart rate variability and neuroimaging studies: Implications for heart rate variability as a marker of stress and health. *Neuroscience and Biobehavioral Reviews*, 36(2):747 – 756, 2012. URL: //www.sciencedirect.com/science/article/pii/S0149763411002077, doi:http://dx.doi.org/10.1016/j.neubiorev.2011.11.009.

[tba17] tbaggett. XFGloss, 2017. Repository on GitHub. URL: https://github.com/tbaggett/xfgloss [accessed May 28th, 2017].

[Tec17] Technofres. Fitbit Charge 2 review: Highly recommended if you are looking for a fitness tracker, 2017. URL: https://technofres.com/2016/11/02/fitbit-charge-2-review/ [accessed January 30th, 2017].

[TH17] Linus Torvalds and Junio Hamano. Git: Fast version control system, 2017. URL: http://git-scm.com [accessed January 19th, 2017].

[tra05] Addressing Sustainability in Transportation Systems: Definitions, Indicators, and Metrics. *Journal of Infrastructure Systems*, 11(1), 2005.

[twb16] twbs. bootstrap, 2016. Package on the npm registry. URL: https://www.npmjs.com/package/bootstrap [accessed May 28th, 2017].

[val17] valorkin. ng2-bootstrap, 2017. Package on the npm registry. URL: https://www.npmjs.com/package/ng2-bootstrap [accessed May 28th, 2017].

[Vil96] Bryan Vila. Tired cops: Probable connections between fatigue and the performance, health and safety of patrol officers. *American Journal of Police*, 15(2):51–92, 1996.

[vka17] vkarpov15. mongoose, 2017. Package on the npm registry. URL: https://www.npmjs.com/package/mongoose [accessed May 28th, 2017].

[vOiP11] Jos N. van Ommeren and Eva Gutiérrez i Puigarnau. Are workers with a long commute less productive? an empirical analysis of absenteeism. *Regional Science and Urban Economics*, 41(1):1 – 8, 2011. URL: //www.sciencedirect.com/science/article/pii/S0166046210000633, doi:http://dx.doi.org/10.1016/j.regsciurbeco.2010.07.005.

[Wan13] Jiguang Wang. Pearson correlation coefficient. In *Encyclopedia of Systems Biology*, pages 1671–1671. Springer, 2013.

[Wee17] Weebly. Cereal, 2017. Repository on GitHub. URL: https://github.com/Weebly/Cereal [accessed May 28th, 2017].

[Wes15] Liam Richard West. Strava: challenge yourself to greater heights in physical activity/cycling and running. *British journal of sports medicine*, 49(15):1024–1024, 2015.

[Wod17] Carey Wodehouse. What is AngularJS? The Powerful JavaScript Framework, 2017. URL: https://www.upwork.com/hiring/development/angularjs-basics/ [accessed April 17th, 2017].

[ZP14] W. Zhou and S. Piramuthu. Security/privacy of wearable fitness tracking IoT devices. In *2014 9th Iberian Conference on Information Systems and Technologies (CISTI)*, pages 1–5, June 2014. doi:10.1109/CISTI.2014.6877073.