Universidade de Coimbra
Faculdade de Ciências e Tecnologia

Quantification of SMEs' work environment, happiness, and productivity, using a Microservices Architecture

Daniel Fernandes Lopes

Thesis submitted for the degree of Master of Science
in Physics Engineering

September 2015

Universidade de Coimbra
Faculdade de Ciências e Tecnologia

Quantification of SMEs' work environment, happiness, and productivity, using a Microservices Architecture

Daniel Fernandes Lopes

Thesis submitted for the degree of Master of Science
in Physics Engineering

Under the supervision of:
Jorge Landeck, University of Coimbra
Rafael Jegundo, Whitesmith

September 2015

Abstract

On a study performed by Sierra-Cedar HR Systems in 2014, only 12% of organisations adopted some form of HR analytics. On the same study it was concluded that Quantified Workplaces out-perform other organisations - they see higher levels of financial performance, as well as positive HR and Return on Equity (ROE) outcomes [1]. Currently, most of the data-driven companies are only focused on the data provided by consumers - data-driven decisions are made practically only in product development, marketing, and pricing - or data provided by time tracking software installed on employees computers [1].

With the current state of the art, there's potential to build tools that quantify collective behaviour at an organisation on a real-time basis, thus providing insights that can help improving processes and practices that enable to achieve organisational goals. Tools that can be part of a framework for designing a strategy and preparing the business to execute it with iterative and adaptive Agile methodologies, rather than a biennial or triennial major initiative.

This master thesis work consists in building a platform to achieve such goals. Its challenges are: continuous collection, processing, and storage of SME's productivity, happiness, and office environment metrics; using a Microservices Architecture, that is modular for easy attachment and decoupling of services and devices; integration with Internet of Things devices, with the goal of measuring physical environmental quantities; building a data visualization platform to display the collected data for analysis.

At the end, the built platform was able quantify and display various variables related with productivity, happiness, and office environment, providing valuable information to the company at which the platform was applied - Whitesmith. The platform and its parts were praised by various people from the Quantified Self and Data Science communities, such as Ernesto Ramirez from Quantifedself.com, and Dan Kador from Keen IO.

Keywords: Quantified Self; Quantified Workplace; Internet of Things; Service Oriented Architecture; Microservices Architecture

Resumo

Num estudo feito pela Sierra-Cedar HR Systems em 2014, apenas 12% das organizações adotaram alguma forma de recolha de dados relacionados com recursos humanos. No mesmo estudo foi concluido que as *Quantified Workplaces* têm melhores resultados que outras organizações [1]. Atualmente, a maioria das empresas *data-driven* estão apenas focadas em dados provenientes dos consumidores - i.e. a nível do desenvolvimento de produto, marketing, e preços - ou em dados proveninentes de software de monitorização do tempo.

Através do atual estado da arte, há potencial para criar ferramentas que quantificam em tempo real o comportamento coletivo de uma organização, e consequentemente providenciar informação relevante também ele em tempo real. O uso destas ferramentas possibilita melhorar os processos e práticas que permitem atingir os objetivos organizacionais, usando metodologias iterativas e adaptativas Agile, ao invés de grandes iniciativas bianuais e trianuais.

O trabalho executado nesta Tese de Mestrado tem como objectivo a aplicação destes conceitos na construção de uma plataforma que permite atingir este tipo de metas. Os seus desafios são: coleção, processamento, e armazenamento continuado de dados relativos à produtividade, felicidade, e ambiente no escritório das PMEs; uso de uma arquitetura orientada a microserviços, que seja modular para fácil acoplamento e dissociação de serviços e dispositivos; integração com dispositivos da *Internet of Things*, com o objetivo de medir grandezas físicas; construção de uma plataform de visualização de dados para representação dos dados recolhidos para análise.

No final, a plataforma construida foi capaz de quantificar e representar várias variáveis relacionadas com a produtividade, felicidade, e ambiente no escritório, providenciando informação relevante à empresa na qual esta plataforma foi instalada - Whitesmith. A plataforma e as suas partes foram elogiadas por várias pessoas pertencentes às comunidades de Quantified Self e Data Science, como por exemplo Ernesto Ramirez da Quantifiedself.com, e Dan Kador da Keen IO.

Keywords: *Quantified Self*; *Quantified Workplace*; *Internet of Things*; *Service Oriented Architecture*; *Microservices Architecture*

Acknowledgements
Agradecimentos

Esta minha Tese de Mestrado marca o final de um caminho longo, demarcado por todas as boas vivências e, mais que tudo, todas as pessoas com que me cruzei neste meu percurso escolar.

Obrigado a Coimbra, à Universidade de Coimbra, e a tudo o que destas fazem parte, por me proporciarem um conjunto de aprendizagens que foram muito para além das salas de aula.

Obrigado ao professor Jorge Landeck, pelo conteúdo e discussões das suas aulas, e pela liberdade que me proporcionou enquanto orientador académico.

Obrigado ao Big, Carlos, Manuel, e Abílio. Obrigado ao Laurens, Paulo, e Ramesh. Obrigado ao Gonçalo, Johny, Rui, e Neto. Obrigado a todos eles por serem os enormes amigos que são, e por todas as vivências que experienciámos.

Obrigado ao Veloso, Peruzzi e David, por durante 6 anos partilharem comigo um telhado, e várias peripécias.

Obrigado ao Rafael por ter sido meu mentor, e toda confiança depositada, durante todos estes anos. Obrigado por haver sempre algo mais para aprender contigo.

Obrigado à jeKnowledge e todos os que desta fizeram parte. Convosco eu aprendi e cresci imenso. Convosco eu descobri o que queria ser quando fosse grande.

Obrigado a todos os que da Whitesmith fazem parte, por não me fazerem pensar duas vezes sobre com quem quero trabalhar e continuar a crescer.

Obrigado ao Marcelo por alegrar os meus dias e me fazer uma pessoa melhor.

Obrigado ao meu cunhado e sobrinho. Obrigado à minha irmã por me apoiar incondicionalmente em todas as minhas escolhas e tudo o aquilo que sou.

Obrigado aos meus pais por serem, de longe, o maior exemplo de trabalho, esforço, e dedicação que eu conheço.

Every man I meet is my superior in some way. In that, I learn of him.

– Emerson

Contents

List of Figures

List of Abbreviations

CFL Compact Fluorescent Lamps

CIE Commission Internationale de l'Éclairage International Commission on Illumination

HR Human Resources

HTTP Hypertext Transfer Protocol

IEEE Institute of Electrical and Electronics Engineers

IoT Internet of Things

KPIs Key Performance Indicators

LAN Local Area Network

MSA Microservices Architecture

NFC Near Field Communication

REST Representational State Transfer

SI International System

SME Small Medium Enterprises

SOA Service Oriented Architecture

SPL Sound Pressure Level

WAN Wide Area Network

YoY Year Over Year

Chapter 1

Introduction

This section presents the context of this Master Thesis. It explains its motivation, importance, and goals, and finishes with the defined scope of work.

1.1 Motivation

1.1.1 Context

If you can not measure it, you can not improve it.

– Lord Kelvin

Companies are always in the search of improving their performance. They do it by applying different theories to their various variables such as processes, tools, company culture, company structure, and workplace conditions.

Normally they approach this change as an initiative – i.e. a one-off, top-down movement outside the flow of daily work, which might or might not produce improvement [2].

But there are various challenges with this approach [2]:

- Companies don't have practical systems to monitor the health and effectiveness of the majority of their practices.
- When companies do have systems to monitor their practices, the results of such measurement tend to take an ample period of time to be published (Ex: quarterly auditorships). Thus conditioning the velocity and effectiveness of how companies can adapt and change their practices.
- To have a noticeable impact, changes applied on organisations need to be on a series of expensive high-impact projects, where a pre-defined model is imposed.

Organisations are lacking systems that can monitor their health, and effectiveness of their organisational structures and practices on an ongoing basis. Organisations are lacking the tools for improving their performance in short and frequent iterations, done by fine measurement of multimodal variables, which can then be crossed to provide insightful knowledge in constant and frequent feedback loops.

1.1.2 Convergence

The term Quantified Self was proposed in 2007 by Gary Wolf and Kevin Kelly, both editors from Wired Magazine [3]. They described Quantified Self as "a collaboration of users and tool makers who share an interest in self knowledge through self-tracking" [3].

The practice consists in the collection of data about own's self "in terms of inputs (Ex: food consumed, quality of surrounding air), states (Ex: mood, arousal, blood oxygen levels), and performance (mental and physical)" [4], thus resulting in the increase of self-awareness and knowledge about what can be improved on themselves [3].

Even though the term is relatively recent, there are records of people quantifying their daily life for many years in the past. For example, Benjamin Franklin famously tracked 13 personal virtues in a daily journal to push himself toward moral perfection [5]. He shared this insight in his autobiography: "I was surprised to find myself so much fuller of faults than I had imagined, but I had the satisfaction of seeing them diminish." [5].

But recently - with the technology advancements, price falls, and the rise of the Internet of Things [6], - data collection has not only become cheaper and more convenient, but is also allowing non-specialised individuals to quantify metrics that were impossible before.

The movement has evolved and, due to the appearance of specialised sensors and apps on smartphones and smartwatches directed to self-quantification [7] [8], it's now reaching mass-market [7].

What few have tried to achieve yet, is to explore the application of technologies and principles of the Quantified Self on organisations [1]. There are several challenges, as described on the previous section, that can be solved by creating systems that monitor organisations' health and practices on an ongoing basis - the Quantified Workplace.

This is what sets the vision for this work: we want to create the foundations for a framework of organisational health measures. A framework informed by theory and company goals, that can guide ongoing change in an agile, iterative way, and evaluate the success or failure of change actions against a desired future operating state.

We believe that organisations should have a management system for ongoing change actions that puts control in the hands of teams who want to improve their area of operations. But rather than offering a single fixed methodology, this approach enables people and organisations to assess and assimilate new theories and models that emerge in a way that allows comparison between them and, most important of all, consistent measurement of progress and results.

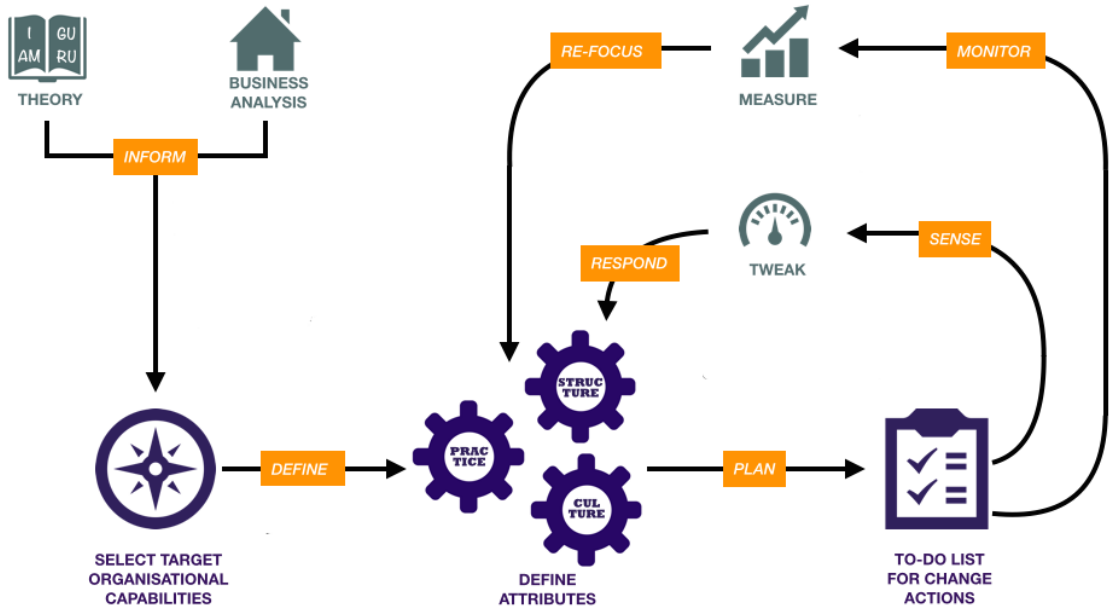

Figure 1.1: Diagram representing the Quantified Workplace workflow [2].

1.2 Project

In this section we translate the proposed vision into an actionable project. We also define the scope of this master thesis in concrete goals.

1.2.1 Goals

The general goals for this project are:

- Measure organisations health and performance on an ongoing basis, by collecting data from different range of sources
- Have a system that can easily integrate new data sources.
- Provide teams insightful, near real-time, metrics about their performance, which can be then translated into actionable items.

1.3 Thesis Scope

This Master Thesis focuses on the following scope:

- **A microservices based architecture for data collection on SMEs**, modular for easy attachment and decoupling of future services and devices.
- **Continuous data collection, processing, and storage** of SME's workplace metrics
- **Integration with Internet of Things devices**, with the goal of measuring physical environmental quantities.
- **A data visualisation platform** to make the collected data available for analysis.

1.3.1 Whitesmith

This Master Thesis was performed at Whitesmith. All produced work was done taking into account Whitesmith's tools, processes, and office environment.

Whitesmith is a software and hardware development SME, founded in 2012. The company builds web, mobile and IoT products for startups and media companies, particularly in the UK, Australia, and USA. Whitesmith also has products of their own such as:

- Qold - a cold-chain monitoring system
- Unplugg - an energy monitoring platform

Whitesmith has a distributed team - currently, its employees work from different places of Portugal and Brazil, with the majority working from its main office in Instituto Pedro Nunes, Coimbra, Portugal.

1.3.2 Core Premises

All the developed work was done taking the following assumptions about the collected and processed data at the company which it's applied:

- Data is used to focus on measuring quality, not quantity.
- Individual data is only used to gather insights about teams.
- All data is directly related with the work performed at the company.
- Every employee knows which data is being collected.

1.3.3 Data Collection and Processing

With the goal of measuring organisations' health and performance, we need to build specific individual services to collect and process data. The chosen variables to be quantified for this project took into account the following criteria:

- Variables that employees at Whitesmith have commented to be affecting their performance.
- Variables that science has proven to affect employees performance, and which research is documented in the State of the Art chapter of this Master Thesis.
- Ensure that the chosen variables cover the three categories: Office Environment, Happiness, and Productivity.

Some variables - as temperature - were already being measured by already built devices, and are going to be later integrated with the system, but outside the scope of this project. Other variables - as humidity - were not seen as impactful on the the company's performance, and due to the natural time constraints for developing this project, were left off for future work.

By taking these considerations, at the beginning of the project we set the goal of collecting and processing data about the following variables:

- Office environment
  - Office sound levels
  - Office luminosity levels
  - Employees at the office
  - Company activities
- Happiness
  - Organisation Happiness
- Productivity
  - Number of completed tasks
  - Number of defects
  - Code produced
  - Time a given step of task takes in the context of the process

The importance of collection of data about each variable is better described in the State of the Art chapter of this Master Thesis.

1.3.4 Support Components

To be able to cross and visualise data, other components were necessary to be built:

- Data Visualisation
  - Dashboard for data visualisation
- Company Info
  - Platform to store and manage employees information
  - Platform to store and manage projects information

Chapter 2

State of the Art

2.1 Quantified Self

The current Wikipedia entry for quantified self describes it as "a movement to incorporate technology into data acquisition on aspects of a person's daily life in terms of inputs (Ex: food consumed, quality of surrounding air), states (Ex: mood, arousal, blood oxygen levels), and performance (mental and physical)." [4].

This is not a new idea. Humans have since a long time been interested in measuring their activities with the goal of increasing the awareness of their actions, and then improve their performance by taking into account that information. For example, athletes and coaches have been making detailed notes on nutrition, training sessions, and other elements for years [9].

But new technologies are making it simpler than ever to gather and analyse personal data. The size and cost of sensors are much smaller than before - accelerometers, which measure changes in direction and speed, used to cost hundreds of dollars but are now cheap and small enough to be routinely included in smartphones [7]. This makes it much easier to take the quantitative methods used in science and business and apply them to the personal sphere. Now much of the data-gathering can be automated, and the record-keeping and analysis can be delegated to a host of simple apps and gadgets.

People are taking advantage of the current technology to track metrics that can go from health (blood analysis, weight, and frequency of migraines), up to more diverse type of metrics such as the number of hours they sleep, the food they eat, and their sports performance [10]. These people are an eclectic mix of early adopters from different areas such as, fitness, technology, personal-development, hacking, and even patients suffering from a wide variety of health problems [10] [11]. They share the belief that gathering and analysing data about their everyday activities can help them improve their lives [3].

Self quantification is becoming a common practice - according to a study made in 2013 by Pew Research, about 69% of US adults track at least one health

metric [12]. This goes hand-in-hand with the increasing number of Quantified-Self solutions in the market, and with increased funding raised for this sector - on a year-over-year basis, investments on Quantified Self startups has jumped 165% while YoY deal activity has accelerated over 40% [13].

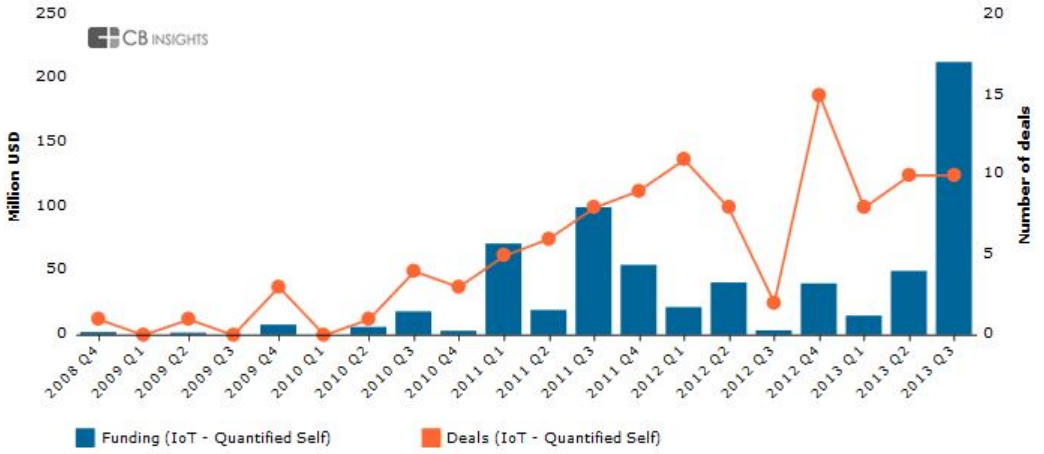

Figure 2.1: YoY evolution of raised funding and number of deals in the Quantified Self sector [13].

The movement captured the attention of entrepreneurs, who have iterated on top of homemade solutions, and built products for the mass markets: nowadays we have several personal activity and health trackers such as Fitbit[1], Jawbone[2], and more recently the Apple Watch[3]. These track various elements such as steps taken, distance traveled, calories burned, hours and quality of sleep, perspiration, skin temperature, and heart rate. There are also solutions being built for other verticals. An example is the automobilistic industry, where new devices give the user the ability to acknowledge new information about their car's usage and health[4] - this opens the possibility for, in the future, ask insurance companies for more adequate insurance plans by providing them the collected data. Other companies are developing platforms - as TicTrac[5] - that aggregate the data from people's various trackers in just one place. These meta-trackers, or aggregators, aim to find hidden insights in all the collected data, that ultimately might help users improve their lives.

[1]https://fitbit.com/
[2]https://jawbone.com/
[3]http://www.apple.com/watch/
[4]https://www.automatic.com/
[5]https://tictrac.com/

2.2 Quantified Workplace

The Quantified Workplace serves the idea of applying Quantified Self to companies. The origin of the term is uncertain, but it has been mentioned several times in several articles through the internet and newspapers. A Quantified Workplace is one that invests in technologies to quantify collective behaviour at an organisation, thus providing knowledge that can help improving processes and practices that enable to achieve organisational goals [14] [15]. Quantified Organisations support an environment of data-driven decision making [14] [15].

Neither data-informed management or workplace monitoring are new ideas: Management accounting emerged during the first half of the 20th Century [16]. It requires the design of the right balance of appropriate metrics to guide performance and secure organisational alignment in conjunction with the associated strategy. Workplace monitoring has been becoming normal, especially in blue-collar professions such as logistics and delivery, where telemetry technologies track employees' physical locations, and in the hospitality and retail sectors, where closed-circuit cameras are common [17].

But these methodologies and technologies have been applied to gather a narrow set of metrics. Specially for an era where more information than ever can be gathered from software and hardware, and sensor prices are on a all time low [6].

Even in companies that are more data driven, the current state of organisation quantification are either focused on the data provided by consumers - data-driven decision is made practically only in product development, marketing, and pricing -, or data provided by time tracking software installed on employees computers - which, in general, can't provide much more information than the time spent on each application or task [1].

There's now potential for developing tools and processes that can take advantage of the current state of the art to provide insightful real-time knowledge to teams and organisations - the Quantified Workplace.

Now it's possible to not only measure the inputs of employees (Ex: time spent coding), but also their outputs (Ex: number of code lines produced). We can measure and collect all kinds of physical quantities (Ex: temperature, light, sound, and humidity) that affect employees productivity, and their states (Ex: happiness, stress.).

There are few companies currently trying to achieve this: in the 2014-2015 survey made by Sierra-Cedar HR systems, only 12% of the organisations in their survey adopted some form of HR analytics. On the same study they concluded that Quantified Workplaces out-performed other organisations [1]. They saw higher levels of financial performance, as well as positive HR and Return on Equity (ROE) outcomes [1].

Google, Procter & Gamble, and Harrah's are some of the companies that are applying analytics approach in addressing human resources needs. They're

gathering all types of data, and always iterating their processes for optimisation of their employees performance and happiness [18].

Some companies - such as Qount Us[6] and Quantified Organisation[7] - are building products that can be installed and adapted to different organisations, to quantify their performance and happiness, each one with their particularities. These are on a very preliminary state, and not much has been achieved in this field yet.

With the current state of the art, there's potential to build tools that can support broad organisational objectives, by providing real-time insight to teams and entire organisations. Tools that can be part of a framework for designing a strategy and preparing the business to execute it with iterative and adaptive Agile methodologies, rather than a biennial or triennial major initiative, which results can only be known several months later. Tools that focus on empowering individual teams and managers, with decentralisation and openness to information that, at the end, can result in benefiting both individuals, teams, and organisations. Those using it are the Quantified Workplaces.

2.3 Workplace Metrics

At the workplace, there are several factors that have impact on employees productivity and happiness, and consequently on an organisation's health and performance. To build workplace data collection services, we must be aware of the impact that those have, and the adequate methodologies for measuring them.

Measuring an organisations' performance should take into account both inputs (Ex: time spent working) and outputs (Ex: number of features delivered). But since workplace variables tend to influence each other in many ways, some variables can be both inputs and outputs. Taking that into account, for this project we decided to divide metrics in three main categories:

- **Environment** - these take not only into account the physical quantities related with the workplace (Ex: noise, luminosity), but also activities that have impact on how people interact (Ex: company activities).

- **Productivity** - measurement of productivity must be suited for the organisation being quantified, and its work processes.

- **Happiness** - which is based on the collection of each employees happiness levels.

[6] https://qount.us
[7] http://www.quantifiedorganisation.com/

2.3.1 Noise

Noise is one of the most common complains among office workers - specially today, where more and more companies adopt the open-office design to increase teamwork, communication and productivity [19]. One downside of open-spaces is the augment of noise levels [20].

Some studies have found that the prolonged exposure to noise tends to increase the rate of illnesses (Ex: elevated blood pressure) and stress [20]. More specifically, in a study made on groups of software developers, it was found that software developers working at noisy environments tend be less happier with their work, and to work late during the week [21].

In part, that's because higher office noise levels tend to be a cause for higher number of interruptions. And, after resuming work from an interruption, it takes a programmer an average of 10-15 minutes to restart the previous task. [22]

Audible sound consists of pressure waves, and one of the ways to quantify the sound is to state the amount of pressure variation relative to atmospheric pressure caused by the sound. Noise is defined as any unwanted sound, that disturb people or make it difficult to hear wanted sounds [23]. This means that noise is a subjective measure that depends not only on the sound pressure, but also on its psychological perception.

The intensity of sound has Pascals (Pa) as the official SI unit, but decibels (dB) or Amplitude RMS - which are not SI units - are commonly used [23]. There's no SI unit for noise measurement, but the Phon, the Sone, and more commonly the dBA, are used for this purpose [23]. All of these noise measurement units take into account sound pressure and frequency - this is because human hearing sensitivity varies with frequency [23].

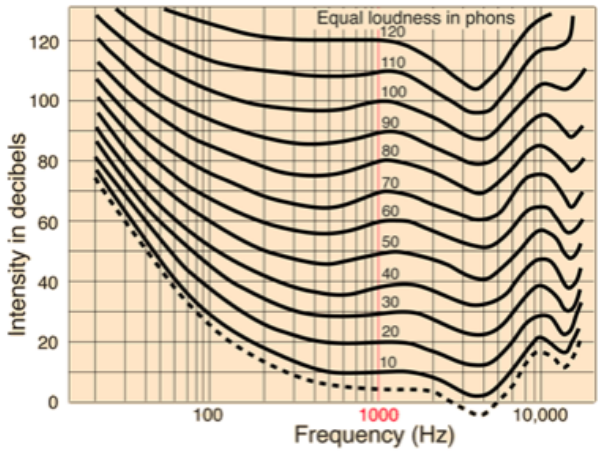

Figure 2.2: Equal loudness curves in Phons [24].

By definition, the number of phon of a sound is the dB SPL (Sound Pressure Level) of a sound at a frequency of 1 kHz that sounds just as loud - each equal loudness curve shown in 2.2, can be referenced to the decibel level at 1000 Hz. So, if a given sound is perceived to be as loud as a 60 dB sound at 1000 Hz, then it is said to have a loudness of 60 phons [23].

When making practical assessments of the sound level, as a part of a general survey of ambient sound levels, the type of measurement which is usually made is that of the sound levels in dBA, also known as A-weighting. A-weighting filter is commonly used to emphasise frequencies around 3-6 kHz where the human ear is most sensitive, while attenuating very high and very low frequencies to which the ear is insensitive. The aim is to ensure that measured loudness corresponds well with subjectively perceived loudness [23]. A-frequency-weighting is mandated by the international standard IEC 61672 to be fitted to all sound level meters [25].

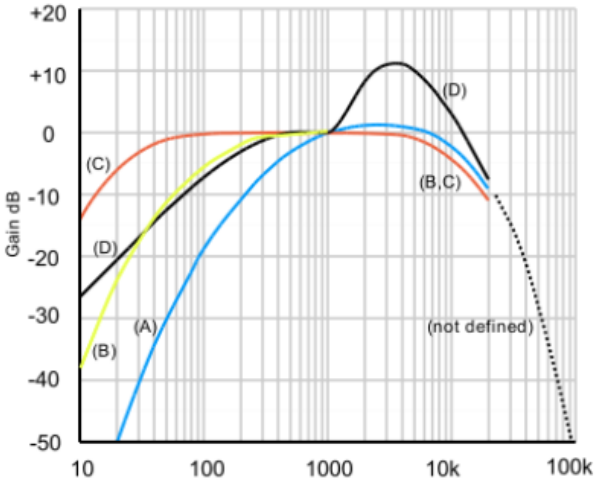

Figure 2.3: Weighting curves in dB. A-wheighting (blue), B (yellow), C (red), and D-wheighting (black) [26].

2.3.2 Illuminance

There are studies indicating that light illuminance, light switching frequency, and light exposure time in the workplace tend to have a big impact on the employees productivity and mood. These are also the variables that tend to distinguish Daylight from Artificial Light [27].

Daylight, as solar light, is relatively continuously distributed in light frequencies. Artificial Light sources such as a tungsten or halogen bulb, which use a heated metal to produce light, have also a relatively continuous spectrum across a limited range of frequencies [28]. But for example, the light peak from the

tungsten bulb is centered around longer wavelengths/lower frequencies than the Daylight distribution, and is yellower and of a lower colour temperature [28].

Artificial sources which excite phosphors with one wavelength of light to cause them to emit light at other wavelengths, produce light in a number of relatively sharp frequency peaks with gaps between with less or no light - the resultant "white" is a phantasm of the brain. These peaks of wavelength are arranged such that the eye and brain system combines them to produce "white" light. This method is used on fluorescent lights, CFL (compact fluorescent), and Phosphor LEDs. Similar results occur when a gas is excited electrically or thermally so it emits light with sharply defined frequencies or when multiple mono-coloured LEDs are used.In terms of light frequency, on contrary to Daylight, some types of Artificial Light such as incandescent light bulbs have a unperceived flickering (at twice the 50Hz or 60Hz of the AC frequency) [28].

Even though this phantasm and flickering cannot be perceived by the vision part of our brain, some studies defend that Artificial Light has different impact in the human body than day light has. On a study performed by the scientist Mirjam Muench, it was concluded that "Compared to the afternoon, people who had DL (Daylight) were significantly more alert at the beginning of the evening, and subjects who were exposed to AL (Artificial Light) were significantly sleepier at the end of the evening." [29]. This is related to the fact that our cortisol levels drop significantly under artificial or poor lighting conditions [29] - a steroid hormone produced by the human body that's responsible for regulating various human functions.

Also, according to various studies, artificial light with a strong blue component affects human circadian cycles and the human hormonal system, which can result in diseases ranging from sleep disorders to immune system disorders [30] [31]. These results also suggest that higher colour temperature light increases the central nervous system activity [30] [31].

Illuminance can be measured in the lux SI unit - which consists in the luminous flux per unit area. It is a measure of how much the incident light illuminates the surface, wavelength-weighted by the luminosity function to correlate with human brightness perception [32].

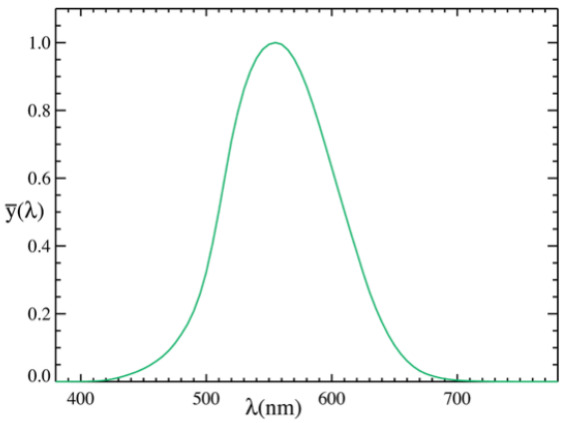

Figure 2.4: The 1931 CIE photopic luminosity function. The horizontal axis is wavelength in nm. The vertical axis is the standard luminosity function (which is dimensionless) [33].

The luminosity function, or luminous efficiency function, describes the average spectral sensitivity of human visual perception of brightness. It's based on subjective judgments of which of a pair of different coloured lights is brighter, to describe relative sensitivity to light of different wavelengths. It should not be considered perfectly accurate in every case, but it is a very good representation of visual sensitivity of the human eye and it is valuable as a baseline for experimental purposes [32].

2.3.3 Telecommute

Telecommuting - also known as remote work - is a significant workplace innovation that allows an increasing portion of employees to work from home or other location remote from the central workplace. This has become an increasingly common practice, with many employers allowing their employees to work remotely [34] [35].

Although telecommuting can blur the boundary between work and non-work activities, it has benefits for both the telecommuter and their employer: improved productivity, job satisfaction, savings of office space, increased flexibility, improved employee morale, and employee acquisition and retention [34] [35].

In part this is due to making it possible for employees to avoid potentially stressful commutes, to the ability to choose to work at places that are quieter and more interruption-free than their offices, and overall flexibility of choice [34] [35].

But it's not clear if telecommute has the same positive effects at every company - how affects organisations' productivity, happiness, code quality, noise,

and other variables of the workplace. This can be achieved by registering the employees telecommuting times, and crossing this information with other data.

Nowadays, there are different forms of automating the functionality of the old punch card - a mechanical (or electronic) timepiece used to assist tracking of the hours worked by an employee. Some examples of automatic and frictionless methods to register employees presence at the office are: NFC tags, computer vision systems equipped with facial recognition, or by querying LAN/WLAN network for nearby devices.

NFC is is the set of protocols that enables smartphones and other devices to establish radio communication with each other by touching the devices together or bringing them into proximity to a distance of typically 10 cm or less [36]. The NFC tag system requires the implementation of a NFC reader at the office, and give each employee a NFC tag. The hardware is relatively inexpensive, but the adoption of such system requires the change of behaviour on every employee, by asking them to pass their NFC tag every time they enter and exit the office.

Facial recognition through computer vision systems, at the level of identifying each individual is currently hard to build from ground up [37], making it necessary acquire the technology from third parties. This makes this option the least feasible.

The later option - query the network for nearby devices - is, in a world of personal smartphones and personal computers, one of the most practical methods of registering employees presence at the office. This can be done using MAC Addresses as a key for employee identification.

MAC Addresses (Media Access Control Address) are unique identifiers assigned to network interfaces for communications on the physical network segment, used as a network address for most IEEE 802 network technologies, including Ethernet and WiFi [38].

2.3.4 Software Development Performance

Software development performance can be grouped into two main categories: subjective performance and objective performance. Subjective performance assessment can be defined as an evaluation method that reflects the opinion of the people involved, and it's usually achieved through questionnaires to the involved individuals. In contrast, objective performance includes more quantifiable measures such as cost, quality, productivity, predictability, and responsiveness, and can usually be automated to a certain extent. While subjective performance assessment has the advantage of easy data collection, it has difficulties with standardisation since the project evaluation is dependent on the person's judgment. [39] [40]

Examples of subjective KPIs [39] [40]:

- Process performance - a performance metric for the software development process that can be described by the (1) learning that occurs during the course of the project, (2) the degree to which management controls the project, and (3) the quality of the interactions between the team and users during the development process.

- Product performance - a metric that captures the performance of the finished product and can be described by the (1) technical performance of the software, (2) the degree to which the software conforms to user needs, and (3) the degree to which the software is flexible in supporting new products and changing user needs.

Each software organisation have their own engineering and management practices, with different methodologies, tools, and levels of complexity. The measurement of software development performance needs to be adapted accordingly to each organisation's practices.

Examples of objective KPIs [41] [42]:

- Responsiveness - Based on the time a given step of task takes in the context of the process. Ex: Time a bug takes to be fixed.

- Throughput - It can consist in the number of completed user stories, number of added lines of code, or number of Git commits, in a given time period.

- Predictability - Based on throughput variability. (The standard deviation of throughput.)

- Costs - The money costs associated with the project development.

- Maturity - Based on the number of defects. The more defects the software has, the less mature it is.

Because software development performance measurement have different implications to different organisations, is often recommended using both subjective and objective performance measures [39] [40]. Some reasons why using only objective measurements may not work are [43] [44]:

- Some programmers can focus on hard problems and have little throughput, but deliver high value.

- Some programmers can produce much bug tracker traffic by being excessively fine-grained, and bring inefficiency to the team.

- Some programmers may not produce much code, but spend a good portion of their time teaching and helping other team members.

- Some programmers may deliver numerous checklist points with tasks, but aggressively externalise much of the work.
- Some programmers may work fewer hours than their counterparts, but deliver high throughput and value.
- Objective measurements may bring arguments about whether bugs are caused by development or by poor quality analysis.
- Objective measurements may incentive developers to game the system (Ex: producing less code equals to a lower number of bugs).
- Objective measurements may incentive developers to avoid collaboration, since that's going to affect their personal results.

These issues can be clogged by doing both objective and subjective measurements, and by focusing the KPIs on the team instead of on the individual. Subjective measures help, for example, to understand the quality of the interactions inside the team, the impact of each individual on the project, and how various of the processes can be improved. By focusing on the teams instead of on the individual, we discourage gaming the system on an individual level, and incentive collaboration [43] [44].

2.3.5 Happiness

Until recently, the concept of the happy-productive worker has often been relegated as an unsubstantiated claim made only by practitioners. In the recent years, psychologists have addressed this topic, by realising a different set of experiments which have showed evidence of the correlation of employees happiness with their performance, and thus, companies' results. This is leading organisations to recognise that the greatest competitive advantage in the modern economy is a positive and engaged workforce.

Isen and Reeve (2005) show that positive events induces subjects to change their allocation of time towards more interestingly challenging tasks, and despite this, the subjects retain similar levels of performance in the less interesting tasks [45]. This suggests that individuals become better able to undertake repetitive tasks as they become happier.

More recently, on a study made by Andrew J. Oswald, Eugenio Proto, and Daniel Sgroi from the University of Warwick [46], it was performed three different styles of experiment. Here, the randomly selected individuals that were made happier, have approximately 12% greater productivity. They also found that those who had a solid reason to be unhappy, such as a recent bereavement, were less productive than their happier counterparts. They concluded that lower happiness is systematically associated with lower productivity.

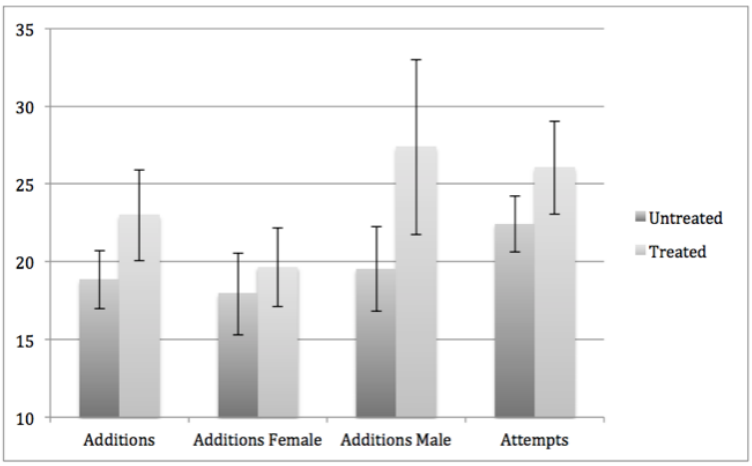

Figure 2.5: Experiment II results from the research of Andrew J. Oswald, Eugenio Proto, and Daniel Sgroi, University of Warwick. Those exposed to the randomised happiness treatment in the laboratory have higher productivity. Uses the timed mathematical-additions task of Niederle and Vesterlund 2007. Here the happiness treatment is a comedy movie clip in the laboratory. (95% confidence intervals) [46]

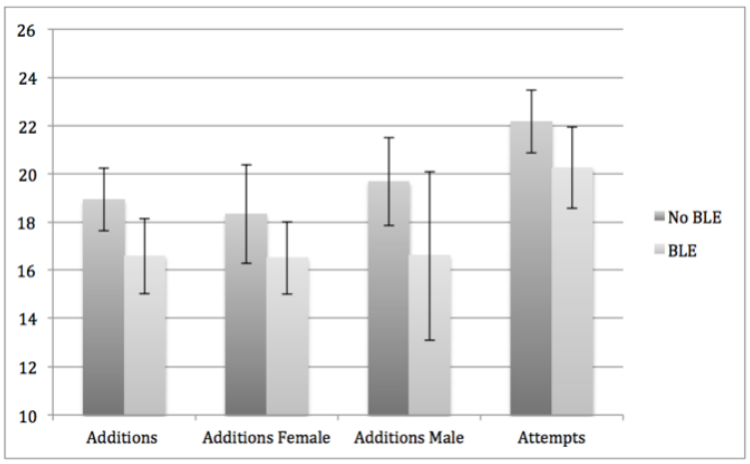

Figure 2.6: Experiment IV results from the research of Andrew J. Oswald, Eugenio Proto, and Daniel Sgroi, University of Warwick. Individuals with a recent Bad Life Event (BLE) have lower productivity. Uses the timed mathematical-additions task of Niederle and Vesterlund 2007. Here a bad life event is bereavement or family illness. (95% confidence intervals) [46]

There are a variety of methodologies to measure people's happiness. Most of them use multi-item happiness measures. Some tap into the cognitive component of happiness (i.e., judgments of life satisfaction) and others assess the affective

component (i.e., the experience of frequent positive emotions and relatively infrequent negative emotions). For example, the popular Satisfaction With Life Scale survey gets into the cognitive component by asking respondents five questions about their feelings regarding their lives (Ex: "In most ways my life is close to my ideal."). While the Affect Balance Scale survey gets into the affective component by inviting people to report how frequently they have experienced various positive and negative emotions over the last 30 days. [47]

The simplest method is a single item that has been posed to hundreds of thousands of representatively sampled people in many countries: "Taken all together, how would you say things are these days would you say you are very happy, pretty happy, or not too happy?" [47].

These methodologies can be applied under different sampling types - the three common types of sampling are event contingent, interval contingent, and signal contingent. The event contingent is where the participant makes reports during a certain type of event - for example, after committing code to git, or opening the computer for the first time in the morning. Interval contingent is when the participant makes recordings at the end of large intervals, like at the end of the day. Signal contingent sampling is where participants get random prompts to record data - this helps avoiding memory biases. [48]

The higher the frequency of the questionnaire, the more we need to take into account its length and difficulty of answer input - questionnaires with a higher response time have lower response rate. This means that we not only need to choose the few questions that can provide us with higher return of knowledge, but also make sure the interaction with the questionnaire is the shortest and more fluid possible. [48]

2.3.6 Microservices Architecture

A software application can adopt different architecture patterns. Service Oriented Architecture and Microservices Architecture have been discussed and adopted by several companies in the last decades, emerging as a reaction against the traditional monolithic architecture. [49]

Monolithic Architecture

Monolithic architecture - a traditional approach to enterprise software - are single applications that package all the their server-side components into a single unit. Monolithic applications are single-programs with many responsibilities. [49] [50]

If we pick the example of an online store, the monolith app will have a component that manages the product catalog, a component that manages the customers' accounts, and other that manages the orders.

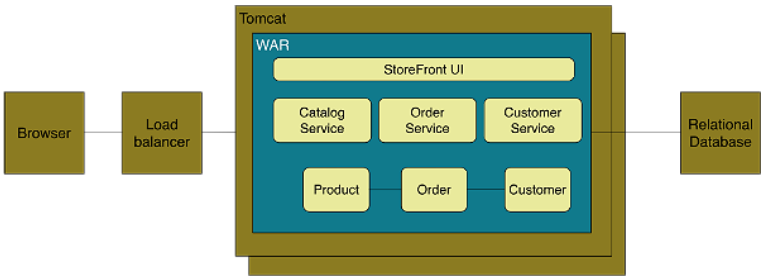

Figure 2.7: Example of a Monolithic application [50].

Monolithitic Architecture has advantages and disadvantages, which are described below. These can be better understood when put in perspective with SOA and MSA.

Advantages of the Monolithic Architecture are [49] [51] [52]:

- Ease of development, due to support from various IDEs.
- Ease of testing, due to dependency on only one application.
- Ease of deployment, due to dependency on only one application.

Disadvantages of the Monolithic Architecture are [49] [51] [52]:

- Harder to maintain, as all components are more coupled to each other.
- High cost of deployments, in the case of complex applications - a change in a single component requires the test, build, and deployment of the entire application.
- Difficulty in the adoption of new technologies - the trial and adoption of new technologies often requires rewriting the entire application.

Microservices Architecture

MSA emerged from SOA. [53] SOA is an architectural pattern in computer software design, where logical and business functions are partitioned into self-contained units of software that may list several discrete services/operations. These application components provide services to other components via a communications protocol, typically over a network. The principles of service-orientation design stress the separation in units of software partitioned into operational capabilities, each designed to solve an individual concern - these units qualify as services. [53] [54]

Although no industry standards exist to define what composes an SOA, some principles are widely accepted as the core of what SOA represents. For example, Microsoft's Don Box "four tenets of service-orientation" are [55]:

- Boundaries are explicit
- Services are autonomous
- Services share schema and contract, not class
- Service compatibility is based on policy

SOA surged with the promise to include increased return on investment, organisational agility and interoperability as well as a better alignment between business and IT. It builds heavily on earlier design paradigms and enhances them with standardisation, loose coupling, and business involvement [55].

But due to the inconsistent application of the SOA term by IT product vendors, the SOA concept gained different meanings for different people, and thus translating into different practices when applying this architecture pattern [56]. This common manifestation of SOA has led some advocates to create MSA - a term that more crisply defines this architectural style. For some advocates, MSA is very distinct from SOA, making them reject the SOA label entirely. Others consider microservices to be a specialization of SOA, or even the correct implementation of SOA. [49] Well known sites such as eBay, Amazon.com, Groupon, and Netflix have been evolving from a monolithic architecture to MSA [50].

MSA can be defined as an approach for developing a single application as a suite of small services, each running in its own process and communicating with lightweight mechanisms, often an HTTP resource API. These services are built around business capabilities and should be independently deployable by fully automated deployment machinery. There is a bare minimum of centralized management of these services, which may be written in different programming languages use different data storage technologies. [50]

As we see, SOA and MSA are very similar in various ways. But MSA contrasts with SOA on parts such as: the used communication mechanisms; the practice of automatic deploys, and; the use of different, non-proprietary, technologies for each service. MSA also tends to differentiate from SOA in the size of each service - MSA's services are typically more partitioned, and thus smaller, than SOA's services. [57] [53] [58]

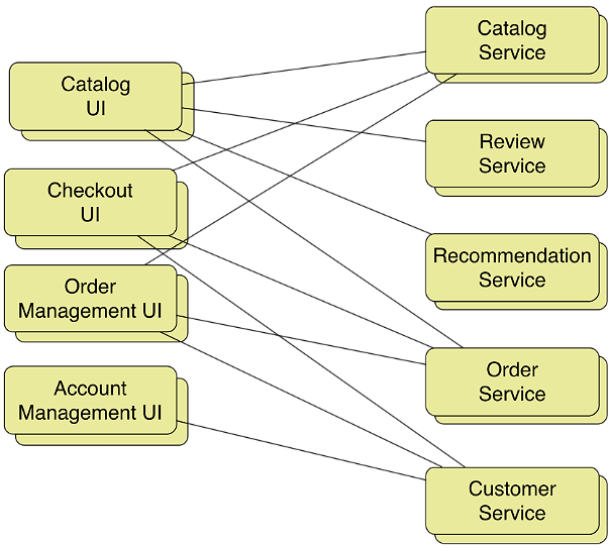

Figure 2.8: Example of a MSA-based application [50].

On the following paragraphs, some of the represented facts can be applied to both SOA and MSA, but for a question of abstraction, we will refer only to MSA.

Figure 2.8 represents the architecture of the example application, in the case we apply MSA. The represented application includes exactly the same logical services as the original monolith.

Communication

While in a monolithic application components call one another via regular method calls, in a MSA different services run in different processes. Consequently, services must use an inter-process communication (IPC) to communicate. Synchronous or asynchronous communication can be used. Responses are typically returned in HTML/JSON/XML format [52].

Synchronous communication can be done through a HTTP-based mechanism, usually REST. This is easy to implement, and it's firewall friendly so it works across the Internet. The drawbacks of HTTP are that it doesn't support other patterns of communication such as publish-subscribe; both the client and the server must be simultaneously available, which is not always the case since distributed systems are prone to partial failures; and, the HTTP client needs to know the host and the port of the server, which is not trivial in cloud deployment that uses auto-scaling. The later can be solved by using a service discovery mechanism, such as Apache ZooKeeper or Netflix Eureka. [50]

Asynchronous message-based mechanism can be achieved for example through a AMQP-based message broker - an open standard application layer protocol for message-oriented middleware. The message broker buffers the messages until the consumer is able to process them, making the producers totally unaware of the consumers - they can just send it into a message bus and possibly, in one moment

in time, a service can start listening. This makes a service discovery mechanism not necessary. One of the drawbacks is the fact that we are adding another component to the system and thus adding complexity. [50]

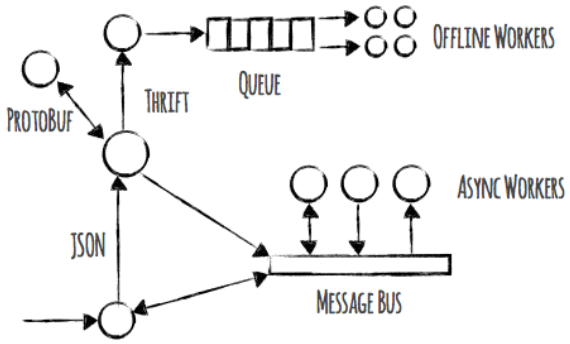

Figure 2.9: Example of communications between microservices [52].

Another issue in MSA communication, is the fact that often - for example for displaying a web page - a large number of calls to the different services is required. This is inefficient and results in a poor user experience. When this happens, an API gateway can be introduced. The API gateway will be responsible to handle the communication between the client and the services, by proxying or aggregating requests, and by using a cache mechanism. This way, the communication efficiency is optimized, and the details of the microservices are encapsulated without impacting the clients - for example, two microservices might be merged, and other microservice might be partitioned into two or more services, with only the API gateway needing to be updated to reflect these changes. [50]

Monit and Metrics

In MSA specifically, proper monitoring is critical, as parts of the system can fail without clear and immediate evidence. We have to ensure that all processes stay up, don't run out of disk space, don't deadlock, and stay performant. There are several tools on the market, such as Sentry[8] and Logentries[9], that can provide that information. [52]

Advantages of the Microservice Architecture are [50] [52] [49] [59] [51]:

- Maintainability - Since every code base maintains a reduced amount of logic, it is easier to understand the code base and correct the flaws in the existing functionality. Loose coupling also contributes to this aspect since a) modifications to their code base have low or no impact to the remaining

[8]https://getsentry.com/
[9]https://logentries.com/

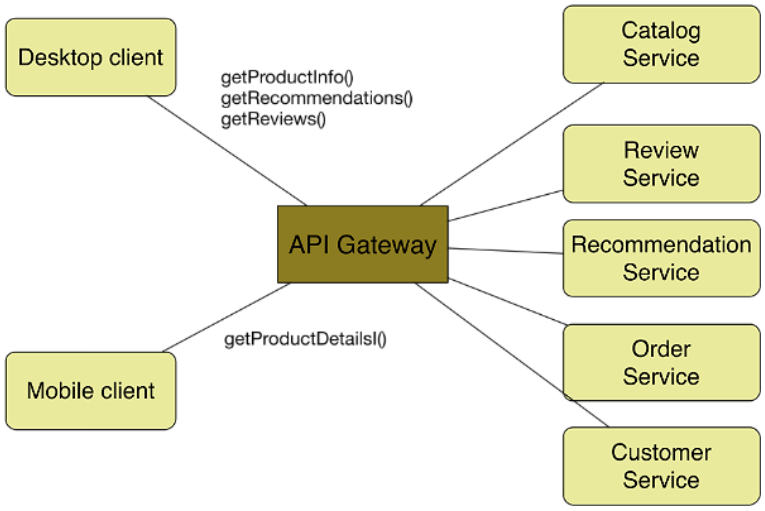

Figure 2.10: By using the API Gateway, fine-grained requests from a desktop client are simply proxied to the corresponding service, whereas each coarse-grained request from a mobile client is handled by aggregating the results of calling multiple services [50].

services; and b) development can be organized around multiple small teams, with each team responsible for a single service or a collection of related services that can be developed, deployed and scaled independently of all of the other teams.

- Extensibility - Which is related to the Maintainability of the system. Promoting loose coupling between services and high cohesion per service allows its extension to be performed without impacting the existing system functionality, either by creating a new service (which benefits from loose coupling) or by extending the functionality of an already existing service (which benefits from high cohesion).

- Scalability - Each service can be deployed independently of other services. Horizontal scalability becomes possible on a per-service basis, meaning that it is possible to run more instances of the services that need more resources, while running fewer instances of the services that have a reduced resource consumption. Vertical scalability comes from delegating the state data management to an external service, reducing the amount of consumed resources per interaction.

- Ease of deployment of new versions of services - Each service can be deployed independently of other services on a frequent basis.

- Fault Isolation - A problem in one service tends to affect only that service and the actions related with it. Other services will continue to handle non-related requests normally. In comparison, the monolithic architecture has a single point of failure, where an error can bring the entire application down.

- Ease of adoption of new technologies - When developing a new service, it's possible to choose the most adequate technology to best deliver the business case. Plus, due to the small code base, it tends to be practical to rewrite the service using the new technology.

Disadvantages of the Microservice Architecture are [50] [52] [49] [59] [51]:

- Distributed system complexity - While the code bases of each service are easier to understand and maintain, how the whole application is distributed and communicates gains a different level of complexity. Once we have a distributed system, we have to consider a whole host of concerns that we didn't before - network latency, fault tolerance, message serialisation, unreliable networks, asynchronicity, versioning, varying loads within our application tiers etc. To absorb this impact, a high-level of automation of application deployments and cloud management is necessary.

- Operational overhead - Where a monolithic application might have been deployed to a small application server cluster, we now have tens of separate services to build, test, deploy and run, potentially in a polyglot set of languages and environments.

- Increased memory consumption - The microservices architecture replaces N monolithic application instances with NxM services instances. If each service runs in its own VM (or equivalent), which is usually necessary to isolate the instances, then there is the overhead of M times as many VM runtimes.

Application Scalability

Applications can be scaled using different practices. In the book The Art of Scalability, a three dimension scalability model is used to represent these - the scale cube [50].

The X-axis represents the approach of scaling an application by running multiple identical copies behind a load balancer.

The Z-axis, similarly to the X-axis, represents the scaling with server that replicate the code, with the major difference that each server is responsible for only a subset of the data (Ex: payment, orders, products). A component in the system is responsible to route each request to the appropriate server.

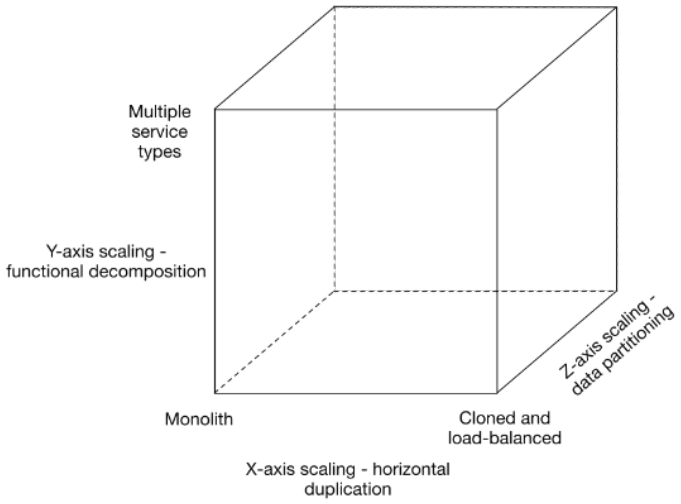

Figure 2.11: Three dimension scale cube [50].

The Y-axis represents scaling in terms of functional decomposition. Here, the application is split into various services, each one responsible for a different functionality (as products, orders, payments).

While Monolithic applications are usually scaled through the X-axis, and in some case through the Z-axis, a MSA application scales through the Y-axis.

Adopting a microservice architecture should not be undertaken lightly - as we seen, there are several advantages and disadvantages. When developing the first version of an application, often we do not have the problems that the MSA approach solves. Some practitioners defend that applications should first be developed using a Monolithic approach, being MSA adopted at a later stage - it gets the development faster, and it lowers the costs of making boundaries' errors or changes. Others defend the adoption of the MSA approach at the beginning of the application's life - not only because the later our application adopts MSA the more difficult the process is going to be, but also because there are other advantages of MSA, such as the ability to use different technologies that best suit the responsibility of the different services. [50] [60]

However at which stage MSA is adopted, for applications that need to scale, it is usually the right choice [50]. It's usually the right choice if, on complex applications, we want that new team members to quickly become productive, if the application must be easy to understand and modify, if we want to practice continuous deployment, and if we want to take advantage of the emerging technologies (frameworks, programming languages, etc) [51]. In sum, MSA offers a clearer and better defined approach on setting up an architecture built around services, with the promise of providing better agility, return of investment, interoperability, and better alignment between business and IT, to the organisation

which the application fits into.

2.3.7 Internet of Things

The Internet of Things (IoT) is the highly distributed network of physical objects or "things" embedded with electronics, software, sensors, and connectivity to enable objects to exchange data with human beings or connected devices without requiring human-to-human or human-to-computer interaction. IoT has evolved from the convergence of wireless technologies, micro-electromechanical systems (MEMS) and the Internet. The connection of physical things to the Internet makes it possible to access remote sensor data, giving the ability to measure, infer, and understand environmental indicators, and to control the physical world from a distance. The mash-up of captured data with data retrieved from other sources, gives rise to new synergistic services that go beyond the services that can be provided by an isolated embedded system. [61]

IoT is opening tremendous opportunities for a large number of novel applications that promise to improve the quality of our lives. Applications of IoT range from various fields and systems such as [62]:

- Internet connected cars;
- Wearable devices including health and fitness monitoring devices, watches, and even human implanted devices;
- Smart meters;
- Home automation systems and lighting controls;
- Smartphones which are increasing being used to measure the world around them; and
- Wireless sensor networks that measuring weather, flood defenses, tides and more.

The the number and variety of devices that are collecting data is inscreasing rapidly. A study by Cisco estimates that in 2010 the number of Internet-connected devices exceeded the human population, and that by 2020 there will be 50 billion Internet-connected devices [62].

Attributes

IoT systems must take into account different attributes. Many of these emerge from the limited form-factors and power available to IoT devices, others come from the way in which IoT devices are manufactured and used. They can be divided in the following categories [61] [63]:

- Cost-effectiveness - Determined by the affordability of the system.
- Efficiency - Described in terms of power and data management of the different devices connected to the system.
- Quality of Service (QoS) - A performance management technique for the prioritization of different data traffic from devices - heterogeneous smart devices with limited buffer capacity need effective buffer management scheme and differentiated service priorities to provide preferential treatment to delay sensitive traffic.
- Connectivity and communications - Defined by how each device communicates and is connected to other devices and network - from communication protocols to the system architecture.
- Scalability - The ability to scale from a small deployment to a very large number of devices, and the ability to scale the server-side out on small cheap servers.
- Manageability - While many IoT devices are not actively managed, this is not necessarily ideal. In many cases it should be possible to perform remote actions such as disconnecting devices, update the software, and enabling or disabling certain hardware capabilities.
- Security and Privacy - Defined by how immune the architecture is to outside attacks. It integrates various issues such as authentication, encryption, etc.

Architecture and Communications

The Internet of Things domain encompasses an extremely wide range of technologies, from stateless to stateful, from extremely constrained to unconstrained, from hard real time to soft real time. The communication among these devices as well as with related services, it is frequently done in a wireless, autonomic and ad-hoc manner, and is expected to happen anytime, anywhere. In addition the services become much more fluid, decentralized and complex. [64] Architecture in this context is defined as a framework for the specification of a network's physical components and their functional organization and configuration, its operational principles and procedures, as well as data formats used in its operation. [64]

The communications between devices and the Internet or to a gateway include many different models [62]:

- Direct Ethernet or Wi-Fi connectivity using TCP or UDP
- Bluetooth Low Energy
- Near Field Communication (NFC)

- Zigbee or other mesh radio networks
- UART or serial lines
- SPI or I2C wired buses

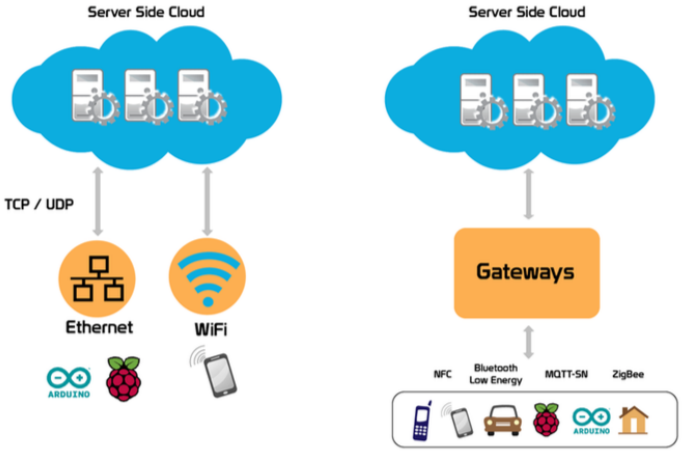

Figure 2.12: IoT communication options.

There are multiple potential protocols for communication between the devices and the cloud. The three most well known potential protocols are [62]:

- HTTP/HTTPS (and RESTful approaches on those)
- MQTT
- Constrained Application Protocol (CoAP)

HTTP is well known, with many libraries that supporting it. Because it is a simple text-based protocol, many small devices such as 8-bit controllers can partially support the protocol. The larger 32-bit based devices can utilize full HTTP client libraries that properly implement the whole protocol. Communications based on HTTP are inefficient and costly - both in terms of network traffic as well as power requirements. [62]

MQTT is a publish-subscribe messaging system based on a broker model. It was invented in 1999 to solve issues in embedded systems and SCADA. The protocol has a very small overhead (as little as 2 bytes per message), and was designed to support lossy and intermittently connected networks. MQTT was designed to flow over TCP.One important aspect with IoT devices is not just for the device to send data to the cloud/server, but also the reverse. This is one of the benefits of the MQTT specification: because it is a brokered model, clients

connect an outbound connection to the broker, whether or not the device is acting as a publisher or subscriber. This usually avoids firewall problems, because this approach works even behind firewalls or via NAT. [62]

The Constrained Application Protocol (CoAP) is a protocol from the IETF that is designed to provide a RESTful application protocol modeled on HTTP semantics, but with a much smaller footprint and a binary rather than text-based approach. CoAP is a more traditional client-server approach rather than a brokered approach. CoAP is designed to be used over UDP. CoAP has a narrow adoption, and has a less simpler connectivity over firewalls and NAT networks, when compared with MQTT and HTTP. [62]

Internet of Things technology appear complex for variety of reasons. There is legitimate heterogeneity in the used networking technology and applications. This variation is necessary and useful, as for instance different applications and environments benefit from varying networking technology. The range and other characteristics of cellular, wireless local area networking, and RFID are very different from each other, for instance. There are literally thousands of different applications, and it is natural that they have differing requirements on what parties need to communicate with each other, what kind of security solutions are appropriate, and other aspects. [65]

Thus, a single reference architecture cannot be used as for all possible concrete implementations. While a reference model can probably be identified, it is likely that several reference architectures will co-exist in the Internet of Things [64]. Service Oriented Architectures are a promising approach for building systems given these boundary conditions, because they provide a high level of abstraction that allows to safely hide hardware specific details from the developer and to integrate components from different vendors [62]. The IoT architecture, like the Internet, will grow in evolutionary fashion from a variety of separate contributions, rather than from a grand plan [64].

Chapter 3

The Platform

In this chapter the platform components and architecture are presented. It should be recalled that the goal for this project is to build a platform for the quantification of SMEs' workplaces on an ongoing basis, through a Microservices Architecture. The platform needs to be modular, so that it provides ease of integration with other services and devices, and thus facilitate the quantification of other variables. The chapter starts by presenting the overall system structure, following with a description of each component.

3.1 Architecture Overview

The platform is composed of various components, which are divided in 5 main categories: Data sources, Data Processing Microservices, Company Info Microservices, Data Storage, and Dashboard.

- Data Sources are third-party services (Ex: project management tools), or sensors from IoT devices installed at the workplace (Ex: luminosity sensors).
- Data Processing Microservices are the microservices built with the purpose of processing the received raw data from each data source, and storing the processed data on the adequate databases. There are Data Processing Services which are installed in servers located in the cloud, and others installed in devices located in the office.
- Company Info Microservices were built with the goal of centralising the meaningful data relative to projects and employees. These are frequently queried by other services to adequately identify to which project or user belongs the correspondent packet of raw data. They both have a user interface for easy manual data input.
- Data Storage, which consist in two databases, each one with characteristics that make them suitable for a specific purpose.

- Data Visualisation, which consists in a Dashboard or visualisation of meaningful insights that are a result from processed data.

The Data Processing Microservices, Company Info Microservices, Data Visualisation, and part of the Data Sources (sensors), were components built during this project. The Data Storage, and the other part of the Data Sources (Third-party Services) already existed before this project.

An overview of the architecture and how each component connects with each other can be seen in figure 3.1.

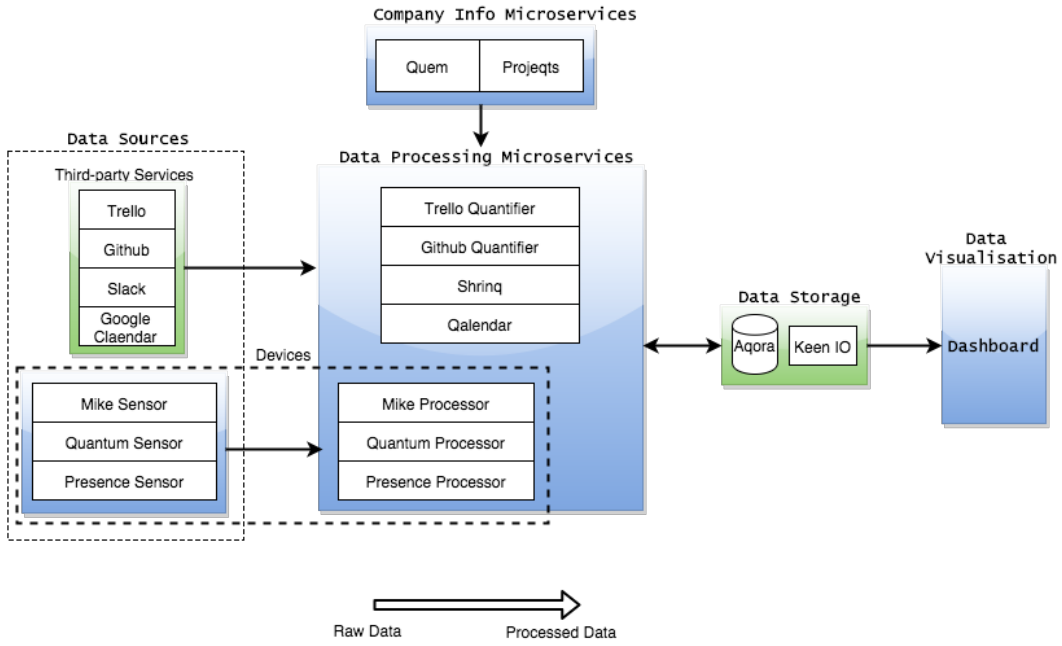

Figure 3.1: Architecture overview. The blue boxes represent the services built for this project, and the green boxes represent third party services. The arrows represent the data flow.

3.2 Components

3.2.1 Data Storage

Aqora

Aqora is a time-series event database, which permits the storage of float numeric values relative to a stream. Each stream corresponds to a variable that tends to oscillate with time.

This platform is specially built for the storage of high-frequency time-series events (i.e. every few seconds or minutes), such as temperature or electricity consumption.

This platform was built by Whitesmith, previously to this project.

Keen IO

Keen IO[1] is a time-series event platform, which permits the storage and analysis of various types of data. The event payload can include data in the form of float, integer, string, boolean, or array.

Keen IO has a powerful built in analysis and visualisation tools, which permits the fast and easy retrieval of some analytics - count, average, sum, etc - about the collected data without the need of processing from our side. For example, Keen IO permits retrieval of the average happiness levels per day, or number of responses per month, through its query API.

Keen IO is best suited for low frequency events (i.e. minutely, hourly, and up).

3.2.2 Company Info Services

Projeqts

A platform built with the purpose of centralising the information relative to each company project - Whitesmith is a software development company, meaning that each project corresponds to a product or service in this area. Every project has its own Projeqt ID, which serves as the identification key on every company service.

Every time a new project is started at the company, a new project entry is created on Projeqts using a manual form. Relevant data such as Name, Description, Github repository ID, and Active or Inactive status, is introduced in each project entry. This data is then used by other services.

Quem

A platform built with the purpose of centralising the information relative to each employee. Every project has its own Quem ID, which serves as the identification key on every company service.

Every time a new employee starts working at the company, a new employee entry is created on Quem using a manual form. Relevant data such as Name, Description, Github repository ID and Active or Inactive status, is introduced in each project entry. This data is then used by other services.

[1] http://keen.io

3.2.3 Data Sources

Trello

Trello[2] is a software administrative tool for project management.

Each project has its own Trello board. The building blocks of Trello boards are lists and cards - each card corresponds to a task, and each list corresponds to a state of a card.

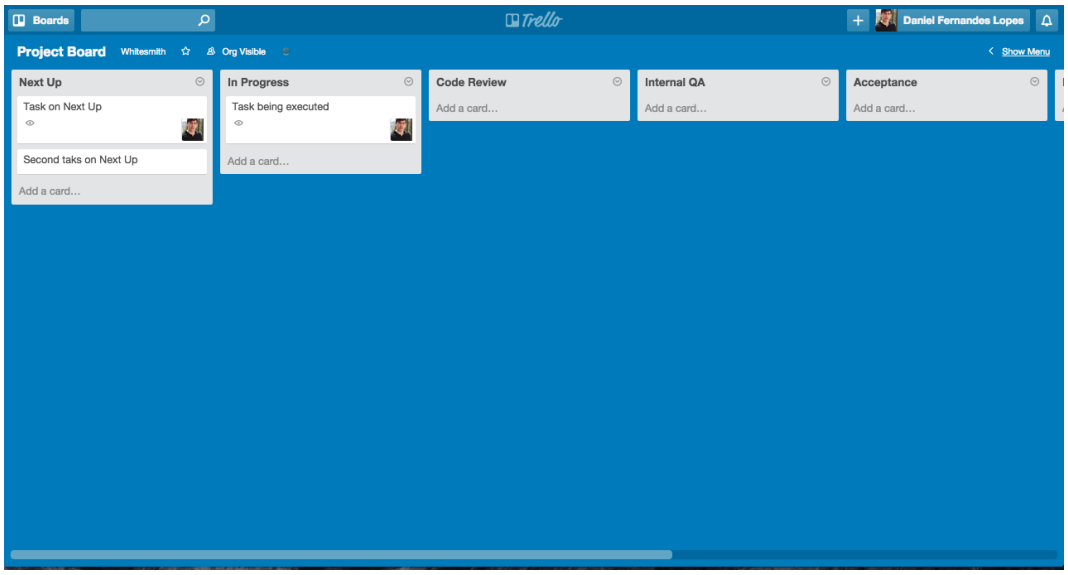

Figure 3.2: Example of a Trello Board with Trello Lists and Cards.

Cards are moved through Lists according to their state - through a flow that tends to be from left to right. For example, cards in the "In Progress" list are currently being tackled by the team. Cards in the "Live" list are finished.

Each card has other information attached besides the name, such as description, comments, labels, and people assigned to the card.

Slack

Slack[3] is the software communications platform used at Whitesmith.

Each employee has its own Slack user account, and each project has its own Slack group where discussions are performed. Slack also permits users or bots to communicate directly to other users.

[2] http://trello.com
[3] http://slack.com

Github

Github[4] is one of the most used Git repository hosting services, and the one used at Whitesmith. Git is a distributed revision control system used in software programming.

Each project has one or various Github repositories, to which employees can contribute.

Google Calendar

Google Calendar[5] is an online calendar that can be shared with various people. Whitesmith has a Google Calendar directed to company events.

Mike Sensor

Mike measures the sound levels in the office. Mike's sensor is a microphone, built-in a Logitech C270 webcam, that's connected to a Raspberry Pi. It's part of a device installed at the office.

Quantum Sensor

Quantum measures the illuminance at the office. Quantum's sensor is Adafruit TSL2561 photodiode digital sensor, which is connected to a RaspberryPi. It's part of a device installed at the office.

Presence Sensor

Presence quantifies employees attendance at the office (non-telecommuting employees). Is built with a RaspberryPi that queries the local network for currently connected MAC addresses. It's part of a device installed at the office.

3.2.4 Data Processing Services

Mike Processor

Mike processor is responsible to capture the microphone readings, calculate the sound RMS Amplitude, and send those values to Aqora.

Mike processor is also responsible to process stored data on Aqora, returning metrics that are then stored on Keen IO (Ex: hourly RMS Amplitude average).

w

[4]http://github.com
[5]http://google.com/calendar

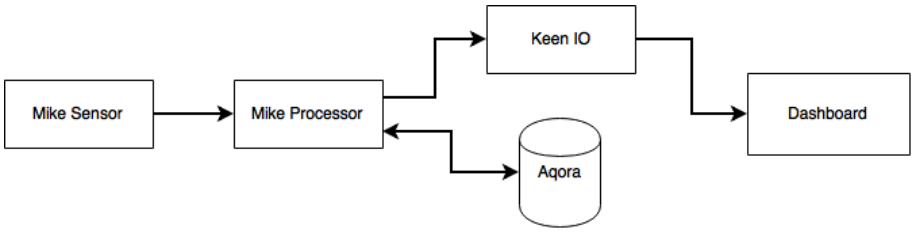

Figure 3.3: Mike scheme, with data flow represented by arrows.

Quantum Processor

Quantum processor is responsible to capture the Adafratuit TSL2561 photodiode's readings, convert those values to lux, and send them to Aqora.

Quantum processor is also responsible to process stored data on Aqora, returning metrics that are then stored on Keen IO (Ex: hourly lux average).

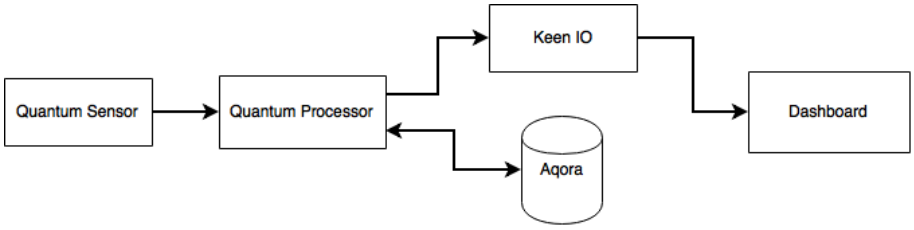

Figure 3.4: Quantum scheme, with data flow represented by arrows.

Presence Processor

Presence processor is responsible to capture the list of MAC Addresses present on the office's network, identify to which employee belongs the correspondent MAC Address, and send that information to Keen IO.

Presence also communicates with Quem to match the returned list of Mac Addresses with the company's employees.

Qalendar

Registers company events scheduled on the company's Google Calendar.

Qalendar communicates with Google Calendar to gather the data relative to the calendar's previous events. The processed data is then sent to Keen IO.

All the data relative to the calendar's event can be retrieved through Google Calendar's API.

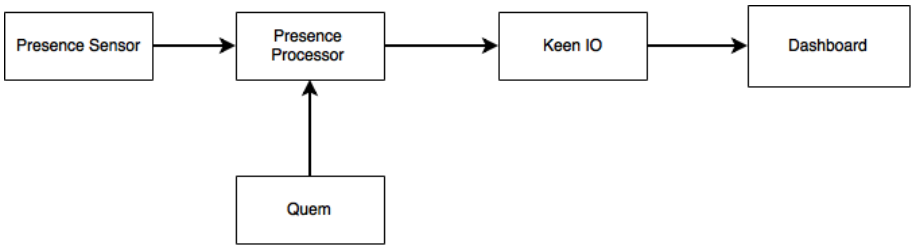

Figure 3.5: Presence scheme, with data flow represented by arrows.

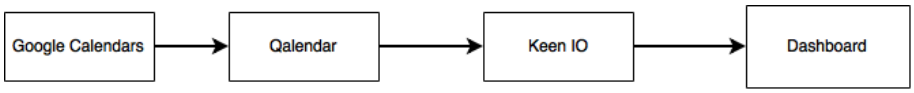

Figure 3.6: Qalendar scheme, with data flow represented by arrows.

Trello Quantifier

Quantifies activity on each Trello board correspondent to projects occurring at Whitesmith. Trello Quantifier collects and stores this data with the goal of measuring throughput, responsiveness, and other metrics in the future.

Trello Quantifier receives data from Trello about every activity occurring at the company's Trello boards. Trello Quantifier also communicates with Quem and Projects to match the received activity data with the correspondent employee and project. If the event is relevant, the processed data is then sent to Keen IO.

Trello Quantifier is also responsible to perform more complex calculations that cannot be retrieved from Keen IO, such as responsiveness.

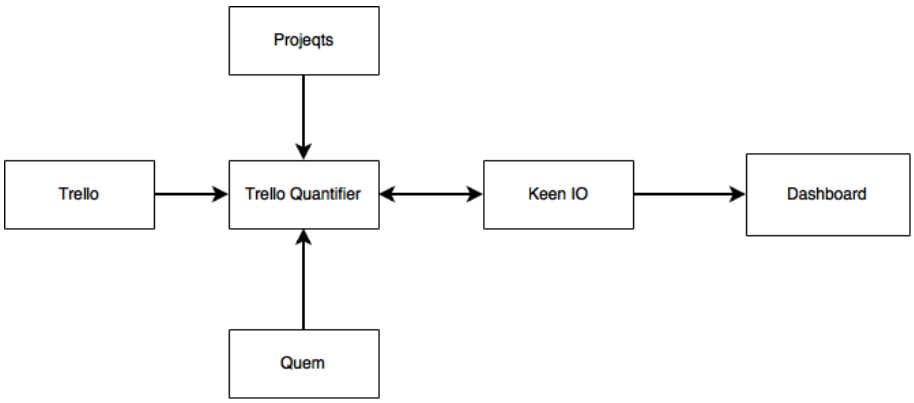

Figure 3.7: Trello Quantifier scheme, with data flow represented by arrows.

All the card's data is sent on every event that comes from Trello's Webhooks.

Github Processor

Quantifies each Github repository that belongs to the organisation, such as the number of lines of code added and deleted, and the number of new commits.

Github Quantifier communicates with Github to gather the statistics relative to each employee and Github repository. Similarly to Trello Quantifier, Github Quantifier also communicates with Quem and Projects to match the received Github data with the correspondent employee and project. The processed data is then sent to Keen IO.

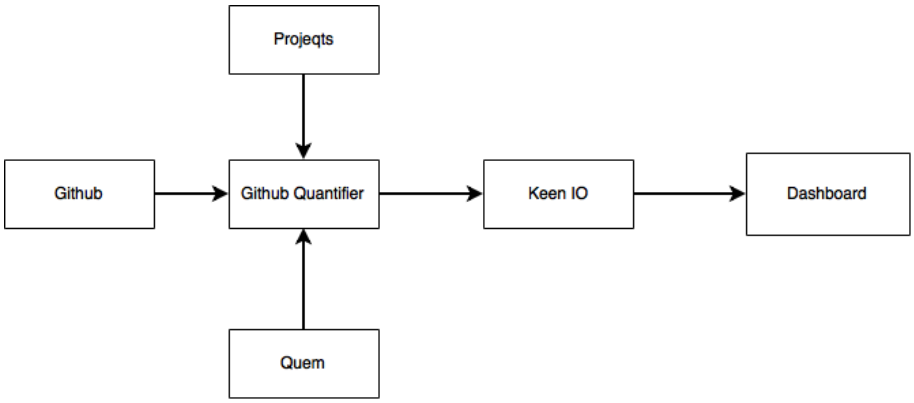

Figure 3.8: Github Quantifier scheme, with data flow represented by arrows.

Shrinq

Quantifies the level of happiness, by querying every employee through it's software communication platform - Slack.

Shrink communicates with Slack to send the survey questions to each employee, and receive the correspondent answers. Shrink communicates with Quem to match the received Slack data with the correspondent employee, and with Presence to acknowledge the employees telecommuting status for the current day. The processed data is then sent to Keen IO.

3.2.5 Data Visualisation

Dashboard

The Dashboard is the central point for the representation of the insights extracted from the processed data. The Dashboard is accessible by every employee.

The metrics are divided in three main categories:

- Happiness - Metrics relative to employees happiness.

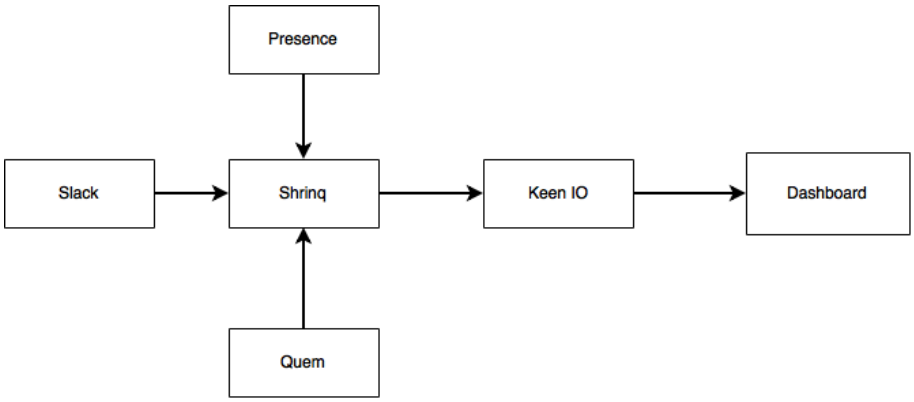

Figure 3.9: Shrinq scheme, with data flow represented by arrows.

- Productivity - Metrics relative to Github and Trello.
- Office Environment - Metrics relative to the office environment, such as noise, illuminance, events, and telecommuting status.

It was not expected for the Dashboard to be rich of insights by the end of this Master Thesis - data analysis will be performed by intersecting data from various data sources (Ex: productivity vs telecommuting status), which we preferred to focus on later after this project. Anyhow, the visualisation of retrieved data along the development of this project helped steer the right direction for the type of metrics to collect and process, and assure the quality of the collected data.

Chapter 4

Implementation

In this chapter we describe the implementation of this platform in more detail. From the languages and frameworks used for this project, to every component that was built during the course of this Master Thesis. These include: Data Processing Microservices, Company Info Microservices, and part of the Data Sources (sensors). In this chapter, Data Sources are described in the same section of the correspondent Data Processing Microservice - this applies to Mike, Quantum, and Presence.

4.1 Languages and Frameworks

In this project, different languages and frameworks are used. Cloud-based microservices use Ruby over the Sinatra framework, and Ruby on Rails. Microservices built on devices use Python. These languages and frameworks were chosen mostly because of velocity of development on each platform.

4.2 Cloud Hosting

With the exception of the case microservices built on devices, the services used in this project are hosted on the cloud - both first and third-party services. Each of the first-party services are hosted in an individual container server at Heroku - a PaaS cloud hosting service, which stands above the infrastructure of Amazon EC2. The cloud enables the ease of deployment of new microservices, and ease of integration with other services and devices - it gives the system modularity. It also enables a quick response to performance demand, by effortlessly scaling up and down the server instances, thus making it easier to adapt to the company's needs in terms of data processing.

4.3 Error Logging and System Logging

Every service performs error logging using Sentry[1] - an exceptions data logger hosted on the cloud. Email alerts are sent every time an error occurs on one of the microservices.

Some microservices perform system logging using Logentries[2] - a system monitoring data logger and analytics hosted on the cloud.

Both of them serve the purpose of general system information and aid during the system analysis and debugging.

4.4 Communication

Every service with each other through Hypertext Transfer Protocol (HTTP) - an application-level protocol that is generic, stateless and object oriented, providing a light and fast solution for communication in distributed, collaborative information systems [50].

We use the Representational Status Transfer pattern (REST) - the most common solution for web applications to provide data, due to its simplicity and focus on data state. It provides a powerful but simple tool for integration between systems. The used format in the system communications is always json.

Data can be collected either on the instant the event data is created on its Data Source, or at a later instant:

- Some metrics are collected through HTTP POST requests performed by Data Sources to the correspondent Processors, at the same instant that the data is created on the Data Source. This data is immediately processed and sent to its Data Storage platform, as it's received by the Data Processing Microservices.

- Other metrics are collected using background jobs - also called workers - scheduled on its Data Processing Microservices to, on a defined frequency, perform HTTP GET requests to the Data Source. This data is immediately processed and sent to its Data Storage platform.

4.5 Workers

Workers serve the purpose of collecting and processing data on a defined frequency. On this project, workers are used for collecting and processing batches of data for situations where it cannot be performed in real-time.

In this project, workers are performed with:

[1] http://getsentry.com
[2] https://logentries.com/

- Heroku Scheduler - an add-on at our hosting provider that allows scheduling of rake tasks.
- Cron - a time-based scheduler used in Unix-like computer operating systems as Raspbian, the OS installed in our devices.

4.6 Components

4.6.1 Data Processing Microservices and Data Sources

Mike

Mike consists in a data logger device built with a Raspberry Pi and a microphone.

The Raspberry Pi has installed Raspbian OS - a Linux based OS - with PyAudio library. PyAudio provides Python bindings for PortAudio - a cross-platform audio I/O library - for access of the microphone readings of the RMS amplitude.

Mike's microphone has a frequency response from 20Hz to 20 kHz - which corresponds to the human hearing range - and it's connected to the Raspberry Pi through a USB connection. No type of weight filter is applied to Mike - as for example A-Weighting, which would account for the relative loudness perceived by the human ear. The microphone produces a voltage which is proportional to the sound pressure. Mike collects this values and calculates the RMS amplitude.

RMS amplitude is defined as the square root of the mean of the square of the instantaneous sound amplitude.

$$Xrms = \sqrt[2]{\frac{1}{n}(x_1^2 + x_2^2 + ... + x_n^2)} \tag{4.1}$$

RMS amplitude vary with the distance from the sound origin - there is a drop of 6 dB per doubling of distance. There are no standards for measuring sound levels. In the case of ambient environmental measurements of background noise, distance is usually not quoted as no single source is present. In the case of Whitesmith's office - which has 52 square meters - the farthest that a sound can be originated from the microphone is approximately 8.4 meters.

As described in the previous chapter, Mike stores data in both Aqora and Keen IO:

Aqora is used for storage of the instantaneous RMS amplitude readings, which are performed every 0.5 seconds and immediately sent to the database. The frequency of every 0.5 seconds is high enough to perform an adequate analysis of the ambient sound levels at the office, without exceeding Aqora's storage capacity in the mid-term, or the Raspberry Pi's processing power.

Keen IO is used for storage of hourly RMS amplitude averages, which are calculated every hour, by processing data on Aqora relative to the previous hour.

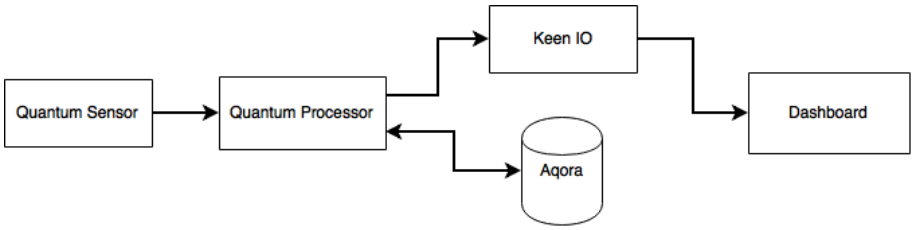

Figure 4.1: Quantum scheme, with data flow represented by arrows.

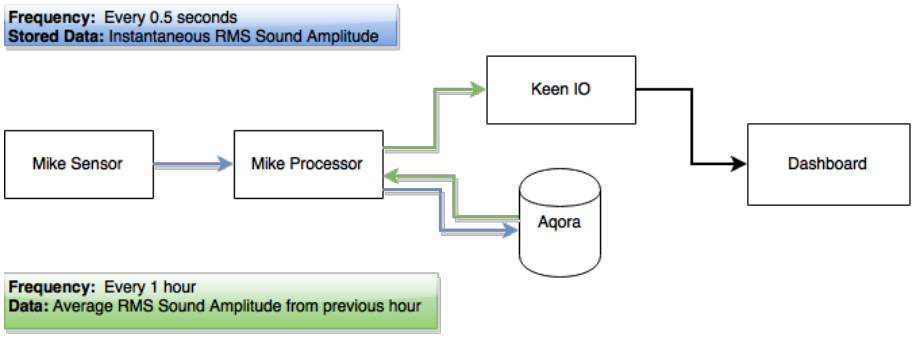

Figure 4.2: Mike scheme, with data flow represented by arrows.

Quantum

Quantum consists in a data logger device built with a Raspberry Pi and a Adafruit TSL2561 digital sensor.

The Raspberry Pi has installed Raspbian OS - a Linux based OS - and a library specifically calibrated for this sensor, for access to the lux readings.

The Adafruit TSL2561 sensor contains both infrared and full spectrum diodes. Adafruit TSL2561 has a dynamic range from 0.1 to 40,000 lux. The draw of electricity current is extremely low - about 0.5mA when actively sensing - making it very suitable for low power data-logging systems [66]. Adafruit TSL2561 sensor is connected to the Rasperry Pi through I2C interface.

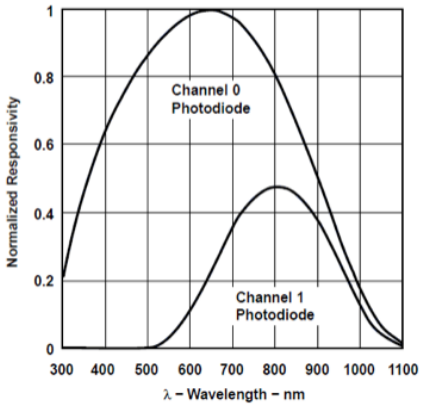

Figure 4.3: Adafruit TSL2561 spectral responsivity diagram [66].

Quantum is mounted in the middle of one of the walls of Whitesmith's office. Quantum also stores data in both Aqora and Keen IO:

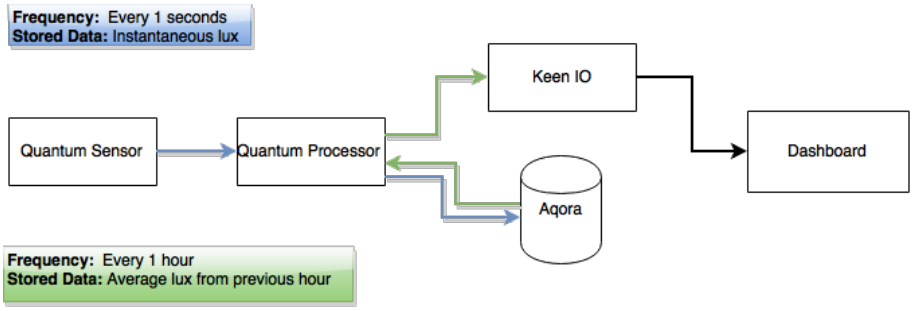

Figure 4.4: Quantum scheme, with data flow represented by arrows.

Aqora is used for storage of the instantaneous lux readings, which are performed every 1 second and immediately sent to the database. The frequency of every 1 second is high enough to perform an adequate analysis of the ambient illuminance levels at the office, without exceeding Aqora's storage capacity in the mid-term, or the Raspberry Pi's processing power.

Keen IO is used for storage of hourly lux averages, which are calculated every hour, by processing data on Aqora relative to the previous hour.

Presence

Presence consists in a data logger device built with a Raspberry Pi.

The Raspberry Pi has installed Raspbian OS - a Linux based OS. A script makes use of Nmap (Network Mapper) - a security scanner used to discover hosts and services on a computer network, thus creating a "map" of the network [67].

Through Nmap, Presence captures the MAC Addresses that are currently connected to the office's network.

Immediately next to receiving the list of MAC Addresses, Presence queries Quem to acknowledge to which employee belongs the correspondent MAC Address. For every employee at the office, an event with its Quem ID is sent to Keen IO.

A worker executes this process every 15 minutes, giving us reasonable granularity to measure employees telecommuting status.

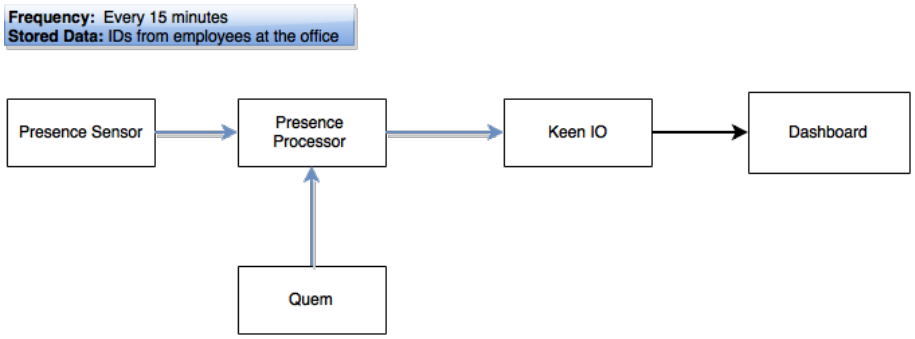

Figure 4.5: Presence scheme, with data flow represented by arrows.

Qalendar

Qalendar is hosted on the cloud, at Heroku's platform. It's built using Ruby language, over the Sinatra framework.

Qalendar communicates with Google Calendar to query the events on Whitesmith's calendar. The retrieved data includes the event's description, place, and date.

A worker executes this process every day, to query and process the calendar events from the previous day. This data is immediately sent to Keen IO.

The frequency of querying data every day permits the registration of events rapidly enough for analysis and comparison with other metrics (Ex: noise at the office), and at the same time guaranteeing that the event has been scheduled on the calendar previously to the calendar event.

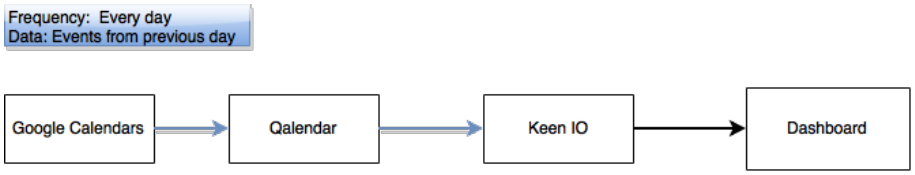

Figure 4.6: Qalendar scheme, with data flow represented by arrows.

Github Quantifier

Github Quantifier is hosted on the cloud, at Heroku's platform. It's built using Ruby language, over the Sinatra framework.

Github Quantifier communicates with Github to query weekly development statistics relative to each user on each repository. The retrieved data includes number of commits, number of lines of code added, number of lines of code deleted.

For each user's statistics received from Github, Github Quantifier queries Quem and Projeqts to acknowledge the Quem ID and Projects ID to which the statistics belong, respectively. The collected and processed data is immediately sent to Keen IO.

A worker executes this process every week - the same frequency that these stats are created and provided by Github.

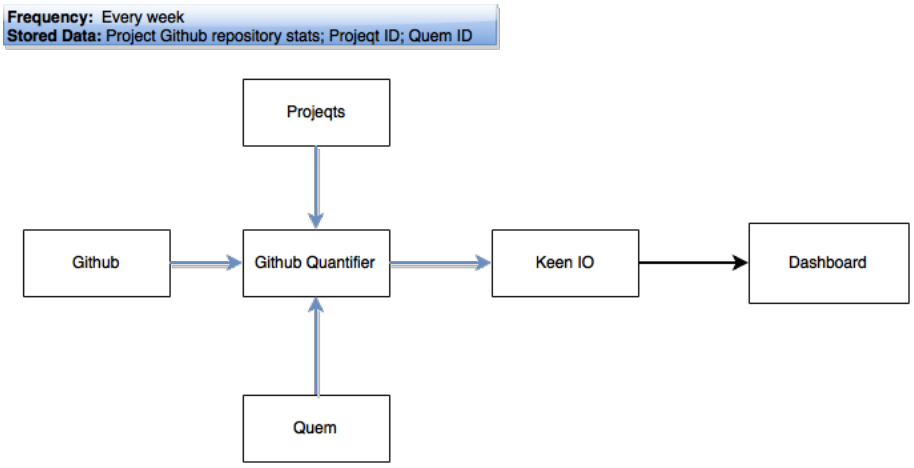

Figure 4.7: Github Quantifier scheme, with data flow represented by arrows.

Shrinq

Shrinq is hosted on the cloud, at Heroku's platform. It's built using Ruby language, over the Sinatra framework.

Shrinq uses the interval sampling method - is programmed to survey employees every day, at the hour preferred by each employee. Shrinq's questions tap into the cognitive component.

Shrinq was built to make the user experience as simple and fluid as possible, with the goal of increasing the response rate. This way, Shrinq makes a low number of simple questions.

For the first question we use the most asked question in happiness surveys: "How happy are you today?" The following two questions have the purpose of

better understanding the answer to the first: "What was the main reason?"; "What is blocking you from doing your work? What suggestions do you have to make it something better (project, company, etc)?".

Question 1 has three possible answers, values of which range from 1 to 3:

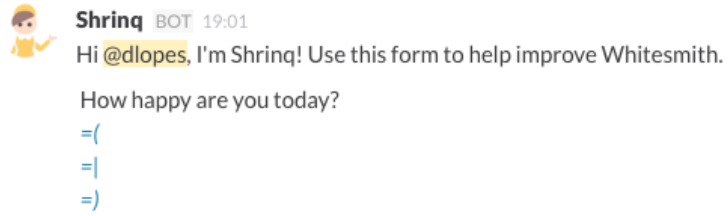

Figure 4.8: Shrinq question 1.

The =(smile corresponds to 1, the =| smile to 2, and the =) smile to 3. The question can be easily and rapidly answered by clicking on the desired option.

The second is also a multiple choice question that users can answer with a click.

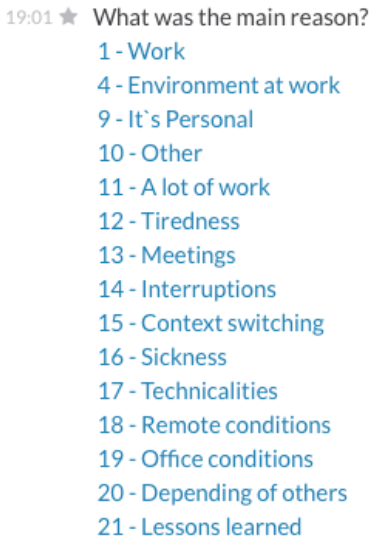

Figure 4.9: Shrinq question 2.

The third is an open answer question, where employees can write down their thoughts in a more expressive manner. This answer is useful to acknowledge information that is not possible to express on the previous two multiple-choice questions.

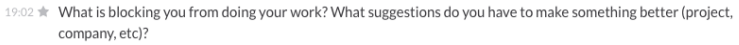

Figure 4.10: Shrinq question 3.

For every answer given by each employee, Shrinq communicates with Quem to query the Quem ID of the correspondent employee's Slack account, and with Presence to acknowledge if this employee was telecommuting on the present day. After having this information, an event with the employee's answer, Quem ID, and telecommuting status, is sent to Keen IO.

A worker executes this process every day, at the hour preferred by the employee. This hour is also defined on Quem. In case a employee doesn't answer the survey on the preferred hour, the employee will be again surveyed three hours after, also with the goal of increasing response rate.

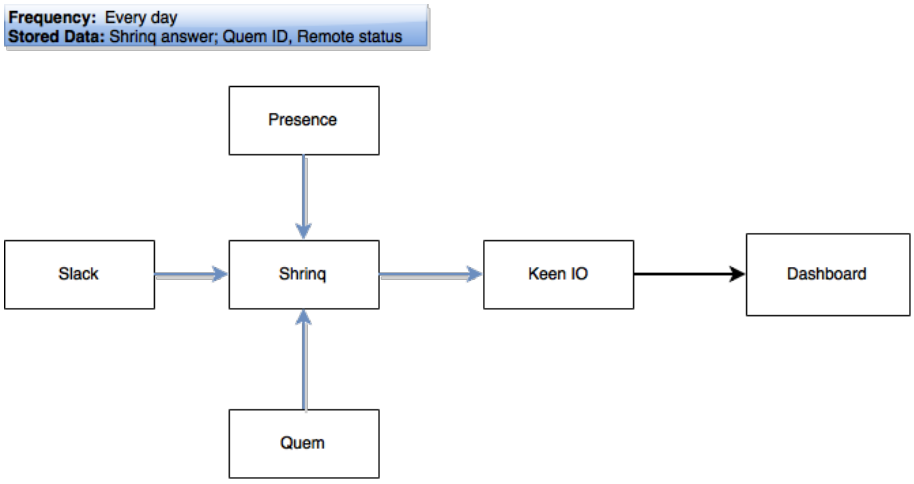

Figure 4.11: Shrinq scheme, with data flow represented by arrows.

Trello Quantifier

Trello Quantifier is hosted on the cloud, at Heroku's platform. It's built using Ruby language, over the Sinatra framework. Trello Quantifier analyses activity on each Trello board correspondent to projects occurring at Whitesmith.

Trello Quantifier has the goal of measuring Throughput, Maturity, and Responsiveness:

- Throughput is based on the number of completed Tasks. Tasks are done when its correspondent Card is moved to a Live list on Trello.

- Maturity is based on the number of defects found in the software. For each defect found, a card is created on the Bugs list. The more found defects, the lower is the level of maturity of the software.
- Responsiveness, which consists in:
  - Number of tasks completed per week.
  - The average time per week that tasks from starting until they're concluded. (Average Time to Market)
  - The average time per week that tasks from starting until they move to the next List (list which corresponds to the Task state). (Average Time to Next List)

Trello communicates with Trello Quantifier using a technology called Webhooks - user-defined HTTP callbacks, which are triggered whenever there is activity on a Trello board.

When a HTTP callback is received, Trello Quantifier will firstly verify if the activity is relevant - if a new Card was created, or moved to a different List.

In case the data is relevant, Trello Quantifier will communicated with Quem and Projeqts to acknowledge to which employees the Card is assigned to, and to which company Project it belongs. After processing, the information is immediately sent to Keen IO. This permits us gather the number of done tasks, which are used display the throughput and number of completed tasks per week.

To quantify the Average Time to Market, and Average Time to Next List, Trello Quantifier needs to process the information stored on Keen IO. It does it by gathering from Keen IO the cards information belonging to each Project, and execute the following calculations:

- Time taken from the current list to the next, in days
- Time taken from the Next Up list to the Live list, in days.

Finishing with the weekly average for each project.

Both processes can be visualised in 4.12.

4.6.2 Company Info Microservices

Quem

Quem is hosted on the cloud, at Heroku's platform. It's built using Ruby language, over the Ruby on Rails framework.

For every new employee at the company, a new employee entry is created on Quem. There, personal employee information is stored such as: Quem ID, Name, Identification Card Number, Date of Birth, Computer MAC Address, Trello ID, Slack User ID, and Github ID.

This information is then used by other services.

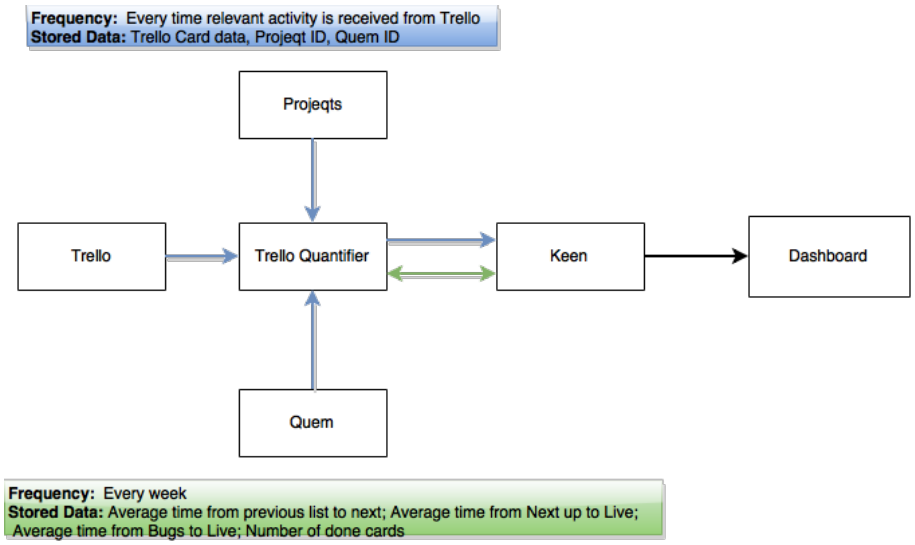

Figure 4.12: Trello Quantifier scheme, with data flow represented by arrows.

Projeqts

Projeqts is hosted on the cloud, at Heroku's platform. It's built using Ruby language, over the Ruby on Rails framework.

For every new project at the company, a new project entry is created on Projeqts. There, project information is stored such as: Projeqt ID, Name, Description, Trello Board ID, and Github Repositories IDs.

This information is then used by other services.

Chapter 5

Results

In this chapter screenshots of the built Data Visualisation Dashboard are shown. The Dashboard is divided in three main categories, correspondent to the categories of the data being collected: Productivity, Happiness, and Office Environment. Inside each of those categories, different data views are displayed. All the displayed charts on the dashboard are interactive - on data point mouse hover, more information is displayed to the user, such as date, variable name, and variable value.

5.1 Productivity

5.1.1 Projects

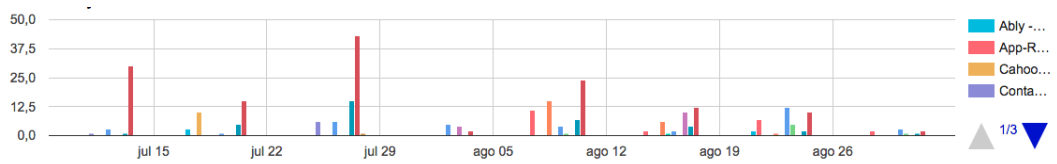

Figure 5.1: Number of finished tasks per week per project, over a period of 8 weeks.

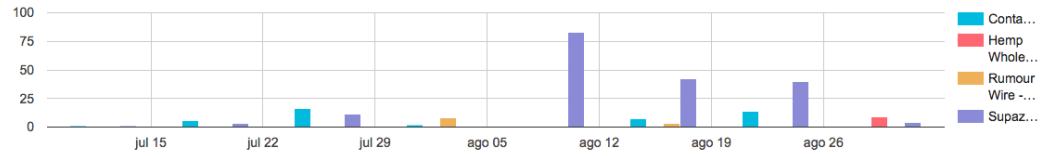

Figure 5.2: Number of encountered software bugs per week per project, over a period of 8 weeks.

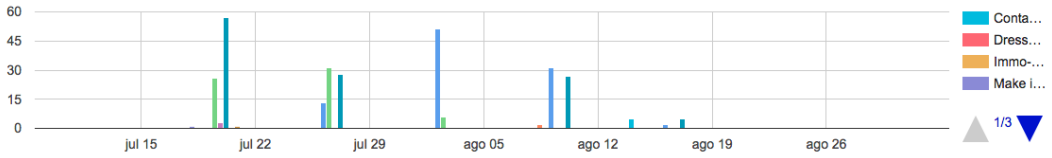

Figure 5.3: Number of git commits per week per project, over a period of 8 weeks.

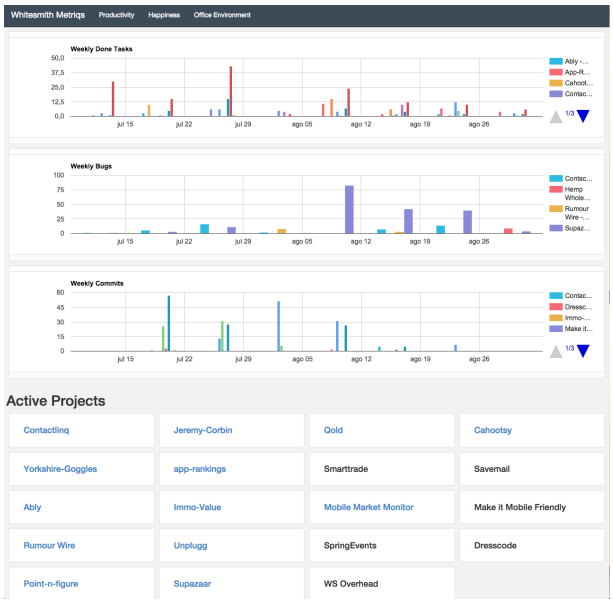

Figure 5.4: Dashboard - Productivity projects overview.

5.1.2 Single Project

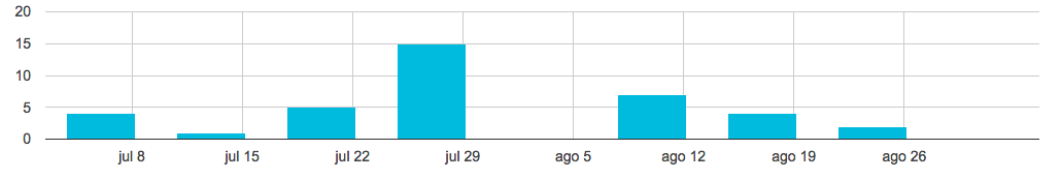

Figure 5.5: Number of finished tasks per week on a single project, over a period of 8 weeks.

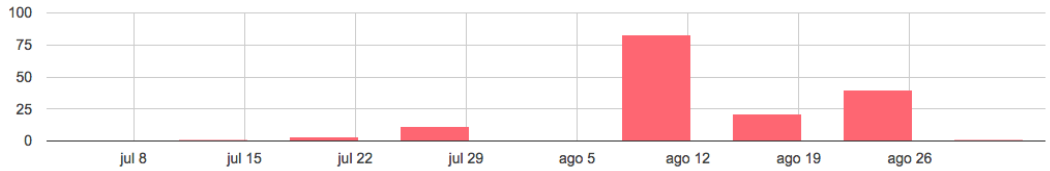

Figure 5.6: Number of encountered software bugs per week on a single project, over a period of 8 weeks.

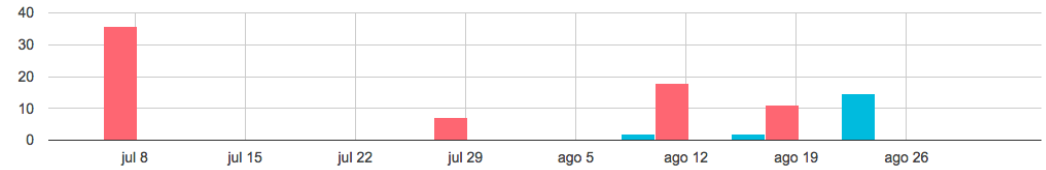

Figure 5.7: Average days from Bugs and Next Up to Live - Average number of days, per week, that new tasks take from their appearance on the Bugs list until it's finished; average number of days, per week, that new tasks take from their appearance on the Next Up list, until they are finished. Data from a period of 8 weeks.

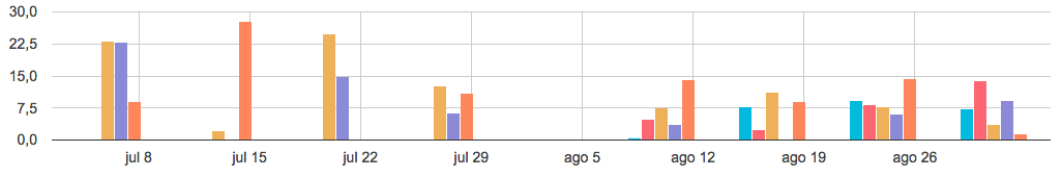

Figure 5.8: Average days from previous list to next list - Average number of days, that tasks take on each list, per week, over a period of 8 weeks. The lists are: Next Up, In Progress, Code Review, Internal QA, Acceptance, Live.

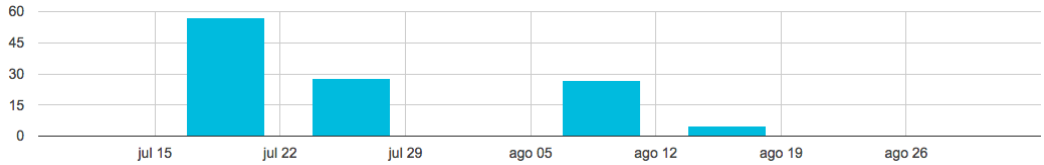

Figure 5.9: Number of git commits on a single project, over a period of 8 weeks.

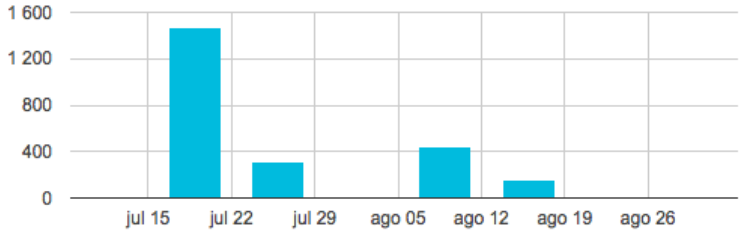

Figure 5.10: Number of lines of code added per week on a single project, over a period of 8 weeks.

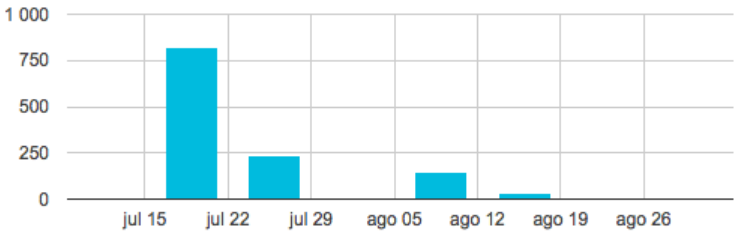

Figure 5.11: Number of lines of code deleted per week on a single project, over a period of 8 weeks.

Figure 5.12: Dashboard - Productivity single project overview.

5.2 Happiness

5.2.1 Short Term

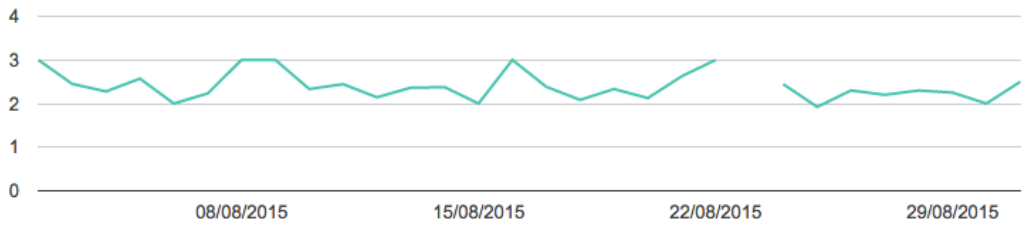

Figure 5.13: Company happiness average per day, over a period of 30 days.

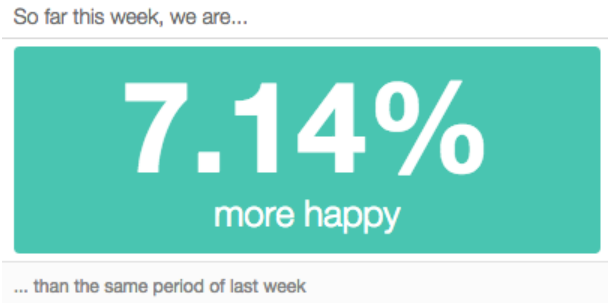

Figure 5.14: Relative difference between the current week's happiness average, and the happiness average of the homologous period from last week.

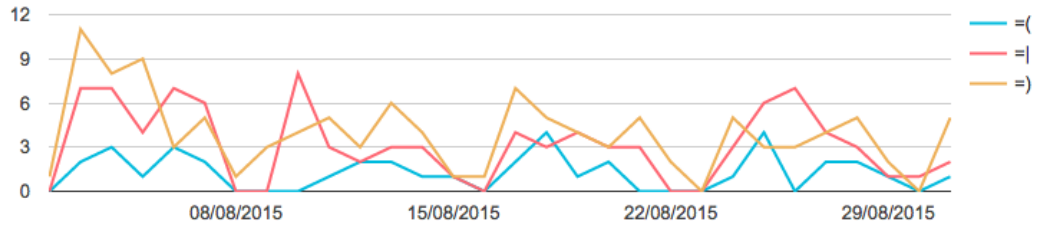

Figure 5.15: Number of responses given to the "How happy are you today?" question, per value, per day.

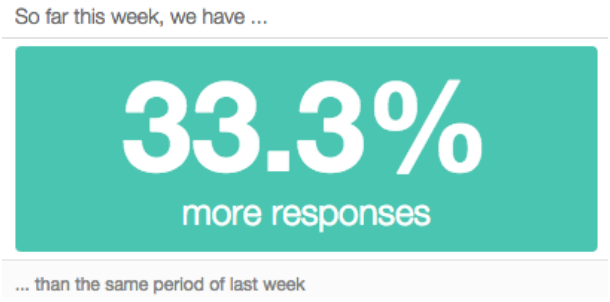

Figure 5.16: Relative difference between the current week's number of responses given to the "How happy are you today?" question, and the the number of responses given to the same question on the the homologous period from last week.

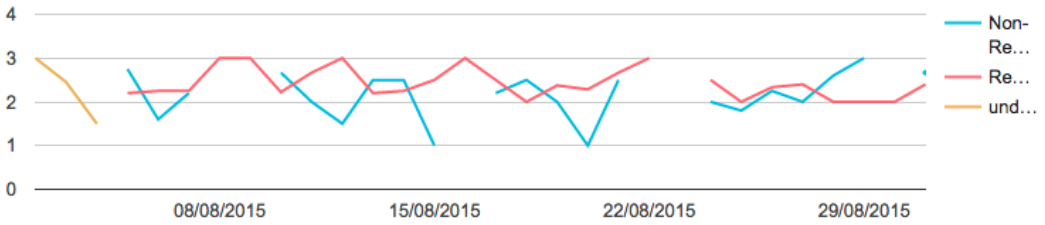

Figure 5.17: Remote vs Non-Remote Happiness Average - Happiness average of employees working remote and employees working non-remote, per day, over a period of 30 days.

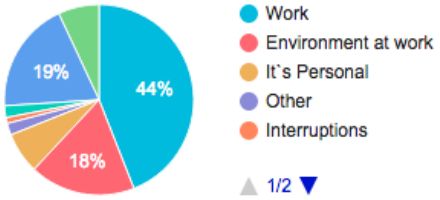

Figure 5.18: Why people are happy - Answers given to the question "What was the main reason?", when answering "=)" for the "How happy are you today?" question, for a period of 30 days.

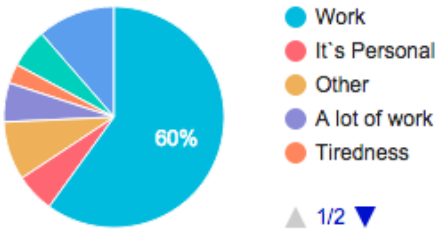

Figure 5.19: Why people are sad - Answers given to the question "What was the main reason?", when answering "=(" for the "How happy are you today?" question, for a period of 30 days.

joao_magalhaes: lack of experience in some technologies areas

renatodeleao: became a svg sprites sensei today. Well not as much as i like, but at least i can create icon systems.

Figure 5.20: Responses given to the "What is blocking you from doing your work? What suggestions do you have to make something better (project, company, etc)?" question, in the last 7 days.

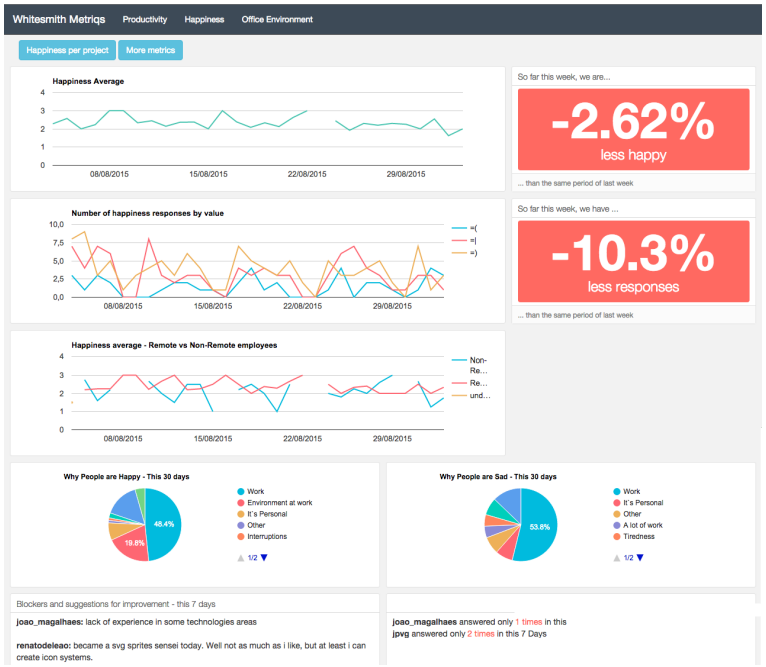

Figure 5.21: Dashboard - Happiness short term overview.

5.2.2 Long Term

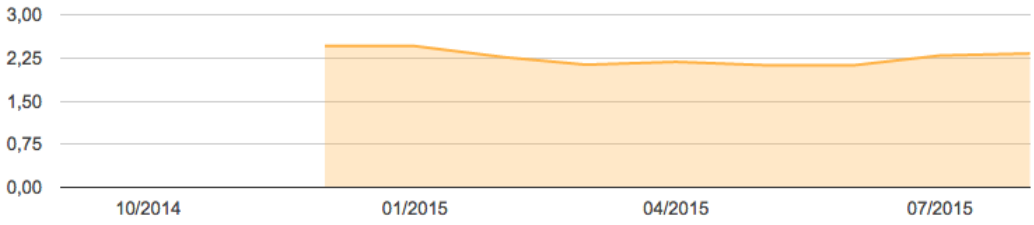

Figure 5.22: Company happiness average per month, over a period of 9 months.

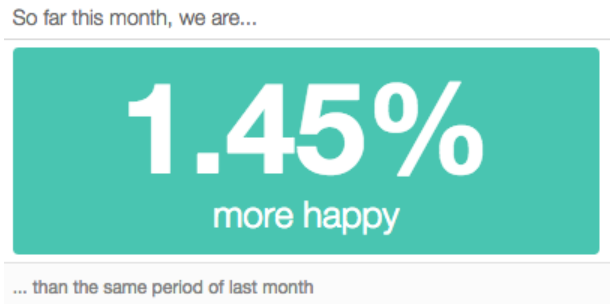

Figure 5.23: Relative difference between the current month's happiness average, and the happiness average of the homologous period from last month.

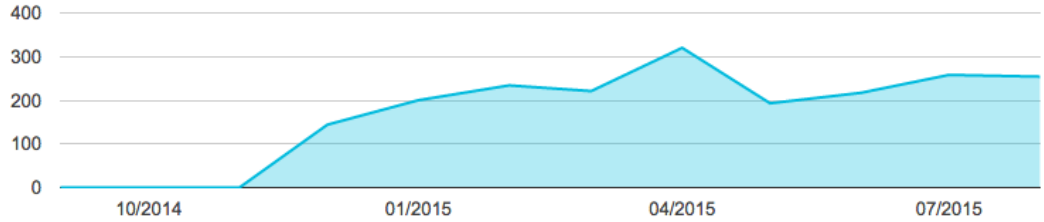

Figure 5.24: Number of responses given to the "How happy are you today?" question, per month.

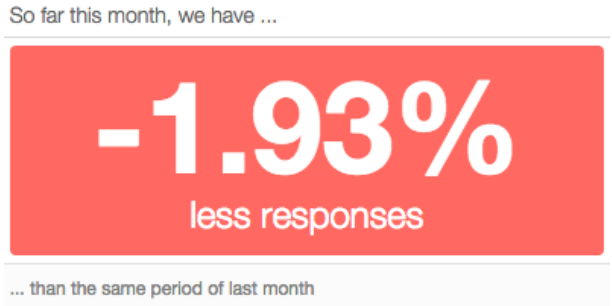

Figure 5.25: Relative difference between the current month's number of responses given to the "How happy are you today?" question, and the the number of responses given to the same question of the homologous period from month.

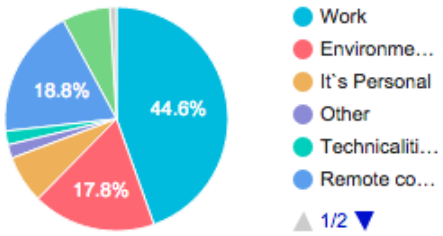

Figure 5.26: Why people are happy - Answers given to the question "What was the main reason?", when answering "=)" for the "How happy are you today?" question, for a period of 3 months.

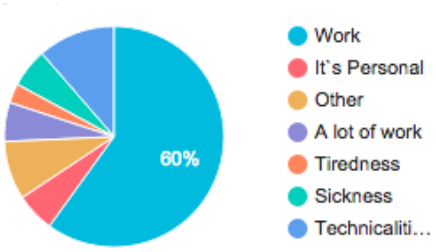

Figure 5.27: Why people are sad - Answers given to the question "What was the main reason?", when answer "=(" for the "How happy are you today?" question, for a period of 3 months.

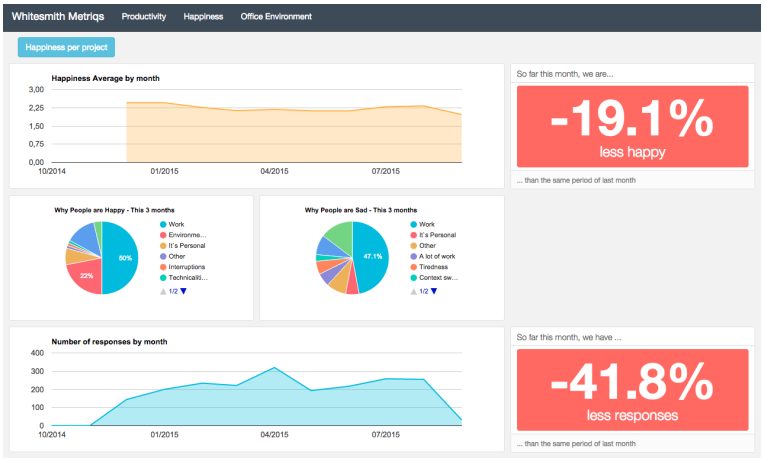

Figure 5.28: Dashboard - Happiness long term overview.

5.3 Office Environment

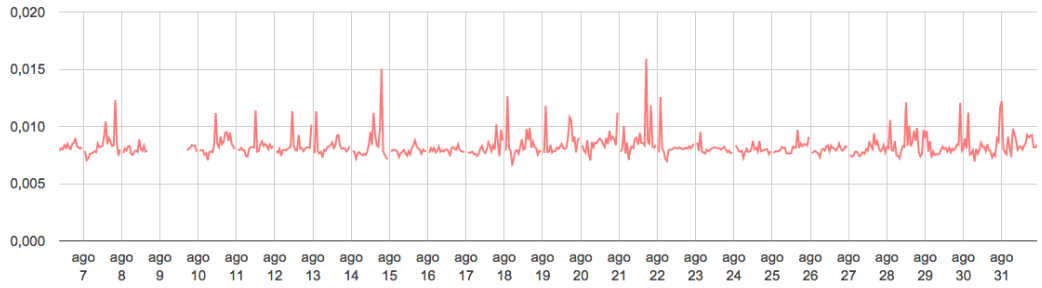

Figure 5.29: Noise - Relative noise in the office per hour, for a period of 30 days.

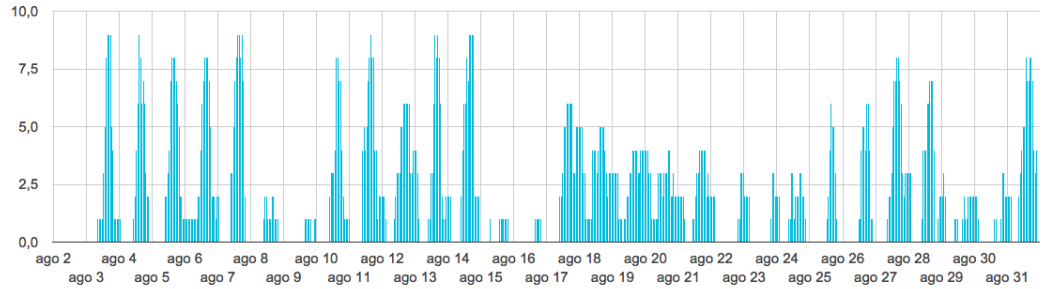

Figure 5.30: People - People at office per hour, for a period of 30 days.

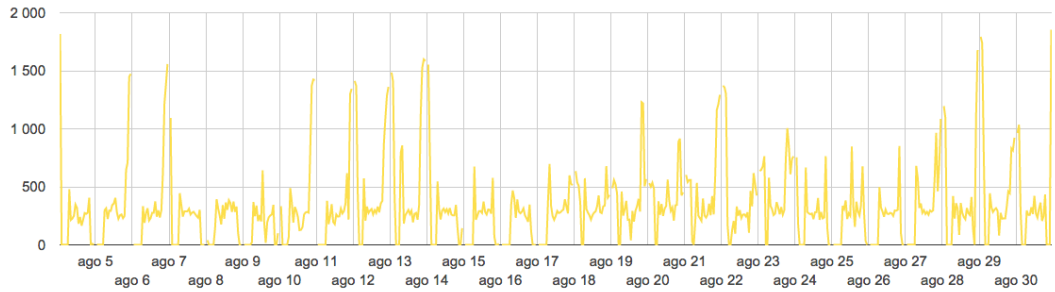

Figure 5.31: People - Illuminance at office per hour, for a period of 30 days.

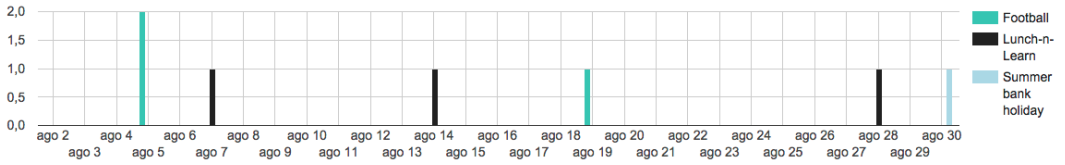

Figure 5.32: Events - Events per day over a period of 30 days.

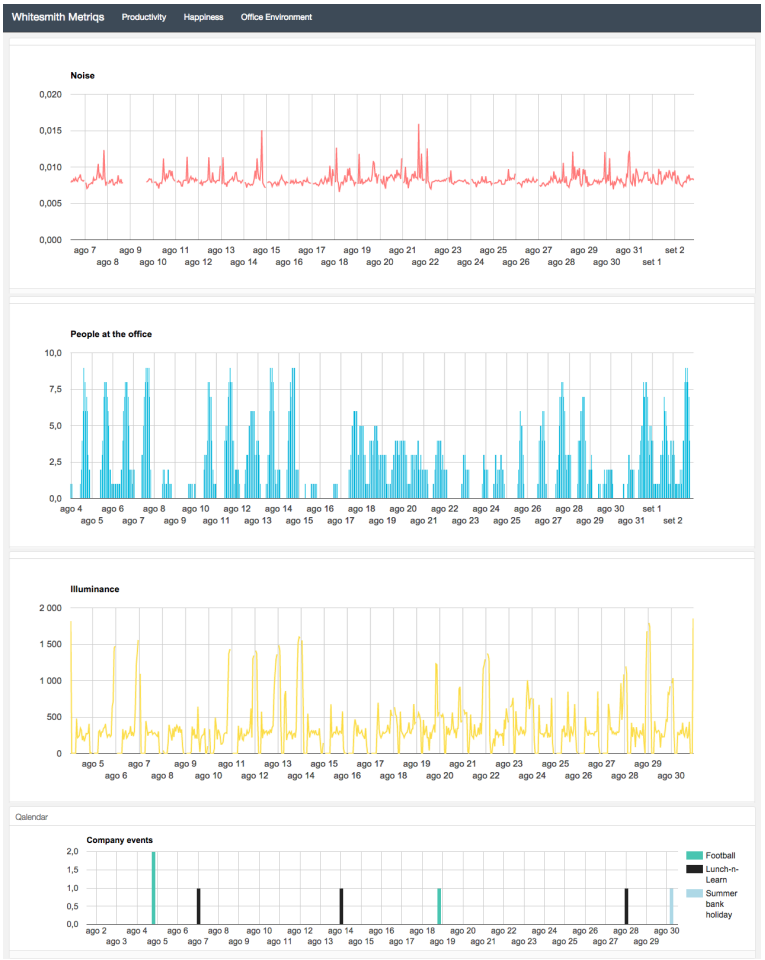

Figure 5.33: Dashboard - Office Environment overview.

Chapter 6

Conclusion and Future Work

The goal of this project was to build an infrastructure to collect workplace data on an ongoing basis. We took advantage of the current State of the Art of Quantified Self, Internet of Things, and Microservices Architecture, and applied it to the workplace - the Quantified Workplace.

In this chapter, the conclusion is discussed from the perspective of the final results and the community validation. In the last section, a vision of the next steps for this platform is described.

6.1 Final Result

For this project seven Data Processing Microservices, two Company Info Microservices, three sensors (part of Data Sources), and a Data Visualization Dashboard were built. These communicate with each other and third party services under a MSA, to collect, process, and display insights about Whitesmith. The insights are divided in three areas - Happiness, Productivity, and Office Environment. This is done automatically on a frequent basis - hourly, daily, or weekly, - contrasting with the current methods used at most companies to measure their performance.

The developed architecture permits the easy couple and decouple of Data Processing Microservices and Data Sources to the platform. These communicate with each other under a common language, but are each one using the technology that best suits the collection and processing of the variables that is responsible for. The developed architecture also makes it possible to improve the measurement of the current Office Environment variables, by extending the mesh of devices for that particular variable, without affecting the other Data Processing Microservices. On the other hand, the adoption of MSA for this project slowed the development of the platform - due to the operations overhead of having multiple microservices to develop and maintain by a single person, and due to the complexity of communications between the various first and third-party services.

The central point of data visualization is the Dashboard, which is accessible by Whitesmith employees only, at http://whitesmith-metriqs.herokuapp.com.

This consummates the scope of work outlined for this project on the first chapter:

- **A microservices based architecture for data collection on SMEs**, modular for easy attachment and decoupling of future services and devices.
- **Continuous data collection, processing, and storage** of SME's workplace metrics
- **Integration with Internet of Things devices**, with the goal of measuring physical environmental quantities.
- **A data visualisation platform** to make the collected data available for analysis.

Even though performing data analysis wasn't the goal for this project, and consequently limited effort being applied on this part, some interesting insights were already found along the way. For example:

- On the Productivity Projects charts, we notice that Supa project has a higher throughput than expected, but with considerable fluctuations through the weeks.
- On the Productivity Single Project charts, for the case of Supa project - the project represented in the Results Single Project section of this Master Thesis, - we notice that:
  - There was a drastic increase of bugs on the week of 5th of August relative to the previous weeks. This was due to the fact that a release date for the project was defined for following weeks, and thus a higher level of testing was taken.
  - Bugs are taking much longer to go to Live than other types of tasks.
  - The tasks on the Internal QA list are taking more time than expected on that state.
- On the Happiness Short Term charts we notice that:
  - The two main reasons for people feeling happy are related with work and the environment at work.
  - The two main reasons for people feeling sad are related with work and technicalities.
- On the Happiness Long Term charts we are starting to notice that:

  - There's a tendency for people working remotely (telecommuters) to be more happy than people working from the office.
  - The same pattern as on the Happiness Short Term charts is emerging: the two main reasons for feeling happy are related with work and the environment at work; and the two main reasons for people feeling sad are related with work, and technicalities.
  - People have been answering less to Shrinq's survey in the latest months. The hour at which the survey is taken may need to change, or other incentives should be implemented.
  - The happiness average has been lower in the months of March, May, and June of 2015, and getting higher again in the latest months.

- On the Office Environment charts we notice that:
  - The noise is tendentiously higher on Fridays - the days where we usually have more events or activity in the office.
  - The number of people working from the office in the past weeks is lower than what is usual at Whitesmith. Probably due to vacations and people choosing to work remotely during part of August.

6.2 Community Validation

During the development of this project, a internet article describing our process of measuring happiness - using Shrinq - was published. This article grabbed the attention of various people from the Quantified Self and Data Science communities. The article was shared on Twitter by 59 people, featured on QuantifiedSelf.com - the official Quantified Self community - and directly praised by several people working at Keen IO[1] - a company that already raised $17.8M from ventures as Sequoia Capital, 500 Startups, and Amplify Partners [68].

Both Ernesto Ramirez - Program Director of QuantifiedSelf.com - and Dan Kador - CTO of Keen IO - showed interest in our future publication of the Open Source version of Shrinq.

Meetings have been held with Keen IO to discuss the data visualization dashboard. Interest was demonstrated by the Keen IO team in the platform, and in publicizing our dashboard as an unique example of using Keen IO tools.

6.3 Future Work

After this project, there is interest in developing the platform further:

[1] Josh Dzielak, Dan Kador, Maria Dumanis, Alexandra Meyer, and Maggie Jan

- Integration with new devices - No only there are other devices that we can install in the office (Ex: humidity sensors), but there is also the opportunity to support wearables/fitness trackers with the platform - devices such as Fitbit, Basis, Jawbone, and Apple Watch, that some employees already use, can be integrated to collect health information such as sleep quality, body temperature and heart rate. This information can later be crossed with productivity and happiness data already being collected.

- Expand data collection of the current devices - The data collection of some devices can be expanded to collect more information about the physical variables. For example, Mike could also measure the frequency of the sound at the office, and Quantum the light spectrum frequency and lamps flickering frequency.

- Data analysis - Insightful knowledge is born from the intersection of data from the different data sources. For that, a scientific approach of experimentation is necessary, to find correlations between the collected data, and evaluate which correspond to causations. Application of Artificial Intelligence to automatically analyse data and gather insights is also a possibility inside the domain of expertise of Whitesmith.

- Integration with new productivity and communication tools - Developing a out-of-the-box solution, for easy setup at various companies is the final challenge for this work. More and more companies are looking for Quantified Workplace solutions for their organisations, and this marks as an opportunity for developing a product that could be sold to those companies. For that, the platform should be prepared to be able to adapt to the different organisations and processes. This will require further market research about the needs of the different organisations looking for these solutions, followed by the integration of the tools and processes used by them, by providing an API to gather data from the already existent devices in the market.

Bibliography

[1] Sierra Cedar. 2014-2015 HR Systems Survey. Technical Report 17th Annual Edition, 2014.

[2] Lee Bryant. The Quantified Organisation: can change become routine? Available at `https://www.aether.com/quantifiedself`. Accessed on 13/08/2015.

[3] Gary Wolf. Quantified Self. Archived from the original `https://www.aether.com/quantifiedself` on 26/03/2012. Accessed on 27/08/2015.

[4] Wikipedia. Quantified Self. Available at `https://en.wikipedia.org/wiki/Quantified_Self`, accessed on 25/8/2015.

[5] Benjamin Franklin. *The Autobiography of Benjamin Franklin*. Dover Publications, 1791.

[6] Jan Veira Harald Bauer, Mark Patel. The Internet of Things: Sizing up the opportunity. Technical Report December, McKinsey, 2014.

[7] Ming Liu. A Study of Mobile Sensing Using Smartphones. *International Journal of Distributed Sensor Networks*, page 11, 2013.

[8] QuantifiedSelf.com. Guide to Self Tracking Tools. Available at `http://quantifiedself.com/guide/`, accessed on 25/8/2015.

[9] Dr J. R. Brotherhood. Nutrition and Sports Performance. *Sports Medicine*, September 1984, Volume 1, 1984.

[10] Gary Wolf. The Data-Driven Life. Available at `http://www.nytimes.com/2010/05/02/magazine/02self-measurement-t.html?_r=0`, accessed on 25/8/2015.

[11] Melanie Swan. Emerging Patient-Driven Health Care Models: An Examination of Health Social Networks, Consumer Personalized Medicine and Quantified Self-Tracking. 2009.

[12] Maeve Duggan Susannah Fox. Tracking for Health. Technical report, Pew Research Center, jan 2013.

[13] Within the internet of things, quantified self startups grab $318m from vcs in the last year, November 2013. Available at https://www.cbinsights.com/blog/quantified-self-venture-capital/, accessed on 25/8/2015.

[14] Lee Bryant. The Quantified Organisation: people-powered digital transformation, 2015. Available at http://www.druckerforum.org/blog/?p=870, accessed on 25/8/2015.

[15] H. James Wilson. Wearables in the Workplace, 2013. Available at https://hbr.org/2013/09/wearables-in-the-workplace, accessed on 25/8/2015.

[16] Daniel A. Wren. *The History of Management Thought*. Wiley, 2005.

[17] Suzanne M. Crampton Jitendra M. Mishra. Employee Monitoring: Privacy in the Workplace? 1998.

[18] Harvard Business Review David A. Garvin. How Google Sold Its Engineers on Management, 2013. Available at https://hbr.org/2013/12/how-google-sold-its-engineers-on-management, accessed on 25/8/2015.

[19] Chris W. Clegg Matthew C. Davis, Desmond J. Leach. The Physical Environment of the Office: Contemporary and Emerging Issues. Volume 26, 2011.

[20] Lorraine E. Maxwell. Noise in the Office Worplace. Technical report.

[21] London Ruby developer working conditions Survey Summary. Available at http://lety.co/blog/2014/09/16/london-ruby-developers-working-conditions/, accessed on 25/8/2015.

[22] Spencer Rugaber Chris Parnin. The Physical Environment of the Office: Contemporary and Emerging Issues. 2010.

[23] Audio Engineering Society Glen Ballou. *Handbook For Sound Engineers*. Taylor & Francis, 2015.

[24] Equal Loudness Curves. Available at http://hyperphysics.phy-astr.gsu.edu/hbase/sound/eqloud.html#c1, accessed on 04/9/2015.

[25] IEC 61672-1:2013 . Available at https://webstore.iec.ch/publication/5708, accessed on 04/9/2015.

[26] A-weighting. Available at `https://en.wikipedia.org/wiki/A-weighting`, accessed on 04/9/2015.

[27] SCENIHR (Scientific Committee on Emerging and Newly Identified Health Risks). Health effects of artificial light. 2012.

[28] Duco A. Schreuder. *Vision and Visual Perception: The Conscious Base of Seeing*. 2008.

[29] M Münch. Effects of realistic office daylighting and electric lighting conditions on visual comfort, alertness and mood. April 2015 vol. 47 no. 2 192-209, 2014.

[30] Kurt A. Smith Sat Bir S. Khalsa Shantha M. W. Rajaratnam Eliza Van Reen Jamie M. Zeitzer Charles A. Czeisler Steven W. Lockley J Clin Joshua J. Gooley, Kyle Chamberlain. Exposure to room light before bedtime suppresses melatonin onset and shortens melatonin duration in humans. 2010.

[31] George C. Brainard Steven W. Lockley and Charles A. Czeisler. High sensitivity of the human circadian melatonin rhythm to resetting by short wavelength light. 2013.

[32] Wolfgang Jagla Herbert Jagle Lindsay T. Sharpe, Andrew Stockman. A luminous efficiency function, v*(lambda), for daylight adaptation. 2005.

[33] Photopic Vision. Available at `https://en.wikipedia.org/wiki/Photopic_vision`, accessed on 04/9/2015.

[34] John Roberts Zhichun Jenny Ying Nicholas Bloom, James Liang. Does working from home work? evidence from a chinese experiment. The Quarterly Journal of Economics, 2014.

[35] David A. Harrison Ravi S. Gajendran. The good, the bad, and the unknown about telecommuting: Meta- analysis of psychological mediators and individual consequences. Vol. 92, No. 6, 1524-1541, 2007.

[36] Iso/iec 18092:2004 – information technology – telecommunications and information exchange between systems – near field communication – interface and protocol (nfcip-1), 2004.

[37] Alice J. O'Toole Nicholas Furl, P. Jonathon Phillips. Face recognition algorithms and the other-race effect: computational mechanisms for a developmental contact hypothesis. 2013.

[38] Standard Group MAC Addresses. Available at `http://standards.ieee.org/develop/regauth/tut/macgrp.pdf`, accessed on 25/8/2015.

[39] James T. Simpson Ki-Yoon Kim Kwan-Sik Na, Xiaotong Li. Uncertainty profile and software project performance: A cross-national comparison. 2002.

[40] Xiaotong Li Tushar Singh Ki-Yoon Kim Kwan-Sik Na, James T. Simpson. Software development risk and project performance measurement: Evidence in korea. 2006.

[41] The Impact of Agile Quantified. Technical report, Rally Software, 2013.

[42] Diomidis D. Spinellis. Software Quality. Available at `http://www.spinellis.gr/codequality/intro.html`, accessed on 25/8/2015.

[43] What is a fair productivity measurement technique for programmers? Available at `http://stackoverflow.com/questions/324399/what-is-a-fair-productivity-measurement-technique-for-programmers`, accessed on 25/8/2015.

[44] How do I measure employee (software developer) performance based on bugs created? Available at `http://pm.stackexchange.com/questions/5289/how-do-i-measure-employee-software-developer-performance-based-on-bugs-created`, accessed on 25/8/2015.

[45] Johnmarshall Reeve Alice M. Isen. The influence of positive affect on intrinsic and extrinsic motivation: Facilitating enjoyment of play, responsible work behavior, and self-control. 2005.

[46] Eugenio Proto Andrew J. Oswald and Daniel Sgroi. Happiness and productivity. 2014.

[47] Psychology Of Happiness. Available at `http://www.scholarpedia.org/article/Psychology_of_happiness`, accessed on 25/8/2015.

[48] Mihaly Csikszentmihalyi Joel M. Hektner, Jennifer A. Schmidt. 2007.

[49] Martin Fowler. Microservices. Available at `http://martinfowler.com/articles/microservices.html`, accessed on 25/8/2015.

[50] Microservices: Decomposing Applications for Deployability and Scalability. Available at `http://www.infoq.com/articles/microservices-intro`, accessed on 25/8/2015.

[51] Pattern: Microservices Architecture. Available at `http://microservices.io/patterns/microservices.html`, accessed on 25/8/2015.

[52] Micro Service Architecture. Available at `https://yobriefca.se/blog/2013/04/29/micro-service-architecture/`, accessed on 25/8/2015.

[53] Microservices and SOA. Available at http://www.oracle.com/technetwork/issue-archive/2015/15-mar/o25architect-2458702.html, accessed on 25/8/2015.

[54] Fundamental Design Terminology and Concepts. Available at http://serviceorientation.com/whatissoa/fundamental_design_terminology_and_concepts, accessed on 25/8/2015.

[55] Service Orientation. Available at https://en.wikipedia.org/wiki/Service-orientation, accessed on 25/8/2015.

[56] SOA vs Microservices. Available at https://blog.hedges.net/2014/03/31/soa-vs-microservices/, accessed on 25/8/2015.

[57] Difference between Microservices Architecture and SOA. Available at http://stackoverflow.com/questions/25501098/difference-between-microservices-architecture-and-soa, accessed on 25/8/2015.

[58] Mark Little. Microservices and SOA. Available at http://www.infoq.com/news/2014/03/microservices-soa, accessed on 25/8/2015.

[59] José Ribeiro. Integration and optimization of energy data analysis systems. Master's thesis, 2014.

[60] Mark Little. MonolithFirst. Available at http://martinfowler.com/bliki/MonolithFirst.html, accessed on 25/8/2015.

[61] Internet of things (iot): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, Volume 29 Issue 7, September, 2013, 2013.

[62] A Reference Architecture for the Internet of Things. Technical Report 0.0.8, WSO2, 2014.

[63] Internet of things (iot): A vision, architectural elements, and future directions. 2012.

[64] IoT Architecture. Technical report, European Commision, 2014.

[65] D. Thaler H. Tschofenig, J. Arkko and D. McPherson. Architectural Considerations in Smart Object Networking. March 2015.

[66] TSL2560, TSL2561 LIGHT-TO-DIGITAL CONVERTER - Datasheet. Available at https://www.adafruit.com/datasheets/TSL256x.pdf, accessed on 27/8/2015.

[67] Nmap. Available at https://en.wikipedia.org/wiki/Nmap, accessed on 27/8/2015.

[68] Crunchbase - Keen IO. Available at https://www.crunchbase.com/organization/keen, accessed on 25/8/2015.